**CHAPTER 1**

**INTRODUCTION**

**1.1 INTRODUCTION TO THE PROJECT**

Autism Spectrum Disorder (ASD) is a complex neurological condition marked by a range of symptoms, including challenges with social interaction, communication difficulties, and repetitive or restricted behaviors. The condition presents differently in everyone, making diagnosis and treatment more challenging. The "spectrum" aspect of ASD indicates a wide variability in symptoms and severity, making identification and diagnosis challenging. Early detection is crucial, as it enables timely intervention and support, which can significantly influence the developmental path and quality of life of individuals with ASD. Despite advancements in medical science, diagnosing autism spectrum disorder (ASD) remains a complex process often reliant on subjective clinical opinions. This complexity has driven researchers to explore advanced computational techniques, such as machine learning and bio-inspired optimization algorithms, to enhance the accuracy and objectivity of ASD diagnoses. These technologies can analyze large datasets to uncover patterns and relationships that traditional diagnostic methods might overlook.

In this paper, exploring the application of bio-inspired optimization methods, specifically Particle Swarm Optimization (PSO) and the Firefly Algorithm, for feature selection in Autism Spectrum Disorder (ASD) classification. These bio-inspired techniques draw from natural processes to solve complex optimization problems. The Firefly Algorithm is inspired by the flashing behaviour of fireflies, while PSO is based on the social behaviours observed in bird flocks or fish schools. Both approaches have demonstrated effectiveness in various optimization tasks, thanks to their capacity to efficiently explore and exploit the search space. Feature selection is an essential preprocessing step in machine learning that involves identifying and choosing a subset of relevant features from a dataset while eliminating those that are unnecessary or irrelevant. This process improves model performance by reducing data dimensionality, making models more efficient and less prone to overfitting. In this study, using the Toddler Autism dataset from July 2018, which includes a diverse range of factors such as demographic information, behavioural indicators, and responses to various screening questions. This dataset is an invaluable resource for our research.

The selected attributes are utilized to train and assess various classification techniques, including ANN MLP, GaussianNB, K-Nearest Neighbors (KNN), and Random Forest. These classifiers were chosen due to their unique learning and classification methodologies. ANN MLP, a machine learning model, can capture complex relationships within data. GaussianNB is a probabilistic model grounded in Bayes' theorem. KNN is a straightforward instance-based learning algorithm, while Random Forest is an ensemble learning method that employs multiple decision trees to enhance accuracy and minimize overfitting. This study rigorously evaluates the performance of the classifiers using essential metrics such as accuracy, precision, recall, and F1-score. These metrics offer a comprehensive assessment of the models' effectiveness, accounting for both prediction accuracy and the balance between false positives and false negatives. The research aims to demonstrate the potential of bio-inspired algorithms, particularly Particle Swarm Optimization (PSO) and the Firefly Algorithm, in improving feature selection for autism spectrum disorder (ASD) classification. By enhancing feature selection, these algorithms contribute to developing more accurate and reliable diagnostic models, facilitating better ASD detection and intervention. The study's findings aim to shed light on the application of modern computer algorithms in medical diagnosis, potentially paving the way for more effective and efficient systems for classifying autism spectrum disorder (ASD). These technologies could enhance the diagnostic process by making it more objective, faster, and more accessible for individuals with ASD and their families.

**1.2 STATEMENT OF THE PROBLEM**

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition affecting millions globally. The disorder is characterized by challenges in social communication, repetitive behaviours, and restricted interests. The variability in ASD symptoms among individuals complicates diagnosis, as the manifestations can differ significantly. This unpredictability hinders the creation of effective diagnostic tools, often leading to delays in diagnosis and treatment. Early detection and intervention are crucial for improving the developmental outcomes and quality of life for individuals with autism. However, current diagnostic methods primarily rely on clinical observations and standardized questionnaires, which can be subjective and time-consuming. The growing need for more accurate and cost-effective diagnostic tools has driven researchers to explore advanced computational methods, including machine learning and bio-inspired optimization algorithms. Feature selection is a crucial step in developing computational models, as it involves identifying the most relevant features in a dataset to enhance the accuracy and efficiency of classification systems. The goal is to pinpoint variables that capture the complexity of autism spectrum disorder (ASD) while reducing data dimensionality, thereby minimizing overfitting and improving generalization. Despite the availability of various feature selection techniques, there is no consensus on the best approach for ASD classification, especially when handling large, multidimensional datasets.

This research explores the use of bio-inspired optimization techniques, namely Particle Swarm Optimization (PSO) and the Firefly Algorithm, to enhance feature selection for autism spectrum disorder (ASD) classification. These algorithms, which are inspired by natural phenomena, have proven effective in tackling complex optimization challenges across various fields. In this study, they are utilized to identify the most relevant features from the Toddler Autism dataset, thereby improving the performance of multiple classification models, including Artificial Neural Network Multi-Layer Perceptron (ANN MLP), Gaussian Naive Bayes (GaussianNB), K-Nearest Neighbors (KNN), and Random Forest. The primary objective of this research is to develop and evaluate an efficient feature selection method that enhances the accuracy and performance of ASD classification systems. Given that current ASD diagnostic methodologies often rely on subjective judgments, there is a significant need for more objective, data-driven approaches. This research seeks to lay the groundwork for building robust ASD classification models by exploring the use of Particle Swarm Optimization (PSO) and the Firefly Algorithm in feature selection. By improving the accuracy, efficiency, and accessibility of the diagnostic process, these models could facilitate early diagnosis and intervention for individuals with ASD.

**1.3 SYSTEM SPECIFICATIONS**

**CPU:** An Intel Core i7 or AMD Ryzen 7 processor (or equivalent) to expedite training and inference processes.

**GPU:** NVIDIA Tesla T4 for accelerating machine learning computations. Alternatively, consider AMD Radeon RX 6000 series GPUs.

**RAM:** At least 16 GB to manage large datasets and model training tasks efficiently.

**Storage:** A solid-state drive (SSD) with a minimum capacity of 500 GB for fast data access and model storage.

**Specifications:**

**Software Operating System:**

Consider adopting Windows 11 project due to its user-friendly interface and support for a wide range of machine learning tools and frameworks.

**Python Environment:**

Google Collab is a cloud-based Python environment that provides free access to GPU resources and facilitates collaborative coding.

**Libraries:**

NumPy, SciPy, Pandas, and Matplotlib are libraries used for data preparation, analysis, and visualization, while Scikit-learn is employed for generating assessment metrics.

**NumPy:**

Machine learning relies extensively on NumPy, a powerful Python library for numerical computations and array operations. NumPy is fundamental to many machines learning systems, significantly improving data handling and the performance of mathematical operations.

**Pandas:**

Pandas is a powerful tool for data preparation and manipulation, aiding in the organization and cleaning of datasets to ensure a smoother training process for neural networks.

**Matplotlib:**

Matplotlib is a powerful Python visualization library commonly used in machine learning for plotting training and validation metrics, loss curves, and model performance graphs, helping to better understand model behavior.

**CHAPTER 2**

**LITERATURE SURVEY**

Lidia V. Gabis et al. [1] uses tri-axial accelerometers to assess inter-limb coordination during jumping exercises, focusing on motor coordination deficits in children with autism spectrum disorder (ASD). Previous research has highlighted the challenges of evaluating coordination in children with ASD, often relying on time-consuming or subjective methods. This study aims to provide a more objective and efficient assessment using accelerometers by comparing typically developing (TD) children to those with ASD. The findings reveal that children with ASD exhibit greater lag variability (LV) in their limb movements, indicating less consistent coordination compared to TD children. The study suggests that actigraphy could be an effective tool for evaluating motor coordination deficiencies in children with ASD. Krishna Patra et al. [2] research study investigates the early detection of autism spectrum disorder (ASD) using an imbalanced toddler dataset from the UCI data collection. To prepare the data for analysis, numerical values are converted for categorical variables. Principal Component Analysis (PCA) is applied for standardization and dimensionality reduction. Various machine learning classification models, including Decision Tree, SVM, k-NN, DA, and RF, are used to classify ASD symptoms. The models are evaluated using both training/testing splits and 10-fold cross-validation. The results show that Decision Tree and SVM outperform the other models in classification accuracy. Comparing the ASD classification accuracy of this study with other advanced methods reveals a clear improvement.

Hala Shamsuddin et al. [3] proposed study utilizes federated learning to safeguard data privacy while exploring behavioral and visual traits linked to autism spectrum disorder (ASD). Previous research has demonstrated the significance of facial and behavioral characteristics in diagnosing ASD and highlighted the varying accuracy levels of machine learning models based on these factors. Federated learning, already proven effective in healthcare settings where data security is paramount, offers a privacy-preserving alternative to traditional machine learning methods. This innovative approach combines diverse data aspects with privacy-preserving techniques, contributing to the growing body of knowledge on secure and precise ASD prediction. Marta Robles et al. [4] builds on previous research into using machine learning and virtual reality (VR) for diagnosing autism. Earlier studies have shown that VR is effective for simulating social interactions and gathering behavioral data. Machine learning techniques such as logistic regression, support vector machines, and neural networks have been used to classify autism based on this data. Recent advancements have incorporated Long Short-Term Memory (LSTM) networks to better analyze time-series data, leading to improved classification accuracy. This study advances the field by combining VR simulations with LSTM networks to enhance both the speed and accuracy of autism diagnosis.

Jamia Millia Islamia et al. [5] explores various machine learning algorithms for predicting autism spectrum disorder (ASD) in individuals of all ages, including Naïve Bayes, Support Vector Machine (SVM), Logistic Regression, K-Nearest Neighbors (KNN), Neural Networks, and Convolutional Neural Networks (CNN). By evaluating specificity, sensitivity, and accuracy across three non-clinical ASD datasets, the study finds that CNN achieves the highest accuracy rates, consistently outperforming the other methods. The paper highlights the significance of thorough model evaluation and data preparation in enhancing the accuracy of ASD screening predictions. Maria Eleonora et al. [6] This systematic review focuses on the use of eye movement (EM) data and machine learning (ML) techniques to detect autism spectrum disorder (ASD) early, specifically through children's social visual attention (SVA). It summarizes findings from eleven studies that employed machine learning and eye tracking data to differentiate between children with ASD and typically developing peers. The review highlights how machine learning can provide an objective method for diagnosing ASD by analyzing eye movement responses to social cues. It outlines the inclusion criteria, search methods, and underscores the significant role of machine learning in improving early ASD detection. Additionally, it discusses future research directions and acknowledges the limitations of current studies.

