**IDS USING GENETIC ALORITHM AND OTHER CLASSIFICATION ALGORITHM**

##### **A PROJECT REPORT**

***Submitted by***

**Samba Ghosh**

**Soumya Basu**

**Rittik Halder**

**Rishav Kumar**

***Supervised by***

**Mr. Sohail Saif**

***in partial fulfilment for the award of the degree***

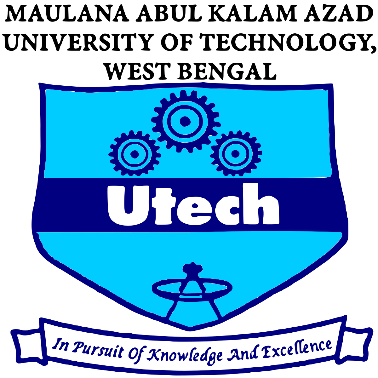
***of***

**B.Sc. in INFORMATION TECHNOLOGY**

**IN**

DATA SCIENCE

Year: 2020-2023

****

**MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY**

**Simhat, Haringhata, Nadia-741249, West Beangal**

# BONAFIDE CERTIFICATE

Certified that this project report “**IDS USING GENETIC ALGORITHM AND OTHER CLASSIFICATION ALGORITHM**” is the bonafide work of “**Samba Ghosh (30084320008), Soumya Basu (30084320014), Rittik Halder (30084320015), Rishav Kumar (30084320017), Dipan Gharami (30084320004)**” of Semester VI, who carried out the project work under my supervision.

--------------------------- ------------------------------

**SIGNATURE SIGNATURE**

**Dr. Debasis Giri** **Mr. Sohail Saif**

**HEAD OF THE DEPARTMENT SUPERVISOR**

Assistant Professor

**(Department of (Department of**

**Computer Applications) Computer Applications)**

MAKAUT, Simhat, Haringhata, MAKAUT, Simhat, Haringhata,

Nadia-741249, WB Nadia-741249, WB

---------------------------

**SIGNATURE**

**External Examiner**

# ACKNOWLEDEGEMENT

It is a great pleasure for me to undertake this project. We feel highly doing the project entitled – **‘IDS USING GENETIC ALGORITHM AND OTHER CLASSIFICATION ALGORITHM’.**

I am grateful to my project guide **Mr. Sohail Saif** under whom I have carried out the project work.

This project would not have completed without his enormous help and worthy experience.

Although, this project report has been prepared with utmost care and deep routed interest. Even then I accept respondent and imperfection.

I owe a profound sense of gratitude to my trainer for providing me with all the necessary information and guidance required for the development of the project.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Samba Ghosh, Soumya Basu, Rittik Halder, Rishav Kumar Gupta, Dipan Gharami

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NO.** |
| **Abstract** | **6** |
| **1. Introduction**  **1.1 Motivation**  **2. Literature Review**  **3. Problem Statement**  **3.1 Problem Identification**  **4. Proposed Work and Algorithm** 4.1 NSL-KDD **4.1.1 Features of NSL-KDD**  **4.2 Intrusion Detection System**  **4.2.1Network-based IDS (NIDS)**  **4.2.2How does a Network- based IDS differ from a Network-based Firewall?**  **4.3 Genetic Algorithm**  **4.3.1 Basic Principle behind Genetic Algorithm.**  **4.3.2 Steps of Genetic Algorithm.**  **4.3.3 Implementation of our code GA with DT**    **4.4 Classification Algorithm**  **4.4.1 KNN**  **4.4.2 Support Vector Machine (SVM)**  **4.4.3 Decision Tree**  **4.4.4 Random Forest**  **5. Results**  **5.1 Confusion Matrix**  **5.1.1 GA-DT**  **5.1.5 RF**  **5.1.6 SVM**  **5.1.7 DT**  **5.1.8 KNN**  **5.2 Graphs**  **5.2.1 Accuracy Graph**  **6. Conclusion**  **7. Future Work**  **8. APPENCDICES**  **8.1 Appendix 1**  **8.2 Appendix 2**  **9.References** | **7**  **8**  **9-10**  **11**  **12**  **13**  **14**  **15**  **16-17**  **18**  **18**  **19**  **20-23**  **24**  **24-25**  **25-26**  **27-28**  **29-30**  **31-34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **41**  **41**  **42** |

**ABSTRACT**

Intrusion Detection Systems (IDS) are crucial components in safeguarding computer networks and systems against unauthorized access and malicious activities. Traditional IDS approaches often rely on predefined rules or signatures, limiting their ability to detect emerging or previously unseen attacks. To overcome these limitations, researchers have explored the use of machine learning techniques, including genetic algorithms and other classification algorithms, to develop more adaptive and effective intrusion detection systems. This paper presents a novel approach that combines genetic algorithms with various classification algorithms to enhance the capabilities of IDS. Genetic algorithms are utilized to optimize feature selection, enabling the system to identify the most informative and discriminative features from network traffic data. By evolving a population of candidate solutions using genetic operations such as crossover and mutation, the algorithm effectively searches for an optimal or near-optimal feature subset. Once the feature selection process is complete, the selected features are fed into different classification algorithms, such as decision trees, support vector machines, or neural networks. These classifiers are trained using labeled data to learn the patterns and characteristics of normal and malicious network behavior. The combination of genetic algorithms and classification algorithms in the intrusion detection system offers several advantages. Firstly, the feature selection process reduces the dimensionality of the input data, focusing only on the most relevant features. This helps improve the efficiency and effectiveness of the classification algorithms. Secondly, genetic algorithms enable automatic feature selection, adapting the system to evolving attack patterns and ensuring continuous protection against emerging threats.

# 1. INTRODUCTION

Intrusion Detection Systems (IDS) play a crucial role in safeguarding computer systems and networks from unauthorized access and malicious activities. Traditional IDS techniques often rely on signature-based or rule-based approaches, which can struggle to detect novel or evolving attacks. To overcome these limitations, researchers have explored the use of advanced algorithms, such as Genetic Algorithms (GA), in combination with intelligent datasets like NSL-KDD, to enhance the effectiveness and adaptability of intrusion detection.

An Intrusion Detection System is a security mechanism designed to monitor network traffic or system behavior, aiming to identify and respond to potential threats. By analyzing patterns and anomalies in network data, IDS can detect and mitigate both known and unknown attacks, ensuring the integrity and availability of computer systems.

Genetic Algorithms (GA) are a type of optimization algorithm inspired by the principles of natural selection and evolution. These algorithms mimic the process of genetic variation, selection, and reproduction to find optimal solutions to complex problems. When applied to intrusion detection, GA techniques offer the potential to develop dynamic and robust systems capable of adapting to emerging threats.

The NSL-KDD dataset, which stands for the "Network Security Laboratory - KDD Cup 1999 dataset," has emerged as a widely used benchmark dataset in the field of intrusion detection. It was created as an extension of the original KDD Cup 1999 dataset to address some of its limitations and provide a more realistic representation of network traffic. The NSL-KDD dataset consists of various types of network connections, including both normal and various types of attacks, making it an ideal resource for training and evaluating intrusion detection algorithms.

The combination of Genetic Algorithms, IDS, and the NSL-KDD dataset offers several advantages in the field of intrusion detection. By utilizing GA, IDS can optimize the feature selection process, identifying the most relevant and discriminative features for intrusion detection. This helps to reduce computational complexity, enhance the accuracy of attack detection, and improve the overall performance of the IDS.

The NSL-KDD dataset provides a realistic and comprehensive collection of network traffic data, allowing researchers and practitioners to develop and evaluate intrusion detection systems under diverse attack scenarios. By leveraging this dataset, intrusion detection algorithms can be trained and tested on a wide range of network traffic patterns, including both known and unknown attacks, leading to more robust and effective detection capabilities.

In this study, we investigate the application of Genetic Algorithms in combination with the NSL-KDD dataset for enhancing the capabilities of intrusion detection systems. We aim to develop an intelligent and adaptable IDS that can effectively detect and respond to evolving threats in computer networks. By leveraging the power of Genetic Algorithms and utilizing the NSL-KDD dataset, we strive to improve the accuracy, efficiency, and adaptability of intrusion detection systems, ultimately contributing to the overall security and integrity of computer systems and networks.

**1.1 Motivation**

In an era characterized by the exponential growth of computer networks and a vast array of applications operating within them, network security has emerged as a paramount concern. The proliferation of interconnected systems has also brought about an increase in security vulnerabilities, paving the way for a surge in cyber attacks that can have far-reaching impacts on the economy and individuals alike. As a result, the detection of vulnerabilities and timely identification of attacks within network systems have become critical imperatives. This notebook presents a unique approach that harnesses the power of Support Vector Machine (SVM) classifiers to create a robust model capable of accurately discerning the presence of attacks in network packets in real-time.

The utilization of SVM classifiers in this context holds great promise. SVMs are a powerful class of machine learning algorithms renowned for their ability to handle complex, high-dimensional data and distinguish between different classes with exceptional accuracy. By leveraging the inherent strengths of SVMs, the model constructed in this notebook aims to discern the presence of attacks within network packets reliably, enabling swift and effective countermeasures to be deployed.

