UNBALANCED DATA CLASSIFICATION USING GENETIC PROGRAMMING

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

By

ARVIND KUMAR (E18SOE822)

Under the Supervision of

Dr. Shivani Goel Dr. Nishant



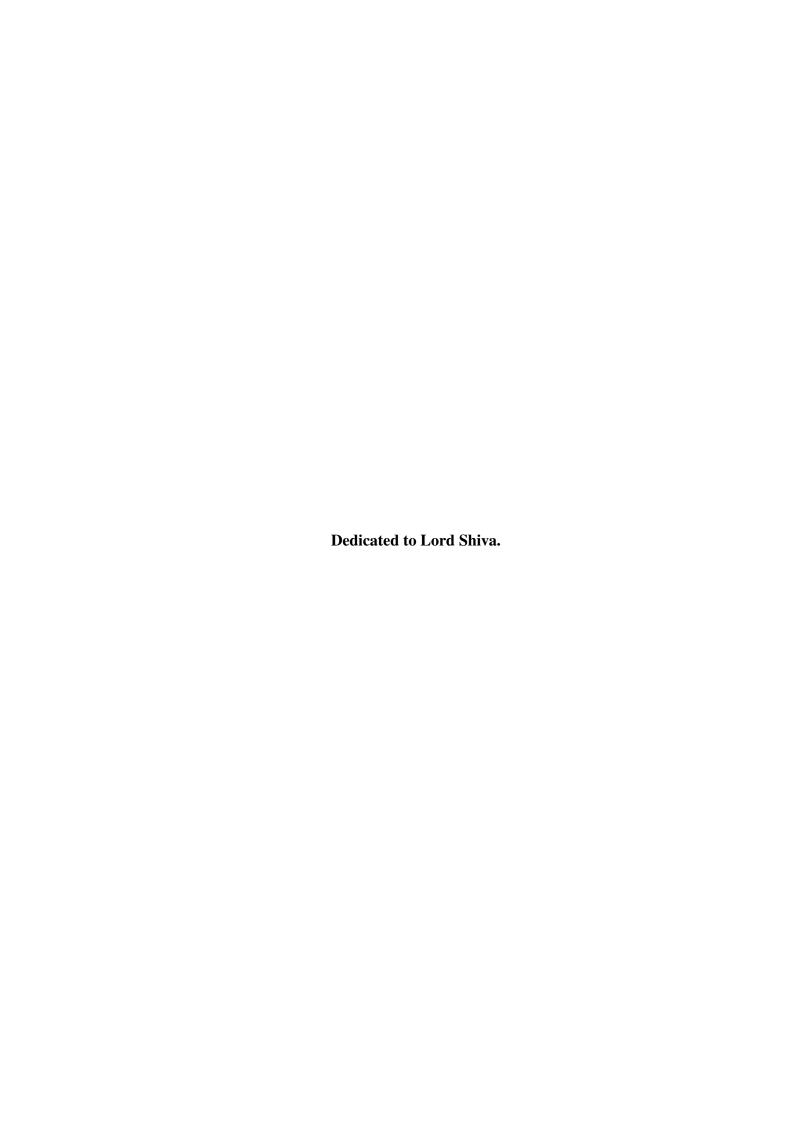
School of Computer Science Engineering and Technology

BENNETT UNIVERSITY

(Established under UP Act No 24, 2016)

Plot Nos 8-11, Tech Zone II, Greater Noida-201310, Uttar Pradesh, India

June, 2022



@ Copyright Bennett University, Greater Noida, India June, 2022 ALL RIGHTS RESERVED

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the Ph.D. thesis entitled "UNBALANCED DATA CLASSIFICATION USING GENETIC PROGRAMMING" submitted at Bennett University, Greater Noida, India is an authentic record of my work carried out under the supervision of **Dr. Shivani Goel** and **Dr. Nishant**. I have not submitted this work elsewhere for any other degree or diploma. I am fully responsible for the contents of my Ph.D. thesis.



Arvind Kumar

Enrollment No. E18SOE822

School of Computer Science Engineering and Technology

Bennett University, Greater Noida, India

Date: 29/05/2022



CERTIFICATE

This is to certify that the work reported in the Ph.D. thesis entitled "UNBALANCED DATA CLASSIFICATION USING GENETIC PROGRAMMING" submitted by Arvind Kumar at Bennett University, Greater Noida, India is a bonafide record of his original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

Shinga God,

Nishami

Dr. Shivani Goel

Professor, School of Computer Science Engineering and Technology

Bennett University, Greater Noida, India

Dr. Nishant

Product Head - Times Internet Limited, Noida, India

(former) Director of Engineering, Pitney Bowes (Precisely) Software, Noida, India

Place: Bennett University

Date: 30-05-2022

(Head of the Department)

Deepay way

(Office Seal)

GREATER NOIDA

ACKNOWLEDGEMENTS

Throughout this research work and writing this dissertation, I have received a great deal of support and assistance from many people; this dissertation would not be complete without acknowledging them.

First of all, I want to thank my esteemed, benevolent mentors, Dr. Shivani Goel, Dr. Nishant, and Dr. Arpit Bhardwaj, for their guidance, constant encouragement, inspiration over the past four years, and constructive feedback in writing research articles and this thesis. You have provided extensive personal and professional guidance and taught me a great deal about scientific research and life in general.

Thank my RAC committee members Dr. Deepak Garg, Dr. Neelanchali Bhalla, Dr. Deepika Vats, Dr. Deepak Singh, Dr. Suyel Namasudra, and Dr. Rishav Singh, for their guidance, encouragement and positive criticism throughout this research journey. Your insightful feedback made me sharpen my thinking and brought my work to a higher level. Dr. Deepak Garg and Dr. Arpit Bhardwaj were the first persons at Bennett University whom I met even a long before starting this journey, and I made my mind to be a scholar at Bennett University.

I also want to thank my company, Pitney Bowes (Precisely) Software India, for being so generous and sponsoring this Ph.D. program. My sincere Thanks to Monika Narula (Director, Human Resources) for encouragement, moral support, and facilitating organizational sponsorship for this study.

I also like to thank my friend and colleague, Mr. Tejalal Chaodhary (Research-scholar at Bennett University), for providing stimulating discussions, moral support and encouragement through this journey.

Last but not least, I want to give special thanks to my family, for their encouragement, without whose moral support I never imagine completing this research work. Most importantly, I wish to thank my loving and supportive wife, Anuradha, my son Aditya, and my daughter Srishti, who provide unending inspiration.