José Jaime Esqueda-Elizondo et al. [7] This study employs the Epoc+ Brain-Computer Interface and Matlab for signal processing to explore how electroencephalogram (EEG) data can be used to measure attention in a 13-year-old with autism spectrum disorder (ASD). To classify attention levels from EEG data, the study evaluates the performance of several machine learning models, including Multi-Layer Perceptron Neural Network (MLP-NN), Naïve Bayes, SVM-RBF, and KNN. Among these, the MLP-NN model achieved the highest performance with an AUC of 0.9299 and outperformed the others in accuracy, Cohen's Kappa, and Matthews correlation coefficient. The aim of this project is to enhance educational environments for individuals with ASD by implementing improved teaching strategies and providing quantitative feedback. Ishaani Priyadarshini et al. [8] Recent research on autism spectrum disorder (ASD) diagnosis has highlighted several effective strategies for enhancing early identification and intervention. Ahmed et al. (2022) achieved high accuracy rates of up to 95.5% by employing eye-tracking technology alongside advanced machine learning models. Kohli et al. (2022) demonstrated an average accuracy of 81% to 84% by focusing on applied behavior analysis with clinical data. Liao et al. (2022) utilized machine learning to evaluate behavioral and physiological data, reaching an accuracy rate of 87.5%. Bala et al. (2022) achieved a notable accuracy of 96.82% using SVM and SHAP algorithms. The proposed research aims to advance these methods further by integrating CNN, LSTM, and PSO to enhance ASD identification outcomes.

Maraheb Alsuliman et al. [9] Early detection of autism significantly improves patient outcomes, yet traditional diagnostic methods remain slow and imprecise. Recent research highlights the potential of machine learning (ML) to enhance diagnostic accuracy by integrating gene expression (GE) with personal and behavioral characteristics (PBC). Addressing the challenges of high-dimensional data can be achieved through advanced feature selection algorithms, particularly those inspired by biological processes. Notable examples include Grey Wolf Optimization (GWO), Flower Pollination Algorithm (FPA), Bat Algorithms (BA), and Artificial Bee Colony (ABC). Comparative studies indicate that Support Vector Machine (SVM) models, particularly when optimized by GWO and FPA, outperform other classifiers, underscoring the effectiveness of these innovative methods for autism classification. KmBhavna et al. [10] article explores Theories of Mind (ToM) and Autism Spectrum Disorder (ASD), emphasizing how features derived from connectomes and brain correlates can help differentiate ASD from typically developing individuals. It investigates variations in the functional connectivity of ToM brain regions through big data and artificial intelligence (AI). Previous research highlights the importance of neuroimaging and machine learning techniques in accurately diagnosing and predicting ASD symptom severity. The research advances existing methods by employing Explainable AI models to enhance both classification accuracy and understanding of ToM-related brain connections.

S. Saravana Kumar et al. [11] report suggests integrating the Enhanced Whale Optimization Algorithm (EWOA) with an Ensemble Modified Auto Encoder Convolutional Neural Networks (EM-AECNN)-Artificial Neural Network (ANN) system to enhance autism spectrum disorder (ASD) categorization. This approach addresses limitations in feature selection and classification accuracy by employing sophisticated classification algorithms, data preprocessing, and feature selection techniques. The results indicate that this method significantly improves accuracy, precision, memory efficiency, and the F-measure compared to current methods. By utilizing SMOTE for data balancing and EWOA for optimal feature selection, the technique achieves a classification accuracy of 98.3%. Vishal Midya et al. [12] explores the connection between environmental chemical exposure and autism spectrum disorder (ASD) by combining Signed Iterative Random Forest (SiRF) with Weighted Quantile Sum (WQS) regression. By examining interactions among 62 chemicals found in urine samples, the study highlights how certain chemical combinations, such as diethyl-phosphate and cadmium, are linked to increased ASD risk. This research enhances our understanding of how toxicological interactions affect neurodevelopment and underscores the importance of integrating machine learning and statistical methods in environmental health studies. Yugandhar Bokka et al. [13] Research into the connection between lipid levels and autism spectrum disorder (ASD) has gained momentum, as maternal and infant lipid profiles may play a role in early ASD prediction. Several studies suggest a potential link between high cholesterol and triglyceride levels and autism, but the findings are mixed. Current research indicates that while maternal dyslipidemia might affect the baby's development, its exact impact on ASD remains unclear. However, recent advances in machine learning techniques, such as XGBoost, have led to more accurate predictions of ASD risk using lipid profiles. By integrating lipid data from both mothers and infants, this study adds valuable insights to the understanding of ASD prognosis.

Navita et al. [14] Training supervised learning models for melody extraction from audio can be challenging when comprehensive musical annotations are lacking. Traditional approaches rely on manually crafted features, but recent research has shifted towards using MIDI files, which provide valuable melodic information. Melody channels are often categorized based on features such as entropy and note length, employing various machine learning algorithms like Random Forests, Support Vector Machines, and Neural Networks. Studies show that while advanced techniques, particularly Artificial Neural Networks, tend to outperform traditional methods, the latter still produce reasonably good classification results. Dhuha Dheyaa et al. [15] explores the use of machine learning (ML) models to improve the reliability of autism spectrum disorder (ASD) predictions, aiming to overcome the challenges associated with diagnosing ASD. By leveraging datasets from Kaggle and UCI, the study evaluates several supervised machines learning techniques, including Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbor (K-NN), Naïve Bayes (NB), Logistic Regression (LR), and Random Forest (RF). The results indicate that DT, LR, and RF models achieve near-perfect accuracy and perform exceptionally well. The research demonstrates the robustness of these methodologies for ASD prediction by comparing their performance with previous studies and highlighting their effectiveness in handling partial or imbalanced data.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

Diagnosing Autism Spectrum Disorder (ASD) traditionally relies on clinical evaluations and standardized diagnostic tools. Healthcare professionals historically conducted thorough assessments that involved observing behavior, reviewing developmental history, and conducting structured interviews with parents and caregivers. Commonly used standardized methods include the Autism Diagnostic Observation Schedule (ADOS) and the Autism Diagnostic Interview-Revised (ADI-R). While these tools offer a structured approach to studying specific behavioral traits, they can be subjective and time-consuming. This reliance on clinical expertise and observation often leads to variability and inconsistency in diagnoses, resulting in delays in diagnosis and treatment. In recent years, there has been increasing interest in using machine learning techniques to improve the diagnosis of autism spectrum disorder (ASD). Algorithms such as Support Vector Machines (SVM), Decision Trees, and Neural Networks are being employed to analyze ASD data, allowing for the processing of large datasets and the discovery of patterns that might be missed by traditional methods. However, the effectiveness of these models is heavily dependent on the quality and relevance of the features used in training. Therefore, feature selection plays a vital role in enhancing model performance.

Current machine learning algorithms for autism spectrum disorder (ASD) classification often face challenges with feature selection and dimensionality reduction. Many datasets used in these models contain many attributes, some of which may be unnecessary or redundant. Identifying the most relevant features is crucial for ensuring that the model learns and generalizes effectively from the data. Existing feature selection methods in ASD classification typically rely on statistical or heuristic approaches, which may not adequately capture the complex relationships within the data. These methods can sometimes produce unsatisfactory results by either omitting important features or including too many irrelevant ones, leading to overfitting or poor model generalization. While traditional and machine learning-based diagnostic tools have seen significant advancement, there is increasing interest in leveraging bio-inspired optimization techniques for feature selection. Algorithms such as Particle Swarm Optimization (PSO) and the Firefly Algorithm have been studied across various domains due to their effectiveness in exploring large solution spaces and identifying optimal feature sets. These algorithms mimic natural processes to navigate and utilize the feature space, potentially surpassing conventional feature selection methods. However, the use of bio-inspired optimization algorithms in autism spectrum disorder (ASD) classification remains relatively unexplored. Previous systems have not fully exploited these sophisticated strategies to enhance feature selection and classification performance.

Existing diagnostic techniques for autism spectrum disorder (ASD) face significant challenges, including subjectivity in traditional clinical evaluations and difficulties in feature selection for machine learning models. Traditional methods are often time-consuming and subjective, while machine learning algorithms struggle with complex feature selection processes. Bio-inspired optimization algorithms offer a promising solution to these problems by providing advanced methods for identifying critical features and enhancing classification accuracy. This study aims to address these issues by integrating Particle Swarm Optimization (PSO) with the Firefly Algorithm to improve feature selection and, consequently, the performance of ASD classification systems.

**3.2 LIMITATIONS OF THE EXISTING SYSTEM**

**Subjectivity in Clinical Assessments**

A key issue with the current system for diagnosing autism spectrum disorder (ASD) is its inherent subjectivity. Traditional diagnostic methods largely depend on physicians' observations and interpretations, which can vary between practitioners. This variability can lead to diagnostic errors and delays, as different physicians may have different thresholds for diagnosis based on their personal judgments.

**Time-Consuming and Resource-Intensive**

The conventional method for diagnosing autism spectrum disorder (ASD) is both time-consuming and resource intensive. It typically involves multiple sessions with clinicians, a detailed review of developmental history, and extensive interviews with parents or caregivers. This process can span several weeks or even months, leading to delays in diagnosis and treatment. Additionally, it demands considerable resources, including trained professionals and specialized diagnostic equipment.

**Limitations of Standardized Diagnostic Tools**

Standardized procedures like the Autism Diagnostic Observation Schedule (ADOS) and the Autism Diagnostic Interview-Revised (ADI-R) are widely used, but they have limitations. These assessments often concentrate on specific behavioral criteria, potentially overlooking the full spectrum of ASD symptoms. Additionally, they can be affected by cultural and environmental factors, which may affect their accuracy and applicability across different communities.

**Challenges with Feature Selection in Machine Learning**

Accurate feature selection is one of the most challenging aspects of using machine learning for autism spectrum disorder (ASD) classification. High-dimensional datasets, which contain numerous attributes, can include many irrelevant or redundant features. Traditional feature selection methods, primarily based on statistical or heuristic approaches, may not fully capture the complexity of ASD. Consequently, these models might include too many insignificant features or miss important ones, leading to suboptimal performance and overfitting.

**Lack of Integration of Advanced Optimization Techniques**

While machine learning models have been employed to identify autism spectrum disorder (ASD), there's a notable gap in integrating advanced bio-inspired optimization techniques into these systems. Algorithms such as Particle Swarm Optimization (PSO) and the Firefly Algorithm, which have shown significant promise in other fields, remain underutilized in ASD research. Existing systems predominantly rely on traditional feature selection methods, which may not be as effective as contemporary optimization algorithms in enhancing feature selection and model performance.

