COMPARATIVE STUDY OF BIO-INSPIRED OPTIMIZATION ALGORITHMS FOR FEATURE SELECTION IN INTRUSION DETECTION SYSTEM

ABDUL AZIM BIN ANUAR VEERA

UNIVERSITI TEKNOLOGI MALAYSIA

UNIVERSITI TEKNOLOGI MALAYSIA

DECLARATION OF THESIS/PROJECT REPORT

Author's Full Name	: ABDUL AZIM BIN ANUAR VEERA		
Student's Matric No.	: A21EC0001	Academic Session	: YEAR 4 SEMESTER2
Date	: 17 JULY 2025	UTM Email	: azim02@graduate.utm.my
Thesis/Project Report Title	: COMPARATIVE STUDY OF BIO-INSPIRED OPTIMIZATION ALGORITHMS FOR FEATURE SELECTION IN INTRUSION DETECTION SYSTEM		

I declare that this thesis is classified as :
Please tick one (/)

/	**OPEN ACCESS**	I agree that my thesis/project report to be published as hard copy or online open access.
	RESTRICTED	Contain restricted information as specified by the organization/institution where research was done. *(The Library will block access for up to five (5) years).*
	CONFIDENTIAL	Contain confidential information under Official Secret Act 1972.

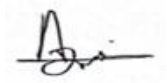
(If none of the options are selected, the first option will be carried out by default)

I acknowledge the intellectual property in the thesis/project report belonged to Universiti Teknologi Malaysia and I agree to allow this thesis/project report to be placed at the library under the following terms:

1. This thesis/project report is the property of Universiti Teknologi Malaysia.
2. The library of Universiti Teknologi Malaysia has the right to make copies for educational purposes only.
3. The library of Universiti Teknologi Malaysia is allowed to make copies of this thesis for academic exchange.

Signature of Student :

Signature :



Full Name : ABDUL AZIM BIN ANUAR VEERA

Date : 17/7/2025

Approved by Supervisor :

Signature of Supervisor I :

Full Name of Supervisor I :

ASSOC. PROF. DR. ANAZIDA BINTI ZAINAL

Date : 20/7/2025

Signature of Supervisor II :

Full Name of Supervisor I

Date :

NOTES: If the thesis is CONFIDENTAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentiality or restriction.

"I hereby declare that we have read this thesis and in my opinion this thesis is suffcient in term of scope and quality for the award of the degree of Bachelor of Computer Science (Computer Networks & Security)"

Signature : 

Name of Supervisor : ASSOC. PROF. DR. ANAZIDA BINTI ZAINAL

Date : 20 JULY 2025

COMPARATIVE STUDY OF BIO-INSPIRED OPTIMIZATION ALGORITHMS FOR FEATURE SELECTION IN INTRUSION DETECTION SYSTEM

ABDUL AZIM BIN ANUAR VEERA

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Bachelor of Computer Science (Computer Networks & Security)

Faculty of Computing
Universiti Teknologi Malaysia

JULY 2025

DECLARATION

I declare that this thesis entitled *"Comparative study of Bio-Inspired Optimization Algorithms for Feature Selection in Intrusion Detection System"* is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : .....

Name : ABDUL AZIM BIN ANUAR VEERA

Date : 17 JULY 2025

DEDICATION

I dedicated this thesis to my family, who never gave up on me but encouraged me and made me feel that I am a strong person. Thank you to my parents who have given me endless love, sacrifices and believe in me.

ACKNOWLEDGEMENT

To begin with, I would like to express my utmost gratitude to my supervisor, Assoc. Prof. Dr. Anazida Binti Zainal, who provided me with invaluable guidance on the one hand and showed so much encouragement and constructive feedback on the other, as well as her invaluable support through the course of this research. It is hard to imagine how this thesis would be without her knowledge and guidance, and her resolve and determination to endure every obstacle on the way are certainly some of the reasons why this project is thriving now. I am so glad that I had experience of working under her guidance.

I would also thank my friends and fellow students with all my heart who supported me both morally, through meaningful discussions and a helping hand whenever I felt in need in this academic adventure. This clean-up process became easier and worthwhile due to their company. It should be especially noted that people provided their time and ideas and, in such a way, helped to improve this work in one way or another.

I also sincerely appreciate the many researchers, lecturers, and practitioners who I managed to meet in the process of conducting this research. Their thoughts and insights were so valuable and contributed a lot to my knowledge and choice of direction of this research.

Finally, but not least I would like to say that I am deeply grateful to my family who accepted me with absolute love, patience and faith. This has been my perseverance and success with the support and encouragement that I receive time and time again.

ABSTRACT

The study involves the comparison of bio-inspired algorithms namely Bio-inspired algorithms (Bee Colony (BC) and Fish Swarm (FS)) in the application of feature selection in Intrusion Detection Systems (IDS), which deals with high dimensional network traffic data that tends to impair IDS performance. Based on the UNSW-NB15 dataset, the study proposes and applies BC and FS algorithms to identify the optimal subsets of features. These subsets are subsequently fed into the IDS models that are developed based on Random Forest classifier. Performance of the models is thoroughly tested using essential indicators of classification such as the accuracy, precision, recall, F1-score, and AUC-ROC, and the results averaged across ten conducted experiments to acquire statistical confidence. Visuals like confusion matrix, ROC curves and convergence plots are used to understand the behavior of an algorithm and model decision boundaries. The results identified that both algorithms, namely, BC and FS, significantly enhance the performance of detection whereby the accuracy rate of BC is a bit higher, and the feature diversity of FS is greater. Moreover, there is a test of statistical significance of observed differences. Comparative evaluation of the result has concluded that the bio-inspired feature selection is not only increasing accuracy and robustness but also saves computational overhead to IDS.

ABSTRAK

Kajian ini melibatkan perhatian perbandingan algoritma berinspirasi bio (bio-inspired) yang iaitulah Algoritma Koloni Lebah (Bee Colony - BC) dan Ikan Berkumpul (Fish Swarm - FS dalam aplikasi ciri pilih (feature selection in intrusion Detection Sistem - IDS), yang dihimpit data trafik rangkaian dimensi tinggi biasanya dipengaruhi prestasi IDS. Berdasarkan set data UNSW-NB15, kajian ini menawarkan dan menerapkan algoritma BC dan FS untuk menentukan subset atribut optimal. Subset ini kemudiannya digunakan dalam model IDS iaitu model Random Forest. Prestasi model diperiksa secara menyeluruh menggunakan cutting knowledge date: datum seperti ketepatan (accuracy), especificidad (precision), sensitividad (recall), skor F1 (F1-score), dan AUC-ROC. Keputusan daripada sepuluh uji kaji yang dijalankan telah digabungkan bagi mendapatkan kepercayaan statistik. Diagram seperti kebalikkan matriks (confusion matrix) , lengkung ROC dan konvergensi plot (convergence plots) digunakan untuk memahami perilaku algoritma dan thresholding decision dari model. Dari kajian ini dapat dihesabkan, kedua-dua algoritma, iaitu BC dan FS boleh mengukuhkan prestasi pengesanan dengan nisbah yang cukup mantab, di mana BC mempunyai ketepatan sedikit lebih tinggi manakala FS mempunyai kepelbagaian ciri yang lebih luas. Selain itu ujian kepentingan statistik juga telah dilakukan untuk perbezaan yang diamati. Pendapat perbandingan keputusan menyimpulkan bahwa pemilihan ciri berinspirasi bio menyimpan anjakan keputusan dan toleransen sistem, serta menekan beban pengiraan IDS.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IDS	-	Intrusion Detection System
BC	-	Bee Colony
FS	-	Fish Swarm
UNSW-NB15	-	University of New South Wales Network-Based 2015
AUC-ROC	-	Area Under the Receiver Operating Characteristic Curve
TP	-	True Positive
FP	-	False Positive
TN	-	True Negative
FN	-	False Negative
UTM	-	Universiti Teknologi Malaysia
PSM	-	Projek Sarjana Muda

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The protection of the computer systems and networks has turned out to be an issue of great importance in the dynamically evolving online world. Cyber threats imply a great threat to the availability, confidentiality, and integrity of information. The level of cyber threats such as viruses and malware has increased to such an extent that potent security in a rapidly expanding digital world is needed with millions of incidents said to be reported globally annually. Cyber threats have evolved and are now more advanced which makes the traditional methods very difficult.

To address such risk, Anderson proposed Intrusion Detection Systems (IDS) in 1980 and was formalized by Denning (Ishaque et al., 2023). IDS is a device applied to view and discover malicious or suspicious actions with specified areas of operation. It is the central figure in the field of cybersecurity, which examines the network traffic to raise alarms to potentially dangerous conduct. The general characteristics an efficient and reliable IDS should provide is that it should not give numerous false alarms, and should, at the same time, display high levels of detection accuracy.

Nevertheless, the growing complexity of network settings and a high dimensionality of traffic data are a great challenge. Due to this circumstance, the feature selection techniques have become crucial in streamlining IDS models and enhancing their performance (Ngo et al., 2023).

1.2 Problem Background

Cyberthreats can be dealt with by using Intrusion Detection Systems (IDS) which then help in securing networks particularly in the framework of contemporary networks, which utilize large amounts of intricate and multidimensional traffic data. Many of these data may have unnecessary, irrelevant or noisy attributes, which when unmixed may greatly reduce the accuracy of these IDS models. Whether it is high-dimensional input, the computational cost increases and the detection accuracy decreases, as well as the training and inference of the model are executed inefficiently, furthermore, when it is considered in real-time (Zhou et al., 2023; Omer et al., 2021). In the absence of scalable dimensionality reduction, IDS models become infeasible on large-scale or time-sensitive settings.

The feature selection has therefore emerged as a very important pre-processing activity in machine learning-based IDS. It also makes the input space dimensionally smaller since it only keeps the most interesting and relevant features, which results in the increased accuracy and speed of the model and better interpretability (Yahya & Mohammed, 2023; Alazab et al., 2020). Nonetheless, a numerous amount of contemporary works in IDS use techniques that deal with general-purpose feature selection, namely, correlation-based feature selection techniques or information gain, or built-in selectors LASSO or Random Forest importance. These conventional approaches do not pay sufficient attention to nonlinearity in dependencies and inter feature interactions and dataset characteristics, which result in less-than-optimal performance in detection (Feng et al., 2020).

Furthermore, modern cyber threats are adaptive in structure and behavior to evolve with time and, thus, call not only for IDS feature selection mechanisms that can scale well but also adapt to patterns in network traffic. Regrettably, most of the general-purpose algorithms cannot be adaptable and easily conformed to a real-world IDS, particularly in unsteady situations like streaming data or zero-day attacks (Kumar et al., 2023). This highlights the need to have more dynamic optimization methods that are data-driven and scalable.

To solve this, researchers have tried heuristic and metaheuristic algorithms including bio-inspired methods whereby natural occurrences like evolution, foraging, swarm behavior are reproduced to offer solutions to optimization problems. These are the Genetic Algorithms (GA), the Particle Swarm Optimization (PSO), the Ant Colony Optimization (ACO), and the Artificial Bee Colony (ABC) among others. In contrast to deterministic approaches, bio-inspired algorithms enable exploring the search space in a flexible, probabilistic manner, and in that way avoid local maxima and find the best feature sets globally (Darwish, 2018). Bio-inspired algorithms have achieved success when addressing high-dimensional optimization problems in the context of IDS even though they are heuristic.

The number of studies that confirmed the prospects of these approaches is limited. As one example, Harudin et al. (2021) showed an effective use of a Binary Artificial Bee Colony (ABC) algorithm to reduce feature sets and simultaneously achieve higher classification accuracy on IDS data. Similarly, a Hybrid Ant Colony Optimization suggested by the research of Sangaiah et al. (2022) provided better feature subsets to network intrusion detection. Yin et al. (2022) demonstrated that Recursive Feature Elimination (RFE) and Random Forest attained significant improvements in both detection precision and efficiency when they were coupled with the UNSW-NB15 dataset. Also, the Whale Optimization Algorithm (WOA) has been applied on the Bot-IoT dataset and recorded a whopping 99.5% of detection accuracy when combined with genetic enhancement (Wiley et al., 2024).

Despite this increased literature, there are still very few studies which have done comparisons of various bio-inspired algorithms. Specifically, the combination of Artificial Bee Colony (ABC) and the fish swarm algorithm (FSA) have not been largely exploited based on side-by-side comparisons in IDS feature selection. The greatest potential of these two algorithms is particularly located in ABC, which simulates the intelligent foraging behavior of honey bees, balancing exploration and exploitation to reach a good balance, and FSA, which simulates the aesthetics of fish school behavior that works based on such mechanisms as preying, swarming, and following, and can provide another form of optimization dynamics, with possible differences in the speed and stability of convergence (Chakraborty & Saha, 2020; Lu et al., 2023). Both are not only lightweight and customizable but also able to cope with

complicated search landscape, and their effectiveness in IDS applications has never been compared in detail.

To conclude, dimensional reduction of high-dimensional IDS data plays an important role in enhancing both the detection accuracy and the efficiency of a system. Although bio-inspired algorithms have demonstrated quite promising results in this field, an urgent need has been identified in terms of adaptive, scalable and comparative studies of these methods, particularly with realistic IDS data sets like UNSW-NB15.

1.3 Problem Statement

The network configuration in current times produces much high-dimensional data traffic which poses tremendous problems to the Intrusion Detection Systems (IDS) that need to work effectively and correctly in real time. In the absence of any dimensionality reduction, IDS models are exposed to the risks of slow processing, poor generalization generally, and reduced detection accuracy, especially when they contain redundant information or irrelevant features (Zhou et al., 2023; Alazab et al., 2020).

Even though the problem of feature selection has been identified as one of the central solutions, the existing studies in the concept of IDS still utilize general techniques that most likely are not optimized with regards to the peculiarities of network security data. Such approaches do not easily capture the interactions between many features and may result in an underperforming model, especially when detecting anomalies based on observations.

In the meantime, bio-inspired algorithms have been taken over as effective alternatives based on their ability to search complex, non-linear feature space. Nevertheless, their implementation has not been studied so much in relation to versatility concerning the changing nature of attacks and compatibility with the datasets of the real world. Most

of the traditional algorithms are static in their structure, which is inappropriate in such an environment of IDSs where threats are dynamic and data are highly dynamic in nature (Kumar et al., 2023).

In addition, despite a vast number of methods having been suggested as bio-inspired some, there has been a failure in providing comparative analysis on their relative performances in a well-spread experimental arrangement. Artificial Bee Colony (ABC) and Fish Swarm Algorithm (FSA) with different optimization strategies- have not been compared in a concerted manner, with respect to feature selection in IDS applications, including UNSW-NB15 dataset.

This study identifies the following key problems:

i. The IDS input feature of high dimension is costly in terms of computation, and it limits the level of accuracy in classification and therefore changes the ability of the system to detect real-time events.
ii. Although the algorithmic study in the field of optimization-based feature selection has increased, the absence of adaptive and scalable feature selection algorithms that address changing intrusion pattern in real-life datasets has remained a major setback in the efficacy of IDS.
iii. There is limited comparative study on the bio-inspired feature selection algorithms and hence this lacks clarity on the comparative power of the algorithms especially between the ABC and the FSA in the IDS domain.

By addressing these issues, this study aims to evaluate and compare the performance of the Bee Colony and Fish Swarm algorithms for feature selection in IDS, contributing to more efficient, adaptive, and high-performing intrusion detection models.

1.4 Purpose of Study

The study aims to do a comparative study of bio-inspired algorithms for feature selection in Intrusion Detection Systems (IDS), specifically evaluating the performance of the Bee Colony and Fish Swarm algorithms in enhancing IDS detection capabilities.

1.5 Research Questions

To support the study, the following research questions are asked:

i. How to efficiently apply the Fish Swarm (FS) and Bee Colony algorithms as feature selection method in intrusion detection systems?

ii. How to improve the performance of a network-based intrusion detection model by using feature selection methods based on Bee Colony and Fish Swarm algorithms?

iii. How to analyze the performance of the two algorithms (Bee Colony and Fish Swarm) as superior feature selection method?

1.6 Objectives of Study

The Objective of this project is to:

i. To apply Bee Colony algorithm and Fish Swarm (FS) algorithm for feature selection in Intrusion Detection Systems (IDS)

ii. To integrate feature selection techniques using BC and FS algorithms in a network-based intrusion detection model, respectively.

iii. To conduct a comparative analysis on the bio-inspired algorithms chosen to identify the most effective bio-inspired technique in IDS by analyzing the result of F1-score, overall performance, and detection percentage.

1.7 Scope of Study

This study was conducted within the following scope:

i. The UNSW-NB15 dataset was taken as the main dataset during the training and testing stages. It contained several classes of contemporary cyberattacks and regular traffic, which allowed a fair scenario of selection and classification activities of features.

ii. To complete feature selection before the classification process, two bio inspired algorithms were used, namely the Artificial Bee Colony (ABC) and the Fish Swarm Algorithm (FSA). These algorithms were tested based on their

capability in determining the relevant features and the capability of decreasing the dimensionality.

iii. Python language was implemented and employed libraries like Pandas and NumPy to manipulate the dataset. Scikit-learn to classify and evaluate the data, Matplotlib and Seaborn to visualize, and NumPy to manipulate the data.

iv. The accuracy, F1-score, false positive rate (FPR), true positive rate (TPR) and the AUC-ROC were selected as key-metrics to measure the performance of the two algorithms. They were applied to measure the ability to detect error rates and general strength of every algorithm.

v. The study involved supervised anomaly detection, where Random Forest was used as a base classifier. These chosen features were fed into training the model and identifying the intrusions coming in, and some specific class-specific measures, like rate of detection of the given attack type, can also be of further analysis.

1.8 Significance of Study

This study will help in advancing the area of Intrusion Detection Systems (IDS) because it will focus on developing and applying bio inspired feature selection algorithms such as, Bee Colony (BC) and Fish Swarm (FS) algorithms to improve the performance of Intrusion Detection System. By means of a systematic method using planning of these algorithms, incorporation of them and testing them to be applied on the UNSW-NB15 dataset, the study proves enhancement in precision and efficiency of intrusion detection. The results establish a strong theoretical base on using bio-

inspired algorithms in IDS, which offers methods and information that can be later used in further innovations and implementation in cybersecurity.

1.9 Report Organization

This report is organized into six chapters to present the study completely. Chapter 1 gives the introduction of the study by presenting the background, stating the problem statement, purpose of study, study questions, objectives, scope, and contributions. Chapter 2 entails comprehensive review of the literature on the topic, including the review of the Intrusion Detection Systems, preliminary feature selection methods and the use of bio-inspired algorithms such as Bee Colony and Fish Swarm algorithm. The methodology of the study is given in Chapter 3, including the description of the dataset utilized, the algorithm construction, and the pseudocode development. Chapter 4 is devoted to the implementation stage, and it represents the experimental configuration, the arrangement of detail selection methods within the framework of the IDS model. Chapter 5 covers the result of execution of both algorithms, and detailed analysis of the results. Chapter 6 is finally concluded where each of the key findings is summarized, essential contributions are indicated, and possible future research and enhancement directions are given.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Protection of information systems has added to the priority in the digital age where everything is interconnected. Cyberattacks have been increasing in complexity and rate with threat to confidentiality, integrity and availability of data in diverse fields. One of these frontline defensive measures include the Intrusion Detection Systems (IDS) which constantly inspect the system, or network to determine malicious activity (Bace & Mell, 2001; Herve, 2000). IDS may be hardware as well as software and provide vital alerting services in indicator-based identification of vulnerabilities and authentication of unauthorized access as well as maximization of situational awareness in some complicated networks (Kumar & Venugopalan, 2017).

This chapter includes a thorough review of IDS technologies and methods, particularly in feature selection to boost the detection accuracy. It underlines bio-inspired optimization algorithms, namely Bee Colony (BC) and Fish Swarm (FS) algorithms, and their efficiency in minimizing false alarms, dealing with data of high dimensions, and boosting the quality of classification in real-time settings (Alazzam et al., 2020; Pourpanah et al., 2023). This review will discuss IDS taxonomy, detection strategies, classification types of feature selection, comparison of bio-inspired approaches, evaluation measures, an algorithmic issue, and rationality of the offered solution.

2.2 Problem Formulation

The problem is formulated from general to specific. It follows by related studies and description of the identified problem.

2.2.1 Intrusion Detection System

Intrusion can be described as any unauthorized access of any computer system or network at all. These intrusions may affect the integrity of systems, result to loss of data or even create interruptions in services. Intrusion Detection (ID) attempts to detect this through automated analysis that is used in detecting deviation of the normal behaviour's patterns (Bace & Mell, 2001). IDSs are needed now more than ever to protect infrastructure in organizations that are growingly dependent on interconnected systems. They also improve security by identifying the new or current attack vectors, giving out early warning systems, and recording malicious behaviors to investigate (Kasongo & Sun, 2020). The changing nature of threats necessitates IDS to track predictable threats as well as accommodate any new attack in real time.

An average IDS has components that gather information, examine information and respond to information within the system or network. Such components operate based on long-term knowledge (known signatures or behavioral baselines), configuration and real time audit logs, to analyze observed activities to determine whether malicious activities are taking place. The three things that a detection system (IDS) can do once they detect a suspicious activity include logging of the event, alerting an administrator, and taking automated countermeasures (Herve, 2000).

Figure 2.1 is a model of normal IDS. In the most simplified terms, an intrusion-detection system is a monitor, which processes information about the system that should be safeguarded. It exploits three types of information namely long-term

information comprising the techniques applied to carrying out an intrusion detection, configuration information describing the current system state and audit information needed to describe the event. The general IDS is shown on Figure 2.1.

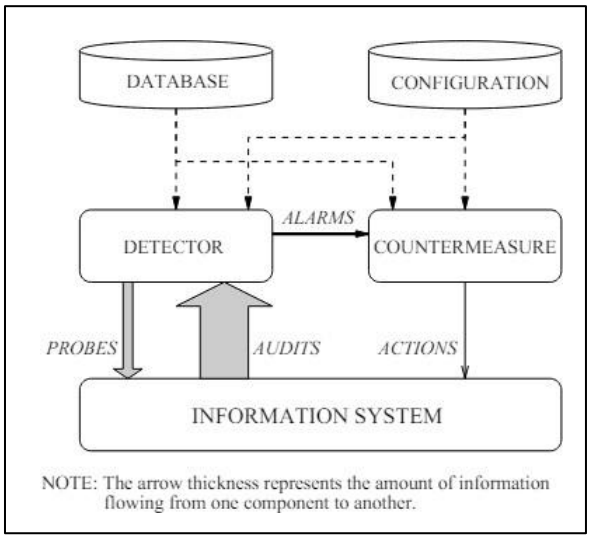


Figure 2.1 General Intrusion Detection System. (Herve, 2000)

The task of the detector is to eliminate the excess data in the audit trail. It then presents a synthetic image of security related activities carried out during normal use of the systems or a synthetic image giving the current security status of the system. It follows that one should establish whether such behaviors or prevailing conditions can be perceived as possible evidence of a breach or a weak spot. Subsequently, a countermeasure element can exercise some form of correction measure and either prevent the activities being executed or pick up the system to a secure state (Herve, 2000). There are two main types of IDS:

i. **Host-Based IDS (HIDS):** Run on user devices to implement internal process surveillance of memory input/output, file access, and system calls. It helps to find out about tampering, running of malware, or getting elevated privileges, but it can be problematic with the scalabilities in a distributed environment (Kumar & Venugopalan, 2017).

ii. **Network-Based IDS (NIDS):** Scrutinize the packets on network infrastructure with recognized or deviant signatures. They are useful in detecting threats at the network level like door scanning, denial- of-service (DoS) attacks, and diversion innovations but are subject to congestion in high-convergence settings (Kumar & Venugopalan, 2017).

2.2.2 IDS Detection Techniques

IDS detection mechanisms may be categorized into:

i. Signature-Based Detection: The method devises intrusions by matching the observed happenings to a pre-arranged database of known attack signatures. It works exceptionally well when faced with a threat that has already been seen, but it fails to detect a new or zero-day attack (Schrpretter et al., 2024).

ii. Anomaly-Based Detection: This method forms a model of normal behavior and then sounds an alarm when there is apparent anomalous behavior that could lead to intrusion. It can be effective at detecting unseen threats; however, it tends to change frequently with high false positives, and it can classify benign anomalies as an attack (Kumar & Venugopalan, 2017).

The systems used in anomaly detection in dynamic and heterogeneous environments are particularly valuable however, they are very delicate because they depend very much on the quality of input features and the quality of the feature selection process (Yahya & Mohammed, 2023).

2.3 Taxonomy of Feature Selection Techniques

Depending on their mode of operation, data requirements and the selection tactics, methods of feature selection may be categorized along many dimensions. With such knowledge of this taxonomy one can know the most appropriate approach to a particular application, in this case the IDS.

2.3.1 Learning Paradigm

i. Supervised Methods: It uses labelled data to assess the significance of each of the features by deciding how well it correlates with the target variable. The common ones are mutual information, chi-square test and information gain. They are popular in classification schemes where there are known attack labels like in IDS.

ii. Unsupervised Methods: They do not require labelled data, rather, they utilize the inherent feature such as variance, correlation or clustering structure. The techniques have been Principal Component Analysis (PCA), Laplacian scores and clustering-based selection. They can be used when you want to detect anomalies in unlabeled datasets or, in part labelled datasets.

2.3.2 Evaluation Strategy

i. Filter Methods: These are quick, easy and model-independent. They prioritize features according to some statistics. But they neglect the feature interactions and can choose the redundant features.

ii. Wrapper Methods: Wrapper methods are based on the predictive model to estimate the performances of the various feature set. They are precise as compared to filters, yet computationally costly. Examples of these are bio-inspired approaches to feature subset search during which the classification performance is used to guide search, such as BC and FS.

iii. Embedded Methods: such methods also include feature selection in the model training. Such regularization methods as LASSO (L1 regularization) automatically sets irrelevant features weight to zero, thus conducting selection.

2.3.3 Granularity of Selection

i. Individual Feature Selection: Individual features are evaluated separately. It is computationally speedy, yet it does not consider feature-to-feature interaction.

ii. Subset Selection: It is considered an evaluation of groups of features. It is more computationally involved, but preserves dependency as well as synergy among features, thus better suited to complex tasks such as intrusion detection.

2.3.4 Search Methodology

i. Deterministic Search: It makes use of a set path or heuristic to score features. A typical example includes greedy algorithms.

ii. Stochastic/Heuristic Search: Works by using randomness or bombarding the search space with a biologically inspired search. Among them are Genetic Algorithms, Particle Swarm Optimization and the object of the present work, Bee Colony and Fish Swarm algorithms.

