Enhancing Classifier Performance with Evolutionary Weighting and Priority-Based Aggregation

Project-II (MA47202) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Integrated Master of Science

in

Mathematics and Computing

by

Atharv Bajaj

(20MA20014)

Under the supervision of

Professor Debjani Chakraborty

Department of Mathematics

Indian Institute of Technology Kharagpur

Spring Semester, 2023-24

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: (Atharv Bajaj)

Place: Kharagpur (20MA20014)

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Enhancing Classifier Performance with Evolutionary Weighting and Priority-Based Aggregation**" submitted by **Atharv Bajaj** (Roll No. 20MA20014) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Integrated Master of Science in Mathematics and Computing is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2023-24.

Date:
Place: Kharagpur

Professor Debjani Chakraborty
Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Ensemble methods represent a sophisticated approach to machine learning, aimed at harnessing the collective power of multiple base classifiers to construct a superior classifier. This research introduces an innovative ensemble technique that revolves around the iterative training of a base classifier, each time applying diverse weightings to the training samples. Inspired by principles of evolutionary computation and aggregation theory, our approach leverages evolutionary algorithms to explore the vast space of possible weightings, seeking those that best enhance classifier performance.

By employing crossover and mutation operations within the weighting space, we iteratively refine and optimize the set of weights across successive generations. The base classifier is trained on the training examples alongside their respective weightings, and over a finite number of generations, the weights evolve and converge towards optimal configurations. The resulting classifiers from different generations are aggregated to form the final ensemble.

Central to our approach is the concept of priority-based averaging aggregation, where different generations are assigned varying levels of importance. This allows us to incorporate diverse perspectives and insights gained throughout the evolutionary process, enriching the final ensemble's robustness and generalization capabilities. Our ensemble algorithm is evaluated using the UCI benchmark dataset, demonstrating superior accuracy, consistency, and reliability compared to existing state-of-the-art methods.

In addition to assessing the performance of the ensemble algorithm, we conduct SHAP analysis to elucidate the contribution of individual features to model predictions. By analyzing SHAP values, we gain valuable insights into the underlying relationships within the data, validating the effectiveness of our ensemble approach in capturing and leveraging these relationships for enhanced classification accuracy.

Acknowledgement

I wish to extend my heartfelt appreciation to Professor Debjani Chakraborty, my esteemed supervisor, for her consistent and invaluable guidance, unwavering support, addressing all my inquiries, and assisting me whenever I encountered obstacles. She also helped me focus on the crucial aspects of the project and ensured that I conducted my work in a goal-driven manner.

Moreover, I would like to acknowledge the Department of Mathematics, IIT Kharagpur, for providing the essential resources that significantly contributed to the execution of this project. Lastly, my sincere thanks go out to my family and friends, whose constant encouragement has been a vital source of motivation throughout this journey.

Contents

Declaration i

Certificate ii

Abstract iii

Acknowledgement v

Contents vi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research Goals . . . 2
- 1.3 Paper Structure . . . 5

2 Background 6
- 2.1 Related Work . . . 6
  - 2.1.1 Ensemble classification . . . 6
  - 2.1.2 Evolutionary Algorithms . . . 7
  - 2.1.3 Aggeregation Operators . . . 9
- 2.2 Fundamental Concepts . . . 10
  - 2.2.1 Ensemble Classifiers . . . 10
  - 2.2.2 Aggregation Operators . . . 11

3 Proposed methodology 13
- 3.1 Evolutionary-Based Classifier Generation . . . 13
- 3.2 Ensemble of classifiers using PA with priority degrees . . . 16
- 3.3 Exploring Model Interpretability . . . 19
  - 3.3.1 Shapely Values . . . 19
    - 3.3.1.1 Introduction . . . 19
    - 3.3.1.2 Estimating Shapley Values . . . 20
    - 3.3.1.3 Conclusion . . . 21

4 Experiments **22**
4.1 Dataset . . . 22
4.2 Performance evaluation . . . 23
4.2.0.1 Performance Metrics . . . 23
4.2.0.2 Mean Accuracy . . . 24
4.2.0.3 Precision . . . 24
4.2.0.4 Recall . . . 24
4.2.0.5 F1-Score . . . 24
4.2.0.6 Area Under the ROC Curve (AUC Score) . . . 24
4.2.0.7 Matthews Correlation Coefficient (MCC) . . . 25
4.3 Experimental setup . . . 25
4.4 Results and comparative analysis . . . 28
4.4.1 Weights Obtained for Each Layer . . . 28
4.4.2 Performance Metrics of Various Models . . . 29
4.4.3 Performance Metrics of the Proposed Methodology . . . 29
4.4.4 Inferences . . . 29
4.4.5 Performance Metrics Comparison . . . 30
4.4.6 Feature Importance Analysis using SHAP Values . . . 31

5 Conclusion **36**
5.1 Limitations . . . 36
5.2 Future Scope . . . 37

A **39**
A.1 Important Links . . . 39

Bibliography **40**

Chapter 1

Introduction

1.1 Motivation

The quest to enhance generalization performance in machine learning has been a longstanding pursuit, with ensemble learning emerging as a prominent approach over the past two decades. By harnessing the collective wisdom of multiple base classifiers, ensemble techniques offer a promising avenue for improving predictive accuracy across a variety of applications.

At the heart of ensemble learning lies the challenge of constructing a diverse set of base classifiers and effectively combining their outputs. Various methods have been devised to train base classifiers, ranging from perturbing training samples and parameters to exploring diverse model structures. For instance, Bagging employs bootstrap sampling to generate diverse training sets, while innovative techniques like multi-modal perturbation aim to foster diversity among base classifiers.

Equally crucial is the fusion strategy employed to combine the outputs of base classifiers. While traditional approaches such as simple voting and weighted voting have been widely used, recent studies have shown that selectively combining subsets of base classifiers can yield even better performance. The concept of selective ensembles, where only a portion of base classifiers are combined, has shown promise in improving overall accuracy.

A pivotal aspect of ensemble learning revolves around the strategic generation of a diverse array of classifiers, each contributing its unique perspective towards deriving a consensus opinion. This crucial step typically involves either employing a single base learning classifier with varied parameters or leveraging a diverse set of classifiers, ranging from Support Vector Machines (SVM) to Artificial Neural Networks (ANN) and beyond.

Hence, the foremost challenge lies in devising a reliable technique for efficiently generating this ensemble of classifiers. While numerous ensemble classification algorithms have been developed in recent decades, such as Bagging, Boosting, Random Forest, XGBoost, LightGBM, and CatBoost, they often rely on simplistic approaches like random sampling or feature-based subset selection for classifier generation.

1.2 Research Goals

This study seeks to address this limitation by first tackling the problem of generating multiple classifiers tailored to a specific task. The underlying rationale behind generating new classifiers is to manipulate the training examples to create instances that are challenging to classify. This manipulation entails assigning weightings to the training examples, thereby guiding the base learning classifier to focus on different aspects of the data.

However, the key challenge lies in determining the optimal weightings for the training examples to ensure the effectiveness of the ensemble. To this end, we propose a novel methodology inspired by evolutionary processes. By conceptualizing the assignment of different weightings to the training samples as an evolutionary optimization problem, we aim to iteratively refine and optimize the weightings over multiple rounds of training.

Evolutionary algorithms (EA) stand out as a prominent category of optimization techniques, drawing inspiration from nature's evolutionary processes like selection, mutation, crossover, and reproduction. These algorithms start with a random population of individuals, each representing a potential solution to the optimization problem at hand. Through successive iterations guided by a fitness function, individuals compete and evolve, leading to the emergence of increasingly optimal solutions.

Incorporating EA, we effectively manage the assignment of weights to individual features within the training examples. Initially, random weights are allocated to each feature, initiating the evolutionary process where these weights evolve and optimize over successive generations. Through the iterative refinement facilitated by crossover and mutation operations, the diversity of weight configurations is enhanced, enabling comprehensive exploration of the feature space.

Over a finite number of generations, the training examples, alongside their feature-specific weightings, are inputted into the designated base learning algorithm. This initiates the creation of new classifiers, with each classifier tailored to a unique feature weighting configuration. By continually adjusting the feature weightings guided by a fitness function, EA facilitates the generation of a diverse ensemble of classifiers, collectively enriching the predictive capacity of the ensemble.

This approach underscores the significance of feature weighting optimization in enhancing the ensemble's adaptability and predictive performance. Through the iterative refinement of feature weightings driven by evolutionary principles, our methodology aims to harness the full potential of the feature space, thereby advancing the efficacy of ensemble learning in machine learning tasks.

Continuing with the ensemble model construction, the subsequent crucial phase involves the combination step, where the insights from various classifiers are strategically integrated. The overarching aim of ensemble techniques is to harness the collective intelligence of multiple classifiers to enhance detection accuracy. Existing literature categorizes ensemble methods into two main types: homogeneous and heterogeneous. While homogeneous ensembles employ base models of the same type, heterogeneous ensembles leverage diverse base models.

Traditionally, the combination step in ensemble methods relies on aggregation operators to fuse the predictions of individual classifiers. Common aggregation operators like arithmetic mean and weighted arithmetic mean are frequently employed to consolidate information across classifiers. However, these operators typically treat all classifiers equally, without considering potential variations in their performance or priority.