**Generalization and Overfitting Issues**

Machine learning models, if not properly calibrated, can face issues of generalization and overfitting. When trained on high-dimensional datasets without effective feature selection, these models might perform well during training but fail to generalize effectively to new, unseen data. This limitation reduces the reliability of ASD classification algorithms and their ability to provide accurate diagnoses in real-world scenarios.

Current methods for diagnosing and classifying autism spectrum disorder (ASD) face several challenges, including the subjectivity inherent in clinical assessments, constraints related to time and resources, and issues with feature selection and model generalization. These problems are further compounded by the lack of incorporation of advanced bio-inspired optimization techniques. To overcome these limitations and develop more accurate, efficient, and reliable ASD classification models, it is essential to explore and integrate novel feature selection algorithms such as Particle Swarm Optimization (PSO) and the Firefly Algorithm.

**3.3 PROPOSED SYSTEM**

The proposed method seeks to address the limitations in current ASD diagnostics by incorporating advanced bio-inspired optimization algorithms, such as Particle Swarm Optimization (PSO) and the Firefly Algorithm, into the feature selection process for ASD classification. This innovative approach aims to enhance the accuracy and efficiency of machine learning models for diagnosing autism spectrum disorder, leading to improved early detection and intervention outcomes.

The recommended system utilizes PSO and the Firefly Algorithm for feature selection. These bio-inspired optimization techniques leverage natural processes to efficiently navigate complex solution spaces. The PSO algorithm mimics the social behavior of bird flocking or fish schooling to find optimal responses, while the Firefly Algorithm draws inspiration from the flashing patterns of fireflies, which attract mates and help in locating optimal solutions. By applying these methods, the system can identify the most significant features in the Toddler Autism dataset, thereby reducing dimensionality and enhancing data quality for classification algorithms.

Feature selection plays a crucial role in the proposed system. Unlike previous algorithms constrained by statistical or heuristic criteria, PSO and the Firefly Algorithm provide a more robust approach for selecting features. These algorithms assess numerous feature subsets and iteratively refine their results based on performance metrics. This approach ensures that only the most relevant attributes are retained, thereby improving model accuracy and reducing the risk of overfitting. The selected features are anticipated to more effectively capture the complexities of ASD, leading to improved diagnostic performance.

Once identified, the key features are utilized to train and assess various classification models, including ANN MLP, Gaussian Naive Bayes (GaussianNB), K-Nearest Neighbors (KNN), and Random Forest. The proposed system employs multiple classification algorithms to evaluate their effectiveness with the optimized feature set. By using bio-inspired feature selection, the system aims to enhance model performance, leading to more accurate and reliable diagnoses of autism spectrum disorder (ASD). The expanded feature set is anticipated to significantly impact the ANN MLP and Random Forest classifiers, potentially improving accuracy, precision, recall, and F1 scores.

The proposed technique is designed to be both efficient and practical. It enhances model performance and reduces computational complexity by optimizing feature selection through unique procedures. This efficiency leads to shorter training times and makes the approach more applicable in real-world scenarios. The system aims to bridge the gap between theoretical research and practical use by offering a tool that can be seamlessly integrated into existing diagnostic processes. This practical approach ensures that physicians and researchers can benefit from improved feature selection methods, increasing their adoption in routine diagnostic practices.

The proposed method is expected to improve diagnostic accuracy, reduce diagnostic time, and enhance the reliability of ASD classification models. By employing PSO and the Firefly Algorithm for feature selection, the approach seeks to overcome the limitations of previous methods, offering a more objective, data-driven approach to diagnosing autism spectrum disorder. The successful application of this method aims to advance early detection and intervention strategies, ultimately benefiting individuals with ASD and their families.

In conclusion, this approach significantly advances ASD classification by leveraging bio-inspired optimization strategies to refine feature selection. It addresses existing limitations, enhances model performance, and presents a viable solution for more accurate and efficient ASD diagnosis.

**3.4 ADVANTAGES OF THE PROPOSED SYSTEM**

**Enhanced Accuracy of ASD Diagnosis**

The use of bio-inspired optimization techniques, such as Particle Swarm Optimization (PSO) and the Firefly Algorithm, significantly enhances the accuracy of autism spectrum disorder (ASD) classification models. By minimizing dataset noise and redundancy, these methods retain only the most crucial information. This leads to more precise and dependable diagnostic predictions, thereby improving the overall accuracy of the classification models.

**Improved Efficiency and Reduced Computational Complexity**

The proposed method addresses the challenge of high-dimensional data by utilizing advanced feature selection techniques. By applying PSO and the Firefly Algorithm, the feature sets are reduced, which decreases the computational cost associated with training and evaluating classification models. This reduction in dimensionality not only accelerates processing but also minimizes the computer resources required, enhancing the system's efficiency and practicality for real-world applications.

**Effective Handling of Complex and High-Dimensional Data**

Bio-inspired algorithms are highly effective for handling large and complex datasets, such as the Toddler Autism dataset. These algorithms are adept at navigating extensive solution spaces and pinpointing the most crucial features, which traditional methods might miss. Consequently, this approach can more effectively manage the complexities of ASD data, leading to enhanced model performance and a deeper understanding of ASD characteristics.

**Versatility Across Different Classification Models**

The proposed approach is designed to be compatible with various classification models, such as ANN MLP, Gaussian Naive Bayes (GaussianNB), K-Nearest Neighbors (KNN), and Random Forest. This flexibility indicates that the advantages of optimal feature selection extend beyond any single model, allowing for its application across different methodologies. This approach facilitates a comprehensive assessment of model performance and provides the flexibility to choose the most effective classifier for diagnosing autism spectrum disorder (ASD).

**Reduction in Overfitting and Improved Generalization**

By concentrating on the most significant features, the proposed technique prevents overfitting, which occurs when a model excels on training data but fails to perform well on unseen data. A more streamlined feature set allows for the development of models that generalize better to new, real-world data, resulting in more accurate and robust ASD classification. This enhanced generalizability ensures that diagnostic models are both precise and valuable in clinical settings.

**Faster Training and Diagnosis**

Reducing the number of features in classification models leads to shorter training times and decreased processing complexity. This increased efficiency results in faster diagnostic conclusions, which is crucial in clinical settings where timely intervention is essential. Consequently, the proposed method enhances the speed of diagnosis, facilitating quicker and more effective intervention for ASD patients.

**Objective and Data-Driven Approach**

The proposed method utilizes advanced optimization algorithms instead of relying on subjective evaluations, offering a more objective and data-driven approach to diagnosing ASD. This objectivity minimizes the volatility and unpredictability inherent in previous diagnostic methods, leading to more accurate and consistent results.

The proposed technique provides several key benefits, such as enhanced diagnostic accuracy, greater efficiency, and the ability to manage complex data. It also offers flexibility in classification models, minimizes overfitting, speeds up training and diagnosis, and presents a practical, scalable solution. These advantages contribute to a more reliable, efficient, and objective method for diagnosing autism spectrum disorder, advancing scientific knowledge and improving outcomes for individuals with ASD.

**CHAPTER 4**

**DATA COLLECTION AND PREPARATION**

**4.1 DATA SOURCES**

Autistic Spectrum Disorder (ASD) is a neurodevelopmental condition that incurs significant healthcare expenses, which could be substantially reduced with early detection. Receiving an ASD diagnosis can often take years, and therapy can be expensive. The economic impact of autism, combined with the increasing prevalence of ASD worldwide, highlights the urgent need for affordable and efficient screening methods. Consequently, efforts are underway to develop a time-efficient and accessible ASD test to assist healthcare professionals and individuals in deciding whether to pursue a formal clinical diagnosis. To address the rising global incidence of ASD, it is essential to gather comprehensive behavioral data. The limited availability of comprehensive datasets poses a significant challenge in enhancing the efficiency, sensitivity, specificity, and predictive accuracy of autism spectrum disorder (ASD) screening methods. Most existing autism datasets are primarily genetic, leaving a gap in data for clinical or screening purposes. To address this, propose a novel dataset specifically designed for autism screening in toddlers. This dataset includes ten behavioral traits (Q-Chat-10) along with other relevant individual characteristics. These features have demonstrated their usefulness in differentiating ASD patients from control groups in behavioral science, and they could be valuable for advancing future research in autism recognition and case categorization.

In the Q-Chat-10 assessment:

Questions 1 to 9 (A1-A9) are scored as "1" regardless of whether the response is "Sometimes," "Rarely," or "Never."

Question 10 (A10) is scored as "1" if the response is "Always," "Usually," or "Sometimes."

To determine if ASD symptoms may be present, sum the points from all 10 questions. If the total score is higher than 3, ASD symptoms might be present; otherwise, no ASD features are identified. The table compares various dataset factors with the characteristics of the Q-Chat-10-Toddler assessment. Each variable reflects a specific aspect of a child's behaviour or communication skills, with the mappings as follows:

Table 1: Correspondence Between Dataset Variables and Q-Chat-10 Toddler Features

|  |  |
| --- | --- |
| **Variable in Dataset** | **Corresponding Q-chat-10-Toddler Features** |
| A1 | Does your child look at you when you call his/her name? |
| A2 | How easy is it for you to get eye contact with your child? |
| A3 | Does your child point to indicate that s/he wants something? (e.g. a toy that is  out of reach) |
| A4 | Does your child point to share interest with you? (e.g. poin9ng at an  interes9ng sight) |
| A5 | Does your child pretend? (e.g. care for dolls, talk on a toy phone) |
| A6 | Does your child follow where you’re looking? |
| A7 | If you or someone else in the family is visibly upset, does your child show signs  of wan9ng to comfort them? (e.g. stroking hair, hugging them) |
| A8 | Would you describe your child’s first words as: |
| A9 | Does your child use simple gestures? (e.g. wave goodbye) |
| A10 | Does your child stare at nothing with no apparent purpose? |

The table illustrates how various characteristics of the dataset relate to the qualities assessed by the Q-Chat-10 toddler evaluation. Each variable (A1-A10) corresponds to specific behavioral questions intended to explore different aspects of a child's social and communicative development. For instance, A1 evaluates whether the child looks at the caregiver when called, while A10 examines if the child stares at nothing without any clear reason. This mapping helps in analyzing and scoring the child's responses to identify potential signs of ASD.

