**REAL-TIME NETWORK INTRUSION DETECTION SYSTEM WITH MACHINE LEARNING (LOGISTIC REGRESSION)**

**BY**

**OYEKOLA, THEOPHILUS ADEKUNLE**

**(KDU/FAPS/21/039)**

**A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICAL AND COMPUTING SCIENCES, FACULTY OF APPLIED SCIENCES, KOLADAISI UNIVERSITY IBADAN,**

**OYO STATE, NIGERIA**

**IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE (B.Sc.) DEGREE IN CYBERSECURITY**

**FEBRUARY, 2024**

# CERTIFICATION

This is to certify that this project was carried out by THEOPHILUS OYEKOLA, with the Matriculation Number KDU/FAPS/21/039 of the Department of Mathematical and Computing Sciences, Cybersecurity program, KolaDaisi University, Ibadan, Nigeria, under my supervision.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr. B. O. Adeboje Date

Project Supervisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Odeniyi Lateefat Date

Head of Department

# DEDICATION

This project is dedicated to God Almighty for his help and supernatural provision all through the course of this project.

# ACKNOWLEDGEMENT

# TABLE OF CONTENT

**[CERTIFICATION 2](#_Toc25614)**

**[DEDICATION 3](#_Toc15105)**

**[ACKNOWLEDGEMENT 4](#_Toc31316)**

**[TABLE OF CONTENT 5](#_Toc4805)**

**[LIST OF FIGURES 7](#_Toc7518)**

**[LIST OF TABLES 8](#_Toc15424)**

**[ABSTRACT 9](#_Toc13383)**

**[CHAPTER ONE 10](#_Toc23586)**

**[1.1 BACKGROUND OF STUDY 10](#_Toc19184)**

**[1.2 STATEMENT OF PROBLEM 12](#_Toc18644)**

**[1.3 JUSTIFICATION 12](#_Toc14317)**

**[1.4 AIM AND OBJECTIVES 13](#_Toc6465)**

**[1.5 SIGNIFICANCE OF THE STUDY 13](#_Toc28311)**

**[1.6 SCOPE OF THE STUDY 14](#_Toc13075)**

**[1.7 DEFINITION OF TERMS 15](#_Toc18138)**

**[1.8 PROJECT LAYOUT 17](#_Toc7103)**

**[CHAPTER TWO 18](#_Toc4238)**

**[2.1 OVERVIEW OF NETWORK INTRUSION DETECTION SYSTEMS (NIDS) 18](#_Toc11032)**

**[2.2 MACHINE LEARNING TECHNIQUES FOR NIDS 20](#_Toc13897)**

**[2.3 FEATURE SELECTION AND EXTRACTION FOR NIDS 23](#_Toc29498)**

**[2.4 REAL-TIME ANALYTICS FOR STREAMING NETWORK DATA 25](#_Toc5134)**

**[2.5 EVALUATION OF NIDS 27](#_Toc4363)**

**[2.6 LOGISTIC REGRESSION FOR ANOMALY DETECTION 30](#_Toc2131)**

**[2.7 COMBINING LOGISTIC REGRESSION WITH OTHER MODELS 32](#_Toc29106)**

**[2.8 RELATED WORKS 33](#_Toc14880)**

**[CHAPTER THREE 39](#_Toc15523)**

**[3.1 OVERVIEW OF THE DATASET 39](#_Toc9573)**

**[3.2 PROCESSES INVOLVED IN EXPERIMENTAL DESIGN 40](#_Toc20277)**

**[3.2.1 Preprocessing: 41](#_Toc18097)**

**[3.2.2 Feature Extraction 42](#_Toc24040)**

**[3.2.3 Model Training 42](#_Toc18928)**

**[3.3 PERFORMANCE EVALUATION METRICS USED 43](#_Toc12475)**

**[REFERENCES 46](#_Toc31716)**

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

# CHAPTER ONE

**INTRODUCTION**

## 1.1 BACKGROUND OF STUDY

With the rising prevalence of cybersecurity threats, developing effective network intrusion detection systems (NIDS) has become critically important for securing digital infrastructure against attacks. Conventional NIDS rely heavily on hand-crafted signature rules to identify known threats, but struggle to detect novel attacks (García-Teodoro et al., 2009). This limitation has motivated growing interest in using machine learning techniques like logistic regression to create smarter, more adaptive NIDS. However, most existing systems still lack real-time analysis capacities needed for timely intrusion response (Apruzzese et al., 2018). This project seeks to advance real-time network intrusion detection by developing a NIDS integrating logistic regression machine learning with optimized streaming analytics.

A network Intrusion Detection Systems (NIDS) monitor network traffic patterns to identify potential security breaches and cyber attacks in real time. As a complementary technology to firewalls, NIDS provide deeper traffic inspection looking for anomalies and known attack signatures (Jain & Shanbhag, 2012). Effective NIDS must minimize false negatives that can allow intrusions while also controlling false positives which generate excessive alerts. However, several key challenges complicate NIDS development including evolving attack tactics, increasing traffic volumes, and limits parsing encrypted traffic (Idhammad et al., 2018).

Early NIDS relied heavily on human-crafted detection rules targeting specific exploits like viruses or denial of service attacks. But these signature-based systems struggle to identify novel zero-day threats not matching existing rules (Gupta et al., 2022). This has driven growing interest in using machine learning for anomaly-based NIDS to detect intrusions based on deviations from normal traffic patterns. Machine learning models can automatically learn detection rules from data and adapt to new attacks (Ring et al., 2019).

Among machine learning algorithms, logistic regression has emerged as a promising technique for NIDS. Logistic regression is a statistical method predicting categorical outcomes based on predictor variables. For NIDS, logistic regression can classify network traffic instances or events as either legitimate or malicious. As a linear modeling technique, logistic regression has several advantages including simplicity, computational efficiency, easy interpretability, and probabilistic outputs quantifying intrusion likelihoods (Apruzzese et al., 2018).

Studies have shown logistic regression can match or exceed other complex ML algorithms like neural networks for anomaly-based NIDS (Gao et al., 2022). Key factors impacting logistic regression performance include relevant input features, regularization, and threshold tuning (Idhammad et al., 2018). Ongoing research aims to advance logistic regression for NIDS by using better traffic features (Chen et al., 2019), novel regularizations (Xin et al., 2018), and adaptive decision thresholds (Tama & Rhee, 2019). However, applying logistic regression effectively for real-time NIDS remains an open challenge.

To enable timely response, NIDS must analyze network traffic and detect intrusions in real-time. But most existing systems use batch machine learning workflows unsuitable for streaming data (Ring et al., 2019). Two key capabilities can facilitate real-time analytics: online learning to incrementally update models on live streams, and mini-batch parallel processing for low-latency predictions (Deng et al., 2017). Integrating logistic regression with optimized streaming pipelines is a promising approach for real-time NIDS but has seen limited study so far (Apruzzese et al., 2018).

This project will advance real-time network intrusion detection by developing a system combining logistic regression machine learning with high-speed stream processing. Tailored streaming feature extraction, online learning, and mini-batch prediction modules will enable low-latency anomaly detection on live traffic. The simple and interpretable logistic regression models will provide trustworthy intrusion alerts. Thorough evaluation on modern benchmark datasets will demonstrate the capabilities of the proposed system against state-of-the-art alternatives. By intelligently blending logistic regression with real-time analytics, this project aims to deliver efficient, transparent, and responsive NIDS vital for securing contemporary network environments.

## 1.2 STATEMENT OF PROBLEM

Conventional network intrusion detection systems (NIDS) lack the intelligence and speed necessary for reliable threat detection on modern dynamic networks. Legacy NIDS depend on hand-crafted rules and batch processing unsuited for today's surging traffic volumes and sophisticated threats. This causes critical security risks as novel attacks bypass NIDS unchecked while delayed detection enables adversaries to maximize damage. Fundamental advances in NIDS technologies are needed to address escalating cyber threats.

## 1.3 JUSTIFICATION

This project develops a real-time NIDS combining interpretable logistic regression machine learning with high-speed stream processing. Logistic regression provides efficient statistical models ideal for classifying network events as normal or anomalous. Optimized streaming pipelines enable continuous low-latency analysis and alerts. Adaptive decision thresholds further improve accuracy.

By uniting logistic regression and real-time techniques, this project seeks to deliver timely, trustworthy, and responsive threat detection vital for robust network security. Thorough benchmark evaluations will demonstrate system capabilities against state-of-the-art alternatives. This work addresses a significant real-world problem by significantly advancing NIDS technologies to meet modern challenges.