To address this limitation, our study introduces the concept of prioritized aggregation operators, a subclass of aggregation operators that assigns importance based on

specified priorities. By integrating prioritized aggregation into ensemble modeling, we aim to account for variations in classifier performance and prioritize more reliable classifiers.

Our proposed ensemble method leverages the priorities assigned to different generations of the evolutionary process. By incorporating these priorities into the aggregation process, we assign weights to the average generation-wise classifiers, ensuring that higher-priority classifiers contribute more significantly to the ensemble's decision-making process.

In essence, our methodology facilitates the creation of ensemble classifiers that adaptively prioritize the contributions of individual classifiers based on their performance and assigned priorities. By integrating prioritized aggregation into ensemble modeling, we enhance the practical utility and effectiveness of ensemble methods in real-world applications.

In addition to our ensemble methodology, we also integrate SHAP (SHapley Additive exPlanations) analysis to further enhance our understanding of model predictions and feature importance. By applying SHAP analysis, we gain insights into the contribution of individual features to the ensemble's decision-making process. This allows us to identify the most influential features driving the model's predictions, thereby improving the interpretability and trustworthiness of the ensemble classifier. Through the combination of ensemble modeling and SHAP analysis, our research aims to provide a comprehensive framework for accurate and interpretable machine learning models tailored to real-world applications.

The study's contributions are multifaceted and can be delineated as follows:

- Introduction of a novel approach to classifier generation through the strategic assignment of weightings to features during training. This process, inspired by evolutionary algorithms, optimizes the weightings to enhance classifier performance.
- Implementation of a novel classifier ensemble technique leveraging prioritized aggregation operators. By assigning priorities to generations of evolutionary algorithms based on their average fitness values, the ensemble method assigns priority-based weights to classifiers, enriching the ensemble's predictive power.

- Proposal of a priority degree-based ensemble method, wherein classifiers from different generations of evolutionary algorithms are combined with varying degrees of priority. The degree of prioritization is determined by the fitness values of the respective classifiers, offering insights into the relative importance of different generations in the ensemble process.
- Conduct of extensive experimental evaluations using benchmark UCI datasets, providing empirical validation of the proposed techniques.
- Comprehensive comparative analysis against state-of-the-art methods, offering insights into the effectiveness and superiority of the proposed approaches.
- Integration of SHAP (SHapley Additive exPlanations) analysis to provide insights into the contribution of individual features to the ensemble's decision-making process.

1.3 Paper Structure

The subsequent sections of the paper are organized as follows:

- **Section 2**: Thorough review of relevant literature, providing context and background for the proposed methodologies.
- **Section 3**: Elucidation of the proposed methodology for classifier ensembling using aggregation operators, detailing the underlying principles and processes.
- **Section 4**: Presentation of the experimental setup, results, and comparative analysis, shedding light on the performance and efficacy of the proposed techniques.
- **Section 5**: Summary of conclusions, limitations and recommendations for future research directions, highlighting avenues for further exploration and advancement in the field.

Chapter 2

Background

2.1 Related Work

In this section, we delve into the existing body of literature surrounding ensemble learning, evolutionary algorithms, and the application of aggregation operators. Additionally, we provide fundamental concepts essential for comprehending the proposed study.

2.1.1 Ensemble classification

Ensemble classification, a cornerstone of machine learning, involves amalgamating a multitude of classifiers to surpass the predictive capabilities of any single classifier(Rokach (2010)). This amalgamation entails training various classifiers on the same dataset using diverse methodologies. Numerous methodologies for combining the outputs of these classifiers have been explored in research literature.

Among the basic ensemble methodologies are simple voting, selective classifier ensemble, and weighted classifier ensemble(Liu et al. (2019)). Simple majority voting, a widely adopted technique, offers simplicity and computational efficiency while yielding commendable results. Extending this concept, heterogeneous ensembles utilize a majority voting approach to integrate diverse classifiers, as demonstrated in Atallah and Al-Mousa (2019).

Plurality voting (Lin et al. (2003)), an alternative to majority voting, has demonstrated superior performance in certain contexts. Recent research has proposed innovative voting methods based on weighted approaches, showcasing promising results in enhancing ensemble accuracy. Selective classifier ensemble methods discard the outputs of underperforming classifiers, often utilizing weighted approaches to assign significance to each classifier's output.

Dynamic and static weighting approaches have been explored in Tripathi et al. (2021) to enhance classifier performance within ensemble frameworks. Our study focuses on proposing an ensemble technique informed by the principles of evolutionary algorithms (EA) and aggregation operators.

Central to our approach is the strategic generation of new classifiers using EA, each tasked with classification objectives. Subsequently, the ensemble method efficiently aggregates the outputs of these classifiers, leveraging EA principles and aggregation operators to optimize performance.

By emphasizing the synergy between EA and aggregation operators, our study aims to contribute to the advancement of ensemble learning methodologies. The strategic generation and aggregation of classifiers represent pivotal steps in our proposed approach, promising enhanced predictive capabilities in classification tasks.

2.1.2 Evolutionary Algorithms

Evolutionary algorithms (EAs) have garnered significant attention for their prowess in tackling complex problems, owing to their ability to navigate diverse populations, expedite convergence rates, and pinpoint optimal solutions with remarkable accuracy. Recent studies showcase the versatility and efficacy of EAs across various domains.

For example, Deng et al. (2022) introduced a quantum-inspired differential evolution algorithm to tackle intricate problems, demonstrating its potential for addressing complex challenges. Zhao et al. (Nov 2022) combined the elastic net broad learning system (ENBLS) with the gray wolf optimization (GWO) algorithm, yielding improvements in model parameters and overall performance.

Huang et al. (2023) devised a novel three-phase co-evolutionary strategy, integrating a multiphase cooperative evolutionary technique with a competitive swarm optimizer (TPCSO) to enhance algorithm diversity and convergence. Similarly, Zhao et al. (Mar 2022) utilized an improved sparrow search algorithm to optimize deep belief network parameters, showcasing enhanced model optimization.

Furthermore, EAs find application in selection problems, as demonstrated by Maleki et al. (2021) and Guha et al. (2021), where feature reduction techniques were employed to enhance convergence. Multi-objective optimization problems also benefit from EAs, as evidenced by studies such as Zhou et al. (2021).

In the realm of machine learning, EAs offer adaptability across diverse tasks, including multi-task learning, decision tree induction, image segmentation, Bayesian optimization, and training deep learning models like recurrent neural networks (neuroevolution). Qing et al. (2021) utilized EAs to convert multiclass problems into binary class problems, streamlining the search space and improving overall performance.

Moreover, EAs serve as effective ensemble learning methods, enabling the assignment of weights to individual learners (Sylvester and Chawla 2005, 2006; García-Mendoza et al. 2020). Eskandari et al. (2023) employed a genetic algorithm to optimize classifier weights in a voting technique, aiming to minimize classification errors.

In another instance, Talatian Azad et al. (2022) leveraged EAs to optimize parameters of MLP Neural Networks (MLP-NN), enhancing model performance. Similarly, Sharma et al. (2022) utilized evolutionary-inspired techniques to optimize support vector machine parameters for stress detection using EEG signals, achieving improved accuracy.

Addressing imbalanced classification challenges, Rosales-Pérez et al. (2022) utilized an evolutionary approach to optimize hyper-parameters and support vector determination in SVM, enhancing model performance on severely imbalanced datasets.

These diverse applications underscore the versatility and effectiveness of EAs in tackling a myriad of challenges across various domains, showcasing their potential for driving innovation and advancement in computational methodologies.

2.1.3 Aggeregation Operators

Aggregation functions play a pivotal role in consolidating diverse sources of information into a cohesive representation. These functions, comprising mathematical operations, amalgamate inputs to yield a singular representative outcome. Throughout the literature, a plethora of aggregation models have been explored within the context of ensemble methods.

For instance, Li et al. (2015) introduced a support vector machine (SVM) integrated algorithm based on the Choquet integral, showcasing superior predictive performance compared to individual SVM algorithms. Pacheco and Krohling (2018) proposed an aggregation methodology utilizing the Choquet integral as a fuzzy measure, demonstrating efficacy in enhancing outcomes for neural classifiers.

Among the array of aggregation operators, the averaging operator stands out as one of the most widely employed for combination processes. Notably, the ordered weighted averaging (OWA) operator, as exemplified by Yager (1988, 1993), represents a prominent instance of such an averaging function. Zhou et al. (2010) introduced the type-1 ordered weighted averaging (OWA) operator, designed to directly aggregate uncertain information with uncertain weights, showcasing its applicability in domains such as breast cancer treatments.

Recent advancements have witnessed the emergence of prioritized aggregation operators, offering nuanced approaches to aggregation. Farid and Riaz (2022) developed prioritized operators, including the Pythagorean fuzzy prioritized geometric operator and the Pythagorean fuzzy prioritized averaging operator with priority degrees, demonstrating their efficacy in decision-making contexts. Garg and Rani (2021) proposed a unique prioritized averaging aggregation operator tailored for decision-making problems, ensuring fairness in outcomes by preventing higher satisfaction values from compensating for lower ones. In this study, we aim to leverage the concept of prioritized averaging aggregation operators to fuse results obtained from base learning classifiers. By harnessing the capabilities of these operators, we seek to enhance the aggregation process within ensemble methodologies, ultimately contributing to improved predictive performance and decision-making accuracy.