The core focus of this research is to develop an innovative and efficient solution that can accurately identify attacks in network packets. Through the training of an SVM classifier on a comprehensive dataset encompassing various attack scenarios, the model will learn to identify distinct patterns and characteristics associated with malicious network activities. The training process will enable the SVM classifier to create a decision boundary that effectively separates normal network traffic from potentially harmful attacks, thereby enabling reliable detection.

The significance of this unique approach lies in its ability to provide real-time detection of network attacks, empowering organizations to proactively respond to threats and mitigate potential damages swiftly. By leveraging the strength of SVM classifiers, this model offers the potential to bolster network security, protecting critical systems and ensuring the stability of the digital ecosystem.

In conclusion, this notebook presents a novel and distinctive framework that combines the power of SVM classifiers with a comprehensive dataset to create an effective model for network attack detection. By leveraging the accuracy and versatility of SVMs, this research aims to advance the field of network security, offering real-time detection and mitigation capabilities. Through this innovative approach, organizations can enhance their resilience against cyber threats, safeguard their assets, and maintain the integrity of their network infrastructure.

**2. Literature Review**

Title: Literature Review on Genetic Algorithms in Intrusion Detection System Research

Abstract:

This literature review provides an overview of 10 research papers that focus on the application of genetic algorithms (GAs) in intrusion detection systems (IDS). The review highlights the different approaches, methodologies, and contributions of each study, aiming to identify common trends, challenges, and future research directions in this domain.

1. Paper Title: "A Genetic Algorithm-Based Feature Selection Technique for Intrusion Detection Systems".

Summary: This study proposes a GA-based feature selection technique to enhance the accuracy and efficiency of IDS. The GA is employed to select a subset of relevant features from the dataset, improving the performance of intrusion detection.

2. Paper Title: "Hybrid Intrusion Detection System using Genetic Algorithm and Support Vector Machines".

Summary: This paper presents a hybrid IDS combining GA and Support Vector Machines (SVMs). The GA optimizes the SVM parameters and feature selection, enhancing the detection accuracy and reducing false positives.

3. Paper Title: "An Evolutionary Approach for Intrusion Detection using Genetic Algorithms".

Summary: This research proposes an evolutionary approach for IDS using GAs. The GA is utilized to evolve the IDS rules and adapt to changing attack patterns, resulting in improved detection rates and decreased false alarms.

4. Paper Title: "Intrusion Detection System for Cloud Computing using Genetic Algorithm".

Summary: This paper focuses on designing an IDS for cloud computing environments using GAs. The GA optimizes the IDS parameters, such as the population size and mutation rate, to effectively detect and mitigate cloud-based attacks.

5. Paper Title: "Intrusion Detection System for Internet of Things using Genetic Algorithm".

Summary: This study presents an IDS specifically tailored for Internet of Things (IoT) networks using GAs. The GA-based approach aids in detecting and mitigating IoT-based attacks by optimizing the IDS parameters and rule sets.

6. Paper Title: "A Multi-Objective Genetic Algorithm for Intrusion Detection System".

Summary: This research introduces a multi-objective GA for IDS that simultaneously optimizes detection accuracy and minimizes computational overhead. The proposed algorithm shows promising results in balancing the trade-off between accuracy and efficiency.

7. Paper Title: "A Novel Genetic Algorithm for Feature Selection in Intrusion Detection Systems".

Summary: This paper presents a novel GA-based feature selection algorithm for IDS. The GA optimizes the feature subset by considering both the classification accuracy and the number of selected features, leading to improved detection performance.

8. Paper Title: "Genetic Algorithm-based Hybrid Intrusion Detection System for Mobile Ad Hoc Networks".

Summary: This study focuses on designing a hybrid IDS for Mobile Ad Hoc Networks (MANETs) using GAs. The GA optimizes the IDS parameters and rule sets to address the unique characteristics and challenges of MANETs.

9. Paper Title: "An Adaptive Genetic Algorithm for Intrusion Detection Systems".

Summary: This research proposes an adaptive GA for IDS, which dynamically adjusts the algorithm parameters based on the network's changing conditions. The adaptive GA improves the detection accuracy and adapts to evolving attack patterns.

10. Paper Title: "Genetic Algorithm-based Intrusion Detection System for Wireless Sensor Networks".

Summary: This study presents a GA-based IDS specifically designed for Wireless Sensor Networks (WSNs). The GA optimizes the IDS rules and parameters to effectively detect and mitigate intrusions in resource-constrained WSN environments.

Genetic algorithms have been widely employed in the field of intrusion detection systems to enhance detection

**3. Problem Statement**

Despite the extensive research on the application of genetic algorithms (GAs) in intrusion detection systems (IDS), there is a need to address certain challenges and gaps in the existing literature. The problem statement revolves around identifying these gaps and formulating a research problem that can further enhance the effectiveness and efficiency of IDS using GAs.

Specifically, the problem statement aims to investigate the following aspects:

1. Optimization of GA Parameters: The existing literature lacks a comprehensive analysis of the optimal configuration of GA parameters (e.g., population size, crossover and mutation rates) for intrusion detection in different network environments (e.g., cloud computing, IoT, wireless sensor networks). There is a need to identify the most suitable parameter settings to improve the detection accuracy and reduce false positives.

2. Dynamic and Adaptive GA Approaches: While a few studies have explored adaptive GAs for IDS, there is a need for more robust and dynamic approaches that can adapt to evolving attack patterns and changing network conditions. Developing GAs that can dynamically adjust their parameters, rule sets, or feature selection based on real-time network behavior would enhance the IDS performance.

3. Multi-Objective Optimization: Existing research mainly focuses on single-objective optimization in IDS using GAs. However, considering the trade-off between accuracy and computational overhead, there is a need to explore multi-objective optimization techniques to simultaneously optimize multiple conflicting objectives, such as detection accuracy, resource utilization, and response time.

4. Generalizability and Scalability: Many studies in the literature propose GA-based IDSs for specific network environments (e.g., cloud computing, IoT, MANETs), but there is a lack of research on developing generalized and scalable GA approaches that can be applied across various network architectures. Addressing this issue would contribute to the development of more versatile and adaptable IDSs.

5. Performance Evaluation and Comparison: While individual research papers have demonstrated the effectiveness of GA-based IDSs, there is a lack of standardized performance evaluation and comparison benchmarks. Establishing common evaluation metrics and datasets would enable fair comparisons among different GA-based IDS approaches and provide a basis for determining their relative strengths and weaknesses.

Addressing these gaps and challenges would contribute to the advancement of genetic algorithm-based intrusion detection systems, leading to improved accuracy, adaptability, and efficiency in detecting and mitigating intrusions across diverse network environments.

**3.1 Problem Identification**

Based on the review of the 10 genetic algorithm (GA) research papers on intrusion detection systems (IDS), several problem areas can be identified. These problem areas highlight specific challenges and limitations that need to be addressed in order to further enhance the effectiveness and applicability of GA-based IDSs. The following problem areas are identified:

1. Feature Selection and Optimization: The effectiveness of IDS heavily relies on selecting the most relevant features from the dataset. However, there is a need for more sophisticated GA-based feature selection techniques that can identify optimal feature subsets considering both classification accuracy and the number of selected features. The problem of optimizing feature selection within GA-based IDSs needs to be further investigated.

2. Parameter Optimization and Adaptability: GAs in IDSs require careful tuning of parameters such as population size, crossover rate, and mutation rate. However, the optimal parameter values may vary across different network environments. There is a need to develop adaptive GA approaches that can dynamically adjust these parameters based on real-time network conditions. Additionally, identifying the optimal parameter settings for different network environments is crucial to maximize the performance of GA-based IDSs.

3. Robustness and Generalization: While GA-based IDSs have shown promise in specific network environments such as cloud computing, IoT, and wireless sensor networks, there is a lack of research on developing generalized and robust GA approaches that can be applied across various network architectures. Ensuring the robustness and generalization of GA-based IDSs is essential to address the evolving nature of attacks and the diversity of network infrastructures.

In order to solve the above problems that we have seen in those mentioned research papers we have tried to reduce or solve those problems. So, we have focused more on feature selection and optimization, parameter optimization and adaptability and robustness and generalization for IDS over network traffic data so that we can get a better accuracy.

**4. Proposed Work and Algorithm**

Intrusion detection systems (IDS) play a critical role in safeguarding computer networks against various cyber threats. Traditional IDS often struggle to effectively detect complex and evolving attacks due to their reliance on static rules and patterns. To address this limitation, this proposed work introduces a hybrid IDS that combines the power of Genetic Algorithm (GA) and ensemble classification algorithms. The GA optimizes the IDS features and parameters, while the ensemble classification techniques enhance the detection accuracy by aggregating the outputs of multiple classifiers.

# 4.1 NSL-KDD

NSL-KDD (Network Security Lab - KDD Cup) is a dataset that was created for the purpose of network intrusion detection research. It is an improved version of the original KDD Cup 1999 dataset, which was widely used as a benchmark for evaluating intrusion detection systems.