Mkumar (Arvind Kumar)

Contents

Al	bstrac	t				i
Li	st of I	Publicati	ions			iii
Li	st of T	Tables				v
Li	st of I	Figures				vi
Ll	IST O	F ACRO	ONYMS & ABBREVIATIONS			viii
1	INT	RODUC	CTION			1
	1.1	MOTIV	ATION			3
	1.2	RESEA	ARCH OBJECTIVES			5
	1.3	THESI	S OUTLINE			6
	1.4	BENCI	HMARK PROBLEMS			6
	1.5	PERFO	DRMANCE EVALUATION METRICS			9
2	REV	IEW O	F LITERATURE			13
	2.1	MACH	INE LEARNING			14
		2.1.1	Classification			15
		2.1.2	K Nearest Neighborhood (KNN)			15
		2.1.3	Support Vector Machine (SVM)			17
	2.2	EVOLU	UTIONARY COMPUTING			18
	2.3	GENE	ΓΙC PROGRAMMING			19
		2.3.1	Initialization of population			20
		2.3.2	Standard GP Fitness Function for Classification			21
		2.3.3	Reproduction			23

		2.3.4 Crossover	23
		2.3.5 Mutation	24
		2.3.6 Termination Criteria	24
		2.3.7 Classification using GP	24
	2.4	UNBALANCED DATA CLASSIFICATION	26
		2.4.1 Challenges of unbalanced data	27
		2.4.2 Handling Unbalanced Data Classification	28
		2.4.3 Data level approaches	28
		2.4.4 Algorithm level approaches	30
		2.4.5 Ensemble of classifiers	33
	2.5	SUMMARY	35
3	ASS	ESSMENT OF WEIGHT FACTOR IN GP's FITNESS FUNCTION FOR UN-	
	BAL	ANCED DATA CLASSIFICATION	36
	3.1	INTRODUCTION	37
	3.2	PROPOSED METHODOLOGY	38
	3.3	EXPERIMENTAL SETUP	38
	3.4	RESULTS	40
	3.5	SUMMARY	41
4	AN]	EUCLIDEAN DISTANCE AND WEIGHT-BASED FITNESS FUNCTION IN	
	GP I	FOR UNBALANCED DATA CLASSIFICATION	42
	4.1	INTRODUCTION	43
	4.2	PROPOSED METHODOLOGY	44
	4.3	EXPERIMENTAL SETUP	46
	4.4	RESULTS	49
	4.5	SUMMARY	50
5	MU	LTI-OBJECTIVE GP APPROACH FOR CLASSIFICATION OF UNBALANC	ED
	DAT	'A	52
	5.1	INTRODUCTION	53
	5.2	MULTI-OBJECTIVE OPTIMIZATION	54
	5.3	NON-DOMINATED SORTING GENETIC ALGORITHM II (NSGA-II)	57

RI	REFERENCES					
6	SUN	IMARY AND CONCLUSIONS	71			
	5.7	SUMMARY	70			
	5.6	RESULTS	67			
	5.5	EXPERIMENTAL SETUP	66			
	5.4	PROPOSED METHODOLOGY	61			

Abstract

In many real-world classification applications, such as medical diagnosis, fraud detection, bioinformatics, or fault diagnostics, it is common that one class has only a limited number of
training instances (called the minority class), while the other class (called the majority class)
conceive the rest. Such types of data sets are called unbalanced. In data classification, machine
learning (ML) methods can face a performance bias when the nature of data sets is unbalanced.
In this case, the trained classifiers may have good accuracy on the majority class but lower
accuracy on the minority class. Genetic Programming (GP) is an optimistic machine learning
method based on the Darwinian theory of evolution to automatically emerge computer programs
to solve problems without any domain-specific knowledge. Although GP has revealed much
success in developing reliable and precise classifiers for typical classification jobs, GP, like
many other ML algorithms, can produce biased classifiers when the nature of data is unbalanced.
This biasing is because traditional training standards such as the overall success rate in the
fitness function in GP can be influenced by the more significant number of instances from the
majority class.

This research focuses on algorithmic methods assuming that the whole training data is important and valuable, and no data sample should be removed from the training process. The second consideration in this work is that the proposed methods should be problem-independent, and they should not expect any a-priori domain-specific or expert knowledge. Thus, this research focuses on developing GP-based approaches for unbalanced data-set classification, based on internal cost alteration in the GP fitness function and facilitating the unbalanced data set to be used "as is" in the training process. This research work demonstrates that by designing various methods in GP, we can evolve classifiers with good classification performance on the majority and the minority classes. These developed methods are evaluated, on publicly available, UCI-based binary benchmark classification problems with varying levels of imbalanced factors.

Despite many methods in literature which focus on balancing training sample counts of the minority and majority class by over-sampling of the minority class samples or under-sampling of the majority class samples, this research work evaluates the impact of custom learning weight assignment to the minority class samples while training. This research work uses the distance between the predicted and the actual outcomes of the classifier, and then combines the custom weight factor and the distance factor for designing new fitness functions in GP to handle unbalanced data classification. The proposed method performance is compared with standard GP and other classification techniques, namely support vector machine (SVM) and K-nearest neighbourhood (KNN) algorithms. This research work also focuses on developing an Evolutionary multi-objective optimization (EMO) approach by considering minority and majority class accuracies as distinct conflicting objectives. The developed method is evaluated on the quality of generated Pareto-front with reference to the standard MOGP approach.

LIST OF PUBLICATIONS

INTERNATIONAL JOURNALS

- 1. *A. Kumar, N. Sinha, and A. Bhardwaj*, "A novel fitness function in genetic programming for medical data classification." *Journal of Biomedical Informatics*, 112 (2020): 103623. https://doi.org/10.1016/j.jbi.2020.103623. (Elsevier, SCIE, IF: 6.317)
- 2. A. Kumar, N. Sinha, A. Bhardwaj, and S. Goel, "Clinical risk assessment of chronic kidney disease patients using genetic programming." *Computer Methods in Biomechanics and Biomedical Engineering*, (2021). https://doi.org/10.1080/10255842.2021.1985476. (Taylor and Francis, SCIE, IF: 1.763)

INTERNATIONAL CONFERENCES

- A. Kumar, N. Sinha, and A. Bhardwaj, "Predicting the presence of newt-amphibian using genetic programming". Advances in Data and Information Sciences, Lecture Notes in Networks and Systems. [Springer's 3rd International Conference on Data and Information Sciences (ICDIS 2021) RBS Engineering Technical Campus, Agra (India), May 14-15, 2021]. https://doi.org/10.1007/978 981 16 5689 7_19.
- 2. A. Kumar, S. Goel, N. Sinha, and A. Bhardwaj, "Assessment of weight factor in genetic programming fitness function for imbalanced data classification". *Proceedings in Adaptation, Learning and Optimization*. [Springer's International Conference on Intelligent Vision and Computing (ICIVC 2021), Sur University, Oman, October 3-4, 2021]. https://doi.org/10.1007/978-3-030-97196-0_1.
- 3. A. Kumar, S. Goel, N. Sinha, and A. Bhardwaj, "A Review on Unbalanced Data Classification". Algorithms for Intelligent Systems. [Springer's 5th International Joint Con-

ference on Advances in Computational Intelligence (IJCACI 2021) – Jahangirnagar University, Bangladesh and South Asian University, India, October 23-24, 2021]. https: $//doi.org/10.1007/978 - 981 - 19 - 0332 - 8_14$.