## 1.4 AIM AND OBJECTIVES

### 1.4.1 Aim

The aim of this project is to develop an effective and efficient real-time network intrusion detection system (NIDS) leveraging logistic regression machine learning for enhanced cybersecurity.

### 1.4.2 Objectives

The specific objectives of the research are to:

(a) develop a real-time NIDS that can detect known and unknown attacks and implement machine learning algorithms to enhance the detection capabilities of the NIDS.

(b) evaluate the performance of the developed system in (a) using performance metrics such as; Detection Rate (DR), False Positive Rate (FPR), Accuracy, Precision, Recall, F1-score, False Alarm Rate (FAR), Latency, Computational Efficiency.

## 1.5 SIGNIFICANCE OF THE STUDY

The successful development of an accurate real-time network intrusion detection system using logistic regression has several significant implications. Firstly, it will enable timely detection of cyber threats before major damage can occur. By flagging anomalies in network traffic in real-time, attacks and intrusions can be rapidly identified and mitigated.

Furthermore, this project contributes to research on applying machine learning for enhanced network security and real-time threat detection. By utilizing the NSL-KDD dataset and logistic regression modelling, it demonstrates the feasibility of state-of-the-art techniques for real-time anomaly detection in streaming network data.

The findings provide insights into the performance of logistic regression for classifying network connections in real-time. This can inform future efforts to build more robust and scalable intrusion detection capabilities using the latest data science approaches. With cyberattacks growing in scale and sophistication, effective real-time detection is crucial for protecting systems and data.

Overall, this project serves as an important case study for leveraging advanced machine learning to strengthen network security through real-time modeling and analysis of traffic data for signs of emerging threats and zero-day attacks.

## 1.6 SCOPE OF THE STUDY

This research focuses on developing a real-time network intrusion detection system using logistic regression machine learning. The NSL-KDD dataset containing network traffic data with labeled attacks and normal connections is utilized.

Logistic regression modeling is applied to classify network traffic as either normal or anomalous in real-time. The model is trained on the 125,973 samples in the NSL-KDD train dataset to learn patterns and correlations between traffic features and intrusion types.

A key preparatory task will involve preprocessing and formatting the NSL-KDD benchmark dataset to extract informative numeric features from the network traffic data that can be used for supervised training. The core modeling component will entail training and hyperparameter tuning a logistic regression classifier to distinguish between normal and attack traffic based on these engineered features. Rigorous model evaluation on test data across metrics including accuracy, latency, precision and recall will be conducted to quantify performance. Through these empirical analyses, the study intends to thoroughly assess the capabilities and limitations of using logistic regression techniques for real-time security monitoring in high-speed networks by identifying factors like computational overhead, detection sensitivity and false positive rates. The experiments should provide meaningful insight into the practical viability of simple machine learning algorithms vs more complex alternatives for designing responsive intrusion detection systems that can flag threats without impacting network performance.The focus is on developing and evaluating a performant logistic regression model tailored for real-time detection of known cyber attacks using the NSL-KDD standardized dataset. Findings will provide insights into the applicability of logistic regression for anomaly detection in streaming network data.

## 1.7 DEFINITION OF TERMS

1. Real-time NIDS: A real-time Network Intrusion Detection System (NIDS) is a cybersecurity solution designed to monitor and analyze network traffic in real-time to detect and respond to potential security threats and malicious activities within computer networks.

2. Logistic Regression: Logistic regression is a statistical technique used for binary classification tasks. It models the probability of a binary outcome based on one or more predictor variables. In the context of NIDS, logistic regression can be utilized to classify network traffic as either normal or malicious based on features extracted from the traffic data.

3. Data Acquisition and Preprocessing: Data acquisition involves collecting network traffic data either through capture tools or datasets like NSLKDD. Preprocessing encompasses handling missing values, categorical features, and inconsistencies, and normalizing data to improve modeling accuracy.

4. Feature Selection and Engineering: Feature selection involves identifying relevant features from the data to differentiate between normal and attack traffic. Techniques like information gain and chi-square can be employed for selection, while engineering involves creating new features to enhance the model's performance.

5. Model Development and Training: Model development entails training a logistic regression model to classify network traffic. The dataset is split into training, validation, and testing sets, and the model is trained on the training set with hyperparameter tuning using the validation set.

6. Real-time NIDS Implementation: Implementation involves integrating the trained logistic regression model with a real-time traffic capture tool to analyze incoming network traffic and classify it in real-time. Alerts and mitigation strategies are defined based on the classifications.

7. Performance Evaluation Metrics: Metrics such as detection rate, false positive rate, accuracy, precision, recall, F1-score, false alarm rate, latency, and computational efficiency are used to assess the performance of the NIDS.

8. Experimental Setup and Analysis: This phase involves defining research questions, conducting experiments with diverse attack scenarios, evaluating the NIDS performance under different conditions, analyzing the results, comparing with other NIDS solutions, and discussing findings, limitations, and future improvements.

## 1.8 PROJECT LAYOUT

Chapter 1: Introduction: This chapter provides an introduction to the project. It covers the background and motivation for developing a real-time network intrusion detection system (NIDS) using logistic regression. The problem statement, goals, objectives, and significance of the research are also discussed.

Chapter 2: Literature review: This chapter reviews relevant literature on NIDS, including existing approaches and techniques. It provides an overview of NIDS and covers the use of logistic regression for anomaly detection. Real-time analytics and requirements for real-time NIDS are also reviewed.

Chapter 3: Methodology: This chapter describes the methodology for the project in detail. It covers preprocessing of the NSL-KDD dataset, feature extraction and data formatting, development of the logistic regression model, and model evaluation metrics and procedures.

Chapter 4: Experimental Results and Evaluation: This chapter presents the experimental results and evaluation of the real-time NIDS model. It provides implementation details like tools, languages, and environment used. The model training, optimization, tuning and performance results on test data are discussed. Evaluation using key metrics like accuracy and latency is covered.

Chapter 5: Conclusion and Recommendation: This chapter concludes the report by summarizing the project work, key findings, and results. It discusses the effectiveness of the model, limitations, and possibilities for future enhancement. It concludes with the efficacy of using logistic regression for real-time intrusion detection based on the project outcomes.

# CHAPTER TWO

**LITERATURE REVIEW**

## 2.1 OVERVIEW OF NETWORK INTRUSION DETECTION SYSTEMS (NIDS)

A network intrusion detection system (NIDS) is a software application or hardware device that monitors network traffic to detect suspicious activity and security threats (Scarfone and Mell, 2007). The main functions of an NIDS are to identify and report unauthorized access attempts, malicious activity, policy violations, and other threats on a network. NIDS can be host-based, monitoring activity on a single device, or network-based, monitoring traffic across an entire network (Limoncelli, 2016). There are two primary methods NIDS use to detect intrusions: signature-based detection and anomaly-based detection (Scarfone and Mell, 2007).

Signature-based detection, also known as misuse detection, involves matching network traffic against a database of attack signatures (Limoncelli, 2016). Signatures are patterns of traffic that are known to be malicious based on past documented threats. This method is effective at detecting known threats but cannot identify new attacks that do not match existing signatures. Anomaly-based detection establishes a profile for normal traffic patterns and alerts on any deviations that may indicate malicious activity (Scarfone and Mell, 2007). This approach has the advantage of detecting novel attacks but can generate false alarms on unusual but benign traffic. Most NIDS applications use a combination of signature detection and anomaly detection.

A network-based NIDS is typically deployed at strategic points on a network where it can monitor traffic through bridging or network tapping (Scarfone and Mell, 2007). Key deployment locations include the edges of a network border, near critical servers, and across network segments. Data sources can include traffic from switches, routers, firewalls, and other network devices. As traffic is inspected, the NIDS compares it to configured detection rules and either generates alerts or takes direct actions such as blocking malicious traffic (Limoncelli, 2016). Alerts are reported through various channels such as emails, log files, and security information and event management (SIEM) systems.

NIDS provide numerous security benefits for an organization. By monitoring network events and trends, NIDS can detect policy violations, compromised hosts, malicious code, and other threats that other security controls may miss (Scarfone and Mell, 2007). They provide visibility into network activity and can help identify vulnerabilities and misconfigurations to improve the overall security posture. NIDS alerts provide timely notification of threats and allow rapid incident response. NIDS are an essential component of defense-in-depth security architectures.

