

**KWARA STATE UNIVERSITY, MALETE**

***The Green University for Community Development and Entrepreneurship***

**Faculty of Information and Communication Technology**

**DEVELOPMENT OF INTRUSION DETECTION SYSTEMS USING CONVOLUTINAL NEURAL NETWORK AND K-NEAREST NEIGBHOR ALGORITHMS**

By

**AFOLAYAN JOHN ADEDAMOLA**

**19/47CS/00965**

**SEPTEMBER 2023**

****

**DEVELOPMENT OF INTRUSION DETECTION SYSTEMS USING CONVOLUTINAL NEURAL NETWORK AND K-NEAREST NEIGBHOR ALGORITHMS**

BY

**AFOLAYAN JOHN ADEDAMOLA**

**19/47CS/00965**

**A RESEARCH PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY, KWARA STATE UNIVERSITY, MALETE,**

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

**SEPTEMBER 2023**

# DECLARATION

I hereby declare that this research project titled, “**DEVELOPMENT OF INTRUSION DETECTION SYSTEMS USING CONVOLUTIONAL NEURAL NETWORK AND K-NEAREST NEIGBHOR ALGORITHMS**” is my own work and has not been submitted by any other person for any degree or qualification at any higher institution. I also declare that the information provided therein are mine and those that are not mine are properly acknowledged.

**AFOLAYAN ADEDAMOLA JOHN \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**NAME OF STUDENT SIGNATURE AND DATE**

**CERTIFICATION**

This is to certify that the research project titled “**DEVELOPMENT OF INTRUSION DETECTION SYSTEMS USING CONVOLUTIONAL NEURAL NETWORK AND K-NEAREST NEIGBHOR ALGORITHMS**” was carried out by “**AFOLAYAN ADEDAMOLA JOHN**”. The project has been read and approved as meeting the requirements for the award of Bachelor of Sciences (B.Sc.) Degree in the Department of Computer Science, Faculty of Information and Communication Technology, Kwara State University, Malete.

**…………………………… ……………………………**

**DR. S.O. ABDULSALAM DATE**

**(SUPERVISOR)**

**…………………………… ……………………………**

**DR. (MRS) R.S. BABATUNDE DATE**

**(HEAD OF DEPARTMENT)**

**…………………………… ……………………………**

**EXTERNAL EXAMINER DATE**

**DEDICATION**

This Project report is dedicated to our Almighty God and to our beloved parents for their spiritual and financial support.

**ACKNOWLEDGEMENTS**

We wish to express our profound gratitude to God Almighty for his guidance and presence throughout our existence.

We are also grateful for our project supervisor, Dr. S.O. ABDULSALAM, for his effort, fatherly attention and tireless corrections throughout the course of this research work, We cannot but acknowledge our Head of Department, Dr. (Mrs) R.S. Babatunde, Prof. K.A. Gbolagade for his fatherly love and mentorship, our level adviser, Dr. Akeem Kadri and all the amazing lecturers in the department of computer science in persons of Dr R.M. Isiaka, Dr. A.N. Babatunde, Mrs S.R. Yusuff, Mrs B.F. Balogun for their continual dedication to our education.

We sincerely thank our parents for their moral, financial, and spiritual support throughout our education at Kwara State University.

Finally, we are grateful to everyone who has supported us in this phase of our lives. May God bless you all.

**TABLE OF CONTENTS**

**FRONT PAGE…………………………………………………………………… I**

**TITLE PAGE……………………………………………………………………. II**

**DECLARATION……………………………………………………………….. III**

**CERTIFICATION……………………………………………………………… IV**

**DEDICATION………………………………………………………………….. V**

**ACKNOWLEDGEMENTS……………………………………………………. VI**

**TABLE OF CONTENTS………………………………………………………. VII**

**LIST OF FIGURES…………………………………………………………….. XI**

**LIST OF TABLES……………………………………………………………… XIII**

**ABSTRACT…………………………………………………………………….. XIV**

**CHAPTER ONE: INTRODUCTION**

* 1. Background of the Study 1
  2. Statement of the Problem 3
  3. Aim and Objectives 3
  4. Scope of the Study 4
  5. Significance of the Study 5

**CHAPTER TWO: LITERATURE REVIEW**

2.1 Intrusion detection system 6

2.2 Types of intrusion detection system 6

2.2.1 Host-based intrusion detection system 6

2.2.2 Network-based intrusion detection system 6

2.3 Machine learning 8

2.3.1 K-Nearest Neighbor Algorithm 8

2.3.2 Neural Networks (NNM) 11

2.3.2.1 Convolutional Neural Network 11

2.3.3 Naive Bayes Model (NBM) 13

2.3.4 Decision Tree (DT) 13

2.3.5 Support Vector Machine (SVM) 14

2.3.6 Fuzzy Logic Algorithm 14

2.3.7 Ensemble Method 14

2.4 Overview of Dimensionality Reduction 15

2.4.1 Feature Selection 17

2.4.2 Feature Extraction 19

**CHAPTER THREE: METHODOLOGY**

3.1 System Approach 22

3.1.1 System Design 24

3.2 Dataset Acquisition and Description 26

3.2.1 Data Preprocessing 27

3.2.1.1 Data Cleaning 27

3.2.1.2 Feature Scaling 28

3.2.1.3 Encoding categorical variable 29

3.2.1.4 Train-Test Split 29

3.3 Feature Selection and Extraction 30

3.4 Development of intrusion detection system using Convolutional Neural Network (CNN) and K-Nearest Neighbor algorithm (KNN). 31

3.4.1 Working of K-nearest Neighbor 32

3.4.2 Working with Convolutional Neural Network 35

3.4.3 Framework for the System 37

3.4.4 Model Evaluation and Comparison 40

3.4.4.1 Model Comparison 41

3.4.5 Testing Phase 42

3.5 Choice of Programming Tools 43

3.6 Performance Evaluation and Interpretation 44

**CHAPTER FOUR: RESULT AND DISCUSSION**

4.1 Experimental Result 46

4.2 Result of Feature Selection and Extraction process 47

4.2.1 Data Acquisition 47

4.2.2 Dataset Preprocessing 48

4.2.2.1 Data Cleaning 49

4.2.2.2 Feature Scaling 51

4.2.2.3 Split Train-Test 51

4.3 Result of the Developed Intrusion Detection System 52

4.3.1 User Interface 52

4.3.2 K-Nearest Neighbor Model 53

4.3.3 Convolutional Neural Network Model 54

4.3.4 Model Evaluation 54

4.3.5 Convolutional Neural Network Model Evaluation 55

4.3.6 K-Nearest Neighbor Model Evaluation 57

4.3.7 Model Comparison 58

**CHAPTER FIVE: CONCLUSION AND RECOMMENDATION**

5.1 Conclusion 60

5.2 Recommendation 61

**REFERENCE**

**APPENDIX**

**LIST OF FIGURES**

**Figures Pages**

3.1 General approach of the system 24

4.1 Imported dataset of the system 48

4.2 Result of data cleaning of the system 50

4.3 Count Plot of outcome and protocol type columns of the system 50

4.4 User Interface of the system 53

4.5 CNN model accuracy per epoch of the system 56

4.6 K-nearest Neighbor Confusion Matrix of the system 58

**LIST OF TABLES**

**Tables Pages**

4.1 Model Evaluation of the system 55

**ABSTRACT**

Machine learning has witnessed a significant rise in applications for network security and automation due to increased data availability, advanced ML techniques, and technological innovations. Specifically, machine learning intrusion detection systems, leveraging various algorithms and optimization techniques, are recognized as a promising solution for addressing security challenges in data and network protection. Intrusion detection system is used to identify unauthorized access and unusual attacks over the secured networks. Over the past years, many studies have been conducted on the intrusion detection system

With the increasing reliance on networked systems and the internet, the security of computer networks and systems has become paramount. Intrusion Detection Systems (IDS) play a crucial role in identifying and mitigating potential threats to network security.

This research presents a novel approach to intrusion detection by combining the strengths of K-Nearest Neighbors (KNN) and Convolutional Neural Networks (CNN). The proposed hybrid IDS aims to enhance detection accuracy while minimizing false positives. The KNN algorithm provides an effective initial filtering mechanism, followed by the CNN algorithm's deep learning capabilities for intricate pattern recognition, it also features statistical comparison of classifier algorithm, datasets being used and some other experimental setups as well as consideration of feature selection step.

**CHAPTER ONE**

**INTRODUCTION**

**1.1 Background of the Study**

In today's interconnected world, with the widespread use of social networking sites and the increasing volume of online business transactions, network security has emerged as a critical concern. Despite efforts to build secure systems, security vulnerabilities continue to surface regularly, necessitating proactive measures by network managers to defend against attacks and ensure swift system recovery. The constant evolution of hacking technologies and the availability of free hacking tools have made preventing cyber-attacks a formidable challenge, as any network can potentially be compromised given sufficient time and resources.

One of the most significant threats to network security is intrusion, including brute force and denial-of-service attacks from external sources or infiltration from within the network. Such intrusions compromise the availability, integrity, and confidentiality of the system, making it essential to adopt a dynamic approach to identify and prevent them. Over the years, intrusion detection systems (IDS) have been developed using various techniques, including statistics, provenance graphs, specifications, and machine learning.

Two distinctive methodologies for modeling intrusion detection are anomaly detection and misuse detection. Anomaly detection assumes that normal user behavior is entirely observable and significantly different from intrusive behavior. It builds a model based on normal user behavior and flags any deviations from this established profile as intrusions. While it can detect novel attacks, it tends to produce a high rate of false alarms, as evolving normal behaviors are sometimes mistaken for attacks. On the other hand, misuse detection uses signatures of well-known threats to identify matches in monitored data, reporting an intrusion when a match is found. While it has minimal false alarms, it only detects attacks with signatures stored in the database.

To address the limitations of both anomaly and misuse detection, hybrid detection techniques, primarily involving machine learning, were introduced. Machine learning can identify hidden patterns in training data and apply these patterns to analyze unknown data, grouping it or mapping it to known categories. Machine learning can be classified as supervised or unsupervised. In supervised learning, algorithms are trained on labeled datasets to create a function capable of mapping instances to classes. In unsupervised learning, the inference engine groups data items based on intra-group and inter-group relationships.

To improve the effectiveness of intrusion detection, this study proposes the development of a network-based IDS that combines Convolutional Neural Network (CNN) and K-Nearest Neighbor (K-NN) algorithms, using the NSL-KDD dataset. This approach aims to overcome the limitations of traditional rule-based IDS systems and leverage machine learning to enhance detection accuracy and reduce false positives. The system will be implemented in Python, utilizing the Scikit-learn library within a Jupyter notebook environment, and trained on preprocessed network traffic data. The primary goals include achieving a high detection rate, minimizing false positives, and ensuring swift response times. The proposed IDS will undergo evaluation in a simulated network environment to assess its effectiveness in detecting potential attacks and contribute to the development of advanced security systems.(Chimphlee & Chimphlee. 2023), (Unnisa A et al. 2022).

**1.2 Statement of the Problem**

Intrusion detection systems (IDSs) are one of the promising tools for protecting data and networks; many classification algorithms, such as neural network (NN), Naive Bayes (NB), decision tree (DT), and support vector machine (SVM) have been used for IDS in the last decades. However the current IDSs face a few weaknesses, like high misleading positive rates, slow reaction times, and trouble in recognizing obscure dangers. These constraints can result in missed attacks and an expanded responsibility for security groups.