4. *A. Kumar, S. Goel, N. Sinha, and A. Bhardwaj*, "A logarithmic distance-based multi-objective genetic programming approach for classification of imbalanced data". *Communications in Computer and Information Science*. [Springer's 11th International Advanced Computing Conference (IACC 2021), University of Malta, Malta, December 18-19, 2021]. *https://doi.org/10.1007/978-3-030-95502-1_23*.

List of Tables

1.1	Summary of Unbalanced Benchmark Problems Used	7
1.2	Binary Classification Outcomes	10
2.1	Evolutionary Computing vs. Problem Solving	19
3.1	GP Parameter Setting For Minority Class Weight Assessment	39
3.2	Custom Minority Class Weight vs Produced AUC Values	40
4.1	GP Parameter Setting for Proposed Method	48
4.2	Proposed GP Method Performance Evaluation	49
4.3	KNN Grid Test Result for AUC values	51
5.1	MOGP Parameter Setting	66
5.2	Generated Pareto-front Hyper-area by Proposed MOGP Method	67

List of Figures

1.1	Abalone data set	7
1.2	YEAST-1 data set	8
1.3	YEAST-2 data set	8
1.4	CKD data set	9
1.5	WDBC data set	10
2.1	KNN Illustration for 2 Features and K=5	16
2.2	SVM Illustration of a 2-d Hyper-plane, Margin, Support-vectors	18
2.3	GP Framework	20
2.4	Program's Tree Representation	21
2.5	Crossover Between Two Parents	23
2.6	Mutation of a Program	24
2.7	Example of a Three-class Classification Problem	29
2.8	Illustration of Boundary Samples	30
2.9	Illustration of Data Sampling Methods	31
3.1	Custom Minority Class Weight vs Produced AUC Values	41
4.1	Illustration of Classifier Performance Evaluation	45
5.1	Pareto-front for a MOO Problem with 2 Objective	54
5.2	Dominance Rank	55
5.3	Dominance Count	56
5.4	Calculation of Crowding-distance	59
5.5	NSGA-II Procedure	62
5.6	Fuclidean-distance vs Logarithmic-distance for Different Data-samples	64

5.7	Generated Pareto-front by MOGP for ABL-9-18	68
5.8	Generated Pareto-front by MOGP for YEAST-2	68
5.9	Generated Pareto-front by MOGP for YEAST-1	69
5.10	Generated Pareto-front by MOGP for WDBC	69

LIST OF ACRONYMS & ABBREVIATIONS

ABL Abalone

AI Artificial Intelligence

ANN Artificial Neural Network

CKD Chronic Kidney Disease

DBN Deep Belief Network

DL Deep Learning

DYS Dynamic Sampling

EC Evolutionary Computing

ECS-DBN Evolutionary Cost-Sensitive Deep Belief Network

EHS Entropy-Based Hybrid Sampling

EID Entropy-Based Imbalance Degree

EIG-GA Entropy And Information Gain Based Genetic Algorithm

ELM Extreme Learning Machine

EMO Evolutionary Multi-Objective Optimization

EMOSAIC Ensemble of Classifiers Based on the Multi-Objective Genetic Sampling for

Imbalanced Classification

EOS Entropy-Based Oversampling

EMO Evolutionary Multi-objective Optimization

EP Evolutionary Programming

ES Evolutionary Strategies

EUS Entropy-Based Under Sampling

FN False Negative

FP False Positive

GA Genetic Algorithm

G-MEAN Geometric Mean

GONN Genetically Optimized Neural Network

GP Genetic Programming

IR Imbalance Ratio

KEEL Knowledge Extraction Based On Evolutionary Learning

KNN K nearest neighborhood

MAUC Multi-Class Area Under The Curve

ML Machine Learning

MLP Multi-Layer Perceptron

MLP Machine Learning

MOGA Multi-Objective Genetic Algorithm

MOGP Multi-Objective Genetic Programming

MWMOTE Majority Weighted Minority Oversampling Technique

NN Neural Network

NOS No Sampling

NSGA Non Dominated Sorting Genetic Algorithm

NPGA Niche Pareto Genetic Algorithm

ROS Random Over Sampling

RUS Random Under Sampling

SPEA Strength Pareto Evolutionary Algorithm

TN True Negative

TP True Positive

SVM Support Vector Machine

VEGA Vector-Evaluated Genetic Algorithm

WDBC Wisconsin Diagnostic Breast Cancer

WELM Weighted Extreme Learning Machine

CHAPTER 1 INTRODUCTION

Data classification is the process of sorting and categorizing data into various distinct classes or distinct groups such that we can assign labels to each class. There is a high demand for methods that can automatically search and identify useful and valid patterns in data given for classification, with as little human interference as possible in real-world applications. Aim of the classification is to map an input vector $X = [x1, ..., x_n]$ to one of the K classes C_k , where k=1,2...K. A classifier, also called the model, is trained to perform this mapping of X to K [93].

The high availability of digital information in the real world creates demand for designing mathematical models, which automatically investigate this data and determine applicable and valid patterns in data for classification. The desirable characteristics of such models are that the model should be fast enough, requiring limited human intervention while applying those models in real-time. Another desirable characteristic is that very minimal or no domain expertise should be required while applying these models [27, 89, 118]. However, designing intelligent learning systems for real-world applications that perform classification reliably and with enough accuracy is difficult [132]. Genetic Programming (GP) is an evolutionary machine learning (ML) search and optimization technique successfully applied for creating reliable classifiers to resolve classification problems. In GP, various solutions to a given problem, represented as programs, are combined with other programs to produce the new and hopefully improved programs over a generation by generation until a reasonable and acceptable solution is evolved.

In many real-world classification applications, such as bio-informatics [3, 20], fraud detection [35, 116], medical diagnosis [39, 66, 115, 147], or fault diagnostics [44, 76, 112], it is common to have an unequal number of training examples in one class against to the other class(es). This characteristic is said as unbalanced-class or Unbalanced class. A data set is called unbalanced or Unbalanced if at least one class has limited examples, and the class with a smaller number of samples is called the minority class. Similarly, a class having a more significant number of data samples is called the majority class [132, 68]. Multiple state-of-the-artwork has been done in the literature to handle unbalanced data classifications, and proposing new methods is always an interest of researchers.

