Report for Minor Project of Bachelor of Computer Engineering

Network Intrusion Detection System

Submitted by:

Prajjwal Adhikari [BEC-2020-20]

Rozal Dahal [BEC-2020-27]

Aman Poudel [BEC-2020-04]

Raj Gurung [BEC-2020-23]

Advisor:

Er. Pukar Neupane

Submitted To:

United Technical College

Faculty of Science and Technology

Pokhara University, Nepal

June 2024

Abstract

The escalating growth of the Internet and communication networks has led to a surge in transmitted data, making it a prime target for attackers who continually devise novel methods to compromise or manipulate this information. Network security, therefore, faces significant challenges, necessitating robust intrusion detection systems (IDS) to safeguard against evolving cyber threats. This project proposes the development of a Network Intrusion Detection System (NIDS) by leveraging machine learning algorithms, specifically focusing on real-time monitoring, anomaly-based detection, and efficient response and reporting.

Keywords: *Network Intrusion Detection System, Machine Learning, Cybersecurity, Anomaly Detection, Real-time Monitoring, Security Protocols, Software Defined Networking, Intrusion Detection, Deep Learning, SDN-based NIDS.*

\

Table of Contents

List of Figures

Abbreviation and Acronyms

CNN	Convolution Neural Network
CPU	Central Processing Unit
DARPA	Defense Advance Research Project Agency
DL	Deep Learning
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Protection System
KNN	K-Nearest Neighbor
LAN	Local Area Network
ML	Machine Learning
NIDS	Network Intrusion Detection System
RAM	Random Access Memory
SDN	Software Defined Networking Technology
SOHO	Small Office Home Office
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transfer Control Protocol
TLS	Transport Layer Security

Chapter 1: Introduction

1.1 Background

In today's interconnected and digitally dependent world, the security of computer networks is of paramount importance. Organizations face a constant and evolving threat landscape where cyberattacks are becoming increasingly sophisticated and frequent. Cyber attackers continually devise novel methods to compromise or manipulate information transmitted over networks, making it a prime target for malicious activities. This necessitates the implementation of robust security measures to safeguard sensitive data, maintain business continuity, and protect against unauthorized access [1].

One critical component in the arsenal of network security is the Network Intrusion Detection System (NIDS). A NIDS is a specialized security solution designed to monitor, analyze, and detect suspicious or malicious activities within a computer network. It acts as a vigilant sentry, tirelessly observing network traffic in real-time or near-real time, with the goal of identifying and responding to potential security threats . Unlike firewalls, which primarily block unauthorized access based on predefined rules, NIDS focuses on detecting threats that have bypassed initial security defenses [2].

Network Intrusion Detection Systems are generally categorized into two types: signature-based and anomaly-based detection systems. Signature-based detection involves comparing network traffic against a database of known attack signatures. While this method is effective at identifying known threats, it falls short in detecting new or unknown attacks (zero-day attacks). On the other hand, anomaly-based detection systems establish a baseline of normal network behavior and flag deviations from this baseline as potential threats. This approach is more adept at identifying novel threats but often suffers from high false-positive rates [3].

In recent years, the application of machine learning techniques in NIDS has shown promising results. Machine learning algorithms can analyze vast amounts of network traffic data, identify patterns, and predict potential threats with higher accuracy compared to traditional methods. Algorithms such as Random Forests, k-nearest neighbors (KNN), Convolutional Neural Networks

(CNNs), and Support Vector Machines (SVMs) have been employed to improve the detection rates and reduce false positives in NIDS [4].

The integration of Software Defined Networking (SDN) with NIDS is another emerging trend. SDN allows for more flexible and programmable network management, which can be leveraged to enhance the capabilities of NIDS. By integrating SDN, NIDS can dynamically adjust to changing network conditions and deploy security measures more efficiently [5].

Overall, the development of a robust Network Intrusion Detection System is crucial for protecting organizational networks against the evolving landscape of cyber threats. By leveraging advanced technologies and integrating scalable solutions, NIDS can provide a formidable defense mechanism to ensure network security and integrity.

1.2 Statement of Problem

In contemporary network security, the surge in sophisticated cyber threat poses a substantial challenge to the resilience of organizational networks. Existing intrusion detection mechanisms, while fundamental, encounter limitations in effectively addressing the dynamic nature of modern cyber threats. Signature-based detection often falls short in identifying novel or zero-day attacks, and anomaly-based approaches may struggle with high false-positive rates [6].

Additionally, the scalability of intrusion detection systems becomes a concern as network expands in complexity. Furthermore, the lack of a unified and intelligent Network Intrusion Detection System (NIDS) that seamlessly integrates signature and anomaly detection, adapts to emerging threats and ensures scalability, hampers the ability of organization to proactively safeguard their networks against a diverse range of cyber threats [1].

1.3 Objective

Our project Network Intrusion Detection System is going to be developed to meet the following objectives:

- To have Real-time Monitoring of network packets in system
- To build the features of Anomaly-Based Detection of network intrusion
- To achieve the Response and Reporting after detection of intrusion at system

1.4 Application

A NIDS plays a pivotal role in bolstering network security by providing a second line of defense. While Firewalls and others preventive measures are crucial, they may not catch all potential threats. NIDS complements these preventive measures by actively monitoring traffic and identifying anomalies or patterns indicative of malicious activity.

1.5 Limitations and Scope

This project is built with intention to be implemented in SO-HO devices or the small networks like LAN network to detect the packets lively if possible else to analyze the network packets to detect the intrusion in the system.

This project is limited to monitoring the network traffic working as the second line of defense to coordinate with firewalls and monitor the overall traffic either in real-time or nearly real time in network to identify the anomalies.

Chapter 2: Literature Review

1.1 Introduction

With the intention to meet the objective we have set for the project accomplishment we have decided to go through the different paperwork that has been made in the history. Network Intrusion Detection System is a system that will be able to monitor network traffic and report anomalies. A lot of research has been done for this project by our team on Network Intrusion Detection System and here are some of the study's results.