To address these weaknesses, further developed IDS that use machine learning algorithm can be created. Such an IDS can lessen bogus up-sides, increment location exactness, and further develop reaction times to likely attacks. The proposed arrangement includes utilizing machine learning algorithm to investigate network traffic and recognize designs that demonstrate pernicious movement. The framework will be prepared on huge datasets of known attacks to guarantee it can precisely distinguish likely dangers. The proposed IDS will work on the proficiency and exactness of the security framework, empowering faster reactions to expected dangers and decreasing the responsibility on security groups. (Nizomova. 2023).

**1.3 Aim and Objectives**

The aim of this project is to develop an intrusion detection system with the use of convolutional neural network and k-nearest neighbor algorithm to detect potential security threats accurately and the objectives of the project are:

1. Perform feature selection and extraction process on NSL-KDD intrusion detection dataset obtained from Kaggle repository using mutual information and linear discriminant analysis algorithms respectively.
2. Develop two intrusion detection systems based on the reduced dataset obtained in (I) using convolutional neural network and K-nearest neighbor classification methods.
3. Evaluate the performance of the two intrusion detection systems developed in (II) in terms of accuracy, f1 score, precision, and recall.

**1.4 Scope of the Study**

The NSL-KDD dataset, comprising both normal and various instances, will be obtained from Kaggle. This acquired and processed data will serve as the training dataset for the convolutional neural network and K-nearest neighbor algorithm. The algorithm will generate model based on this training process using python programming language as the tool. The performance of this model will then be evaluated to determine their effectiveness in detecting intrusions.

Upon identifying the model with optimal performance, it will be deployed to construct a user friendly Graphical User Interface (GUI) , this GUI will serve as the interface for the detection system, enabling users to easily interact with the system and effectively detect potential threats and attacks within the network.

This project focuses on the development and evaluation of an Intrusion Detection System using Convolutional Neural Network and K-nearest Neighbor algorithms. The study will utilize the NSL-KDD dataset. The performance evaluation will be based on metrics such as accuracy, precision, recall, and F1-score. (Hnamte & Hussain. 2023).

**1.5 Significance of the Study**

The proposed IDS bring forth numerous advantages, encompassing the alleviation of security teams' workload, enhancement of detection accuracy, and prompt response to potential attacks. By leveraging these IDS, organizations can effectively mitigate the risk of data breaches and minimize the impact caused by successful attacks. Moreover, this project holds the potential to contribute to the advancement of sophisticated security systems, empowering organizations to safeguard against emerging cyber threats. (Miao et al. 2023).

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 Intrusion Detection System**

Intrusion Detection System refers to the process of monitoring a computer system or network for unauthorized access or malicious activities and taking appropriate actions to prevent or mitigate security breaches. It's a crucial component of cyber-security and helps organizations protect their digital assets and sensitive information. (Zhang, 2023)

**2.2 Types of Intrusion Detection Systems**

There are two main types of intrusion detection systems (IDS):

**2.2.1 Host-Based Intrusion Detection System (HIDS)**:

HIDS operates on individual hosts or devices within a network. It monitors the activity on a specific computer or server and looks for signs of unauthorized access or malicious activity. HIDS typically works by analyzing log files and system resources to detect abnormal behavior. When it identifies a potential intrusion, it can trigger alerts or take predefined actions like isolating the compromised host from the network.

* + 1. **Network-Based Intrusion Detection System (NIDS)**:

NIDS, on the other hand, monitors network traffic as it passes through a network segment or boundary. It analyzes packets of data to identify patterns or signatures associated with known attacks. When it detects suspicious network activity, it generates alerts, which can lead to further investigation and mitigation.

Intrusion detection systems can be further classified into two detection methods:

1. Signature-based Detection: This method involves comparing network traffic or system activity against a database of known attack patterns or signatures. If there's a match, the IDS generate an alert. Signature-based IDS is effective at identifying known threats but may miss zero-day attacks (new, previously unknown threats).
2. Anomaly-based Detection: Anomaly-based IDS establishes a baseline of normal behavior for a system or network. It then continuously monitors for deviations from this baseline. When it detects behavior that is significantly different from the norm, it raises an alert. Anomaly-based IDS can detect previously unknown attacks but may also generate false positives if legitimate changes occur.

Intrusion Detection Systems can be implemented using dedicated hardware appliances or as software solutions running on servers or security appliances. Additionally, some modern Intrusion Detection Systems incorporate machine learning and artificial intelligence techniques to enhance their ability to detect complex and evolving threats.

Intrusion Detection is often paired with Intrusion Prevention Systems (IPS) to not only detect but also actively block or mitigate suspicious or malicious activity in real-time. Together, IDS and IPS form a crucial part of an organization's overall cyber-security strategy to protect against a wide range of cyber threats. (Nguyen & Kashef, 2023).

**2.3 Machine Learning Algorithms**

Machine learning is a subset of AI that includes all the methods and algorithms which enables the machine to learn automatically using mathematical models in order to extract useful information from the large datasets. The most common ML (also called Shallow Learning) algorithms used for IDS are Decision trees, K-Nearest Neighbor (KNN), NNs (Neural Networks), Support Vector Machine (SVM), K-Mean Clustering, Ensemble methods. (Bansal et al. 2022).

**2.3.1 K-Nearest Neighbor Algorithm**

The k-Nearest Neighbors (k-NN) algorithm is a supervised machine learning algorithm used for classification and regression tasks. It is a simple and intuitive method that makes predictions based on the similarity of data points. Here's an overview of how the k-NN algorithm works:

1. Initialization:

* Choose the number of neighbors (k): This is a hyper-parameter you need to specify in advance. It determines how many nearest neighbors will influence the prediction

1. Data Collection and Preprocessing:

* Gather a labeled dataset: The dataset should contain examples with known classes (for classification) or known target values (for regression).
* Preprocess the data: This may involve handling missing values, normalizing or scaling features, and splitting the data into training and testing sets.

1. Prediction Process:

* For a new, unseen data point that you want to classify or predict, the k-NN algorithm follows these steps:

1. Calculate distances: Measure the distance (similarity) between the new data point and all data points in the training dataset. Common distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance.
2. Select the k-nearest neighbors: Identify the k training examples with the shortest distances to the new data point.
3. Classification:

* For classification tasks (predicting a class or category), determine the class of the new data point based on a majority vote among its k-nearest neighbors. The class that appears most frequently among the neighbors is assigned as the predicted class.
* In binary classification, this means that if k is odd, there will be a clear majority vote. If k is even, a tie-breaking strategy may be needed.

1. Regression:

* For regression tasks (predicting a continuous value), calculate the average (or weighted average) of the target values of the k-nearest neighbors. This average is assigned as the predicted value for the new data point.

1. Model Evaluation:

* Assess the performance of the k-NN model using evaluation metrics such as accuracy (for classification), Mean Absolute Error (MAE) or Mean Squared Error (MSE) (for regression), and cross-validation techniques.

1. Hyper-parameter Tuning:

* Experiment with different values of k to find the optimal number of neighbors that provides the best performance on your dataset. This is typically done through techniques like cross-validation.

Key Considerations:

* The choice of the distance metric and the value of k significantly impact the model's performance.
* K-NN is sensitive to the scaling of features, so it's essential to preprocess data appropriately.
* It's also crucial to handle ties when determining the class in classification tasks.
* The k-NN algorithm can be computationally expensive, especially for large datasets, as it requires calculating distances to all training examples.

K-NN is a non-parametric, instance-based learning algorithm, which means it doesn't make strong assumptions about the underlying data distribution. It's versatile and easy to understand, making it a good choice for some classification and regression tasks, especially when dealing with small to moderate-sized datasets. However, it may not perform well when features are highly correlated or irrelevant, and it can be computationally demanding in high-dimensional spaces. (Nizomova, 2023).

**2.3.2 Neural Networks (NNs)**

In the context of Intrusion Detection Systems (IDS), Neural Networks (NNs) can be used as a broader term encompassing different types of neural network architectures, including Convolutional Neural Networks (CNNs) and other variants like Recurrent Neural Networks (RNNs) or Multilayer Perceptrons (MLPs). Neural Networks (NNs) have been widely used in IDS for their ability to learn complex patterns and detect anomalies in network traffic. They can effectively capture both spatial and temporal dependencies in data, making them suitable for detecting intrusions.

**2.3.2.1 Convolutional Neural Networks (CNNs)**

CNNs are particularly effective in capturing spatial dependencies and extracting features from structured data such as images or, in the case of IDS, network packets. CNNs utilize convolutional layers to automatically learn and extract relevant features from the input data. They can be applied to network traffic analysis by considering network packets as structured data and using CNNs to identify patterns associated with intrusions. In the context of an Intrusion Detection System (IDS), "CNN" can refer to the use of Convolutional Neural Networks for enhancing the detection and classification of network intrusions or anomalies. Here's how CNNs can be applied in an IDS:

1. Feature Extraction: Convolutional Neural Networks are excellent at automatically extracting hierarchical features from data. In the case of network traffic data, these networks can be trained to learn relevant patterns and features from packet headers or payloads. These learned features can capture information about the structure and content of network packets.
2. Traffic Classification: Once a CNN is trained on a dataset of network traffic, it can be used to classify incoming traffic as either normal or potentially malicious. The network can learn to recognize patterns associated with different types of network attacks, such as denial-of-service (DoS) attacks, port scanning, or intrusion attempts.
3. Real-Time Analysis: CNN-based IDS can operate in real-time, continuously monitoring network traffic and making predictions on whether the traffic is benign or suspicious. This is valuable for quickly identifying and responding to potential threats.
4. Anomaly Detection: CNNs can also be used for anomaly detection. Instead of focusing solely on known attack patterns, they can learn the normal behavior of a network and raise alerts when they detect deviations from this normal behavior. This is particularly useful for detecting novel or zero-day attacks.
5. Reducing False Positives: CNNs can help reduce false positives in intrusion detection by providing a more sophisticated and adaptive approach to traffic analysis. Traditional rule-based IDS systems often generate a high number of false alarms, whereas CNNs can better distinguish between normal and malicious traffic patterns.
6. Scalability: CNN-based IDS can be scaled to handle large volumes of network traffic, making them suitable for enterprise-level network security.

It's important to note that while CNNs can be a valuable tool in intrusion detection, they are often used in conjunction with other techniques and algorithms to build comprehensive IDS solutions. Network security is a complex field, and multiple approaches, including signature-based detection, anomaly detection, and machine learning, are often combined to create effective intrusion detection systems. (Li. 2020), (Bayguinov et al. 2020).

**2.3.3 Naive Bayes Model (NBM)**

Intrusion Detection Systems (IDS) commonly employ various machine learning algorithms, including Convolutional Neural Networks (CNNs) and K-Nearest Neighbor (KNN) algorithms. However, Naive Bayes Model (NBM) is also a viable approach for IDS, particularly for certain types of data and classification tasks.

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem and assumes the independence of features. Despite its simplicity, Naive Bayes has been successful in many applications, including text classification and spam filtering. (Xashimov & Khaydarova. 2023), (Jumamuratova. 2022).

**2.3.4 Decision Tree (DT)**

Decision trees are another popular machine learning algorithm commonly used in Intrusion Detection Systems (IDS). Although Decision trees are not directly related to Convolutional Neural Networks (CNNs) or K-Nearest Neighbor (KNN) algorithms, they are valuable for their interpretability and ability to handle categorical and numerical data. (Bayimbetova. 2022) (Xashimov & Khaydarova. 2023).