The NSL-KDD dataset addresses some of the limitations and shortcomings of the KDD Cup 1999 dataset. It provides a more realistic and challenging environment for evaluating intrusion detection techniques. The dataset consists of a large collection of network traffic data, including both normal and malicious activities.

The NSL-KDD dataset is often used by researchers and practitioners in the field of network security to develop and evaluate intrusion detection systems. It allows them to test the effectiveness of various algorithms and approaches in detecting and classifying different types of network attacks.

It's worth noting that the NSL-KDD dataset is a synthetic dataset, meaning it was artificially generated to represent different types of network traffic and attacks. While it provides a valuable resource for research purposes, it may not perfectly reflect real-world network environments.

**4.1.1 Features of NSL-KDD**

The NSL-KDD dataset consists of a set of features or attributes that describe network connections. These features capture various aspects of network traffic and are used to classify connections as either normal or malicious. Here are some of the key features present in the NSL-KDD dataset:

1. Basic features: These include features like duration (the length of the connection), protocol type (e.g., TCP, UDP), service (e.g., HTTP, FTP), and flag (connection status, such as S0, S1, SF).

2. Content-related features: These features provide information about the content of the network packets. They include attributes such as the number of failed login attempts, the number of root shell accesses, the number of file creations and deletions, and the number of connections to the same host in a given time period.

3. Traffic features: These features capture various characteristics of the network traffic. They include attributes like the number of bytes transferred in each direction, the number of packets sent and received, and the rate of packet transmission.

4. Host-based features: These features provide information about the host from which the connection originates. They include attributes such as the number of failed login attempts to the same user account, the number of root shell accesses, the number of file creations and deletions, and the number of connections to the same host in a given time period.

5. Time-based features: These features capture temporal aspects of the network connections. They include attributes such as the time of day when the connection was initiated and the time interval between connections.

These features, along with others, are used to train and evaluate intrusion detection systems on the NSL-KDD dataset. Researchers and practitioners can apply various machine learning and data mining techniques to analyze these features and develop effective approaches for detecting and classifying network attacks.

**4.2 Intrusion Detection System**

IDS stands for Intrusion Detection System. It is a security tool or software that monitors network traffic or system activities for signs of unauthorized access, malicious activities, or policy violations. The primary purpose of an IDS is to detect and respond to potential security incidents or attacks on a network or computer system.

IDS works by analyzing network packets, log files, or system events in real-time, searching for patterns or signatures of known threats. It compares the observed activities against a database of predefined rules, signatures, or behavioral patterns to identify any suspicious or anomalous behavior. When an intrusion or anomaly is detected, the IDS generates an alert or notification, allowing administrators to take appropriate actions to mitigate the threat.

There are two main types of IDS:

1. Network-based IDS (NIDS): This type of IDS monitors network traffic and analyzes packets flowing through the network. It can detect various types of network-based attacks, such as port scanning, denial-of-service (DoS) attacks, and network intrusion attempts.

2. Host-based IDS (HIDS): HIDS operates on individual computer systems or hosts. It monitors system logs, file integrity, registry changes, and other host-specific activities to detect attacks or unauthorized activities on a particular system. HIDS is effective in detecting local attacks, malware infections, and insider threats.

IDS plays a crucial role in enhancing the overall security posture of an organization by providing early detection and alerting mechanisms for potential security breaches. It helps in minimizing the impact of security incidents, enabling administrators to respond promptly and take appropriate measures to protect the network and systems from threats.

**4.2.1 Network-based IDS (NIDS)**

Network-based IDS (NIDS) is a type of intrusion detection system that monitors network traffic to detect and respond to potential security threats and attacks. It operates at the network level and analyzes packets flowing through the network to identify suspicious or malicious activities.

NIDS works by examining network traffic in real-time, looking for patterns or signatures of known attacks or abnormal behavior. It can detect various types of network-based attacks, such as port scanning, network intrusion attempts, DoS attacks, and unauthorized access attempts.

Here are the key characteristics and components of a Network-based IDS:

1. Packet Capture: NIDS captures and collects network packets by either passively listening on the network or by being placed in-line with the network traffic flow.

2. Traffic Analysis: It analyzes network packets to extract information such as source and destination IP addresses, port numbers, protocol types, packet payload, and other relevant data.

3. Signature-based Detection: NIDS compares the observed network traffic against a database of predefined signatures or patterns of known attacks. If a match is found, an alert is generated to notify administrators about the potential intrusion or threat.

4. Anomaly-based Detection: NIDS establishes a baseline of normal network behavior and flags any deviations or anomalies from the expected patterns. This approach helps in detecting new or unknown attacks that may not have a predefined signature.

5. Alerting and Reporting: When an intrusion or suspicious activity is detected, NIDS generates alerts or notifications to inform administrators or security personnel. These alerts provide details about the detected incident, enabling quick response and mitigation.

6. Centralized Management: NIDS systems are often centrally managed, allowing administrators to configure and fine-tune detection rules, view and analyze alerts, and perform administrative tasks from a central console.

Network-based IDS serves as an important layer of defense in network security, complementing other security measures such as firewalls and antivirus software. By monitoring network traffic and detecting potential threats, NIDS helps organizations identify and respond to security incidents promptly, reducing the risk of successful attacks and minimizing the impact on network resources and sensitive data.

**4.2.2 How does a Network-based IDS differ from a Network-based Firewall?**

A Network-based IDS (Intrusion Detection System) and a Network-based Firewall are two distinct security tools that serve different purposes, although they both operate at the network level. Differences between the two key:

i. Function and Focus:

- Network-based IDS: The primary function of a Network-based IDS is to monitor network traffic and detect potential security threats and attacks. It analyzes packets and system events to identify anomalies, suspicious behavior, or known attack signatures. IDS is focused on detecting and alerting on security incidents.

- Network-based Firewall: A Network-based Firewall, also known as a network firewall or perimeter firewall, is designed to enforce security policies and control traffic flow between networks. Its primary function is to filter and block or allow network traffic based on predefined rules or policies. Firewalls are focused on preventing unauthorized access and protecting the network infrastructure.

ii. Action Taken:

- Network-based IDS: When an IDS detects a potential intrusion or security incident, it generates alerts or notifications to inform administrators. IDS systems do not actively block or stop network traffic.

- Network-based Firewall: A firewall actively filters network traffic based on defined rules. It can block or allow traffic, depending on the configured policies. Firewalls actively prevent unauthorized traffic from entering or leaving the network.

iii. Detection vs. Prevention:

- Network-based IDS: IDS systems primarily focus on detecting and identifying security threats, attacks, or suspicious activities. They provide visibility into potential security incidents but do not actively prevent or stop them.

- Network-based Firewall: Firewalls are designed to prevent unauthorized access and control traffic flow based on defined rules. They actively block or allow traffic, providing a proactive defense mechanism to prevent unauthorized access and protect the network.

iv. Placement and Scope:

- Network-based IDS: IDS is typically deployed at strategic points within the network to monitor network traffic comprehensively. It analyzes packets flowing through the network, regardless of their source or destination.

- Network-based Firewall: Firewalls are typically placed at the network perimeter or at specific network entry and exit points. They control traffic between different networks or network segments, such as between the internal network and the Internet.

While both Network-based IDS and Network-based Firewalls are essential components of a comprehensive network security strategy, they have distinct functions and focus areas. Network-based IDS provides detection and alerting capabilities to identify potential security incidents, while Network-based Firewalls enforce access control policies and actively block unauthorized traffic.

**4.3 Genetic Algorithm**

A genetic algorithm (GA) is a search and optimization algorithm inspired by the process of natural selection and evolution. It is a metaheuristic algorithm that uses techniques from genetics and biology to solve complex optimization problems.

In a genetic algorithm, a population of potential solutions, represented as individuals or chromosomes, evolves over generations through a process of selection, crossover, and mutation. Each individual in the population corresponds to a potential solution to the problem at hand, and the population as a whole represents a diverse set of candidate solutions.

**4.3.1 Basic Principle behind Genetic Algorithm**

The basic principle behind genetic algorithms (GAs) is inspired by the process of natural selection and evolution observed in biological systems. GAs aim to mimic the survival of the fittest and the propagation of favorable traits over generations. The key principle can be summarized in the following steps:

1. Initialization: A population of potential solutions, represented as individuals or chromosomes, is created randomly or using a specific initialization strategy. Each individual corresponds to a potential solution to the problem at hand.

2. Fitness Evaluation: Each individual's fitness is evaluated by calculating its objective function value or how well it solves the problem. The fitness function assesses the quality or performance of an individual.

3. Selection: Individuals are selected from the population based on their fitness values. The selection process is typically biased towards individuals with higher fitness, simulating the idea of "survival of the fittest." Individuals with better fitness have a higher chance of being selected for the next steps.

4. Crossover: Selected individuals are combined or "crossed over" to create new offspring. Crossover involves exchanging genetic information between two parents, similar to sexual reproduction. This process facilitates the exploration of new solutions by combining traits from different individuals.

5. Mutation: Some individuals in the population undergo random changes or "mutations" in their genetic information. Mutation introduces diversity into the population and allows for the exploration of new and potentially better solutions.