1.2 Case Study

1. **Network Intrusion Detection System Using Neural Network**

   This research introduces a neural network-driven approach for detecting internet-based attacks on computer networks. Intrusion Detection Systems (IDS) have been developed to anticipate and prevent both existing and upcoming cyber threats. The method relies on neural networks to recognize and forecast abnormal behaviors within the system, specifically employing feedforward neural networks with the backpropagation training algorithm. The study utilized training and testing data extracted from the Defense Advanced Research Projects Agency (DARPA) intrusion detection evaluation datasets. The empirical findings, based on actual data, demonstrated encouraging outcomes for the detection of intrusion systems through the application of neural networks [7].

2. **Survey on SDN based network intrusion detection system using machine learning approaches**

   Software Defined Networking Technology (SDN) presents an opportunity to efficiently identify and monitor network security issues by virtue of its programmable features. Recently, Machine Learning (ML) approaches have been integrated into SDN-based Network Intrusion Detection Systems (NIDS) to safeguard computer networks and address security challenges. A series of advanced machine learning techniques, particularly deep learning technology (DL), is emerging within the SDN framework. This survey explores recent research on ML methods that leverage SDN for NIDS implementation, with a specific focus on the application of deep learning techniques. Additionally, the survey covers tools applicable for developing NIDS models within the SDN environment. It concludes with a discussion on ongoing challenges in implementing NIDS using ML/DL and outlines potential future research directions [8].

3. **A high-performance network intrusion detection system**

   This paper presents a new approach for network intrusion detection systems based on concise specifications that characterize normal and abnormal network packet sequences. They have specification language geared for a robust network intrusion detection by enforcing a strict type of discipline via a combination of static and dynamic type of checking. Unlike most of other approaches in network intrusion detection this approach can easily support new network protocols as information relating to the protocols are not hard coded into the system instead, they have added suitable type definitions in the specifications and define intrusion patterns on these types [9].

4. **A Study of NIDS using Artificial Intelligence/ Machine Learning**

   The surge in Internet and communication growth has led to a substantial increase in transmitted data, attracting continuous novel attacks from adversaries seeking to steal or corrupt this valuable information. Intrusion Detection Systems (IDS) play a crucial role in detecting such intrusions by examining network traffic. Despite numerous research efforts to enhance IDS solutions, there remains a need for improvement in detection accuracy and reduction of false alarm rates, especially against zero-day attacks. Machine learning algorithms have gained popularity for efficient and accurate network intrusion detection. This paper introduces the concept of IDS, presents a taxonomy of machine learning methods, discusses key metrics for IDS assessment, and reviews recent ML-based IDS solutions, highlighting their strengths and weaknesses. It also examines datasets used in these studies, discusses result accuracy, and concludes with observations, research challenges, and future trends in the field [10].

With the study of different project, surveys and case studies a conclusion is made that using the machine learning technique with predefined sets of anomalies should be trained to achieve the network intrusion detection system with use of algorithms like KNN, Random Forest Classifiers etc. for our project of Network Intrusion Detection System.

Chapter 3: Methodology

3.1 Flow of project

A project flow diagram is a visual representation that outlines the sequence and dependencies of tasks or activities in a project. It illustrates the flow of work from start to finish, highlighting the order and relationships between different project components. The diagram helps stakeholders understand the project's timeline, milestones, and critical paths, facilitating effective project planning, coordination, and communication.

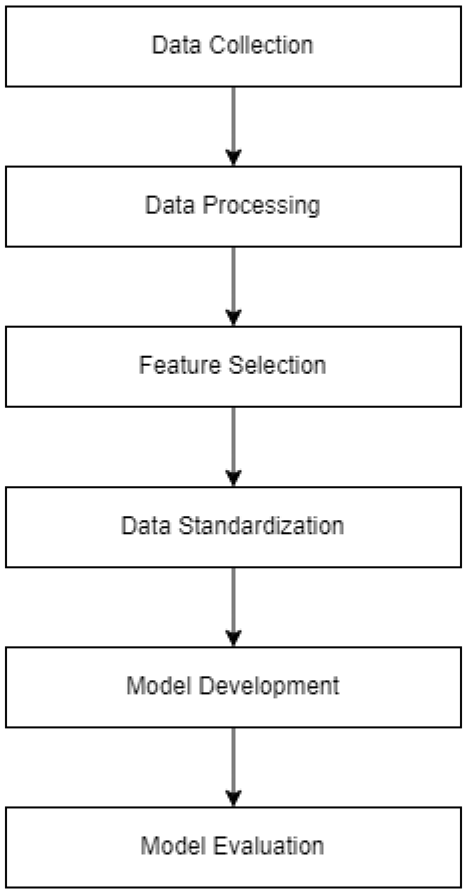

Figure 1 : Project Flow Diagram (Group Study)

3.1.1 Data Collection

The dataset used for training and testing the Network Intrusion Detection System (NIDS) was obtained from Kaggle [11]. This dataset, titled "Network Intrusion Detection," includes various features representing network traffic attributes and is labeled to indicate whether a given traffic record is normal or an anomaly.

3.1.2 Data processing

Data processing involved several steps to clean and prepare the data for analysis:

- **Loading Data**: The dataset was loaded into data frames for both training and testing purposes, ensuring that the data was ready for subsequent analysis.
- **Exploratory Data Analysis (EDA)**: An initial exploration of the dataset was conducted to understand its structure and characteristics. This included checking for missing values, data types, and general statistics of the features. This step helped in identifying potential issues and gaining insights into the distribution and nature of the data.
- **Label Encoding**: Categorical variables in the dataset were converted into numerical format using label encoding. This conversion is essential as most machine learning algorithms require numerical input. By encoding categorical variables, we ensured that the models could process the data effectively.
- **Feature Dropping**: Features that were deemed unnecessary or redundant were removed from the dataset. This step was crucial for streamlining the dataset and improving the efficiency of the model training process. By eliminating irrelevant features, we reduced the complexity of the models and enhanced their performance.