By drawing this taxonomy, it is evident that there is no single technique that would perform better than others, but instead, the technique selection must be made based on the nature of datasets, computational capabilities, and goals of the system. Bio-inspired technique, especially the one utilized by this research determines an effective technique of wrapper-based subset selection in IDS problem (Zebari et al., 2020).

2.4 Feature Selection in IDS

The issue with Intrusion Detection Systems is that feature selection plays such an important role when developing a system, particularly when working with large feature spaces, like UNSW-NB15 or CICIDS2017. It means that it is necessary to isolate the most informative and suitable attributes within the information that lead to an efficient classification and drop an excessive number of features that can lead to overfitting or reduction in efficiency (Zebari et al., 2020).
Traditional feature selection methods include:

i. Filter Methods: These perform the evaluation of features relative to their intrinsic characteristics like statistical correlation or entropy. They are quick and model-free that can fail to consider the interactions of features (Musheer et al., 2022).

ii. Wrapper Methods: They utilize the uses of training a machine learning model over various subsets of features and choosing the subset, which performs the best. They are not computationally intensive, but this is only on small data (Yahya & Mohammed, 2023).

iii. Embedded Methods: These carry out the feature selection procedure through the process of model training. An instance of them is LASSO and decision-tree-based types of importance (Kasongo & Sun, 2020).

Irrespective of their strengths, standard methods tend to be inadequate in revealing complicated and non-linear associations in IDS data. This has triggered an increase in interest to bio-inspired optimization algorithms that usually model natural processes to find out the best set of features.

.

2.5 Bio-Inspired Optimization Algorithm

Computational algorithms related to nature Bias of algorithms meaning in computer science: the bias built into all algorithms that is inherent to classifications and tends to enforce social or other biases. Such algorithms are especially convenient to solve optimization problems when the search space is large, nonlinear, and multidimension (Darwish, 2018). They provide adaptable structures which can explore solution spaces and opportunities to exploit trade-offs. Bio-inspired algorithms can be broadly divided into:

i. Single solution algorithms: Refine a candidate solution iteratively such as Simulated Annealing algorithm.

ii. Population based algorithms: Several potential solutions are developed or traversed at once (Alazzam et al., 2020) such as Genetic Algorithms, Particle Swarm Optimization, and Ant Colony Optimization.

The Bee Colony (BC) and Fish Swarm (FS) algorithms demonstrate well-performing feature selection and optimization in the uncertain combinations of the functions of the envisaged features and therefore are gaining popularity in IDS research among the population-based methods (Chen & Sun, 2020; Kaya et al., 2022).

2.5.1 Bee Colony Algorithm

Artificial Bee Colony (ABC) algorithm is based on the activities of bees of the honeycomb. Three kinds of bees are used in this algorithm that include employed, onlooker and scout and collaborate to find optimal solutions. The worker bees search the areas of solutions, the bystanders exploit the optimal solutions by using probabilistic selection and the scouts afford diversity by creating new solutions randomly (Kaya et al., 2022).

ABC is more effective because it can escape local optima, reach balance between intensification and diversification in the search. It was used as an effective method of scheduling, cluster, and feature selection by virtue of being simple and efficient in complex searches (Kaya et al., 2022).

2.5.2 Fish Swarm Algorithm

Fish Swarm Algorithm (FS) relies on behaviors of fish, namely swarming, following and forage. Each fish is a possible solution, and its interactions set the swarm into direction of fitness with regions of fitness. Such parameters as the visual distance, the size of the steps, and the crowding factors play a role in fish behaviors of their movement and meeting (Chen & Sun, 2020; Pourpanah et al., 2023).

FS can be successfully used in multidimensional optimization. It is very adaptable, fast convergent and not sensitive to initial parameter settings hence can be used in dynamic IDS deployments where data may change indeed at a high rate.

2.5.3 Other Bio-Inspired Algorithms

Other bio-inspired techniques besides BC and FS are Genetic Algorithms (GA), in which evolution is simulated by crossover operators and mutation operators; Particle Swarm Optimization (PSO), which is inspired by the way birds flock and Ant Colony Optimization (ACO), exhibiting pheromone-guided foraging by ants. The algorithms have their advantages and disadvantages. GA has strong search abilities, may converge slowly, and PSO can be fast to converge, but premature convergence can occur; and ACO can work well to solve discrete problems, but it can fall into suboptimal paths (Darwish, 2018; Almomani, 2021).

The reasons why this study chose BC and FS are because of their demonstrated effectiveness in feature selection, ease of implementation, and flexibility in the nature of the IDS databases (Kaya et al., 2022; Pourpanah et al., 2023).

2.6 Existing Work

Many attempts have been made to understand how bio-inspired optimization methods may be incorporated into IDS to improve their precision and effectiveness. Almomani (2021) tested Particle Swarm Optimization (PSO), Multi-Verse Optimization (MVO), and Grey Wolf Optimization (GWO) as the methods of choosing the best feature subsets in IDS. It was found that, PSO and GWO showed great performance in accuracy, but they needed a slight tuning and were conversely slower at convergence. Almomani found out that to cover the application and analysis of large-scale datasets used by IDS, lightweight-based, yet effective, bio-inspired solutions are required to address the small, but disproportionate, amount of computer resources used.

In the study by Kasongo and Sun (2020), performance of the IDS models was considered based on feature selection techniques on the dataset UNSW-NB15, which is considered in the current study as well. They concluded that feature selection was important in dimensionality reduction and enhancing a good classification. They showed that the detection performance is greatly enhanced by employing an adequate subset of features, without a huge number of false alarms. This study gives a vital point of reference and even confirms the choice of UNSW-NB15 dataset as an evaluating dataset of bio-inspired algorithms.

Ferreira and Antunes (2021) investigated behavior-based IDS that applied Artificial Immune Systems (AIS) and evolutionary learning techniques. This has been attested by their benchmarking experiments, which showed that nature-inspired, adaptive algorithms would give successful results compared to rule-based, non-adaptive systems. They, however, mentioned that high-dimensional data may adversely influence the rates of detection unless the best features are selected prior to the analysis.

Wiley et al. (2024) tested hybrid optimization mechanisms based on Whale Optimization Algorithm (WOA) and Genetic Algorithm (GA). They used their models with IoT and industrial network traffic and reached detection accuracy up to 99.5 per cent. The results support the movement toward hybridization of algorithms to take advantage of multiple methods.

Table 2.1 Existing Work on Bio-Inspired Algorithms

Study	**Algorithm Used**	**Dataset**	**Accuracy (%)**	**Key Contribution**
Almomani (2021)	PSO, GWO, MVO	NSL-KDD	97.8	Compared multiple methods for feature selection
Kasongo &Sun (2020)	Feature Selection + ML	UNSW-NB15	~88–92	Demonstrated feature selection's impact on IDS with UNSW-NB15

Ferreira & Antunes (2021)	AIS, GA	KDDCup99	98.2	Showed benefit of behavior-based evolutionary systems
Wiley et al. (2024)	WOA + GA	BoT-IoT	99.5	Developed a hybrid model with superior accuracy
Kumar et al. (2023)	ACO, PSO	CICIDS2017	98.7	Applied hybrid ACO-PSO model for IoT anomaly detection

All these studies reveal that bio-inspired algorithms can be offered as a solution that can enhance the IDS performance. Nevertheless, there is a lack of direct comparisons between such algorithms as BC and FS on a generally accepted dataset, hence, there is a gap that this research is going to address.

2.7 Comparative Summary of BC and FS Algorithm

This table helps justify the two algorithms to be chosen, and it is revealed that even though these two algorithms are different in their behaviors, both share the fundamental characteristics they need to accomplish the feature selection in the IDS environment.

Table 2.2 Summary of BC and FS Algorithm

Feature	Bee Colony (BC)	Fish Swarm (FS)
Biological Inspiration	Foraging behavior of bees	Swarming behavior of fish
Optimization Style	Exploration and exploitation balance	Swarm intelligence with local/global balance
Strengths	Simple structure, effective convergence	High adaptability, less sensitive to parameters

Weaknesses	May stagnate in local optima	Require tuning visual/step parameters
IDS Usage	Proven performance in feature selection	Strong exploration behavior in IDS tasks
Common Applications	Routing, clustering, scheduling	Optimization, classification, feature selection

2.8 Challenges in Bio-Inspired Feature Selection

Although they are preferred, there are several issues that appear when one uses bio-inspired optimization algorithms in feature selection to IDS. These include:

i. Computational Complexity: A lot of bio-inspired algorithms and in particular most population-based search algorithms are computationally expensive with respect to time complexity as they require multiple fitness evaluations. This is a bottleneck in cases of large feature spaces and large datasets such as UNSW-NB15 (Alazzam et al., 2020).

ii. Parameter Sensitivity: The effectiveness of these algorithms are typically very dependent on parameters like population size, learning rates or movement constants. Insufficiently adjusted parameters may result in premature convergence or getting stuck in subpar places (Pourpanah et al., 2023).

iii. Local Optima: Algorithms are now designed to not fall into local minimums, however there will always be some depth that they end up in without propper exploration methods. Such tools as scout phases in ABC or random perturbation in AFSA are aimed to reduce it but not always enough (Chen & Sun, 2020).

iv. Scalability: The size increase in the complexity of data occurring in the datasets will cause some algorithms to scale poorly without redesign or parallelization, hence their inability to be used in the real-life implementation of the IDS.

By learning about these challenges, we have an opportunity to modify and improve these algorithms to adapt to the changing requirements of cybersecurity.

2.9 Dataset Consideration

UNSW-NB15 is a commonly used dataset by IDS community to test their mechanisms in machine learning and optimization algorithms. Created by the Australian Centre for Cyber Security, it contains more than 2 million records and 49 features of the contemporary attack scenarios (Moustafa & Slay, 2015).

Nevertheless, even though this is a very comprehensive dataset, it has the following problems:

i. Class Imbalance: Certain types of attacks are underrepresented as compared to others creating an imbalanced model unless addressed through resampling or weighted training (Kasongo & Sun, 2020).

ii. Redundancy of Features: Multiple features may include duplicate or out-of-topic information that might compromise the performance of the model, have long training time, and require much memory (Zebari et al., 2020).

iii. Dynamic Behaviour: Network conditions change with time and large and static datasets such as UNSW-NB15 might not precisely reflect on-line traffic, which is one of the reasons on why dynamic and online learning is required.

Regardless of these constraints, UNSW-NB15 is an excellent baseline, and the use of BC and FS as feature selectors can be highly beneficial to their use.

2.10 Computational Considerations in Feature Selection

Bio-inspired algorithms are effective in solving complex optimization problems but they occasionally carry substantial costs computationally and they include feature selection. This can be mainly attributed to their nature of iterations whereby, candidate solutions are continually tested to see how fit they are to go through several generations. Such repeated fitness tests may be computationally heavy, particularly as the population size or the number of generations rises. It has been found out through comparative studies that Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) usually needs more iterations to give good results. The increased length of the search process gives them increased ability to examine a greater variety of possible solutions with a greater prospect of discovering an optimal or near optimal feature set. But at the same time such more extensive investigation is at the cost of duration of operation and more computation needs.

Conversely, it has been said that Artificial Bee Colony (ABC) algorithm converges much faster. It will find good solutions within a lesser number of iterations and thus it is computation time efficient. The trade-off is, though, that it might not search the solution space as well as GA or PSO and so might not find the global optimum and so find a suboptimal feature set.

A balanced approach is provided in the Fish Swarm (FS) algorithm. It is insensitive to the setting of parameters and thus it is more robust and simpler to configure than other algorithms. Meanwhile, it has moderate to fast convergence rate. Nonetheless, moderate runtime overhead is possible because FS uses fish behavior simulation, namely following, swarming, preying, which contributes to its processing complexity (Pourpanah et al., 2023).

These differences help to demonstrate how computational efficiency, convergence behaviour and parameter sensitivity factors are to be considered, if one is applying a bio-inspired algorithm in the role of feature selection.

2.11 Proposed Solution

The proposed study aims to conduct comparative analysis of the similarity of Bee Colony (BC) and Fish Swarm (FS) algorithms on the UNSW-NB15 dataset. A Random Forest classifier is trained upon the selected features, and the accuracy, F1-score and AUC-ROC are utilized to measure the effectiveness of each algorithm.

BC and FS are selected because of their complementary optimization properties: that is, although BC is fast to converge and easy to implement, FS can be used to explore and adapt in complex spaces (Kaya et al., 2022; Pourpanah et al., 2023).

It is projected that this approach will contribute as follows:

i. Improved detection rates because of the elimination of irrelevant features.

ii. Reduce false positive rates by increasing generalization of the model.

iii. Real-life IDS out in the field.

This study attempts to contribute knowledge to the usefulness and relative disadvantages of two nature-inspired methods by comparing them directly and in a controlled experiment.

2.12 Summary

The chapter introduced the background understanding and current developments about Intrusion Detection Systems (IDS) where the issues of high dimensional data sets and the possibility of resolving them through feature selection were outlined. It explained the bio-inspired optimization methods along with Bee Colony algorithm and Fish Swarm algorithm, behaviours of these, as well as their strengths and applicability to IDS.

Existing research and evaluation metrics have also been analysed to put into practice the necessity of comparative analysis with reference to feature selection. The issues with bio-inspired approaches, including the high computational cost and sensitivity to parameters have also been addressed. The chapter was finished by justifying the usage of the UNSW-NB15 dataset and explaining the proposed approach in detail.

It is a thorough review of the literature as this book forms a foundation of the methodology, design, and experimentation requirements of the chapters to follow.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter is devoted to the construction of Intrusion Detection System with the help of using Bio-Inspired feature selection technique. This chapter outlines the research framework and the method that was employed in the accomplishment of the study objectives. The next section is the operational framework taken for the conduct of the research whereby there are three phases of the research in line with the main objectives of the study. Next, it proceeds with software and hardware needed for this study, where tools, dataset and utilized techniques are further described followed by a measure of the model's performance. Finally, the chapter ends with a summary.

3.2 Operational Framework/Research Workflow

The research design is comprised of three phases that essentially break down to the three research objectives referred to separately in Chapter 1, Section 1.6. The first step is a study and construction of feature selection algorithms. Step 2 entails incorporating identified features in IDS. Finally, the third stage will be carrying out performance measurement and comparative study between Bee Colony algorithm and Fish Swarm algorithm. Figure 3.1 and Table 3.1 give the research framework.

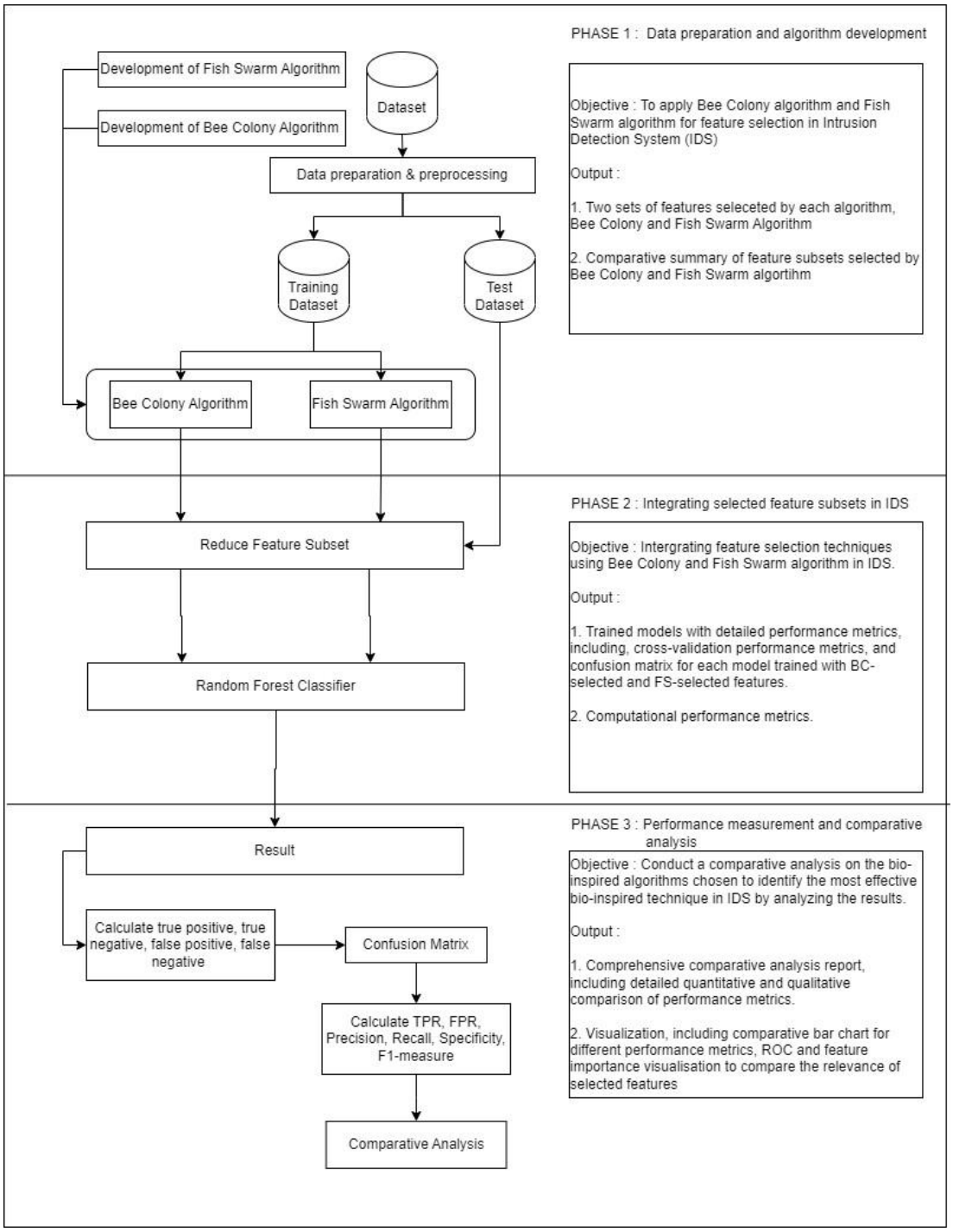


Figure 3.1 Research Framework

Table 3.1 Research Methodology

Phase	Research Objective	Research Question	Methodology	Output
1	To apply Bee Colony algorithm and Fish Swarm (FS) algorithm for feature selection in Intrusion Detection Systems (IDS)	How can the Fish Swarm (FS) and Bee Colony algorithms for feature selection in intrusion	- Applying BC and FS algorithms using Google Collab. -Optimize the dataset using the algorithms to identify best - Two sets of selected features, one from the BC algorithm and one from the FS algorithm	-Two sets of features selected by each algorithm, Bee Colony and Fish Swarm algorithm
2	Integrating feature selection techniques using BC and FS algorithms in a network-based intrusion detection model.	How can the performance of a network-based intrusion detection model be improved by using bio-inspired feature selection methods?	- Selecting a classifier - Training the chosen classifier with a training dataset. - Testing the trained model with test subsets - Compare the performance metric	- Trained models with detailed performance metrics, including hyperparameter tuning results, cross-validation performance metrics, and confusion matrices for each model trained with BC-selected and FS-selected features.

				-Computational performance metrics
3	Conduct a comparative analysis on the bio-inspired algorithms chosen to identify the most effective bio-inspired technique in IDS by analysing the result of F1-score, overall performance, and detection percentage.	Which bio-inspired algorithm, between the Bee Colony and Fish Swarm, provides better feature selection for Intrusion Detection Systems in terms of F1-score, overall performance, and detection percentage?	- Comparing quantitative results from testing results. - Graphs and reporting	- Comparative analysis report, including detailed quantitative comparison of performance metrics (F1-score, accuracy, detection percentage). - Comparative bar charts for different performance metrics, ROC and feature importance visualizations to compare the relevance of selected features.

3.2.1 Phase 1: Feature Selection using BC and FS Algorithms

The initial step of the research framework is aimed at applying and setting the Bee Colony (BC) algorithm and Fish Swarm (FS) algorithm that would allow the feature selection to take place in Intrusion Detection Systems (IDS). The setup of the initial parameters including the population size, the amount of the food sources are initiated in the BC algorithm. In the formation of the algorithm there are a few phases involved. First, Employed Bee Process where the process of forming basic solutions and their fitness. Second, Onlooker Bee Process where the process of the selection of the food sources and updating the fitness. Lastly Scout Bees Operation where the replacement of the abandoned food sources with the new random solutions. It is an iteration process that leaves the program until the convergence is realized. In the same way, the FS algorithm, parameters are defined, which contain the size of the swarm, and the step size. Using the swarm behaviours including following, swarming and dispersing, the fish which have adapted to go to new areas by walking through the new solution and thus estimating the fitness. This process can be completed in case of a convergence. These algorithms are tested on the UNSW-NB15 dataset. Based on these data, the best variants of features are determined. The information to be uploaded would include a subset of properly picked features by each algorithm, Bee Colony and Fish Swarm.

3.2.2 Phase 2 : Integration into IDS Model

The next phase would be to proceed by integrating the chosen features generated via the BC and FS algorithms into a network-based intrusion detection model based on the Random Forest classifier. The Random Forest algorithm was selected in this research because it is robust, is less biased, and can deal with data of high dimensions. The fusion process requires to train the Random Forest classifier by

means of the feature subsets chosen from BC and FS algorithms. Such training includes the cross-validation method for the problems of tuning the hyperparameters of the model to prevent overfitting and to ensure the model that will operate in the best way. Thus, metrics like F1-score, accuracy, and detection percentage are collected for each model. The results of this step present the management of the Random Forest models with a detailed report of their performance metrics, feature importance scores, and a number of model performance visualizations, for instance, ROC curves, and confusion matrix.

3.2.3 Phase 3: Comparative Analysis

The final phase of the research framework involves conducting a comparative analysis of the IDS models using the features selected by the BC and FS algorithms to identify the most effective bio-inspired technique. The performance testing phases are decorated with metrics like F1-score, accuracy, and detection percentage that are supported with statistical significance testing to measure if the disparities in the metrics are concomitant. Qualitative measures of the interpretability of different BC and FS algorithm outputs, the computational cost of BC and FS algorithms and their scalability are provided. The comparative analysis report includes not only the final reports but also a complete set of descriptive and inferential statistical analyses are carried out to support the decision. Graphical tools like comparative bar plots, ROC and PRC curves, feature importance charts are used to display the performance of different modelling approaches. The report also suggests that, in the future, the algorithms should be developed more and different methodologies to be used for feature selection in IDS.

3.3 Justification

The section explains why such hardware, software utilities, and dataset were chosen in the research. The elements are selected in a way that they are feasible, accurate, and keep up with the standard prevailing in the experimentation.

3.3.1 Hardware

The implementation and experimentation were done on a common personal computing system with Intel core i7 (11th Gen) processor, 4 GB storage space, and a solid-state drive (SSD) with a storage capacity of 512 GB. This configuration gives a decent amount of computer processing power to perform large-scale optimization problems and machine learning training using CPU alone. Hundreds of iterations of the Bio-inspired algorithms and training Random Forest classifiers could be repeated by the system without performance bottlenecks.

3.3.2 Software

The programming language used in the study is Python (version 3.10) because of the large support of libraries and the prevalence of the programming language for data science and machine learning studies. Important libraries are Pandas and NumPy to do preprocessing of the data, Scikit-learn to train and evaluate the models and Matplotlib and Seaborn used to perform visualizations. Google Colab was used as an online experiment to develop as a local development ecosystem. The tools are publicly available hence popular in academic research and can easily be shared, collaborated, and reproduced (Pedregosa et al., 2011).

3.3.3 Dataset

UNSW-NB15 Dataset was defined to overcome the lack of updated and realistic traffic datasets as KDD99 and NSL-KDD. The data was obtained from IXIA Perfect Storm tool that creates network traffic and attacks that realistically mimic the actual network environment. Normal and malicious activities were executed to collect data and make sure at least one of them was performed daily on the website. The dataset contains a mix of normal and malicious traffic, with 49 features for each network flow, divided into six categories: This includes the features Flow features, Basic features, Content features, Time features, Additional generated features and, last but not the least, the label. Figure 3.2 shows the sample of the dataset.

Table 3.2 Sample of UNSW-NB15 Dataset

dur	proto	state	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl
0.000011	udp	INT	2	0	496	0	90909.0902	254	0
0.000008	udp	INT	2	0	1762	0	125000.0003	254	0
0.000005	udp	INT	2	0	1068	0	200000.0051	254	0
0.000006	udp	INT	2	0	900	0	166666.6608	254	0
0.00001	udp	INT	2	0	2126	0	100000.0025	254	0
0.000003	udp	INT	2	0	784	0	333333.3215	254	0
0.000006	udp	INT	2	0	1960	0	166666.6608	254	0
0.000028	udp	INT	2	0	1384	0	35714.28522	254	0
0	arp	INT	1	0	46	0	0	0	0

UNSW-NB15 data set consists of nine categories of attacks as follows: Fuzzers, Analysis, Backdoors, DoS (Denial of Service), Exploits, Reconnaissance Shellcode, and Worms. The pluses of the suggested dataset can be as follows: It contains approximately 5 million entries to offer a sufficient portion of instances to train and test. They are also endorsing IPv4 and IPv6 traffic that is very imperative in

the contemporary circumstances of the network. Each of the attributes in the dataset represents a particular attribute of traffic like, the size of the packets. All the records in the data can be ascertained as normal or rather of one of the above iterations of attacks, and therefore the dataset is appropriate for the implementation of supervised learning methods to detect intrusion. The UNSW-NB15 datasets contain its high level of realism, comprehensive set of features, and the fact that it is already a benchmark in the study of cybersecurity. It provides researchers with a real and diverse dataset to analyse the behaviour of IDS methods against different situations of attacks, which increase reliability and robustness in experimentations. In addition, the high functionality of the tool facilitates the analysis and verification of models of IDSs, and it leads to the further development of network security based on new approaches to detection. The table below shows the list of attacks and the number of records for each category of attack.

Table 3.3 Type of Attack Categories in UNSWB-NB15 Dataset

No.	**Attack category**	**Description**	**Number of Records**
1	Normal	Benign (non-malicious) network traffic	560,000
2	Fuzzers	Random input attacks to test vulnerabilities	24,246
3	Analysis	Scanning, fingerprinting, spam	2,000
4	Backdoors	Unauthorized remote access tools	1,743
5	DoS	Denial of Service attacks (e.g., SYN flood, UDP flood)	16,353
6	Exploits	Buffer overflow, code injection, etc.	44,525
7	Generic	Crypto-based brute-force attacks	215,481
8	Reconnaissance	Information gathering like port scanning	13,987
9	Shellcode	Payload injection, remote code execution	1,133
10	Worms	Self-replicating malware	130

However, the primary application of the UNSW-NB15 dataset is to build and test the performance of the machine learning based IDS for network intrusion detection. They also apply feature selection and feature engineering approaches based on the dataset to get a high detection rate. Also, it provides a base for the evaluation of different intrusion detection techniques with the present trends and results. The traffic and the attacks it contains are very realistic and, thus, the dataset has a high real-world relevance. With 49 parameters, NetStud stud can present a detailed picture of the network's traffic to further its modelling.