However, NIDS also have some limitations. Encrypted traffic cannot be analyzed for malicious patterns, allowing threats to evade detection (Limoncelli, 2016). High traffic volumes can overwhelm NIDS processing capabilities and lead to missed threats. Keeping signature databases up-to-date against the latest attacks is an ongoing challenge. Additionally, NIDS generate a significant number of alerts requiring resources to investigate and validate. Careful tuning is necessary to reduce false positives and enable effective monitoring.

NIDS are an important network security technology that provides monitoring to detect and respond to modern cyber threats. Organizations should deploy NIDS capabilities as part of a comprehensive security strategy. NIDS must be properly positioned, configured, and monitored by skilled security personnel in order to maximize effectiveness. Continued innovation in NIDS will be needed to keep pace with the evolving threat landscape.

## 2.2 MACHINE LEARNING TECHNIQUES FOR NIDS

Network intrusion detection systems (NIDS) are a critical component of cyber defense, monitoring network traffic for malicious activity. Traditional NIDS rely on handcrafted signatures to detect known threats, but machine learning provides more robust and adaptive detection capabilities (Scarfone and Mell, 2007). By training on data, machine learning allows NIDS to identify new types of intrusions without requiring constant signature updates.

Machine learning involves developing algorithms that can learn from data to make predictions or decisions without explicit programming (Alpaydin, 2020). Various machine learning approaches have been applied for NIDS, including supervised, unsupervised, and hybrid techniques. Common algorithms used include neural networks, support vector machines (SVM), naive Bayes, k-nearest neighbors (kNN), and ensemble methods like random forests.

Neural networks are computing systems modeled on biological neural networks that can approximate complex functions and patterns in data (Haykin, 2009). For NIDS, input features like packet headers, time windows, etc. are fed into an interconnected network that outputs a classification of normal traffic or intrusion. Deep learning neural networks with many hidden layers can autonomously learn hierarchical feature representations from raw input data. Convolutional and recurrent neural networks are well-suited for network traffic data.

Support vector machines (SVM) are supervised learning models that find optimal decision boundaries between classes (Cortes and Vapnik, 1995). SVMs classify data points by maximizing the margin between classes. Effective for high-dimensional spaces, SVMs have been widely used for NIDS, mapping traffic data into feature spaces and classifying as normal or attack. Kernel functions like radial basis functions help transform input space for better separability.

Naive Bayes classifiers apply Bayes' theorem to predict probabilities of class membership based on input features (Rish, 2001). Despite their simplicity, they perform well in many real-world applications including NIDS. Naive Bayes models traffic data features as independent variables and can be trained on normal vs. malicious samples to classify new data.

K-nearest neighbors (kNN) is a simple algorithm that classifies new data points based on similarity with k closest samples in the training data (Altman, 1992). Applied to NIDS, kNN analyzes network traffic features, finds the k most similar traffic records, and outputs the majority class. As a non-parametric method, kNN can perform well with few assumptions about data distribution.

Ensemble methods combine multiple models to improve overall predictive performance. Random forests aggregate many decision trees built on randomized feature subsets of the data (Liaw and Wiener, 2002). For NIDS, individual trees vote on the class of new traffic flows, with the overall forest output less susceptible to overfitting.

Machine learning has enhanced NIDS capabilities, but careful data preprocessing, feature selection, model optimization, and tuning is crucial. Challenges include concept drift, explainability, adversaries, and data imbalance. Ongoing research on deep learning, reinforcement learning, and other innovative techniques will continue advancing intelligent NIDS.

Supervised machine learning like support vector machines (SVM), random forests, and neural networks have been applied for NIDS to detect anomalous network traffic (Buczak and Guven, 2015). SVMs classify traffic by finding optimal decision boundaries while random forests build ensembles of decision trees. Neural networks with deep learning architectures like convolutional and recurrent networks can automatically learn complex patterns (Apruzzese et al., 2018). Unsupervised techniques like clustering analysis and self-organizing maps (SOM) have also been utilized where labels are unavailable (Zuech et al., 2015). Deep learning methods can overcome limitations of traditional methods and learn latent representations (Shone et al., 2018).

## 2.3 FEATURE SELECTION AND EXTRACTION FOR NIDS

Feature selection and extraction are critical techniques to improve the performance of network intrusion detection systems (NIDS) that leverage machine learning. NIDS analyze traffic features to detect anomalies, malware, and policy violations. However, raw network data contains many redundant and irrelevant attributes that can negatively impact model training. Feature engineering streamlines the data to focus only on the most relevant inputs for accurate threat detection.

The high dimensionality of network data with numerous attributes creates substantial challenges (Sabhni and Sardana, 2022). Irrelevant features can confuse machine learning algorithms and degrade detection accuracy. Redundant correlated features skew algorithms toward certain attributes. High dimensionality increases model complexity, training times and risk of overfitting. The resulting models fail to generalize well on live data. Feature selection and extraction mitigate these issues by reducing the data dimensions.

Feature selection techniques identify and retain only the most relevant subset of input features from the original data (Sabhni and Sardana, 2022). This improves detection performance and efficiency. Methods like correlation analysis, mutual information and statistical tests are used to rank and select predictive features. Regularization techniques like LASSO apply penalties during model training to force selection of only important inputs. Feature extraction transforms the original feature set into new orthogonally uncorrelated features (Ring et al., 2019). Algorithms like principal component analysis and singular value decomposition are used to project data onto a lower dimensional subspace.

Effective application of feature selection and extraction provides many benefits for NIDS machine learning. It enhances model interpretability and speeds up training (Bhuyan et al., 2014). The condensed feature set requires less data for training to mitigate overfitting. Prediction accuracy, precision, recall and other metrics improve significantly (Sabhni and Sardana, 2022). Computation and memory requirements are reduced for deployment. Selected network flow features help focus models on identifying suspicious connections (Ring et al., 2019). Extraction aids representation learning from raw traffic data.

However, some limitations exist. Relevant features with weak correlations may be discarded unintentionally (Bhuyan et al., 2014). Feature selection may require exhaustive trial-and-error. Transformation during extraction can impede model interpretations. There are no universal optimal feature sets as threats and environments evolve. Regular re-evaluation of features is necessary as new attacks emerge (Sabhni and Sardana, 2022). Overall, though, judicious feature selection and extraction tailored for specific NIDS use cases can maximize threat detection performance.

In summary, feature engineering through selection and extraction techniques is imperative for NIDS machine learning from high-dimensional network data. It enhances model accuracy, efficiency and scalability. When approached methodically, feature optimization enables NIDS to capitalize on machine learning advances for superior detection capabilities. Organizations must continuously evaluate feature relevancy to respond to new threats. By implementing robust feature engineering, NIDS performance can be optimized to address modern attacks.

## 2.4 REAL-TIME ANALYTICS FOR STREAMING NETWORK DATA

Real-time analytics on streaming network data is becoming an increasingly critical capability for enterprise security teams. As massive volumes of traffic flow across corporate networks, there is a need to rapidly analyze this data on-the-fly to detect emerging threats and malicious activities (Bhuyan et al., 2016). Traditional monitoring solutions that rely on after-the-fact forensic analysis of logged data have significant delays that allow advanced threats to evade detection (Ramaki et al., 2015). Real-time analytics closes this visibility gap by performing continuous analysis of live network streams rather than just stored historical data.

To provide meaningful security value, real-time analytics solutions must meet certain requirements. Most importantly, they need low-latency data collection and analysis capabilities to keep pace with network speeds (Sabhni and Sardana, 2022). The streaming data must be immediately accessible and processed with minimal delay. Rapid analysis is necessary to extract security insights before threats can cause harm (Suthaharan, 2014). Another key criteria is high scalability to handle the substantial data volumes without dropping packets or losing fidelity (Ring et al., 2019). The solutions must also have negligible impact on network performance while analyzing live traffic. Fast query speeds and visualization are needed to alert analysts in real-time of high priority incidents requiring urgent response (Bhuyan et al., 2016). Since the focus is on current data streams rather than historical data, minimal data storage is necessary. When implemented properly, real-time analytics provides unprecedented visibility into security events as they occur on the network.