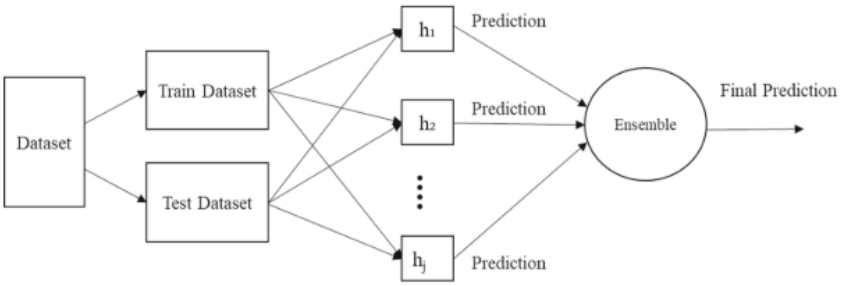

FIGURE 2.1: Ensemble classifier framework

2.2 Fundamental Concepts

In this section, we delve into fundamental concepts essential for grasping the essence of the proposed research. Initially, we provide a succinct overview of ensemble classifiers, followed by an elucidation of aggregation operators.

2.2.1 Ensemble Classifiers

Ensemble classifiers are a cornerstone in classification tasks, aiming to enhance system accuracy by amalgamating multiple weak classifiers. The fundamental principles underpinning the modeling of ensemble classifier systems for this study are delineated as follows.

Let X denote the feature space and Y represent the label space, where each $x \in X$ is associated with a label $y \in Y$. Consider a training dataset comprising a finite sequence of examples, $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where y_i denotes the label corresponding to the feature x_i. Consequently, a classifier is encapsulated as a mathematical function $h : X \to Y$, also referred to as the hypothesis, which endeavors to predict $y \in Y$ for any arbitrary $x \in X$.

Let $H = \{h_1, h_2, \ldots, h_j\}$ denote a finite set of classifiers. The ensemble is constructed by synthesizing these classifiers in a coherent and systematic manner. Figure 2.1 illustrates a visual representation of an ensemble of classifiers.

2.2.2 Aggregation Operators

Aggregation operators serve as a pivotal tool for amalgamating information from diverse data sources. The definition of an aggregation operator is articulated as follows.

Definition 1 Grabisch et al. (2009): An n-ary function $A : [0,1]^n \to [0,1]$ is deemed an aggregation operator if it satisfies the following properties:

1. A is increasing in each of its arguments, i.e., if $y_i \leq y_i'$ for each $i = 1, \ldots, n$, then $A(y_1, \ldots, y_n) \leq A(y_1', \ldots, y_n')$.
2. $A(0, \ldots, 0) = 0$ and $A(1, \ldots, 1) = 1$, i.e., it adheres to the boundary conditions.

Significant aggregation operators over the domain $[0,1]^2$ encompass t-norms, overlap functions, and copulas. These operators find extensive utility in diverse real-world phenomena.

Definition 2 Beliakov et al. (2010): An aggregation operator $f : [0,1]^n \to [0,1]$ is categorized as:

- Averaging: if it satisfies $\min(Y) \leq f(Y) \leq \max(Y)$ for all $Y \in [0,1]^n$.
- Conjunctive: if it fulfills the condition $f(Y) \leq \min(Y)$ for all $Y \in [0,1]^n$.
- Disjunctive: if it satisfies the condition $f(Y) \geq \max(Y)$ for all $Y \in [0,1]^n$.
- Mixed: if it does not fall under any of the above categories.

Definition 3 Yager (2008): Let, $\{y_1, y_2, \ldots, y_n\}$ be the set of criteria and let the criteria be strictly ranked in order of importance expressed by the strong priority rankings $y_1 > y_2 > \ldots > y_n$, where $y_v > y_{v+1}$ implies that the criterion y_v comes before y_{v+1} for every $v \in \{1, 2, \ldots, n-1\}$. Let $y_v(a)$ denote the performance of an alternative 'a' under the criteria y_v, where $y_v(a) \in [0,1]$. The prioritized averaging operator (PA) is given by

$$PA(y_1(a), y_2(a), \ldots, y_n(a)) = \sum_{v=1}^{n} \xi_v y_v(a), \tag{2.1}$$

where normalized importance weights of criteria y_v is given by $\xi_v = \frac{T_v}{\sum_{v=1}^{n} T_v}$, for $v = 1, \ldots, n$; $T_1 = 1$, and $T_v = \prod_{l=1}^{v-1} y_l(a)$, for $v = 2, 3, \ldots, n$.

Definition 4 Li and Xu (2019): Consider a strict prioritization ordering among the criteria $\{y_1, y_2, \ldots, y_n\}$, defined by specific priority orders as $y_1 >_{d_1} y_2 >_{d_2} \ldots >_{d_{n-1}} y_n$, where $y_v >_{d_v} y_{v+1}$ indicates that criterion y_v precedes criterion y_{v+1} with degree d_v, and $0 \leq d_v < \infty$ for $v \in \{1, 2, \ldots, n-1\}$. The performance of any alternative $'a'$ under criterion y_v is denoted by $y_v(a) \in [0, 1]$. Then, the priority averaging operator with priority degree d, denoted by P_{Ad}, is given by:

$$P_{Ad}(y_1(a), y_2(a), \ldots, y_n(a)) = \sum_{v=1}^{n} \xi_v y_v(a),$$

where the normalized importance weights of criteria y_v are calculated as $\xi_v = \frac{T_v}{\sum_{v=1}^{n} T_v}$ for $v = 1, \ldots, n$, with $T_1 = 1$, and $T_v = \prod_{l=1}^{v-1} (y_l(a))^{d_l}$ for $v = 2, 3, \ldots, n$.

Chapter 3

Proposed methodology

In this section, we begin by outlining the methodology for generating various base learning classifiers utilizing the concept of Evolutionary Algorithms (EA). Subsequently, we illustrate the process of classifier ensemble utilizing the notion of prioritized averaging aggregation operators with priority degree.

3.1 Evolutionary-Based Classifier Generation

Consider a dataset $S = \{(x_i, y_i)|x_i \in X, y_i \in Y, i = 1, 2, ..., N\}$, where X represents the feature space and Y denotes the label space. Initially, instead of creating a single classifier, we aim to develop a fixed number of classifiers by applying various weightings to the features in the training examples. Each distinct weighting yields a unique classifier, making the generation of an optimal set of weights paramount to our objective. To achieve this, we employ Evolutionary Algorithms (EAs). The EA process involves a population comprising K chromosomes evolving over J generations, resulting in a collection of chromosomes $\{w_{j,k}\}$, where $j = 1, 2, ..., J$ and $k = 1, 2, ..., K$. Each chromosome $w_{j,k}$ is defined as $w_{j,k} = (w_{j,k}(1), ..., w_{j,k}(D))$, satisfying $\sum_{d=1}^{D} w_{j,k}(d) = 1$ and $w_{j,k}(d) \geq 0$ for every $d = 1, 2, ..., D$, where D represents the number of features. In this scenario, we opt for real-valued coding rather than the more conventional bit string coding. Since we treat "weighting" as "chromosomes," these terms are interchangeable throughout our discussion. By feeding S

along with $w_{j,k}$ into a base learning algorithm, we derive a corresponding classifier $h_{j,k}$. Consequently, we obtain a diverse set of classifiers $h_{j,k}$ for different weightings $w_{j,k}$. This approach enables us to explore a range of classifier configurations, contributing to the overall robustness and adaptability of the ensemble system.

In our approach, we opted for an Artificial Neural Network (ANN) as the foundational algorithm for our ensemble. To evaluate the performance of each classifier generated by our Evolutionary Algorithm (EA), we employed the Mean Squared Error (MSE) metric to gauge the weighted training error of each classifier $h_{j,k}$ across the N examples. The fitness value $f_{j,k}$ of a chromosome $w_{j,k}$ serves as a measure of its effectiveness, determined by the accuracy achieved by the corresponding classifier $h_{j,k}$. Notably, a higher fitness value correlates with a lower training error, indicating the adaptability of each chromosome to the training data.

In our methodology, we ensure that the computational cost remains proportional to that of Bagging and AdaBoost techniques. Specifically, the final ensemble comprises $J \times K = T$ classifiers across all three methods. During the initial generation of the EA, the population of chromosomes $\{w_{1,k}\}_{k=1}^{K}$ is randomly generated. We derive $D * K$ random values from a uniform distribution in the range $(0, 1)$, which are then normalized to produce reliable weightings. Subsequently, we train a classifier corresponding to each chromosome in the population and evaluate its fitness based on the achieved accuracy. This iterative process enables the refinement of the ensemble, ensuring that each classifier contributes optimally to the overall performance.