1.1 MOTIVATION

In the real world, most data sets are unbalanced [115]. The quality and ability of the classifier is highly impacted by the nature of data used to train the classifier [54]. Therefore in the case of unbalanced nature, the ability of the classifier may also be unbalanced. Most of the classifiers show biased or lower accuracy on unbalanced data [46]. Recent research works in the literature show that this Unbalanced nature of data misleads classification performance [16]. As there are fewer examples in the minority class for training and testing, the weight of these data on model accuracy will also be limited. Thus, for the Unbalanced data set, higher accuracy is not necessarily an indicator of its goodness. In fact, in most cases, the correct classification of the minority class is more needed than the majority class. For example, the fraud transaction case (minority class here) is even less than 1% in fraud transaction data. However, this minority class is of more interest to any user here. In this case, a classifier may have 98% accuracy, but it still fails to detect any fraud transactions. This point needs to be considered when we compare the accuracies of classifiers.

In the case of the real-world application, the primary interest of the class is the minority class. Therefore in such classification problems, classifying samples belonging to the minority class is similar or, in some scenarios, more crucial than correctly classifying samples from the majority class. Therefore resolving this learning bias to accurately performing classification of samples belonging to both classes, i.e., the minority and the majority classes in an equal and similar manner, has become a vital domain of research. For handling the unbalanced data classification issues, multiple approaches have been proposed by various researchers. These approaches may be categorized into three groups: data level, the ensemble of classifiers, and the algorithm level. The data level approach focus on balancing a trade-off between the minority and the majority class samples count either by under-sampling of the majority class data points or over-sampling of the minority class data points. Ensemble-based techniques focus on adopting the technology of multiple classifiers to improve the performance of a single classifier and predict. Here, voting of classifier's results or averaging all classifier results is done to give the final decision. Although data-level approaches can handle the issue of unbalanced data classification, there are a few significant downsides of these methods. These techniques take a lot of time and suffer from the loss of critical information contained samples during the

under-sampling of unbalanced data [142]. Similarly, the under-sampling of the instances, belongs to the majority class may remove some essential training data, and the model becomes under-fitted. Thus under sampling-based techniques may create a loss of valuable information [74]. On another side, an eminent problem with these ensemble approaches is that they are computationally intensive [144]. Also, both methods require domain expertise in the respective area of dataset [15].

Due to these limitations, ML researchers have focused on the algorithm level approach (also called internal approaches) using cost adjustment in the learning method. Here focus is on either modifying existing algorithms or proposing new algorithms to overcome the issue of unbalanced data classification [141, 15].

One approach to resolve this learning issue is to assign a custom high weight to the classes during training. This custom weight assignment may nullify the impact of higher counts of any classes during the learning phase of the classifier. Numerous works in literature, based on the criteria of assigning custom weight to the majority and minority classes, are done to improve the classification performance [9, 82, 88, 126]. Although the proper adjustment of weights during the training phase improves the model classification performance, assigning these weights can require a priori domain knowledge [15].

Also, several techniques are proposed in the literature based on the distance between the predicted and the actual outcomes [15, 21, 34]. GP, inspired by the Darwinian theory of evolution, is emerged as a prominent field to solve different research problems [72]. In GP, many researchers focused attention on adapting the new fitness function to reward solutions that are accurate on both the majority and the minority classes [15, 100]. These techniques give attention to developing new performance criteria in the fitness function, which can develop solutions with good classification ability in both classes without incurring an ample increase in training times and should be problem-independent.

Evolutionary multi-objective optimization (EMO) is also gaining popularity in research which facilitates solutions to learning with multiple conflicting objectives [37]. EMO achieves this by treating the goals independently in the learning procedure using the notion of the Pareto Dominance concept in fitness [16, 29, 63]. The main advantage of the Multi-Objective GP (MOGP) approach is that it can produce a set of good-performing classifiers in a single experimental execution. Against the MOGP approach, the canonical GP approach requires multiple

experimental runs and a priori objective-based fitness function. Another advantage of MOGP is that it explicitly includes the learning bias into the algorithms. These works are another motivation for this research work.

Thus allowing the unbalanced data to be used "as is" and adapting the diversity measures in the form of GP fitness function is the primary motivation behind this research work.

1.2 RESEARCH OBJECTIVES

To address the above said problem statement, this research is organized with the following five objectives.

- 1. To investigate and improve unbalanced data classification methods based on GP.
- 2. Develop new internal cost-adjustment techniques in GP (also called fitness functions) through balancing the weighted accuracies between the majority and minority class.
- 3. Develop new fitness functions in GP through optimizing the distance between predicted and actual outcomes.
- 4. Develop new fitness functions for GP to improve classification performances based on multi-objective evolutionary optimization approaches.
- 5. Comparative analysis of proposed methods with other state-of-the-art methods available in the literature.

By concentrating on internal cost alteration in the GP fitness function, this research work facilitates the utilization of the unbalanced data set to be used "as is" in the training process. Also, it would not require any data sampling based on external data-balancing methods by artificial means to adjust sample counts in the input data before applying the learning mechanism. This research work focuses on algorithmic methods with below important criteria:

1. Whole training data is important and valuable. Therefore no data sample should not be removed from the training process.

2. The proposed methods should be problem-independent, and they should not expect any a-priori domain-specific or expert knowledge.

1.3 THESIS OUTLINE

The remaining part of the thesis is organized as follows. Chapter 2 contains a brief background and literature review related to unbalanced data classification. The contribution work chapters, Chapters 3–5, address research objectives in this thesis; and Chapter 6 summarizes and concludes the thesis.

Chapter 2 has two distinct sections: background materials and related work. The background part includes the three substances: machine learning (ML), evolutionary computing, and genetic programming (GP). The second part gives a detailed literature review of work related to unbalanced data classification. This chapter also discusses the limitations of the present approaches in the literature and the challenges this research work attempts to handle. Chapter 3 proposes the use of the custom learning weight assignment to the minority class samples while training the GP-based classifiers. Chapter 4 develops a new fitness function based on the distance among the predicted and the actual outcomes of the classifier, and the weight adjustment of the minority class highlighted in Chapter 3. Chapter 5 proposes a MOGP method based on the accuracies of the majority and the minority classes as the two competing objectives, with emphasis on a better Pareto-front. Chapter 6 concludes the thesis by summarizing the work and research objectives achieved in the individual chapters.

1.4 BENCHMARK PROBLEMS

For evaluation and performance measurement, five unbalanced problems, with various levels of the Unbalanced factor, are taken. These Unbalanced classification problems are based on the publicly available UCI data set repository [42]. This Unbalanced factor varies from 6:94 to 37:63. Table-1.1 shows a summary of these data-sets. These Unbalanced classification problems are briefly described as follows:

Table 1.1: Summary of Unbalanced Benchmark Problems Used

Data-Set	Unbalanced	Features	Minority	Majority	Total	Data-Set
Name	Ratio	Count	Count	Count	Count	Description
ABL-9-18	06 : 94	8	42	689	731	Abalone (class 9 vs 18)
YEAST-2	11:89	8	163	1321	1484	Yeast (ME3 vs all)
YEAST-1	16 : 84	8	244	1240	1484	Yeast (MIT vs all)
CKD	27:73	24	53	115	168	Chronic kidney disease
WDBC	37 : 63	31	212	357	569	Wisconsin diagnostic breast cancer

Abalone (ABL-9-18)

The abalone dataset is a popular Unbalanced real-world problem used in various research works [50, 67]. In this benchmark problem, a classifier is evolved to predict the age of abalone shell-fish. The abalone data-set has eight attributes and a multi-class classification problem. We have converted this to a binary class classification problem by selecting class 9 vs class 9 plus class 18. Thus out of 731 total samples, 42 samples (6%) belong to the minority class. Fig. 1.1 shows a summary of this data-set.