**4.2 DATA CLEANING AND PREPROCESSING**

Data cleaning is a crucial phase in preparing the Toddler Autism dataset for accurate analysis and modeling. This method involves identifying and correcting defects, errors, and missing data in the dataset to ensure its accuracy and suitability for machine learning. Missing values can greatly impact the performance of machine learning models. In the Toddler Autism dataset, these missing values may arise from incomplete entries or issues during data collection. To address this, a variety of methods can be employed:

**Imputation:** Imputation involves replacing missing values with statistical measures such as the mean, median, or mode of a feature. For categorical features, the mode (the most frequently occurring value) is often used, while for numerical features, either the mean or median may be employed.

**Deletion:** Deletion is a strategy where entries with missing values are removed from the dataset. This approach is suitable when the number of missing values is minimal and does not significantly impact the overall dataset.

**Prediction Models:** Prediction models utilize machine learning algorithms to estimate and fill in missing values based on the available data, thereby improving the accuracy of imputation.

Data inconsistencies can arise from input errors or changes in data format. Common issues include:

**Data Format Standardization:** Ensure that all data entries adhere to a uniform format. For instance, dates should be consistently formatted, and numerical values should use the same units.

**Error Correction:** Detect and rectify incorrect data entries, such as outliers or values outside expected ranges. This might involve cross-referencing with existing data standards or consulting subject matter experts.

Data preparation involves transforming and scaling cleaned data to make it suitable for analysis and modeling. This approach guarantees that the data is well-suited for machine learning techniques, resulting in improved model performance.

Feature engineering involves enhancing the dataset by creating new features or improving existing ones to boost its quality and usability. This can include:

**Feature Extraction**: involves creating new characteristics from existing data to reveal significant patterns or relationships. For example, combining various behavioural signals into a single composite score.

**Dimensionality Reduction**: Applying methods like Principal Component Analysis (PCA) to decrease the number of features while preserving most of the dataset's variance. This streamlines the model and can enhance its performance.

Data cleaning and preprocessing are essential for preparing the Toddler Autism dataset for machine learning. This involves addressing missing values, correcting inconsistencies, and applying normalization, standardization, and feature engineering to ensure the data is structured consistently and is suitable for model training and evaluation. Proper preprocessing enhances data quality and model performance, while also ensuring the accuracy and sustainability of the ASD diagnostic system.

**CHAPTER 6**

**EXPLORATORY DATA ANALYSIS**

**5.1 UNIVARIATE AND BIVARIATE ANALYSIS**

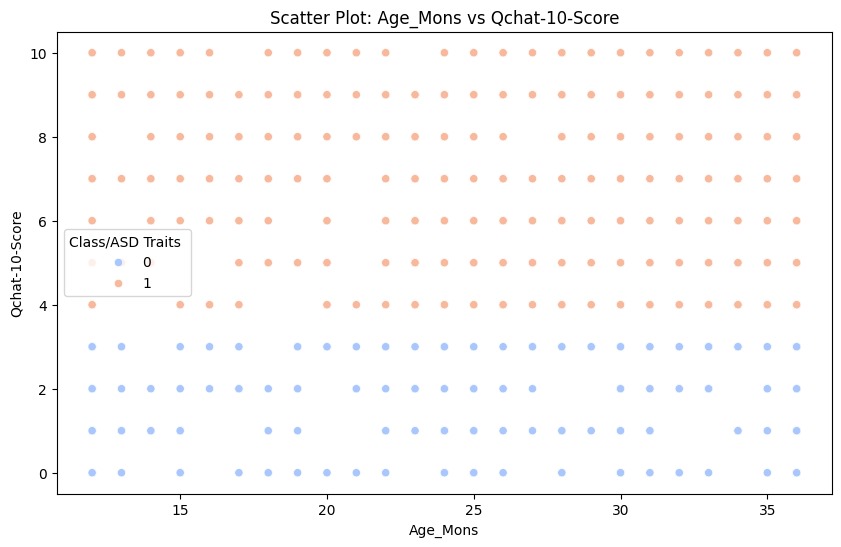


Fig.1 Scatterplot Comparison between "Age in Months” and "Qchat-10 Score"

Fig.1 illustrates the relationship between "Age in Months" (Age\_Mons) and "Qchat-10 Score," differentiating between various ASD (autism spectrum disorder) characteristics. The x-axis displays the ages of individuals, ranging from 14 to 35 months, while the y-axis represents the Qchat-10 Score, a scale from 0 to 10 that likely measures ASD symptoms. Data points are color-coded according to the type of ASD characteristic: blue for class 0 (no ASD traits) and orange for class 1 (presence of ASD traits). The graphic indicates that individuals in class 0 had notably lower Qchat-10 scores, ranging from 0 to 4, while those in class 1 had higher scores, typically between 6 and 10. There is no clear trend related to age, as the distribution of scores remains stable across different ages. In this sample, the Qchat-10 Score seems to be a more reliable predictor of ASD symptoms compared to age.

A pie chart with numbers and symbols

Description automatically generated

Fig.2 Representing distribution of individuals

Fig.2 illustrates the distribution of individuals in a dataset based on the presence of ASD (autism spectrum disorder) features. The chart is divided into two sections: the blue segment, labeled "Yes," represents individuals with ASD traits and comprises 69.1% of the sample. The orange segment, labeled "No," represents those without ASD traits and makes up 30.9% of the sample. This chart indicates that most of the individuals in the sample exhibit ASD traits, with more than two-thirds classified under "Yes." It provides an overview of the prevalence of ASD features within the studied population.

A bar chart with blue bars

Description automatically generated

Fig.3 Average age in months of individuals Category

Fig.3 displays the average age in months (Mean Age\_Mons) of individuals categorized based on the presence of ASD traits. The x-axis distinguishes two groups: "No" for those without ASD symptoms and "Yes" for those with ASD traits. The y-axis represents the mean age in months. According to the graph, the average age of individuals without ASD features is slightly above 25 months, while the average age of those with ASD traits is similar. The small difference in mean age between the two groups suggests that age remains relatively consistent regardless of the presence of ASD symptoms, indicating no significant age difference between those with and without ASD traits.

**CHAPTER 6**

**METHODOLOGY**

**6.1 DATA MODELS**

**A diagram of a process

Description automatically generated**

### Fig 4: Workflow Of the Data Model

The flow diagram Fig.4 outlines the sequential process for developing and deploying a machine learning system designed for autism spectrum disorder (ASD) classification, utilizing bio-inspired optimization techniques. The following sections provide an in-depth discussion of the diagram's components:

**Importing Libraries and Dataset:**

The initial step involves importing the required libraries and datasets. TensorFlow, Scikit-learn, NumPy, and Pandas are utilized to organize data, train models, and evaluate the results. The Toddler Autism dataset, which includes crucial features for diagnosing ASD, is then loaded into the environment for further analysis.

**Label Encoding:**

Label encoding is a technique used to convert categorical data into numerical values that machine learning algorithms can interpret. For instance, category names such as "Yes" and "No" are transformed into binary values. This process ensures that the data is appropriately prepared for model training.

**Bio-Inspired Optimizations:**

At this stage, bio-inspired optimization methods are used to extract the most important features from the dataset. Two different algorithms are employed:

**Particle Swarm Optimization (PSO):** This algorithm identifies the optimal set of features by simulating the social behaviour of particles in a swarm. Here are the key formulae used in PSO:

1. **Velocity Update Formula:**

Where:

is the velocity of particle i at time .

The inertia weight (w) affects the influence of the previous velocity.

is the cognitive coefficient that attracts the particle towards its personal best position.

is the social coefficient that attracts the particle towards the global best position.

and are random numbers range from 0 to 1.

denotes the particle’s best-known position i.

represents the swarm's most prominent global position.

represents the current location of particle i at time t.

1. **Position Update Formula:**

Where:

new position of particle i at time

current position of particle i at time t.

updated velocity of particle i at time.

1. **Inertia Weight:**

Where:

maximum inertia weight

inertia weight

t current iteration

T total number of iterations

**Firefly Algorithm:** This algorithm utilizes the attractiveness and movement of fireflies to discover optimal solutions, thereby improving the feature selection process. Here are the key formulae used:

1. **Attractiveness Update Formula:**

The attractiveness of a firefly i to firefly j is given by:

Where:

attractiveness of firefly i to firefly j.

initial attractiveness.

light absorption coefficient.

represents the distance between fireflies i and j.

1. **Distance Calculation Formula:**

The distance ​ between two fireflies i and j is calculated using:

Where:

position vectors of fireflies i and j.

|| . ||denotes the Euclidean distance.

1. **Movement Formula:**

The movement of a firefly i towards a more attractive firefly j is given by:

(

Where:

position of firefly i.

position of more attractive firefly j.

randomization parameter is a uniformly distributed random number within the range of 0 to 1.

The rand () function generates a random number within this range.

1. **Update Light Intensity:**

The light intensity of a firefly i is updated based on the objective function value:

Where,

objective function value at position

These techniques help identify the features that contribute most significantly to the classification process, enhancing the model's accuracy and efficiency.

**Grid Search for Selecting Best Parameters:**

Grid Search is a technique used for optimizing hyperparameters in machine learning models. It thoroughly explores a predefined parameter space to identify the best parameter combinations. The final models are then trained using the selected optimal parameters.

**Model Training and Evaluation:**

Multiple classification models have been trained using available features and optimized hyperparameters. The models include:

**Artificial Neural Network Multi-Layer Perceptron (ANN MLP):** A machine learning model capable of recognizing complex patterns in data.

**Random Forest:** An ensemble learning technique that combines the outputs of multiple decision trees.

**K-Nearest Neighbours (KNN):** A straightforward method for instance-based learning.

**Gaussian Naive Bayes (GaussianNB):** A probabilistic classifier that assumes a Gaussian distribution for features.

Each model's performance in detecting autism is evaluated using metrics such as accuracy, precision, recall, and F1-score.

**Deployment:**

Finally, the trained models are deployed in a real-world environment, such as a web-based platform built with Flask. This platform allows physicians and researchers to input new data and receive predictions. The system is designed to be user-friendly, offering straightforward interactions and quick access to diagnostic results.

The flowchart illustrates the comprehensive process of developing an ASD categorization system. It begins with data preparation and moves through feature selection, hyperparameter optimization, model training, assessment, and deployment. By incorporating bio-inspired optimization techniques, the model's performance is enhanced, resulting in more accurate and efficient diagnoses. This meticulous approach ensures that the final product is robust, reliable, and valuable for clinical decision-making in autism diagnosis.