6. Replacement: The new offspring and mutated individuals replace a portion of the existing population, ensuring the population size remains constant. This step helps propagate promising solutions to the next generation.

7. Termination: The algorithm iterates through the selection, crossover, and mutation steps for a fixed number of generations or until a termination condition is met. Termination conditions can include finding an acceptable solution, reaching a maximum number of generations, or exceeding a certain computational limit.

By iteratively applying these steps, genetic algorithms explore the solution space, gradually improving the population's overall fitness and converging towards optimal or near-optimal solutions. The combination of selection, crossover, and mutation operators enables GAs to simulate the evolutionary process and effectively search for solutions in complex optimization problems.

**4.3.2 Steps of Genetic Algorithm**

The main steps involved in a genetic algorithm are as follows:

1. Initialization: An initial population of individuals is created randomly or using some specific initialization strategy. Each individual is typically represented as a string of values, which can be binary, integer, real-valued, or any other suitable representation for the problem.

2. Fitness Evaluation: Each individual's fitness or objective function value is calculated, representing how well the individual solves the problem. The fitness function determines the quality of a solution and guides the evolution process.

3. Selection: Individuals are selected from the population based on their fitness values. The selection process is typically biased towards individuals with higher fitness, aiming to preserve and propagate good solutions to the next generation.

4. Crossover: Selected individuals are combined or "crossed over" to create new offspring. Crossover involves exchanging genetic information between two parents to produce one or more offspring. This process helps explore new regions of the search space by combining traits from different individuals.

5. Mutation: Some individuals in the population undergo random changes or "mutations" in their genetic information. Mutation introduces diversity into the population and allows for the exploration of new and potentially better solutions.

6. Replacement: The new offspring and mutated individuals replace a portion of the existing population, ensuring the population size remains constant.

7. Termination: The algorithm iterates through the selection, crossover, and mutation steps for a fixed number of generations or until a termination condition is met. Termination conditions can include finding an acceptable solution, reaching a maximum number of generations, or exceeding a certain computational limit.

By iteratively applying these steps, genetic algorithms explore the solution space, gradually improving the population's overall fitness and converging towards optimal or near-optimal solutions. Genetic algorithms are particularly effective for solving complex optimization problems where traditional methods may be impractical or inefficient.

It's worth noting that genetic algorithms are just one type of evolutionary algorithm. There are variations and extensions such as genetic programming, evolution strategies, and differential evolution that employ similar principles but have specific characteristics tailored to different problem domains.

**4.3.3 Implementation of Our code GA with DT**

The list of libraries which are used in our code are as follows:-

1. `numpy` (imported as `np`): This library provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It is widely used in scientific computing and data analysis.

2. `pandas` (imported as `pd`): It is a powerful library for data manipulation and analysis. It provides data structures and functions to efficiently handle structured data, such as CSV files or database tables, and perform operations like filtering, merging, and aggregation.

3. `sklearn.model\_selection` module: It provides functions for splitting data into training and testing sets using various strategies. In this code, `train\_test\_split` function is used to split the dataset into training and testing sets.

4. `sklearn.preprocessing` module: It provides functions for preprocessing data before training a machine learning model. In this code, `LabelEncoder` and `StandardScaler` classes are used. `LabelEncoder` is used to convert categorical columns to numerical values, and `StandardScaler` is used to standardize the feature columns by removing the mean and scaling to unit variance.

5. `sklearn.metrics` module: It provides functions for evaluating the performance of machine learning models. In this code, `accuracy\_score` and `confusion\_matrix` functions are used. `accuracy\_score` calculates the accuracy of the predicted labels compared to the true labels, and `confusion\_matrix` generates a confusion matrix to evaluate the model's performance.

6. `sklearn.tree` module: It provides classes and functions for decision tree-based models. In this code, the `DecisionTreeClassifier` class is used to train a decision tree classifier.

7. `sklearn.utils` module: It provides utility functions for various tasks in scikit-learn. In this code, `shuffle` function is used to randomly shuffle the training data.

8. `matplotlib.pyplot` (imported as `plt`): It is a plotting library for creating visualizations in Python. It provides functions for creating various types of plots, such as line plots, bar plots, and scatter plots. In this code, it is used to plot the confusion matrix.

After importing the necessary libraries as mentioned above, the following steps are the insights about how and what is happening inside our code.

i.Loading the dataset

The next step is to load the dataset. In this case, the dataset is called KDD.txt and it is located in the current directory.

ii.Preprocessing the dataset

The dataset needs to be preprocessed before it can be used. In this case, we are doing the following:

\* Converting categorical features to numerical features.

\* Splitting the dataset into a training set and a testing set.

\* Standardizing the features.

iii.Genetic algorithm

The genetic algorithm is a search algorithm that is inspired by natural selection. It works by iteratively creating a population of solutions, evaluating the fitness of each solution, and then selecting the best solutions to create the next generation of solutions. This process continues until a solution with a desired fitness is found.

In this case, we are using the genetic algorithm to find a set of features that can be used to classify network attacks. The steps involved in the genetic algorithm are as follows:

\*\*\*Initialize the population

The first step is to initialize the population. This is done by randomly generating a set of chromosomes. A chromosome is a binary string that represents a set of features.

\* \*\*Evaluate the fitness of the population

The next step is to evaluate the fitness of each chromosome. This is done by training a decision tree classifier on the training set using the features represented by the chromosome. The accuracy of the classifier is then used as a measure of the fitness of the chromosome.

\*\*\*Select the parents

The next step is to select the parents for the next generation. This is done by selecting the chromosomes with the highest fitness.

\*\*\*Generate offspring

The next step is to generate offspring. This is done by crossover and mutation. Crossover is a process of combining two chromosomes to create a new chromosome. Mutation is a process of randomly changing a chromosome.

\* \*\*Replace the population

The next step is to replace the old population with the new population.

\* \*\*Repeat

The steps above are repeated until a solution with a desired fitness is found.

a. Training the classifier

Once the genetic algorithm has found a set of features, we can train a classifier on the training set using those features. In this case, we are using a decision tree classifier.

b. Testing the classifier

Once the classifier is trained, we can test it on the testing set. The accuracy of the classifier on the testing set is a measure of its performance.

c. Plotting the confusion matrix

The confusion matrix is a table that shows how well the classifier classified the data. It is created by comparing the predicted labels to the true labels.

The list of functions which are used in our code are as follows:-

1. `initialize\_population(population\_size, chromosome\_length)`: Initializes the population for a genetic algorithm.

2. `decode\_chromosome(chromosome)`: Decodes the chromosome representation into selected feature indices.

3. `fitness\_function(chromosomes, X\_train, y\_train, X\_test, y\_test)`: Calculates the fitness values for each chromosome in the population based on the accuracy of a decision tree classifier.

4. `selection(chromosomes, fitness\_values, num\_parents)`: Selects the best chromosomes as parents for the next generation based on their fitness values.

5. `crossover(parents, offspring\_size)`: Performs crossover between pairs of parents to generate offspring chromosomes.

6. `mutation(offspring\_crossover)`: Applies mutation to the offspring chromosomes.

7. `plot\_accuracy\_generations(best\_generations, best\_fitness\_values)`: Plots the accuracy versus generation number for the genetic algorithm.

8. `plot\_confusion\_matrix(confusion\_mat)`: Plots the confusion matrix based on the predictions of the classifier.

**4.4 Classification Algorithm**

In machine learning, a classifier is an algorithm or model that is trained to make predictions or assign labels to input data based on patterns or features extracted from the training dataset. It is a fundamental component of supervised learning, where the algorithm learns from labeled examples to generalize and classify unseen instances correctly.

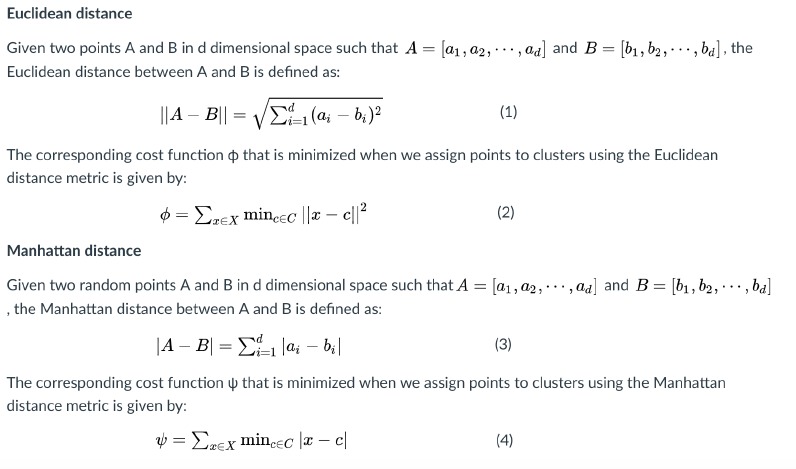
The purpose of a classifier is to take input data and assign it to one or more predefined classes or categories. For example, in a binary classification problem, the classifier would classify data into two classes, such as "positive" and "negative." In multi-class classification, the classifier assigns data to multiple classes, such as categorizing images into different types of objects.