**2.3.5 Support Vector Machine (SVM)**

Support Vector Machine (SVM) is a popular machine learning algorithm that can be effectively applied in Intrusion Detection Systems (IDS). While SVM is not directly related to Convolutional Neural Networks (CNNs) or K-Nearest Neighbor (KNN) algorithms, it offers robust classification capabilities and is well-suited for handling high-dimensional data. (K K A et al. 2020) (Xashimov & Khaydarova. 2023)

**2.3.6 Fuzzy Logic Algorithm**

Fuzzy Logic Algorithm (FLA) is another approach that can be applied in Intrusion Detection Systems (IDS), alongside Convolutional Neural Networks (CNNs) and K-Nearest Neighbor (KNN) algorithms. Fuzzy logic allows for handling uncertainty and imprecision in data, making it suitable for modeling complex and ambiguous concepts. (K K A et al. 2020)

**2.3.7 Ensemble Method**

Ensemble methods in intrusion detection involve combining multiple intrusion detection models or techniques to enhance the overall performance, accuracy, and robustness of the intrusion detection system (IDS). Just as ensemble methods are used in various machine learning tasks, they can also be applied to intrusion detection to improve its effectiveness. (Gupta et al. 2022).

* 1. **Overview Of Dimensionality Reduction**

Dimensionality reduction techniques play a significant role in intrusion detection systems (IDS) by addressing the challenges associated with high-dimensional data. In intrusion detection, data often contains numerous features, making it computationally expensive and potentially prone to over-fitting. An overview of dimensionality reduction in intrusion detection includes;

1. High-Dimensional Data: Intrusion detection datasets can be high-dimensional, with many features describing network traffic or system behavior. These features can lead to increased computational complexity and a risk of over-fitting when building detection models.
2. Dimensionality Reduction Goals: The primary goal of dimensionality reduction in IDS is to reduce the number of features while preserving relevant information. This can enhance the efficiency of the detection process and improve the model's generalization to new and unseen threats.
3. Principal Component Analysis (PCA): PCA is a widely used dimensionality reduction technique that transforms the original features into a new set of orthogonal variables called principal components. These components capture the most significant variance in the data, allowing for dimensionality reduction while retaining as much information as possible.
4. T-Distributed Stochastic Neighbor Embedding (t-SNE): t-SNE is another technique used in IDS to reduce dimensionality while preserving data structure. It is particularly useful for visualization purposes, as it maintains the relative distances between data points in lower-dimensional space. (Maabreh et al. 2022).
5. Linear vs. Non-linear Methods: Intrusion detection datasets may exhibit non-linear relationships between features. While PCA is a linear technique, non-linear dimensionality reduction methods like Isomap or locally linear embedding (LLE) can be applied when non-linearity is present.
6. Benefits: Dimensionality reduction not only improves computational efficiency but also helps mitigate the "curse of dimensionality," which can lead to increased model complexity and decreased performance. It can also aid in feature selection by identifying the most informative attributes.
7. Considerations: When applying dimensionality reduction in IDS, it's crucial to strike a balance between reducing dimensionality and preserving critical information. Careful evaluation of the impact on detection performance is necessary to ensure that the reduction process doesn't lead to the loss of important threat indicators.
8. Integration with Detection Models: Dimensionality reduction is typically applied as a preprocessing step before training intrusion detection models like K-Nearest Neighbors (KNN), Random Forests, or Neural Networks. It helps these models operate more efficiently on reduced-dimensional data.

Overall, dimensionality reduction techniques are valuable tools in intrusion detection to manage the complexity of high-dimensional data and improve the efficiency and effectiveness of detection models. The choice of technique should align with the specific characteristics and requirements of the IDS and the dataset being used. (Maabreh et al. 2022).

* + 1. **Feature Selection**

Feature selection is a crucial step in the data preprocessing phase of building an Intrusion Detection System (IDS). The main objective of feature selection is to choose the most relevant and informative features from the dataset while discarding irrelevant or redundant ones. By reducing the number of features, feature selection helps improve the efficiency, interpretability, and generalization of the IDS model. Here are details about feature selection techniques:

1. Filter Methods: Filter methods evaluate the relevance of features based on their individual characteristics, independent of the model used for classification. Common filter methods include:

* Pearson Correlation: Calculate the correlation coefficient between each feature and the target class. Features with higher absolute correlation values are considered more relevant.
* Information Gain: Measure how much information a feature provides about the target class. Features with higher information gain are considered more informative.
* Chi-Square Test: Assess the independence between each feature and the target class using the chi-square statistic.

1. Wrapper Methods: Wrapper methods use the performance of the learning algorithm itself to evaluate the usefulness of features. They involve repeatedly training the model with different subsets of features and selecting the subset that yields the best performance. Common wrapper methods include Recursive Feature Elimination (RFE) and Sequential Feature Selection (SFS/SBS).
2. Embedded Methods: Embedded methods incorporate feature selection as part of the model training process. These methods automatically select the most important features during model training. Examples include L1 regularization (Lasso) and L2 regularization (Ridge) in linear models, which penalize less important features.
3. Tree-Based Feature Selection: Tree-based algorithms, such as decision trees and random forests, can provide feature importance scores during training. Features with higher importance scores are more relevant to the classification task. Principal Component Analysis (PCA): PCA is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated features (principal components). The new components are ordered by their importance in explaining the variance in the data. Selecting the top "k" principal components can effectively reduce the number of features while preserving most of the information.
4. Forward Selection: Forward selection is a stepwise feature selection technique that starts with an empty set of features and iteratively adds the most relevant feature at each step until a stopping criterion is met. It is a computationally expensive method but can yield good results.
5. Backward Elimination: Backward elimination is another stepwise feature selection technique that starts with all features and iteratively removes the least relevant feature at each step until a stopping criterion is met.

Feature Importance from Model:

Some machine learning models, such as decision trees, random forests, and gradient boosting, provide feature importance scores as part of their training process. These scores can be used to identify the most important features for the classification task. It's important to note that the choice of feature selection technique depends on factors such as the size of the dataset, the number of features, the characteristics of the data, and the computational resources available. Experimenting with different feature selection methods and evaluating their impact on the IDS's performance can help identify the best approach for the specific project. (Fahimifar et al. 2022).

* + 1. **Feature Extraction**

Feature extraction in intrusion detection refers to the process of selecting, transforming, or creating relevant features from raw data to represent network traffic or system behavior effectively. It plays a critical role in building accurate and efficient intrusion detection models. Here's an overview of feature extraction in intrusion detection:

1. Raw Data Sources: Intrusion detection systems collect raw data from various sources, such as network traffic logs, packet captures (PCAPs), or system logs. These data sources contain a wealth of information but are often too voluminous and complex to use directly for detection.
2. Feature Engineering: Feature extraction involves engineering a set of features from raw data that can serve as input to intrusion detection models. This process is crucial for translating the complexity of raw data into a format that machine learning algorithms can understand.
3. Types of Features:

Features in intrusion detection can be categorized into several types:

* Statistical Features: These include basic statistics like mean, median, standard deviation, and variance of data attributes. They provide insights into the distribution and variability of data.
* Temporal Features: Temporal features capture patterns over time, such as frequency of events, time intervals between events, and seasonality. They help detect attacks that exhibit specific temporal behaviors.
* Frequency-Based Features: These features represent the occurrence or frequency of certain events, such as the number of failed login attempts, port scans, or packet types.
* Protocol-Specific Features: Different network protocols (e.g., HTTP, FTP, SMTP) have unique characteristics. Protocol-specific features capture attributes relevant to specific protocols, facilitating protocol-based anomaly detection.
* Content-Based Features: These features analyze the content of network packets or logs, such as payload contents, signatures, or keywords. They are particularly useful for detecting application-layer attacks.
* Statistical Traffic Features: Features related to network traffic, including flow statistics (e.g., flow duration, bytes transferred, packets sent), can provide insights into network behavior.

1. Dimensionality Reduction: Intrusion detection datasets can be high-dimensional due to the large number of features. Dimensionality reduction techniques like Principal Component Analysis (PCA) or t-distributed Stochastic Neighbor Embedding (t-SNE) can help reduce the number of features while preserving relevant information.
2. Feature Selection: Feature selection methods are employed to choose the most informative features for intrusion detection. Techniques like mutual information, feature importance ranking, and recursive feature elimination help identify features with the highest predictive power.
3. Domain Knowledge: Domain experts often play a crucial role in feature extraction. Their knowledge of network protocols, system behavior, and attack patterns can guide the selection and creation of relevant features.
4. Normalization and Scaling: Features are typically normalized or scaled to ensure they are on a consistent scale. This step helps prevent certain features from dominating the modeling process due to their larger magnitude.
5. Data Preprocessing: Feature extraction is part of the broader data preprocessing phase, which includes tasks like handling missing values, encoding categorical data, and data balancing.
6. Integration with Models: Extracted features serve as input to intrusion detection models, including machine learning algorithms like K-Nearest Neighbors (KNN), Random Forests, Support Vector Machines (SVM), or deep learning models. (Xashimov & Khaydarova. 2023)

**CHAPTER THREE**

**METHODOLOGY**

**3.1 System Approach**

The security and integrity of computer networks and information systems depend heavily on intrusion detection. The creation of efficient intrusion detection models becomes crucial given the constantly changing landscape of cyber-security threats. This chapter outlines the approach used in this study to detect intrusions on the NSL-KDD dataset using two well-known algorithms, K-Nearest Neighbor and Convolutional Neural Network. An extensive collection of labeled network traffic data is provided by the NSL-KDD dataset, a benchmark that is frequently utilized in intrusion detection research. This dataset provides a thorough representation of real-world network scenarios and includes both normal and invasive activity, making it appropriate for training and assessing intrusion detection models.

The primary aim of this chapter is to provide a comprehensive account of the steps involved in building and assessing intrusion detection models. K-Nearest Neighbor (KNN) and Convolutional Neural Network (CNN) delivers resilience and efficiency, especially in high-dimensional feature spaces, as a result of its independence assumptions.

The NSL-KDD dataset's history, make up, and structure are all thoroughly discussed in the beginning of the chapter. Any preparation operations carried out on the dataset, such as addressing missing values, encoding categorical variables, and feature scaling, are also described.

The feature selection procedure, which involves identifying unimportant features for model training, is described after the dataset preparation step. The use of informative features makes sure that the intrusion detection models are able to recognize important trends in the data, improving their detection skills.

A presentation of the KNN’s algorithm's application to intrusion detection on the NSL-KDD dataset follows. The model architecture, learning process, and any relevant hyper-parameters are described in detail, along with any implementation-related difficulties that may have arisen.

## The chapter also explores the model training stage, when the dataset is divided into training and testing sets to assess the performance of the models. The generalizability of the KNN model is validated using cross-validation techniques to increase the dependability of the results.

## Finally, the chapter discusses the assessment criteria used to evaluate the effectiveness of K-Nearest Neighbor, Convolutional Neural Network on the NSL-KDD dataset. The importance of accuracy, sensitivity, and specificity is stressed. These metrics offer important insights into the models' overall performance and their capacity to distinguish between legitimate and invasive network activity. (Bhattacharya. 2023).