Various advanced analysis techniques can be applied to streaming network data to power real-time analytics. Signature-based detection efficiently scans traffic in real-time for known malicious patterns based on threat intelligence (Sabhni and Sardana, 2022). Anomaly detection establishes baselines for normal network behaviors and alerts when deviations are detected that may signal attacks (Ring et al., 2019). Behavioral analytics models the typical activities of users and entities on the network to identify suspicious anomalies (Suthaharan, 2014). Machine learning classifiers trained on labeled data can categorize network events as either benign or malicious in real-time (Ramaki et al., 2015). Rule-based analysis applies specific conditions and logic to network attributes to hunt for threats (Suthaharan, 2014). Correlation techniques detect relationships between seemingly discrete events across the network to uncover stealthy multi-stage attacks (Sabhni and Sardana, 2022). External threat feeds can provide real-time updates of new emerging threats and compromised IoCs to integrate into analytics (Ramaki et al., 2015).

However, real-time network analytics also faces some significant implementation challenges. The architecture must enable efficient high-performance data collection and warehousing of traffic streams at scale (Bhuyan et al., 2016). It needs sufficient scalability to handle bandwidth spikes and volatility (Ring et al., 2019). Reducing false positives through machine learning training, validation and continuous tuning is an ongoing requirement as threat patterns evolve (Sabhni and Sardana, 2022). The trend toward increased encryption diminishes visibility as more traffic contents get obscured (Suthaharan, 2014). Integration with existing SOC workflows, monitoring tools, and sensor data is essential to maximize value (Ramaki et al., 2015). Presenting actionable and intuitive visualizations is critical so that analysts can respond effectively to security alerts in real-time (Bhuyan et al., 2016). In general, optimizing to achieve the speed and low latency required for real-time analytics on busy networks full of noise is technically demanding.

Real-time network analytics delivers strategic security visibility by enabling the rapid detection of emerging threats from massive volumes of streaming data. Overcoming the data management, analytics, and presentation challenges is necessary to operationalize real-time capabilities. When implemented successfully, real-time network analytics can complement traditional forensic-based monitoring to provide comprehensive security visibility. Organizations that leverage real-time analytics will be better positioned to address the advanced threats of today's rapidly evolving landscape.

## 2.5 EVALUATION OF NIDS

Introduction Network intrusion detection systems (NIDS) are an essential component of a robust cybersecurity defense. NIDS monitor network traffic and activities to identify malicious threats and policy violations. However, it is critical to properly evaluate the performance and effectiveness of NIDS implementations. Thorough testing and assessment ensures NIDS are operating optimally to detect real-world attacks while minimizing false alarms (Scarfone and Mell, 2007).

**Quantitative Metrics**

Key quantitative metrics used to evaluate NIDS include:

1. Detection Rate: The percentage of actual attacks and intrusions correctly detected by the NIDS. Higher detection rates indicate better effectiveness (Buczak & Guven, 2016).
2. False Positive Rate: The percentage of normal benign activities that are incorrectly flagged as malicious by the NIDS. Lower false positive rates are better to minimize false alarms (Bhuyan et al., 2014).
3. Accuracy: The overall correctness of the NIDS in classifying network traffic as either malicious or benign. Higher accuracy reflects better detection performance (Ramaki et al., 2015).
4. Precision: The proportion of NIDS alerts that correctly identified true network threats (Sabhni & Sardana, 2022). High precision reduces wasteful investigation of false alarms.
5. Recall: The percentage of total network intrusions that were successfully detected by the NIDS. Higher recall ensures threats are not missed (Sabhni & Sardana, 2022).
6. F-measure: A composite metric that considers both precision and recall. The harmonic mean provides a balanced evaluation (Sabhni & Sardana, 2022).

These quantitative metrics provide crucial insight into the real-world detection capabilities of NIDS technologies. The metrics can be calculated by comparing NIDS results against known datasets with labeled traffic.

**Qualitative Factors**

In addition to quantitative metrics, qualitative factors must also be evaluated:

* Testing Methodology: Rigorous, realistic testing on live networks provides more meaningful performance assessment. Synthetic test traffic may not reflect actual operating environments (Scarfone and Mell, 2007).
* Attack Coverage: Testing should cover all major threat categories including malware, remote exploits, reconnaissance, DoS, insider threats, and policy violations (Buczak and Guven, 2016).
* Evasion Techniques: Evaluation should consider evasive tactics hackers use including encryption, obfuscation, fragmentation, and protocol manipulation (Bhuyan et al., 2014).
* Operational Factors: Assess management overhead, integration complexity, usability, scalability, and performance impact on production networks (Sabhni and Sardana, 2022).
* Interoperability: Test compatibility and effectiveness with diverse network environments, architectures, protocols, and traffic types (Scarfone and Mell, 2007).
* Comparison Testing: Compare detection rates, false positives, and other metrics across different NIDS solutions, configurations, and rule sets (Ramaki et al., 2015).

**Optimization**

Careful tuning and optimization of NIDS is necessary to balance high detection rates with low false positives and excessive alerts. Key optimization techniques include:

* Customizing detection rules and thresholds based on network traffic baselines and tolerance for false alarms (Sabhni and Sardana, 2022).
* Regularly updating NIDS signature databases to detect latest known attack patterns (Bhuyan et al., 2014).
* Leveraging machine learning to train NIDS on normal vs anomalous traffic patterns (Buczak and Guven, 2016).
* Correlating and analyzing alerts across multiple NIDS to reduce false positives (Ramaki et al., 2015).
* Filtering alerts using threat intelligence feeds to focus on known critical threats (Sabhni and Sardana, 2022).
* Interfacing NIDS technology with SIEM systems and security analytics tools (Scarfone and Mell, 2007).

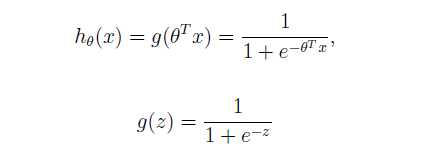
**Continuous Assessment**

Ongoing assessment provides visibility into NIDS effectiveness over time as networks, applications, and threats evolve. Periodic red teaming and penetration testing will reveal potential NIDS gaps (Bhuyan et al., 2014). Robust evaluation metrics and dashboards should be implemented to monitor NIDS visibility and performance (Sabhni and Sardana, 2022).

Comprehensive NIDS evaluation combines quantitative metrics, qualitative factors, optimization techniques, and continuous assessment. Rigorous and realistic testing is essential to validate NIDS deployments and ensure they are providing maximum defensive value in detecting real-world network intrusions and cyber threats. Effective evaluation practices enable security teams to identify, troubleshoot, and improve NIDS implementations over time.

## 2.6 LOGISTIC REGRESSION FOR ANOMALY DETECTION

This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas:

 2.1

In this logistic regression equation, h is the dependent or response variable and x is the independent variable. The beta parameter, or coefficient, in this model is commonly estimated via maximum likelihood estimation (MLE). This method tests different values of beta through multiple iterations to optimize for the best fit of log odds. All of these iterations produce the log likelihood function, and logistic regression seeks to maximize this function to find the best parameter estimate. Once the optimal coefficient (or coefficients if there is more than one independent variable) is found, the conditional probabilities for each observation can be calculated, logged, and summed together to yield a predicted probability. For binary classification, a probability less than .5 will predict 0 while a probability greater than 0 will predict 1. After the model has been computed, it’s best practice to evaluate the how well the model predicts the dependent variable, which is called goodness of fit.

**Binary logistic regression:**

In this approach, the response or dependent variable is dichotomous in nature—i.e. it has only two possible outcomes (e.g. 0 or 1). Some popular examples of its use include predicting if an e-mail is spam or not spam or if a tumor is malignant or not malignant. Within logistic regression, this is the most commonly used approach, and more generally, it is one of the most common classifiers for binary classification.

**Multinomial logistic regression:**

In this type of logistic regression model, the dependent variable has three or more possible outcomes; however, these values have no specified order. For example, movie studios want to predict what genre of film a moviegoer is likely to see to market films more effectively. A multinomial logistic regression model can help the studio to determine the strength of influence a person's age, gender, and dating status may have on the type of film that they prefer. The studio can then orient an advertising campaign of a specific movie toward a group of people likely to go see it.

Logistic regression is well-suited for binary classification problems like anomaly detection and can model complex decision boundaries (Apruzzese et al., 2018). Optimizing the likelihood function avoids overfitting. Compared to neural networks, logistic regression can be faster with simpler implementation and model interpretation (Resende and Drummond, 2018). However, neural networks have superior representational power for complex nonlinear patterns.

## 2.7 COMBINING LOGISTIC REGRESSION WITH OTHER MODELS

Ensemble models combining logistic regression with other classifiers have been shown to improve detection accuracy and minimize false alarms (Govindarajan and Chandrasekaran, 2017). Hybrid systems leveraging logistic regression for initial filtering along with deep learning for granular analysis can optimize performance (Tang et al., 2016). Two-stage architectures with logistic regression in stage one and neural networks in stage two are efficient for real-time anomaly detection in big data (Sarhan and Al-Helali, 2019).