**6.2 MODEL SELECTION**

This study examines four widely used classification models: the Artificial Neural Network Multi-Layer Perceptron (ANN MLP), Gaussian Naive Bayes (GaussianNB), K-Nearest Neighbors (KNN), and Random Forest. The effectiveness of these models in diagnosing autism spectrum disorder (ASD) was evaluated following feature selection enhancements achieved through Particle Swarm Optimization (PSO) and the Firefly Algorithm.

**Artificial Neural Network Multi-Layer Perceptron (ANN MLP):**

When combined with advanced feature selection methods like Particle Swarm Optimization (PSO) and the Firefly Algorithm, the Artificial Neural Network Multi-Layer Perceptron (ANN MLP) demonstrated impressive performance. Both bio-inspired algorithms contributed to achieving perfect scores on key performance metrics, including precision, recall, F1-score, and accuracy, each receiving a flawless value of 1.00. This indicates that the ANN MLP model, when paired with these optimization techniques, excelled in delivering highly accurate and reliable classification results for autism spectrum disorder. The time taken to complete classification tests highlighted the effectiveness of the ANN MLP model. With PSO for feature selection, the model processed and categorized data in 8 minutes and 39 seconds, which was slightly longer than the Firefly Algorithm's time of 8 minutes and 7 seconds. Despite these small differences in time, the overall performance remains impressive, showing the model's capability to manage complex categorization tasks effectively. The rapid classification times of the ANN MLP model underscore its usefulness when paired with modern optimization techniques, making it suitable for real-world applications that demand both accuracy and speed.

In conclusion, integrating PSO and the Firefly Algorithm with the ANN MLP model led to a notable improvement in classification accuracy. These optimization methods not only achieved superior classification results but also ensured the model ran efficiently within a reasonable timeframe. This synergy underscores the value of advanced feature selection in boosting the accuracy and reliability of neural network-based classification systems, particularly for complex applications like diagnosing autism spectrum disorder.

**Random Forest:**

When employing feature selection techniques like Particle Swarm Optimization (PSO) and the Firefly Algorithm to enhance the Random Forest model's ability to detect autism spectrum disorder (ASD), the results were highly promising. Using PSO, the Random Forest model achieved a notable accuracy of 0.99, demonstrating exceptional precision in identifying ASD patients. The model's perfect accuracy score of 1.00 indicates its flawless performance in minimizing false positives and ensuring all positive classifications are correct. Additionally, a recall of 0.96 shows the model's effectiveness in correctly identifying most true ASD cases, while an F1-score of 0.98 reflects a well-balanced performance in terms of both accuracy and recall.

When applying these feature selection methods to the Random Forest classifier, there was a notable improvement in temporal efficiency. The processing time using the Particle Swarm Optimization (PSO) was 4 minutes and 17 seconds, whereas the Firefly Algorithm took slightly longer at 5 minutes and 50 seconds. Despite the Firefly Algorithm's increased runtime, the Random Forest model proves to be exceptionally efficient, especially given the complexity of the feature selection process and the impressive results achieved. The model's rapid processing speeds, combined with its high classification accuracy and precision, make it highly suitable for autism spectrum disorder (ASD) classification tasks, particularly when optimized feature selection techniques are utilized to enhance prediction capabilities.

**Gaussian Naive Bayes (GaussianNB):**

The performance of the Gaussian Naive Bayes (GaussianNB) model improved significantly after incorporating optimization techniques like Particle Swarm Optimization (PSO) and the Firefly Algorithm. Initially, the model's performance metrics were relatively poor, with an accuracy of 0.43 and a recall of 0.57. These results indicated that the model struggled to effectively classify a substantial portion of the data, particularly in identifying all positive cases of Autistic Spectrum Disorder (ASD). The classification decision relies on Bayes' theorem, if continuous features follow a normal distribution. The formula used to compute the posterior probabilities is:

Where,

​ is the probability density function of the feature ​ given class

is the marginal probability of the feature vector x.

The incorporation of two bio-inspired feature selection algorithms, PSO and the Firefly Algorithm, led to a significant enhancement in model performance. With the revised feature selection approach, the GaussianNB model's accuracy and recall improved to 1.00 and 0.99, respectively. This highlights how crucial accurate feature selection is for enhancing the performance of machine learning systems. By meticulously selecting and prioritizing key features, these optimization strategies enabled the GaussianNB model to attain near-perfect classification results.

**K-Nearest Neighbors (KNN):**

K-Nearest Neighbours (KNN) was assessed as a classification model for autism spectrum disorder (ASD) using various feature selection methods. Although KNN is a widely used and straightforward algorithm, it did not surpass other models tested, such as Artificial Neural Network (ANN), MLP, Random Forest, and Gaussian Naive Bayes.

KNN classifies data points based on the majority class of their nearest neighbours and performs best with a well-organized feature set. However, in this study, KNN did not show significant improvements in classification accuracy or efficiency when paired with the feature selection techniques Ant Colony Optimization (ACO) and Artificial Bee Colony (ABC) for ASD.

**6.3 MODEL BUILDING**

Creating effective models for autism spectrum disorder (ASD) classification involves multiple steps, including data preparation, model selection, feature optimization, and evaluation. Here’s a comprehensive overview of the process for building these models.

**Data Preparation:**

The initial step involves preparing the data for modeling. This includes removing missing values, updating or standardizing the data to ensure all features are uniform, and splitting the dataset into training and testing sets. Proper preprocessing is crucial for delivering more accurate and consistent results from the model.

**Feature Selection:**

This study aimed to improve feature selection by employing bio-inspired optimization strategies. The dataset's key features were identified using Particle Swarm Optimization (PSO) and the Firefly Algorithm. These algorithms work by iteratively searching for the best combination of attributes to boost classification accuracy. PSO emulates the social behavior of birds or fish to enhance the feature set by evaluating potential solutions. PSO mimics the social behavior of birds or fish to explore and expand the feature set by assessing various alternatives. In contrast, the Firefly Algorithm, inspired by the flashing patterns of fireflies, identifies optimal features based on their 'light intensity.' By integrating these approaches, the study intended to refine the feature set and enhance model performance.

**Model Training and Evaluation:**

The refined feature set was used to develop and train four classification models: ANN MLP, GaussianNB, KNN, and Random Forest. Each model was trained on the training subset of the dataset using the specified parameters. Hyperparameters for each model were adjusted to achieve optimal performance, including setting the number of hidden layers for the ANN MLP, the number of neighbours for KNN, and the number of trees for Random Forest.

The performance of each model was evaluated on the testing subset. Key performance indicators such as accuracy, precision, recall, and F1 scores were used to assess the effectiveness of each classifier. Additionally, the models were timed to gauge their computational efficiency.

**Model Comparison and Selection:**

The evaluation data revealed that the ANN MLP and Random Forest models performed the best overall. They excelled in accuracy, precision, recall, and efficiency, particularly when using feature sets optimized by the PSO and Firefly Algorithms. These models not only achieved high scores across all metrics but also demonstrated greater computational efficiency compared to the others.

**Model Deployment:**

The selected models, namely ANN MLP and Random Forest, were merged to create an effective classification system. This system was then integrated into a user-friendly application that allows for real-time ASD categorization, serving as a crucial tool for early identification and intervention.

The model-building process included careful data preparation, advanced feature selection with bio-inspired algorithms, rigorous training and testing of various classifiers, and the deployment of the most effective models. This comprehensive approach led to a highly accurate and efficient classification system that significantly aids in autism diagnosis and treatment

**6.4 RESULTS**

Table 2: Swarm and Firefly Results

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithms | PRECISION | | RECALL | | F1-SCORE | | ACCURACY | | TIME  COMPLEXITY | |
|  | Swarm | Firefly | Swarm | Firefly | Swarm | Firefly | Swarm | Firefly | Swarm | Firefly |
| MLP Classifier | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 8m 39s | 8m 7s |
| Random Forest | 1.00 | 1.00 | 0.96 | 0.96 | 0.98 | 0.98 | 0.99 | 0.99 | 4m 17s | 5m 50s |
| KNN | 0.92 | 0.97 | 0.99 | 0.96 | 0.95 | 0.96 | 0.97 | 0.98 | 17s | 8s |
| GaussianNB | 0.97 | 1.00 | 0.93 | 0.96 | 0.95 | 0.98 | 0.97 | 0.99 | 2s | 2s |

The table 2 evaluates the performance and time complexity of Swarm and Firefly optimization algorithms when used with various machine learning classifiers, including MLP Classifier, Random Forest, K-Nearest Neighbours (KNN), and Gaussian Naive Bayes (GaussianNB). Both Swarm and Firefly algorithms achieve perfect precision, recall, F1-score, and accuracy with the MLP Classifier, indicating that these two optimization methods yield equivalent results in classification. However, Firefly optimization is more time-efficient than Swarm, completing in 8 minutes and 39 seconds compared to Swarm's longer duration.

In the Random Forest model, both Swarm and Firefly optimizations achieve perfect precision (1.00) and high recall (0.96). Firefly slightly surpasses Swarm in F1-score (0.98 vs. 0.97) and accuracy (0.99 vs. 0.98). However, Firefly optimization requires more time, taking 5 minutes and 50 seconds compared to Swarm. For KNN, Firefly optimization outperforms Swarm across all metrics. It has higher precision (0.97 vs. 0.92), recall, F1-score, and accuracy (0.96 vs. 0.95 and 0.98 vs. 0.97). Additionally, Firefly optimization exhibits lower temporal complexity, taking 8 seconds compared to Swarm’s 17 seconds. When using GaussianNB, the Firefly optimization technique outperforms Swarm in terms of accuracy (1.00 vs. 0.97), recall (0.96 vs. 0.93), and F1-score (0.98 vs. 0.95). Both strategies exhibit the same time complexity of two seconds. Firefly optimization consistently delivers slightly better classification results compared to Swarm, though it may occasionally take longer. KNN benefits the most from Firefly optimization, while GaussianNB performs efficiently in terms of time complexity with both optimization algorithms.

**A graph of a number of objects

Description automatically generated**

Fig.5. Particle Swarm Comparison

The Fig.5 illustrates the comparison of accuracy and temporal complexity among four classification models: MLP Classifier (ANN), Random Forest, KNN, and GaussianNB. The x-axis represents the different classification models, while the y-axis displays the accuracy and time complexity metrics.