3.1.3 Feature Selection

Feature selection was performed to identify the most relevant features for model training. Recursive Feature Elimination (RFE) was employed in conjunction with a Random Forest classifier to select the top 10 features. This method iteratively removes the least important features based on the model’s performance, ensuring that only the most impactful features are used for training. This process helps in improving the model's accuracy and reduces overfitting by focusing on the most significant predictors.

3.1.4 Data Standardization

The selected features were standardized to ensure that they have a mean of zero and a standard deviation of one. Standardization is crucial for the performance of many machine learning algorithms, as it ensures that all features contribute equally to the model's learning process. By scaling the features, we eliminated any biases that could arise from differences in the units or scales of the input data, thereby enhancing the models' convergence and accuracy.

3.1.5 Model Development

Multiple machine learning models were developed and evaluated to identify the best-performing algorithm for intrusion detection. The models included Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree classifiers.

- **Logistic Regression**: A logistic regression model was developed as a baseline due to its simplicity and effectiveness in binary classification tasks. This model provides a good starting point for comparison with more complex algorithms.
- **K-Nearest Neighbors (KNN)**: The KNN model's hyperparameters were optimized using Optuna, an automatic hyperparameter optimization framework. By determining the optimal number of neighbors, we aimed to enhance the model's accuracy in classifying network traffic records.
- **Decision Tree**: The Decision Tree classifier's depth and the number of maximum features were optimized to improve its performance in detecting network intrusions. By fine-tuning these parameters, we aimed to achieve a balance between the model's complexity and its predictive power.

3.1.6 Model Evaluation

The models were evaluated using various metrics, including accuracy, precision, recall, F1-score, and confusion matrices. Cross-validation was also performed to ensure the robustness of the models.

- **Cross-Validation**: A 10-fold cross-validation was conducted for each model to evaluate their performance consistently and mitigate overfitting. This method involves dividing the dataset into ten subsets, training the model on nine subsets, and validating it on the remaining subset. The process is repeated ten times, with each subset used exactly once as the validation data. This approach provides a comprehensive assessment of the model's performance.

- **Performance Metrics**: Detailed performance metrics were compiled, including confusion matrices and classification reports, to provide a comprehensive evaluation of each model's predictive capabilities. These metrics helped in understanding the strengths and weaknesses of each model in detecting normal and anomalous network traffic.

3.2 Development Tools

Development tools for projects encompass a wide range of software applications, frameworks, and utilities that aid in the creation and management of project deliverables. These tools streamline development processes, enhance collaboration, and provide features for code editing, version control, testing, and project tracking.

1. **Programming Languages:**
    - Python: Widely used for its simplicity and extensive libraries, Python is suitable for implementing various components of a NIDS [12].
    - C/C++: Ideal for low-level operations and performance-critical tasks within the NIDS [13].
    - Java: Suitable for developing cross-platform applications and components [14].

2. **Packet Capture and Analysis:**
    - Scapy: A powerful Python library for capturing, manipulating, and sending network packets [15].
    - Wireshark: A widely used packet analysis tool that can be integrated into the development process for testing and validation [16].

3. **Deep Packet Inspection (DPI):**

- Snort: An open-source NIDS that includes a DPI engine for analyzing network traffic and detecting signatures of known attacks [17].
- Suricata: An open-source IDS/IPS engine that supports multi-threading and is designed for high-performance deep packet inspection [17].

4. **Machine Learning and Anomaly Detection:**

- Scikit-learn: A machine learning library for Python, useful for implementing supervised and unsupervised learning algorithms [18].
- TensorFlow and PyTorch: Deep learning frameworks that can be employed for building neural network-based anomaly detection models [18].

5. **Database Management:**
    - MySQL or PostgreSQL: Relational database management systems that can be used to store configuration data, logs, and other relevant information [19].
    - Elasticsearch: A distributed search and analytics engine, useful for storing and querying large volumes of log data [20].

6. **Data Visualization:**
    - Matplotlib and Seaborn: Python libraries for creating static, animated, and interactive visualizations of data [21].
    - Kibana: A data visualization tool often used with Elasticsearch for exploring and visualizing log data [21].

7. **Security Protocols and Encryption:**
    - SSL/TLS: For securing communication channels and encrypting data [22].
    - IPsec: A suite of protocols for securing Internet Protocol (IP) communications [22].

8. **Logging and Reporting:**
    - Syslog: A standard protocol for sending log messages in a network [23].
    - Logstash: Log data processing tool that can collect, parse, and enrich log data [23].

9. **Network Protocols and Standards:**
    - TCP/IP Stack: Understanding of the TCP/IP protocol suite is fundamental for analyzing network traffic [24].

10. **Version Control:**
    - Git: A distributed version control system for tracking changes in the source code [25].

3.3 System Design

This portion of paper contains the architecture of Network Intrusion Detection System based on the information collected in earlier portion of document. It contains diagrams like System Diagrams, Flow Charts etc.

3.3.1 System Diagram

Picture Shown below is a general view of Network Intrusion System Implementation segment to monitor the traffic.