The process for generating other classifiers using Evolutionary Algorithms (EA) can be summarized as follows:

1. **Selection**: Parents are selected using a roulette wheel selection procedure (Goldberg, 2013). Two chromosomes, $w_{j-1,k1}$ and $w_{j-1,k2}$, are sampled from the previous generation's population $\{w_{j-1,k}\}_{k=1}^{K}$ based on their fitness values. The fitness of a chromosome determines its selection probability.

2. **Crossover**: A uniform crossover is performed between the selected parent chromosomes $w_{j-1,k1}$ and $w_{j-1,k2}$ to produce offspring chromosomes $WF1$ and $WF2$. A uniform random real number U is chosen from the interval $(0, 1)$,

and if $U \leq r_C$, where r_C is the crossover rate (set to 0.5), bits are exchanged between the parent chromosomes to generate the offspring.

3. **Mutation**: The probability of mutation p_m is another parameter in the procedure. The weights in one of the offspring chromosomes, say $WF1$, are randomly paired, and with a probability of p_m, the values of the weights in each pair are swapped. This mutation process helps prevent premature convergence. Typically, a small value such as $p_m = 0.05$ is used, as it has been observed to increase the final classification accuracy.

After each iteration, the modified chromosome is saved as a member of the considered generation j. This process is repeated K times, resulting in a collection of chromosomes $\{w_{j,k}\}_{k=1}^{K}$ for each generation j. Each new generation of classifiers is trained and assessed for fitness using these modified weights. This iterative process is repeated for every generation $j = 1, 2, \ldots, J$. In this study, we set $J = K = 20$ to ensure a fair comparison with Adaboost and Bagging. While this may seem like a relatively small number of chromosomes and generations for optimization in conventional EA, our focus in machine learning is on the classifiers' generalization power, as indicated by their accuracy on novel, previously unobserved cases (estimated using a test set).

Algorithm 1 Main Algorithm for Classifier Generation and Ensemble

Given: Set of training examples S, number of generations J, population size K
Set generation $j = 1$
Randomly initialize the population of weightings $\{w_{1,k}\}_{k=1}^{K}$
Train a classifier $h_{1,k}$ for each $w_{1,k}$
Evaluate the fitness $f_{1,k}$ for each $w_{1,k}$
for $j = 2$ to J **do**
    Generate chromosome $\{w_{j,k}\}_{k=1}^{K}$ from $\{w_{j-1,k}\}_{k=1}^{K}$ with crossover and mutation
    Train a classifier $h_{j,k}$ for each $w_{j,k}$
    Evaluate the fitness $f_{j,k}$ for $w_{j,k}$
    Calculate the average fitness f_j^* for each classifier h_j^* obtained from the average of classifiers $h_{j,1}, h_{j,2}, \ldots, h_{j,K}$ for each generation $j = 1, 2, \ldots, J$
    Arrange classifiers $h_1^*, h_2^*, \ldots, h_J^*$ such that $h_1^* > h_2^* > \ldots > h_J^*$, where $h_v^* > h_{v+1}^*$ implies that $f_v^* > f_{v+1}^*$
    Generate ensemble of classifiers using Algorithm 2
end for

High accuracy on the training set does not guarantee similar accuracy on the test set due to the risk of overfitting. Thus, despite the potential for a converged population of classifiers to offer improved training accuracy, it may exhibit overfitting and have poorer generalization capacity. Consequently, from the EA, we obtain a set of base learning classifiers $h_{j,k}$, where $j = 1, 2, \ldots, J$ and $k = 1, 2, \ldots, K$. The steps for generating the classifiers are outlined in Algorithm 1.

Next, we combine all the classifiers we've created by using different weightings for the features to fix misclassification issues. We'll show way to do this using something called the prioritized averaging operator: considering priority. Let's break it down.

3.2 Ensemble of classifiers using PA with priority degrees

In this subsection, we will combine the classifiers generated from all generations using the prioritized averaging operator, denoted as P_{Ad}, which incorporates priority degrees.

The priority degrees are assigned based on the fitness values of the average classifiers obtained from each generation. Let $h_j = \{h_{j,1}, h_{j,2}, \ldots, h_{j,K}\}$ be the classifiers obtained at generation j, and let h_j^* be the respective average classifier. We calculate the fitness values $f_1^*, f_2^*, \ldots, f_J^*$ of $h_1^*, h_2^*, \ldots, h_J^*$ to prioritize the generations.

Without loss of generality, let $h_1^* > h_2^* > \ldots > h_J^*$, where $h_j^* > h_{j+1}^*$ implies that $f_j^* > f_{j+1}^*$. We then evaluate the priority degrees as follows:

$$\begin{aligned} d_1^* &= (f_1^* - f_2^*) \times 100 \\ d_2^* &= (f_2^* - f_3^*) \times 100 \\ &\vdots \\ d_{J-1}^* &= (f_{J-1}^* - f_J^*) \times 100. \end{aligned}$$

Thus, we have $h_1^* >_{d_1^*} h_2^* >_{d_2^*} \ldots >_{d_{J-1}^*} h_J^*$, where d_t^* indicates by how much percent the fitness of h_t^* is better than the fitness of h_{t+1}^*.

These fitness values and priority degrees are used to calculate the weights ξ_j for each generation $j = 1, \ldots, J$. The formula for ξ_j is given by:

$$\xi_j = \frac{T_j}{\sum_{j=1}^{J} T_j}, \quad \text{for } j = 1, 2, \ldots, J,$$

where $T_1 = 1$, and $T_j = \prod_{l=1}^{j-1} (f_l^*)^{d_l^*}$, for $j = 2, \ldots, J$.

Finally, we ensemble all the classifiers obtained from all generations by applying the P_{Ad} operator as follows:

$$P_{\text{Ad}}(h_1^*, h_2^*, \ldots, h_J^*) = \sum_{j=1}^{J} \xi_j h_j^*.$$

During the test phase, we utilize the ensemble of classifiers obtained from all generations using the weights ξ_j calculated during the training phase.

Algorithm 2 Ensemble using Prioritized aggregation with priority degrees

For the training phase:
Compute the fitness values $f_1^*, f_2^*, \ldots, f_J^*$ of the average classifiers $h_1^*, h_2^*, \ldots, h_J^*$, respectively.
Assume, without loss of generality, that $h_1^* > h_2^* > \cdots > h_J^*$ based on the priorities assigned to the different generation-wise classifiers, where $h_t^* > h_{t+1}^*$ implies $f_t^* > f_{t+1}^*$.
Calculate the degree vector $(d_1^*, d_2^*, \ldots, d_{J-1}^*)$ as follows:
    $d_j^* = (f_j^* - f_{j+1}^*) \times 100$, for $j = 1, 2, \ldots, J-1$
Establish the ordering among generations based on priority degrees:
    $h_1^* >_{d_1^*} h_2^* >_{d_2^*} \cdots >_{d_{J-1}^*} h_J^*$
Compute the weights ξ_j for the generations as follows:
    $T_1 = 1$, $T_j = \prod_{l=1}^{j-1} (f_l^*)^{d_l^*}$, for $j = 2, \ldots, J$
    $\xi_j = \frac{T_j}{\sum_{j=1}^{J} T_j}$, for $j = 1, 2, \ldots, J$
For the test phase:
Ensemble the classifiers obtained from all the generations using the following equation:
    $P_{\text{Ad}}(h_1^*, h_2^*, \ldots, h_J^*) = \sum_{j=1}^{J} \xi_j h_j^*$
where ξ_j are the weights of the generations obtained from the training phase.

The proposed algorithm is visually represented in Figure 3.1.

Remark 3.1: It is important to note that in the proposed algorithm, we have assumed strict prioritization among the average classifiers obtained from different