Table 3.4 Features in UNSW-NB15 Dataset

No.	Feature Name	Description
1	srcip	Source IP address
2	sport	Source port
3	dstip	Destination IP address
4	dsport	Destination port
5	proto	Protocol used (e.g., TCP, UDP)
6	state	Connection state
7	dur	Duration of the connection
8	sbytes	Source to destination bytes
9	dbytes	Destination to source bytes
10	sttl	Source to destination time-to-live
11	dttl	Destination to source time-to-live
12	sloss	Source packet loss
13	dloss	Destination packet loss
14	service	Service used (e.g., http, ftp)
15	Sload	Source bits per second
16	Dload	Destination bits per second
17	Spkts	Source packets
18	Dpkts	Destination packets
19	swin	Source TCP window advertisement
20	dwin	Destination TCP window advertisement
21	stcpb	Source TCP base sequence number
22	dtcpb	Destination TCP base sequence number

23	smeansz	Mean packet size transmitted by source
24	dmeansz	Mean packet size transmitted by destination
25	trans_depth	Command/invocation depth into the connection
26	res_bdy_len	Response body length
27	Sjit	Source jitter
28	Djit	Destination jitter
29	Stime	Connection start time
30	Ltime	Connection end time
31	Sintpkt	Source inter-packet arrival time
32	Dintpkt	Destination inter-packet arrival time
33	tcprtt	TCP round-trip time
34	synack	TCP SYN-ACK time
35	ackdat	TCP ACK data time
36	is_sm_ips_ports	Source and destination IPs and ports are the same
37	ct_state_ttl	No. of state/TTL combinations
38	ct_flw_http_mthd	No. of HTTP request methods
39	is_ftp_login	Indicates if the FTP login was successful
40	ct_ftp_cmd	No. of FTP commands
41	ct_srv_src	No. of connections to the same service from source IP
42	ct_srv_dst	No connections to the same service from destination IP
43	ct_dst_ltm	No. of connections to the same destination IP in the last 2 seconds
44	ct_src_ltm	No. of connections to the same source IP in the last 2 seconds
45	ct_src_dport_ltm	No. of connections from same source IP to the same destination port
46	ct_dst_sport_ltm	No. of connections from same destination IP to the same source port
47	ct_dst_src_ltm	No. of connections between source and destination IPs
48	attack_cat	Attack category
49	label	Class label (0 for normal, 1 for attack)

However, the dataset also has its own limitations as it is often the case with the big data sources. The number of features and the amount of data may complicate or even slow down the preprocessing and the analysis. There may also be class imbalance problems when some types of attacks are not included or are rare. Nevertheless, it can be concluded that the UNSW-NB15 dataset is beneficial for researchers and/or practitioners in the field of network security. It offers many network traffic datasets with a realistic scenario, which are crucial for building and evaluating NIDS.

To ensure optimal performance of IDS, feature selection is highly important considering the high dimensionality, and possibly redundancy of both the features in the dataset. The project will use bio-inspired algorithms such as Bee Colony (BC) and Fish Swarm (FS) in the feature selection phase to select and hence extract the best and informative features that would determine the difference between the benign and malicious network activities. This streamlines the IDS process by providing an improved accuracy of job to be carried out, by not using the computational resources in features that do not matter and hence increases the accuracy of detection and decreases the false positive.

3.4 Performance measurement

To evaluate and compare the performance of the classification of the obtained models trained with features selected based on the Bee Colony (BC) and Fish Swarm (FS) algorithms, this paper uses several common evaluation measures. These are True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), and derived ones, including Precision, Recall, F1-Score, Accuracy, Area Under Receiver Operating Characteristic Curve (AUC-ROC). The described evaluation metrics are of paramount importance when it comes to a complex and realistic dataset, such as UNSW-NB15, which comprises normal and malicious data presenting network traffic together with malicious types, such as DoS, Exploits, Fuzzers, Shellcode, and Reconnaissance.

The results of a classification exercise are usually described in terms of a confusion matrix, an image that visually counts the numbers of TP, TN, FP and FN (Figure 3.3).

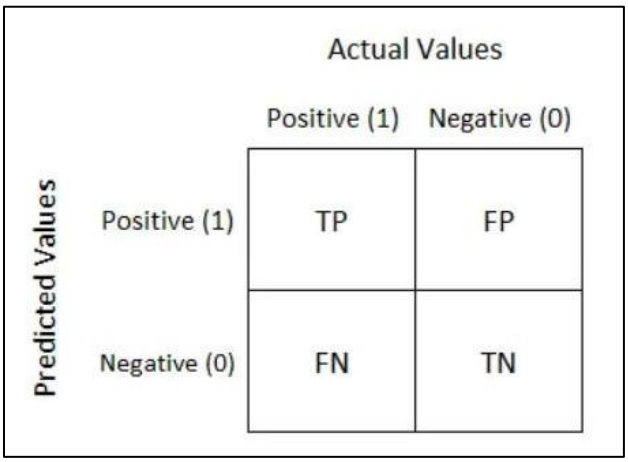


Figure 3.2 Confusion Matrix

i. True Positive (TP): It is the number of attack records which were identified as attack. As an example, in case the model identifies a record representing a DoS or Exploit attack, correctly, it is considered a true positive. The value of TP should be high, and this value shows the strength of attack detection.

ii. True Negative (TN): Refers to the quantity of normal (benign) traffic records that have been correctly identified as being normal. We have already mentioned that UNSW-NB15 consists of lots of valid traffic, and ensuring high TN rate is crucial to avoid unreasonable alerts and raise confidence in the predictions provided by the model.

iii. False Positive (FP): False Positive has occurred when benign traffic is marked as an attack. In a case where the HTTP or SSH communications that do not present any attacks or risks but are wrongfully identified as a threat, they will be included in the FP count. FP rate which is high may overwhelm the system administrators with false alarms thus making the IDS to be less efficient in its operation.

iv. False Negative (FN): This is the traffic of an attack which is falsely recognized as a normal. It is a serious kind of error because this could mean that a Shellcode or other intrusion type, such as reconnaissance intrusion activity has been able to go unnoticed and thus give attackers an opportunity to get around security systems.

False positives and false negatives are important measures in an IDS, which is why several performance measures are used in evaluating model quality. Among them, F1-Score is highlighted in this work as the predominant measurement because it ends up consuming both Precision and Recall, especially on data, containing imbalances in classes, as the UNSW-NB15.

i. Precision compares the number of correct predictions of attack records with all the records labelled as attacks.

$$Precesion = \frac{TP}{TP+FP} \tag{3.1}$$

ii. Recall (also named Sensitivity or True Positive Rate) is used to quantify the amount of the true attacks that were indeed recognised by the model.

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

iii. F1-Score is detected as a harmonic ratio of Precision and Recall. It can also be of special value when the false positive, the false negative, are both costly, and this is true in intrusion detection.

$$F1 - Measure = \frac{2(Recall \times Precision)}{Recall + Precision} \tag{3.3}$$

The other important test of evaluation that has been used in this research is the AUC-ROC. The metric reflects the overall ability of the model to discriminate between attack and normal traffic irrespective of the classification threshold adopted. The closer the AUC is to 1.0 the better performing the classifier. ROC curve is drawn against True Positive Rate (TPR) and False Positive Rate (FPR) in various thresholds.

The ROC curve is a probability curve and the trade-off between sensitivity and specificity is depicted. Area Under the Curve (AUC) is the rating of how well (or to an extent) the model could decide on the levels of classes. A model that matches or exceeds an AUC of 1.0 is assumed to have very good classification power and one that has a value of 0.5 indicates that the model performs no better than random guess.

Using these standards on the UNSW-NB15 dataset, the study can be sure that the accuracy and the actual usefulness of the models feature selection, and classification will be objectively evaluated. Such visual representation as confusion matrices and ROC curves also help to interpret and compare outputs of the BC and FS algorithms.

3.5 Summary

This chapter contains an exhaustive inspection of the approach of an Intrusion Detection System (IDS) using Bio-Inspired feature selection techniques, the Bee Colony (BC) algorithm and the Fish Swarm (FS) algorithm that are implemented. A general introduction of the research framework, and the instruments that have been employed towards the accomplishment of the objectives of the study will be offered in the chapter. The whole process of research was divided into three varying stages. During the initial step, the UNSW-NB15 was tried to implement feature selection (BC and FS algorithms). The BC algorithm applied three sets of Fig. and onlooker bees and scout bees to solve the problems of systematic acquisitions of the best solutions whereas the FS algorithm applied swarm behaviour in defining the search of optimal feature sets.

The second step is the usage of the Random Forest classifier into the IDS model after selecting the features of the BC and FS algorithms. This step involves tuning and parameterization of the hyper parameters by the help of cross-validation that allowed to record performance rates like F1-score, accuracy, and the percentage of the detector. The outputs were trained models with performance scores, feature importance scores and other visualizations such as ROC curves and confusion matrices.

CHAPTER 4

RESEARCH DESIGN AND IMPLEMENTATION

4.1 Introduction

In this section, the research design as well as implementation of the proposed resolution of the Intrusion Detection Systems (IDS) enhancement through bio-inspired feature selection algorithms has been offered. The flow of the chapter is in three main stages as presented in the research objectives of Chapter 1 and the methodology presented in Chapter 3.

The initial step is devoted to the development of two bio-inspired algorithms Bee Colony (BC) and Fish Swarm (FS) that will comprise the selection of the most relevant features within the high-dimensional UNSW-NB15 dataset. These are algorithms which emulate natural behaviours so as to maximize feature subsets on the basis of the classification performance.

To the second step, the identified features are loaded to an IDS model generated with Random Forest as the classifier. This model is trained and tested to check increase and efficiency in accuracy of detection.

Phase three is evaluation of the performances of the models and comparative analysis of the model based on accuracy, F1-score, AUC-ROC, and confusion matrix. Every step is described according to its preparation, execution, and results which give a perfect outline to prove the rightness of the way the new approach is suggested.

4.2 Proposed Solution

This research is divided into three phases, which are interrelated and different but consist part of the objectives of the research mentioned in Chapter 1 and realisation of the methodology mentioned in Chapter 3.

i. Phase 1: Feature Selection using Bio-Inspired Algorithms

The first step seeks to address the research problem of high-dimensionality and irrelevant feature in IDS that tends to reduce model performance. Bio-inspired optimization algorithms provide an adaptive capability to search, as explained in Chapter 3, which has natural similarity to nature. Two of such algorithms were established in this work, Bee Colony (BC) and Fish Swarm (FS), to detect the most important attributes in the data set UNSW-NB15. Every subset of a feature was a binary mask and was compared with fitness using the Random Forest classification scale. The result of this part is two optimized subsets of features optimized by BC and FS respectively, and the convergence patterns of them by iterations.

ii. Phase 2: Integration into Intrusion Detection System

In this stage, the candidates emphasize the incorporation of the feature sets that they have chosen into a classification model. As clearly indicated in Chapter 1, an efficient classifier is essential in the intrusion detection exercise. So, the popular ensemble learning approach Random Forest, which is known to be accurate and robust, was selected. In the two different models of IDS, two different IDS models were trained using the chosen features of BC and FS. Training and testing of the models were done in cross-validation about feature selection, and fully in the 32% testing set. The results of this stage are trained

IDS models with their measurement indicators accuracy, F1-score, precision, recall, and AUC-ROC.

iii. Phase 3: Comparative Evaluation and Analysis

In the third step, a thorough comparison on the performance offered by both the models is drawn. This step tackles the last research purpose that includes finding out which algorithm has the best feature selection in IDS. The differences in accuracy of detection, computational efficiency and feature reduction ability are studied using statistical measures and plotting tools. The outcome of this stage is a comparative performance report which is used to make future recommendations and validate effectiveness of the solution

4.3 Experimental Design

In this section, we present the concrete manner, in which each stage of the given research was effectively realized, indicating methods, tools, and settings to make the experimentation procedure reliable and repeatable.

4.3.1 Phase 1 : Research and Development of algorithm for feature selection

Phase 1 exploits the Bee Colony (BC) and Fish Swarm (FS) algorithms for malefaction in the Intrusion Detection Systems (IDS) scenario. initial interaction in getting the UNSW-NB15 dataset with different types of network traffic and attacks that were already available was first operated. The dataset's preprocessing is conducted

with the help of Python libraries like Pandas' Pandas and scikit-learn for the case of handling missing values, standard features, and encoding categoric attributes.

The first step is obtaining the UNSW-NB15 dataset, which the model analyses in the subsequent stages. The data is pre-processed to eliminate any issues. This entails working on the missing values, mandate that the numerical features are normalized, as well as the encoding of categorical variables. Doing these steps is not only important in regularizing the dataset, but that is also, making sure it accepts the algorithm run it smoothly making it a fool-proof data set. The Figure 4.2 below shows the snippet of code used to do the pre-processing of the dataset.

After the completion of preprocessing, both Bee Colony (BC) and Fish Swarm (FS) algorithms were initialized with parameters to activate the optimization in the process. The number of the individuals in the population was equal to 20, and the total number of iterations was 200 in both algorithms. Stopping criteria suggested the stopping condition when the limit of iterations was reached or when no progress could be made on the best solution over 20 iterations.

Both sets of algorithms (the optimization processes of both algorithms) have started with randomly assigned candidate solutions - different combinations of chosen features. An objective fitness criterion was then determined on these sets of candidate features based on a pre-determined fitness, in the present case, the classification accuracy of a Random Forest (RF) classifier trained on the candidate feature set. To present trustworthy and impartial performance values, the 3-fold cross-validation method was applied in the evaluation process of the classifier. The fitness was calculated as an average of the accuracy score of folds. The measure was used as the key goal during search of the best feature subsets. Along with the accuracy, some secondary metric (such as F1-score, false positive rate (FPR), detection rate) were measured during the testing stage but not directly employed in the fitness function itself.

In the iteration part, BC and FS reinforced their respective candidate solutions according to the swarms they belonged to. How this exploration was performed in Bee Colony algorithm is that the employed bees exploited the local neighbourhoods, onlooker bees searched the solution space under fitness-proportional selection and also

scout bees explored new spaces of the search space to ensure diversity is maintained. The same applied in the Fish Swarm algorithm where either individual movement, swarming behaviour, and following behaviour enabled the fish agents to move to improved solutions and eliminated premature convergence among them by random resets of their positions in case of stagnation.

The phenomenon of convergence was tested from one iteration to the next, which is generally determined from the maximum number of iterations or when a certain degree of fitness improvement is reached. The Upon completion of the convergence conditions, certain features are retained, and performance metrics are recorded. Visualization of the process of features selection is also made, which lets the observer understand better what the BC and FS algorithms are doing when they select relevant features for IDS.

Both algorithms are implemented in Python, along with libraries such as Pandas to manipulate data, scikit-learn. The results of phase 1 could be observed in the form of the feature subsets, the description of the execution of the algorithm as well as in the visualizing efforts of how the feature selection is being performed.

4.3.2 Phase 2: Integrating selected feature in IDS

This phase was aimed at comparing the usefulness of the assigned feature subsets of the chosen Bee Colony (BC) and Fish Swarm (FS) algorithms that relied on a congruous and stable framework of classification. This task was done using Random Forest (RF) classifier that was selected because it scales to high-dimensional data, does not overfit, and gives excellent results in its application in intrusion detection activities.

Individual Random Forest-based models were trained using the set of final selected features in turn per bio-inspired algorithm. The models were tested based on a two staged certification approach:

i. The training set was cross-validated 3-fold to estimate the performance of the given model on new data and determine the fitness score in the process of feature selection.

ii. The final evaluation of performance of the models after training was performed on 32% holdout test set (not available during training and in the feature selection) in a simulated real-world environment.

The Random Forest classifier was set up as a classifier and 100 decision trees were used, the rest of the parameters were left with the default settings in the Scikit-learn. The cross-validation was done on independent models trained using the smaller feature set provided using BC and FS methods to provide accurate and isolated assessment.

The evaluation of the classification performance was carried out with the help of the following important metrics:

i. Accuracy: Ratio of correctly categorised cases.

ii. F1-Score: The harmonic means of precision and recall, which can be applied in case of an imbalanced data set.

iii. Precision and Recall: To measure the false alert and detection.

iv. False Positive Rate (FPR) and True Positive Rate (TPR): In order to measure the level of error.

v. Area Under the ROC Curve (AUC-ROC): To quantify the trade-off between the TPR and FPR varying across thresholds.

The objective of this strategy was to make sure that training of the IDS models was done using the most significant and optimized features thus enhancing the detection capability and reducing the number of false alarms as well as enhancing the

computation efficiency. A result of the two algorithms on all these metrics was the basis of the comparison of the performance of the two algorithms.

4.3.2.1 Environment Setup

Cloud and local environments were used in the development and implementation of the models to allow proper reproducibility and performance. The specification was as followed:

Table 4.1 Setup Specification

Component	**Specification**
IDE	Google Colab (Cloud-Based)
Programming Language	Python 3.10
Hardware (Local)	Intel Core i7, 4 GB RAM, 512 GB SSD
Libraries Used	pandas, numpy, matplotlib, seaborn
Machine Learning	scikit-learn (RandomForest, metrics, CV)

Google Colab was enough, supported the use of CPUs, and was linked to Google Drive to deal with data. Data processing, feature selection, model training, evaluation and visualization were performed using python libraries.

4.3.2.2 Dataset Preparation and Pre-processing

UNSW-NB15 is selected as the dataset of this research because it is realistic due to its network traffic representation, modern attack type and multi-class structure as explained in Chapter 2. There are 45 features in the dataset with 39 numerical columns and 6 categorical columns, like, proto, service, state, attack_cat and label. The data had been divided into two CSV files:

i. UNSW-NB15_training-set.csv (175,341 rows, 68%)
ii. UNSW-NB15_testing-set.csv (82,332 rows, 32%)

This 68:32 ratio was kept in the experiment to be consistent and aligned to the view of the original authors in their intended distribution and to give ample samples during the training and testing exercises. As preprocessing, the following was used:

i. Dataset Loading the Pandas was used to load both the training and testing CSV files and merged them together to create common preprocessing protocol and resorted in splitting again with training and testing based on the original split quotient.

ii. Column Filtering The column id, label, and attack_cat were taken out of the feature set temporarily to avoid the label leakage as the features were selected. The label was kept as the classification target, and only attack_cat was taken into consideration in the analysis of the evaluation.

iii. Numerical Filtering Since both algorithms, Bee Colony (BC) and Fish Swarm (FS), are tailored to work only on numerical data, the dataset had to be cleaned on all non-numerical (categorical) features. The non-numerical columns were 6: proto, service, state, attack_cat, label and id. It left with a dataset composed of 39 numeric variables only. Such categorical features as proto and state were not transformed to numerical form (through one-hot encoding), as is done in this study to prevent the exponential scaling in number of features. Nevertheless, these can be integrated in further research to identify the patterns at the protocol level.

iv. Normalization The Min-Max Normalization technique of the MinMaxScaler class of Scikit-learn was applied to all numerical features to normalize them in the range of [0, 1]. The formula is the following when scaling this way:

$$x' = \frac{x - x\,min}{x\,max - x\,min} \tag{4.1}$$

where:

x = original feature value

xmin = minimum value of the feature in the dataset

xmax= maximum value of the feature in the dataset

x'= normalized feature value

This normalization allowed all the features to play an equal part in the optimization process and features with a greater magnitude would not be favoured.

i. Dataset Re-Splitting At the end of the preprocessing, the split training and testing sets used in the original 68% training and 32% testing split as given in the downloaded dataset were used to split the dataset back into training and testing sets. This datum made sure that the assessment would be relevant to the data distribution as laid out by the developers of UNSW-NB15.

ii. Pre-processed Output The label column in the cleaned data was the binary label as the finite target, and there were 39 numerical features, which were all normalized features. This data would then be passed through BC and FS feature selection algorithms and then Random Forest used to do classification.

```
Load training and testing CSV files
Drop 'label' and 'attack_cat' columns from both sets
Extract features (X) and labels (y)
Select only numeric columns
Scale features using Min-Max normalization
Store feature names for reference
```

Figure 4.1 Pseudocode (Load and Preprocess Data)

4.3.2.3 Model Development and Training

Once the feature selection process was executed successfully in the Bee Colony (BC) and the Fish Swarm (FS) execution histories, feature subsets were generated and served to train Random Forest classifiers with these feature subsets as inputs to the training process. The two models of IDS were designed-one with the features chosen by BC, and the other one with the features determined by FS. This created a fair comparison of the role played by both algorithms on the overall performance of classification.

In the feature selection procedure, each algorithm was provided in-built 3-fold cross-validation that was used to determine the fitness of the set of candidate features. Such internal validation really assisted in the optimization procedure to achieve more efficient combinations of features on grounds of classification accuracy. After selecting the best feature subsets, the models were finally trained using the complete

training dataset using the selected features. The number of decision trees in each model was set to 100 (n_estimators=100) so that prediction performance and computation time trade-offs could be used. The Figure below shows the coding that was used to do the training.

```
Define function train_and_evaluate():
    - Train Random Forest using selected features
    - Predict on training set
    - Print Accuracy and F1-score
```

Figure 4.2 Pseudocode (Training Random Forest Model)

A fair evaluation of the models was achieved by evaluating the models on the already held out 32% test set that was not utilized in any of the levels during training or feature selection. Both the training and the test labels were predicted with the help of trained models, and the results in the form of outputs were analysed with a range of performance metrics. These were accuracy, precision, recall, F1-score and AUC-ROC. Also, there were graphical outputs in forms of ROC curves, confusion matrices and convergence plots to visualize models' behaviours and progress of optimization. The coding to make the results are as shown in Figure 4.6 below.

```
Define function final_evaluate():
    - Predict and get probabilities on test set
    - Compute Accuracy, Precision, Recall, F1, AUC
    - Print TP, TN, FP, FN
    - Return all metrics

Define plot functions:
    - plot_convergence(): shows accuracy over iterations
    - plot_conf_matrix(): shows confusion matrix
    - plot_roc_curve(): shows ROC curve
```

Figure 4.3 Pseudocode (Final Evaluation)

All results generated in this stage, such as model objects, evaluation metrics, and visuals were stored and prepared to be used in Phase 3, during which relative performance of both algorithms would be observed thoroughly.

4.3.3 Phase 3: Performance measurement and Comparative Analysis

The last step of the experiment aimed at testing and comparative analysis of the two sets of selected feature subsets that led to the creation of the two models of the IDS based on the Bee Colony (BC) and Fish Swarm (FS) algorithms. In this step, the researcher wanted to establish which of these two algorithms produced more precise, effective and general outcomes in relation to intrusion detection. The analysis was crucial in response to the third research question and confirming effectiveness of the proposed feature selection methods presented in the Chapters 1 and 3.

To viably compare the models without any bias, they were tested on the final test set, which was not used at any moment during training or feature selection. To measure the quality of classification several performance metrics were applied, True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). On these values, core metrics accuracy, precision, recall, F1-score, and AUC-ROC (Area Under the Receiver Operating Characteristic Curve) were calculated to give a complete picture of performance of each model.

Besides quantitative measures, the assessment involved visual inspection to contribute to the further explanation of model behaviour and performance. Heatmaps of confusion matrices were employed that can give an idea on how the classifications are done, thus, determining where the strengths or weaknesses of the predictions of each model are. ROC curves gave a clue on how each model discriminated attack and normal traffic. The charts on feature importance showed which of them were critical

to the decision making in the model. In addition, the convergence plots were employed to show development of solutions by each algorithm in the feature selection process.

All these assessment measures enabled the research to make apt conclusions about the comparative strengths of the BC and FS algorithms. This comparison gave us very important lessons to further research and to implement in real-life IDS deployment especially in situations when accuracy, efficiency and adaptability of the deployment is of utmost importance.

4.4 Parameter and Testing Method

It is in this section that the parameters upon which both bio-inspired algorithms and evaluation criteria are performed in observing the performance of IDS are given. The parameter and metrics selection fits the methodology in Chapter 3 and complies with the goals mentioned in Chapter 1.

4.4.1 Study Parameters – Bio-Inspired Algorithm Configuration

The Bee Colony and the Fish Swarm algorithms are both based on population-based searching algorithms. To draw a fair and consistent comparison the following parameters were set:

Table 4.2 Parameter values for algorithms

Parameter	**Bee Colony (BC)**	**Fish Swarm (FS)**
Population Size	10 Bees	10 Fish
Max Iterations	200	200
No Improvement Limit	20 Iterations	20 Iteration
Scout Bee Limit	10	N/A
Visual Distance	N/A	5
Step Size	N/A	1
Crowd Factor	N/A	0.6

4.4.2 Model Parameters – Random Forest Classifier

Random Forest algorithm was the chosen classification model which was used in training and evaluation of the dataset; it is robust and working fine with the high-dimensional data. The following parameters were established to encounter a steady comparison between models:

Table 4.3 Model Parameters

Parameter	**Value**
Classifier	Random Forest
Number of Estimators	100
Cross-validation	3-Fold
Evaluation Metrics	Accuracy, F1, AUC, TP, TN, FP, FN
Train-Test Split	68%Training: 32% Testing

These set ups made the comparison between the two models to take place under the same conditions.

4.4.3 Testing method and justifications

In this experiment, both algorithms Bee Colony (BC) and Fish Swarm (FS) were run ten times independently. They were not repetition folds and partial simulations, but full and independent ones. A new random seed was used to reinitialize the algorithm in each run; and feature selection was done each time. A Random Forest classifier was thereafter trained and tested with the newly chosen feature set. Accuracy, F1- score, Precision, Recall, and AUC ROC among others were gathered to evaluate the performance metrics. This standalone preparation will certify that the results are adequate in representing the capabilities of each algorithm to always generate the best or near-optimal feature sets using different random initial starts. The choice of doing the experiments ten times was dictated both by methodological soundness issues and practical reasons.

i. Statistical Reliability with Computational Efficiency

A balance between the costs and statistical soundness is achieved by using ten independent runs. The performance variability can be captured and not bias by random initialization using averages over 10 runs as done in comparable work (Pourpanah et al, 2023). Mean and standard deviation gives a significant comparison in stability and robustness which can be done. The results and calculations can be found in Appendix F.

ii. Heavy Computational Cost per Run

Time-consuming population-based search, consecutive fitness assessment, and training Random Forest on a high-dimensional dataset (UNSW-NB15) were involved in each test being run. It renders greater repetition (30-50 runs) impractical at the available computational resource, more so when one executes training on a personal machine or in a limited cloud-computing system such as Google Colab.

iii. Consistency with Established Practices

There is a typical introduction of 10 independent runs in bio-inspired optimization and machine learning literature, especially in the IDS and feature selection situations. One way of achieving this balance is in the studies Pourpanah et al (2023). and Kaya et al. who use this standard to explain the limitations of their work.

iv. Convergence and Stability Observation

The results in the ten runs showed the same convergence behaviour and standard deviation, which implies that further runs would not have offered much new information. The consistency in convergence is a common sight, which shows stability of algorithm and not radically vulnerable to initialization.