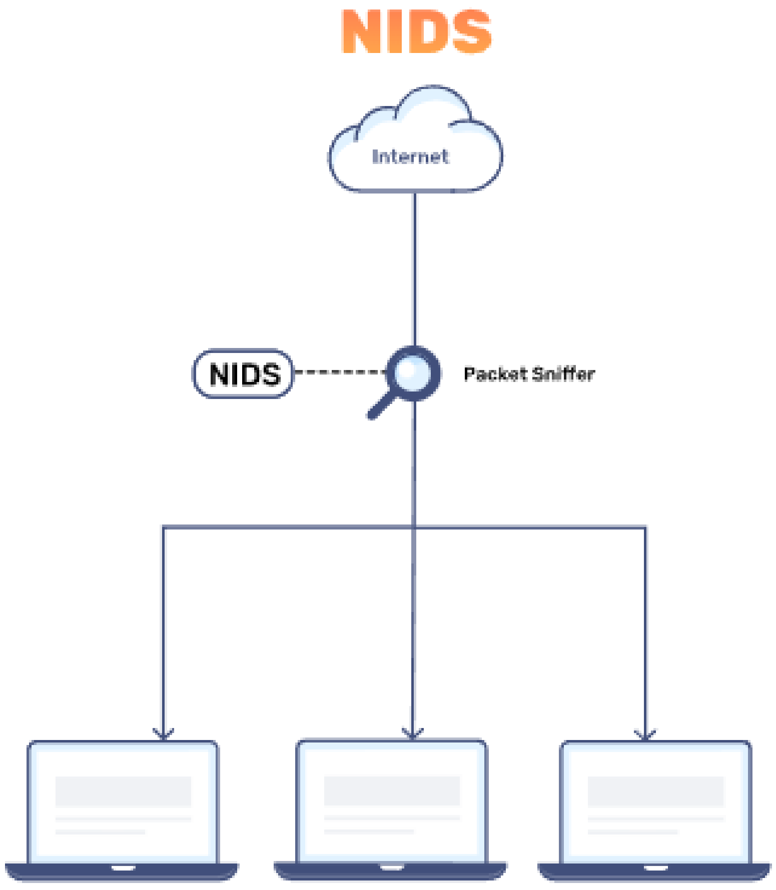

Figure 2: NIDS (Bunny.net)

3.3.2 ER diagram

An ER diagram, short for Entity-Relationship diagram, is a graphical representation of the entities, attributes, and relationships within a database system. It provides a visual tool for designing and understanding the structure and relationships of the data in a database.

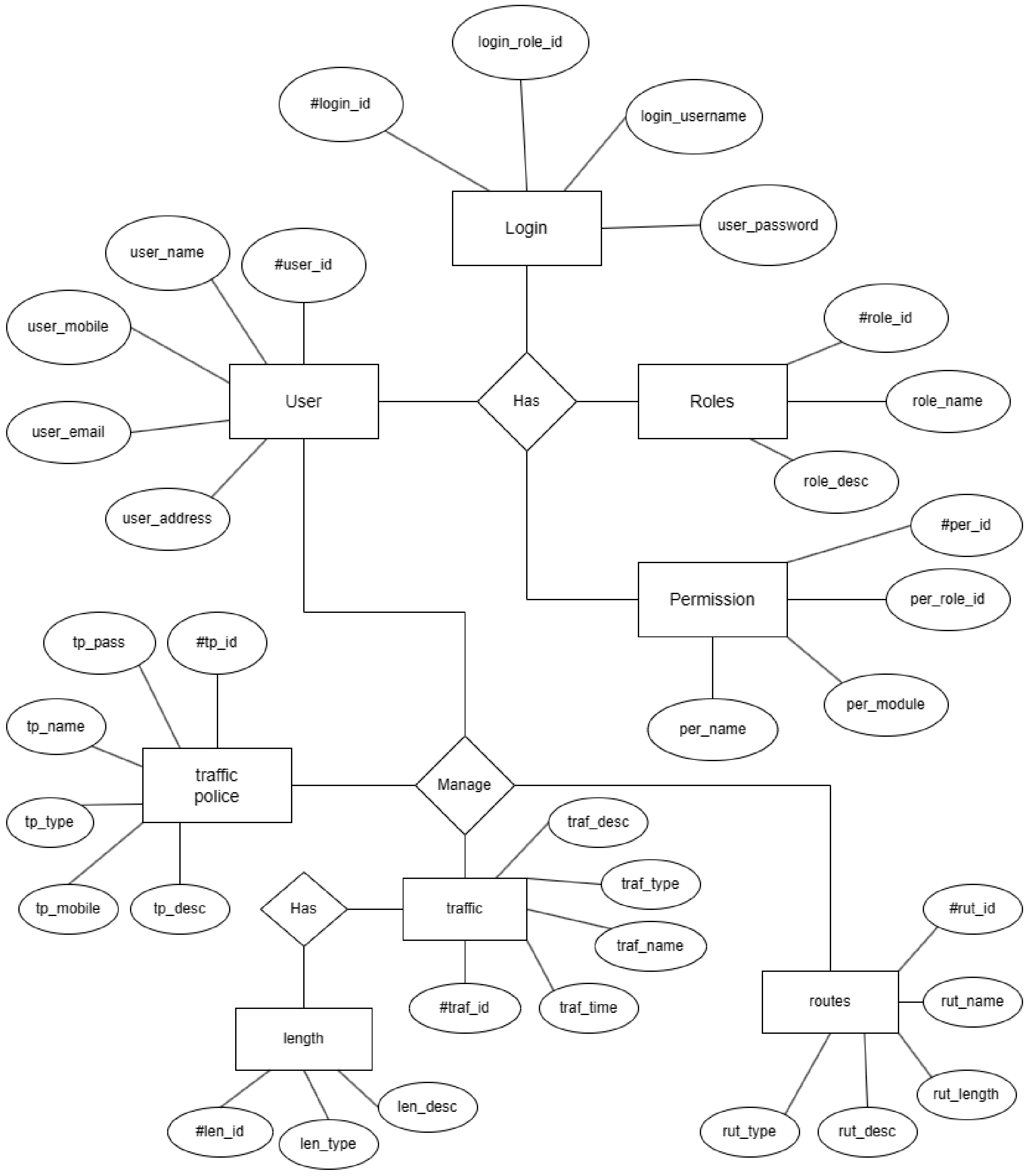

Figure 3: ER Diagram (Group Study)