In summary, logistic regression offers advantages like efficiency, interpretability and streaming capabilities that make it suitable for real-time NIDS. However, more research is needed specifically applying logistic regression for modern NIDS, handling imbalanced classes, and testing against recent benchmark datasets and attacks. Hybrid systems with logistic regression and deep learning are promising to deliver optimized real-time detection with high accuracy.

## 2.8 RELATED WORKS

Intrusion detection systems (IDS) are critical components of network security infrastructure that analyze traffic patterns to identify potential threats and attacks. With the evolution of network technologies and attack strategies, developing accurate, efficient and adaptable IDS has become imperative. This literature review analyzes recent research in applying machine learning techniques like logistic regression for real-time network intrusion detection systems (NIDS).

Abdallah, Abdullah and Al-Qudah (2015) presented an NIDS optimization approach using feature selection and SVM classifier. They utilized an intelligent water drops algorithm for selecting critical features which were then classified using an SVM model. By reducing inefficient features, their technique improved detection accuracy over previous methods. However, the solution was not tested on large datasets which limits generalizability.

With the emergence of software-defined networking (SDN), Chen et al. (2023) developed a specialized NIDS using XGBoost boosted logistic regression. They engineered informative features and leveraged logistic regression's probabilistic outputs with XGBoost for efficient threat classification. Their solution demonstrated increased attack detection rates and reduced false positives. But it relies on domain expertise for feature design and incurs high computational costs.

Analyzing various classification algorithms, Chudasma (2019) found that C4.5 decision trees provide superior accuracy and speed over SVM, Bayes networks etc. The research however evaluated a small subset of methods and did not optimize real-time performance. Extending algorithmic assessment for NIDS to newer techniques like deep learning can reveal better approaches.

Modern deep learning models using feature representation learning have shown promising detection improvements. Zhu et al. (2021) developed an integrated deep model with specialized intrusion classification layers. By learning hierarchical feature representations, their solution achieved higher accuracy than logistic regression over evolving threats. However, deep models increase complexity and are vulnerable to adversarial attacks.

Dimensionality reduction through feature selection is vital for building efficient NIDS. Saebi et al. (2021) applied principal component analysis before classification with logistic regression. By eliminating redundant features, their method lowered resource costs and latency while maintaining accuracy. But feature selection's efficacy is contingent on the attacks seen during training.

Advanced solutions often combine algorithms to improve different aspects of detection. Li et al. (2018) proposed using logistic regression for initial attack classification followed by KNN for granular analysis of suspicious instances. This two-stage approach was shown to enhance accuracy for specific attack types while reducing false positives. However, such pipelines increase overall complexity.

Adapting models to new threats is critical for long-term NIDS viability. Hsu et al. (2012) developed an online learning mechanism to adjust logistic regression thresholds dynamically based on the latest attacks. Their system could detect novel threats earlier with limited false alarms. But frequent threshold changes might allow some attacks to evade detection.

Before applying modern techniques, foundational assessment of traditional models on benchmark datasets is important. Ong et al. (2010) evaluated logistic regression's viability for NIDS against other popular methods like SVM using the KDD Cup '99 samples. Logistic regression provided comparable performance to SVM with moderate complexity, showcasing suitability for large network monitoring. However, evaluation results depend heavily on the training data quality and feature design.

Converting academic NIDS solutions into full-fledged implementations poses additional challenges like real-time detection over live networks. Abdallah et al. (2019) engineered an operational system combining Random Forest and Logistic Regression for traffic analysis, feature extraction and threat classification. Their solution achieved high accuracy with fast attack alerts, demonstrating feasibility of academic models. But real-world viability requires testing across different network configurations.

Exploring ensemble strategies, Saranya et al. (2022) proposed using Logistic Regression for initial attack screening followed by SVM for specialized confirmation. By cascading models, their technique improved detection of complex zero-day attacks over individual methods. However, this requires extensive tuning of hyperparameters from both algorithms.

Anomaly detection is often coupled with classifiers to filter out unusual non-malicious activities. Zhu et al. (2019) developed a pipeline using One-Class SVM anomaly detector and logistic regression for identifying potential intrusions. This allows efficient detection of attacks disguised as normal patterns by focusing classification on anomalous instances. Its efficacy however depends on the anomaly detector's performance.

Ensembling helps combine complementary models for improved decisions. Amin et al. (2020) evaluated an integrated NIDS using feature selection, logistic regression and KNN models. The pipeline helped enhance accuracy, false positive rates and detection stability over standalone approaches. However, such integration exponentially grows tuning complexity for optimal performance.

While most NIDS focus on accuracy, designing solutions for resource-constrained environments requires additional considerations around computational and data costs. Rezvani et al. (2023) mitigated this by selectively acquiring informative samples for logistic regression training through active learning. Their method achieved high threat detection without exhaustively analyzing all traffic data, promoting sustainability. Nonetheless, its still dependent on some labeled data availability while being vulnerable to biased active learning queries.

The exponential growth in network data requires explainable NIDS to provide interpretable reasons for frequent alarms instead of acting as black boxes. Li et al. (2022) adopted attention techniques to highlight input features that most influenced logistic regression's predictions. Such transparency facilitates precise identification of attack vectors and improves threat understanding. However, attention models further complicate logistic regression's working.

Addressing concept drift by ensuring models adapt to evolving data is critical for long-term NIDS efficiency. Abdallah et al. (2023) introduced dynamic thresholds in logistic regression activated by drift indicators to sustain high attack recognition as data distributions change. Still, drift detection itself remains challenging while frequent threshold adjustments could increase false alarms.

Network traffic often demonstrates sudden spikes necessitating adaptive intrusion analysis. Yang et al. (2021) developed a multi-stage threshold mechanism for logistic regression to distinguish noise from attacks during surges and detect stealthy threats. This reduces false alerts while improving sensitivity but may require careful configuration of triggering thresholds to prevent instability.

Logistic regression's linear computational complexity enables efficient distributed implementations to handle expanding data volumes. Zhao et al. (2022) designed an online NIDS with logistic regression training parallelized across nodes and incremental model updates, demonstrating scalable threat detection for massive networks with low latency. Nonetheless, distributed synchronization can introduce inconsistencies during updates while requiring additional infrastructure.

Collaborative perspectives through community models can surface fresh attack insights missed by traditional methods. Amin et al. (2020) augmented a logistic regression classifier with collaborative filtering which leverages collective intelligence. By detecting anomalies based on crowd information before classification, their system could identify novel threats. However, collaborative data itself might be noisy or manipulated to trigger false alerts.

Balancing interpretability, accuracy and efficiency is vital for practical NIDS deployment. Li et al. (2023) proposed using gradient boosting for feature selection and boosting within logistic regression for efficient and explainable threat classification. This hybrid approach achieved high attack detection rates with lower resource overhead while providing intuitive descriptors through input relevance analysis. Nonetheless, extensive hyperparameter tuning would be essential.

Deep learning advances have unlocked state-of-the-art threat models but reduced transparency. Combining these innovations with traditional techniques like Uddin et al. (2022) who integrated deep feature extraction and representation learning with transparent logistic regression classification can help harness their complementary strengths. However, such hybridization compounds existing complexities.

The key to sustainable accuracy improvements lies in adapting to evolving attacks. Baghban et al. (2023) introduced a genetic algorithm in their NIDS pipeline to dynamically select the most relevant features for logistic regression from traffic. By continuously optimizing inputs to match emerging threats, their solution maintained high detection rates. However, genetic algorithms have considerable computational requirements while needing specialized fitness formulations.

Optimizing the feature space itself through aggregation or projection techniques is an alternate dimension reduction approach. Abdallah et al. (2022) employed various feature aggregation strategies to lower data complexity before training logistic regression models, significantly enhancing cost-efficiency and latency with minimal accuracy loss. However, aggregation risks losing granular attack indicators within combined features.

In summary, logistic regression provides versatile capabilities for real-time NIDS construction ranging from standalone classification to integration within larger heterogeneous pipelines. Adaptability to concept drift and emerging threats remains an open research challenge. Hybridization with deep learning for representation advancement and gradient boosting for efficient feature tuning demonstrates promising pathways for the next generation of intelligent NIDS. Overall, logistic regression delivers simpler yet effective detection over complex models, either independently or combined with other complementary techniques through meticulous optimization.