The classifier learns from labeled training data by extracting meaningful patterns or features from the input data and building a model that captures the relationships between the features and the corresponding labels. This model is then used to predict the labels or classes of new, unseen data.

**4.4.1 KNN**

KNN, an acronym for k-nearest neighbours, is an intuitive and versatile non-parametric algorithm used in supervised learning. Unlike traditional models that learn explicit patterns from data, KNN harnesses the power of proximity to classify or predict the grouping of individual data points. This algorithm is particularly suited for classification tasks, although it can also be applied to regression problems. The underlying principle of KNN is based on the idea that similar points tend to cluster together in the feature space.

Determine your distance metrics:



As we have pre-processed our data after loading the data set:-

# Create a KNN classifier

knn = KNeighborsClassifier(n\_neighbors=5)

Creates a KNN classifier with 5 neighbors. The KNeighborsClassifier() constructor takes a single parameter, n\_neighbors, which specifies the number of neighbors that should be used. In this case, 5 neighbors are used.

Code snippet

# Train the classifier

knn.fit(X\_train, y\_train)

Train the classifier on the training set. The fit() method takes two parameters, the training features and the training labels.

# Predict the labels for the testing set

y\_pred = knn.predict(X\_test)

Predict the labels for the testing set. The predict() method takes a single parameter, the testing features.

# Calculate the accuracy

accuracy = np.mean(y\_pred == y\_test)

Calculates the accuracy of the classifier. The np.mean() function calculates the mean of the values in an array. In this case, the array contains the values y\_pred == y\_test. The == operator compares two arrays and returns an array of Boolean values. The np.mean() function then calculates the mean of the Boolean values.

Code snippet

print("Accuracy:", accuracy)

**4.4.2 Support Vector Machine (SVM)**

Support Vector Machine (SVM) is a supervised machine learning algorithm that is commonly used for classification tasks, although it can also be used for regression. SVM aims to find a hyperplane that effectively separates different types of data points. In two-dimensional space, this hyperplane is represented by a line. However, SVM operates in a higher-dimensional space, where each data point is plotted based on its features or attributes.

As we have pre-processed our data after loading the data set:-

# Create the SVM classifier

from sklearn.svm import SVC

clf = SVC(kernel='linear')

The code create a support vector machine classifier object. The SVC class is a class from the scikit-learn library that implements a support vector machine classifier. The kernel='linear' argument specifies that the support vector machine classifier will use a linear kernel.

clf.fit(data[["feature1", "feature2"]], data["target"])

The code fits the support vector machine classifier to the data. The fit() method takes two arguments: the training data and the target variable. The training data is a DataFrame that contains the feature variables, and the target variable is a column in the DataFrame that contains the labels.

# Make predictions

predictions = clf.predict(data[["feature1", "feature2"]])

The code makes predictions on the data. The predict() method takes one argument: the data to be predicted. The predictions are stored in a NumPy array.

# Evaluate the model

accuracy = np.mean(predictions == data["target"])

print("Accuracy:", accuracy)

The code evaluates the model by calculating the accuracy. The accuracy is calculated by taking the mean of the predictions that match the actual values. The accuracy is printed to the console.

Here are some additional details about the support vector machine classifier:

* Support vector machines are a type of supervised learning algorithm that can be used for classification and regression tasks.
* Support vector machines work by finding the hyperplane that best separates the data points into two classes.
* The support vector machine classifier is a powerful algorithm that can be used to solve a variety of machine learning problems.

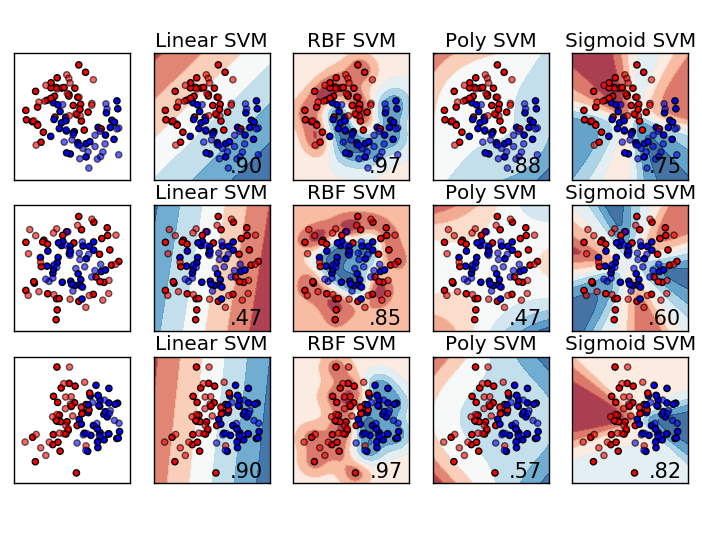


Fig4.4.2.1:Support Vector Machine

**4.4.3 Decision Tree**

A Decision Tree is a widely used and effective tool for classifying and predicting outcomes. It utilizes a tree-like structure resembling a flowchart, where each internal node represents a test on a specific attribute, each branch corresponds to the outcome of the test, and each leaf node holds a class label.

As we have pre-processed our data after loading the data set:-

#Create the decision tree

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()Create a decision tree classifier object. The DecisionTreeClassifier class is a class from the scikit-learn library that implements a decision tree classifier.

clf.fit(data[["feature1", "feature2"]], data["target"])

The code fits the decision tree classifier to the data. The fit() method takes two arguments: the training data and the target variable. The training data is a DataFrame that contains the feature variables, and the target variable is a column in the DataFrame that contains the labels.

# Make predictions

predictions = clf.predict(data[["feature1", "feature2"]])

The predict() method takes one argument: the data to be predicted. The predictions are stored in a NumPy array and make predictions.

# Evaluate the model

accuracy = np.mean(predictions == data["target"])

print("Accuracy:", accuracy)

Here code evaluates the model by calculating the accuracy. The accuracy is calculated by taking the mean of the predictions that match the actual values. The accuracy is printed to the console.

The decision tree classifier is a simple but powerful machine learning algorithm that can be used to solve a variety of classification problems. The accuracy of the model can be improved by adjusting the hyperparameters of the decision tree classifier. Some of the hyperparameters that can be adjusted include the depth of the tree, the minimum number of samples required to split a node, and the minimum number of samples required at a leaf node.

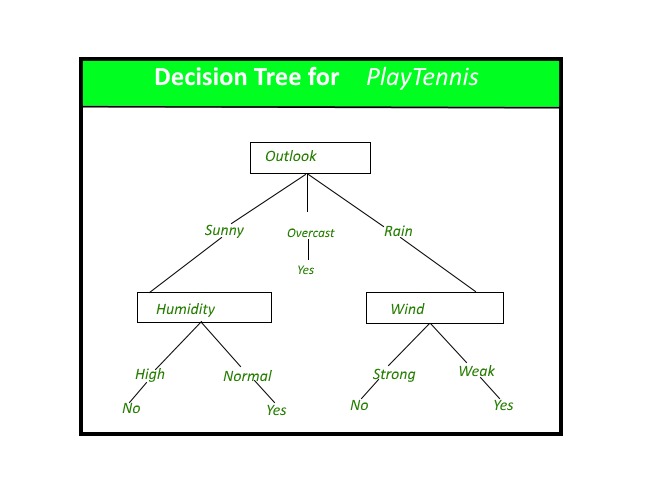


Fig 4.4.3.1 : Decision Tree

**4.4.4 Random Forest**

Random Forest is a versatile and powerful supervised learning algorithm that constructs an ensemble of decision trees through the "bagging" method. In this method, multiple decision trees are trained independently on different subsets of the training data, and their predictions are combined to generate the final result.

As we have pre-processed our data after loading the data set:-

# Create the random forest classifier

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n\_estimators=100, max\_depth=5)

The code create a random forest classifier object. The RandomForestClassifier class is a class from the scikit-learn library that implements a random forest classifier. The n\_estimators=100 argument specifies that the random forest classifier will create 100 trees, and the max\_depth=5 argument specifies that each tree will have a maximum depth of 5.

clf.fit(data[["feature1", "feature2"]], data["target"])

The fit() method takes two arguments: the training data and the target variable. The training data is a DataFrame that contains the feature variables, and the target variable is a column in the DataFrame that contains the labels.

# Make predictions

predictions = clf.predict(data[["feature1", "feature2"]])

The code makes predictions on the data. The predict() method takes one argument: the data to be predicted. The predictions are stored in a NumPy array.

# Evaluate the model

accuracy = np.mean(predictions == data["target"])

print("Accuracy:", accuracy)

The code evaluate the model by calculating the accuracy. The accuracy is calculated by taking the mean of the predictions that match the actual values. The accuracy is printed to the console.

Here are some additional details about the random forest classifier:

* Random forests are a type of ensemble learning algorithm that can be used for classification and regression tasks.
* Random forests work by building a number of decision trees and then aggregating their predictions.
* The random forest classifier is a powerful algorithm that can be used to solve a variety of machine learning problems.