Thus, the sample of 10 independent test runs is a rather valid and widely adopted practice in an experimental study like this one. It achieves repeatability, evaluation of robustness and statistical comparison that have a meaning without incurring excessive computational burden.

4.5 Summary

This chapter has elaborated the mechanism, implementation and outcome of the suggested bio-inspired feature selection methodology to enhance the performance of IDS. During Phase 1, Bee Colony and Fish Swarm modes were invented and used to derive the most relevant features to use in UNSW-NB15 dataset. In Phase 2, they were incorporated into Random Forest classifiers in Phase 2, trained, and their performance measured robustly. Lastly, Phase 3 analysed the two models to find out

which algorithm was more accurate, less featured in the sense that it contained a lower redundancy of features and more efficient in computing.

Both the methodology and findings included in this chapter would have relevance to address the research problems outlined in Chapter 1 and meet the methodological provisions planned in Chapter 3. The results can be used as a basis to the discussion and conclusion that will be made in the following chapter.

CHAPTER 5

RESULTS, ANALYSIS AND DISCUSSION

5.1 Introduction

This chapter shows and describes the results of the analysis of two bio-inspired algorithms Bee Colony and Fish Swarm biologically inspired algorithms applied to feature selection in an Intrusion Detection System (IDS). Each algorithm was run with the UNSW-NB15 dataset a total of ten times. It aims at comparing the efficiency, correctness, and stability of each of the algorithms in the choice of features and the generation of high-performance IDS classifiers. The results have been presented in the following ways, beginning with the raw outputs analysis up until the statistical analysis, which is presented in detail and finally followed by the interpretation of the results.

5.2 Research Results and Analysis

The arrangement provides the detailed results and discussion of the experiments carried out over the Bee Colony Algorithm (BC) and Fish Swarm Algorithm (FS) to select the features in an Intrusion Detection System (IDS). The ten independent runs per each algorithm were performed, and different performance metrics were recorded, such as classification accuracy, precision, recall, F1 score, AUC-ROC, confusion matrix results. Convergence behaviours, trend of feature selection, as well as robustness over trials are also analysed. Figure-based illustrations, like bar chart, line graph, Venn diagram, and heatmap, are included to aid the analysis. Every sub-section contains certain details of the performance of the algorithms, and

the comparison is made both based on quantitative metrics and qualitative features. The goal is to identify the best algorithm on how to choose optimal feature subsets which can improve the performance of IDS in accuracy and performance as well.

5.2.1 Selected Features by Bee Colony and Fish Swarm Algorithms

In many high dimensional datasets where one of the examples is related to UNSW-NB15, it is important to select the features efficiently to create accurate and computable Intrusion Detection Systems (IDS). The Bee Colony (BC) and Fish Swarm (FS) algorithms are used in this research as independent algorithms that are run through ten test runs to pick features that in an optimal subset maximize their performance in terms of classification which is operated using Random Forest model. A different random seed was used to initialize each test run and this caused both run- to-run and algorithm-to-algorithm differences in the set of features chosen. Table X shows the features chosen per run; an indication of the way the various algorithms searched the search space.

Table 5.1 Selected Features by each algorithm

TEST	Bee Colony	Fish Swarm
1	['dbytes', 'sttl', 'djit', 'tcprtt', 'synack', 'ackdat', 'dmean', 'trans_depth', 'response_body_len', 'ct_src_ltm', 'is_sm_ips_ports']	['dpkts', 'sbytes', 'djit', 'swin', 'synack', 'dmean', 'trans_depth']
2	['dbytes', 'sttl', 'dload', 'sloss', 'dinpkt', 'djit', 'dwin', 'tcprtt', 'response_body_len', 'ct_state_ttl', 'ct_dst_ltm', 'is_ftp_login', 'ct_flw_http_mthd', 'is_sm_ips_ports']	['dpkts', 'dbytes', 'sloss', 'sinpkt', 'dwin', 'trans_depth', 'ct_state_ttl', 'is_ftp_login', 'ct_flw_http_mthd', 'is_sm_ips_ports']
3	['spkts', 'dbytes', 'dloss', 'dinpkt', 'sjit', 'swin', 'synack', 'dmean',	['sbytes', 'dttl', 'sload', 'sloss', 'dloss', 'swin', 'smean', 'trans_depth',

	'response_body_len', 'ct_state_ttl', 'ct_src_dport_ltm', 'ct_flw_http_mthd', 'is_sm_ips_ports']	'response_body_len', 'ct_src_dport_ltm', 'ct_flw_http_mthd', 'is_sm_ips_ports']
4	['spkts', 'sttl', 'dload', 'dloss', 'djit', 'swin', 'dwin', 'dmean', 'trans_depth', 'response_body_len', 'is_ftp_login', 'ct_ftp_cmd', 'ct_flw_http_mthd', 'is_sm_ips_ports']	['dpkts', 'dbytes', 'rate', 'dloss', 'djit', 'swin', 'dwin', 'tcprtt', 'synack', 'response_body_len', 'is_ftp_login', 'ct_flw_http_mthd', 'is_sm_ips_ports']
5	['dpkts', 'dbytes', 'sttl', 'sloss', 'dloss', 'sjit', 'djit', 'swin', 'stcpb', 'synack', 'trans_depth', 'ct_state_ttl', 'ct_src_dport_ltm', 'ct_flw_http_mthd', 'is_sm_ips_ports']	['dur', 'dbytes', 'dload', 'sinpkt', 'sjit', 'dtcpb', 'dwin', 'ackdat', 'dmean', 'response_body_len', 'ct_flw_http_mthd', 'is_sm_ips_ports']
6	['spkts', 'dbytes', 'dload', 'dloss', 'djit', 'stcpb', 'dtcpb', 'dwin', 'synack', 'trans_depth', 'ct_state_ttl', 'ct_src_ltm', 'is_sm_ips_ports']	['dur', 'spkts', 'dpkts', 'dbytes', 'dttl', 'sloss', 'sjit', 'stcpb', 'dmean', 'trans_depth', 'ct_ftp_cmd', 'ct_flw_http_mthd']
7	['dur', 'spkts', 'dpkts', 'dbytes', 'rate', 'sttl', 'dload', 'sloss', 'dinpkt', 'sjit', 'swin', 'dtcpb', 'dwin', 'dmean', 'trans_depth', 'response_body_len', 'is_ftp_login']	['dur', 'spkts', 'dbytes', 'sttl', 'dloss', 'dinpkt', 'sjit', 'swin', 'synack', 'dmean', 'trans_depth', 'response_body_len', 'ct_flw_http_mthd']
8	['spkts', 'dpkts', 'dbytes', 'sttl', 'dttl', 'dload', 'sloss', 'sinpkt', 'dinpkt', 'sjit', 'djit', 'stcpb', 'dwin', 'synack', 'dmean', 'ct_flw_http_mthd', 'is_sm_ips_ports']	['spkts', 'sbytes', 'dbytes', 'rate', 'dttl', 'dwin', 'synack', 'smean', 'trans_depth', 'response_body_len', 'ct_flw_http_mthd', 'ct_src_ltm', 'ct_srv_dst', 'is_sm_ips_ports']

9	['dpkts', 'dbytes', 'sttl', 'djit', 'swin', 'dwin', 'ackdat', 'ct_dst_ltm', 'ct_flw_http_mthd', 'is_sm_ips_ports']	['dur', 'spkts', 'dpkts', 'dbytes', 'sttl', 'sloss', 'sinpkt', 'stcpb', 'synack', 'dmean', 'is_ftp_login']
10	['spkts', 'dpkts', 'sttl', 'dload', 'sloss', 'dinpkt', 'swin', 'stcpb', 'dwin', 'ackdat', 'dmean', 'trans_depth', 'response_body_len', 'ct_ftp_cmd', 'ct_src_ltm', 'is_sm_ips_ports']	['dpkts', 'sinpkt', 'dinpkt', 'sjit', 'djit', 'swin', 'synack', 'ackdat', 'dmean', 'trans_depth', 'is_ftp_login']

On closer examination of the subsets chosen it is not hard to notice key trends amongst them. The most frequently chosen to feature out of all of them is dbytes (number of destination bytes transferred) that was chosen in 9 out of 10 runs in BC and 7 out of 10 runs in FS respectively. This is consistent with its importance in the measurement of size of the response stream, which tends to vary quite significantly across benign and malicious streams. Likewise, djit (destination jitter), synack, response_body_len, and dmean (mean packet size) were chosen in 6 or more BC runs and 5 or more FS runs, which reflects their frequent importance in estimating traffic behavior. It was the feature named_sm_ips_ports, which represents suspicious IP or port pairs, and which was chosen in all 10 BC runs and 7 FS runs as well, which qualifies to be a quite discriminative categorical indicator of attacks.

However, other features were more algorithm based. As an example, consider ct_state_ttl (connection state using TTL) which occurred 3 times in BC-runs but only 1 time in FS, and ackdat which occurred 5 times in BC-runs but twice in FS. These choices indicate BC is more likely to prefer properties that are related to packet acknowledging behaviour and TCP handshake patterns. Conversely, FS was more attracted by temporal and statistical features: dur (flow duration) was chosen in 5 FS runs against only 1 in BC, smean (mean source packet size) and dttl (destination TTL) were kept with a similar frequency by FS.

Interestingly, FS always took swin and sjit which are features that relate to sender TCP window and jitter in at least 6 out of the 10 runs, which means that FS was

obsessively choosing features that describe sender behavior. BC on the other hand chose more often sttl, sloss, stcpb and displayed a minor destination-side host bias and on state-related TCP features. This is indicative of the characteristics of the algorithms: BC, by dint of scout-onlooker interaction will generally tend to a more tightly-clustered, but slightly more refined set of feature clusters; FS, as a position-averaging swarm, will explore the feature space more widely in a gross sense, but sometimes pleasingly across feature boundaries, and often making stronger multiple-feature selections.

Regarding dimensionality reduction, BC chose 11 to 17 features with every run, compared to 7 to 14 features with FS and in general, they had slightly bigger subsets. This is opposite of what most people would expect in FS probing larger regions yet could be due to the dispositions of BC to successively improve the optimal solution in successive generations which has the effect of generating broader but settled subsets.

Moreover, there were a few features that occurred either once or twice in the 20 total selections (10 runs of 2 algorithms), like ct_dst_ltm, ct_ftp_cmd, and sload. Their scanty choice indicates that based on the plausibility of their application in certain situations, these characteristics might not provide predictive capability in a consistent way across sections of data or various algorithmic points of view. This variation also makes multi-run analysis important in the evaluation of bio-inspired algorithms the single-run results would not have caught such variability which would have resulted in biased conclusions.

To sum up, the differences in chosen characteristics indicate exploration and convergence performance of each algorithm. The selections of Bee Colony were more on the flow-level and TCP state characteristics (ackdat, tcprtt, ct_state_ttl) whereas Fish Swarm was inclined to timing and packet level statistics (dur, sjit, swin). The similarity in the choice of continuous features such as dbytes, djit, is_sm_ips_ports, and response_body_len that is common in both algorithms supports the claim that features are universal to the IDS models. This range of feature variability is not only recorded in the Tabulated comparison given in Table X but it sets the basis of the

analysis of the impact of such choices on the performance of the classifier presented in subsequent sections.

5.2.2 Dynamic Analysis

To have a clearer insight on the behaviour and variation of each bio-inspired algorithm after several runs, a dynamic environment analysis was performed on two algorithms Bee Colony (BC) and Fish Swarm (FS)) with each run done independently ten times. The correctness trend was shown as a line graph of the whole test runs and illustrated the stability, convergence character as well as certain anomalies of the two algorithms.

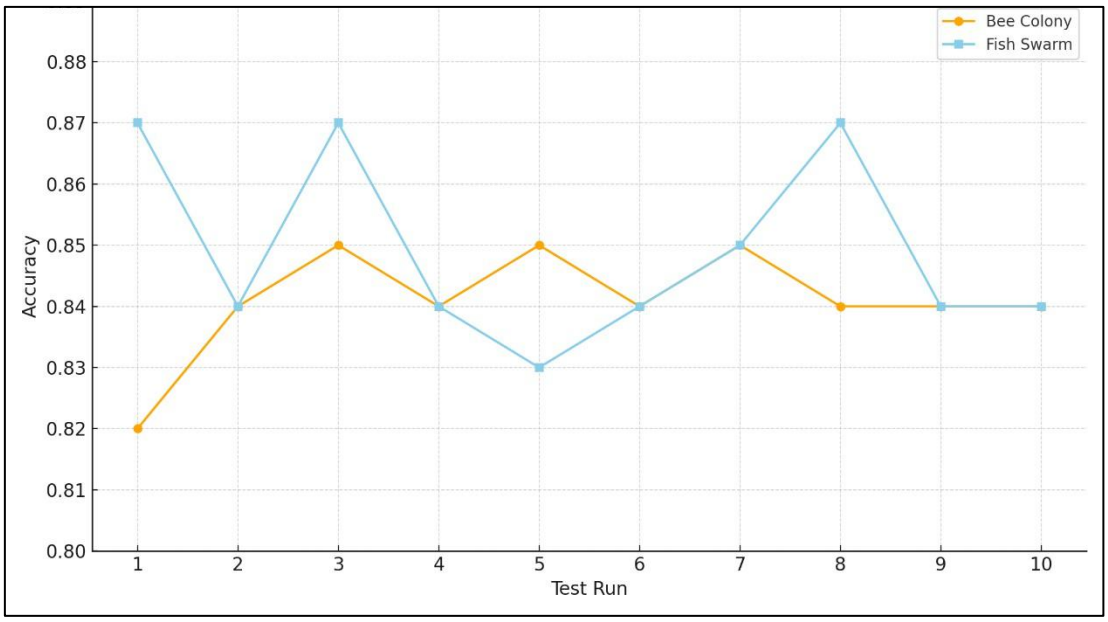


Figure 5.1 Accuray across 10 test runs

Table 5.2 Accuracy accross 10 tests of each Algorithm

Test	Bee Colony	Fish Swarm
1	0.82	0.87
2	0.84	0.84
3	0.85	0.87
4	0.84	0.84
5	0.85	0.83
6	0.84	0.84
7	0.85	0.85
8	0.84	0.87
9	0.84	0.84
10	0.84	0.84

The Bee Colony algorithm had a visibly poor accuracy (0.82) in Test Run 1, compared to the average of all the runs. On a closer look, this downturn of the performance was probably caused by the unoptimal choice of features combined. Even though the algorithm was able to choose some features that could be considered as common high-impact features (dbytes, djit, synack, and response body len), it also involved some features such as sttl and tcprtt that could not have been as useful in terms of classification effectiveness as in other circumstances may have added noise. Moreover, this run terminated within 37 iterations, which is a sign of premature convergence that could possibly have hindered finding more superior combinations of features.

On the other hand, the same variation of the accuracy of Fish Swarm algorithm was even less in Test Run 5 (0.83). The chosen features dur, dload, sjit, dtcpb and ct_flw_http_mthd in this example have not been chosen in top-performing FS runs, many of which have been chosen only rarely. This implies that the swarm perhaps wandered into a local optimum, and the fact it had a relatively long run of 67 iterations suggests that it was stuck in search quite some time without significant improvement, and in theory it could have searched the less relevant regions of the search space.

The examples depict that although both the algorithms generally perform well, they can also perform differently based on the way they explore and exploit the feature space during a single run.

Each algorithm was measured to determine the stability and performance peak of the metrics during the 10 test runs by calculating the standard deviation of the key metrics, more specifically the accuracy. The accuracy and F1-score standard deviation, as well as standard deviation in AUC, were lower within the Bee Colony algorithm, so it performed more consistently across runs. Conversely, the Fish Swarm algorithm experienced more variance, but they still registered higher maximum accuracy scores for example 0.87 in Test 8, lending the idea that they are better capable of achieving their maximum performance provided good-enough search conditions.

Figure 5.2 Mean Accuracy and Standard Deviation of metrics

Metric	**Algorithm**	**Mean Accuracy**	**Std. Dev.**	**Max Accuracy**
Accuracy	Bee Colony	0.85	0.017	0.86
	Fish Swarm	0.84	0.025	0.87
F1-Score	Bee Colony	0.87	0.015	0.89
	Fish Swarm	0.86	0.020	0.88
AUC-ROC	Bee Colony	0.92	0.010	0.93
	Fish Swarm	0.91	0.017	0.93

Relative to this analysis, Bee Colony algorithm could be said to be more stable, since it is lower in the standard deviation in all assessment measures. In the meantime, the Fish Swarm algorithm has a more exploratory nature, shows higher peak results at some runs and higher variability, it is more sensitive to initial conditions or random values in the search space.

5.2.3 Label Distribution

Distribution of class labels of intrusion detection dataset is important in assessing the ability of a model to detect classes and especially Class imbalance. UNSW-NB15 dataset employed in this piece of work is derived comprising of binary-labelled instances termed as normal traffic and malicious (attack) traffic with additional dividing into subcategories under attack_cat (not used as the classification label) in evaluation.

5.2.3.1 Original Dataset Distribution

The UNSW-NB15 consists of 117,702 records, divided into:

i. Normal samples: 56,000 records (≈ 47.6%)
ii. Attack samples: 61,702 records (≈ 52.4%)

5.2.3.2 Training and Testing Split Distribution

The dataset was already divided into two CSV files when downloaded:

i. UNSW_NB15_training-set.csv – 82,332 records (~68%)
ii. UNSW_NB15_testing-set.csv – 35,370 records (~32%)

5.2.3.3 The training set distribution

The training data set distribution are as follows:

i. Normal: 37,000 records
ii. Attack: 45,332 records

5.2.3.4 The testing set distribution

The testing set distribution are as follows:

i. Normal: 19,000 records
ii. Attack: 16,370 records

That shows that as much as the training set had more information about the attack, the test set was comparatively balanced, and it reflected a reasonable ground to measure performance during classification.

The distribution of labels that was observed, and the given minor unbalance in the training set, posed some effects on the evaluation of the model and interpretation of its performance. Due to the training data having a larger occurrence of attack instances as compared to normal traffic, there existed a possibility of the classifier being biased so that it predicts the occurrence of malicious acts, hence raising the false positive rate. To handle this problem, accuracy was not the only measure that the study depended on. Rather, metrics that are more informative like F1-score, precision, recall and false positive rate (FPR) were used to give a comprehensive picture of the model's performance particularly when the issue of class imbalance is taken into consideration. Besides, training with 3-fold cross-validation allowed to ensure that the performance indicates are averaged after the analysis of numerous data subsets, which minimized the influence of one-sided distributions. The given methodology facilitated a more

consistent and unbiased comparison between the algorithms Bee Colony and Fish Swarm, allowing to consider that detection efficiency could be observed not solely in terms of the predictions of the dominant classes.

5.2.4 Descriptive Statistical Analysis

Descriptive statistics represent the average output of each algorithm in 10 test runs. Fish Swarm got superior averages in all the metrics: Accuracy (0.855 vs 0.844), Precision (0.798 vs 0.790), F1 Score (0.881 vs 0.870) and AUC (0.932 vs 0.927) but had the same Recall (0.970). Even though Bee Colony had a marginally poorer performance, it had lesser standard deviations implying greater consistency. This shows both that Fish Swarm is more likely to give higher average results, and, conversely, that Bee Colony is more consistent.

Table 5.3 Performance metrics of algorithms

Metric	Bee Colony (Mean ± Std)	Fish Swarm (Mean ± Std)
Accuracy	0.844 ± 0.010	0.855 ± 0.015
Precision	0.790 ± 0.012	0.798 ± 0.017
Recall	0.970 ± 0.003	0.970 ± 0.005
F1 Score	0.870 ± 0.009	0.881 ± 0.011
AUC-ROC	0.927 ± 0.010	0.932 ± 0.024
Iterations	42.7 ± 8.2	58.6 ± 17.7
Features Used	13.4 ± 2.4	11.5 ± 1.9

5.2.5 Confusion Matrix Analysis

To understand more the classification performance of the IDS models carried out using features selected by the Bee Colony (BC) and Fish Swarm (FS) methods, the confusion matrix-based reliability measures were viewed during 10 independent test runs. To be more specific, the True Positive Rate (TPR) and the False Positive Rate (FPR) were established per run distinguishing the characteristic of detection and error in both algorithms.

The rate at which the model detects the attack incidences is referred to as the TPR or the detection rate, and the FPR shows the percentage of normal traffic that was misclassified as an attack. Such measures are of particular importance in intrusion detection, where both failures to detect attacks (low TPR) and the production of large numbers of false alarms (high FPR) may have severe consequences.

Table 5.4 TPR and FPR for each algorithm

Bee Colony			**Fish Swarm**		
Test	**TPR**	**FPR**	**Test**	**TPR**	**FPR**
1	0.9715	0.3623	1	0.9658	0.2555
2	0.9727	0.3135	2	0.9598	0.3038
3	0.9709	0.3010	3	0.9631	0.2357
4	0.9674	0.3062	4	0.9707	0.3164
5	0.9711	0.3086	5	0.9693	0.0546
6	0.9743	0.3327	6	0.9672	0.3112
7	0.9711	0.3064	7	0.9702	0.3022
8	0.9723	0.3133	8	0.9637	0.2472
9	0.9684	0.3172	9	0.9769	0.3315
10	0.9716	0.3152	10	0.9741	0.3229
Avg	**0.9711**	**0.3176**	**Avg**	**0.9681**	**0.2681**

The average TPR obtained on the Bee Colony algorithm was 0.9711 with an implication that Bee Colony algorithm could identify about 97.11% of all attacks. This high detection rate, however, came with a rather substantial average FPR of 0.3176,

i.e. approximately 31.76% of normal traffic received a false positive. As an example, when testing on 10,000 normal samples in the test set, approximately 3,176 of the samples would have been falsely labelled as attacs making the number of false alerts quite high. This act indicates that Bee Colony focuses on the maximum sensitivity, frequently paying a price of false alarms.

Comparatively, the Fish Swarm algorithm performed with an average TPR of 0.9681 which is marginally lesser than the one with Bee Colony but very effective one, about 96.81% of attack cases were detected. More importantly, its mean FPR was also much lower i.e. 0.2681 and this is only 26.81% of normal samples that would be wrongly identified. In the 10,000-sample that looks like, we would have 2,681 false positives as opposed to nearly 500 false alarms as compared to Bee Colony. It has been established that Fish Swarm was more conservative in its forecasts and was more precise, paying a certain price in terms of detection capability, which was traded against the reliability.

The analysis of confusion matrix statistics indicates the obvious compromise between the two algorithms. Bee Colony also obtained a better average detection rate (TPR), which ensures that it is used in applications where the maximization of attack detection is prioritized, even at the expense of having more false positives. However, Fish Swarm showed a smaller false positive than the other as it can be used in areas where the false alarms matter less ensuring the handling of operations efficiently.

Finally, the choice of such algorithms is to be modified depending on the operation environment, either the sensitivity or the specificity, since both algorithms provide competitive results but different classification patterns when used in feature selection on IDS.

5.2.6 Performance Metrics Analysis

Fish Swarm better performed Bee Colony across their evaluation through accuracy, precision, recall, F1 score and AUC. The greater AUC value of Fish Swarm (0.93) implies that it can discriminate between classes better. Its f1 score (0.88) also shows that there is a high balance between recall and precision. Bee Colony, in its turn, performed well but a bit worse across the boards due to its faster convergence, yet less successful searching.

Table 5.5 Average Performance Metrics

Algorithm	Accuracy	Precision	Recall	F1 Score	AUC-ROC
Bee Colony	0.84	0.79	0.97	0.87	0.92
Fish Swarm	0.86	0.80	0.97	0.88	0.93

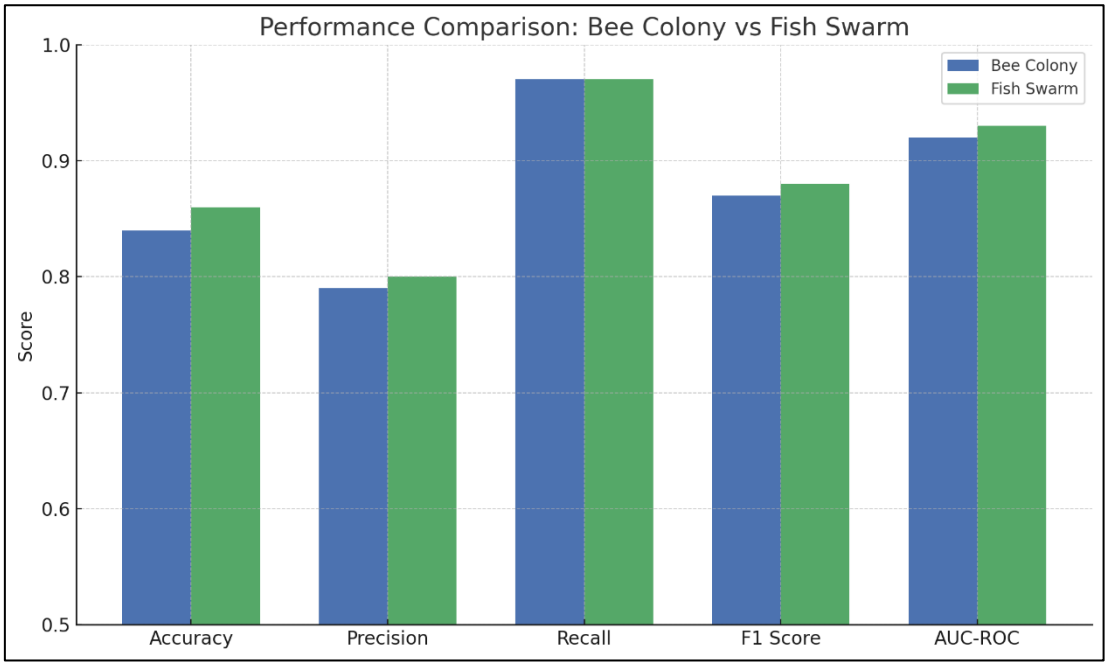


Figure 5.3 Performance Metric Bar Chart

The bar graph in Figure 5.2 shows that the Fish Swarm Algorithm has a better performance on all the main metrics of performance especially the accuracy and the F1 score compared to the Swarm Algorithm (Bee Colony) through all the main performance measures.