3.3.3 System Flow Diagram

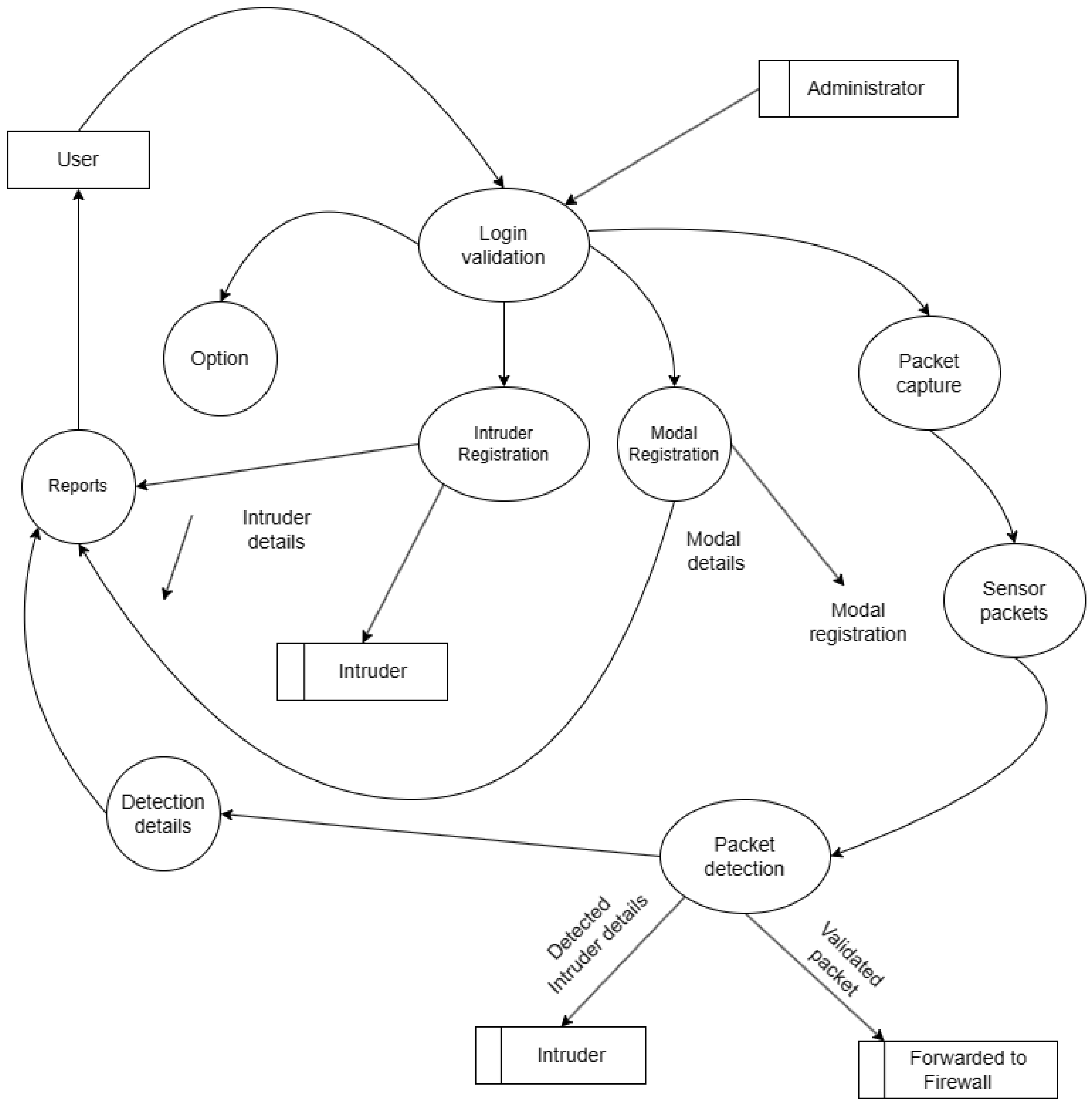

Figure 4 : System Flow Diagram (Group Study)

3.3.4 Use Case Diagram

A use case diagram is a visual representation that illustrates the interactions between actors and use cases within a system, commonly used in software development to capture functional requirements.