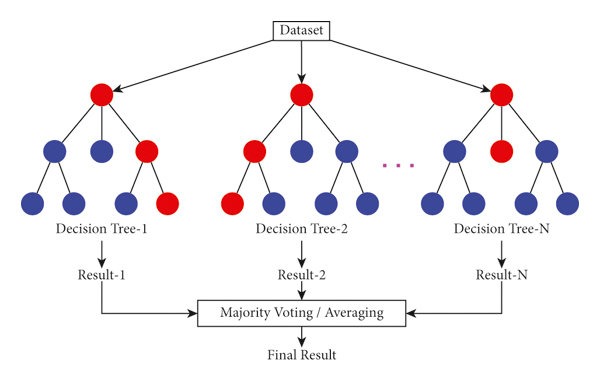


Fig 4.4.4.1 : Random Forest

**5. Results  
 5.1 Confusion Matrix**

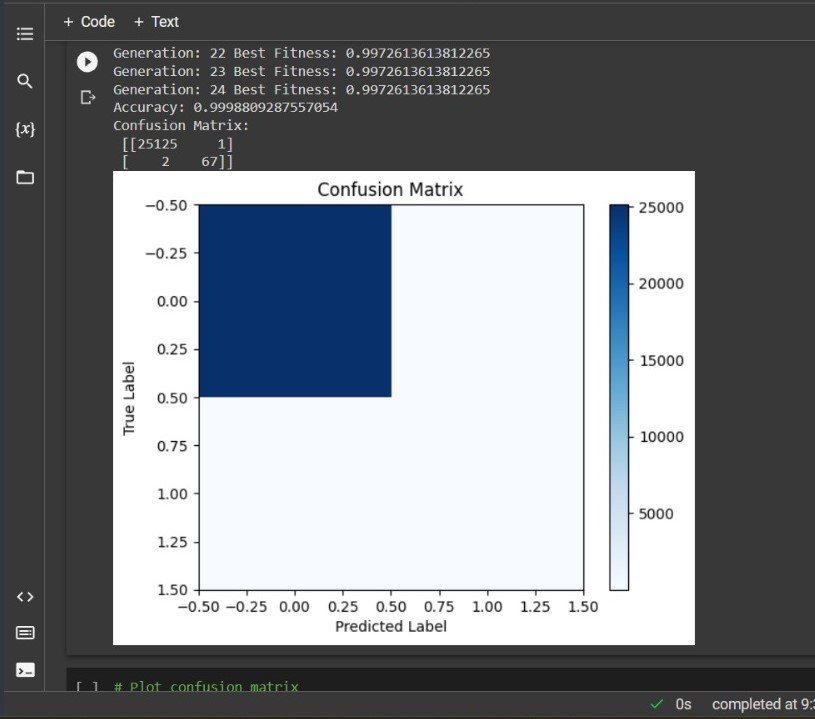


Fig 5.1.1: Population-50, Generation-25

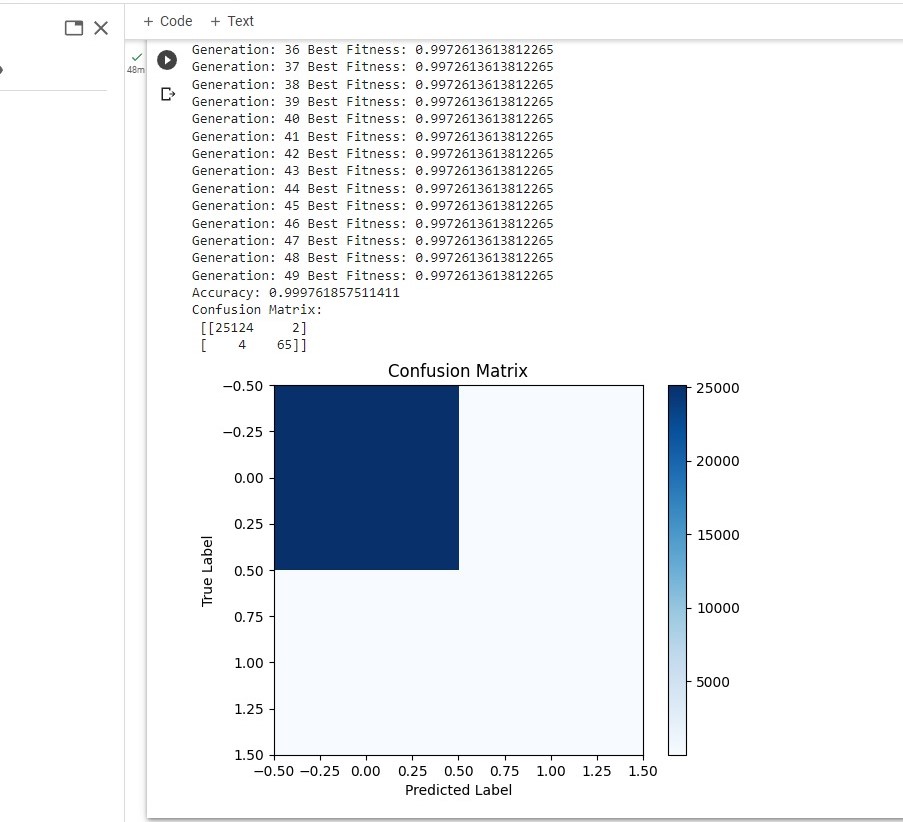


Fig 5.1.2: Population-50, Generation-50

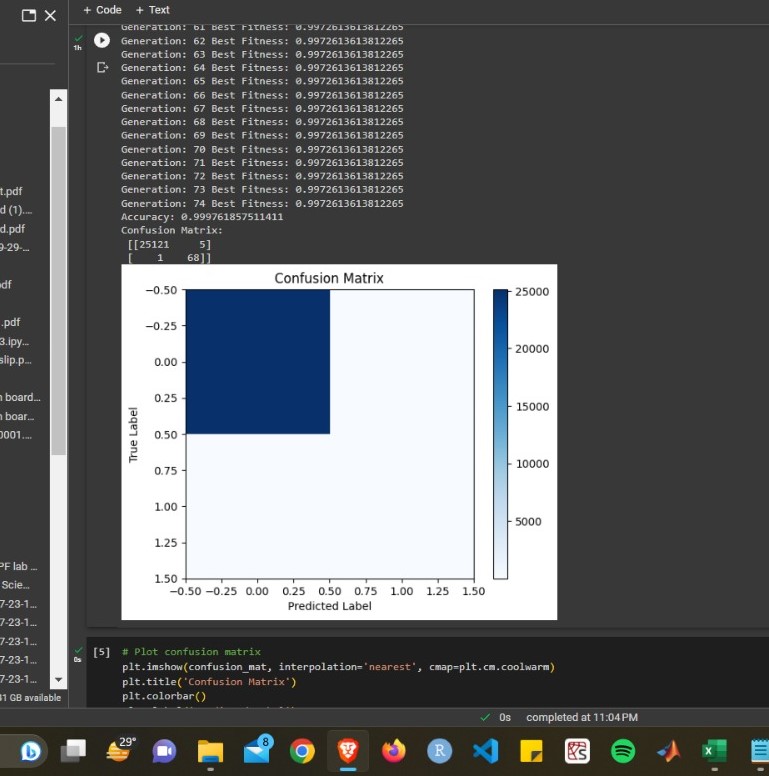