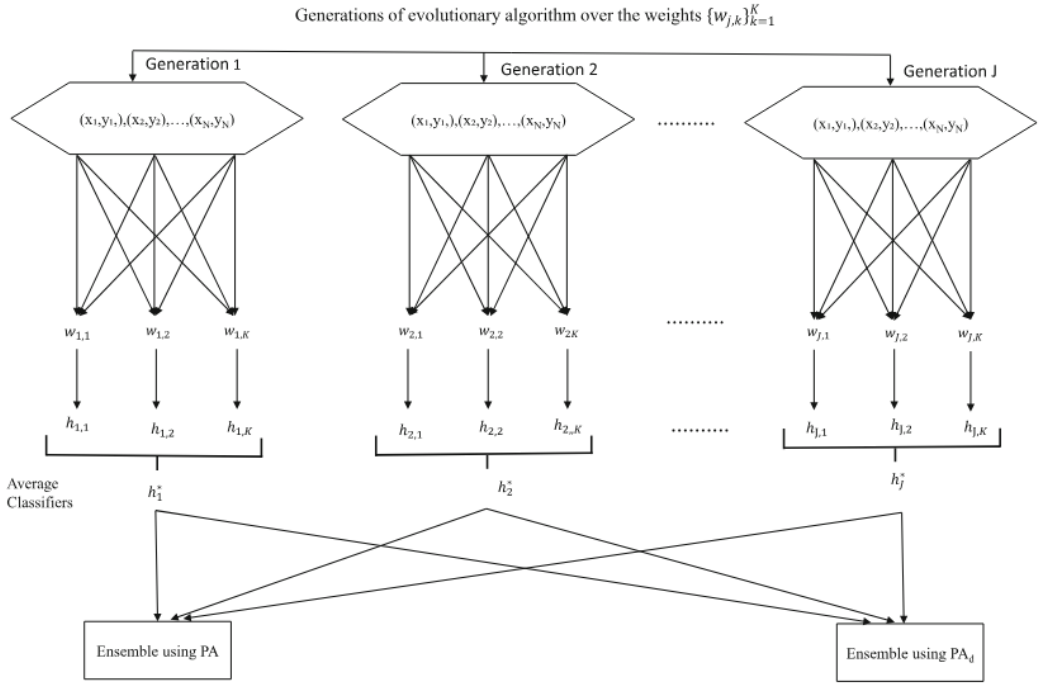

FIGURE 3.1: Classification framework of the proposed approach

generations, i.e., $h_1^* > \cdots > h_J^*$. However, there may be instances where some classifiers are equally prioritized. For example, there could be two classifiers h_j^* and h_k^* with the same fitness values, i.e., $h_j^* = h_k^*$. In such cases, we divide the set of average classifiers obtained from different generations, $\{h_1^*, \ldots, h_J^*\}$, into mutually disjoint subsets, denoted as $A_1^*, \ldots, A_T^*$. The priority among these subsets can be established as $A_1^* > \cdots > A_T^*$, where each classifier within subset A_t^* has the same fitness value for every $t = 1, \ldots, T$. If h_1^* and h_2^* belong to $A_{t_1}^*$ and $A_{t_2}^*$, respectively, where $t_1 < t_2$, then the fitness value of h_1^* is strictly greater than the fitness value of h_2^*. We can then apply prioritized aggregation operator as defined in Definitions 3 and 4 over the prioritized set $A_1^* > \cdots > A_T^*$ to obtain the respective normalized weights of the subsets A_t^*, for $t = 1, \ldots, T$. These normalized weights assigned to the subset A_t^* can then be equally distributed among the classifiers within that subset. This process needs to be repeated for each $t = 1, \ldots, T$.

3.3 Exploring Model Interpretability

3.3.1 Shapely Values

3.3.1.1 Introduction

In the quest to unravel the intricacies of machine learning models, understanding the driving factors behind predictions is paramount. While machine learning algorithms often provide accurate predictions, they can be perceived as "black boxes," making it challenging to comprehend the rationale behind individual predictions. To address this challenge, various interpretability techniques have emerged, among which SHAP (SHapley Additive exPlanations) values stand out as a powerful tool for model interpretation.

SHAP (SHapley Additive exPlanations) values offer a coherent framework for interpreting machine learning models by quantifying the contribution of each feature to the model's output. Unlike traditional feature importance techniques that provide global insights into feature relevance, SHAP values provide local interpretations, attributing specific feature effects to individual predictions.

At its core, SHAP values are rooted in cooperative game theory, specifically the Shapley value concept, which assigns a value to each player in a cooperative game based on their contribution to the game's outcome. In the context of machine learning, SHAP values extend this concept to assign contributions to each feature in a predictive model. These values capture how the inclusion of each feature influences the model's output relative to its absence.

Interpreting SHAP values involves understanding the direction and magnitude of each feature's impact on the model's predictions. Positive SHAP values indicate that the presence of a feature contributes to increasing the model's output, while negative values suggest the opposite. The magnitude of SHAP values reflects the strength of the feature's influence on the prediction, with larger values indicating greater importance.

3.3.1.2 Estimating Shapley Values

The estimation of Shapley values provides valuable insights into the contribution of individual features to model predictions. Approximating Shapley values involves a stochastic process that iteratively evaluates the impact of each feature on the model's output. Here's a breakdown of the process:

1. **Initialization:**
   - Select an instance of interest, denoted as x, and a target feature, denoted as j.
   - Specify the number of iterations, denoted as M, to control the sampling process.
2. **Sampling Procedure:**
   - For each iteration m from 1 to M:
     - Randomly select a sample instance z from the dataset.
     - Generate a random permutation of feature values.
3. **Instance Modification:**
   - Create two new instances:
     - x_{+j}: Combines feature values from x and z, with the j-th feature replaced by the corresponding feature value from z.

$$x_{+j} = (x(1), \ldots, x(j-1), x(j), z(j+1), \ldots, z(p))$$

     - x_{-j}: Similar to x_{+j}, but with the j-th feature replaced by the corresponding feature value from z.

$$x_{-j} = (x(1), \ldots, x(j-1), z(j), z(j+1), \ldots, z(p))$$

4. **Prediction Difference Calculation:**
   - Compute the difference in model predictions between x_{+j} and x_{-j} for each iteration:

$$\phi_m^j = \hat{f}(x_{+j}) - \hat{f}(x_{-j})$$

5. **Shapley Value Estimation:**

    - Average the prediction differences over all iterations to obtain the Shapley value for feature j at instance x:

$$\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \phi_m^j$$

By repeating this procedure for each feature of interest, the Shapley values for all features can be estimated, providing a comprehensive understanding of feature contributions to model predictions.

3.3.1.3 Conclusion

To conclude, the concept of Shapley values offers a principled framework for understanding feature importance in machine learning models. By adhering to fundamental properties such as Efficiency, Symmetry, Dummy, and Additivity, Shapley values provide a fair and interpretable way to attribute contributions to individual features. The Efficiency property ensures that the sum of feature contributions accurately reflects the difference between a specific prediction and the model's average prediction. Symmetry guarantees equitable treatment of features that contribute equally across all possible combinations. The Dummy property identifies features that have no impact on predictions, enhancing the transparency of feature importance assessments. Lastly, the Additivity property enables the combination of individual Shapley values to derive collective insights into feature importance. Together, these properties establish Shapley values as a robust and reliable method for model interpretation and decision support in various domains. As machine learning models continue to evolve in complexity and application, the adoption of Shapley values holds promise for advancing model explainability and fostering trust in AI-driven systems.

Chapter 4

Experiments

In the upcoming section, we delve into the experimental intricacies of our proposed approach. Our experiments are conducted on several benchmark datasets sourced from the UCI Machine Learning Repository. These datasets serve as a basis for comparing the results obtained from our proposed algorithm with those from other existing algorithms in the literature. To gauge the algorithm's learning capability, we utilize the accuracy measure of an ensemble classifier trained on the provided training examples. To commence, we provide a detailed description of the datasets utilized in our study.

4.1 Dataset

In the dataset, there are 918 instances and 12 columns/features. Table 4.1 shows a brief description of each feature: Each instance in the dataset represents a patient, and the target variable 'HeartDisease' indicates the presence (1) or absence (0) of heart disease based on diagnostic criteria.

Table 4.1: Description of Dataset Features

Name	Function	Type
Age	Age of the patient	Numerical
Sex	Gender of the patient	Categorical
ChestPainType	Type of chest pain experienced	Categorical
RestingBP	Resting blood pressure	Numerical
Cholesterol	Cholesterol level	Numerical
FastingBS	Fasting blood sugar level	Categorical
RestingECG	Resting electrocardiographic results	Categorical
MaxHR	Maximum heart rate achieved	Numerical
ExerciseAngina	Presence of exercise-induced angina	Categorical
Oldpeak	ST depression induced by exercise	Numerical
ST_Slope	Slope of peak exercise ST segment	Categorical
HeartDisease	Presence of heart disease	Categorical

4.2 Performance evaluation

In this section, we assess the effectiveness of our proposed approach through performance evaluation. Various metrics are available to measure the performance of algorithms, including mean accuracy, precision, recall, F1-score, area under the ROC curve (AUC score), and Matthews correlation coefficient (MCC). Below, we provide definitions for these performance measures and their mathematical expressions.

4.2.0.1 Performance Metrics

- **True Positive (TP):** Model predicts positive, and actual observation is positive.
- **True Negative (TN):** Model predicts negative, and actual observation is negative.
- **False Positive (FP):** Model predicts positive, but actual observation is negative.
- **False Negative (FN):** Model predicts negative, but actual observation is positive.

4.2.0.2 Mean Accuracy

Mean accuracy measures the overall correctness of the model's predictions.

$$\text{Mean Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100$$

4.2.0.3 Precision

Precision quantifies the accuracy of positive predictions.

$$\text{Precision} = \frac{TP}{TP + FP} \times 100$$

4.2.0.4 Recall

Recall calculates the proportion of correctly classified positive instances.

$$\text{Recall} = \frac{TP}{TP + FN} \times 100$$

4.2.0.5 F1-Score

The F1-score balances precision and recall, providing a comprehensive measure of a model's performance.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 100$$

4.2.0.6 Area Under the ROC Curve (AUC Score)

The AUC score quantifies the model's ability to discriminate between positive and negative instances across different thresholds.

$$\text{AUC Score} = \text{Area Under ROC Curve}$$

4.2.0.7 Matthews Correlation Coefficient (MCC)

The MCC considers all four confusion matrix values and is particularly useful for imbalanced datasets. It ranges from -1 to 1, with 1 indicating perfect prediction, 0 indicating random prediction, and -1 indicating total disagreement between prediction and observation.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

4.3 Experimental setup

The proposed methodology employs a Python implementation, leveraging tenfold cross-validation to mitigate randomness. In this process, the dataset is partitioned into 8:2 parts, where 8 segments serve as training data while the remaining 2 segments are utilized for testing. This procedure is iterated ten times across different folds, and the resultant outcomes are averaged to provide a comprehensive evaluation.