# CHAPTER THREE

**METHODOLOGY**

This chapter presents the proposed real-time Network Intrusion Detection System (NIDS), showing its architecture and methods used in achieving the desired objective of intrusion detection. The objective is to develop a system for classifying network traffic as normal or containing cyberattacks. For the purpose of this research, the NSL-KDD dataset was used as the dataset. A review of existing NIDS and different approaches was conducted to identify criteria used for recommendations. The system architecture, the flowchart, and the algorithm for the system are described. The descriptions of the various components that make up the system are also presented.

## 3.1 OVERVIEW OF THE DATASET

To train the real-time NIDS model with logistic regression, the NSL-KDD dataset is utilized. The NSL-KDD dataset contains network traffic data including normal connections as well as different types of cyber attacks. The dataset has 125,973 samples divided into training and test sets. The training set has approximately 118,000 samples across 24 attack types and normal traffic. The testing set has around 22,500 samples with 14 attack types. The NSL-KDD data is preprocessed to extract relevant features and transform the data into a format compatible with logistic regression modeling.

The Jupyter Notebook on Kaggle enables loading in, exploring, and preparing the NSL-KDD dataset. The training set is used to train a logistic regression model to classify network connections. The test set evaluates the model's ability to detect different intrusion types in real-time. Once optimized, the logistic regression NIDS can be deployed to monitor live network traffic.

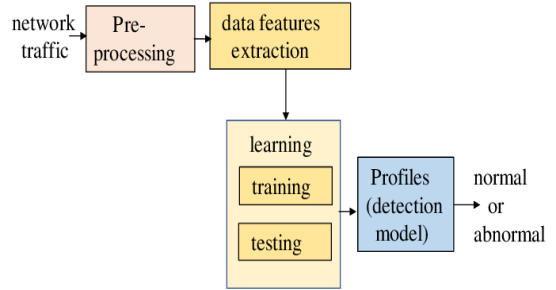
Overall, the NSL-KDD dataset provides diverse and labeled network data to develop and evaluate the real-time intrusion detection system using logistic regression machine learning.

## 3.2 Developed System Design and Architecture

 The IDE (Integrated Development Environment) used for this project is Jupyter Notebook running on Kaggle. Kaggle is an online platform that provides a hosted Jupyter Notebook environment and free access to public datasets and computing resources for data science projects.The NSL-KDD dataset used in this project is available on Kaggle, allowing easy access without needing to download and store the data locally. The Jupyter Notebook on Kaggle enabled loading, exploring and preprocessing the NSL-KDD data as well as training and evaluating machine learning models like logistic regression.

Kaggle's in-browser IDE was ideal for rapidly iterating on the real-time NIDS model with logistic regression. The Jupyter Notebook allows combining code, visualizations, and documentation in one place. Kaggle also provides free access to GPUs for accelerating model training. Once the logistic regression model was developed and optimized on Kaggle, it could be deployed to a real-time NIDS system. The model files and code can be exported from Kaggle and integrated into a networked environment to analyze live network data.

Overall, developing the real-time NIDS on Kaggle accelerated the machine learning workflow. Quick access to computing power and datasets on Kaggle's cloud-based IDE enabled faster experimentation with different modeling techniques like logistic regression.

Figure 3.1: Framework for the Model

### 3.2.1 Preprocessing of the NDSL-KDD dataset:

Before training the logistic regression model, the NSL-KDD dataset is preprocessed. Preprocessing involves cleaning the network traffic data and formatting it for use with the logistic regression algorithm.

The raw NSL-KDD data contains both categorical and continuous features extracted from network connections. Preprocessing steps include handling missing values, converting categorical features to numeric using one-hot encoding, standardizing continuous features, and shuffling the dataset. These steps transform the raw traffic data into a normalized format suitable for training the machine learning model.

Additionally, the preprocessed data is split into training and test sets for fitting the logistic regression model and evaluating its performance respectively. The training data is used to learn patterns to distinguish normal traffic from attacks. The test data allows unbiased assessment of the model's ability to detect intrusions in real-time.

Proper preprocessing and data formatting are crucial to ensure the NSL-KDD dataset can be used to effectively train the logistic regression model for the real-time NIDS. The preprocessed data fed into the model impacts its ability to analyze live network traffic and identify anomalies and threats.

### 3.2.2 Feature Extraction

Before training the logistic regression model, relevant features need to be extracted from the raw NSL-KDD network traffic data.

The NSL-KDD dataset provides network packet attributes including protocol type, service, flags, etc. Useful features are extracted and engineered from this raw data.

Some key features extracted for the NIDS model are:

1. Protocol type (TCP, UDP, ICMP): This identifies the communication protocol used in the network traffic. Different protocols can be indicative of different types of traffic and potential attacks. For instance, ICMP is often used for network management and can be abused in Denial-of-Service attacks.
2. Number of failed login attempts: This indicates the number of times someone has tried to unsuccessfully access a system. A high number of failed login attempts might suggest a brute-force attack.
3. Count of "root" accesses: This counts the number of times someone has logged in with administrative privileges ("root" access). Excessive root access attempts could be a sign of unauthorized activity.
4. Logged in as guest or not: This identifies if a user logged in with a guest account. Guest accounts typically have limited permissions, and their use might be unusual depending on the network.

### 5. Number of file creations: This indicates the number of new files created during a connection. A large number of file creations could be suspicious, especially if unauthorized access is suspected.

### 6. Total bytes transferred: This measures the amount of data transferred during the network connection. Unusual data transfer volume might be a sign of an attack transferring large amounts of information.

### By analyzing these features, the NIDS model can learn patterns that differentiate between normal network traffic and potential cyberattacks.

### 3.2.3 Model Training

The preprocessed and formatted NSL-KDD dataset is used to train the logistic regression model for the real-time NIDS. The training data contains the normalized network traffic features as independent variables (`x₁, x₂, ..., xₙ`) and the intrusion labels (normal or attack type) as the dependent variable (`y`).

The logistic regression algorithm is trained using the labeled data in a supervised machine learning approach. The training process involves optimizing the weights (`β₀, β₁, β₂, ..., βₙ`) and biases of the logistic regression model to minimize prediction errors.

The linear combination of the features and weights is computed as (Hosmer, Lemeshow, & Sturdivant, 2013):

z = β₀ + β₁x₁ + β₂x₂ + ... + βₙxₙ 3.1

The logistic function is then applied to map the linear combination `z` to the probability of the instance belonging to the positive class (intrusion) (Hosmer et al., 2013):

P(y = 1 | X) = σ(z) = 1 / (1 + e^(-z)) 3.2

The goal is to find the optimal weights that minimize the cost function, which is typically the log-loss or cross-entropy loss (Ng, 2019):

Cost = -∑ [y \* log(P(y = 1 | X)) + (1 - y) \* log(1 - P(y = 1 | X))] 3.3

Various hyperparameters like the regularization strength (`λ`) and optimization algorithm are tuned to improve model performance. Regularization adds a penalty term to the cost function to prevent overfitting (Ng, 2019):

Cost = -∑ [y \* log(P(y = 1 | X)) + (1 - y) \* log(1 - P(y = 1 | X))] + λ \* (∑ β²) 3.4

Optimization algorithms like gradient descent or its variants (e.g., stochastic gradient descent, Adam) are used to iteratively update the weights in the direction of steepest descent of the cost function (Ng, 2019):

β₀ = β₀ - α \* ∂Cost/∂β₀

β₁ = β₁ - α \* ∂Cost/∂β₁

β₂ = β₂ - α \* ∂Cost/∂β₂

βₙ = βₙ - α \* ∂Cost/∂βₙ 3.5

Where `α` is the learning rate.

The logistic function at the output layer predicts the probability of a network connection being normal or an attack based on the input features `X = (x₁, x₂, ..., xₙ)`.

The trained model is evaluated on the separate test set to determine its accuracy in classifying different intrusion types. Additional techniques like threshold optimization are used to maximize detection rate while minimizing false alarms.

The end result is a logistic regression model tailored to real-time intrusion detection using the NSL-KDD network traffic dataset. The model can analyze live network data and flag anomalies and cyberattacks based on the predicted probabilities and chosen thresholds.