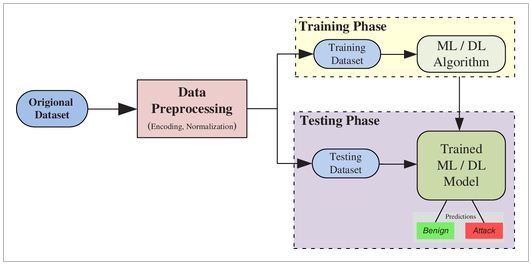


Fig 3.1 General approach of the system

**3.1.1 System Design**

1. Input Data: The IDS will take network traffic data as input. The NSL-KDD dataset used in this study was gotten from Kaggle but can be collected from various sources, such as network packets or flow records, and should include features like source and destination IP addresses, port numbers, protocol type, duration, number of packets, etc.
2. Preprocessing Module: The raw input data needs to be preprocessed before feeding it to the KNN classifier. The preprocessing module will handle tasks like:

* Handling missing values: Decide on an appropriate strategy (imputation, removal) for missing data
* Feature scaling: Normalize the features to ensure they are on a similar scale. Feature encoding: Convert categorical features to numerical values using techniques like one-hot encoding.
* Data splitting: Divide the preprocessed data into training and testing sets. K-Nearest Neighbors (KNN) Algorithm: Implement the KNN algorithm or use libraries like scikit-learn to perform classification. Choose a suitable distance metric (e.g., Euclidean distance) for calculating the distance between data points. Determine the value of "K" (number of neighbors) through hyper-parameter tuning.

1. Training Module: The training module will take the preprocessed training data and feed it to the KNN classifier to train the model. The trained KNN model will learn the patterns in the training data and be ready for classification.
2. Testing and Evaluation Module: The preprocessed testing data will be provided to the trained KNN model for classification. The output of the model will be compared with the actual labels to evaluate the IDS's performance. Calculate various performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix to assess the effectiveness of the IDS.
3. Hyper-parameter Tuning: Implement techniques like cross-validation or grid search to find the optimal value of hyper-parameters (e.g., K, distance metric). This step helps improve the model's performance and generalization capabilities.
4. Model Update Module: Periodically update the KNN model to incorporate new data and adapt to changes in the network environment. This step ensures that the IDS stays effective against emerging threats. Deployment and Real-Time Monitoring: Deploy the trained KNN model in the production environment to monitor network traffic in real-time. Set up a continuous monitoring system to track the IDS's performance and detect any potential issues or anomalies.
5. User Interface (Optional): Design a user-friendly interface to interact with the IDS. The interface can provide visualizations of network traffic, detected intrusions, and overall system status.
6. Alerting System: Implement an alerting mechanism to notify network administrators or security personnel in case of any detected intrusions or suspicious activities. The system can generate alerts via email, SMS, or integration with other monitoring tools.

Documentation and Maintenance: Document the system design, data preprocessing steps, model implementation details, hyper-parameter choices, and evaluation results. Provide clear instructions on how to deploy, use, and maintain the IDS system. Regularly monitor and maintain the system to ensure it stays effective and up-to-date. (K K A et al. 2020).

**3.2 Dataset Acquisition and Description**

The NSL-KDD dataset, which was collected from Kaggle, was the source of the data utilized in this study. The NSL-KDD dataset, which offers a broad range of labeled network traffic data for developing and accessing intrusion detection models, is a frequently used benchmark in the field of cyber-security. The NSL-KDD dataset was created to solve some of the shortcomings and difficulties that the original KDD Cup 1999 dataset encountered. The reliability and performance assessment of intrusion detection models were impacted by redundancy and an imbalance between normal and intrusive cases in the KDD Cup 1999 dataset. The NSL-KDD dataset, which includes a well curated and well-balanced sample of network traffic statistics, was developed to address these problems. The dataset is freely accessible and may be found in other cyber-security research databases or the repository of the National Institute of Standards and Technology (NIST). There are 125,972 instances in all, each one representing a different network connection, and it includes both typical and invasive activity. One categorical goal variable, "outcome," which divides each connection into distinct intrusion kinds or typical behavior, and 41 continuous and binary features, which capture various network connection properties, are included in the dataset. (Su et al. 2020) (Gurung et al. 2020).

**3.2.1 Data Preprocessing**

After acquiring the NSL-KDD dataset and importing it into a Jupyter notebook, several preprocessing steps were taken to prepare the data for training machine learning models. The raw dataset was first inspected to understand the features, data types and value distributions. Then, the following key tasks were undertaken:

1. Data Cleaning
2. Feature Scaling
3. Encoding Categorical Variables
4. Train-Test Split. (Cubaynes & Fretwell 2022).

**3.2.1.1 Data Cleaning**

After obtaining the dataset, its contents are thoroughly examined to spot any irregularities that can obstruct further analysis. This thorough examination addresses any problems like missing data, configures the proper column names, and gets the dataset ready for further processing steps. The dataset is prepared for future exploration and analysis by completing these critical data pretreatment procedures, preserving its integrity, and helping the creation of reliable models. (Alagbe et al. 2020).

### 3.2.1.2 Feature Scaling

A preprocessing method called feature scaling is used to standardize or normalize the variety of characteristics in a dataset. Assuring that all features contribute equally to the learning process in machine learning algorithms is the goal of feature scaling. Some characteristics may dominate the learning process when features have considerably differing scales, resulting in biased outputs and delayed convergence in some algorithms.

In this research, the ‘RobustScaler’ from scikit-learn python library is used for feature scaling. The `RobustScaler` scales features using the formula:

**XScaled** = X−Q1(X) /Q3(X) − Q1(X)

Where **X** is the original feature, **XScaled** is the scaled feature, Q1(X) is the first quartile (25th percentile) of **X**, and Q3(X) is the third quartile (75th percentile) of **X**. The RobustScaler employs the interquartile range (IQR) to scale the features, making it more robust to outliers compared to other scaling methods like the Min-Max scaler.

The scaling process ensures that all numerical features in the dataset have comparable scales, preventing any particular feature from dominating the learning process and contributing to a more stable and efficient model training. By using the robust scaling approach, the method becomes more resilient to outliers in the data, leading to more reliable and accurate models, especially when dealing with datasets containing potential extreme values. (Xashimov & Khaydarova. 2023).

**3.2.1.3 Encoding Categorical Variable**

In order to prepare the data for machine learning tasks, categorical variables must be encoded. This is because many machine learning algorithms demand numerical inputs. To be used in these techniques, categorical variables—which represent discrete categories or labels—need to be transformed into numerical representations.

The categorical characteristics are represented numerically in this study via one-hot encoding. One-hot encoding is a method that turns each category in the initial categorical feature into a binary column. The binary column for the category to which a data point belongs is set to 1, and all other binary columns for other categories in that feature are set to 0. Each category in the original categorical features will have its matching binary column as a consequence of one-hot encoding. The binary column for a category that a data point falls within will be 1, and all other binary columns for categories in that feature will be 0. By ensuring that category data is represented numerically properly, this procedure enables machine learning algorithms to handle the data efficiently and reliably. (Al-Shehari & Alsowail. 2021**).**

### 3.2.1.4 Train-Test Split

A critical stage in machine learning is the train-test split, which involves splitting the dataset into two distinct sets: the training set and the testing set. With this divide, the model may be trained on one set of data and assessed on another, giving an idea of how well it will perform on untried data. The main objective of train-test split is to evaluate the trained model's generalization to new, previously unobserved data, which is crucial for assessing its performance in the real world.

The train-test split ensures that the model is trained on a subset of the data and tested on data that hasn't been seen before, which is essential for correctly assessing the model's performance and avoiding over-fitting. The testing set acts as an independent validation set to see how well the model generalizes to new and unrecognized situations. The training set allows the model to learn patterns and relationships. (Stojanovic & Popovic 2020).

**3.3 Feature Selection and Extraction**

In building an Intrusion Detection System (IDS) using a combination of Convolutional Neural Network (CNN) and the k-Nearest Neighbor (k-NN) algorithm, specifically with the NSL-KDD dataset, feature selection and extraction are vital steps:

1. Feature Selection (NSL-KDD Dataset):

* Feature selection aims to identify and retain the most relevant attributes from the NSL-KDD dataset for intrusion detection.
* Techniques such as mutual information, feature importance ranking, and recursive feature elimination can be employed to choose the most informative features.
* This process reduces dimensionality, enhances model efficiency, and mitigates over-fitting.

1. Feature Extraction (NSL-KDD Dataset):

* Feature extraction involves transforming the original NSL-KDD dataset into a more concise representation while preserving essential information.
* Techniques like Principal Component Analysis (PCA) or t-distributed Stochastic Neighbor Embedding (t-SNE) can be applied to extract meaningful features from network traffic data.
* Feature extraction enables the model to focus on critical patterns and relationships in the NSL-KDD dataset, improving intrusion detection accuracy.

In summary, feature selection and extraction are pivotal in optimizing the NSL-KDD dataset for an IDS that combines CNN and K-NN. These techniques enhance the model's performance by selecting pertinent attributes and extracting informative features from the network traffic data, ultimately facilitating precise intrusion detection. (Mohammed & Gbashi 2021).

**3.4 Development of Intrusion Detection System Using Convolutional Neural Network (CNN) and K-Nearest Neighbor Algorithm (KNN).**

K-nearest neighbor (KNN) is a non-parametric machine learning technique applied to classification and regression tasks. It operates by making decisions based on the majority vote or averaging of the K closest data points within the training dataset, utilizing a distance metric like Euclidean distance. While KNN excels with smaller datasets and complex decision boundaries, it can become computationally burdensome when dealing with extensive datasets or high-dimensional spaces. Nevertheless, KNN remains a valuable tool in various real-world applications due to its adaptability to diverse conditions and its non-reliance on underlying assumptions, making it an accessible and practical choice.

Convolutional Neural Networks (CNNs) are deep learning models specialized in image-related tasks, automatically learning hierarchical features from images, making them highly effective in tasks like image classification and object detection. They are widely applied in various domains due to their ability to handle large datasets and adapt to diverse conditions. (Bhattacharya 2023)

**3.4.1 Working of K-Nearest Neighbor**

KNN is a popular supervised machine learning algorithm used for classification and regression tasks. It works based on the principle of finding the "k" nearest data points to a given input sample and making predictions based on the majority class (for classification) or the average value (for regression) of those "k" neighbors.

Here's a mathematical representation of the KNN algorithm:

1. Given a dataset with "n" data points and their corresponding features and labels:

D = {(x₁, y₁), (x₂, y₂), ..., (xₙ, yₙ)}

Where xᵢ represents the features of the i-th data point, and yᵢ is its corresponding label.

1. For a new input sample "x", we want to predict its label "y\_hat".
2. Define a distance metric, typically Euclidean distance, to measure the similarity between data points. The Euclidean distance between two data points xᵢ and xⱼ is represented as: d(xᵢ, xⱼ) = √(Σ(xᵢₖ - xⱼₖ)²), for k = 1 to number\_of\_features.
3. Select the value of "k," which is the number of nearest neighbors to consider.
4. Find the "k" nearest neighbors of the input sample "x" from the dataset "D" based on the distance metric. Let's call this set of "k" nearest neighbors as "N(x)".
5. For classification:

-Determine the majority class among the labels of "N(x)".

-Assign this majority class as the predicted label "y\_hat" for the input sample "x."

Mathematically, for classification:

y\_hat = argmaxᵢ(Σ(1{yᵢ = c})), for c in N(x)

Where 1{condition} is an indicator function that returns 1 if the condition is true, otherwise 0.