Next, we proceed with preprocessing the dataset to prepare it for model training.

Firstly, we identify categorical and numerical features within the dataset. Categorical attributes such as 'Sex', 'ChestPainType', and 'RestingECG', among others, are distinguished from numerical attributes like 'Age' and 'Cholesterol'.

Subsequently, we utilize the OneHotEncoder tool from the scikit-learn library to transform categorical features into a numerical format compatible with machine learning algorithms. This transformation is pivotal, as it enables the algorithm to interpret categorical data effectively. The OneHotEncoder method converts categorical variables into binary vectors, facilitating accurate interpretation by the model.

To address class imbalance, a common challenge in machine learning, we employ the Synthetic Minority Over-sampling Technique (SMOTE). This technique synthesizes new samples for the minority class, enhancing the model's ability to generalize to unseen data and preventing bias towards the majority class.

Finally, we normalize the numerical features by scaling them to a range between 0 and 1. This normalization process ensures that all features contribute proportionally to the model's learning process and enhances the stability and convergence of machine learning algorithms.

In the pursuit of generating weightings assigned to training examples, our approach harnesses the power of Evolutionary Algorithms (EAs), a class of optimization techniques inspired by natural selection and genetic mechanisms. At the core of this methodology lie two fundamental genetic operations: mutation and crossover. These operations are executed on an elite population, initially comprised of high-performing solutions. The process unfolds iteratively, with the goal of crafting a new population that exhibits desirable traits and characteristics.

Mutation serves as the mechanism for introducing variability and diversity within the population. With a probability of 0.6 initially, mutations occur sporadically, infusing individual solutions with random changes. This stochastic process allows the population to explore different regions of the solution space, potentially uncovering novel and promising solutions. As the population grows, the mutation probability gradually decreases to 0.1, striking a balance between exploration and exploitation.

In parallel, crossover operates to blend genetic information from two parent solutions, generating offspring solutions with a combination of traits inherited from their parents. When mutation is not selected, crossover takes precedence. Randomly chosen parent solutions undergo a process of genetic recombination, producing offspring solutions that inherit genetic material from both parents. This process mirrors natural reproduction, facilitating the exchange of genetic material and the emergence of diverse offspring.

The iterative application of mutation and crossover leads to the generation of a new population, enriched with a diverse array of solutions that have undergone genetic adaptation. This population represents a pool of potential candidates, each carrying a unique set of genetic traits and characteristics. Through further evaluation and selection processes, these candidates are refined and honed, ultimately contributing to the optimization of weightings assigned to training examples. In essence, our EA-based approach leverages the principles of genetic operations to explore and

exploit the solution space, driving the evolution of weightings and enhancing the adaptability of the learning algorithm.

Then the weighted training examples are fed into a base-learning algorithm, such as an Artificial Neural Network (ANN), with a customizable architecture. The model architecture employed for training consists of a Sequential model from the TensorFlow library, meticulously designed to uncover intricate patterns inherent within the dataset.

Beginning with the input layer, normalization is applied using TensorFlow's Normalization layer. This step ensures uniform scaling of feature values, crucial for stabilizing the training process. The input layer is tailored to accommodate the number of features present in the dataset.

Moving deeper into the architecture, multiple densely connected layers, with 50, 32, 16, 8, and 4 neurons, respectively constitute the hidden layers. These layers are strategically configured to gradually distill and encode the dataset's rich information. Each hidden layer incorporates ReLU activation functions, enhancing the model's capacity to capture nonlinear relationships among features.

The hidden layers are meticulously crafted with diminishing neuron counts, promoting the extraction of progressively abstract features. This design principle fosters hierarchical feature representation, facilitating the model's ability to discern intricate patterns embedded within the data.

Culminating the architecture is the output layer, characterized by a single neuron equipped with a sigmoid activation function. This configuration is tailored for binary classification tasks, as it enables the model to output probabilities indicative of an input's likelihood of belonging to the positive class.

The selection of hyperparameters governing the architecture was methodically undertaken through iterative experimentation, leveraging trial-and-error to achieve optimal model performance. This meticulous approach ensured that the model architecture was finely tuned to suit the specific requirements of the task at hand.

Furthermore, to enhance model robustness and predictive performance, we implemented an ensemble of classifiers. This ensemble comprises a default of 100 members,

organized into 10 generations of populations. Each generation consists of 10 distinct classifiers, resulting in a total of 100 classifiers. Priority is assigned to different generations based on fitness values derived from average classifiers across generations.

During the training phase, we assign weights to the classifiers based on their respective generations' priorities. These weights are utilized in the ensembling process to combine the predictions of individual classifiers effectively. By prioritizing different generations, we aim to identify and leverage the most accurate and influential classifiers in the ensemble.

In the testing phase, the ensemble classifier aggregates the predictions of average classifiers from different generations, weighted by the respective generation weights obtained during training. This approach enables us to not only identify the most accurate classifiers but also prioritize the most important ones in the ensemble learning process.

To ensure robustness and reliability, the aforementioned techniques are repeated 40 times on the dataset, with varying training and test data splits. The ensemble classifier accuracy values obtained from these iterations are averaged to provide a final estimation, offering a comprehensive assessment of model performance.

4.4 Results and comparative analysis

In this section, we present the experimental results of our proposed approach and compare them with existing algorithms. We evaluate the performance of various machine learning models using metrics such as Test Accuracy, Precision, Recall, F1 Score, AUC Score, and MCC (Matthews Correlation Coefficient).

4.4.1 Weights Obtained for Each Layer

The weights obtained for each of the 10 layers during the training process are as follows:

```
[0.0749, 0.0889, 0.0962, 0.1021, 0.1041, 0.1064, 0.1066, 0.1069, 0.1069, 0.1069]
```

4.4.2 Performance Metrics of Various Models

Table 4.2 presents the performance metrics of different machine learning models on the dataset.

TABLE 4.2: Performance Metrics of Various Models

Model	Test Accuracy	Precision	Recall	F1 Score	AUC Score	MCC
Logistic Regression	0.917	0.916	0.917	0.916	0.917	0.833
Decision Tree	0.863	0.862	0.863	0.862	0.863	0.725
SVM	0.926	0.926	0.926	0.926	0.926	0.852
RandomForest	0.926	0.926	0.926	0.926	0.926	0.852
AdaBoost	0.881	0.882	0.881	0.881	0.881	0.763
ExtraTrees	0.902	0.901	0.902	0.901	0.902	0.803
XGboost	0.872	0.872	0.873	0.872	0.873	0.745

4.4.3 Performance Metrics of the Proposed Methodology

The proposed methodology achieved the following performance metrics:

TABLE 4.3: Performance Metrics of the Proposed Methodology

Model	Test Accuracy	Precision	Recall	F1 Score	AUC Score	MCC
Proposed Methodology	0.912	0.863	0.973	0.915	0.896	0.808

4.4.4 Inferences

From the experimental results presented in Table 4.2 and Table 4.3, the following inferences can be made:

- The proposed methodology achieved a Test Accuracy of 91.2%, indicating its effectiveness in correctly classifying instances.
- With a Precision of 86.3%, the proposed methodology demonstrates its ability to minimize false positives.
- The Recall of 97.3% suggests that the proposed methodology has a high capability to identify true positive instances.

- The F1 Score of 91.5% indicates a balanced performance between Precision and Recall, signifying the proposed methodology's robustness.
- AUC Score of 89.6% implies that the proposed methodology performs well across various threshold values for classification.
- The MCC value of 80.8% reflects a strong correlation between the predicted and actual classifications by the proposed methodology.

4.4.5 Performance Metrics Comparison

The following bar plot 4.1 illustrates the comparison of performance metrics across different machine learning models. Each model is represented by a group of bars, and each metric is displayed as a separate bar within the group. The metrics evaluated include Test Accuracy, Precision, Recall, F1 Score, AUC Score, and Matthews Correlation Coefficient (MCC).

The plot provides a visual overview of how each model performs across these metrics, allowing for easy comparison and identification of the top-performing models.