Figure 1.1: Abalone data set

YEAST-1

The yeast data set has 1484 samples. The objective is to predict the protein localization sites in yeast cells. Based on the properties of amino acid sequences, there are eight input features. There are nine classes in this data set, which makes it a multi-class classification problem. We

converted this multi-class problem to a binary class classification problem by considering only one class (the minority class) and the rest as the majority class samples. Two sets of data set, YEAST-1 and YEAST-2, are generated. In YEAST-1, data set is generated by considering MIT (Mitochondrial) versus all classes samples. Thus YEAST-1 data set has 16% of minority samples count. Fig. 1.2 shows a summary of this dataset.

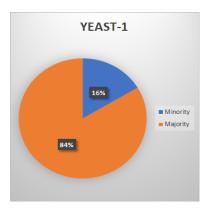


Figure 1.2: YEAST-1 data set

YEAST-2

The YEAST-2 classification problem is also generated using Yeast data set by considering ME3 (Membrane protein, no N-terminal signal) versus all classes samples. Thus YEAST-2 data set has 11% of minority samples count. Fig. 1.3 shows a summary of this dataset.



Figure 1.3: YEAST-2 data set

Chronic kidney disease (CKD)

The chronic kidney disease (CKD) data set has 24 attributes. This clinical data were collected by a doctor (nephrologist) *Soundarapandian et al.* at Apollo Hospital, Karai-kudi, Tamilnadu (India), in 2015. The information is collected in two months duration. Out of 158 samples, 73% of the data belongs to the Non-CKD class, and 27% belongs to the CKD class. Fig. 1.4 shows a summary of this data-set.

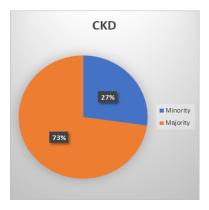


Figure 1.4: CKD data set

Wisconsin diagnostic breast cancer (WDBC)

WDBC data set has a sample of 569 patients. It has 31 input features. The patient may belong to be either the Benign or Malignant breast tumour group. Out of this 569 sample, 212 patients (37%) belong to the minority class. Fig. 1.5 shows a summary of this dataset.

1.5 PERFORMANCE EVALUATION METRICS

For binary classification, there are four different outcomes possible. These outcomes are described in Table. 1.2. Based on these outcomes, We calculated classification accuracy, recall, specificity, G-mean and the area under the curve (AUC) on benchmark data sets to evaluate and compare our proposed technique. A detailed description of these metrics is given in [32]. These

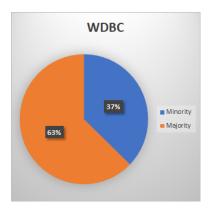


Figure 1.5: WDBC data set 0

metrics are summarized as follows:

Table 1.2: Binary Classification Outcomes

Predicted

Actual		Possitive class	Negative class			
	Positive class	True Positive (TP)	False Negative (FN)			
	Negative class	False Positive (FP)	True Negative (TN)			

Accuracy

The Accuracy of a classifier represents how often this classifier provides the correct result.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.1}$$

Specificity

Specificity measures how often a classifier predicts non-object (no). In the unbalanced data classification scenario, the negative class sample count is high, so this negative class is also called the majority class. Due to the higher counts, the learning ability of standard algorithms inclines toward the majority class. Therefore, standard techniques give a higher value of specificity.

$$Specificity = \frac{TN}{TN + FP} \tag{1.2}$$

Recall

The *Recall* is also called *Sensitivity* or true positive rate, which measures how often a classifier correctly predicts an object if it's an object.

$$Recall = \frac{TP}{TP + FN} \tag{1.3}$$

G-mean

G-mean is defined as the square root of the product of recall (minority class accuracy) and specificity (majority class accuracy).

$$G - mean = \sqrt{Recall * Specificity} \tag{1.4}$$

AUC

AUC is a probability curve generated by the plot of the false-positive rate (FPR) on the x-axis and true positive rate (TPR) on the y-axis. It denotes how much the trained model is capable of distinguishing between classes.

$$AUC = \frac{S_p - n_p(n_n + 1)/2}{n_p + n_n}$$
 (1.5)

where S_p is the sum of all positive samples ranked, while n_n and n_p are the numbers of negative and positive samples, respectively.

These metrics may be summarized by equations: 1.1, 1.3, 1.2, 1.4, and 1.5.

CHAPTER 2 REVIEW OF LITERATURE

This chapter gives a brief background and related literature work for the thesis. The initial three sections of the chapter talk about the background material, which includes the three substances: machine learning (ML), evolutionary computing, and genetic programming (GP). The last section discusses the literature review of unbalanced data classification. This chapter also discusses the challenges with unbalanced data classification, various approaches to handle it, the limitations of the existing methods available in the literature. The different techniques to address the unbalanced data classification are categorized into three groups: Data level, Algorithm level, and Ensemble of classifiers.

2.1 MACHINE LEARNING

Machine learning (ML) is a fast-growing area of research. ML refers to the technique where computer programs learn to solve various problems without being explicitly programmed. In ML, programs (also called the agent) automatically learn with their experiences in a given environment. These ML techniques may be broadly categorized into supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, training data is available as input-output pairs. Thus, in supervised learning, an agent learns how to map input data to output labels [33]. A well-trained agent correctly predicts the class labels for unseen cases. The focus here is that from the training data, the learning method should be able to generalize enough, to predict unseen samples correctly. Unsupervised learning refers to algorithms that discover unique patterns in data sets without explicitly providing any detail [58]. Thus, unsupervised learning techniques allow an agent to explore the underlying structure, detect similarities or patterns as clustering in a given data set on its own. In reinforcement learning, the agent gets feedback based on its outputs actions in terms of punishments or rewards [119]. Against supervised learning, the desired outputs are not explicitly provided. The agent learns only based on interactions with the environment and its observation in the form of reward and punishment.

2.1.1 Classification

Data classification is a process to categorize data into predefined groups. It is a supervised machine learning technique in which we first trained our model for a given number of predefined classes and then tried to classify data, which are unseen to our developed model. Cancer detection, skin disease prediction, fraud transaction detection, land-type classification, etc., are some leading examples.