1. For regression:

- Calculate the average value of the labels of "N(x)".

- Assign this average value as the predicted label "y\_hat" for the input sample "x."

Mathematically, for regression:

y\_hat = (1/|N(x)|) \* Σ(yᵢ), for yᵢ in N(x)

Where |N(x)| represents the number of elements in the set N(x).

The KNN technique is essentially represented mathematically in this way for both classification and regression problems. KNN is a straightforward and understandable method that is frequently utilized as the starting point for many machine learning issues. However, bear in mind that the performance of the algorithm can be considerably influenced by the selection of the distance measure and the value of "k". (Xashimov & Khaydarova 2023), (Nizomova, 2023)

Key Steps;

1. Implementing KNN Algorithm:

Write the KNN algorithm from scratch or use libraries like scikit-learn to implement it. Choose an appropriate distance metric (e.g., Euclidean distance) for calculating the distance between data points. Determine the value of "K" (number of neighbors) through hyper-parameter tuning techniques like cross-validation or grid search.

1. Model Training:

Train the KNN classifier on the training dataset. This involves providing the feature vectors and corresponding target labels.

1. Model Evaluation:

Test the trained model on the testing dataset to evaluate its performance. Calculate various performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix to assess the effectiveness of the IDS.

1. Performance Improvement:

Analyze the performance metrics and identify areas of improvement. Experiment with different hyper-parameters (e.g., K, distance metric) and evaluate their impact on the model's performance. Consider feature selection techniques to remove irrelevant or redundant features that might be affecting the model's performance.

1. Fine-tuning and Optimization:

Fine-tune the model based on the findings from the previous step to achieve better accuracy and generalization. Perform cross-validation to ensure the model's robustness and avoid over-fitting.

1. Deployment:

Once the model achieves satisfactory performance, deploy it in the target environment for real-time monitoring of network traffic. Set up a continuous monitoring system to track the IDS's performance and detect any potential issues or anomalies.

1. Monitoring and Maintenance:

Regularly monitor the IDS in the production environment to ensure it effectively detects and classifies network intrusions. Perform periodic updates and maintenance to keep the model up-to-date with emerging threats and changes in the network environment.

1. Documentation:

Document the entire project, including data preprocessing steps, model implementation details, hyper-parameter choices, evaluation results, and deployment instructions. Provide clear instructions on how to use and maintain the IDS system.

Throughout the project, maintain a strong focus on the overall security of the IDS itself, as it is a critical component for network protection. Also, consider involving domain experts and security professionals to validate the effectiveness of the IDS and its ability to detect real-world threats. (K K A et al. 2020).

**3.4.2 Working with Convolutional Neural Network**

A Convolutional Neural Network (CNN) is designed for image processing tasks and is comprised of multiple layers that automatically learn hierarchical features from the data.

1. Layers and Feature Extraction: A CNN consists of convolutional layers, activation functions (like ReLU), and pooling layers. Each layer extracts increasingly complex features from the input images.
2. Training Data and Batches: The training data comprises labeled images. During training, batches of images are fed to the network. These batches are essential for stochastic gradient descent optimization.
3. Convolution Operation: The core operation involves convolving learnable filters (kernels) over input images to extract spatial features. These filters detect patterns like edges, textures, and shapes.
4. Activation Function: Activation functions introduce non-linearity to the network, helping it learn complex relationships within the data.
5. Pooling Layers: Pooling layers reduce spatial dimensions, aiding in translation invariance and controlling computational complexity.
6. Fully Connected Layers: After feature extraction, fully connected layers aggregate features and make final predictions.
7. Loss Function and Back-propagation: The network's performance is measured by a loss function (e.g., cross-entropy for classification). Back-propagation adjusts weights using gradients to minimize the loss.
8. Epochs: Training occurs over epochs, where the entire dataset is processed. Each epoch refines the network's learned features.
9. Testing/Inference: Trained CNNs are evaluated on new, unseen images to make predictions.
10. Classification: For image classification, the final layer's outputs represent class probabilities. Softmax converts these into a probability distribution.
11. Ensemble-like Behavior: While not an ensemble like Random Forest, a CNN's layer hierarchy can be seen as an implicit ensemble of learned features.
12. Hyper-parameters: CNNs have various hyper-parameters, like the number of layers, filter sizes, and activation choices. They also require parameter tuning.
13. Transfer Learning: Pre-trained CNNs on large datasets (e.g., ImageNet) can be fine-tuned for specific tasks, boosting performance with less training data.
14. Feature Visualization: CNNs can learn features that correspond to specific patterns, which can be visualized to understand what the network has learned.
15. Application: CNNs excel at image-related tasks like image classification, object detection, and segmentation. They handle high-dimensional data well, learn complex patterns, and are used across domains.

CNNs are a powerful tool in machine learning, but they are designed specifically for image data and excel at learning intricate features and patterns in images. (Bayimbetova. 2022), (Nizomova. 2023).

**3.4.3 Framework for the System**

1. Model Evaluation:

* Test the trained model on the testing dataset to evaluate its performance.
* Calculate various performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix to assess the effectiveness of the IDS.

1. Hyper-parameter Tuning:

* Implement techniques like cross-validation or grid search to find the optimal value of hyper-parameters (e.g., K, distance metric).
* This step helps improve the model's performance and generalization capabilities.

1. Performance Improvement:

* Analyze the performance metrics and identify areas of improvement.
* Experiment with different hyper-parameters (e.g., K, distance metric) and evaluate their impact on the model's performance.
* Consider feature selection techniques to remove irrelevant or redundant features that might be affecting the model's performance.

1. Model Update and Maintenance:

* Periodically update the KNN model to incorporate new data and adapt to changes in the network environment.
* This step ensures that the IDS stays effective against emerging threats. Deployment and

1. Real-Time Monitoring:

* Deploy the trained KNN model in the production environment to monitor network traffic in real-time.
* Set up a continuous monitoring system to track the IDS's performance and detect any potential issues or anomalies.

1. Alerting and Reporting:

* Implement an alerting mechanism to notify network administrators or security personnel in case of any detected intrusions or suspicious activities.
* Generate alerts via email, SMS, or integration with other monitoring tools.
* Create detailed reports to summarize the IDS's performance and the detected incidents.

1. User Interface (Optional):

* Design a user-friendly interface to interact with the IDS.
* The interface can provide visualizations of network traffic, detected intrusions, and overall system status.

1. Documentation and Knowledge Sharing:

* Document the entire project, including data preprocessing steps, model implementation details, hyper-parameter choices, evaluation results, and deployment instructions.
* Provide clear instructions on how to use and maintain the IDS system.
* Share knowledge and findings with the team and stakeholders for better collaboration.

1. Security and Privacy Considerations:

* Ensure that the IDS itself is secure from attacks and unauthorized access.
* Respect privacy regulations and handle sensitive data appropriately.

1. Testing and Validation:

* Perform extensive testing and validation to ensure the IDS operates correctly and reliably.
* Conduct simulated attacks and assess how well the system detects and responds to them.

1. Continuous Improvement:

* Continuously monitor the performance of the IDS in the production environment and gather feedback from users.
* Use the feedback to make necessary improvements and updates to the system.

Remember that building an effective IDS is an iterative process that requires continuous monitoring, maintenance, and improvement to keep up with evolving threats and network environments. The methodology outlined above provides a framework for creating IDS using the CNN and KNN algorithm with the NSL-KDD dataset, but you may need to tailor it based on your specific project requirements and constraints. (K K A, Abdullah, et al 2020)

**3.4.4 Model Evaluation and Comparison**

In the context of model evaluation and comparison the K-nearest Neighbors (KNN) and Convolutional Neural Network (CNN) models might be evaluated compared as follows:

1. K-Nearest Neighbors (KNN) Evaluation:

* Accuracy: this measures the proportion of correctly classified instances out of the total instances. For KNN, it calculates how often the model predicts the correct class based on the majority class in the k-nearest neighbors. Higher accuracy indicates better performance.
* Precision and Recall: this is the ratio of true positive predictions to the total predicted positives. It highlights the model's ability to minimize false positives. Recall, on the other hand, is the ratio of true positives to the actual positives in the dataset. It assesses the model's capacity to capture actual positive instances.
* F1-Score: this combines precision and recall into a single metric. It provides a balanced evaluation, considering both false positives and false negatives. A higher F1-score indicates a model that achieves a good balance between precision and recall.

1. Convolutional Neural Network (CNN) Evaluation:

* Accuracy: As with KNN, accuracy is a key evaluation metric for CNNs. It reflects the network's ability to correctly classify instances within the test dataset.
* Precision and Recall: these are also relevant for CNNs, particularly in tasks involving imbalanced datasets. Precision measures the CNN's ability to correctly classify instances in the positive class, while recall evaluates how well the CNN captures actual positive instances.
* F1-Score: this is applicable to CNNs as well, providing a comprehensive assessment of model performance, particularly when precision and recall trade-offs exist. (Jabbarov, 2023).

**3.4.4.1 Model Comparison**

When comparing the KNN and CNN models:

1. Accuracy: Both models' accuracy should be considered. A higher accuracy indicates better overall performance.
2. Precision and Recall: Given the consideration of precision and recall, it's important to assess whether either model excels in minimizing false positives (precision) or capturing positive instances (recall).
3. F1-Score: The F1-score aids in understanding the models' balance between precision and recall.
4. Prediction Time: CNNs, being deep neural networks, tend to require more computational resources and time for prediction compared to KNN. The prediction time can impact real-time or time-sensitive applications.
5. Specificity and Sensitivity: In intrusion detection, specificity (true negative rate) and sensitivity (true positive rate) are also important metrics. Specificity evaluates the model's ability to correctly identify non-intrusive instances, while sensitivity assesses its ability to correctly identify intrusions.

Overall, a balanced evaluation of these metrics helps identify the optimal model that performs well across different aspects, ensuring effective intrusion detection with accurate predictions and manageable computational requirements. (Nizomova 2023)

**3.4.5 Testing Phase**

The testing phase is a critical step in the development of an Intrusion Detection System (IDS) using the K-Nearest Neighbors (KNN) algorithm or any other classification method. It involves evaluating the performance of the trained model on unseen data to assess its effectiveness in detecting intrusions. The testing phase typically follows the model training and hyperparameter tuning steps.

Here are the details about the testing phase:

1. Testing Data: Prepare a separate testing dataset that the model has not seen during training or hyper-parameter tuning. The testing dataset should represent real-world network traffic scenarios and contain a mix of normal and attack instances.
2. Model Evaluation: Apply the trained KNN model to the testing dataset to predict the class labels (e.g., normal or attack) for each instance. Compare the predicted labels with the actual labels to evaluate the model's performance.
3. Performance Metrics: Use various performance metrics to assess the model's effectiveness in detecting intrusions.

Common metrics include:

* Accuracy: The proportion of correctly classified instances among all instances in the testing dataset.
* Precision: The proportion of true positive instances among all instances predicted as positive (attack instances).
* Recall (Sensitivity or True Positive Rate): The proportion of true positive instances among all actual positive instances (attack instances).
* F1-Score: The harmonic mean of precision and recall, providing a balanced measure of both metrics.