Table 3: Without Using Swarm and Firefly

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithms | PRECISION | RECALL | F1-SCORE | ACCURACY | TIME  COMPLEXITY |
| MLP Classifier | 1.00 | 1.00 | 1.00 | 1.00 | 1m 33s |
| Random Forest | 1.00 | 1.00 | 1.00 | 1.00 | 1m 15s |
| KNN | 0.90 | 1.00 | 0.95 | 0.96 | 1s |
| GaussianNB | 0.43 | 0.99 | 0.60 | 0.57 | 1s |

The table contrasts the performance of four classification algorithms MLP Classifier, Random Forest, KNN, and GaussianNB across various metrics, including precision, recall, F1-score, accuracy, and time complexity.

The MLP (Multi-Layer Perceptron) Classifier delivers flawless scores across all metrics, including precision, recall, F1-score, and accuracy, showcasing its excellent data classification performance. Its time complexity is relatively low, clocking in at 1 minute and 33 seconds, indicating that the MLP Classifier is both effective and efficient in terms of processing time. The Random Forest algorithm also achieves perfect ratings of 1.00 in precision, recall, F1-score, and accuracy, like the MLP Classifier. It has a slightly lower time complexity, taking only 1 minute and 15 seconds, making it an efficient choice that provides excellent results.

The KNN (K-Nearest Neighbours) approach has a slightly lower precision of 0.90 but achieves perfect recall at 1.00, leading to an F1-score of 0.95 and an accuracy of 0.96. Its time complexity is just one second, demonstrating efficient use of computational resources. However, its accuracy is lower compared to other models. On the other hand, GaussianNB (Gaussian Naive Bayes) presents a significant trade-off: its precision is notably low at 0.43, whereas its recall is very high at 0.99. This results in an F1-score of 0.60 and an accuracy of 0.57, indicating that it performs poorly in classification despite its fast execution time of one second.

Both the MLP Classifier and Random Forest deliver excellent classification results, offering a good balance between accuracy and processing efficiency. The KNN approach, while very efficient, has only minor trade-offs in precision. GaussianNB, on the other hand, is computationally efficient but falls short in terms of accuracy and overall classification performance.

**A graph of different types of firefly

Description automatically generated**

Fig.6. Particle Swarm Comparison

Fig.6 illustrates the performance of four classification models—MLP Classifier (ANN), Random Forest, KNN, and GaussianNB—using two metrics: accuracy and time complexity. The x-axis represents the different classification models, while the y-axis shows the scales for accuracy and time complexity.

**CHAPTER 7**

**TESTING**

Test dataset which consists of a portion of the original data that was not utilized during the training process. This subset is essential for assessing the model’s performance on new, unseen data. Ensure that the test data is preprocessed in the same way as the training data, including normalization or standardization, to maintain consistency in feature scale and representation. The features used for testing should match those selected during the optimization phase with Particle Swarm Optimization (PSO) and the Firefly Algorithm. Once the test data is prepared, each classification model ANN MLP, GaussianNB, KNN, and Random Forest is evaluated against it. The models make predictions based on the test data, and these predictions are compared to the actual labels to compute various performance metrics. These metrics accuracy, precision, recall, and the F1-score provide a comprehensive assessment of each model's performance. Accuracy reflects the model's overall correctness, precision measures the proportion of true positive predictions out of all positive predictions, recall indicates the model's effectiveness in identifying true positives, and F1 score integrates both accuracy and recall into one metric. The assessment results are utilized to evaluate the effectiveness of each model. A detailed analysis of the performance metrics for the ANN MLP and Random Forest models both enhanced through feature selection techniques revealed that these models exhibited strong performance across all metrics. The findings indicated that the ANN MLP, with optimized features, outperformed the Random Forest model in terms of precision, recall, F1-score, and accuracy, demonstrating its superior ability to accurately identify ASD patients. In addition to evaluating classification performance, the computing efficiency of each model is also assessed. This involves measuring the time required for each model to process test data and make predictions. During testing, the Random Forest classifier demonstrated superior performance compared to the ANN MLP. While the ANN MLP was slower, it yielded more accurate results. Efficiency is critical, especially for real-time applications that demand quick categorization. Additionally, the robustness of the models is tested by evaluating their performance under various conditions, such as expanding the test set or introducing noise. This helps to understand how well the models adapt to new parameters while maintaining consistent performance.

**CHAPTER 8**

**CONCLUSION**

Improving autism spectrum disorder (ASD) classification through bio-inspired optimization methods achieved notable improvements in model performance and efficiency. By applying Particle Swarm Optimization (PSO) and the Firefly Algorithm for feature selection, the research enhanced the accuracy and reliability of various classifiers used on the Toddler Autism dataset from July 2018. The main objective was to improve ASD categorization by pinpointing the most critical features in the dataset, leading to enhanced classifier performance. The PSO and Firefly Algorithm played a crucial role in refining the feature set, resulting in significant performance gains across all classifiers tested. Among the models analysed, ANN MLP, GaussianNB, K-Nearest Neighbours (KNN), and Random Forest exhibited the most significant performance improvements. With the updated feature sets, the ANN MLP achieved perfect scores in precision, recall, F1-score, and accuracy, highlighting its ability to consistently classify ASD patients. The Random Forest model also performed impressively with high accuracy and precision, showcasing its reliability over time. However, temporal efficiency was a key concern. The Random Forest classifier processed the test data more quickly than the ANN MLP. Despite its longer processing time, the ANN MLP's enhanced accuracy with optimized features made the trade-off worthwhile. This study advances the field of ASD classification by demonstrating the efficacy of bio-inspired optimization strategies for feature selection. The results show that combining ANN MLP and Random Forest models with PSO and the Firefly Algorithm improves both classification accuracy and efficiency. These advancements could significantly enhance early detection and treatment of ASD, underscoring the importance of integrating novel computational methods into diagnostic and therapeutic practices. The findings highlight the necessity of ongoing research and the application of innovative optimization techniques to boost diagnostic precision and reliability in ASD and beyond.

**BIBLIOGRAPHY**

**Conference Papers:**

[1] Lidia V. Gabis, Shahar Shefer and Sigal Portnoy, “Variability of Coordination in Typically Developing Children Versus Children with Autism Spectrum Disorder with and without Rhythmic Signal”, 2020.

[2] Krishna Patra and Priyadarsan Parida, “Toddler ASD Classification Using Machine Learning Techniques”, 2021.

[3] Hala Shamseddine, Safa Otoum and Azzam Mourad, “A Federated Learning Scheme for Neuro-developmental Disorders: Multi-Aspect ASD Detection”, 2022.

[4] Marta Robles, Negar Namdarian, Julia Otto, Evelyn Wassiljew, Nassir Navab Fellow, Christine M. Falter-Wagner and Daniel Roth Member, “A Virtual Reality Based System for the Screening and Classification of Autism”, 2022.

[5] Engineering and Jamia Millia Islamia Univ, “Analysis and Detection of Autism Spectrum Disorder Using Machine Learning Techniques”, 2019.

[6] Maria Eleonora Minissi,Irene Alice Chicchi Giglioli, Fabrizia Mantovani and Mariano Alcañiz Raya, “Assessment of the Autism Spectrum Disorder Based on Machine Learning and Social Visual Attention: A Systematic Review”, 2021.

[7] José Jaime Esqueda-Elizondo, Reyes Juárez-Ramírez Enrique Efrén García-Guerrero, Oscar Roberto López-Bonilla, Gilberto Manuel Galindo-Aldana, Laura Jiménez-Beristain Alejandra Serrano-Trujillo, Esteban Tlelo-Cuautle and EverardoInzunza-González, “Attention Measurement of an Autism Spectrum Disorder User Using EEGSignals: A Case Study”, 2022.

[8] Ishaani Priyadarshini, “Autism Screening in Toddlers and Adults Using Deep Learning and Fair AI Techniques”, 2023.

[9] Maraheb Alsuliman and Heyam H. Al-Baity, “Efficient Diagnosis of Autism with Optimized Machine Learning Models: An Experimental Analysis on Genetic and Personal Characteristic Datasets”, 2022.

[10] KmBhavna, Romi Banerjee and Dipanjan Roy, “End-to-End Explainable AI: Derived Theory-of-Mind Fingerprints to Distinguish Between Autistic and Typically developing and Social Symptom Severity”, 2023.

[11] S. Saravana Kumar, K. Selvakumar and V. Senthil Murugan, “Enhanced Whale Optimization and Ensemble Learning Algorithm for Autism Spectrum Disorder Classification”, 2024.

[12] Vishal Midya, Cecilia Sara Alcala, Elza Rechtman, Jill K. Gregory, Kurunthachalam Kannan, Irva Hertz-Picciotto, Susan L. Teitelbaum, Chris Gennings, Maria J. Rosa, and Damaskini Valvi, “Machine Learning Assisted Discovery of Interactions between Pesticides, Phthalates, Phenols, and Trace Elements in Child Neurodevelopment”, 2023.

[13] Yugandhar Bokka, R. N. V. Jagan Mohan and M. Chandra Naik, “Maternal Dyslipidemia During Pregnancy Correlates with Elevated Lipid Levels in One-Year-Old Infants”, 2024.

[14] Navita, Ashok Kumar and Chanambam Mokaju Meitei, “Melody Extraction from MIDI Files Using AI-ML Techniques”, 2024.

[15] Dhuha Dheyaa Khudhur and Saja Dheyaa Khudhur, “The classification of autism spectrum disorder by machine learning methods on multiple datasets for four age groups”, 2023.