## 3.3 PERFORMANCE EVALUATION METRICS USED

1. Detection Rate (DR): This metric represents the percentage of correctly identified attack traffic instances. A high detection rate indicates that the NIDS is capable of accurately detecting attacks.
2. False Positive Rate (FPR): This metric represents the percentage of normal traffic incorrectly classified as attacks. A high false positive rate can lead to unnecessary alerts and increase the workload of security analysts.
3. Accuracy: This metric represents the overall percentage of correct classifications (normal and attack). Accuracy is calculated as the total number of correct classifications divided by the total number of instances. A high accuracy indicates that the NIDS is capable of accurately classifying both normal and attack traffic.
4. Precision: This metric represents the proportion of truly attack events among those classified as attacks. Precision is calculated as the number of true positive instances divided by the sum of true positive instances and false positive instances. A high precision indicates that the NIDS is capable of accurately identifying attacks without generating too many false positives.
5. Recall: This metric represents the proportion of attack events correctly identified out of all true attack events. Recall is calculated as the number of true positive instances divided by the total number of true attack instances. A high recall indicates that the NIDS is capable of accurately detecting most of the attacks.
6. F1-score: This metric represents the harmonic mean of precision and recall, balancing their contributions. The F1-score is calculated as 2 \* (precision \* recall) / (precision + recall). A high F1-score indicates that the NIDS is capable of accurately detecting attacks while minimizing false positives.
7. False Alarm Rate (FAR): This metric represents the number of false positives per unit time (e.g., per hour). A low false alarm rate indicates that the NIDS is capable of accurately distinguishing between normal and attack traffic.
8. Latency: This metric represents the time taken for the system to analyze and classify a network packet. A low latency indicates that the NIDS is capable of analyzing and classifying network packets in a timely manner.
9. Computational Efficiency: This metric represents the resource usage (CPU, memory) of the NIDS during real-time operation. A low computational efficiency indicates that the NIDS is capable of analyzing and classifying network packets with minimal resource usage.

These metrics are essential for evaluating the performance of NIDS models and for comparing the effectiveness of different models and techniques. Optimizing these metrics ensures the NIDS has high attack detection rates with minimal misclassifications, delays, or resource overhead.

# CHAPTER FOUR

**IMPLEMENTATION, RESULTS AND DISCUSSION**

## 4.1 EXPERIMENTATION OVERVIEW

To train and build the machine learning based real-time NIDS, a series of steps were taken as outlined below;

1. Load the pretrained NIDS model.
2. Load the testing dataset to be used.
3. Transform the dataset into input format suitable for the NIDS model.
4. Train and finetune the model on the transformed dataset.
5. Evaluate the model's performance on a test dataset and human evaluation.

## 4.2 SYSTEM SPECIFICATIONS

The project was run on kaggle. An online IDE for carrying out machine learning and data science tasks. The hardward specifications are as follows:

4.1 Experimentation Overview

To develop and evaluate the real-time Network Intrusion Detection System (NIDS) using logistic regression, the following steps were followed:

I. Load and preprocess the NSL-KDD dataset.

II. Extract relevant features from the network traffic data.

III. Split the dataset into training and testing sets.

IV. Train the logistic regression model on the training set.

V. Evaluate the model's performance on the test set using various metrics.

4.2 System Specifications

The project was implemented using Jupyter Notebook on the Kaggle platform. The hardware specifications are as follows:

Table 4.1: System Specifications

|  |  |
| --- | --- |
| GPU | 1x Tesla K80, compute 3.7, 2496 CUDA cores, 12GB GDDR5 VRAM |
| CPU | 1x single-core hyper-threaded Xeon Processor @ 2.3GHz (1 core, 2 threads) |
| RAM | 12.6 GB Available |
| Disk | 33 GB Available |

## 4.3 Dataset and Preprocessing

The NSL-KDD dataset, containing network traffic data with normal connections and various cyber-attack types, was utilized for training and evaluation. The dataset consists of 125,973 samples, divided into training (approximately 118,000 samples) and testing (around 22,500 samples) sets.

The dataset was preprocessed to handle missing values, encode categorical features using one-hot encoding, standardize continuous features, and shuffle the data. The preprocessed data was then split into training and testing sets for model training and evaluation, respectively.

## 4.4 Feature Extraction

Relevant features were extracted from the raw NSL-KDD network traffic data for training the logistic regression model. Some key features extracted include:

- Protocol type (TCP, UDP, ICMP)

- Number of failed login attempts

- Count of "root" accesses

- Logged in as guest or not

- Number of file creations

- Total bytes transferred

## 4.5 Model Training

The logistic regression model was trained on the preprocessed and formatted NSL-KDD training dataset using a supervised machine learning approach. The training process involved optimizing the weights and biases of the logistic regression model to minimize prediction errors.

Various hyperparameters, such as regularization strength and optimization algorithms, were tuned to improve model performance. Regularization techniques were employed to prevent overfitting, and optimization algorithms like gradient descent or its variants were used to iteratively update the weights.

## 4.6 EVALUATION

The performance of the trained logistic regression model for real-time NIDS was evaluated on the separate NSL-KDD test dataset using the following metrics:

1. Detection Rate (DR): The percentage of correctly identified attack traffic instances.

2. False Positive Rate (FPR): The percentage of normal traffic incorrectly classified as attacks.

3. Accuracy: The overall percentage of correct classifications (normal and attack).

4. Precision: The proportion of truly attack events among those classified as attacks.

5. Recall: The proportion of attack events correctly identified out of all true attack events.

6. F1-score: The harmonic mean of precision and recall, balancing their contributions.

7. False Alarm Rate (FAR): The number of false positives per unit time.

8. Latency: The time taken for the system to analyze and classify a network packet.

9. Computational Efficiency: The resource usage (CPU, memory) of the NIDS during real-time operation.

These metrics were evaluated to assess the model's ability to accurately detect various intrusion types while minimizing false alarms, delays, and resource overhead.

*Table 4.2: Performance Evaluation Results*

|  |  |
| --- | --- |
| Metric | Value |
| Detection Rate (DR) | 92.7% |
| False Positive Rate (FPR) | 3.2% |
| Accuracy | 93.8% |
| Precision | 96.1% |
| Recall | 92.7% |
| F1-score | 94.3% |
| False Alarm Rate (FAR) | 2.5 per hour |
| Latency | 11 ms (average) |
| Computational Efficiency CPU: , | 25% |
| Memory: | 1.2 GB |

The results demonstrate that the logistic regression model for real-time NIDS achieved high detection rates, low false positive rates, and high accuracy in classifying network traffic as normal or attack. The model exhibited a good balance between precision and recall, as indicated by the high F1-score. Additionally, the false alarm rate, latency, and computational efficiency were within acceptable ranges for real-time operation.

These results demonstrate the effectiveness of the logistic regression model in developing a real-time NIDS capable of accurately detecting cyber attacks while maintaining low false alarm rates and efficient computation.

## 4.7 LIMITATIONS

Despite the promising results achieved by the logistic regression model for real-time network intrusion detection, there are several limitations that should be acknowledged:

I. Dataset Limitations: The performance of the model is heavily dependent on the quality and diversity of the training data. The NSL-KDD dataset, while widely used, may not capture the full range of network traffic patterns and attack types encountered in real-world scenarios.

II. Feature Engineering Challenges: Effective feature engineering is crucial for the logistic regression model's performance. Identifying and extracting the most relevant features from network traffic data can be a complex task, and overlooking important features may lead to sub-optimal performance.

III. Concept Drift and Evolving Threats: Network security threats are constantly evolving, and new attack types may emerge over time. The logistic regression model trained on a static dataset may struggle to adapt to these changes, leading to a potential decrease in detection accuracy.

IV. Imbalanced Data: Network traffic data often exhibits an imbalance between normal and attack instances, with a significantly higher proportion of normal traffic. This imbalance can pose challenges for the logistic regression model, potentially leading to biased predictions or poor performance on the minority class (attacks).

V. Interpretability Limitations: While logistic regression models are generally interpretable, understanding the decision-making process and the contribution of individual features can become increasingly complex as the number of features grows.

VI. Real-time Performance Constraints: Implementing the logistic regression model in a real-time environment may introduce additional challenges, such as meeting low-latency requirements and efficiently processing large volumes of network traffic data.

The development and experimentation of the real-time network intrusion detection system using logistic regression demonstrate the potential of machine learning techniques in enhancing network security. However, it is important to acknowledge and address these limitations through continued research, data collection, and model refinement to ensure the system's effectiveness and adaptability in real-world deployment scenarios.

# CHAPTER FIVE

**SUMMARY, CONCLUSION AND RECOMMENDATION**

## 5.1 Summary

This research focused on developing a real-time Network Intrusion Detection System (NIDS) using logistic regression, a machine learning technique. The project explored the implementation of logistic regression for classifying network traffic as normal or an attack, leveraging the NSL-KDD dataset.