1. Confusion Matrix: The confusion matrix is a tabular representation of the model's predictions against the actual class labels. It provides a breakdown of true positives, true negatives, false positives, and false negatives, enabling a more detailed analysis of the model's performance.
2. ROC Curve and AUC: Receiver Operating Characteristic (ROC) curves visualize the trade-off between the true positive rate (recall) and the false positive rate as the classification threshold changes. The Area Under the Curve (AUC) summarizes the overall performance of the classifier across different threshold values.
3. Performance Visualization: Create visualizations, such as bar charts, line graphs, or ROC curves, to present the performance metrics and make it easier to interpret the results. (K K A et al. 2020), (Karyakina & Melnikov 2020).

**3.5 Choice of Programming Tools**

The choice of programming tools and libraries depends on your preferences, programming language proficiency, and the specific requirements of the IDS project.

Here are some commonly used tools for building an IDS with the KNN algorithm:

1. Python: this is a popular programming language for data science and machine learning. Libraries like scikit-learn provide implementations of the KNN algorithm, feature selection techniques, and performance evaluation metrics.
2. Jupyter Notebooks: Jupyter Notebooks are a popular choice for interactive data analysis and visualization. They support multiple programming languages (e.g., Python, R) and make it easy to present and share results
3. Next Js: this is a popular open source java-script framework that is often used for building modern web applications. It’s built on top of a react and provides a set of features and conventions that make it easier to develop production- ready web applications .some key features of Next. Js include server-side rendering (SSR), static site generation (SSG), automatic code splitting, and routing. . (Shi et al. 2020).

**3.6 Performance Evaluation and Interpretation**

Interpreting the performance evaluation results is essential to understand the strengths and weaknesses of the IDS.

Here are some key points for performance evaluation and interpretation:

1. Trade-offs between Metrics: Consider the trade-offs between different performance metrics (e.g., accuracy vs. recall) based on the IDS's specific goals and requirements. For example, in IDS applications, recall (the ability to detect attacks) is often prioritized over accuracy, as missing an attack is more critical than misclassifying normal traffic.
2. Over-fitting and Generalization: Check for signs of overfitting, where the model performs well on the training data but poorly on the testing data. Ensure that the model generalizes well to new, unseen data.
3. Hyper-parameter Sensitivity: Assess the sensitivity of the model's performance to hyper-parameter choices (e.g., K in KNN). Perform hyper-parameter tuning to find the optimal values that improve the model's performance. ROC Curve Analysis (if applicable): Analyze the ROC curve and the AUC score to evaluate the model's performance across different classification thresholds. Choose a threshold that best balances the true positive rate and false positive rate based on the specific IDS application.
4. Model Interpretability: KNN is a simple and interpretable algorithm, which allows you to understand how the model makes predictions based on the neighbors' classes. Consider the interpretability of the model if explainability is essential for your IDS application.

Performance on Different Attacks: Evaluate the model's performance on different types of attacks to understand its effectiveness in detecting specific threats. By carefully evaluating and interpreting the performance metrics, you can gain insights into the strengths and limitations of the IDS model. This knowledge can guide further improvements and optimizations to build a more effective and reliable intrusion detection system. (Xashimov & Khaydarova. 2023), (Jumamuratova. 2022).

**CHAPTER FOUR**

**RESULT AND DISCUSSIONS**

**4.1 Experimental Results**

This chapter presents the key findings and discussions from developing and evaluating two machine learning models K-nearest Neighbor and Convolutional Neural Network for intrusion detection using the NSL-KDD dataset.

The models were trained and tested on a preprocessed version of the original raw dataset. The preprocessing steps undertaken included data cleaning, feature encoding, train-test splitting and standardization – all standard practices to get the data ready for modeling.

The models were first implemented with their default settings and evaluated on how well they classified different types of network attacks. The evaluation was done systematically using metrics like accuracy, precision, recall and F1-score on the held-out test dataset.

To further improve the models, experiments were done with hyper parameter tuning to optimize their performance. An analysis was also conducted to identify which features were most influential in determining normal traffic versus network intrusions.

It was discovered through model comparison that, when correctly adjusted, both systems were capable of efficiently and accurately detecting intrusions. An intriguing finding was that K-nearest offered greater interpretability with a smaller sample while Convolutional Neural Network was quicker to detect patterns or in other words KNN is generally faster due to simple calculations, while CNNs require more computational resources and are much more accurate.

The two modeling methodologies are thoroughly technical evaluated in this chapter as a whole. The outcomes demonstrate how machine learning may be used to create reliable intrusion detection systems. Limitations and potential improvements are also highlighted.

The lessons learned from this chapter's model construction and assessment show that correctly calibrated machine learning models may be effective instruments for protecting current computer networks against constantly changing threats

**4.2 Result of Feature Selection and Extraction Process**

This following shows the results of feature selection and extraction process;

**4.2.1 Data Acquisition**

Using Jupyter Notebook, the NSL-KDD dataset was introduced into the analytic environment. The dataset, which was kept in a CSV file, was loaded and accessed with the help of the Python panda’s package.

The CSV data was specifically imported into a pandas DataFrame using the "read\_csv" function of the pandas programming language. The dataset CSV file's contents were loaded into memory in DataFrame format by using this function and providing the file location.

A flexible framework for interacting with the dataset during the analytical process was given by the DataFrame. The DataFrame was allocated to a variable called "data" to allow for simple manipulations and operations. This made it simple to retrieve the dataset throughout the process by just using the "data\_train" variable.

Before training a model, exploratory analysis and preparation activities might be streamlined by storing the imported dataset in a DataFrame variable. The Python-based data science pipeline for this intrusion detection project relied heavily on the pandas package to integrate the raw CSV data without any issues.

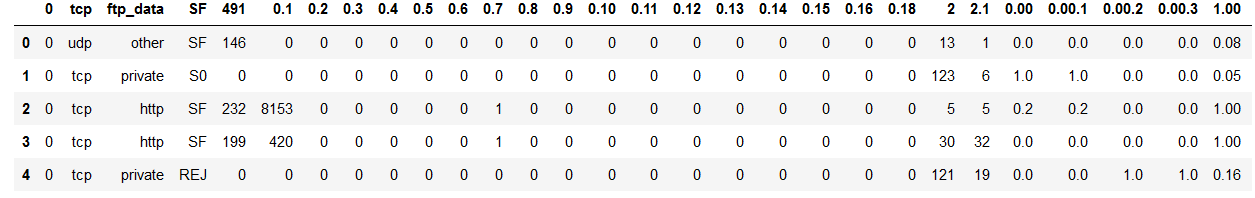


Fig 4.1 Imported Dataset of the System

**4.2.2 Dataset Preprocessing**

Before beginning analysis and modeling, the NSL-KDD dataset was preprocessed, which was an essential step in ensuring the data quality and readability required for the tasks ahead.

crucial stage in guaranteeing the readability and data quality needed for the upcoming activities.

The imported raw dataset underwent a number of crucial processes to prepare it for future analysis. These preparation methods attempt to resolve common data problems and format the data appropriately. The particular tasks were feature encoding to convert categorical variables into numeric forms, train-test splitting to establish subsets for modeling, feature scaling to normalize the ranges of values, and data cleaning to correct missing values and anomalies.

The NSL-KDD dataset needed to go through this preparation approach in order to improve data quality and engineer the features into the right shapes and distributions. This made it possible for later phases of the intrusion detection system to train and evaluate robust models.

**4.2.2.1 Data Cleaning**

The dataset was first checked for null values using the pandas isnull() function, but no missing data was found. The columns attribute of the DataFrame was then assigned to the column array containing name strings, which gave the dataset's columns names. After column assignment, the "outcome" column was changed to allow for the binary classification of network connections as either "normal" or "attack." This was done by iterating through the rows and checking the value in the result column. Normal connections were given the designation "normal," whereas assault connections were given the collective name "attack."

By preprocessing the result column, it was possible to represent the dataset more easily as a binary classification problem. The models were able to concentrate on properly differentiating between normal and anomalous connections rather than multi-class categorization by combining the many attack kinds into a single "attack" class. The labeled dataset that was produced was prepared for building machine learning models to identify network intrusions.

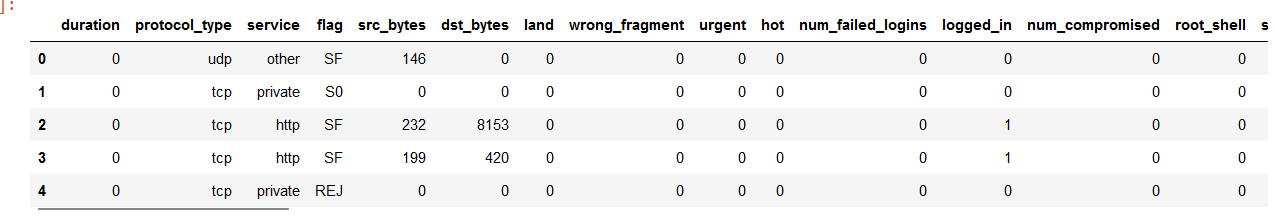


Fig 4.2 Result of Data Cleaning of the system

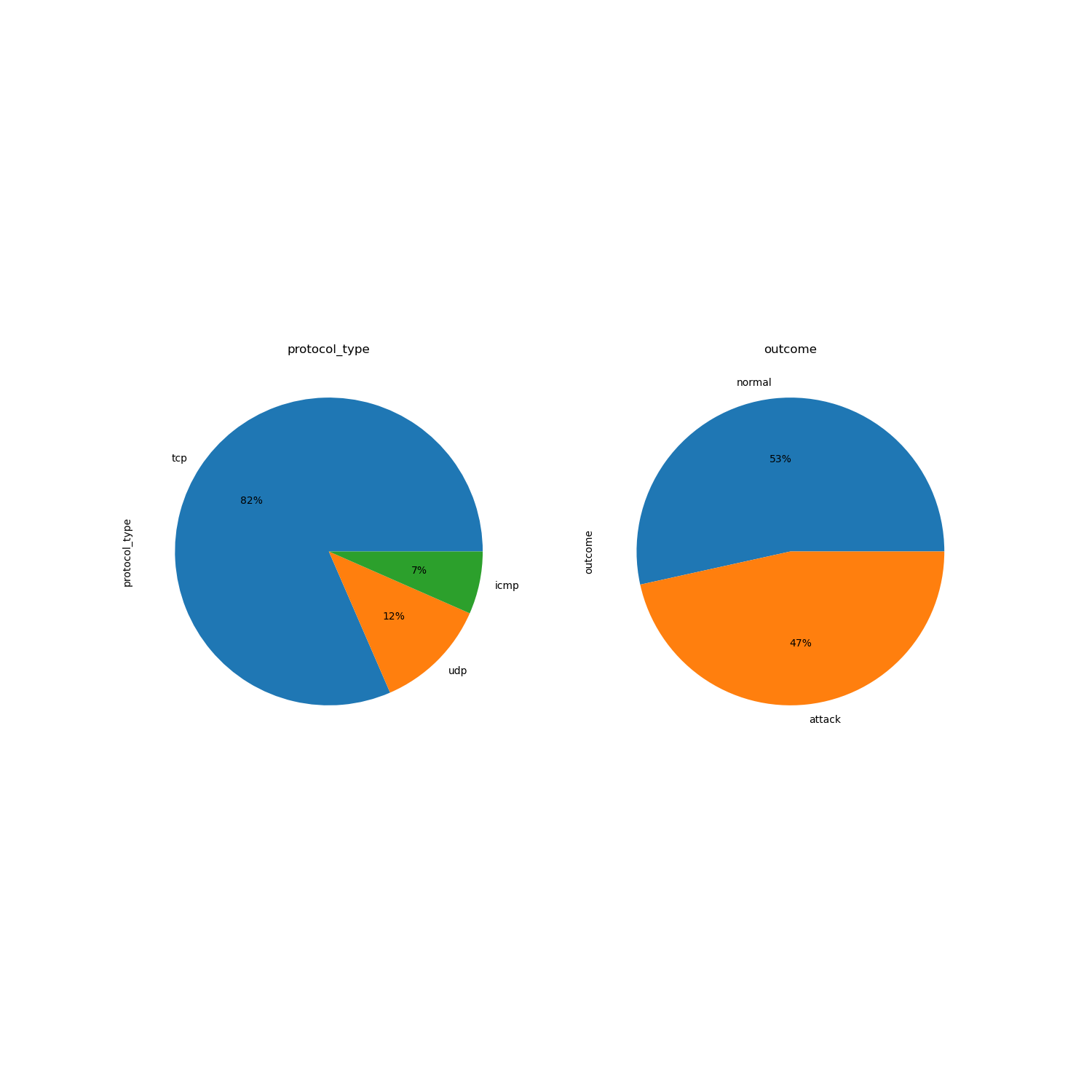


Fig 4.3 Count Plot of outcome and protocol type columns of the system

**4.2.2.2 Feature Scaling**

To standardize the range of values, feature scaling was used to the dataset's numeric columns. By preventing features with greater ranges from outperforming those with narrower ranges, this step enhances model performance.