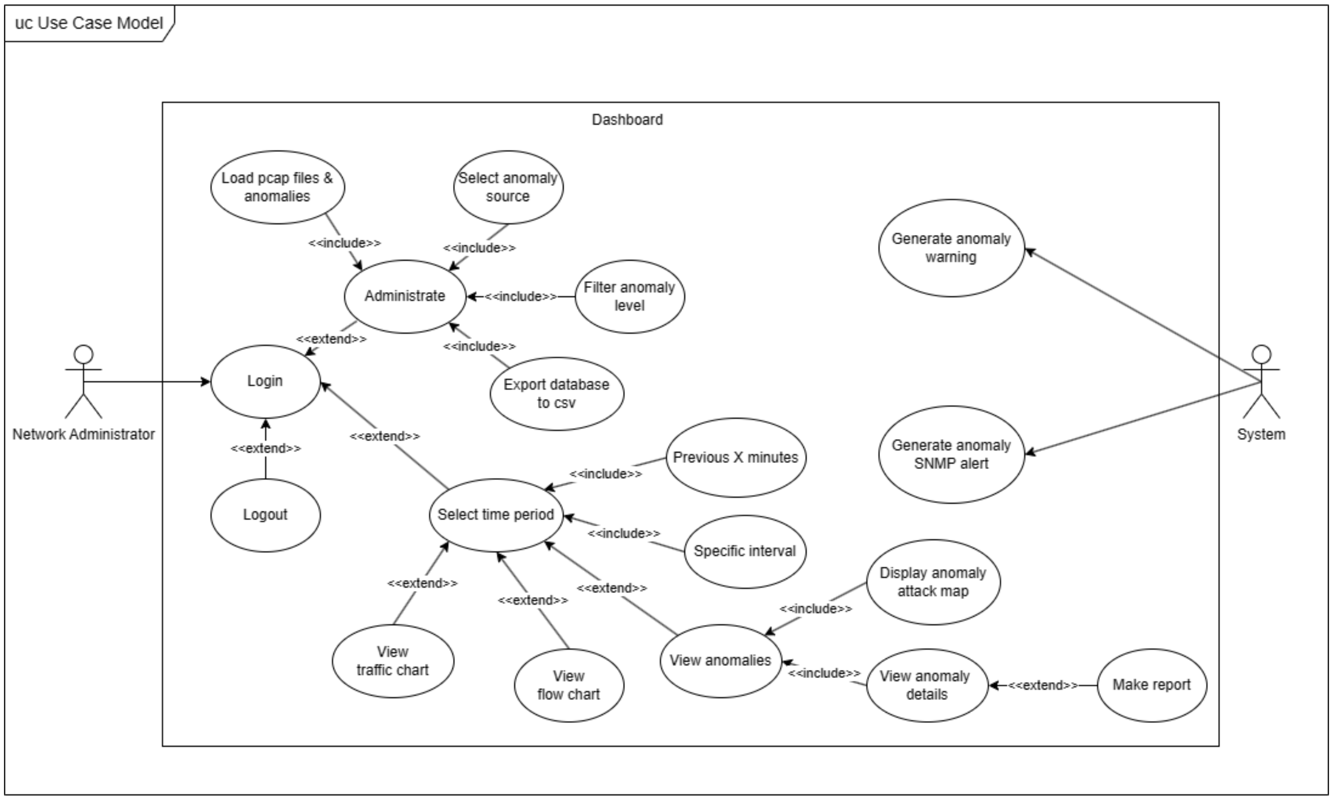

Figure 5: Use Case Diagram (Group Study)

3.3.5 System Flow Chart

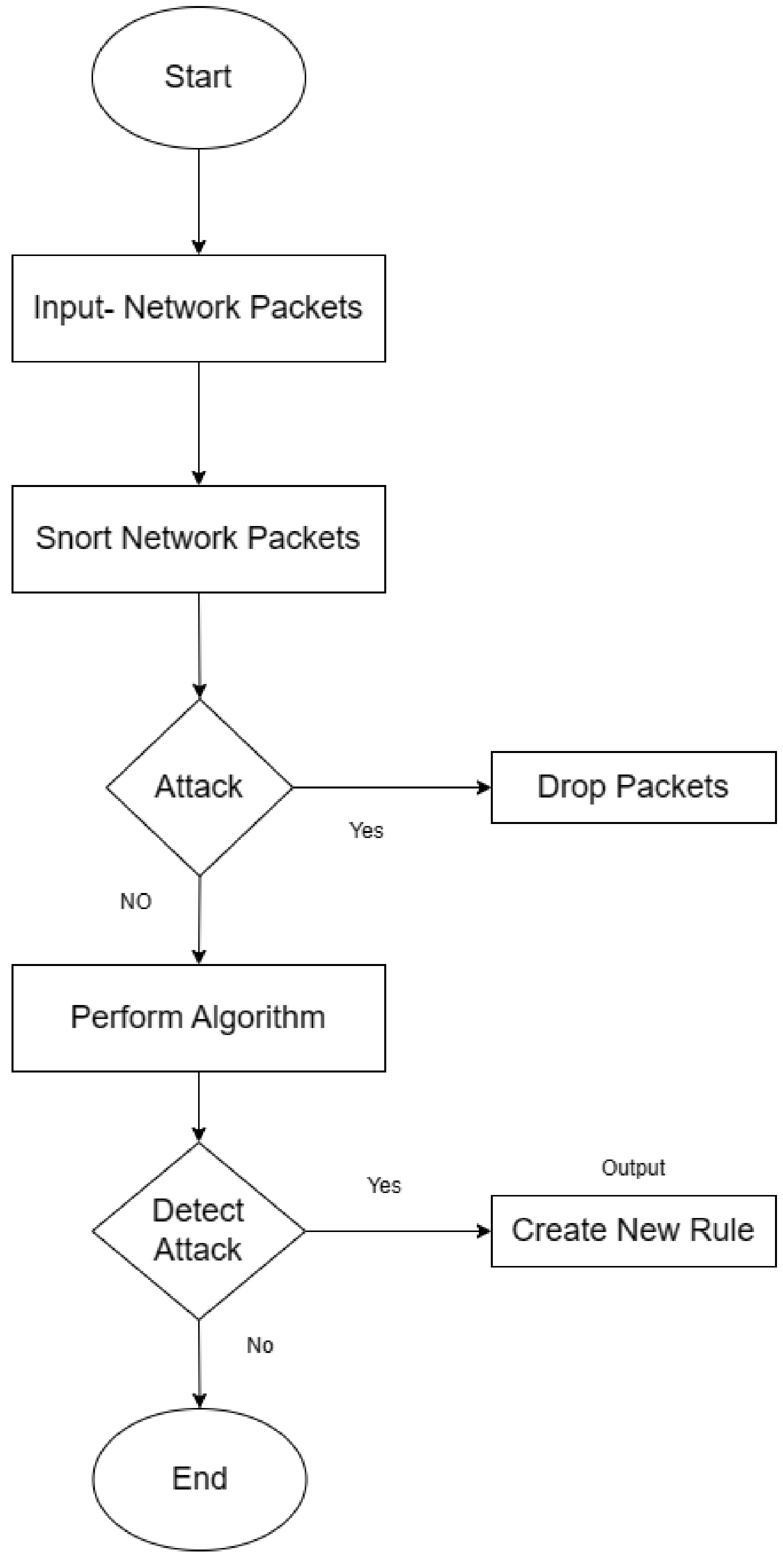

Figure 6: Flow Chart of NIDS (Group Study)

3.4 Software Development and Hardware Required

Minimum Hardware Requirements:

Processor (CPU):

Quad-core processor or equivalent: Adequate processing power is essential for real-time packet analysis and anomaly detection.

Memory (RAM):

8 GB or more: Sufficient RAM is crucial for efficiently storing and processing data during network monitoring.

Storage:

100 GB or more: Adequate storage space is needed for log files, captured packets, and any database used for storing detection data.

Network Interface Cards (NICs):

Multiple Gigabit Ethernet NICs: To capture and analyze network traffic effectively, having multiple NICs can be beneficial, especially if deploying on a high-traffic network.

Minimum Software Requirements:

Operating System:

Linux distribution (e.g., Ubuntu, CentOS, Debian): Linux is commonly preferred for NIDS implementations due to its stability and support for open-source tools.

Packet Capture and Analysis:

Wireshark or tcpdump: For capturing and analyzing network packets.