Based on the nature of predicted output variables, the classification techniques can be divided into two groups:

- 1. *Binary classification:* In the *binary classification*, there are only two output classes. So, the data point may be whether this belongs to some group (true) or not (false). The simplicity of this case is that we have to focus on how to identify boundaries or thresholds.
- 2. *Multi-class classification:* In the *multi class classification*, the number of output classes (N) is greater than two. Designing classifiers for multi-class classification is more complicated and trickier. The majority of Unbalanced data classification works in the literature have been done for binary classification, and multi-class classification is still a challenging task [78].

This research work focuses on supervised learning. In this arena, there are many popular ML techniques available. These techniques include but are not limited to K nearest neighbourhood (KNN) and Support Vector Machine (SVM) (used in the experimental results to compare the proposed methods), briefly described as follows:

2.1.2 K Nearest Neighborhood (KNN)

The K-nearest neighbours (KNN) is a non-parametric, supervised ML technique. KNN doesn't require domain knowledge [30, 88]. In a multidimensional feature space, training samples are assumed to be a vector. All training samples are labelled with some class labels. The underline principle behind the KNN method is that data samples with similar characteristics

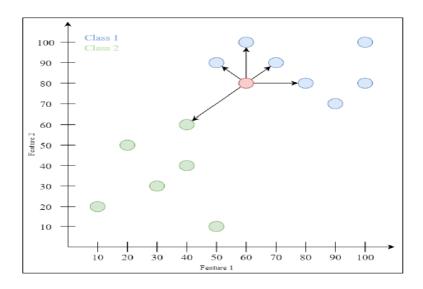


Figure 2.1: KNN Illustration for 2 Features and K=5

live near each other. Therefore while predicting, KNN considers K nearest samples, where K is a user-defined constant. At the time of classification, KNN calculates the distance of the testing sample vector from the labelled vector. It takes K nearest point from a testing sample, calculates the distance from these samples, and assigns a class label based on the most labelled points. The distance between two samples is called the degree of dissimilarity. Multiple-way of calculating similarity or distance exists for measuring the similarity/distance. One such metric is the Euclidean distance, defined as Equ. 2.1

$$D_{Euclidean} = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$
 (2.1)

where X, and Y, are two data samples, with N features. x_i , and y_i are i^{th} feature of X, and Y respectively.

A demonstration is shown in fig. 2.1 [65]. This figure explains how KNN with K=5 works for a classification problem with two features. For classifying the red circle, it looks at the five instances, which are closest to it. Since there are four blue circles nearest to the red circle, and blue circles belong to class 1, this red circle will be assigned to class 1. The main steps of the KNN are as follows:

1. Set the value of K.

- 2. Calculate the distance from the new data-sample (holdout from all of the data-sample in the dataset).
- 3. Discover the K samples in the provided training sample data that are 'closest or nearest' to the measurements of the unknown sample data point.
- 4. Predict the label or class of the considered data point using the most appropriate class or label from the KNN.

Selecting a value of K depends on the data. A higher value of K reduces the effect of noise data on classification. Against this, a lower value of K makes the boundary between classes less thin. Thus, a small value of K generates a highly complex ML model, resulting in the overfitting of the model. The high value of K makes the model highly generalized, i.e. under-fitted. In this work, we did *the grid test* [10] by performing multiple experiments with a different value of K and chose the best value of K, based on the performance shown.

2.1.3 Support Vector Machine (SVM)

In recent years, SVM emerged as a useful classification method [6, 53, 84, 108]. It is a non-parametric, supervised learning technique. SVM, proposed by *Vapnik*, maps the input sample data points to different categories or classes by creating a hyper-plane and maximizing the margin between these classes and hyperplane [22]. SVM technique is based on mapping given input data to a high-dimensional feature space so that the sample data points can be grouped into different classes, even when the provided data are not linearly separable. The mapping from an input space to a high-dimensional feature space is performed by a free defined set of mathematical functions. These functions are called kernels. The mapping concept is based on Cover's theorem [31], which states that there is a high probability that a given non-linearly separable data set may be transformed to linearly-separable data set by projecting it into a higher-dimensional space via some non-linear transformation. Once a separator is generated for the data, the data should be transformed so that a separator can be generated as a hyper-plane. This plane can be employed to classify new or unknown cases. Therefore, the SVM algorithm creates an optimal hyper-plane that categorizes new examples. A demonstration is shown in fig. 2.2 [65], where in a 2-d space, a hyper-plane is transformed into a line.

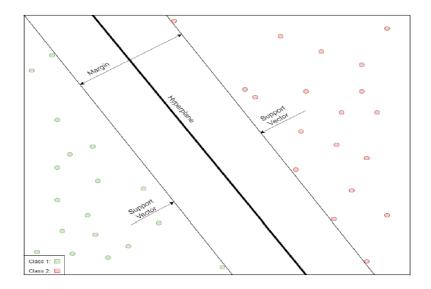


Figure 2.2: SVM Illustration of a 2-d Hyper-plane, Margin, Support-vectors

The main benefits of SVM are that they are perfect in high dimensional spaces and use a sub-set of training sample points in the decision functions. These decision functions are called support vectors. Another advantage of SVM is that it is memory efficient. The disadvantage of the SVM is that if the features count is higher than the training-samples count, it leads to overfitting. Another drawback of SVM is that it does not directly give probability estimates required in most classification tasks. Also, if the dataset is huge, SVMs are not very computationally efficient.

2.2 EVOLUTIONARY COMPUTING

The idea of adapting Darwin's principle of 'survival of fittest' for machine-driven problem-solving arose around the 1940s [48]. In 1948, Turing proposed a new computational model, called 'unorganized machines', which becomes the base of evolutionary computation [127]. In 1962. Bremermann presented formal computer experiments on 'optimization through evolution and recombination' [19]. During the 1960s, three variances were developed. First, Fogel et el. proposed Evolutionary programming (EP) with no fixed structure [47]. Second, Holland introduces the Genetic algorithm (GA) [59]. Third Rechenberg and Schwefel introduced Evolutionary strategies (ES) [12]. All these approaches were nurtured separately, but after the early 1990s, they have been collectively seen as evolutionary computing (EC) [45]. In 1992, a fourth

stream was emerged, known as genetic programming (GP), by Koza [71].

Table 2.1: Evolutionary Computing vs. Problem Solving

Problem Solving	Natural Evolution
Problem	Environment
Candidate Solution	Individual
Quality	Fitness

The central idea of evolutionary computation is to relate effective natural evolution to a specific form of problem-solving. This may be term as 'trial-and-error' [45]. For simplicity, consider natural evolution as follows. In nature, the environment is created with a population or pool of individuals. These individual beings attempt for their survival, and if successful in surviving, then perform reproduction. The survival fitness of these individuals is driven by the natural environment and refers to how well they succeed in attaining their goals. Thus fitness represents their survival and reproduction chance. Similar to this analogy, there is a collection of solutions in the stochastic trial-and-error (generate-and-test) style process of problem-solving. How well these individuals solve the problem (i.e. their quality) decides the probability of their existence and future use as seeds for constructing new candidate solutions. This similarity among the two scenarios may be summarized in Table. 2.1.