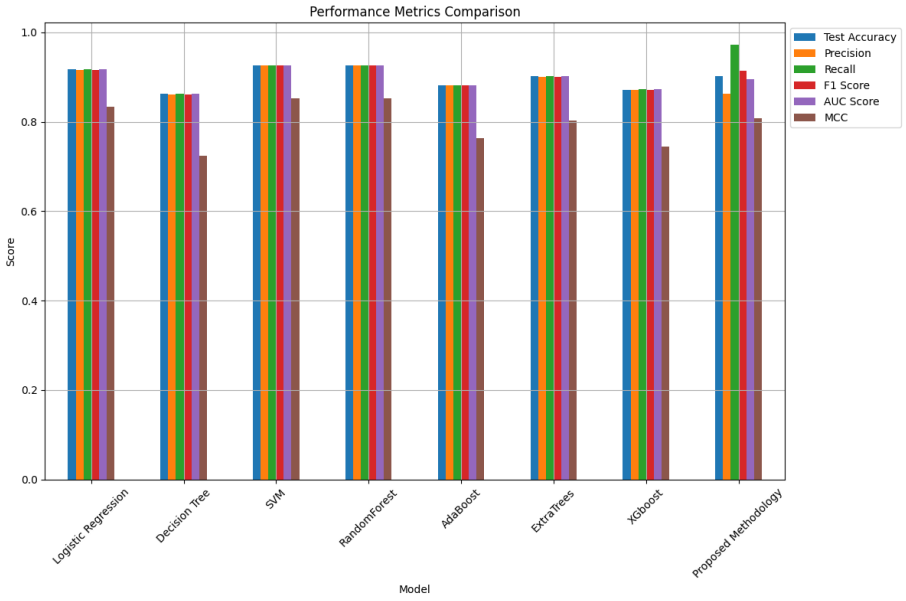

FIGURE 4.1: Bar plot comparing performance metrics across different models

As observed from the plot, the proposed methodology demonstrates competitive performance across various metrics compared to existing models, showcasing its effectiveness in addressing the problem at hand.

This box plot 4.2 illustrates the distribution of performance metrics (accuracy, precision, recall, F1 score, AUC score, and MCC) for various machine learning models, including Logistic Regression, Decision Tree, SVM, RandomForest, AdaBoost, ExtraTrees, XGBoost, and the Proposed Methodology. Each box represents the distribution of a specific metric, with the central line indicating the median value, the bottom and top edges of the box representing the first and third quartiles, and the whiskers showing the range of values within 1.5 times the interquartile range. Outliers, if present, are depicted as individual data points beyond this range.

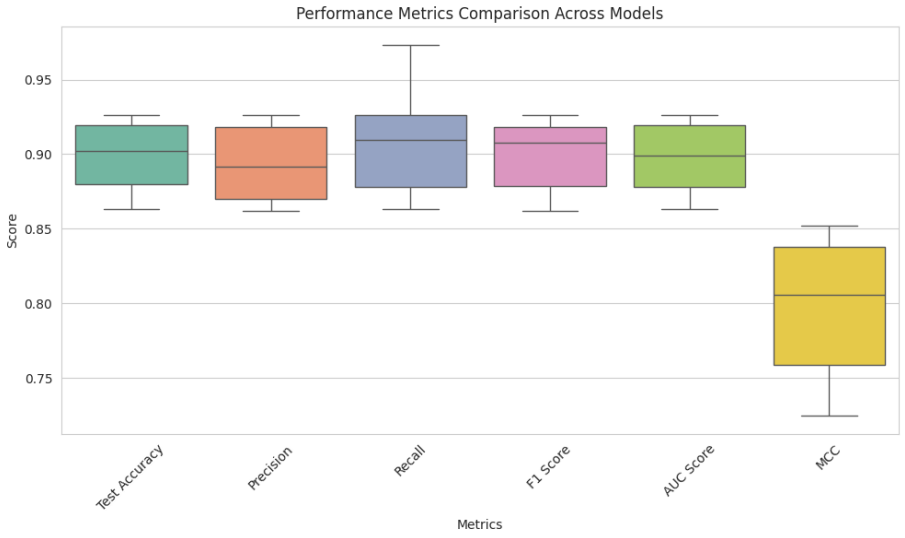

FIGURE 4.2: Box plot showing the distribution of performance metrics across different models.

4.4.6 Feature Importance Analysis using SHAP Values

Shapley values offer a comprehensive understanding of feature contributions to model predictions. In practice, Shapley values are estimated for individual instances to assess the impact of each feature on the model's output. For the sake of brevity and interpretability, we present in table 4.5 Shapley values for the first 5 instances and the first 15 features.

Table 4.4: SHAP Values for the First 5 Instances

Instance	1	2	3	4	5	6	7	8
1	0.0035	0.0024	0.0469	0.0702	0.0048	0.0093	0.0193	-.0027
2	0.0065	0.0058	-.1204	-.0224	0.0136	0.0377	0.0859	-.0100
3	-.0130	-.0061	-.0095	-.0512	-.0153	-.0463	-.1678	0.0082
4	0.0072	0.0004	0.1778	-.0183	0.0190	-.0252	0.0655	-.0103
5	0.0036	-.0010	0.0472	0.0671	0.0074	0.0212	0.0204	-.0025

Instance	9	10	11	12	13	14	15
1	0.0090	0.0067	0.0314	0.0032	-0.0063	-0.0178	0.0330
2	-0.0583	0.1239	0.1846	0.0505	0.2742	-0.0317	-0.0093
3	0.0151	-0.0950	0.0463	0.0021	-0.0564	-0.0158	0.0642
4	-0.0074	0.0783	0.0337	0.0300	-0.0310	0.0089	-0.0094
5	0.0093	0.0076	0.0331	0.0036	-0.0063	-0.0179	0.0312

The SHAP values offer valuable insights into the significance of various features in influencing the model's predictions. Presented below (table 4.5) are the average absolute SHAP values for each feature, accompanied by a corresponding bar plot (figure 4.3).

Table 4.5: Average Absolute SHAP Values for Each Feature

Feature	Average Absolute SHAP Value
Age	[0.0108081138863343]
RestingBP	[0.003454881859940456]
Cholesterol	[0.03778061930955585]
FastingBS	[0.05081283756257856]
MaxHR	[0.008782968232589419]
Oldpeak	[0.036485046217770094]
Sex_F	[0.06977414834801256]
Sex_M	[0.008934534727085525]
ChestPainType_ASY	[0.010469792897591269]
ChestPainType_ATA	[0.04806057694021399]
ChestPainType_NAP	[0.1104313659925543]
ChestPainType_TA	[0.01290918150835934]
RestingECG_LVH	[0.0599278172188007]
RestingECG_Normal	[0.013092280415920372]
RestingECG_ST	[0.02193916076116779]
ExerciseAngina_N	[0.029453245234782096]
ExerciseAngina_Y	[0.010625095103158464]
ST_Slope_Down	[0.012289820827793497]
ST_Slope_Flat	[0.08528479747723021]
ST_Slope_Up	[0.14959776867078883]

The local bar plot presented (figure 4.4) showcases the SHAP values attributed to each feature for a specific data instance, specifically instance 90. These SHAP values provide insights into the impact of individual features on the model's prediction for that particular instance. By examining the lengths of the bars in the plot, one can

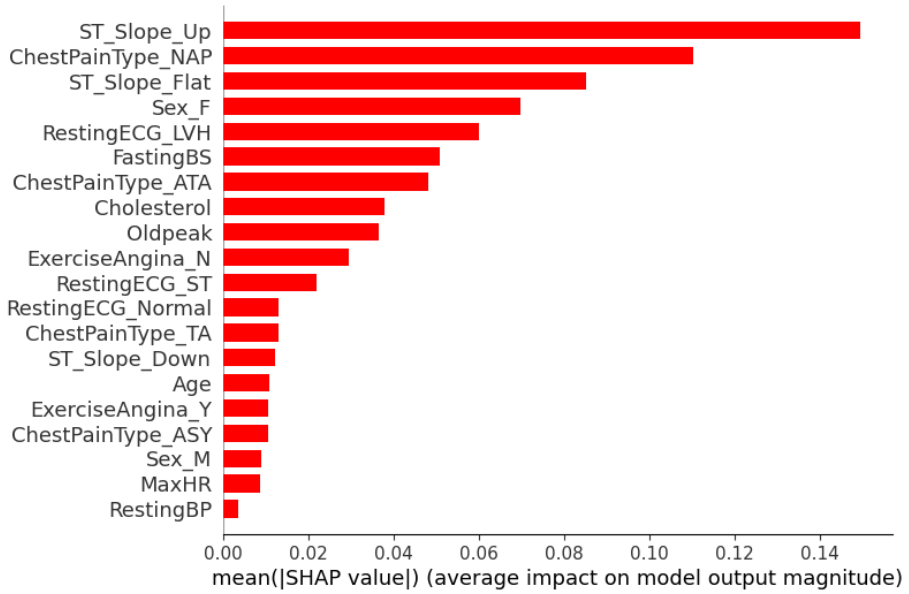

FIGURE 4.3: Average Absolute SHAP Values for Each Feature

discern the magnitude and direction of influence each feature has on the model's output for the chosen data point. Features with longer bars indicate a stronger influence on the prediction, with positive values contributing towards increasing the prediction while negative values indicate a decrease.