Scapy: Python library for crafting and analyzing packets.

Intrusion Detection Software:

Snort or Suricata: Open-source intrusion detection systems with signature-based and anomaly-based detection capabilities.

3.5 Algorithm Selection

Choosing the right algorithms is crucial for developing an effective Network Intrusion Detection System (NIDS). Our selection process considered performance, scalability, complexity, and interpretability. We evaluated three machine learning algorithms: Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree. Additionally, we used Recursive Feature Elimination (RFE) with a Random Forest classifier for feature selection.

Feature Selection with RFE and Random Forest: Before selecting the primary algorithms, we needed to identify the most relevant features from our dataset. Using Recursive Feature Elimination (RFE) with a Random Forest classifier, we systematically removed less important features to improve the efficiency and performance of our models. This step helped us focus on the most significant attributes of the network traffic data, making our models more effective [26].

Logistic Regression: Logistic Regression was chosen for its simplicity and effectiveness in binary classification problems, which fits our task of distinguishing between normal and anomalous network traffic. This algorithm predicts the probability of a binary outcome based on one or more predictor variables. It's easy to implement and interpret, providing a good baseline for comparison with other, more complex models. However, Logistic Regression may not perform as well on datasets with complex, non-linear relationships [27].

Decision Tree: Decision Trees are versatile and powerful models capable of handling both categorical and numerical data. They work by recursively splitting the data based on feature values to make predictions. Decision Trees are easy to interpret and visualize, making them highly valuable for understanding the decision-making process. They handle non-linear relationships between features well, but they can be prone to overfitting, especially when the trees are too deep [28].

Chapter 4: Implementation

The implementation of our Network Intrusion Detection System (NIDS) involved several key steps, including setting up the development environment, preprocessing the data, training the models, and optimizing them for performance.

Development Environment

The project was developed using Python, leveraging several libraries and frameworks to facilitate data manipulation, model training, and evaluation. Key libraries used include 'pandas' for data handling, 'scikit-learn' for machine learning models, 'optuna' for hyperparameter optimization and 'matplotlib' for visualization.

Data Preprocessing

Data preprocessing involved cleaning and preparing the dataset for analysis. Steps included loading the data, encoding categorical variables, and standardizing features.

1. **Loading Data**:
    - The dataset was loaded from CSV files into pandas DataFrames.
2. **Label Encoding**:
    - Categorical features were converted to numerical values using label encoding to ensure compatibility with machine learning algorithms.
3. **Feature Selection**:
    - Recursive Feature Elimination (RFE) with a Random Forest classifier was used to identify and retain the top 10 most relevant features.
4. **Standardization**:
    - Features were standardized to have a mean of zero and a standard deviation of one, which is crucial for algorithms that rely on distance measurements.

Model Training

Three machine learning models were trained to detect network intrusions: Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree.

1. **Logistic Regression**:
    - Chosen for its simplicity and effectiveness in binary classification problems.
    - Model trained using standardized features.
2. K-**Nearest Neighbors (KNN)**:
    - Selected for its robustness in handling non-linear data.
    - Hyperparameters optimized using Optuna to determine the best number of neighbors.
3. **Decision Tree**:
    - Chosen for its interpretability and flexibility.
    - Model parameters tuned to prevent overfitting and ensure optimal performance.

Hyperparameter Optimization

Optuna was used to optimize the hyperparameters of the KNN and Decision Tree models. This involved conducting multiple trials to find the most effective hyperparameter settings, thus enhancing the models' performance.

Model Integration

The models were integrated into the NIDS framework, where they work together to analyze network traffic and detect anomalies. The system continuously monitors network packets, processes them using the trained models, and flags any suspicious activity for further investigation.

Chapter 5: Testing and Performance

Thorough testing and validation are crucial to ensure the effectiveness and reliability of the Network Intrusion Detection System. This section outlines the methods used to evaluate the performance of the models and the overall system.

Cross-Validation

Cross-validation was employed to assess the robustness of the models. A 10-fold cross-validation technique was used, where the dataset was split into 10 parts. Each model was trained in 9 parts and tested on the remaining part, rotating the test set through all 10 parts to ensure comprehensive evaluation.

Evaluation Metrics

The models were evaluated using several key metrics:

- **Accuracy:** The proportion of correctly classified instances out of the total instances.
- **Precision:** The proportion of true positive instances out of the instances predicted as positive.
- **Recall:** The proportion of true positive instances out of the actual positive instances.
- **F1-Score:** The harmonic mean of precision and recall, providing a single metric that balances both concerns.

Confusion Matrix

Confusion matrices were used to provide a detailed breakdown of the models' performance, showing the number of true positives, true negatives, false positives, and false negatives.

Performance

The Decision Tree model outperformed the others, achieving the highest accuracy and F1-Score. This model was selected as the primary detector for the NIDS.

Chapter 6: Result and Discussion

6.1 Model Performance

The performance of the three machine learning models—Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree—was evaluated based on accuracy, precision, recall, and F1-score. The results are summarized below:

1. **Logistic Regression**:
    - **Accuracy:** 93.88%
    - **Precision:** 93.57%
    - **Recall:** 95.65%
    - **F1-Score:** 0.94
2. **K-Nearest Neighbors (KNN)**:
    - **Accuracy:** 98.02%
    - **Precision:** 98.39%
    - **Recall:** 98.30%
    - **F1-Score:** 0.98
3. **Decision Tree**:
    - **Accuracy:** 99.42%
    - **Precision:** 99.42%
    - **Recall:** 99.42%
    - **F1-Score:** 0.99

6.2 Discussion

The Decision Tree model outperformed the other models, achieving the highest accuracy and F1-Score. This indicates that the Decision Tree is highly effective at identifying both normal and anomalous network traffic. The high precision and recall values further suggest that the model is reliable, with a low rate of false positives and false negatives.

Logistic Regression provided a solid baseline performance. However, its linear nature may have limited its ability to capture the complex patterns present in network traffic data. Despite this, its high precision and recall demonstrate its effectiveness for simpler intrusion detection tasks.