2.3 GENETIC PROGRAMMING

Genetic programming (GP) is a nature-inspired algorithm introduced by Koza [71]. The algorithm is based on Darwin's "principle of survival of the fittest" in nature. The building block of GP is individuals, called programs. These programs denote different mathematical formulas to solve the problem at hand. These programs are represented as a tree in memory (Figure-2.4). A set of individuals is processed for a predefined number of generations or till the satisfaction criteria are not met. This processing is done using three operators: reproduction, crossover, and mutation. These three operators are inspired by nature, where a good individual can create a copy of itself and select to go in next-generation (reproduction), two individuals can generate a new child individual, which contains some gene from first and some gene from the second

(crossover) and in some cases, one or more genes get mutated in the child (mutation). The functioning of these operators depends on an evaluation equation called fitness function. Based on the problem context, the objective will be to either minimize or maximize the fitness function. The value of the fitness function converges over generation by generation. Thus, in summary, the main steps of GP are population initialization and process this population over generation to generation by applying nature-inspired operations: *reproduction, crossover, and mutation*. When applying these crossover and mutation operators, it is always ensured that the generated child program satisfies the closure property, i.e., it is a valid solution.

The GP framework is summarized in Fig. 2.3.

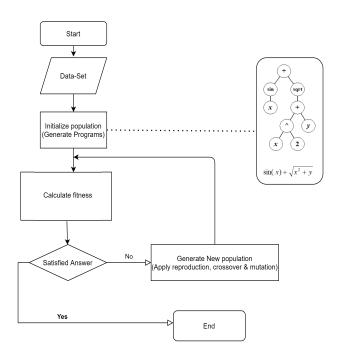


Figure 2.3: GP Framework

2.3.1 Initialization of population

In GP, we initialize a set of individuals. This set is called population, and individuals are called programs. Each program is represented as a tree. A program's in-memory representation is shown in Figure-2.4 [98]. The nodes will be either an internal node or a leaf node. All internal nodes are an element of a predefined set of functions F, and all leaf nodes are an element of either part of the feature vector (X) or a constant (T).

A range of tree depth, called initial depth, for the initial population is defined. Initial tree heights for a program is randomly chosen in this range. There are the following three methods to generate the initial population:

- 1. Grow: Here, nodes are randomly chosen from function set (F) or terminal set (T). This method will create trees of more varied sizes and shapes.
- 2. Full: Here, nodes are chosen from function set (F) till random initial depth is reached, and after that, the nodes are chosen from the terminal set (T).
- 3. Ramped half and half: This initialization method is also called the 'half and half' method. In this tree are grown 50-50 mix of the grow and full method.

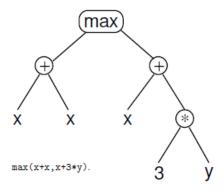


Figure 2.4: Program's Tree Representation

2.3.2 Standard GP Fitness Function for Classification

Overall classification accuracy has been used as a standard fitness function in GP. This is defined as the ratio of correctly classified samples versus total samples.

$$F_{acc} = \frac{TN + TP}{TN + TP + FN + FP} \tag{2.2}$$

where, TN denotes *true negative*, TP denotes *true positive*, FN denotes *false negative* and FP denotes *false positive* prediction counts.

The objective is to increase this classification accuracy close or equal to 1. This objective is achieved generation by generation in the GP framework, i.e. fitness of programs converge to 1.

2.3.2.1 Drawback with standard the fitness function

This can be easily observed that during fitness calculations, equation-2.2, gives equal importance to all examples, irrespective of whether it belongs to the minority class or it belongs to the majority class. As majority class examples have a higher count than minority class, this fitness function will favour the majority class. This favouritism leads to making our classifier biased toward the majority class. Thus during testing, this classifier will show better accuracy over the majority class and lower accuracies over the minority class. Also, as the majority class example count is higher, classifier accuracy will be still high.

Let's assume that there are 92% majority samples and 8% minority samples. If a classifier gives classification decisions favouring the majority class in all cases, its accuracy becomes 92%, but it fails to classify any minority class samples correctly. Also, another classifier having classification accuracy 90%, but for both classes, i.e., minority, as well as the majority, will be better than the first one.

In the case of binary classification, generally, model output is defined as the following:

$$f(x) = \begin{cases} 1, & \text{if } x \ge 0.5\\ 0, & \text{otherwise} \end{cases}$$
 (2.3)

For binary classification, the value 0.5 is treated as a boundary. Any value less than 0.5 is treated as a minority class, and any value greater than 0.5 is treated as a majority class. If two classifiers give output values, 0.99 and 0.51, respectively, both will be treated as 1. The point to be noted here is that, in the first case, the distance of the predicted value is 0.01, but in the second case, this distance is 0.49. Therefore, in reality, the first classifier is better in terms of classification accuracy. Thus, the issue with the above fitness function is that it may show 100% classification accuracy for even an under-fitted model. In another word, equation-2.2

don't respect this information. This is the second issue with standard fitness function.

Thus, the standard fitness function has two significant drawbacks. These drawbacks may lead the GP framework to generate biased and under-fitted classifiers. Therefore GP with standard fitness function is unsuitable for the classification problems with the unbalanced nature.

2.3.3 Reproduction

Once the population is initialized and the program's fitness values are calculated, a fixed percentage (e.g., 5%) of top fitness individuals are selected and will go to the next generation. This operation is inspired by nature's principle of *survival of fittest*.

2.3.4 Crossover

In the crossover, two programs are selected, and children are generated by taking some part of each parent. The most popular crossover mechanism in GP is a sub-tree crossover. This is illustrated in Figure-2.5 [98].

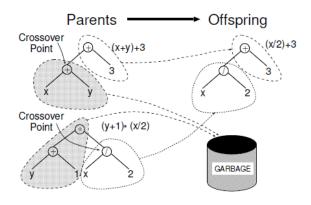


Figure 2.5: Crossover Between Two Parents

2.3.5 Mutation

In the mutation, some portion of the program is randomly changed. The most popular mutation is a sub-tree mutation, in which a mutation point is randomly selected, and a randomly generated sub-tree is inserted at that point. This is illustrated in Figure-2.6 [98].

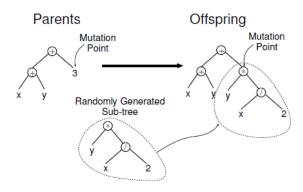


Figure 2.6: Mutation of a Program

2.3.6 Termination Criteria

The algorithm is terminated if either population is processed for a predefined maximum generation or training accuracy reaches to 100%.

The GP algorithm may be summarized as Algorithm-1. Since Population size (N) is constant for each generation, the sum of reproduction, crossover and mutation probabilities should be 1.00. We did extensive experiments with different values, and good results were produced when the crossover, mutation, and reproduction probabilities were set to 0.70, 0.20, and 0.10, respectively.