****

Fig 5.1.3: Population-50, Generation-75

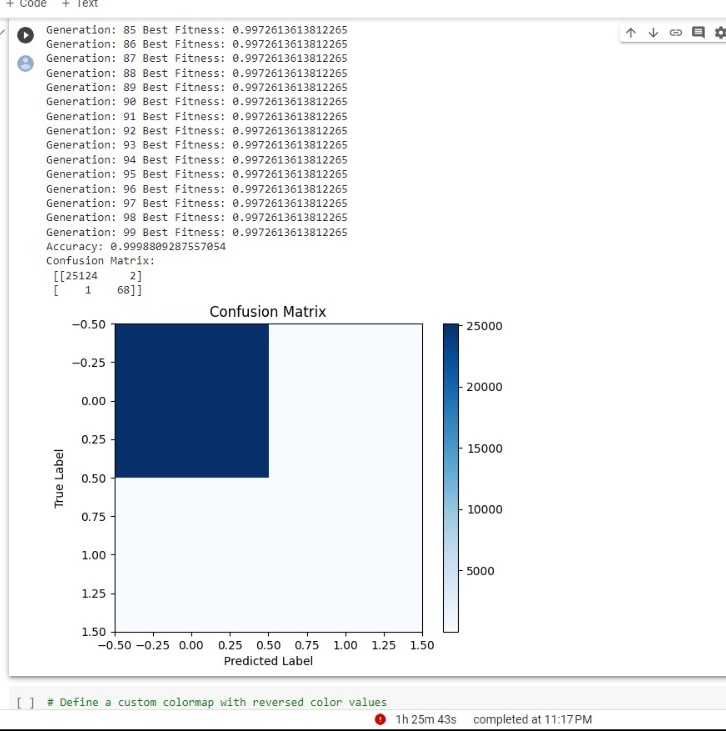
****

Fig 5.1.4: Population-50, Generation-100

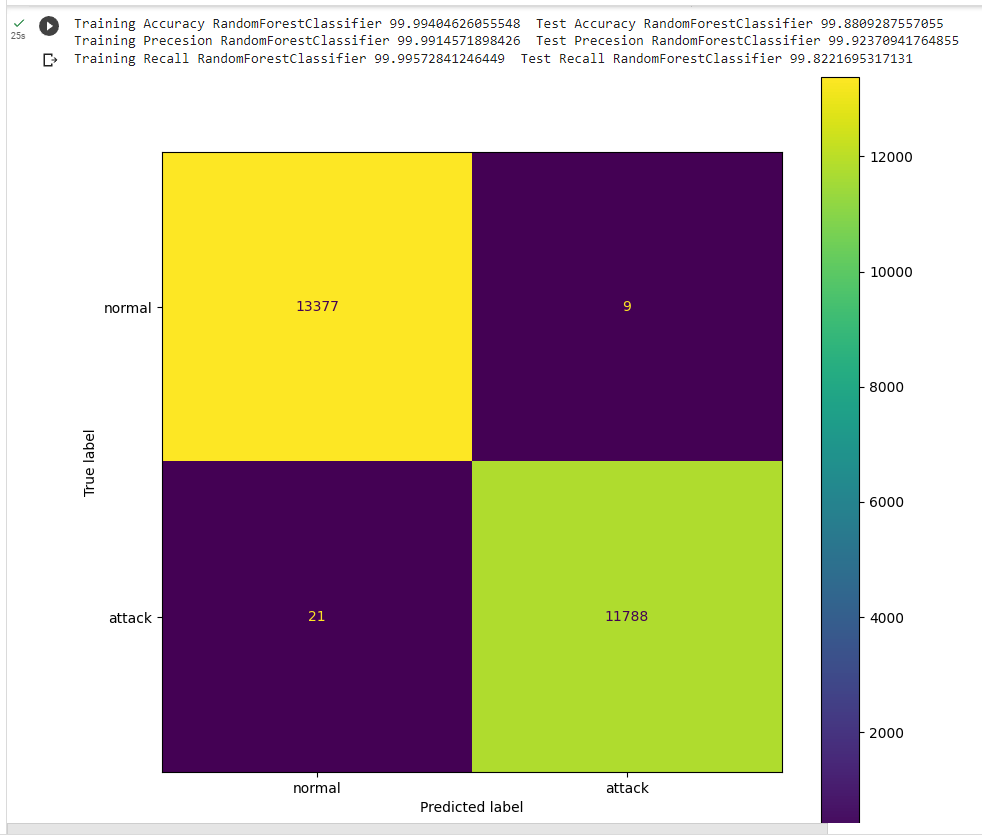
****

Fig 5.1.5: Random Forest

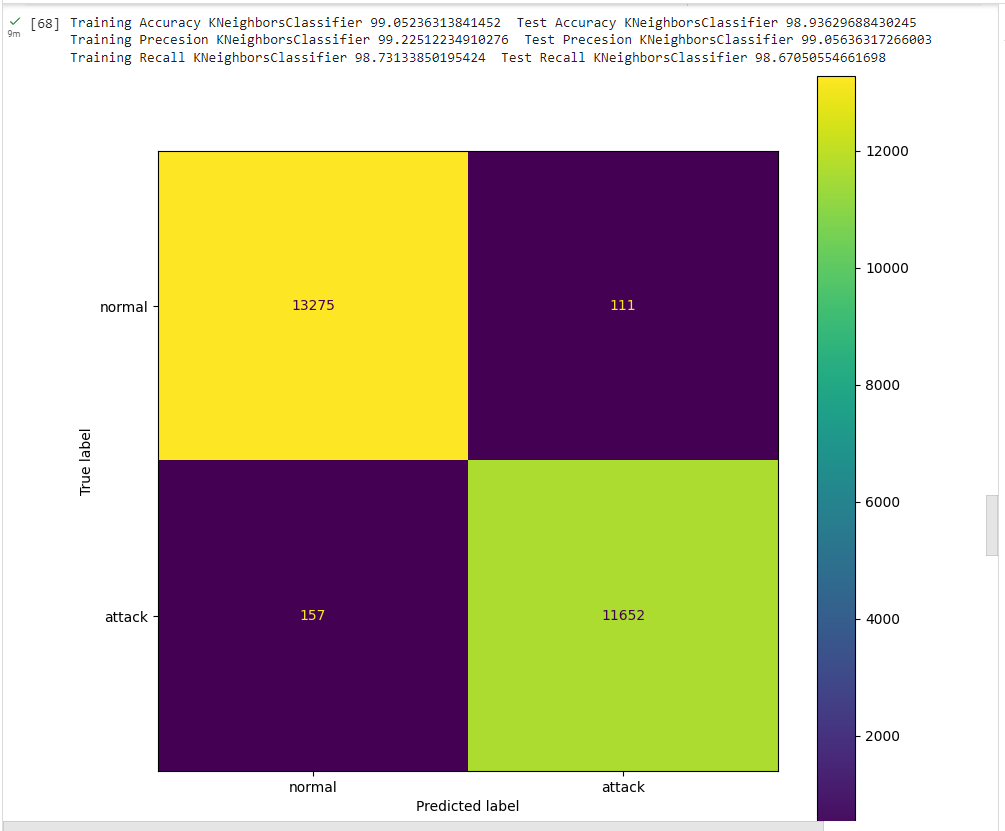
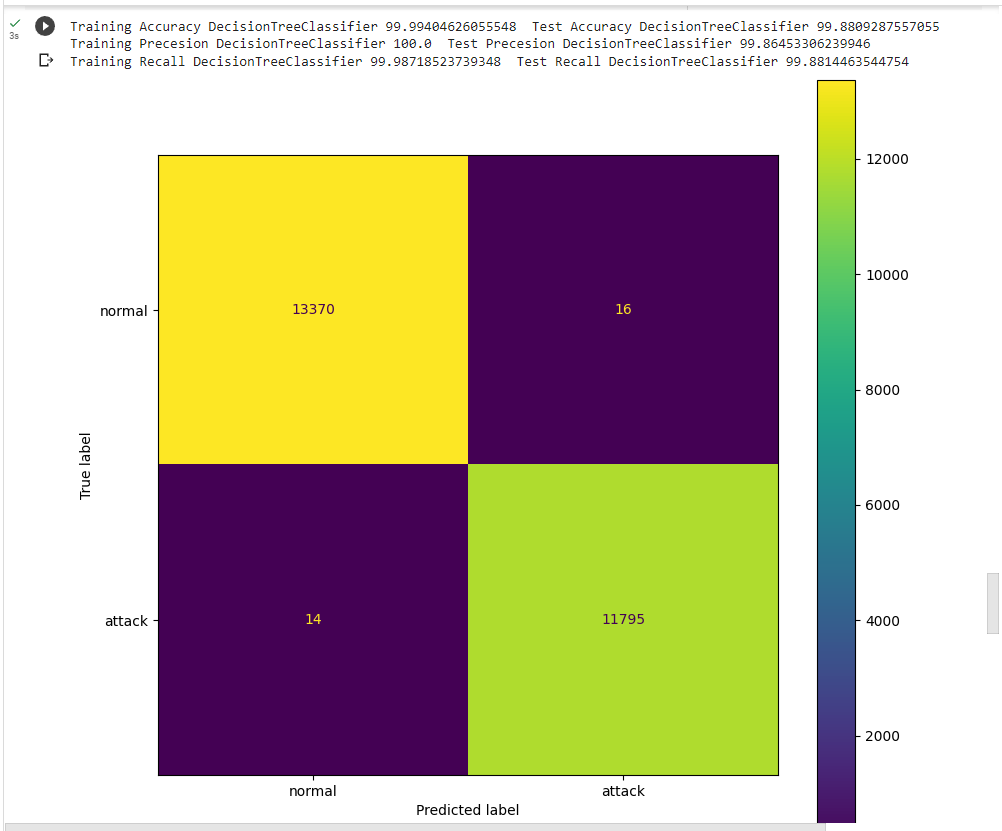
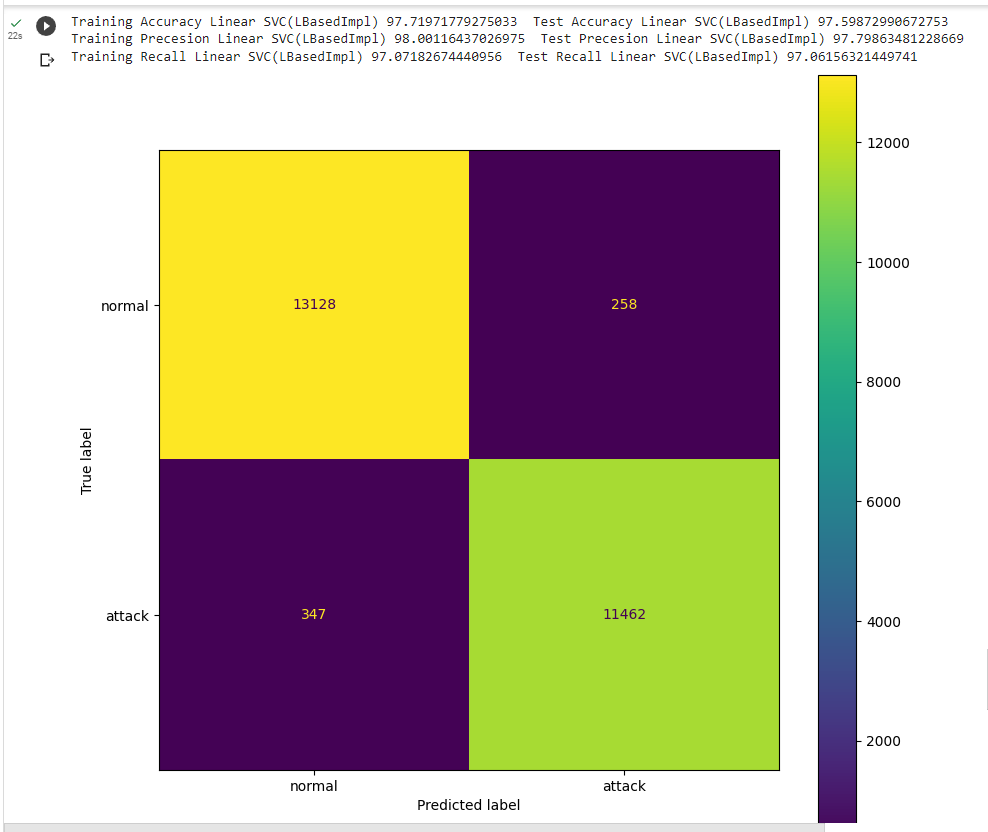
****