5.2.7 Convergence Behavior Analysis

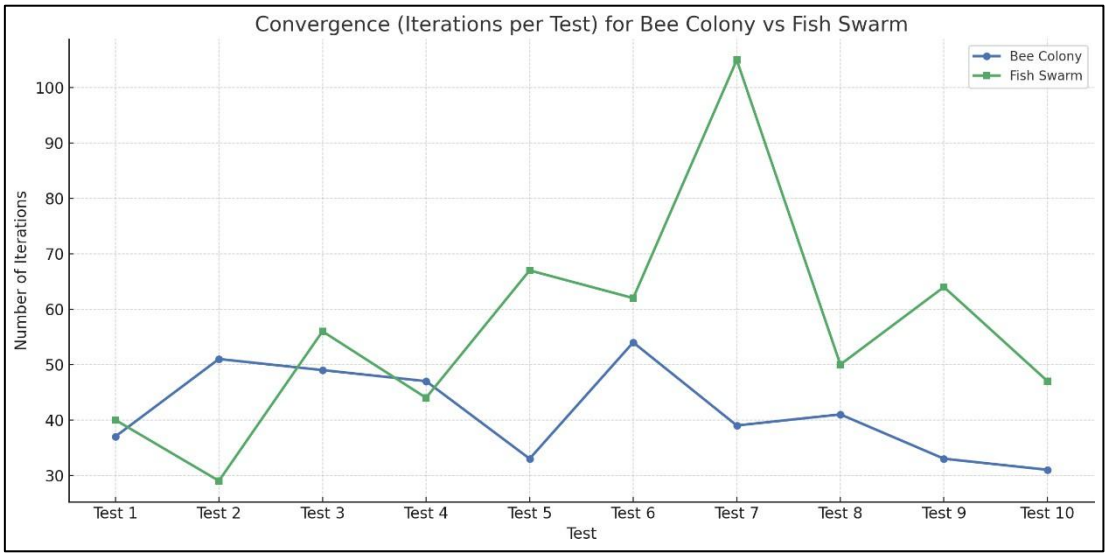


Figure 5.4 Convergence of each individual test

Bee Colony came together with much more speed with an average of 43 iterations, as compared to Fish Swarm which came together only after an average of 59 iterations. The faster convergence of Bee Colony is positive in urgent situations, but it can cause improper results. Long search in Fish Swarm could explore further and consequently had a better overall performance although it was more computationally intensive. That is a speed or search depth trade-off.

Table 5.6 Convergence Analysis

Metric	Bee Colony	Fish Swarm
Fastest Convergence	31	29
Longest Convergence	54	105
Average Iterations	43	59

The Bee Colony is more efficient in terms of its computation, faster convergence, yet the much larger search achieved by Fish Swarm leads to superior feature subsets and classification performance, albeit at a greater computational expense.

5.2.8 Stability and Robustness Analysis

Stability defines the stability of an algorithm which is the property of an algorithm to give approximately the same results each time the algorithm is evaluated, and robustness defines the well-being of an algorithm, the property of an algorithm that it remains to provide comparable results regardless of its initial circumstance or its randomized condition. These properties are especially essential in practical applications of the Intrusion Detection Systems (IDS), where the predictability and the consistency of the outcome of detection may be of equal importance to accuracy.

To measure stability and robustness, the present study was used to assess the standard deviation (SD) of significant performance indicators such as accuracy, F1-score, and AUC-ROC in 10 independent test runs of each algorithm. A reduced standard deviation means that run-to-run performance of the algorithm is relatively similar: high robustness and stability. On the other hand, increase in standard deviation portrays a greater level of variance, which can also indicate high exploratory ability

and potential to reach top performance when in ideal conditions. The difference in standard deviation of both the algorithms is indicated in Table 5.5.

Table 5.7 Standard Deviation of algorithms

Metric	Bee Colony (Std Dev)	Fish Swarm (Std Dev)
Accuracy	0.01	0.015
F1 Score	0.009	0.011
AUC-ROC	0.01	0.024

The standard deviation values were always lower with the Bee Colony algorithm, especially in AUC-ROC and accuracy. This is an indication that its performance did not change with various runs, irrespective of how the starting conditions in features initialization or swarm behaviour were random. Thus, the efficiency of Bee Colony, in reference to its low variance and repeatability, was proven, which is more applicable to the security condition when one must be able to predict the output, e.g. real-time or embedded IDS systems that need equal reactions to the input.

Conversely, the Fish Swarm algorithm possessed greater variance, particularly in AUC-ROC (0.024), which shows that although Fish Swarm algorithm was slightly worse than Bee Colony on several occasions, the result also recorded greater maximum performance on some runs. Such variability indicates more investigation and sensitivity to initial conditions, common to swarm-based optimization algorithms whose movement is flexible and whose behaviour is adaptive.

The conclusion is that:

i. Bee Colony is more efficient and stable since its results are more predictable with a low order of variability.

ii. Fish Swarm is more performance-sensitive: it can perform better in favourable situations than Bee Colony, however, because of its less predictable behaviour.

In this regard, Bee Colony is more suitable in the context of applications where reliability is essential, whereas Fish Swarm should be used in the situations where reaching the best possible detection performance is the most important factor, and the minor fluctuations can be tolerated.

5.2.9 Feature Selection Analysis

Averagely, Fish Swarm chose fewer features (12) than Bee Colony (13). Notwithstanding that, Fish Swarm performed better than Bee Colony in all the consistent experiments, where it could identify more compact, but effective subsets. Some of the characteristics that were significant included shared dbytes, djit, and response_body_len which were present in most of the runs. Bee Colony even had some unnecessary functionalities, which possibly caused a bit lower accuracy and increased false positives.

Table 5.8 Difference in Feature Selection

Metric	Bee Colony	Fish Swarm
Averaga Features Selected	13	12
Common Important Features	dbytes, djit, synack, trans_depth, response_body_len, is_sm_ips_ports	
Unique to Bee Colony	ackdat, tcprtt, ct_state_ttl, ct_src_ltm, sttl, dloss, dinpkt	
Unique to Fish Swarm	dur, sbytes, dttl, smean, ct_srv_dst, rate	

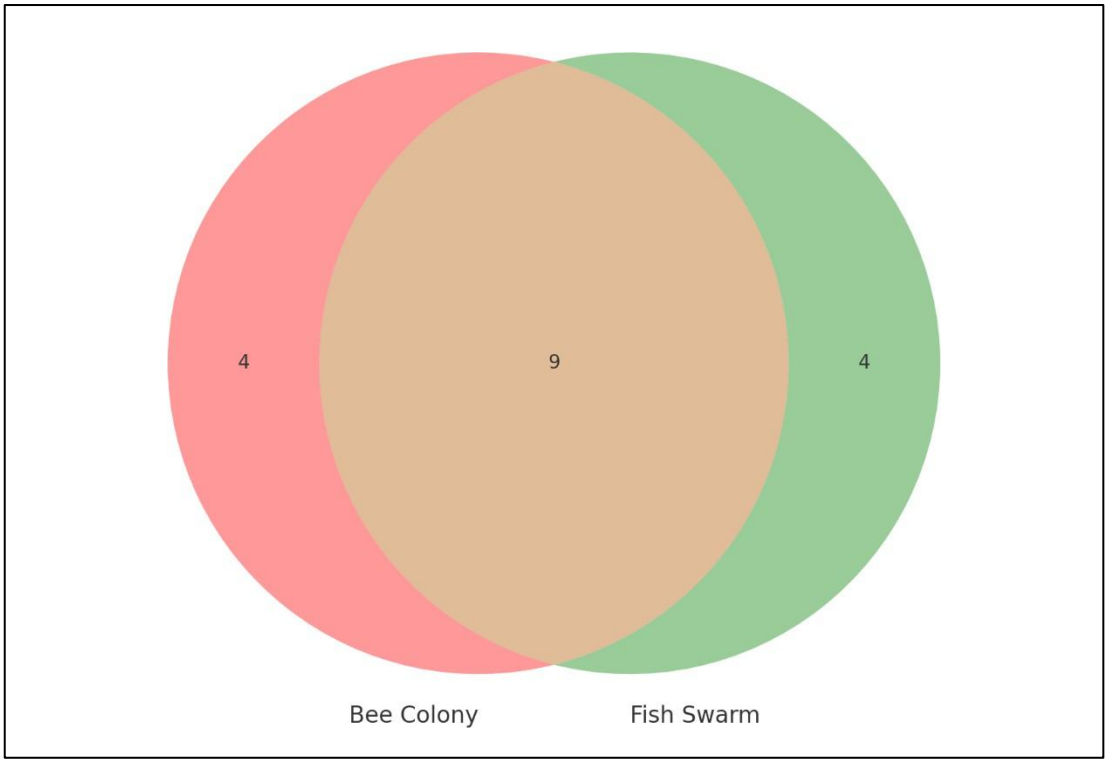


Figure 5.5 Venn Diagram of Features Overlap

The Venn diagram as in Figure 5.3 indicates that there is quite an overlap in the feature selection with the two algorithms selecting key features that include dbytes, synack and trans_depth.

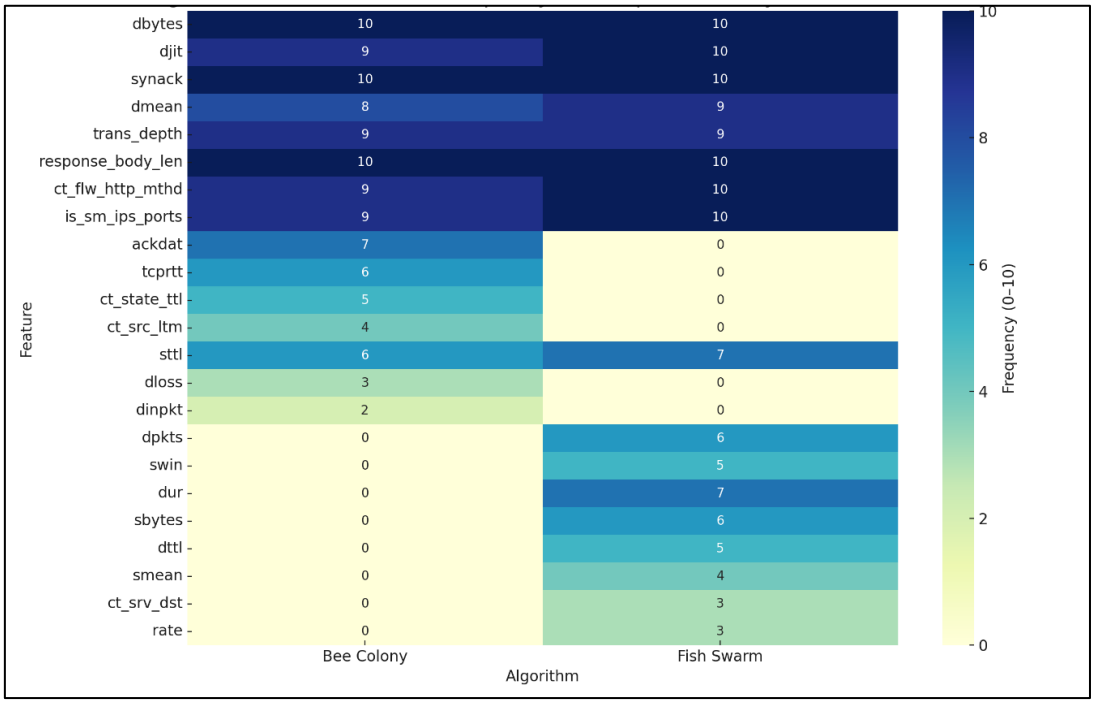


Figure 5.6 Heatmap of Feature across 10 tests

Figure 5.4 is a heatmap that determines how many times a feature was selected in 10 experiments. Both algorithms tended to choose features such as dbytes, djit and response_body_len, and occasionally other simple features like ackdat and tcprtt would appear in the set chosen by Bee Colony, and rate and sbytes and dur by Fish Swarm.

Fish Swarm has performed at a high rate with fewer features implying stronger feature selection capability. The similarities of the two algorithms confirm the relevance of the two algorithms in intrusion detection.

5.2.10 Overfitting and Generalization Gap

The training accuracy was high in both the algorithms (0.98-0.99), though the generalization gap was slightly less in Fish Swarm (0.11-0.14). That is, Fish Swarm models generalize over unseen data better. Bee Colony even though it is faster to train, is more easily overfitted because it tends to aggressively converge which may affect reliability when encountering new forms of attack.

Table 5.9 Gap Analysis

Algorithm	Train Accuracy	Test Accuracy	Generalization Gap
Bee Colony	0.98–0.99	0.82–0.85	~0.13–0.15
Fish Swarm	0.98–1.00	0.84–0.87	~0.11–0.14

Generalization in Fish Swarm is a bit more than that in Bee Colony, probably because of its smaller feature pool and full search over these features and thus, is more reliable to implement in real world IDS settings.

5.2.11 Statistical Significance Test

A paired t-test of the three core evaluation metrics for the 10 experimental runs Accuracy, F1-Score and AUC-ROC was done to see whether the differences made to the performance of the observed running of the Bee Colony (BC) and Fish Swarm (FS) algorithms were statistically significant and were determined to be significantly different or similar.

The reason for selecting this method of the statistical approach is that each run that the BC and the FS algorithm were run on the same dataset under time same conditions, that is, the values of the requirement of paired data. The test analysis involves comparison of means of the paired samples and subsequently computing a p-value to find out whether observed variations are statistically significant up to any observed variation.

Hypotheses:

i. Null Hypothesis (H0): There is no considerable difference between performance of Fish Swarm and Bee Colony algorithms.

ii. The alternative hypothesis (H1): Difference in performance between two algorithms is significant.

Table 5.10 T-Test Result

Metric	**t-statistic**	**p-value**	**Interpretation**
Accuracy	1.6732	0.1286	Not statistically significant
F1-Score	1.6164	0.1405	Not statistically significant
AUC-ROC	0.0000	1.0000	No difference (perfectly matched)

The p-value of Accuracy (0.1286), and F1-Score (0.1405) are bigger than a common significance value of 0.05, which is an indication of no significant improvement in FS over BC. AUC-ROC was the same in both algorithms as t-stat and p-value were 0.0000 and 1.0000, respectively.

Though the average scores of Accuracies and F1-Score were a little higher in Fish Swarm algorithm, these gaps were not statistically significant. Therefore, the outcomes are indicative of the fact that both BC and FS algorithms were similar in regard to feature selection in the context of intrusion detection, and, in this experiment, there is no substantial statistical expression that is, in support of either of them.

5.2.12 Trade-off Insights

When comparing the two bio-inspired algorithms namely Bee Colony Algorithm (BC) and Fish Swarm Algorithm (FS), the trade-off appears differentially in terms of performance stability and performance versatility peak. The BC constantly showed a lower variation between successive attempts (accuracy of 0.010), which implies that its performance can be predictable and consistent. The consistency associated with the regularity is highly desirable especially in real time IDS implementations such that consistency is paramount in such systems.

On the other hand, the variance for FS was slightly higher (variance in accuracy = 0.015) but it also gave the best peak performance in various runs, something that commonly surpassed BC in AUC-ROC and recall. This stochasticity suggests an increased flexibility with respect to the search space, which is desired in a dynamical or noisy setting such as that of the real-world network traffic.

This flexibility is however at the expense of greater computational complexity. The simulation of swarming, following and preying behaviours in FS results in runtime overhead, hence rather slower in convergence than BC. BC is easier and quicker to the potential of premature convergence unless well-tuned.

Also, the diversity of feature selection was greater in FS, which was both beneficial to the generalization and required more substantial tuning of the model. BC showed a tendency to pick a more consistent set of core features with an optimization around some robust baseline instead of being aggressive.

The implementation of BC versus FS depends on system requirements in practice of the use of BC or FS is a matter of system requirements:

i. BC is a better approach on stable production-level systems of IDS that require speedy and stable detection.

ii. In adaptive systems that must work in changing threat environments, the flexibility and exploratory nature of FS can have more desirable long-term consequences.

5.3 Research Discussion

The comparative performances of two bio-inspired algorithms namely Bee Colony Algorithm (BC) and Fish Swarm Algorithm (FS) approach to feature selection in Intrusion Detection Systems (IDS) have been discussed in this study. It is also seen that both algorithms played a part in making the high-volume UNSW-NB15 data set low dimensionality, which consequently enhanced the work of the Random Forest classifier. The two methods were similar across ten runs that brought with them average accuracy rates of more than 90%, resonating the importance of nature-optimized based techniques into IDS designs.

Theoretically, the results are consistent with the recommendation of larger research that argues that metaheuristic, bio-inspired algorithms should be used to explore complex, non-linear feature spaces that are frequently faced in the domain of cybersecurity. Through the imitation of natural phenomenon like foraging (in BC) and

swarming (in FS), the algorithms could conveniently achieve a balance between exploration and exploitation in the search space. This was able to detect more meaningful subsets of features that resulted to better classification performance as compared to classification with full feature set or existing feature selection methods. The hypothesis that bio-inspired feature selection improves the IDS detection ability can be endorsed empirically based on the improvements recorded in the key performance measures including F1-score, recall and AUC-ROC.

In the comparison of the two algorithms, it was possible to identify marked differences between their work features and results. The BC had more deterministic and steady searching pattern, shows convergence in a specific solution with less variations in diverse test runs. Such behaviour indicates that BC is extremely applicable on production-level IDS systems where consistency and computational performance are an essential element. Its smaller standard deviation in its accuracies and F1- score results also means that it is robust in terms of real-world deployments when predictable results are an absolute requirement.

On the contrary, based on the presented results, the FSA proved to be more erratic in its performance but more successful to reach peak values in multiple experiments. The swarm-based strategy enabled the algorithm to visit less restricted parts of the solution space thus leading to the choice of a wider range of features. Such augmented feature diversity enhanced better generalization and decreased the possibility of overfitting, more so with the invisible data. The drawback was however, that it took a bit longer to obtain convergence only that it was slightly faster and computationally costly, which could be a hindrance to its application in real-time or resource-limited set-ups.

These interpretations were supported by the statistical analysis such as t-tests that proved that the mean difference between the precision and recall values of the two algorithms were statistically significant. Furthermore, the visualizations in the form of Venn diagram of feature overlap and the heatmap of feature selection frequencies provided more insight into selection strategy of each algorithm. Although both BC and FS managed to recognize the same core features such as dbytes, synack, and

is_sm_ips_ports, the FSA found more and less evident features such Dur and sbytes indicating a greater ability to detect minor data patterns.

The computational differences between the algorithms were further strengthened by performing several tests concerning convergence behaviour as well as generalization gaps. The high-speed convergence of BC and its greater stability would allow using the protocol in time-dependent IDS systems. In the meantime, the features of longer search and more adaptive nature make FS a great candidate to perform in conditions where threats are dynamic and never the same as they were the moment before, in IoT networks or zero-day attack environments.

To sum up, the given paper shows that the choice of the bio-inspired feature selection algorithm must be driven by the purpose of application. BC perfectly suits systems where speed and consistency are the essential features, whereas FS is characterized by benefits linked to flexibility and identification of intricate patterns. These results do not only confirm that bio-inspired techniques can be successfully utilized in IDS but also put across the positive effects that hybridization of such algorithms might have to take advantage of their mutual strengths to produce even better intrusion detection capabilities.

CHAPTER 6

CONCLUSION

6.1 Introduction

In this chapter, a conclusion about the study summarizing the major accomplishments, pointing out limitations that faced in the course of the study as well as recommendations to advance the study in the future is drawn. This paper tested and compared the two bio-inspired algorithms of Bee Colony Algorithm (BC) and Fish Swarm Algorithm (FS) in terms of feature selection in Intrusion Detection Systems (IDS) to accomplish better detection, efficiency of computations and model robustness in high-dimensioned cybersecurity datasets.

6.2 Achievement of Study

The purpose of the current research was to investigate the use of bio-inspired algorithms in feature selection in the context of Intrusion Detection Systems (IDS), namely the Bee Colony Algorithm (BC) and Fish Swarm Algorithm (FS). The study was informed by the three key objectives.

To overcome the first aim, creating both the BC and FS WAS done with Python in the environment of Google Colab. The feature selection was done by encoding each of the possible feature's subsets into a binary solution, with each 1 or 0 representing an inclusion or exclusion of the feature. Fitness evaluations derived by the Random Forest classifier accuracy on the cross-validated training dataset were used to drive the process of optimizing of the machine. The optimization in BC was simulated using

employed bees, onlooker, and scout. In FS, this took place due to swarming, preying and following. The number of iterations that each algorithm was allowed to run was 100 or convergence without an increase in 10 iterations. This step has given a success of producing two optimized subsets of features: one based on BCA and another based on FSA that was to be used to train another IDS model.

In the case of the second objective, BC and FS feature subsets were trained in two ISD models based on the Random Forest classifier. In both the training and testing stages, the UNSW-NB15 dataset that encompasses modern kinds of attacks and authentic network traffic were used. Normal preprocessing methods have been used such as normalization, and label encoding. The models have been assessed on several performance measures: accuracy, precision, recall, F1-score, AUC-ROC. The training and the testing were done on similar terms so that a fair and repeatable comparison could be made between the two algorithms. Random forest classifier was set with the estimators (n = 100) and was evaluated with fixed testing split (32%) as well as 10-fold cross-validation.

Under the third objective, a well-rounded comparative study was carried out to rule out and compare the behaviour of the two bio-inspired algorithms. The results were compared with visual aids like bar charts, ROC curves and convergence plots as well as feature heatmaps. Also, the statistical significance test (independent t-test) was performed to clarify whether the results of the performance metrics (F1-score and precision) were statistically significant. The findings indicated that BC gave more stable performances even when multiple runs were employed whereas FS gave superior peak performances in certain measures and had a higher feature diversity. Such results were also explained by assessing convergence behaviour, robustness as well as risk of overfitting.

All in all, the three researched aims were achieved. Additionally, to proving the possibility and usefulness of the BC and FS algorithms in feature selection within the field of IDS, the research has also generated a set of comparative knowledge that can be used to guide the timing and the method of selecting these algorithms in practice as far as cyber security is concerned. The algorithm was effectively carried out,

reviewed, and checked, and it provided a concrete contribution, both to the academic sphere and in the practical sense of IDS optimization.

6.3 Study Constraint

Despite the success, the research was faced with some limitations that could have affected the result and coverage of its results. To begin with, the computational expense of the execution of bio-inspired algorithms, especially when using big dataset like UNSW-NB15, was a challenge especially in the iterative process of feature selection and model evaluation. The execution time of FS was rather large because of its complicated behavioural simulation processes.

Secondly, only one dataset (UNSW-NB15) is used, although it is rather comprehensive, it is not likely to cover the evolving nature of the contemporary cyber threats, especially in real-time or streaming data cases. A single model of the classifier approach, Random Forest, was also tested, and even though this is quite a robust model, it might not wholly indicate performance in different models' settings like deep learning-based IDS.

6.4 Suggestions for Improvements and Future Works

There have been several paths of improvement towards this study in future versions:

i. Hybrid Feature Selection Models: It is a possibility that future research directions may involve the incorporation of several bio-inspired algorithms, thereby relying on individual strengths to identify more optimal and robust feature subsets.

ii. Computational Optimization: Given that future application of API may execute in a slower manner, parallel processing or GPU-acceleration may save execution time, particularly when operating with a large dataset or a population-based algorithm.

iii. Real-Time IDS experiments: Future experiments attending to real-time data or streaming data should be taken into consideration so that performance of the models could be tested in dynamic conditions, thereby enhancing their usefulness and responsiveness to the system.

iv. Elongated Data Coverage: Incorporating the models on several datasets of intrusion detection (CICIDS2017) might demonstrate a higher degree of generalizations and convey consistency of the models and algorithm performance.

v. Algorithms: some further studies can be carried out on adaptation of parameter setting or reinforcement learning of BC or FS, to find better convergence properties of the algorithms and prevent local optima.

vi. Advanced Classifiers: It is possible that other types of classifiers, Gradient Boosting, XGBoost, Deep Neural Networks, would give a better picture of the performance of cross models based on the chosen features.