Analyzing this plot enables a deeper understanding of how each feature contributes

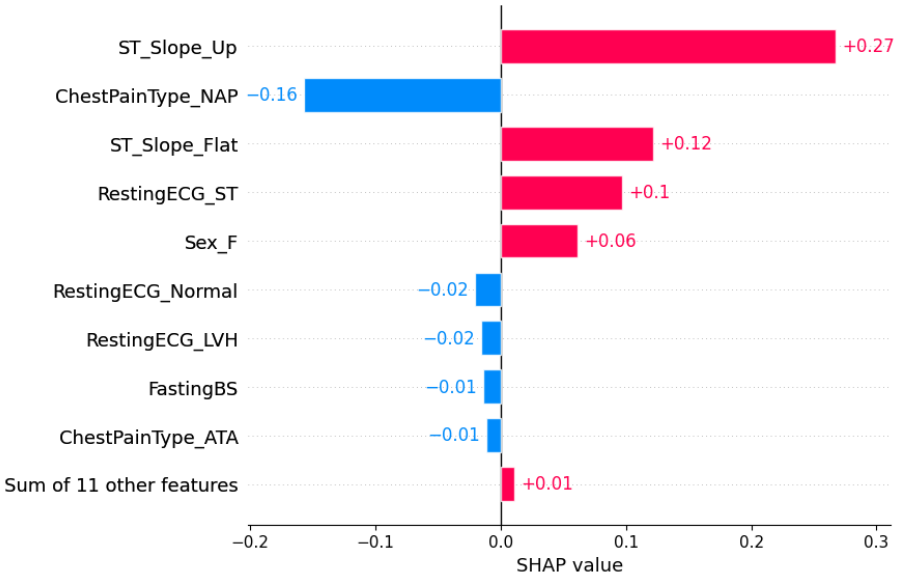

FIGURE 4.4: Local bar plot for instance 90

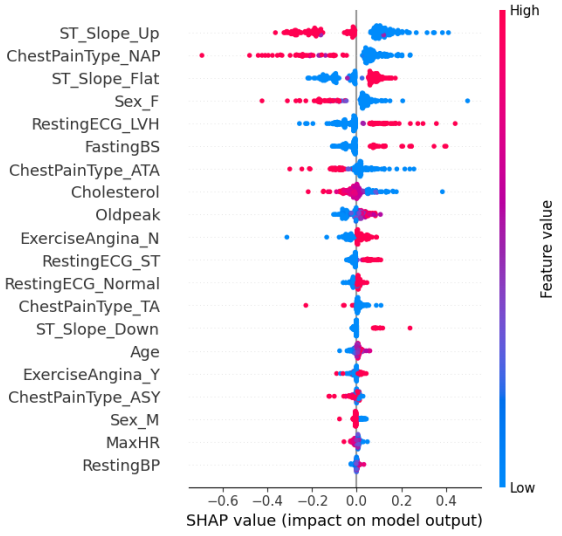

FIGURE 4.5: Beeswarm plot showing the distribution of SHAP values for each feature

to the model's decision-making process for the given instance, aiding in model interpretation and identifying key factors driving predictions.

Beeswarm plots are particularly effective in showcasing the density and spread of SHAP values for each feature, allowing for a granular understanding of their contributions. They provide a clear depiction of how individual SHAP values are distributed across different features, highlighting any outliers or clusters that may influence the model's prediction. On the other hand, violin plots offer a broader view of the distribution of SHAP values, presenting insights into their overall distributional characteristics, such as skewness and multimodality. Figure 4.5 , we

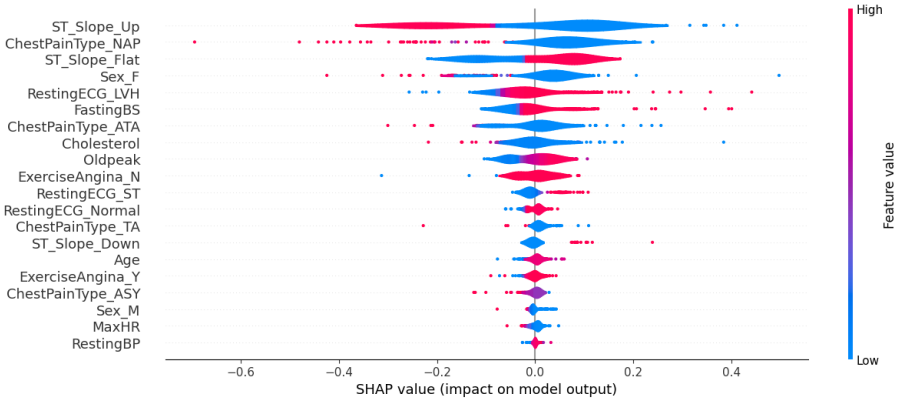

FIGURE 4.6: Violin plot illustrating the overall distributional characteristics of SHAP values across features

present the beeswarm plot depicting the distribution of SHAP values for each feature, while figure 4.6 , illustrates the violin plot showcasing the overall distributional characteristics of SHAP values across features. Together, these visualizations offer complementary insights into the importance and variability of features in driving model predictions at the instance level.

Chapter 5

Conclusion

In conclusion, this project introduces an innovative ensemble method that leverages evolutionary computation and prioritized aggregation to construct robust and reliable classifiers. By strategically assigning weightings to training examples and prioritizing generations of classifiers based on their performance, our approach enhances the predictive power of ensemble models. Through extensive experimental evaluations and comparative analysis against state-of-the-art methods, we have demonstrated the effectiveness and superiority of our proposed techniques. Additionally, the integration of SHAP analysis enhances the interpretability and trustworthiness of the ensemble model's predictions, making it well-suited for real-world applications. Overall, our research contributes to the advancement of ensemble learning methodologies and offers valuable insights for future developments in the field.

5.1 Limitations

While our proposed ensemble method shows promising results and offers several advantages, it is not without limitations. These limitations include:

1. **Computational Complexity:** The evolutionary computation-based approach involves iterative training and optimization, which can be computationally intensive, especially for large datasets or complex models. Additionally, due to

computational constraints, we were limited to generating up to 20 generations when using the free version of Google Colab.

2. **Sensitivity to Hyperparameters:** The performance of the ensemble method may be sensitive to the choice of hyperparameters, such as the population size, crossover and mutation rates, and the number of generations. Tuning these hyperparameters is essential for optimizing performance; however, due to the high computational complexity involved, exhaustive hyperparameter tuning was not feasible within the scope of this project.

3. **Interpretability:** Although SHAP analysis enhances the interpretability of the ensemble model, understanding the combined effects of multiple classifiers and their prioritized contributions may still be challenging for users without expertise in machine learning.

4. **Generalization:** While the ensemble method performs well on benchmark datasets, its generalization to unseen or real-world datasets may vary, and its effectiveness may depend on the characteristics of the data and the problem domain.

5. **Scalability:** Scaling the ensemble method to larger datasets or more complex models may pose challenges in terms of memory and computational resources, potentially limiting its applicability to certain scenarios.

Addressing these limitations could be a focus of future research efforts aimed at further improving the efficiency, robustness, and applicability of ensemble learning techniques.

5.2 Future Scope

- **Scalability Enhancement:** Investigating methods to improve the scalability of the proposed ensemble method to handle larger datasets and more complex models. This could involve optimizing algorithms and data structures to reduce computational overhead and memory requirements.

- **Hyperparameter Optimization:** Conducting comprehensive hyperparameter tuning experiments to identify the optimal configuration for the ensemble method. Advanced techniques such as automated hyperparameter optimization using Bayesian optimization or genetic algorithms could be explored to efficiently search the hyperparameter space.

- **Evaluation on Unbalanced Datasets:** Evaluating the performance of the ensemble method on datasets with class imbalance or skewed distributions. Investigating techniques to handle class imbalance effectively and mitigate potential biases in predictions could enhance the reliability and fairness of the ensemble model.

Appendix A

1. TensorFlow
2. Keras
3. Seaborn
4. Scikit-learn
5. shap
6. Matplotlib
7. Pandas
8. NumPy

A.1 Important Links

1. Link to GitHub repository (https://github.com/atharv825/Enhancing-Classifier-Performance-with-Evolutionary-Weighting-and-Priority-Based-Aggregation)

Bibliography

Atallah, R. and Al-Mousa, A. (2019). Heart disease detection using machine learning majority voting ensemble method. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pages 1–6.

Beliakov, G., James, S., Mordelová, J., Rückschlossová, T., and Yager, R. R. (2010). Generalized bonferroni mean operators in multi-criteria aggregation. *Fuzzy Sets Syst.*, 161:2227–2242.

Grabisch, M., Marichal, J.-L., Mesiar, R., and Pap, E. (2009). Aggregation functions (encyclopedia of mathematics and its applications).

Li, B. and Xu, Z. (2019). Prioritized aggregation operators based on the priority degrees in multicriteria decision-making. *International Journal of Intelligent Systems*, 34:1985 – 2018.

Lin, X., Yacoub, S. M., Burns, J., and Simske, S. J. (2003). Performance analysis of pattern classifier combination by plurality voting. *Pattern Recognit. Lett.*, 24:1959–1969.

Liu, H., Du, Y., and Wu, Z. (2019). Aem: Attentional ensemble model for personalized classifier weight learning. *Pattern Recognition*, 96:106976.

Rokach, L. (2010). Ensemble-based classifiers. *Artif. Intell. Rev.*, 33:1–39.

Tripathi, D., Shukla, A. K., Reddy, B. R., Bopche, G. S., and Chandramohan, D. (2021). Credit scoring models using ensemble learning and classification approaches: A comprehensive survey. *Wireless Personal Communications*, 123:785 – 812.

Yager, R. R. (2008). Prioritized aggregation operators. *Int. J. Approx. Reason.*, 48:263–274.