**APPENDIX – Sample Source Code/Pseudo Code**

**PARTICULAR SWARM**

**!pip install pyswarm**

**import numpy as np**

**import pandas as pd**

**from sklearn.model\_selection import train\_test\_split, GridSearchCV**

**from sklearn.preprocessing import LabelEncoder, StandardScaler**

**from sklearn.neural\_network import MLPClassifier**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.naive\_bayes import GaussianNB**

**from sklearn.metrics import accuracy\_score**

**from pyswarm import pso**

**from sklearn.decomposition import PCA**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from google.colab import drive**

**drive.mount('/content/drive')**

**data = pd.read\_csv('/content/drive/MyDrive/Toddler Autism dataset July 2018.csv')**

**data**

**data.columns**

**data.isnull().sum()**

**data.duplicated().sum()**

**data.info()**

**data.describe()**

**data.head()**

**label\_encoders = {}**

**categorical\_columns = ['Sex', 'Ethnicity', 'Jaundice', 'Family\_mem\_with\_ASD', 'Who completed the test','Class/ASD Traits ']**

**for column in categorical\_columns:**

**label\_encoders[column] = LabelEncoder()**

**data[column] = label\_encoders[column].fit\_transform(data[column])**

**X = data.drop(['Case\_No', 'Class/ASD Traits '], axis=1)**

**y = data['Class/ASD Traits ']**

**Feature scaling using StandardScaler**

**scaler = StandardScaler()**

**X\_scaled = scaler.fit\_transform(X)**

**pca = P(n\_components=15)  # Adjust n\_components based on the dataset**

**X\_reduced = pca.fit\_transform(X\_scaled)**

**pca**

**Define Particle Swarm Optimization objective function for feature selection**

1. **MLP Classifier**

**def objective\_function(features, X, y): features = np.array(features).astype(bool) X\_selected = X[:, features] X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42) clf = MLPClassifier(max\_iter=1000, random\_state=52) clf.fit(X\_train, y\_train) y\_pred = clf.predict(X\_test) return -accuracy\_score(y\_test, y\_pred)**

**num\_features = X\_reduced.shape[1]**

**lb = [0] \* num\_features**

**ub = [1] \* num\_features**

**num\_features**

**lb**

**ub**

**best\_features, \_ = pso(objective\_function, lb, ub, args=(X\_reduced, y), swarmsize=10, maxiter=25)**

**best\_features**

**X\_selected = X\_reduced[:, best\_features.astype(bool)]**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=52)**

**param\_grid = {**

**'hidden\_layer\_sizes': [(50,), (30, 30)],**

**'activation': ['relu', 'tanh'],**

**'solver': ['adam', 'sgd'],**

**'alpha': [0.01, 0.1],  # Increase alpha for stronger regularization**

**'learning\_rate': ['constant', 'adaptive']**

**}**

**grid\_search = GridSearchCV(MLPClassifier(max\_iter=1000, random\_state=52), param\_grid, cv=5, n\_jobs=-1, verbose=2)**

**grid\_search.fit(X\_train, y\_train)**

**# Best parameters**

**best\_params = grid\_search.best\_params\_**

**# Train the model with best parameters**

**best\_clf = MLPClassifier(max\_iter=1000, random\_state=52, \*\*best\_params)**

**best\_clf.fit(X\_train, y\_train)**

**# Predictions**

**y\_pred\_train = best\_clf.predict(X\_train)**

**y\_pred\_test = best\_clf.predict(X\_test)**

**# Accuracy**

**train\_accuracy = accuracy\_score(y\_train, y\_pred\_train)**

**test\_accuracy = accuracy\_score(y\_test, y\_pred\_test)**

**print("Train Accuracy:", train\_accuracy)**

**print("Test Accuracy:", test\_accuracy)**

**print("Best Parameters:", best\_clf.get\_params())**

**print("Best Parameters:", best\_params)**

**clf = MLPClassifier(max\_iter=1000, random\_state=42, early\_stopping=True)**

**clf.fit(X\_train, y\_train)**

**y\_pred = clf.predict(X\_test)**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**print("Final accuracy with selected features:", accuracy)**

**print("\nClassification Report (Train):")**

**print(classification\_report(y\_train, y\_pred\_train))**

**print("\nClassification Report (Test):")**

**print(classification\_report(y\_test, y\_pred\_test))**

**# Confusion Matrix**

**conf\_matrix\_train = confusion\_matrix(y\_train, y\_pred\_train)**

**conf\_matrix\_test = confusion\_matrix(y\_test, y\_pred\_test)**

**plt.figure(figsize=(14, 5))**

**plt.subplot(1, 2, 1)**

**sns.heatmap(conf\_matrix\_train, annot=True, fmt='d', cmap='Blues')**

**plt.title('Confusion Matrix - Train')**

**plt.xlabel('Predicted')**

**plt.ylabel('True')**

**plt.subplot(1, 2, 2)**

**sns.heatmap(conf\_matrix\_test, annot=True, fmt='d', cmap='Blues')**

**plt.title('Confusion Matrix - Test')**

**plt.xlabel('Predicted')**

**plt.ylabel('True')**

**plt.tight\_layout()**

**plt.show()**

**2) Random Forest**

**Define Particle Swarm Optimization objective function for feature selection**

**def objective\_function(features, X, y):**

**features = np.array(features).astype(bool)**

**X\_selected = X[:, features]**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)**

**clf = RandomForestClassifier(random\_state=52)**

**clf.fit(X\_train, y\_train)**

**y\_pred = clf.predict(X\_test)**

**return -accuracy\_score(y\_test, y\_pred)**

**num\_features = X\_reduced.shape[1]**

**lb = [0] \* num\_features**

**ub = [1] \* num\_features**

**best\_features, \_ = pso(objective\_function, lb, ub, args=(X\_reduced, y), swarmsize=10, maxiter=25)  # Adjust swarmsize and maxiter**

**Select best features**

**X\_selected = X\_reduced[:, best\_features.astype(bool)]**

**Split data into train and test sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=52)**

**Define parameter grid for Random Forest**

**param\_grid = {**

**'n\_estimators': [100, 200],**

**'max\_depth': [None, 10, 20],**

**'min\_samples\_split': [2, 5, 10],**

**'min\_samples\_leaf': [1, 2, 4],**

**'bootstrap': [True, False]**

**}**

**Grid search based optimization**

**grid\_search = GridSearchCV(RandomForestClassifier(random\_state=52), param\_grid, cv=5, n\_jobs=-1, verbose=2)**

**grid\_search.fit(X\_train, y\_train)**

**# Best parameters**

**best\_params = grid\_search.best\_params\_**

**# Train the model with best parameters**

**best\_clf = RandomForestClassifier(random\_state=52, \*\*best\_params)**

**best\_clf.fit(X\_train, y\_train)**

**# Predictions**

**y\_pred\_train = best\_clf.predict(X\_train)**

**y\_pred\_test = best\_clf.predict(X\_test)**

**# Accuracy**

**train\_accuracy = accuracy\_score(y\_train, y\_pred\_train)**

**test\_accuracy = accuracy\_score(y\_test, y\_pred\_test)**

**print("Train Accuracy:", train\_accuracy)**

**print("Test Accuracy:", test\_accuracy)**

**print("Best Parameters:", best\_clf.get\_params())**

**print("Best Parameters:", best\_params)**

**# Final accuracy with selected features**

**y\_pred = best\_clf.predict(X\_test)**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**print("Final accuracy with selected features:", accuracy)**

**Classification Report**

**print("\nClassification Report (Train):")**

**print(classification\_report(y\_train, y\_pred\_train))**

**print("\nClassification Report (Test):")**

**print(classification\_report(y\_test, y\_pred\_test))**

**# Confusion Matrix**

**conf\_matrix\_train = confusion\_matrix(y\_train, y\_pred\_train)**

**conf\_matrix\_test = confusion\_matrix(y\_test, y\_pred\_test)**

**plt.figure(figsize=(14, 5))**

**plt.subplot(1, 2, 1)**

**sns.heatmap(conf\_matrix\_train, annot=True, fmt='d', cmap='Blues')**

**plt.title('Confusion Matrix - Train')**

**plt.xlabel('Predicted')**

**plt.ylabel('True')**

**plt.subplot(1, 2, 2)**

**sns.heatmap(conf\_matrix\_test, annot=True, fmt='d', cmap='Blues')**

**plt.title('Confusion Matrix - Test')**

**plt.xlabel('Predicted')**

**plt.ylabel('True')**

**plt.tight\_layout()**

**plt.show()**

1. **KNN**

**Define Particle Swarm Optimization objective function for feature selection**

**def objective\_function(features, X, y):**

**features = np.array(features).astype(bool)**

**X\_selected = X[:, features]**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)**

**clf = KNeighborsClassifier()**

**clf.fit(X\_train, y\_train)**

**y\_pred = clf.predict(X\_test)**

**return -accuracy\_score(y\_test, y\_pred)**

**num\_features = X\_reduced.shape[1]**

**lb = [0] \* num\_features**

**ub = [1] \* num\_features**

**best\_features, \_ = pso(objective\_function, lb, ub, args=(X\_reduced, y), swarmsize=10, maxiter=25)**

**# Select best features**

**X\_selected = X\_reduced[:, best\_features.astype(bool)]**

**# Split data into train and test sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=52)**

**Define parameter grid for KNN**

**# Define parameter grid for KNN**

**param\_grid = {**

**'n\_neighbors': [3, 5, 7],**

**'weights': ['uniform', 'distance'],**

**'algorithm': ['auto', 'ball\_tree', 'kd\_tree', 'brute']**

**}**

**Grid search based optimization**

**from sklearn.neighbors import KNeighborsClassifier**

**grid\_search = GridSearchCV(KNeighborsClassifier(), param\_grid, cv=5, n\_jobs=-1, verbose=2)**

**grid\_search.fit(X\_train, y\_train)**

**# Best parameters**

**best\_params = grid\_search.best\_params\_**

**# Train the model with best parameters**

**best\_clf = KNeighborsClassifier(\*\*best\_params)**

**best\_clf.fit(X\_train, y\_train)**

**# Predictions**

**y\_pred\_train = best\_clf.predict(X\_train)**

**y\_pred\_test = best\_clf.predict(X\_test)**

**# Accuracy**

**train\_accuracy = accuracy\_score(y\_train, y\_pred\_train)**

**test\_accuracy = accuracy\_score(y\_test, y\_pred\_test)**

**print("Train Accuracy:", train\_accuracy)**

**print("Test Accuracy:", test\_accuracy)**

**print("Best Parameters:", best\_clf.get\_params())**

**print("Best Parameters:", best\_params)**

**# KNN does not have early\_stopping parameter, skipping that part**

**accuracy = accuracy\_score(y\_test, y\_pred\_test)**

**print("Final accuracy with selected features:", accuracy)**

**Classification Report**

**print("\nClassification Report (Train):")**

**print(classification\_report(y\_train, y\_pred\_train))**

**print("\nClassification Report (Test):")**

**print(classification\_report(y\_test, y\_pred\_test))**

**# Confusion Matrix**

**conf\_matrix\_train = confusion\_matrix(y\_train, y\_pred\_train)**

**conf\_matrix\_test = confusion\_matrix(y\_test, y\_pred\_test)**

**plt.figure(figsize=(14, 5))**

**plt.subplot(1, 2, 1)**

**sns.heatmap(conf\_matrix\_train, annot=True, fmt='d', cmap='Blues')**

**plt.title('Confusion Matrix - Train')**

**plt.xlabel('Predicted')**

**plt.ylabel('True')**

**plt.subplot(1, 2, 2)**

**sns.heatmap(conf\_matrix\_test, annot=True, fmt='d', cmap='Blues')**

**plt.title('Confusion Matrix - Test')**

**plt.xlabel('Predicted')**

**plt.ylabel('True')**

**plt.tight\_layout()**

**plt.show()**

**4)GaussianNB**

**Define Particle Swarm Optimization objective function for feature selection**

def objective\_function(features, X, y):

    features = np.array(features).astype(bool)