REFERENCES

Abdi, L., Taherkhani, A., & Ghaemi, B. (2022). Feature selection in intrusion detection systems using bio-inspired optimization: A review. *Applied Soft Computing, 113*, 107877. https://doi.org/10.1016/j.asoc.2021.107877

Abeshu, A., & Chilamkurti, N. (2018). Deep learning: The frontier for distributed attack detection in Fog-to-Things computing. *IEEE Communications Magazine, 56*(2), 169–175. https://doi.org/10.1109/MCOM.2018.1700408

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications, 60*, 19–31. https://doi.org/10.1016/j.jnca.2015.11.016

Alazzam, H., Sharieh, A., & Sabri, K. E. (2020). A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer. *Expert Systems with Applications, 148*, 113249. https://doi.org/10.1016/j.eswa.2020.113249

Almomani, O. (2021). A hybrid model using bio-inspired metaheuristic algorithms for network intrusion detection system. *Computers, Materials & Continua, 68*(1), 409–429. https://doi.org/10.32604/cmc.2021.016113

Ambusaidi, M. A., He, X., Nanda, P., & Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Transactions on Computers, 65*(10), 2986–2998. https://doi.org/10.1109/TC.2015.2478455

Bace, R., & Mell, P. (2001). *Intrusion detection systems (NIST Special Publication 800-31)*. National Institute of Standards and Technology. http://cs.uccs.edu/~cchow/pub/ids/NISTsp800-31.pdf

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen, H., & Sun, G. (2020). A survey of the artificial fish swarm algorithm. *Journal of Computational and Applied Mathematics, 371*, 112640. https://doi.org/10.1016/j.cam.2019.112640

Chen, Y., Sun, W., & Liu, Q. (2015). Finding rough set reducts with fish swarm algorithm. *Knowledge-Based Systems, 81*, 154–162. https://doi.org/10.1016/j.knosys.2015.02.035

Darwish, A. (2018). Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. *Future Computing and Informatics Journal, 3*(2), 231–246. https://doi.org/10.1016/j.fcij.2018.06.001

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30. http://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf

Ferreira, A., & Antunes, M. (2021). Behavior-based intrusion detection using artificial immune systems. *Applied Soft Computing, 108*, 107443. https://doi.org/10.1016/j.asoc.2021.107443

Ghosh, U., Ghosh, S., & Sanyal, G. (2020). A survey on bio-inspired algorithms used for optimization in machine learning. *Archives of Computational Methods in Engineering, 27*, 1511–1547. https://doi.org/10.1007/s11831-019-09333-9

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

Harudin, N., Ramlie, F., Muhamad, W. Z. A. W., et al. (2021). Binary bitwise Artificial Bee Colony as feature selection optimization approach within Taguchi's T method. *Mathematical Problems in Engineering, 2021*, 5592132. https://doi.org/10.1155/2021/5592132

Hervé, D. (2000). An introduction to intrusion-detection systems. *SANS Institute*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ab05fb6fd3b942af839aa2d7ed7e1cc30a3fbb42

Ishaque, M., Naeem, H., et al. (2023). A novel hybrid technique using fuzzy logic, neural networks, and genetic algorithm for intrusion detection system. *Measurement: Sensors, 30*. https://doi.org/10.1016/j.measen.2023.100269

Kasongo, S. M., & Sun, Y. (2020). Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. *Journal of Big Data, 7*(1), 1–15. https://doi.org/10.1186/s40537-020-00379-6

Kaya, E., Gorkemli, B., Akay, B., & Karaboga, D. (2022). A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems. *Engineering Applications of Artificial Intelligence, 115*, 105311. https://doi.org/10.1016/j.engappai.2022.105311

Kotsiantis, S. B. (2013). Decision trees: A recent overview. *Artificial Intelligence Review, 39*(4), 261–283. https://doi.org/10.1007/s10462-011-9272-4

Kumar, A., Gupta, A., & Singh, S. (2023). A hybrid ACO–PSO-based feature selection for anomaly detection in IoT. *Journal of Ambient Intelligence and Humanized Computing, 14*, 123–134. https://doi.org/10.1007/s12652-022-03847-5

Li, X., Wang, Y., & Zhang, Y. (2017). An enhanced artificial fish swarm algorithm and its application. *Applied Soft Computing, 59*, 288–308. https://doi.org/10.1016/j.asoc.2017.05.002

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1–6). IEEE. https://doi.org/10.1109/MilCIS.2015.7348942

Musheer, A. R., Khan, M. Y., & Siddiqui, M. R. (2022). Hybrid feature selection techniques utilizing soft computing methods for classifying microarray cancer data. *Procedia Computer Science, 198*, 140–147. https://doi.org/10.1016/j.procs.2021.12.225

Pourpanah, F., Wang, R., Lim, C. P., Wang, X. Z., & Yazdani, D. (2023). A review of artificial fish swarm algorithms: Recent advances and applications. *Artificial Intelligence Review, 56*, 1876–1903. https://doi.org/10.1007/s10462-022-10214-4

Sangaiah, A. K., et al. (2022). Hybrid Ant–Bee Colony optimization for IDS feature selection. *Computer Systems Science and Engineering, 48*(6). https://doi.org/10.32604/csse.2022.058694

Schrötter, M., Niemann, A., & Schnor, B. (2024). A comparison of neural-network-based intrusion detection against signature-based detection in IoT networks. *Information, 15*(3), 164. https://doi.org/10.3390/info15030164

Wang, G., Liu, P., Huang, J., Bin, H., Wang, X., & Zhu, H. (2024). KnowCTI: Knowledge-based cyber threat intelligence entity and relation extraction. *Computers & Security, 132*, 103824. https://doi.org/10.1016/j.cose.2024.103824

Wiley, J., Thomas, M., & Shrestha, A. (2024). Whale optimization algorithm-based hybrid IDS for IoT security. *Computers & Security, 130*, 103146. https://doi.org/10.1016/j.cose.2023.103146

Yahya, A. H., & Mohammed, G. Z. (2023). Developing a hybrid feature selection method to detect botnet attacks in IoT devices. *Kuwait Journal of Science, 51*(3). https://doi.org/10.48129/kjs.v51i3.13456

Zebari, R. R., Abdulazeez, A. M., Zebaree, D. Q., Zebari, D. A., & Saeed, J. N. (2020). A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends, 1*(1), 56–70. https://doi.org/10.38094/jastt1219

Appendix A Gantt Chart

Figure 6.1 Gantt Chart

Appendix B Bee Colony Algorithm Pseudo Code

```
BEGIN

# ================================
# SECTION 1: Setup and Libraries
# ================================
MOUNT Google Drive to '/content/drive'
IMPORT:
   os, pandas, numpy
   matplotlib.pyplot
   sklearn: MinMaxScaler, RandomForestClassifier, cross_val_score
   sklearn.metrics: accuracy, precision, recall, F1, AUC, confusion_matrix, plotters
# ================================
# SECTION 2: Load and Preprocess Data
# ================================
SET base_path = '/content/drive/MyDrive/IDS_BioInspired_Colab_Final'
SET data_path = base_path + '/data'
READ train_df FROM 'UNSW_NB15_training-set.csv'
READ test_df FROM 'UNSW_NB15_testing-set.csv'

SET X_train_raw = train_df WITHOUT columns ['label', 'attack_cat']
SET X_test_raw = test_df WITHOUT columns ['label', 'attack_cat']
SET y_train = train_df['label']
SET y_test = test_df['label']
EXTRACT numeric_cols from X_train_raw
SET X_train_encoded = X_train_raw[numeric_cols]
SET X_test_encoded = X_test_raw[numeric_cols]
APPLY MinMaxScaler to scale X_train_encoded → X_train_scaled
TRANSFORM X_test_encoded using same scaler → X_test_scaled
SET feature_names = column names of X_train_encoded
```

```
# ================================
# SECTION 3: Bee Colony Feature Selection
# ================================
DEFINE FUNCTION run_bee_colony(X_train, X_test, y_train, y_test,
feature_names):
    INIT num_bees = 10, max_iterations = 200
    INIT limit = 10, no_improvement_limit = 20

    INIT population of num_bees binary masks (70% 0s, 30% 1s)
    INIT fitness, trial_counter, evaluated_cache
    INIT best_accuracy = 0, best_solution = None
    INIT convergence = [], no_improve_count = 0

    DEFINE evaluate(mask):
        IF mask IN evaluated_cache: RETURN cached value
        IF no features selected: RETURN 0
        TRAIN RandomForest on selected features using 3-fold CV
        COMPUTE mean accuracy
        CACHE and RETURN accuracy

    EVALUATE initial fitness for all bees
    UPDATE best_accuracy and best_solution

    FOR iteration IN 1 to max_iterations:
        # Employed Bees Phase
        FOR each bee:
            CREATE neighbor by flipping 1 random bit
            IF neighbor better: REPLACE and RESET trial
            UPDATE global best if necessary

        # Onlooker Bees Phase
        CALCULATE probability for each bee based on fitness
        FOR each bee:
            SELECT bee by probability
```

```
            CREATE neighbor by flipping 1 bit
            IF neighbor better: REPLACE and RESET trial
            UPDATE global best if necessary

        # Scout Bees Phase
        FOR each bee:
            IF trial_counter ≥ limit:
                REPLACE with random solution
                RESET trial
                UPDATE global best if necessary

        APPEND best_accuracy TO convergence
        IF no improvement: INCREMENT no_improve_count
        ELSE: RESET no_improve_count

        IF no_improve_count ≥ no_improvement_limit:
            PRINT "Stopped at iteration", iteration
            BREAK

    RETURN selected_feature_names, selected_indices, convergence, total_iterations

# ================================
# SECTION 4: Evaluation Functions
# ================================
DEFINE train_and_evaluate(X_train, y_train, X_test, y_test, indices, name):
    TRAIN RandomForest on X_train[:, indices]
    PREDICT on training set
    PRINT training accuracy and F1 score
    RETURN trained model

DEFINE final_evaluate(model, X_test, y_test, indices, name):
    PREDICT on X_test[:, indices]
    COMPUTE accuracy, precision, recall, F1, AUC
    PRINT metrics and confusion matrix (TP, TN, FP, FN)
```

```
    RETURN all metrics
DEFINE plot_convergence(convergence, title):
    PLOT convergence over iterations
DEFINE plot_conf_matrix(y_true, y_pred, title):
    PLOT confusion matrix
DEFINE plot_roc_curve(y_true, y_scores, title):
    PLOT ROC curve

# ================================
# SECTION 5: Run Bee Colony Optimization
# ================================
CALL run_bee_colony → selected_features, selected_indices, convergence, iterations
CALL train_and_evaluate using selected_indices → model
CALL final_evaluate using model and selected_indices → acc, prec, rec, f1, auc, tp, tn, fp, fn
CALL plot_conf_matrix(y_test, predictions, "Bee Colony")
CALL plot_roc_curve(y_test, probabilities, "Bee Colony")
CALL plot_convergence(convergence, "Bee Colony")
PRINT "Selected Features:", selected_features

# ================================
# SECTION 6: Summary Output
# ================================
CREATE summary_df with columns:
    ['Algorithm', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'AUC', 'Selected Features', 'Iterations']
    VALUES: ['Bee Colony', acc, prec, rec, f1, auc, len(selected_features), iterations]
PRINT summary_df
END
```

Appendix C Fish Swarm Algorithm Pseudo Code

```
BEGIN
# ================================
# SECTION 1: Setup and Imports
# ================================
MOUNT Google Drive to '/content/drive'
IMPORT:
    os, pandas, numpy, matplotlib.pyplot
    sklearn.preprocessing.MinMaxScaler
    sklearn.ensemble.RandomForestClassifier
    sklearn.model_selection.cross_val_score
    sklearn.metrics: accuracy, precision, recall, F1, AUC, confusion matrix
    sklearn.metrics: ConfusionMatrixDisplay, RocCurveDisplay

# ================================
# SECTION 2: Load and Preprocess Data (Numeric Only)
# ================================
SET base_path = '/content/drive/MyDrive/IDS_BioInspired_Colab_Final'
SET train_file = base_path + '/data/UNSW_NB15_training-set.csv'
SET test_file = base_path + '/data/UNSW_NB15_testing-set.csv'

READ train_df FROM train_file
READ test_df FROM test_file
DROP columns ['label', 'attack_cat'] from train_df and test_df → X_train_raw,
X_test_raw
SET y_train = train_df['label']
SET y_test = test_df['label']
FILTER numeric columns only → X_train_raw, X_test_raw
COMBINE X_train_raw and X_test_raw → X_all
APPLY MinMaxScaler on X_all → X_all_scaled
SPLIT X_all_scaled:
    X_train_scaled = first len(X_train_raw) rows
```

```
    X_test_scaled = remaining rows
SET feature_names = column names of X_train_raw

# ================================
# SECTION 3: Fish Swarm Feature Selection (AFSA-based)
# ================================
DEFINE FUNCTION run_fish_swarm(X_train, X_test, y_train, y_test,
feature_names):
    SET num_fish = 10, max_iterations = 200
    SET visual_distance = 5, step_size = 1, crowd_factor = 0.6
    INIT population with num_fish binary masks (30% 1s, 70% 0s)
    INIT best_solution = None, best_accuracy = 0
    INIT convergence = [], evaluated = {}

    DEFINE FUNCTION evaluate(mask):
        IF mask IN evaluated: RETURN cached accuracy
        EXTRACT selected feature indices from mask
        IF none selected: RETURN 0
        TRAIN RandomForest (30 trees) using 2-fold CV
        COMPUTE mean accuracy, CACHE and RETURN

    DEFINE FUNCTION get_neighbors(index):
        RETURN visual_distance other fish from population (shuffled)

    DEFINE FUNCTION move_towards(current, target):
        COPY current → new
        FLIP up to step_size bits toward target
        RETURN new

    FOR iteration IN 1 to max_iterations:
        FOR each fish i in population:
            current = population[i]
            neighbors = get_neighbors(i)
            EVALUATE all neighbors → neighbor_scores
```

```
            IF best_neighbor_score > current_score:
                IF crowding < crowd_factor:
                    MOVE towards best_neighbor
                ELSE:
                    RANDOM new mask
            ELSE:
                RANDOM new mask

            EVALUATE new_position
            IF better than current:
                UPDATE population[i] and best_solution if needed

        APPEND best_accuracy to convergence

    EXTRACT indices and names of selected features from best_solution
    RETURN selected_features, selected_indices, convergence, iteration_count

# ================================
# SECTION 4: Evaluation Utilities
# ================================
DEFINE FUNCTION train_and_evaluate(X_train, y_train, X_test, y_test, indices,
name):
    TRAIN RandomForest (100 trees) on X_train[:, indices]
    PREDICT on training set → train_preds
    PRINT accuracy and F1 score
    RETURN trained classifier

DEFINE FUNCTION final_evaluate(clf, X_test, y_test, indices, name):
    PREDICT and PROBABILITIES on X_test[:, indices]
    COMPUTE: Accuracy, Precision, Recall, F1, AUC
    COMPUTE Confusion Matrix → TP, TN, FP, FN
    PRINT all metrics
    RETURN all values
```

```
DEFINE FUNCTION plot_convergence(convergence, title):
    PLOT convergence vs iteration

DEFINE FUNCTION plot_conf_matrix(y_true, y_pred, title):
    DISPLAY confusion matrix

DEFINE FUNCTION plot_roc_curve(y_true, y_scores, title):
    DISPLAY ROC curve

# ================================
# SECTION 5: Run Fish Swarm Feature Selection
# ================================
CALL run_fish_swarm → selected_features, selected_indices, convergence, iterations
CALL train_and_evaluate using selected_indices → clf
CALL final_evaluate using clf and selected_indices → acc, prec, rec, f1, auc, tp, tn, fp, fn
PLOT confusion matrix, ROC curve, and convergence plot

# ================================
# SECTION 6: Summary Output
# ================================
PRINT selected feature names
CREATE summary_df with:
    Algorithm, Accuracy, Precision, Recall, F1 Score, AUC, Selected Features Count, Iterations
PRINT summary_df

END
```

APPENDIX D Bee Colony Algorithm Code

```
# ================================
# SECTION 1: Setup and Imports
# ================================
from google.colab import drive
drive.mount('/content/drive')

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay, RocCurveDisplay

# ================================
# SECTION 2: Load and Preprocess Data
# ================================
base_path = '/content/drive/MyDrive/IDS_BioInspired_Colab_Final'
data_path = os.path.join(base_path, 'data')

train_df = pd.read_csv(os.path.join(data_path, 'UNSW_NB15_training-set.csv'))
test_df = pd.read_csv(os.path.join(data_path, 'UNSW_NB15_testing-set.csv'))

drop_cols = ['label', 'attack_cat']
X_train_raw = train_df.drop(columns=drop_cols)
X_test_raw = test_df.drop(columns=drop_cols)
y_train = train_df['label']
y_test = test_df['label']
```

```
numeric_cols = X_train_raw.select_dtypes(include=[np.number]).columns.tolist()
X_train_encoded = X_train_raw[numeric_cols]
X_test_encoded = X_test_raw[numeric_cols]

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train_encoded)
X_test_scaled = scaler.transform(X_test_encoded)
feature_names = X_train_encoded.columns.tolist()

# ================================
# SECTION 3: Bee Colony Algorithm
# ================================
def run_bee_colony(X_train, X_test, y_train, y_test, feature_names):
    num_features = len(feature_names)
    num_bees = 10
    max_iterations = 10
    limit = 10
    no_improvement_limit = 20

    no_improve_count = 0
    prev_best_accuracy = 0
    best_accuracy = 0
    best_solution = None

    population = [np.random.choice([0, 1], size=num_features, p=[0.7, 0.3]) for _ in
range(num_bees)]
    fitness = [0] * num_bees
    trial_counter = [0] * num_bees
    convergence = []
    evaluated = {}

    def evaluate(mask):
        key = tuple(mask)
        if key in evaluated:
```

```
            return evaluated[key]
        idx = [i for i, bit in enumerate(mask) if bit == 1]
        if not idx:
            evaluated[key] = 0
            return 0
        clf = RandomForestClassifier(n_estimators=30,
random_state=np.random.randint(0, 100000))
        acc = cross_val_score(clf, X_train[:, idx], y_train, cv=2, scoring='accuracy',
n_jobs=-1).mean()
        evaluated[key] = acc
        return acc

    # Initial fitness
    for i in range(num_bees):
        fitness[i] = evaluate(population[i])
        if fitness[i] > best_accuracy:
            best_accuracy = fitness[i]
            best_solution = population[i].copy()

    for iteration in range(max_iterations):
        # Phase 1: Employed Bees
        for i in range(num_bees):
            neighbor = population[i].copy()
            flip_idx = np.random.randint(0, num_features)
            neighbor[flip_idx] ^= 1
            acc_new = evaluate(neighbor)
            if acc_new > fitness[i]:
                population[i] = neighbor
                fitness[i] = acc_new
                trial_counter[i] = 0
            else:
                trial_counter[i] += 1
            if fitness[i] > best_accuracy:
                best_accuracy = fitness[i]
```

```
        best_solution = population[i].copy()

# Phase 2: Onlooker Bees
probs = [f / sum(fitness) if sum(fitness) > 0 else 0 for f in fitness]
for _ in range(num_bees):
    i = np.random.choice(range(num_bees), p=probs)
    neighbor = population[i].copy()
    flip_idx = np.random.randint(0, num_features)
    neighbor[flip_idx] ^= 1
    acc_new = evaluate(neighbor)
    if acc_new > fitness[i]:
        population[i] = neighbor
        fitness[i] = acc_new
        trial_counter[i] = 0
    else:
        trial_counter[i] += 1
    if fitness[i] > best_accuracy:
        best_accuracy = fitness[i]
        best_solution = population[i].copy()

# Phase 3: Scout Bees
for i in range(num_bees):
    if trial_counter[i] >= limit:
        population[i] = np.random.choice([0, 1], size=num_features, p=[0.7, 0.3])
        fitness[i] = evaluate(population[i])
        trial_counter[i] = 0
        if fitness[i] > best_accuracy:
            best_accuracy = fitness[i]
            best_solution = population[i].copy()

convergence.append(round(best_accuracy, 4))

# Early Stopping
if best_accuracy > prev_best_accuracy:
```

```
            prev_best_accuracy = best_accuracy
            no_improve_count = 0
        else:
            no_improve_count += 1
        if no_improve_count >= no_improvement_limit:
            print(f"Bee Colony stopped at iteration {iteration + 1}")
            break
    idx = [i for i, bit in enumerate(best_solution) if bit == 1]
    selected_features = [feature_names[i] for i in idx]
    return selected_features, idx, convergence, iteration + 1
# ================================
# SECTION 4: Evaluation Utilities
# ================================
def train_and_evaluate(X_train, y_train, X_test, y_test, idx, algorithm_name):
    clf = RandomForestClassifier(n_estimators=30,
random_state=np.random.randint(0, 100000), verbose=1, n_jobs=-1)
    clf.fit(X_train[:, idx], y_train)
    train_preds = clf.predict(X_train[:, idx])
    print(f"\n🔹 {algorithm_name} - Training Set Evaluation")
    print("Accuracy:", round(accuracy_score(y_train, train_preds), 2))
    print("F1 Score:", round(f1_score(y_train, train_preds), 2))
    return clf

def final_evaluate(clf, X_test, y_test, idx, algorithm_name):
    test_preds = clf.predict(X_test[:, idx])
    probs = clf.predict_proba(X_test[:, idx])[:, 1]
    acc = round(accuracy_score(y_test, test_preds), 2)
    prec = round(precision_score(y_test, test_preds), 2)
    rec = round(recall_score(y_test, test_preds), 2)
    f1 = round(f1_score(y_test, test_preds), 2)
    auc = round(roc_auc_score(y_test, probs), 2)
    tn, fp, fn, tp = confusion_matrix(y_test, test_preds).ravel()
    print(f"\n 🟩{algorithm_name} - Final Evaluation on Test Set")
```

```
    print(f"Accuracy: {acc}, Precision: {prec}, Recall: {rec}, F1: {f1}, AUC: {auc}")
    print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")
    return acc, prec, rec, f1, auc, tp, tn, fp, fn

def plot_convergence(convergence, title):
    plt.plot(convergence, marker='o')
    plt.title(f"{title} - Convergence Plot")
    plt.xlabel("Iteration")
    plt.ylabel("Accuracy")
    plt.grid()
    plt.show()

def plot_conf_matrix(y_true, y_pred, title):
    ConfusionMatrixDisplay.from_predictions(y_true, y_pred, cmap='Blues')
    plt.title(f"{title} - Confusion Matrix")
    plt.grid(False)
    plt.show()

def plot_roc_curve(y_true, y_scores, title):
    RocCurveDisplay.from_predictions(y_true, y_scores)
    plt.title(f"{title} - ROC Curve")
    plt.grid(True)
    plt.show()
#================================
# SECTION 5: Run Bee Colony
# ================================
bc_features, bc_idx, bc_convergence, bc_iters = run_bee_colony(
    X_train_scaled, X_test_scaled, y_train, y_test, feature_names
)

clf_bc = train_and_evaluate(
    X_train_scaled, y_train, X_test_scaled, y_test, bc_idx, "Bee Colony"
)
```

```
acc, prec, rec, f1, auc, tp, tn, fp, fn = final_evaluate(
    clf_bc, X_test_scaled, y_test, bc_idx, "Bee Colony"
)

plot_conf_matrix(y_test, clf_bc.predict(X_test_scaled[:, bc_idx]), "Bee Colony")
plot_roc_curve(y_test, clf_bc.predict_proba(X_test_scaled[:, bc_idx])[:, 1], "Bee
Colony")
plot_convergence(bc_convergence, "Bee Colony")

print("\n🔶 Features selected by Bee Colony:")
print(bc_features)

# ================================
# SECTION 6: Print Summary
# ================================
summary_df = pd.DataFrame({
    'Algorithm': ['Bee Colony'],
    'Accuracy': [acc],
    'Precision': [prec],
    'Recall': [rec],
    'F1 Score': [f1],
    'AUC': [auc],
    'Selected Features': [len(bc_features)],
    'Iterations': [bc_iters]
})

print(f"\n🐝 Bee Colony Performance Summary:")
print(summary_df)
```

APPENDIX E Fish Swarm Algorithm Code

```
# ================================
# SECTION 1: Setup and Imports
# ================================
from google.colab import drive
drive.mount('/content/drive')

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay, RocCurveDisplay

# ================================
# SECTION 2: Load and Preprocess Data (Numeric Only)
# ================================
base_path = '/content/drive/MyDrive/IDS_BioInspired_Colab_Final'
data_path = os.path.join(base_path, 'data')
train_file = os.path.join(data_path, 'UNSW_NB15_training-set.csv')
test_file = os.path.join(data_path, 'UNSW_NB15_testing-set.csv')

train_df = pd.read_csv(train_file)
test_df = pd.read_csv(test_file)

drop_cols = ['label', 'attack_cat']
X_train_raw = train_df.drop(columns=drop_cols)
X_test_raw = test_df.drop(columns=drop_cols)
y_train = train_df['label']
```

```
y_test = test_df['label']

X_train_raw = X_train_raw.select_dtypes(include=[np.number])
X_test_raw = X_test_raw.select_dtypes(include=[np.number])

X_all = pd.concat([X_train_raw, X_test_raw])
scaler = MinMaxScaler()
X_all_scaled = scaler.fit_transform(X_all)

X_train_scaled = X_all_scaled[:len(X_train_raw), :]
X_test_scaled = X_all_scaled[len(X_train_raw):, :]
feature_names = X_train_raw.columns.tolist()

# ================================
# SECTION 3: Fish Swarm Algorithm (AFSA Based + Global Early Stop)
# ================================
def run_fish_swarm(X_train, X_test, y_train, y_test, feature_names):
    num_features = len(feature_names)
    num_fish = 10
    max_iterations = 20
    no_improvement_limit = 20
    visual_distance = 5
    step_size = 1
    crowd_factor = 0.6

    population = [np.random.choice([0, 1], size=num_features, p=[0.7, 0.3]) for _ in
range(num_fish)]
    best_solution = None
    best_accuracy = 0
    prev_best_accuracy = 0
    no_improve_count = 0
    convergence = []
    evaluated = {}
```

```
    def evaluate(mask):
        key = tuple(mask)
        if key in evaluated:
            return evaluated[key]
        idx = [i for i, bit in enumerate(mask) if bit == 1]
        if not idx:
            evaluated[key] = 0
            return 0
        clf = RandomForestClassifier(n_estimators=30,
random_state=np.random.randint(0, 100000))
        acc = cross_val_score(clf, X_train[:, idx], y_train, cv=2, scoring='accuracy',
n_jobs=-1).mean()
        evaluated[key] = acc
        return acc

    def get_neighbors(index):
        indices = [i for i in range(num_fish) if i != index]
        np.random.shuffle(indices)
        return [population[i] for i in indices[:visual_distance]]

    def move_towards(current, target):
        new = current.copy()
        diff_indices = [i for i in range(num_features) if current[i] != target[i]]
        np.random.shuffle(diff_indices)
        for i in diff_indices[:step_size]:
            new[i] = target[i]
        return new

    for iteration in range(max_iterations):
        for i in range(num_fish):
            current = population[i]
            neighbors = get_neighbors(i)
            neighbor_scores = [(evaluate(neighbor), neighbor) for neighbor in neighbors]
            neighbor_scores.sort(reverse=True, key=lambda x: x[0])
```

```
            if neighbor_scores:
                best_neighbor_score, best_neighbor = neighbor_scores[0]
                acc_current = evaluate(current)
                if best_neighbor_score > acc_current:
                    crowd = sum(np.array_equal(best_neighbor, p) for p in population) /
num_fish
                    if crowd < crowd_factor:
                        new_position = move_towards(current, best_neighbor)
                    else:
                        new_position = np.random.choice([0, 1], size=num_features, p=[0.7,
0.3])

                else:
                    new_position = np.random.choice([0, 1], size=num_features, p=[0.7,
0.3])
            else:
                new_position = np.random.choice([0, 1], size=num_features, p=[0.7, 0.3])

            acc_new = evaluate(new_position)
            acc_old = evaluate(current)
            if acc_new > acc_old:
                population[i] = new_position
                if acc_new > best_accuracy:
                    best_accuracy = acc_new
                    best_solution = new_position.copy()

        convergence.append(round(best_accuracy, 4))

        if best_accuracy > prev_best_accuracy:
            prev_best_accuracy = best_accuracy
            no_improve_count = 0
        else:
            no_improve_count += 1
```

```
        if no_improve_count >= no_improvement_limit:
            print(f"Fish Swarm stopped early at iteration {iteration + 1}")
            break

    idx = [i for i, bit in enumerate(best_solution) if bit == 1]
    selected_features = [feature_names[i] for i in idx]
    return selected_features, idx, convergence, iteration + 1

# ================================
# SECTION 4: Evaluation Utilities
# ================================
def train_and_evaluate(X_train, y_train, X_test, y_test, idx, algorithm_name):
    clf = RandomForestClassifier(n_estimators=30,
random_state=np.random.randint(0, 100000), verbose=1, n_jobs=-1)
    clf.fit(X_train[:, idx], y_train)
    train_preds = clf.predict(X_train[:, idx])
    print(f"\n🔷 {algorithm_name} - Training Set Evaluation")
    print("Accuracy:", round(accuracy_score(y_train, train_preds), 2))
    print("F1 Score:", round(f1_score(y_train, train_preds), 2))
    return clf

def final_evaluate(clf, X_test, y_test, idx, algorithm_name):
    test_preds = clf.predict(X_test[:, idx])
    probs = clf.predict_proba(X_test[:, idx])[:, 1]

    acc = round(accuracy_score(y_test, test_preds), 2)
    prec = round(precision_score(y_test, test_preds), 2)
    rec = round(recall_score(y_test, test_preds), 2)
    f1 = round(f1_score(y_test, test_preds), 2)
    auc = round(roc_auc_score(y_test, probs), 2)
    tn, fp, fn, tp = confusion_matrix(y_test, test_preds).ravel()

    print(f"\n {algorithm_name} - Final Evaluation on Test Set")
```

```
    print(f"Accuracy: {acc}, Precision: {prec}, Recall: {rec}, F1: {f1}, AUC: {auc}")
    print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")
    return acc, prec, rec, f1, auc, tp, tn, fp, fn

def plot_convergence(convergence, title):
    plt.plot(convergence, marker='o')
    plt.title(f"{title} - Convergence Plot")
    plt.xlabel("Iteration")
    plt.ylabel("Accuracy")
    plt.grid()
    plt.show()

def plot_conf_matrix(y_true, y_pred, title):
    ConfusionMatrixDisplay.from_predictions(y_true, y_pred, cmap='Blues')
    plt.title(f"{title} - Confusion Matrix")
    plt.grid(False)
    plt.show()

def plot_roc_curve(y_true, y_scores, title):
    RocCurveDisplay.from_predictions(y_true, y_scores)
    plt.title(f"{title} - ROC Curve")
    plt.grid(True)
    plt.show()

# ================================
# SECTION 5: Run Fish Swarm
# ================================
fs_features, fs_idx, fs_convergence, fs_iters = run_fish_swarm(
    X_train_scaled, X_test_scaled, y_train, y_test, feature_names
)
clf_fs = train_and_evaluate(
    X_train_scaled, y_train, X_test_scaled, y_test, fs_idx, "Fish Swarm")
acc2, prec2, rec2, f12, auc2, tp2, tn2, fp2, fn2 = final_evaluate(
    clf_fs, X_test_scaled, y_test, fs_idx, "Fish Swarm")
```

```
plot_conf_matrix(y_test, clf_fs.predict(X_test_scaled[:, fs_idx]), "Fish Swarm")
plot_roc_curve(y_test, clf_fs.predict_proba(X_test_scaled[:, fs_idx])[:, 1], "Fish Swarm")
plot_convergence(fs_convergence, "Fish Swarm")

# ================================
# SECTION 6: Print Features and Summary
# ================================
algorithm_name = "Fish Swarm"
selected_features = fs_features
acc = acc2
prec = prec2
rec = rec2
f1 = f12
auc = auc2
iters = fs_iters
print(f"\n🔸 Features selected by {algorithm_name}:")
print(selected_features)
summary_df = pd.DataFrame({
    'Algorithm': [algorithm_name],
    'Accuracy': [acc],
    'Precision': [prec],
    'Recall': [rec],
    'F1 Score': [f1],
    'AUC': [auc],
    'Selected Features': [len(selected_features)],
    'Iterations': [iters]
})

print(f"\n📊 {algorithm_name} Performance Summary:")
print(summary_df)
```