K-Nearest Neighbors (KNN) showed robust performance with high accuracy and F1-score. Its ability to handle non-linear data made it a strong contender. However, the computational cost associated with KNN, especially with large datasets, was a notable limitation.

The Decision Tree model's performance highlights its capability to handle complex, non-linear relationships in the data. Its interpretability is an added advantage, allowing for easy understanding of the decision-making process. The model's robustness is evident from its top performance across all evaluation metrics.

Overall, the evaluation results indicate that the Decision Tree model is the most suitable for our Network Intrusion Detection System, providing a balance of high performance and interpretability.

6.3 Budget Analysis

Total cost for development of this project is mentioned below:

Name	Cost	Quantity	Sub Total
Printing	250	3	750
Human Resources	3000	4	12000
Internet Package	100	20	2000
Total Cost			NRs.14750

Chapter 7: Future Work

References

[1] J. Fox, "Coblalt," 7 Oct 2022. [Online]. Available: https://www.cobalt.io/blog/biggest-cybersecurity-attacks-in-history. [Accessed 22 Jan 2024].

[2] C. D. L. Ashiku, "Network Intrusion Detection System using Deep Learning," Procedia Computer Science, Bhubanneswar, Odhisha, India, 2021.

[3] N. C. W. P. & R. A. Nasrin Sultana, "doi.org," Peer to Peer Networking and Applications, 12 Dec 2019. [Online]. Available: https://doi.org/10.1007/s12083-017-0630-0. [Accessed 22 Jan 2024].

[4] F. K. a. A. K. S. Zaman, "Machine learning techniques for intrusion detection: An overview," in *International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2013.

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

[6] D. B. M. Jabez, "Intrusion Detection System," Procedia Computer Science, Bhubaneswar, Odhisha, India.

[7] H. A. M. Jimmy Shun, "Network Intrusion Detection System Using Neural Network," in *2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*, 2020.

[8] W. P. &. R. A. Nasrin Sultana. Naveen Chilamkurti, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer to Peer Networking and Applications,* vol. 12, no. https://doi.org/10.1007/s12083-017-0630-0, pp. 493-501, 2019.

[9] R. S. Y. G. S. V. T. Shanbhag, "A high-performance network intrusion detection system," 1999.

[10] T. N. L. L. D. E. O. D. O. B. L. M. R. Patrick Vanin, "A Study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning," *Applied Sciences,* vol. 12, no. https://doi.org/10.3390/app122211752, p. 22, 2022.

[11] SAMPADA BHOSALE, "kaggle," [Online]. Available: https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection.

[12] C. Staff, "Coursera.org," Coursera, 21 Nov 2023. [Online]. Available: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python. [Accessed 22 1 2024].

[13] C. BasuMallick, "SpiceWorks," 20 Mar 2023. [Online]. Available: https://www.spiceworks.com/tech/devops/articles/c-vs-cplus/. [Accessed 22 1 2024].

[14] R. Juillet, "BOCASAY_," 19 Apr 2022. [Online]. Available: https://www.bocasay.com/what-you-need-know-about-java-programming-language/. [Accessed 22 1 2024].

[15] R. R. M. M. S. G. Rohith Raj S, "SCAPY - A powerful interactive packet manipulation program," *IEEE,* no. https://doi.org/10.1109/ICNEWS.2018.8903954, p. 14, 2018.

[16] J. Breeden, "Networkworld," 08 Jun 2022. [Online]. Available: https://www.networkworld.com/article/971145/what-is-wireshark.html. [Accessed 22 1 2024].

[17] K. ,. G. A. S. S. Neha V Sharma, "RETRACTED: Performance Study of Snort and Suricata for Intrusion Detection System," in *IOP Conference Series: Materials Science and Engineering*, 2021.

[18] Ł. Ruczyński, "netguru," 8 Dec 2022. [Online]. Available: https://www.netguru.com/blog/top-machine-learning-frameworks-compared. [Accessed 22 Jan 2024].

[19] S. Ravoof, "Kinsta," 29 Dec 2023. [Online]. Available: https://kinsta.com/blog/postgresql-vs-mysql/. [Accessed 22 Jan 2024].

[20] J. Gopalakrishnan, "Knowi," [Online]. Available: https://www.knowi.com/blog/what-is-elastic-search/. [Accessed 22 Jan 2024].

[21] Simplilearn, "Simplilearn," 16 Jan 2024. [Online]. Available: https://www.simplilearn.com/data-visualization-tools-article. [Accessed 22 Jan 2024].

[22] R. Dickens, "Encryption Consulting," 12 May 2021. [Online]. Available: https://www.encryptionconsulting.com/what-are-encryption-protocols-and-how-do-they-work/. [Accessed 22 Jan 2024].

[23] F. Kane, "Coralogix," 12 Jan 2021. [Online]. Available: https://coralogix.com/blog/a-practical-guide-to-logstash-syslog-deep-dive/. [Accessed 22 Jan 2024].

[24] R. J. Tara Salman, "NETWORKING PROTOCOLS AND STANDARDS FOR INTERNET OF THINGS," vol. 1, no. 9781119173601, pp. 215-238, 2017.

[25] S. H. Perveez, "SimpliLearn," 27 Jul 2023. [Online]. Available: https://www.simplilearn.com/tutorials/git-tutorial/what-is-git. [Accessed 22 Jan 2024].

[26] Y. J.-J. Yin, "springer," 21 January 2023. [Online]. Available: https://doi.org/10.1186/s40537-023-00694-8. [Accessed 05 February 2023].

[27] V. Kanade, "spiceworks," 8 April 2022. [Online]. Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/.

[28] D. L. F. B. N. G. a. N. S. Blockeel H, "frontiers," 15 Dec 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frai.2023.1124553/full. [Accessed 10 July 2023].

[29] S. Das, "Browser Stack," 14 Mar 2023. [Online]. Available: https://www.browserstack.com/guide/top-python-web-development-frameworks. [Accessed 22 Jan 2024].