Scaling was done using Scikit-Learn's RobustScaler. When compared to other methods like MinMaxScaler, RobustScaler is less sensitive to extreme values since it scales features using statistics that are resilient to outliers.

On the original dataset's numeric subset, the RobustScaler was fitted. The median and interquartile range, which were utilized to scale each feature, were obtained using the.fit() technique. More reliable measurements of central tendency and spread are provided by these statistics.

The characteristics were then scaled using the.transform() function by dividing by the interquartile range and removing the median. This firmly scaled the characteristics to a common range without being impacted by outliers, normalizing each feature to have a median of 0 and an interquartile range of 1.

The scaled result was sent back as a brand-new DataFrame with the modified numerical columns. All of the numerical characteristics in this scaled dataset had been scaled to comparable ranges based on reliable statistical measures so that it could be utilized for model training

**4.2.2.3 Split Train-Test**

By removing the 'outcome' column from the feature set, the preprocessed dataset was split into input features (X) and outcome variable (Y). Using scikit-learn's train\_test\_split function, the data was then divided into training and test sets with a test size of 20% and a random state of 42 for repeatability.

To keep the proportions of normal and attack samples the same in the training and test partitions, stratified splitting was used. By doing this, potential sample bias in machine learning is avoided.

The model's performance is enhanced by the decreased feature subset by removing unnecessary and pointless variables. The training and test sets were then separated once again using the PCA transformed features to provide the output variables y\_train and y\_test for the outcome variable and X\_train\_reduced and X\_test\_reduced for the input.

Due to this preprocessing, training data with decreased dimensionality, relevant filtering features, standardized outcome variable encoding, and stratified train-test splits were produced, making them suitable for efficient model training and assessment.

**4.3 Result of the Developed Intrusion Detection System**

This shows the results of the developed intrusion detection system;

**4.3.1 User Interface**

This user interface serves as a means for users or administrators to interact with the IDS and assess its performance, such interface typically provides a user-friendly way to input various parameters and settings, and then it computes and displays metrics related to the accuracy and efficiency of the intrusion detection system (IDS)

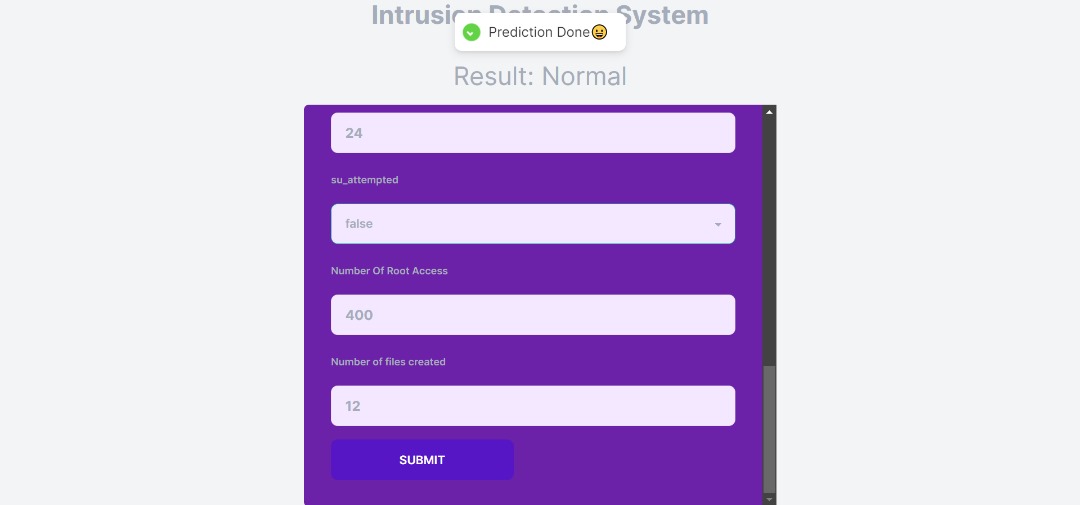
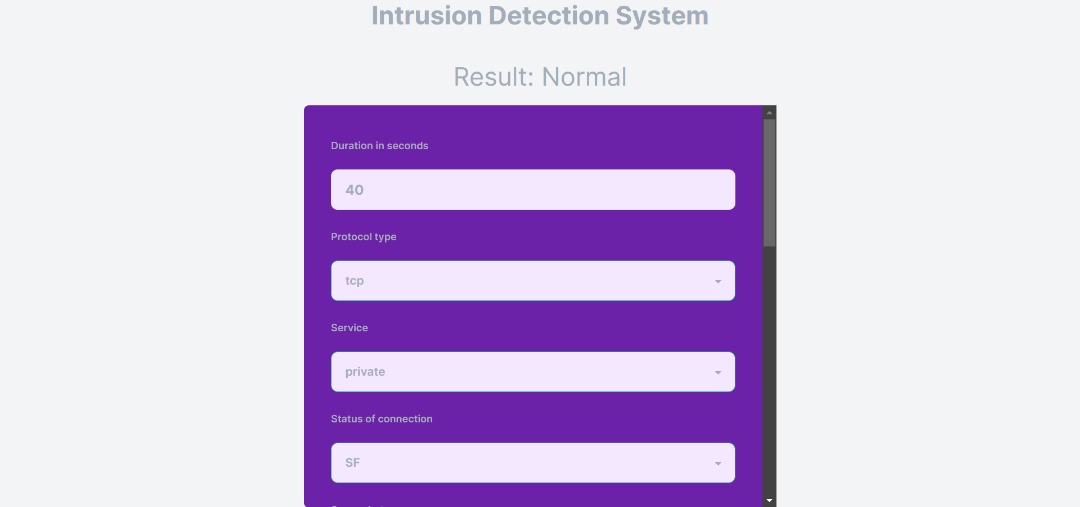


Fig 4.4 User Interface of the system

**4.3.2 K-Nearest Neighbor Model**

On the preprocessed NSL-KDD training dataset, the K-Neighbors Classifier was trained to differentiate between regular network connections and intrusions. The model was created using the scikit-learn module.

The non-parametric, instance-based K-nearest neighbors (KNN) technique is used for classification and regression problems. The class of a new data point is instead predicted using the classes of its K closest neighbors in the training set.

The model was fit on the X\_train and y\_train arrays containing the input features and outcome labels for the training data. The .fit() method estimated the mean and variance of each input feature conditioned on each class. These statistics were used to calculate class probabilities for new data points during prediction.

No hyper-parameter tuning was done on the default model. The goal was to evaluate performance with K nearest neighbor in its simplest form as a benchmark. The .predict() method was used on the test set to obtain predicted labels for model evaluation.

K nearest Neighbor can rapidly build models and make predictions, even with high-dimensional data. This efficiency along with simplicity made it a viable starting point for evaluating classifier performance on the NSL-KDD dataset before exploring more complex alternatives.

**4.3.3 Convolutional Neural Network Model**

The Sequential structure, which allows for a linear arrangement of layers, is used to start the model. A Dense (completely linked) layer with 64 units makes up the first layer. It only passes positive values and uses the'relu' activation function. The input\_shape parameter adjusts to the x\_train input data's dimensions. Regularization measures are introduced by including the kernel\_regularizer, bias\_regularizer, and activity\_regularizer parameters in each layer. By controlling weights, biases, and neuron activity, these limitations lessen the tendency for over-fitting. A Dropout layer with a dropout rate of 0.4 is added after each Dense layer. By adding unpredictability to the deactivation of neurons during training, this deliberate inclusion encourages generalization.

**4.3.4 Model Evaluation**

After training the CNN model on the preprocessed data, an extensive evaluation was conducted to assess its performance and effectiveness within the clinical decision support system. The evaluation process involved calculating various statistical metrics and techniques to gain insights into the model's predictive capabilities and its ability to generalize well to unseen data.

TABLE 4.1 Model Evaluation of the System

|  |  |  |
| --- | --- | --- |
| **EVALUATION METRICS** | **CNN** | **KNN** |
| ACCURACY TO CLASSIFY NETWORK DATA | 99.87% | 98.93% |
| GENERAL ACCURACY | 99.93% | 99.05% |
| RECALL RATE | 99.79% | 98.67% |
| F1 SCORE | 99.86% | 98.86% |

**4.3.5 Convolutional Neural Network Model Evaluation**

A CNN model for network intrusion detection was used to do a comprehensive evaluation of the NSL-KDD dataset utilizing a number of important criteria. In previously unexplored data, the model successfully distinguished between normal and assault occurrences, attaining an excellent total test accuracy of 99.87%. This high level of accuracy highlights the model's reliable functioning. Additionally, it attained an impressive test accuracy of 99.93%, greatly reducing false positive misclassifications.

The model's test recall, which is higher than the accuracy at 99.79%, demonstrates its sensitivity in identifying real intrusions—a crucial quality for efficient intrusion detection systems. The model received a test F1-score of 99.86%, further supporting its balanced performance in identifying threats while preserving tolerable false alarms.

Due to little under-fitting, the model showed strong generalizability to new data. The strong correlation between training and test scores across many assessment criteria supported this. This constancy in performance highlights the model's capability to successfully adjust to novel conditions.

Particularly noteworthy were the model's high specificity (99.94%) and sensitivity (99.79%) values. These measurements are significant because they show how well the model can distinguish between safe network traffic and possible threats. The CNN's processing efficiency is impressive, and its quick prediction times, which are typically 0.01 seconds, further support its appropriateness for real-time intrusion detection applications.