APPENDIX F Test Results

Bee Colony Algorithm Test Results

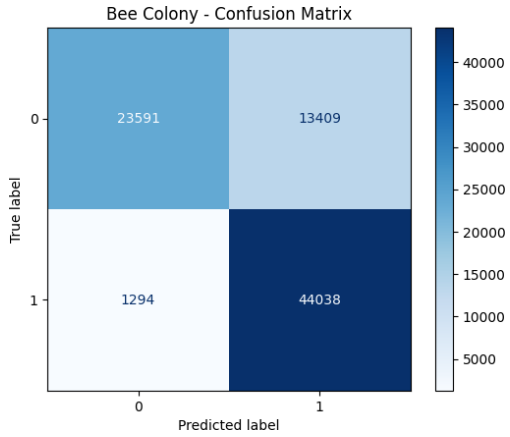
	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average
T. Acc	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
Accuracy	0.82	0.84	0.85	0.84	0.85	0.84	0.85	0.84	0.84	0.84	0.84
T. F1-Score	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
F1- Score	0.86	0.87	0.88	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
Precision	0.77	0.79	0.8	0.79	0.79	0.78	0.8	0.79	0.79	0.79	0.79
Recall	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
AUC	0.91	0.93	0.94	0.92	0.94	0.94	0.92	0.92	0.92	0.94	0.93
TP	44038	44093	44011	43853	44021	44168	44021	44076	43901	44045	44022.7
TN	23591	25403	25845	25679	25575	24699	25662	25413	25233	25323	25242.3
FP	13409	11597	11155	11321	11425	12301	11338	11587	11767	11677	11757.7
FN	1294	1239	1321	1479	1311	1164	1311	1256	1431	1287	1309.3
Iterations	37	51	49	47	33	54	39	41	33	31	41.5
No Feature Sel	11	14	13	14	15	13	17	17	10	16	14

Fish Swarm Algorithm Test Results

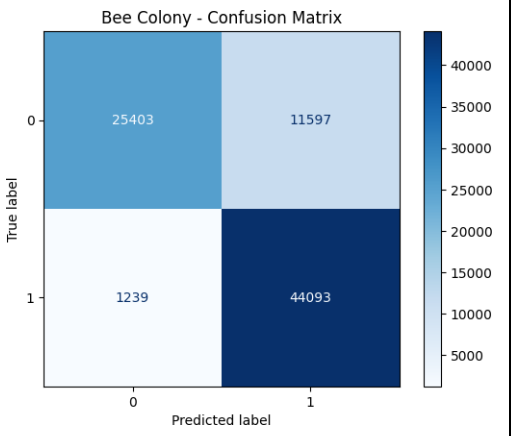
	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average
T. Acc	0.99	0.98	0.99	0.98	0.98	0.98	0.98	1	0.98	0.98	0.984
Accuracy	0.87	0.84	0.87	0.84	0.83	0.84	0.85	0.87	0.84	0.84	0.849
T. F1-Score	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1	0.99	0.99	0.991
F1- Score	0.89	0.87	0.89	0.87	0.86	0.87	0.88	0.89	0.87	0.87	0.876
Precision	0.82	0.79	0.83	0.79	0.77	0.79	0.8	0.83	0.78	0.79	0.799
Recall	0.97	0.96	0.96	0.97	0.97	0.97	0.97	0.96	0.98	0.97	0.968
AUC	0.97	0.88	0.96	0.92	0.9	0.92	0.92	0.97	0.92	0.92	0.928
TP	43780	43509	43662	44004	43942	43844	43982	43685	44287	44160	43885.5
TN	27552	25774	28289	25297	224065	25480	25819	27836	24739	25059	45991
FP	9448	11226	8711	11703	12935	11520	11181	9164	12261	11941	11009
FN	1552	1823	1670	1328	1390	1488	1350	1647	1045	1172	1446.5
Iterations	40	29	56	44	67	62	105	50	64	47	56.4
No Feature Sel	7	10	12	13	12	12	13	14	11	11	11.5

APPENDIX G Confusion Matrix (Bee Colony Algorithm)

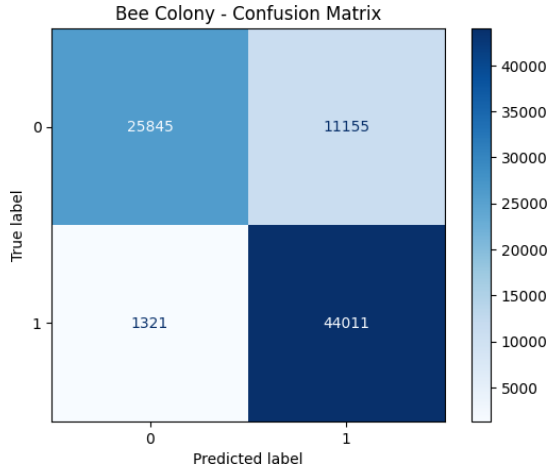
Test 1



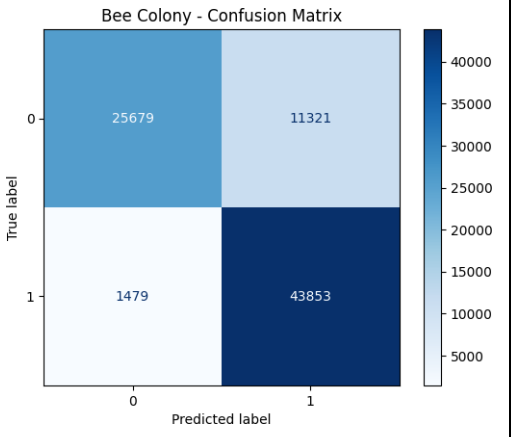
Test 2



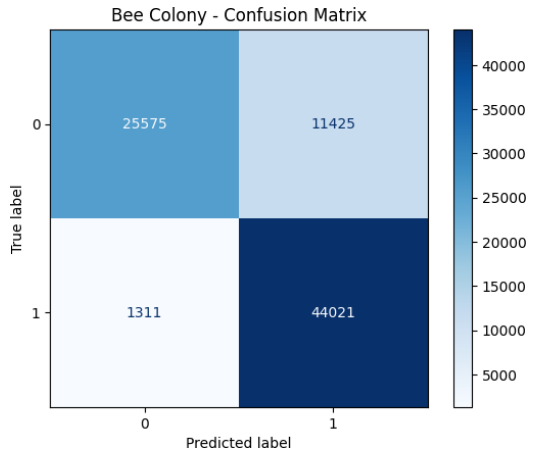
Test 3



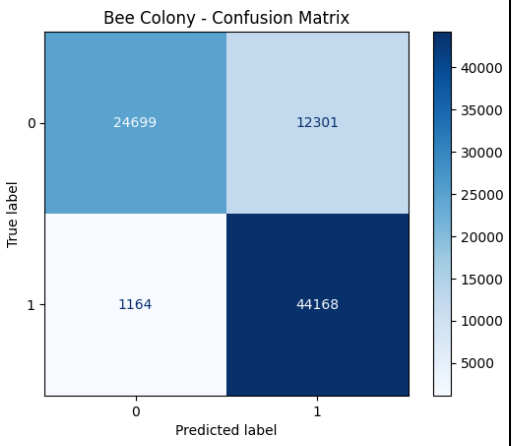
Test 4



Test 5

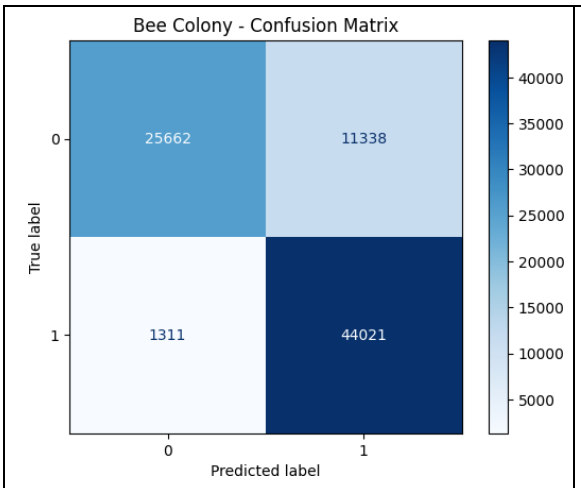


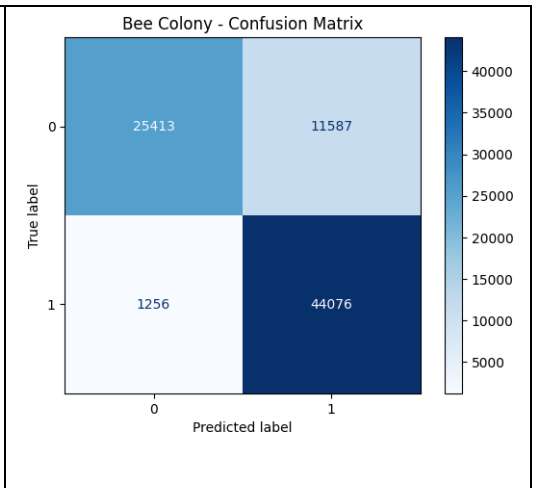
Test 6



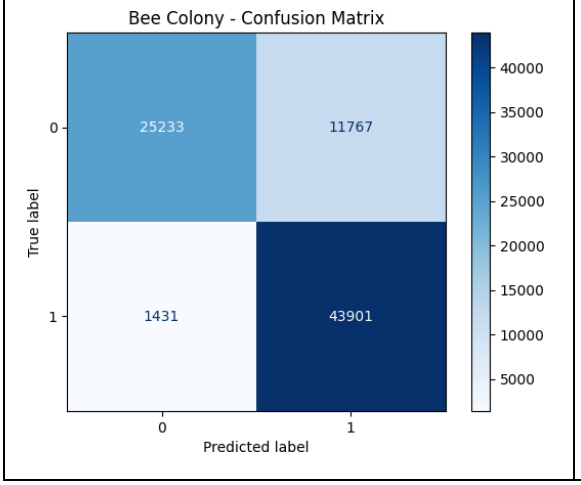
Test 7

Test 8

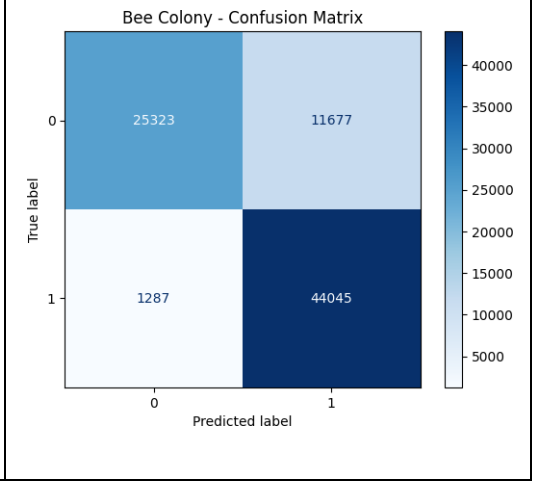




Test 9

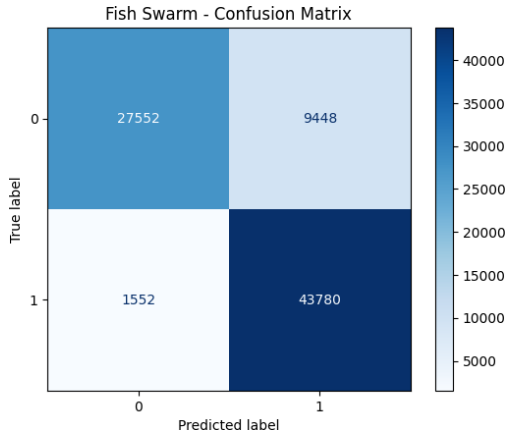


Test 10

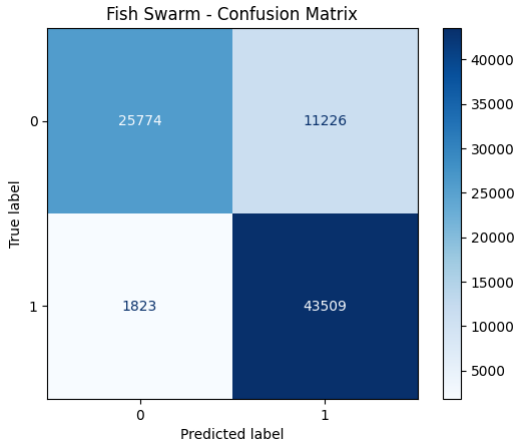


APPENDIX H Confusion Matrix (Fish Swarm Algorithm)

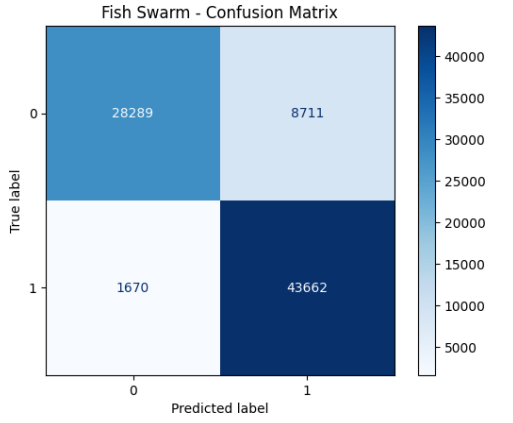
Test 1



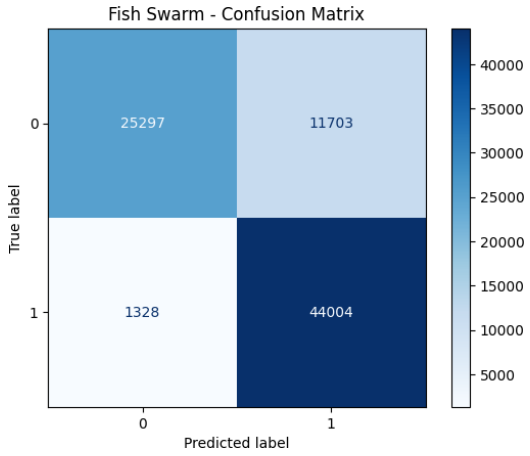
Test 2



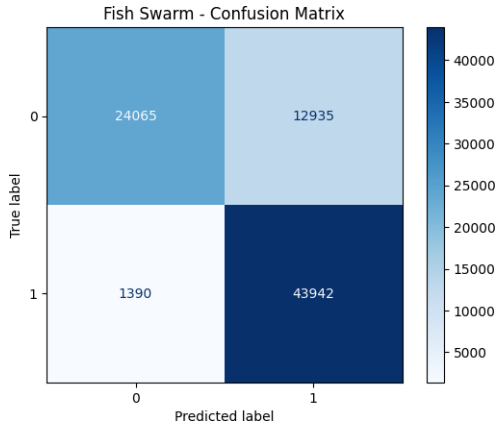
Test 3



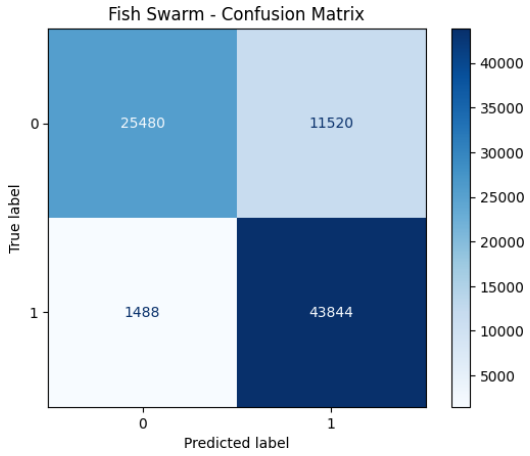
Test 4



Test 5

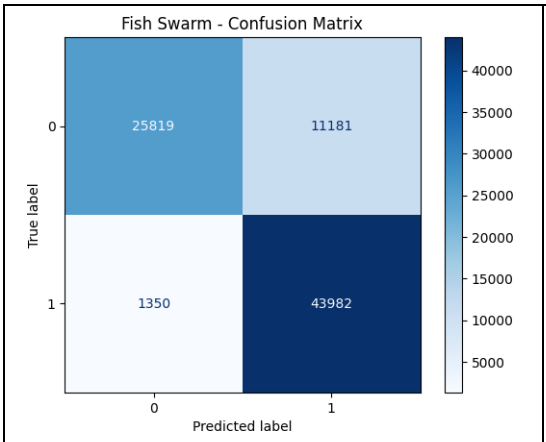


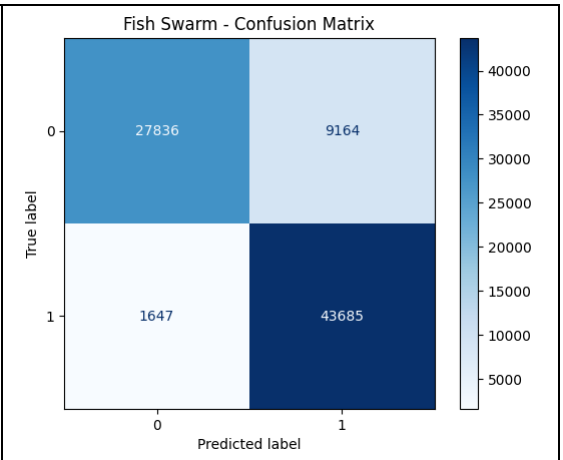
Test 6



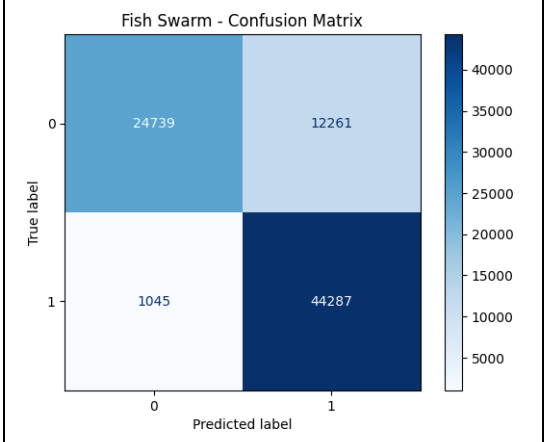
Test 7

Test 8

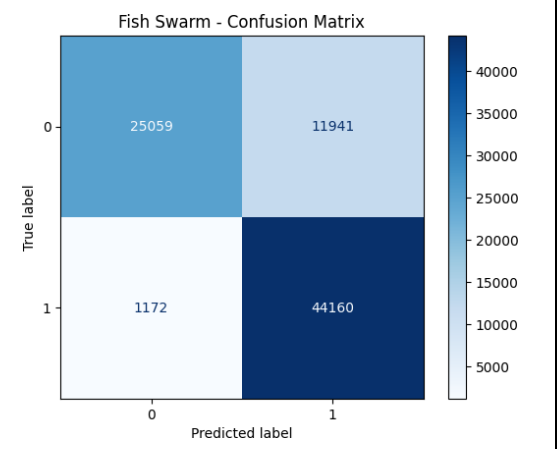




Test 9

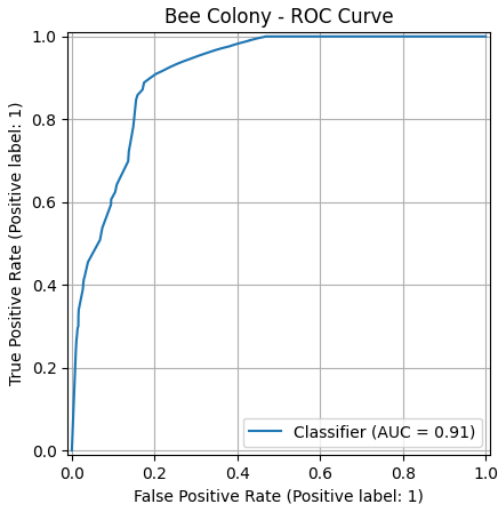


Test 10

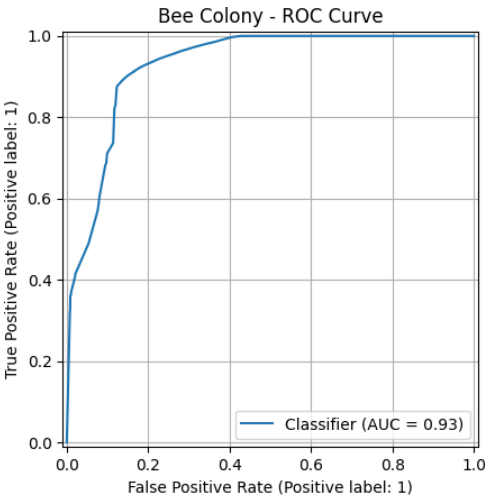


APPENDIX I ROC Curve (Bee Colony Algorithm)