Fig 5.1.6: Support Vector Machine

Fig 5.1.7: Decision Tree Classifier

Fig 5.1.8: KNN

**5.2 Graphs**

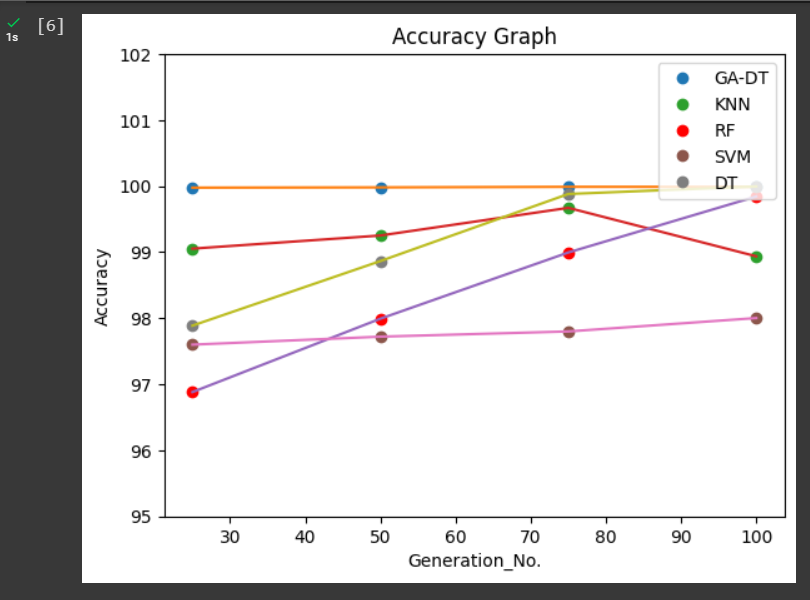


Fig 5.2.1: ACCURACY GRAPH (GA-DT,KNN, DT,SVM,RF)

**6. Conclusion**

The genetic algorithm aims to find the optimal subset of features from the given dataset to improve the classification performance. So, we have our nslkdd data set which is a benchmark dataset that was created to evaluate intrusion detection systems for network security. It is based on a large volume of network traffic data collected from a simulated environment. The dataset consists of various network connection records, where each record represents a network connection with a set of features.

The dataset used in the code is referred to as "KDD" and is loaded from a file named "KDD.txt". Unfortunately, without additional information or access to the specific dataset file, it is difficult to provide a detailed description of its contents. However, I can provide some general information about the KDD Cup 1999 dataset, which is commonly used in the field of intrusion detection and network security.

Our code provides implementation of genetic algorithm for feature selection using a decision tree classifier. The genetic algorithm aims to find the optimal subset of features from the given dataset to improve the classification performance.

The genetic algorithm iteratively evolves a population of binary chromosomes, where each chromosome represents a subset of features. The fitness of each chromosome is evaluated using a decision tree classifier trained on the selected features. The algorithm then selects the best-performing chromosomes as parents for the next generation, performs crossover and mutation operations to create offspring, and replaces the population with the new generation.

After running the genetic algorithm for a specified number of generations, the best solution (chromosome) is obtained. The selected features corresponding to the best solution are used to train a decision tree classifier. The classifier is then evaluated on the testing set, and the accuracy and confusion matrix are calculated and printed.

In the code, the dataset is preprocessed by assigning column names to the features, converting categorical columns to numerical values using LabelEncoder, and performing one-hot encoding on certain categorical columns. The dataset is then split into training and testing sets using the train\_test\_split function.

The conclusion is that the genetic algorithm successfully identifies a subset of features that improve the classification accuracy compared to using the entire feature set. The decision tree classifier trained on the selected features achieves a certain accuracy level on the testing set. The confusion matrix provides insights into the model's performance for each class, showing the number of true positives, true negatives, false positives, and false negatives. We have also compared other classification algorithms like KNN, Decision Tree, Random Forest, Support Vector Machine(SVM) in order to compare our classification algorithm accuracy with those algorithms accuracy, so that we can see the difference between them and ours.

**7. Future Work**

This future work aims to enhance the accuracy of intrusion detection systems (IDS) by exploring alternative combinations of algorithms. Traditional IDS approaches often rely on a single algorithm or a limited set of algorithms, which may have limitations in detecting complex and evolving cyber threats. This research proposes the integration of multiple diverse algorithms, such as deep learning, ensemble methods, and anomaly detection, to create a hybrid IDS that achieves higher detection accuracy and better adaptability to emerging attack vectors.

The main objective of this future work is to investigate the impact of combining different algorithms on the accuracy of intrusion detection systems. By leveraging the strengths of diverse algorithms, this research aims to enhance the detection capabilities, reduce false positives, and improve the overall accuracy of IDS. The study also explores the optimal integration of these algorithms to ensure efficient collaboration and utilization of their unique features.

# APPENCDICES

**Appendix 1: Abbreviation**

1. NSLKDD - Network Security Lab - KDD Cup
2. IDS - Intrusion Detection System
3. NIDS- Network-based IDS
4. HIDS - Host-based IDS ()
5. GA- Genetic algorithms
6. KNN- K-nearest neighbours
7. DT- Decision Tree
8. RF- Random Forest
9. SVM- Support Vector Machine

**Appendix 2: List of Figures**

1. Fig 4.4.2.1 : Support Vector Machine (Page no. 27)
2. Fig 4.4.3.1 : Decision Tree (Page no. 28)
3. Fig 4.4.4.1 : Random Forest (Page no. 30)
4. Fig 5.1.1 : Population -50, Generation -25 (Page no. 31)
5. Fig 5.1.2 : Population -50, Generation -50 (Page no.32)
6. Fig 5.1.3 : Population -50, Generation -75 (Page no. 33)
7. Fig 5.1.4 : Population -50, Generation -100 (Page no. 34)
8. Fig 5.1.5 :Random Forest (Page no. 35)
9. Fig 5.1.6 : Support Vector Machine (Page no. 36)
10. Fig 5.1.7 : Decision Tree Classifier (Page no. 37)
11. Fig 5.1.8 : Knn (Page no. 38)
12. Fig 5.2.1 : Accuracy Graph (Ga\_DT, Knn, DT, SVM, RF) (Page no. 39)

# REFERENCES

1. Alkinani, M., & El-Khatib, K. (2017). Intrusion detection system using genetic algorithm optimized neural networks. Future Generation Computer Systems, 76, 76-84.

Visit : <https://www.sciencedirect.com/science/article/abs/pii/S0167739X17303126>

1. Yildirim, T., & Akan, B. (2015). Genetic algorithm-based feature selection for anomaly-based intrusion detection systems. Security and Communication Networks, 8(18), 3854-3865.

Visit: <https://onlinelibrary.wiley.com/doi/epdf/10.1002/sec.1236>

1. Liu, Y., Liu, Z., Wang, K., & Li, Z. (2020). Intrusion detection model based on genetic algorithm and recurrent neural network. In Proceedings of the 4th International Conference on Cloud Computing and Internet of Things (pp. 29-32).

Visit: <https://dl.acm.org/doi/abs/10.1145/3442669.3442674>

1. Teymourlouei, H. M., & Movaghar, A. (2017). An intrusion detection system using multi-objective genetic algorithm. In 2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA) (pp. 1-6).

Visit: <https://ieeexplore.ieee.org/abstract/document/7991697>

1. Biswas, K., & Reddy, K. P. (2018). Intrusion detection system using genetic algorithm with artificial neural network. In Proceedings of the International Conference on Inventive Communication and Computational Technologies (pp. 732-736). IEEE.

Visit: <https://ieeexplore.ieee.org/abstract/document/8473133>