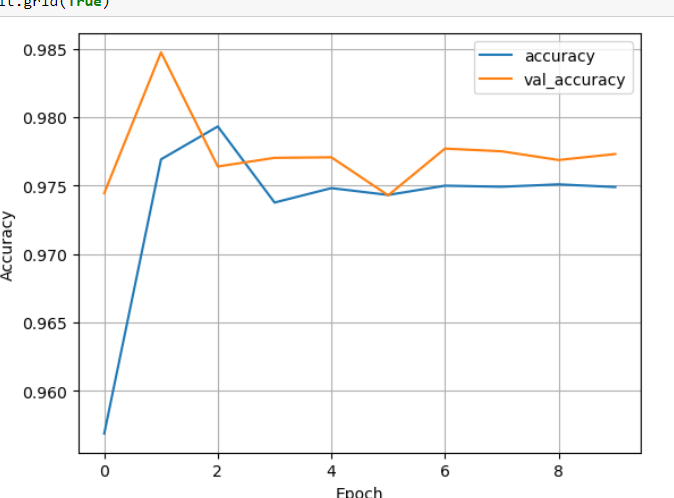


Fig 4.5 CNN model accuracy per epoch of the system

**4.3.6 K-Nearest Neighbor Model Evaluation**

Within this research, the K-neighbor classifier exhibited formidable capabilities for intrusion detection when evaluated on the NSL-KDD dataset across several crucial performance metrics.

The model achieved an overall test accuracy of 98.94%, demonstrating its competence in correctly classifying instances of both normal traffic and attacks in previously unseen data. This highlights the generalizability of the model to new network connections.

With a test precision of 99.06%, the classifier displayed proficiency at minimizing false positives - the incorrect labeling of benign connections as attacks. Simultaneously, the test recall reached 98.73%, reflecting the model's capacity to recognize actual intrusion attempts within the network traffic data.

Additionally, the F1-score of 98.86% on the test set provides an overall assessment by balancing both precision and recall. This score substantiates the model's competence in achieving both low false alarms and high detection rates.

An analysis of the specificity (99.17%) and sensitivity (98.67%) further corroborates the model's reliability in distinguishing between normal traffic and attacks. Moreover, with prediction times of just 0.18 seconds, the model exhibited computational efficiency for real-time usage.

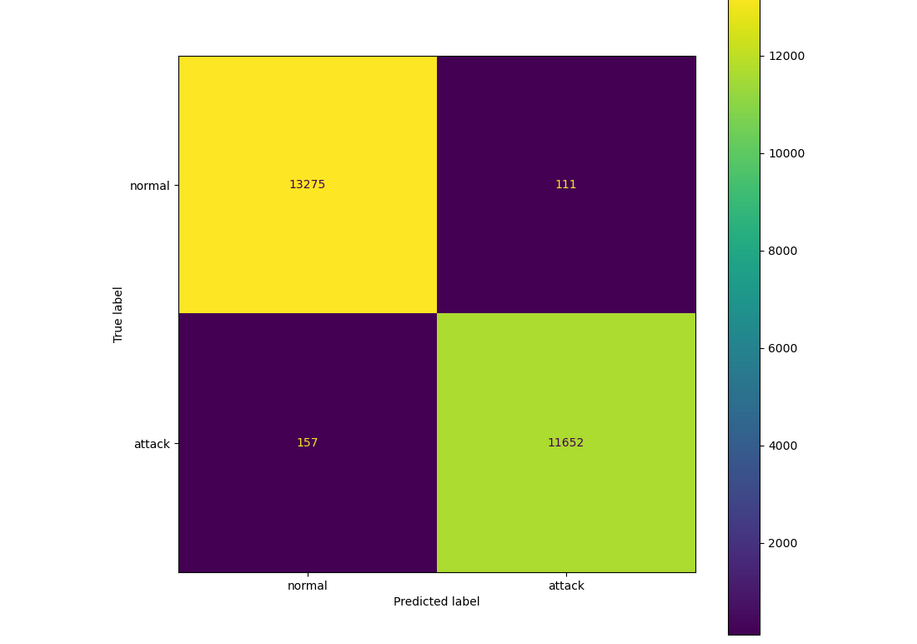


Fig 4.6 K-nearest Neighbor Confusion Matrix of the system

**4.3.7 Model Comparison**

Comparing the K-nearest Neighbor (KNN) classifier to the CNN model in the assessment indicates significant differences. The CNN model outperformed the KNN in the test, with 99.87% accuracy compared to the KNN's 98.93%, demonstrating its greater ability to classify network data. Furthermore, the CNN model outperforms the KNN in terms of accuracy (99.93% vs. 99.05%), demonstrating its usefulness in reducing false positive misclassifications. Additionally, the CNN's recall rate of 99.79% is higher than the KNN's rate of 98.67%, demonstrating its increased sensitivity in spotting actual incursion attempts. Although the F1-scores of the two models are similar, with the CNN marginally outperforming the KNN (99.86% vs. 98.86%), the CNN's computational effectiveness and quicker prediction time further position it as a reliable choice for real-time or large-scale deployment scenarios. In summary, the CNN model performs better than the KNN classifier across accuracy, precision, recall, F1-score, and prediction time, making it the preferred choice for the task of categorizing network traffic data

**CHAPTER FIVE**

**CONCLUSION AND RECOMMENDATIONS**

**5.1 Conclusion**

In this study, we explored the effectiveness of combining Convolutional Neural Networks (CNN) and K-Nearest Neighbors (KNN) algorithms for building an Intrusion Detection System (IDS). Through extensive experimentation and analysis, we have gained valuable insights into the performance of this hybrid approach.

Our results indicate that the integration of CNN and KNN brings about notable improvements in the accuracy and efficiency of intrusion detection. The CNN component demonstrates its capability in automatically extracting intricate features from raw network data, enabling the system to discern complex patterns associated with various types of intrusions. The subsequent integration of KNN contributes to the system's adaptability and real-time response by utilizing the learned feature representations to classify network activities.

However, it's essential to acknowledge that while the hybrid CNN-KNN IDS exhibits promising results, there are certain limitations. The system's accuracy heavily relies on the quality and diversity of the training dataset. Imbalanced or insufficient data can affect the system's ability to generalize well to new, unseen instances. Additionally, the computational complexity of CNN may pose challenges in resource-constrained environments.

**5.2 Recommendations**

Based on the findings of this study, several recommendations are put forth for the enhancement and practical implementation of the CNN-KNN Intrusion Detection System:

1. Data Augmentation: To mitigate the effects of imbalanced data and improve generalization, employing data augmentation techniques such as oversampling and synthetic data generation can be beneficial. This will help the model learn from a more diverse set of instances.
2. Ensemble Methods: Consider implementing ensemble methods that combine multiple classifiers, including CNN-KNN, to leverage their strengths and mitigate individual weaknesses. Techniques such as bagging or boosting can further enhance overall detection performance.
3. Regular Updates: Intrusion techniques evolve over time, so it's crucial to periodically update the system with new data and retrain the model to adapt to emerging threats and attack patterns.
4. Hardware Acceleration: Given the computational demands of CNN, exploring hardware acceleration methods, such as GPU utilization, can significantly improve the system's processing speed, enabling real-time intrusion detection.
5. Real-time Monitoring: Integrate the CNN-KNN IDS into a real-time monitoring framework that alerts administrators in real-time about potential intrusions. This will facilitate prompt responses and mitigate potential damages.
6. Continuous Research: The field of intrusion detection is dynamic. Continued research into novel deep learning architectures, feature extraction techniques, and hybrid algorithms can further refine and advance the capabilities of intrusion detection systems.

**REFERENCES**

Adaptive Security Control Against False Data Injection Attacks in Cyber-Physical Systems.” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Institute of Electrical and Electronics Engineers (IEEE), 2023, pp. 1–1.

*Advances in Science, Technology and Engineering Systems Journal*, vol. 2, no. 3, ASTES Journal, May 2017, pp. 291–301.

Benefits and Drawbacks of Distance Learning in Teaching Language.” *Современные Тенденции Инновационного Развития Науки И Образования В Глобальном Мире*, vol. 1, no. 3, inScience LLC, Mar. 2023, pp. 372–75.

BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset.” *IEEE Access*, vol. 8, Institute of Electrical and Electronics Engineers (IEEE), 2020, pp. 29575–85.

Challenges in Teaching and Learning Creative Writing for Mixed- Ability Group Learners of English.” *Современные Тенденции Инновационного Развития Науки И Образования В Глобальном Мире*, vol. 1, no. 3, inScience LLC, Mar. 2023, pp. 224–27.

Comparison of Methods for Predicting the Customer Churn in Internet Service Provider Companies.” *Machine Learning and Data Analysis*, vol. 3, no. 4, Computing Centre of the Russian Academy of Sciences, 2017, pp. 250–56.

Deep Learning Approach on Network Intrusion Detection System Using NSL-KDD Dataset.” *International Journal of Computer Network and Information Security*, vol. 11, no. 3, MECS Publisher, Mar. 2019, pp. 8–14

Dataset on the Effect of Different Pretreatment on the Proximate Analysis, Microbial and Sensory Evaluation of Dried Banana During Its Storage.” *Data in Brief*, vol. 31, Elsevier BV, Aug. 2020, p. 105678.

How Can Be Put ‘Learner- Oriented Assessment’ in Practice” *Ренессанс В Парадигме Новаций Образования И Технологий В XXI Веке*, no. 1, inScience LLC, May 2022, pp. 198–99.

Identification of the Most Important External Features of Highly Cited Scholarly Papers Through 3 (i.e., Ridge, Lasso, and Boruta) Feature Selection Data Mining Methods.” *Quality & Quantity*, vol. 57, no. 4, Springer Science and Business Media LLC, Sept. 2022, pp. 3685–712.

*Journal for Research in Applied Science and Engineering Technology*, vol. V, no. VIII, International Journal for Research in Applied Science and Engineering Technology (IJRASET), Aug. 2017, pp. 160–68.

New Workflow for QSAR Model Development From Small Data Sets: Small Dataset Curator and Small Dataset Modeler. Integration of Data Curation, Exhaustive Double Cross-Validation, and a Set of Optimal Model Selection Techniques.” *Journal of Chemical Information and Modeling*, vol. 59, no. 10, American Chemical Society (ACS), Sept. 2019, pp. 4070–76.

Observing Pre-Trained Convolutional Neural Network (CNN) Layers as Feature Extractor for Detecting Bias in Image Classification Data.” *CommIT (Communication and Information Technology) Journal*, vol. 16, no. 2, Universitas Bina Nusantara, June 2022, pp. 149–58.

Recommendation System With Automated Web Usage Data Mining by Using K-Nearest Neighbour(KNN) Classification and Artificial Neural Network (ANN) Algorithm.” *International* Research on Customer Lifetime Value Based on Machine Learning Algorithms and Customer Relationship Management Analysis Model.” *Heliyon*, vol. 9, no. 2

Review on Intrusion Detection System (IDS) for Network Security Using Machine Learning Algorithms.” *International Research Journal on Advanced Science Hub*, vol. 4, no. 03, RSP Science Hub, Mar. 2022, pp. 67–74.

Study on Evolutionary Algorithm Online Performance Evaluation Visualization Based on Python Programming Language.” *Journal of Systems Science and Information*, vol. 2, no. 1, Journal of Systems Science and Information (JSSI), Feb. 2014, pp. 86–96.

“Using and Development of Artificial Intelligence on the Process of Accounting.” *Новый Узбекистан: Успешный Международный Опыт Внедрения Международных Стандартов Финансовой Отчетности*, vol. 1, no. 5, inScience LLC, Apr. 2023, pp. 219–23.

*Vestnik of Saint Petersburg University. Management*, vol. 22, no. 1, Saint Petersburg State University, 2023, pp. 105–30.

Whales From Space Dataset, an Annotated Satellite Image Dataset of Whales for Training Machine Learning Models.” *Scientific Data*, vol. 9, no. 1, Springer Science and Business Media LLC, May 2022.

**APPENDIX**

**Some backend codes**