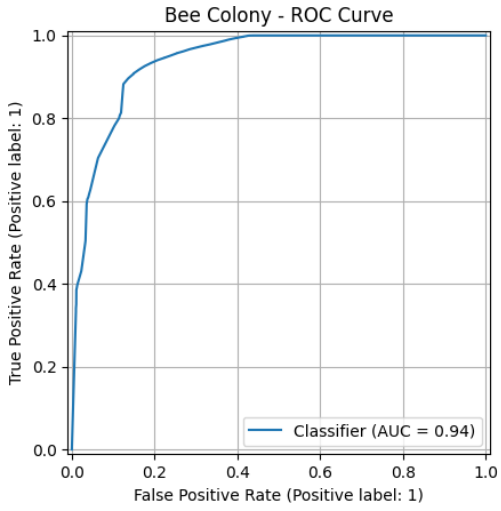
Test 1



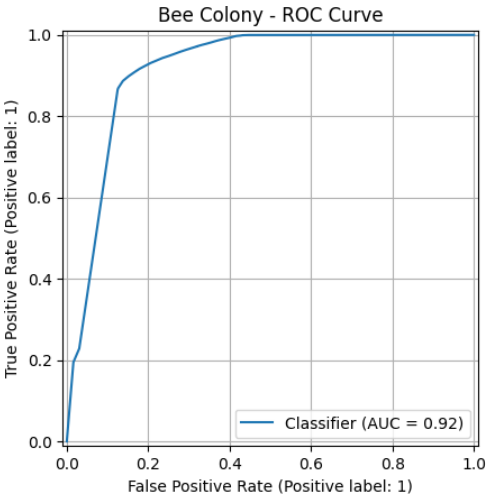
Test 2



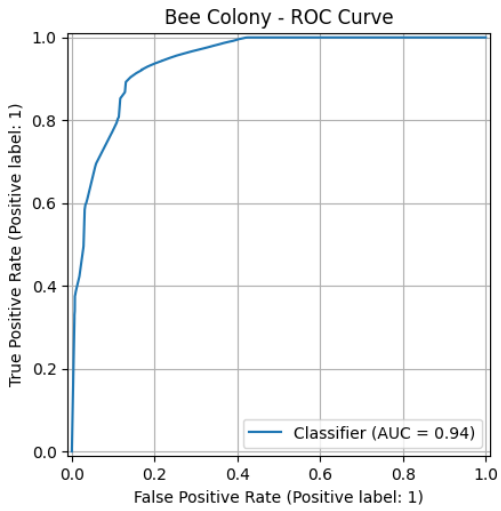
Test 3



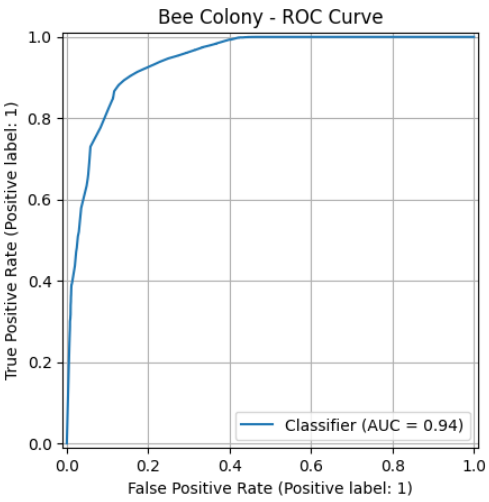
Test 4



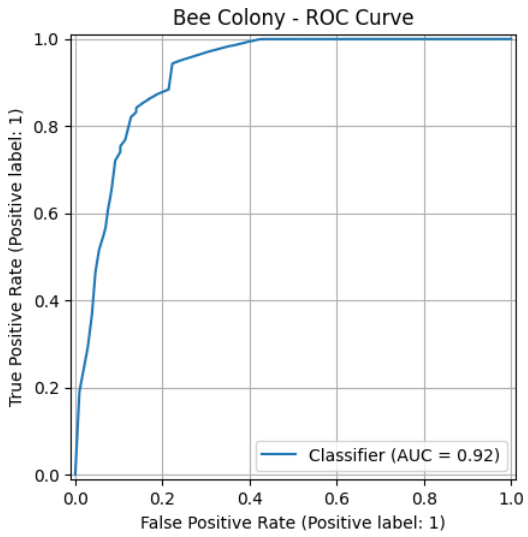
Test 5



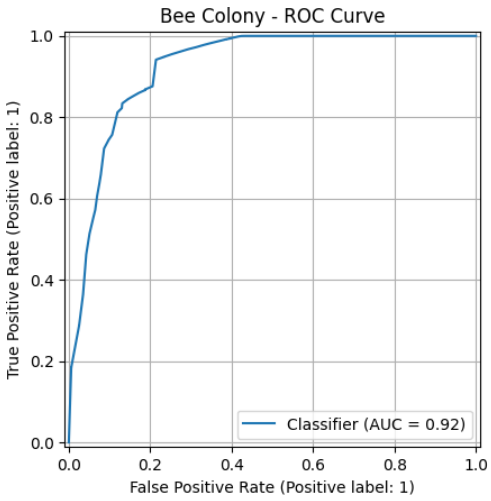
Test 6



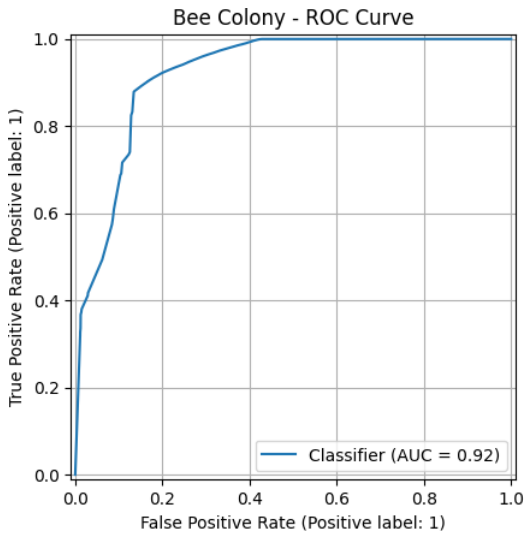
Test 7



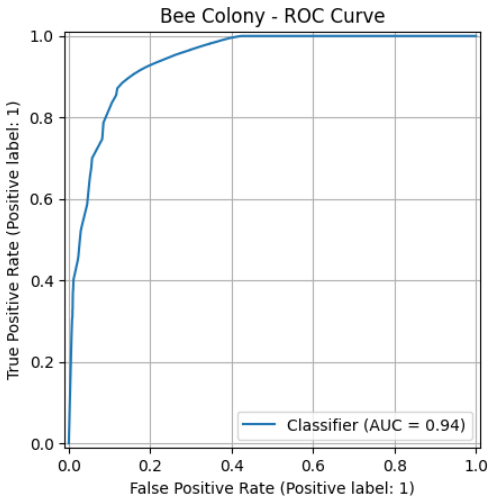
Test 8



Test 9

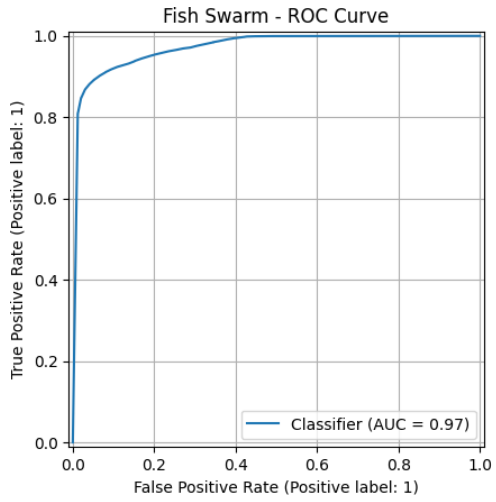


Test 10

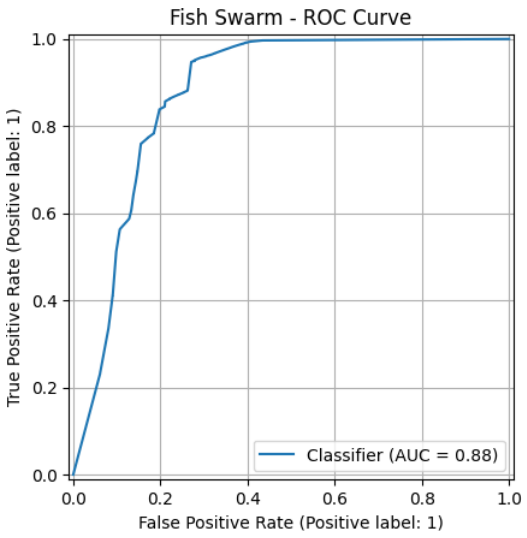


APPENDIX J ROC Curve (Fish Swarm Algorithm)

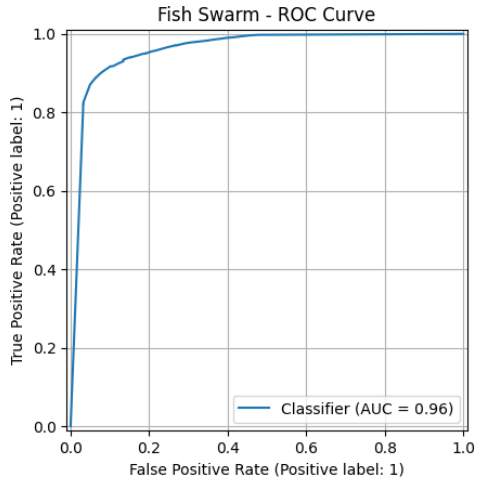
Test 1



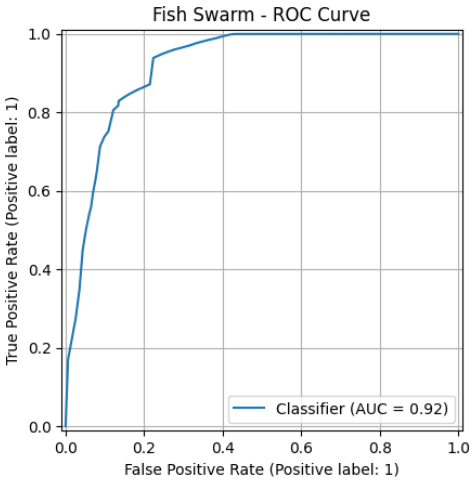
Test 2



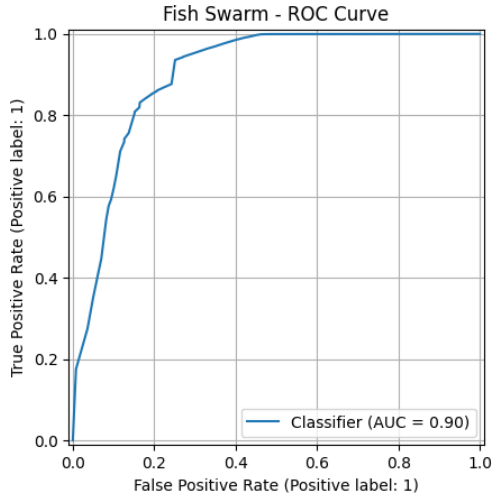
Test 3



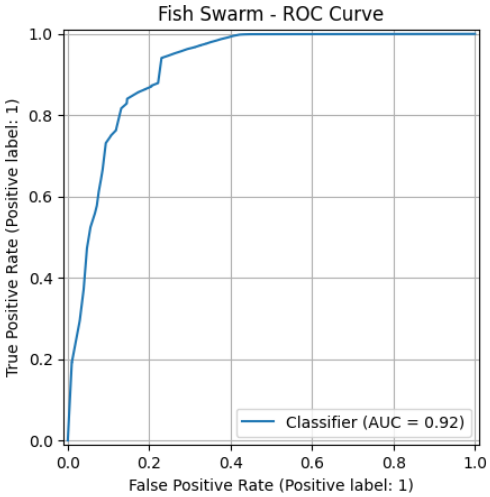
Test 4



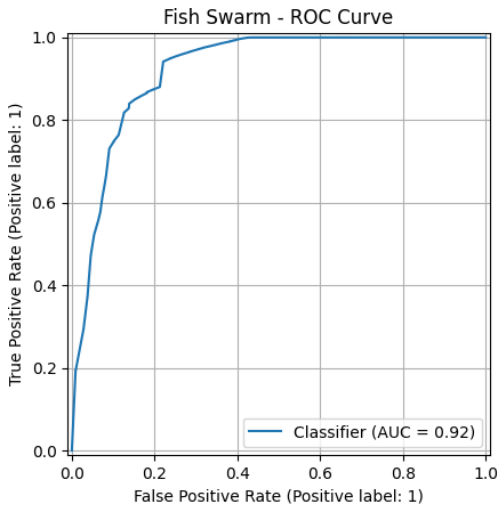
Test 5



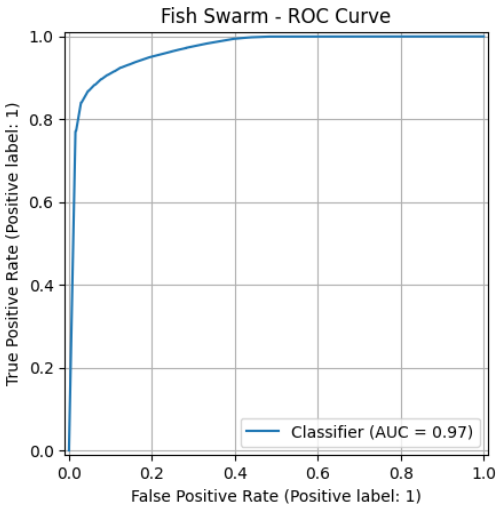
Test 6



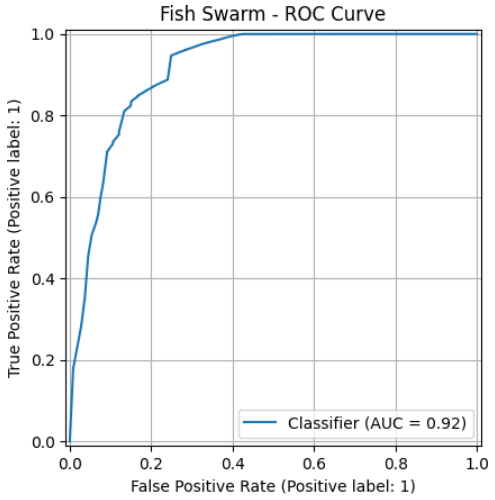
Test 7



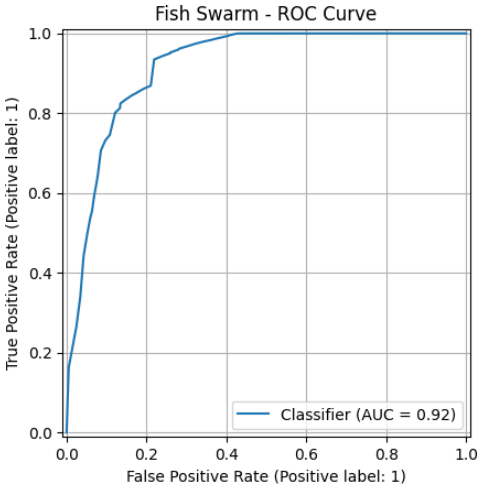
Test 8



Test 9

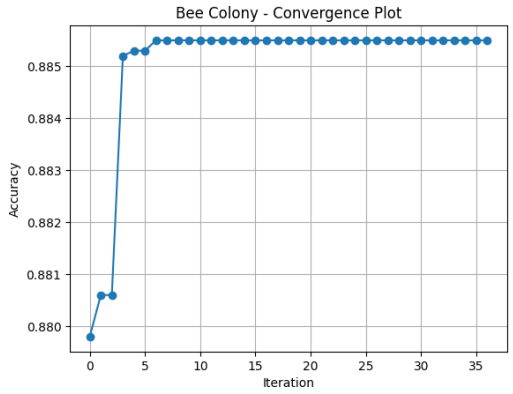


Test 10

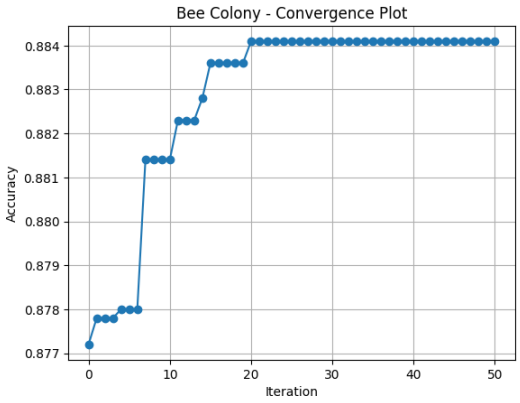


APPENDIX K Convergence Plot (Bee Colony)

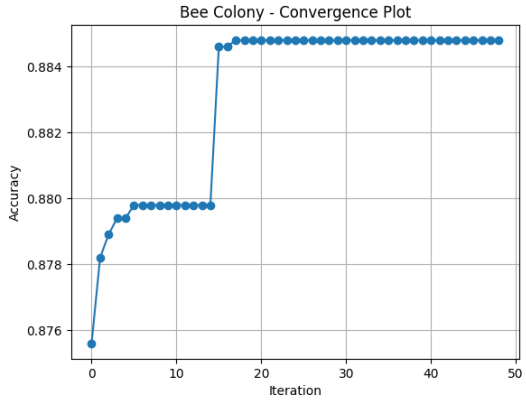
Test 1



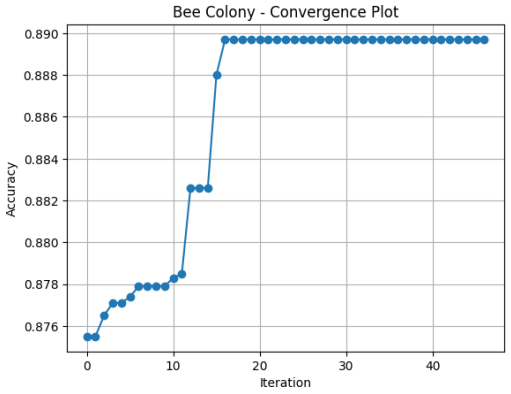
Test 2



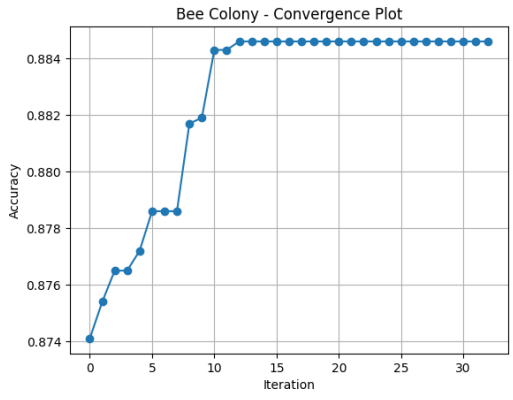
Test 3



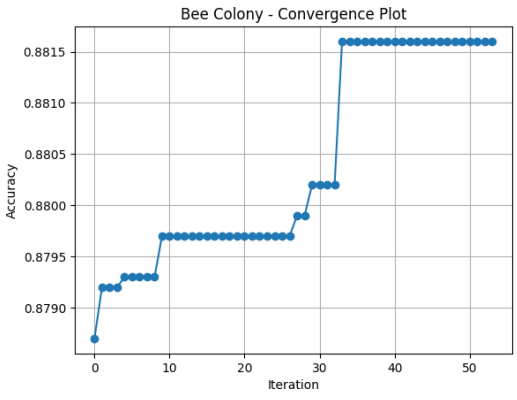
Test 4



Test 5

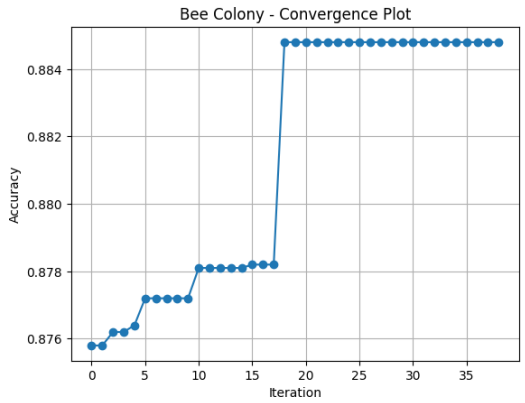


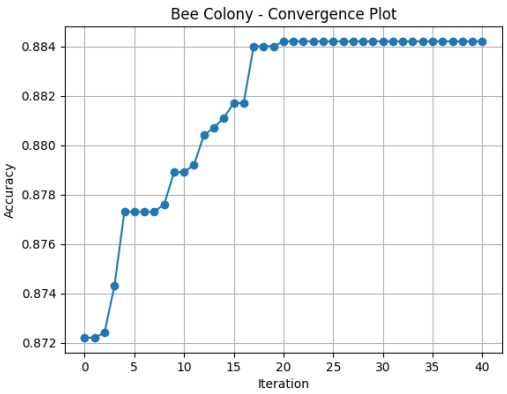
Test 6



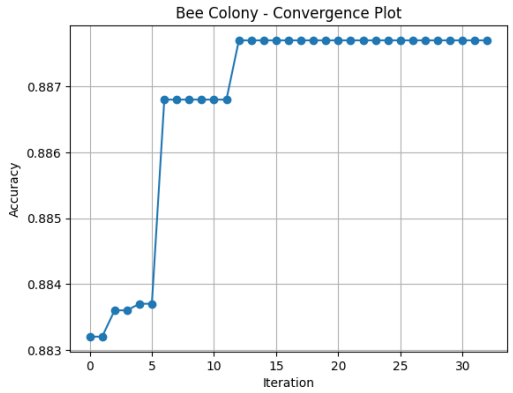
Test 7

Test 8

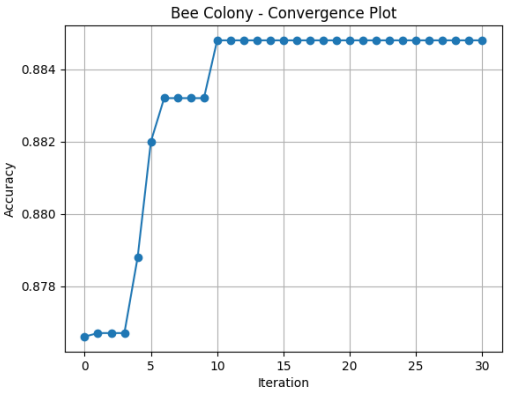




Test 9

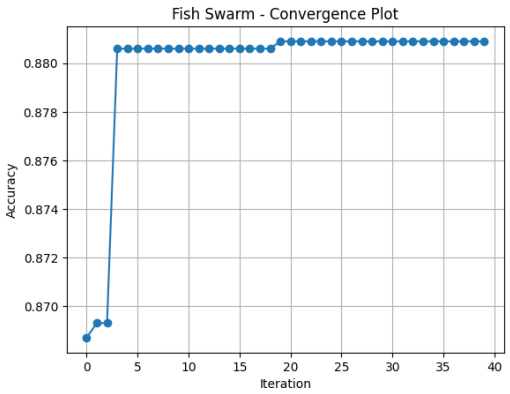


Test 10

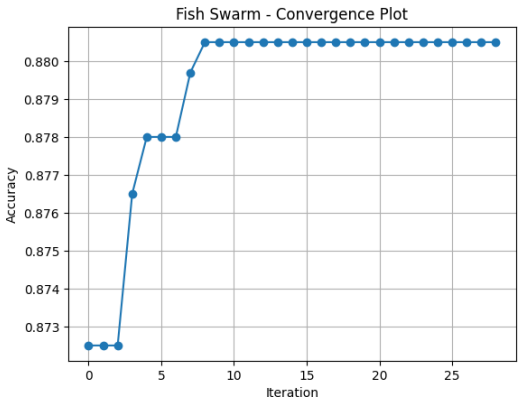


APPENDIX L Convergence Plot (Fish Swarm Algorithm)

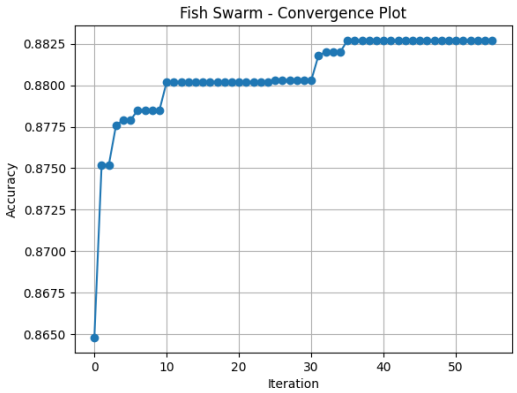
Test 1



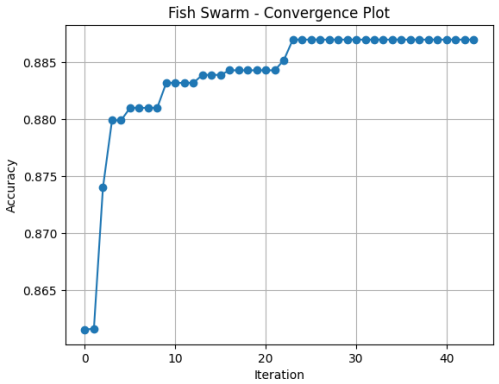
Test 2



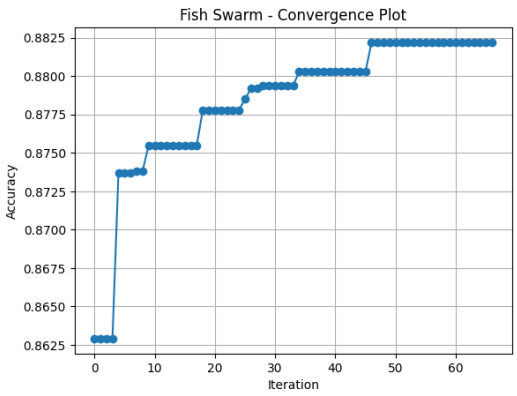
Test 3



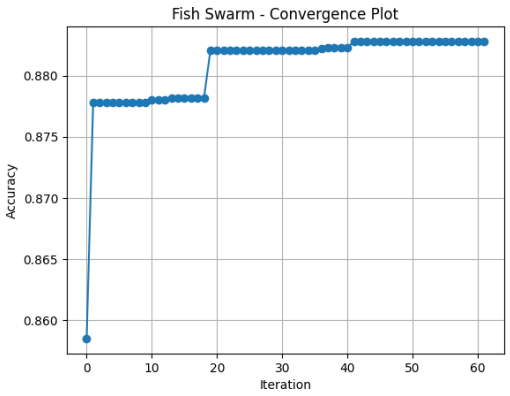
Test 4



Test 5

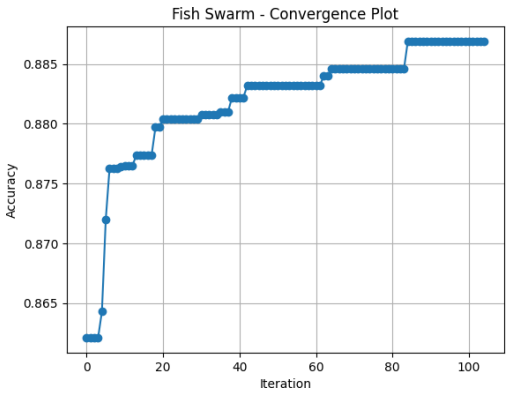


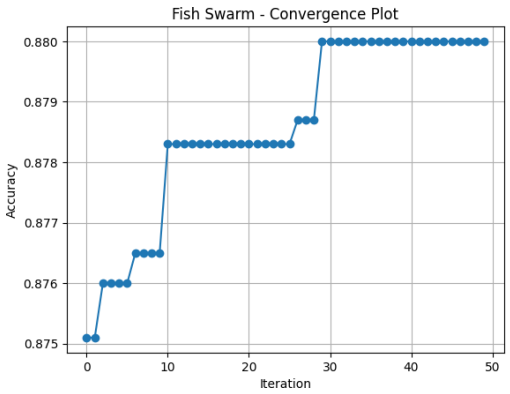
Test 6



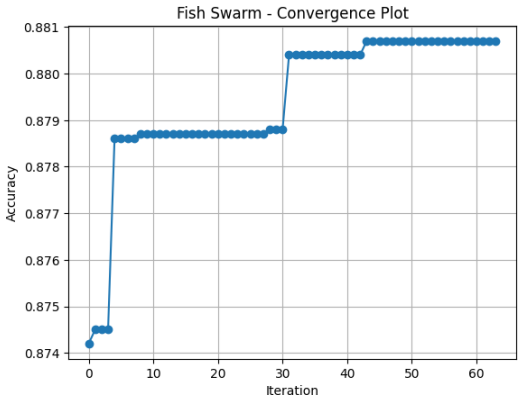
Test 7

Test 8





Test 9



Test 10