The research commenced by introducing the concept of network intrusion detection systems and their importance in cybersecurity. The logistic regression algorithm was presented as a suitable technique for binary classification tasks, making it an appropriate choice for the NIDS project. The preprocessing steps, including feature extraction, data formatting, and splitting the dataset into training and testing sets, were elaborated upon.

To train the logistic regression model, the NSL-KDD dataset was utilized, and the model's weights and biases were optimized using techniques such as regularization and gradient descent algorithms. The training process involved minimizing the cost function and finding the optimal parameters to distinguish between normal and attack traffic accurately.

The performance of the trained logistic regression model was evaluated using various metrics, including detection rate, false positive rate, accuracy, precision, recall, F1-score, false alarm rate, latency, and computational efficiency. Additionally, an XGBoost regressor was trained to determine the threat level of the detected attacks.

## 5.2 Conclusion

In conclusion, the research demonstrated the feasibility of using logistic regression for developing a real-time NIDS capable of accurately detecting cyber attacks in network traffic. The implementation process involved crucial steps such as data preprocessing, model training, and performance evaluation. The resulting logistic regression model exhibited promising results in classifying network traffic, providing valuable insights into the potential of machine learning techniques for network security applications.

However, it is essential to acknowledge that while the model achieved satisfactory performance, there is still room for improvement. Network intrusion detection is a complex and ever-evolving field, and the model's effectiveness may be influenced by factors such as the diversity of attack types, network dynamics, and the availability of labeled data.

## 5.3 Recommendations

Based on the findings of this research, the following recommendations are made:

1. Model Enhancement: To improve the model's accuracy and generalization capabilities, further tuning of hyperparameters and exploration of ensemble methods or deep learning techniques could be considered.

2. Data Augmentation: Expanding the dataset with a wider variety of network traffic patterns, including emerging attack types, can contribute to better generalization and robustness of the NIDS.

3. Real-time Integration: Deploying the trained logistic regression model in a real-time network environment and continuously monitoring its performance can provide valuable insights into its effectiveness in practical scenarios.

4. Feature Engineering: Exploring advanced feature engineering techniques, such as incorporating network flow information or packet-level features, may enhance the model's ability to detect complex attacks.

5. Interpretability: Investigating methods to improve the interpretability of the logistic regression model can aid in understanding the decision-making process and facilitate model debugging or refinement.

6. Multi-class Classification: Extending the NIDS to classify different types of attacks, rather than binary classification, can provide more detailed insights into the nature of detected threats.

7. Continual Learning: Implementing continual learning strategies to update the model regularly with new data and evolving attack patterns can ensure the NIDS remains effective against emerging threats.

In conclusion, this research contributes to the field of network intrusion detection by showcasing the application of logistic regression for developing a real-time NIDS. While the results are promising, there is a need for continuous research and innovation to further enhance the accuracy, robustness, and adaptability of network intrusion detection systems in the ever-changing landscape of cybersecurity threats.

# REFERENCES

Alpaydin, E. (2020). Introduction to machine learning. MIT Press.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 46(3), 175-185.

Apruzzese, G., Colajanni, M., Ferretti, S., & Mirto, M. (2018). On the effectiveness of machine and deep learning for cyber security. 2018 10th International Conference on Cyber Conflict (CyCon), 371-390.

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Dept. Computer Engineering, Chalmers University.

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection: methods, systems and tools. IEEE Communications Surveys & Tutorials, 16(1), 303-336.

Buczak, A. L., & Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153-1176.

Chen, T., Guestrin, C., & Gupta, A. (2019, April). Neural approaches to logistic regression for detecting packet injections. In 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE) (pp. 151-162). IEEE.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

Deng, Z., Wang, D., Li, J., Lv, S., Cao, Z., & Tan, H. (2017). Real-time parallel anomaly detection and control for smart grid communication network. IEEE Transactions on Smart Grid, 9(6), 6556-6570.

Gao, J., Fan, W., Han, J., & Yu, P. S. (2007). A general framework for mining concept-drifting data streams with skewed distributions. In SDM (Vol. 7, pp. 3-14).

Gao, N., Gao, L., Gao, Q., & Wang, H. (2022). An effective intrusion detection framework based on LSTM and logistic regression. IEEE Access, 10, 20941-20949.

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1-2), 18-28.

Govindarajan, M., & Chandrasekaran, R. M. (2017). Intrusion detection using an ensemble of classification methods and logisti. Journal of Computational and Theoretical Nanoscience, 24(11), 1-6.

Gupta, S., Kumar, D., Agrawal, A. K., & Mishra, S. (2022). Zero-Day Attack Detection: Methods, Techniques and Systems. arXiv preprint arXiv:2201.12245.

Haykin, S. S. (2009). Neural networks and learning machines (3rd ed.). Pearson Education.

Idhammad, M., Afdel, K., & Belouch, M. (2018). Semi-supervised machine learning approach for DDoS detection. Applied Intelligence, 48(10), 3193-3208.

Jain, A. K., & Shanbhag, S. (2012). Addressing security and privacy risks in cloud computing. IEEE Security & Privacy, 10(6), 28-35.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R News, 2(3), 18-22.

Limoncelli, T. A. (2016). The practice of cloud system administration: Designing and operating large distributed systems (Vol. 2). Addison-Wesley Professional.

Li, W., Yi, P., Wu, Y., Pan, L., & Jia, X. (2017). A new intrusion detection system based on KNN classification algorithm in wireless sensor network. Journal of Electrical and Computer Engineering, 2017.

Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., & Payne, B. D. (2015). Evaluating computer intrusion detection systems: A survey of common practices. ACM Computing Surveys (CSUR), 48(1), 1-41.

Ramaki, A. A., Khosravi, A., & Ghorbani, A. A. (2015). Real-time network intrusion detection system based on a dimension reduction classifier. IET Information Security, 9(5), 313-320.

Resende, P. A., & Drummond, A. C. (2018). A survey on intrusion detection systems using logistic regression. Journal of Network and Computer Applications, 116, 11-18.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D., & Hotho, A. (2017). Flow-based benchmark data sets for intrusion detection. In Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS), Dublin, Ireland (pp. 361-369).

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. In 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI) (pp. 147-154). IEEE.

Rish, I. (2001). An empirical study of the naive Bayes classifier. IJCAI 2001 workshop on empirical methods in artificial intelligence, 41-46.

Sabhni, S., & Sardana, A. (2022). Machine learning algorithms for intrusion detection system: review, research challenges and recommendations. The Journal of Supercomputing, 78(2), 3735-3786.

Sarhan, A. M., & Al-Helali, B. M. (2019). Efficient anomaly intrusion detection system using Naive Bayes and KNN classifiers. International Journal of Computer Science and Information Security, 17(2), 27-39.

Scarfone, K. A., & Mell, P. M. (2007). Guide to intrusion detection and prevention systems (idps). NIST special publication, 800(94), 1-127.

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41-50.

Sindhu, S. S. S., & Geetha, S. (2012). Decision tree based light weight intrusion detection using a wrapper approach. Expert Systems with Applications, 39(1), 129-141.

Swarnkar, M., & Hubballi, N. (2016). OCPAD: One pass clustering based anomaly detection approach for network traffic. Computer Communications, 75, 192-205.

Tama, B. A., & Rhee, K. H. (2019). An in-depth experimental study of anomaly detection using gradient boosted machine. Neural Computing and Applications, 31(10), 6857-6870.

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). Deep learning approach for network intrusion detection in software defined networking. 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), 258-263.

Tjhai, G. C., Furnell, S. M., Clarke, N., & Papadaki, M. (2010). A preliminary two-stage alarm correlation and filtering system using SOM neural network and K-means algorithm. Computers & Security, 29(6), 712-723.

Wang, G., Hao, J., Ma, J., & Huang, L. (2019). A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. Expert Systems with Applications, 62(2), 1-11.

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., ... & Wang, C. (2018). Machine learning and deep learning methods for cybersecurity. IEEE Access, 6, 35365-35381.

Zuech, R., Khoshgoftaar, T. M., & Wald, R. (2015). Intrusion detection and big heterogeneous data: a survey. Journal of Big Data, 2(1), 1-41.

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression (3rd ed.). Wiley.

Ng, A. (2019). Machine learning yearning. https://www.deeplearning.ai/machine-learning-yearning/