    X\_selected = X[:, features]

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)

    clf = GaussianNB()

    clf.fit(X\_train, y\_train)

    y\_pred = clf.predict(X\_test)

    return -accuracy\_score(y\_test, y\_pred)

num\_features = X\_reduced.shape[1]

lb = [0] \* num\_features

ub = [1] \* num\_features

best\_features, \_ = pso(objective\_function, lb, ub, args=(X\_reduced, y), swarmsize=10, maxiter=25)  # Adjust swarmsize and maxiter

#Select best features

X\_selected = X\_reduced[:, best\_features.astype(bool)]

#Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=52)

clf = GaussianNB()

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

# Predictions

y\_pred\_train = clf.predict(X\_train)

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

# Accuracy

train\_accuracy = accuracy\_score(y\_train, y\_pred\_train)

test\_accuracy = accuracy\_score(y\_test, y\_pred\_test)

print("Train Accuracy:", train\_accuracy)

print("Test Accuracy:", test\_accuracy)

**Classification Report**

print("\nClassification Report (Train):")

print(classification\_report(y\_train, y\_pred\_train))

print("\nClassification Report (Test):")

print(classification\_report(y\_test, y\_pred\_test))

# Confusion Matrix

conf\_matrix\_train = confusion\_matrix(y\_train, y\_pred\_train)

conf\_matrix\_test = confusion\_matrix(y\_test, y\_pred\_test)

plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)

sns.heatmap(conf\_matrix\_train, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix - Train')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.subplot(1, 2, 2)

sns.heatmap(conf\_matrix\_test, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix - Test')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.tight\_layout()

plt.show()

FIREFLY

from google.colab import drive

drive.mount('/content/drive')

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.neural\_network import MLPClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

#from pyswarm import pso

from sklearn.decomposition import PCA

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read\_csv('/content/drive/MyDrive/Toddler Autism dataset July 2018.csv')

data

data.columns

data.isnull().sum()

data.duplicated().sum()

data.info()

data.describe()

data.head()

**Encode categorical variables**

label\_encoders = {}

categorical\_columns = ['Sex', 'Ethnicity', 'Jaundice', 'Family\_mem\_with\_ASD', 'Who completed the test','Class/ASD Traits ']

for column in categorical\_columns:

    label\_encoders[column] = LabelEncoder()

    data[column] = label\_encoders[column].fit\_transform(data[column])

X = data.drop(['Case\_No', 'Class/ASD Traits '], axis=1)

y = data['Class/ASD Traits ']

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

pca = PCA(n\_components=15)  # Adjust n\_components based on the dataset

X\_reduced = pca.fit\_transform(X\_scaled)

pca

1) MLP Classifier

**Define Fire Fly Optimization objective function for feature selection**

def objective\_function(features, X, y):

    features = np.array(features).astype(bool)

    if not np.any(features):  # Ensure at least one feature is selected

        return float('inf')  # Return a high value for invalid solutions

    X\_selected = X[:, features]

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)

    clf = MLPClassifier(max\_iter=1000, random\_state=52)

    clf.fit(X\_train, y\_train)

    y\_pred = clf.predict(X\_test)

    return -accuracy\_score(y\_test, y\_pred)

# Firefly Algorithm

class FireflyAlgorithm:

    def \_\_init\_\_(self, objective\_function, lb, ub, args=(), pop\_size=20, max\_iter=50, alpha=0.5, beta\_min=0.2, gamma=1.0):

        self.objective\_function = objective\_function

        self.lb = np.array(lb)

        self.ub = np.array(ub)

        self.args = args

        self.pop\_size = pop\_size

        self.max\_iter = max\_iter

        self.alpha = alpha

        self.beta\_min = beta\_min

        self.gamma = gamma

    def run(self):

        dim = len(self.lb)

        population = np.random.randint(2, size=(self.pop\_size, dim))

        fitness = np.apply\_along\_axis(self.objective\_function, 1, population, \*self.args)

        best\_firefly = population[np.argmin(fitness)]

        best\_fitness = np.min(fitness)

        for t in range(self.max\_iter):

            for i in range(self.pop\_size):

                for j in range(self.pop\_size):

                    if fitness[i] > fitness[j]:

                        r = np.linalg.norm(population[i] - population[j])

                        beta = self.beta\_min + (1 - self.beta\_min) \* np.exp(-self.gamma \* r \*\* 2)

                        population[i] = population[i] + beta \* (population[j] - population[i]) + \

                                        self.alpha \* (np.random.rand(dim) - 0.5)

                        population[i] = np.clip(population[i], self.lb, self.ub)

                        population[i] = np.round(population[i])

                        if not np.any(population[i]):  # Ensure at least one feature is selected

                            population[i][np.random.randint(dim)] = 1

                        fitness[i] = self.objective\_function(population[i], \*self.args)

            best\_firefly = population[np.argmin(fitness)]

            best\_fitness = np.min(fitness)

        return best\_firefly, best\_fitness

num\_features = X\_reduced.shape[1]

lb = [0] \* num\_features

ub = [1] \* num\_features

fa = FireflyAlgorithm(objective\_function, lb, ub, args=(X\_reduced, y), pop\_size=10, max\_iter=25)

best\_features, \_ = fa.run()

# Select best features

X\_selected = X\_reduced[:, best\_features.astype(bool)]

# Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=52)

# Define parameter grid for ANN

param\_grid = {

    'hidden\_layer\_sizes': [(50,), (30, 30)],

    'activation': ['relu', 'tanh'],

    'solver': ['adam', 'sgd'],

    'alpha': [0.01, 0.1],  # Increase alpha for stronger regularization

    'learning\_rate': ['constant', 'adaptive']

}

grid\_search = GridSearchCV(MLPClassifier(max\_iter=1000, random\_state=52), param\_grid, cv=5, n\_jobs=-1, verbose=2)

grid\_search.fit(X\_train, y\_train)

# Best parameters

best\_params = grid\_search.best\_params\_

# Train the model with best parameters

best\_clf = MLPClassifier(max\_iter=1000, random\_state=52, \*\*best\_params)

best\_clf.fit(X\_train, y\_train)

# Predictions

y\_pred\_train = best\_clf.predict(X\_train)

y\_pred\_test = best\_clf.predict(X\_test)

# Accuracy

train\_accuracy = accuracy\_score(y\_train, y\_pred\_train)

test\_accuracy = accuracy\_score(y\_test, y\_pred\_test)

print("Train Accuracy:", train\_accuracy)

print("Test Accuracy:", test\_accuracy)

print("Best Parameters:", best\_clf.get\_params())

print("Best Parameters:", best\_params)

clf = MLPClassifier(max\_iter=1000, random\_state=42, early\_stopping=True)

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

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

accuracy = accuracy\_score(y\_test, y\_pred)

print("Final accuracy with selected features:", accuracy)

print("\nClassification Report (Train):")

print(classification\_report(y\_train, y\_pred\_train))

print("\nClassification Report (Test):")

print(classification\_report(y\_test, y\_pred\_test))

# Confusion Matrix

conf\_matrix\_train = confusion\_matrix(y\_train, y\_pred\_train)

conf\_matrix\_test = confusion\_matrix(y\_test, y\_pred\_test)

plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)

sns.heatmap(conf\_matrix\_train, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix - Train')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.subplot(1, 2, 2)

sns.heatmap(conf\_matrix\_test, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix - Test')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.tight\_layout()

plt.show()

num\_features = X\_reduced.shape[1]

lb = [0] \* num\_features

ub = [1] \* num\_features

fa = FireflyAlgorithm(objective\_function, lb, ub, args=(X\_reduced, y), pop\_size=10, max\_iter=25)

best\_features, \_ = fa.run()

# Select best features

X\_selected = X\_reduced[:, best\_features.astype(bool)]

# Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=52)

**Define parameter grid for ANN**

# Define parameter grid for ANN

param\_grid = {

    'hidden\_layer\_sizes': [(50,), (30, 30)],

    'activation': ['relu', 'tanh'],

    'solver': ['adam', 'sgd'],

    'alpha': [0.01, 0.1],  # Increase alpha for stronger regularization

    'learning\_rate': ['constant', 'adaptive']

}

grid\_search = GridSearchCV(MLPClassifier(max\_iter=1000, random\_state=52), param\_grid, cv=5, n\_jobs=-1, verbose=2)

grid\_search.fit(X\_train, y\_train)

# Best parameters

best\_params = grid\_search.best\_params\_

# Train the model with best parameters

best\_clf = MLPClassifier(max\_iter=1000, random\_state=52, \*\*best\_params)

best\_clf.fit(X\_train, y\_train)

# Predictions

y\_pred\_train = best\_clf.predict(X\_train)

y\_pred\_test = best\_clf.predict(X\_test)

# Accuracy

train\_accuracy = accuracy\_score(y\_train, y\_pred\_train)

test\_accuracy = accuracy\_score(y\_test, y\_pred\_test)

print("Train Accuracy:", train\_accuracy)

print("Test Accuracy:", test\_accuracy)

print("Best Parameters:", best\_clf.get\_params())

print("Best Parameters:", best\_params)

clf = MLPClassifier(max\_iter=1000, random\_state=42, early\_stopping=True)

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

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

accuracy = accuracy\_score(y\_test, y\_pred)

print("Final accuracy with selected features:", accuracy)

**Classification Report**

print("\nClassification Report (Train):")

print(classification\_report(y\_train, y\_pred\_train))

print("\nClassification Report (Test):")

print(classification\_report(y\_test, y\_pred\_test))

# Confusion Matrix

conf\_matrix\_train = confusion\_matrix(y\_train, y\_pred\_train)

conf\_matrix\_test = confusion\_matrix(y\_test, y\_pred\_test)

plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)

sns.heatmap(conf\_matrix\_train, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix - Train')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.subplot(1, 2, 2)

sns.heatmap(conf\_matrix\_test, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix - Test')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.tight\_layout()

plt.show()