2.3.7 Classification using GP

Individuals, also called programs are the building block of GP. These programs are represented as a tree in memory. The leaf nodes of the tree are feature variables or random numerical constants. The internal nodes of tree are mathematical function or operators like +, -, *, /, log.

Algorithm 1 Genetic Programming Algorithm

1: **procedure** GENPROG(X,T)

⊳ X is input variables and T is constant

- 2: $N \leftarrow populationSize$
- 3: $genTotal \leftarrow 100$

4: $pReproduction \leftarrow 0.10$

⊳ reproduction probability

5: $pCrossover \leftarrow 0.70$

6:

▷ crossover probability▷ mutation probability

7: $bestFtnessGeneration \leftarrow 0$

 $pMutation \leftarrow 0.20$

- 8: $P_{current} \leftarrow \text{generate N random programs}$
- 9: Calculate *fitness* of all programs using defined fitness function.
- 10: $generation \leftarrow 0$
- 11: **while** generation \neq genTotal **do**
- 12: $P_{new} \leftarrow \phi$
- 13: Copy top pReproduction*N best fitness programs to P_{new}
- 14: Generate pCrossover*N programs by applying crossover and add to P_{new} .
- 15: Generate pMutation*N programs by applying mutation and add to P_{new}
- 16: $P_{current} \leftarrow P_{new}$
- 17: Calculate fitness of all programs in $P_{current}$.
- 18: // check stopping criteria, if satisfied exit from while loop
- 19: **return** program with best fitness

Thus, these programs denote different mathematical formulas to solve the problem at hand. As a mathematical formula, it produces a single floating-point number which is the calculated output of the GP tree for any input data sample. This tree's output is then mapped to a set of class labels. In binary classification, as there are only two class labels, a threshold value is used as class boundaries to bifurcate the class labels [92, 128, 131, 143]. For example, consider a program represented as Fig. 2.4, with two input features x and y. If we take the threshold value 0.5, any output value greater or equal to 0.5 will be treated as the positive class label. Any output value less than 0.5 will be treated as the negative class label.

2.4 UNBALANCED DATA CLASSIFICATION

In various real-world classification applications, such as medical diagnosis [39, 66, 115, 147], fault diagnostics [112], fraud detection [35, 116], or bio-informatics [3, 20], it is common to have an unequal number of training examples in one class against to the other class(es). This characteristic is said as unbalanced-class or Unbalanced class. A data set is called unbalanced or Unbalanced if at least one class has limited examples, and the class with a smaller number of samples is called the minority class. Similarly, a class having a more significant number of data samples is called the majority class [132, 68].

These unbalanced data sets have their inherent complexity, and in this case, the model's accuracy is not straightforward. The quality and ability of classifiers are highly impacted by the nature of data used to train [54]. Therefore in the case of unbalanced data classification, the ability of classifiers may also be unbalanced. Most of the classifiers show biased or lower accuracy on unbalanced data [46]. Recent research works done by different researchers show that this Unbalanced nature of data misleads classification accuracy [16]. As there are fewer examples in the minority class for training and testing, the weight of these data on model accuracy will also be limited. Thus, in the Unbalanced data set, higher accuracy is not necessarily an indicator of its goodness. In fact, in most cases, the correct classification of the minority class is more needed than the majority class. For example, in fraud transaction data, the fraud transaction case (minority class here) is even less than 1%. Nevertheless, here this minority class is of more interest to any user. In this case, a classifier may have 98% accuracy, but it still

fails to detect any fraud transactions. This point needs to be considered when we compare the performances of classifiers.

2.4.1 Challenges of unbalanced data

The minority class samples represent rare events. Examples of these rare events are fraud detection, natural clematis, software defects, etc. Although these events rarely happen in day-to-day life, it has a high impact on the outcome. Classification of these datasets has its challenges, and some of the significant difficulties may be summarized as follows:

- 1. Performance evaluation metrics like overall classification accuracy, mean square classification error, etc., are biased and inclined toward the majority class [81].
- 2. Standard classification techniques like logistic regression, decision tree, Naive Bayes classifiers, support vector machine (SVM), etc., are more suitable for balanced data sets. In the case of unbalanced data sets, they give sub-optimal solutions. These sub-optimal solutions are inclined towards better majority class classification [80].
- 3. In some cases, minority class samples are treated as noise by classifiers [11].
- 4. It is possible that some of the noise may be treated as a minority class sample by the classifier [11].
- 5. Sometimes, minority class samples boundary overlapped with majority class sample boundaries, which make it hard to learn or false learning by the model during training [81].
- 6. As the minority class sample counts are very low, the learning impact of these samples is significantly less during the training of the model [15].
- 7. In the case of a high dimensional dataset, learning from minority class samples becomes the most tedious task during training [18].

2.4.2 Handling Unbalanced Data Classification

For handling unbalanced data classification problems, multiple approaches are presented in the literature. These approaches can be categorized into the below groups:

- 1. Data level approaches
- 2. Algorithm level approaches
- 3. Ensemble of classifiers

2.4.3 Data level approaches

Data level approaches concentrate on modifying the original data set to improve the classification accuracy of models. Data sampling is a common approach coming into this category. Data sampling is also called an external approach. Pre-sampling techniques try to create a trade-off between minority class samples and majority class samples by oversampling the minority class samples or under-sampling of majority class samples. In random oversampling (ROS), minority class samples are randomly chosen, and duplicate copies of these samples are generated. Against this, in random undersampling (RUS), some majority class samples are removed. For ROS of the minority class, a popular method is the synthetic minority over-sampling technique (SMOTE) [25]. In this technique, to generate new minority class examples, SMOTE utilizes the concept of K nearest neighbourhood ML algorithm. This is possible that SMOTE may generate some wrong and noisy minority samples, which highly impact the performance of a classifier. Also, oversampling may lead to the overfitting of models, whether under-sampling may lead to the removal of some vital training data and the model becomes under fitted. Thus under sampling-based techniques may create a loss of important information [74].

For training multi-layer perceptron (MLP), [78], introduced a dynamic sampling (DyS), based technique. In this work authors focused on two main aspects during training. The first aspect gives more importance to minority class examples during training, and the second aspect gives more importance to examples more difficult to classify. Figure-2.7[78], shows a scenario where classifying class 2 samples are more difficult than class 3. During MLP training, for each

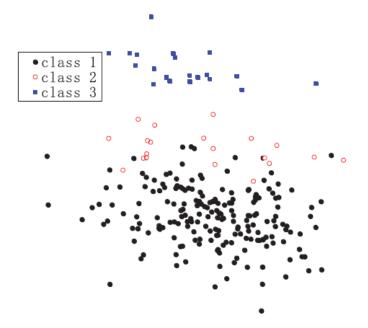


Figure 2.7: Example of a Three-class Classification Problem

epoch, the training selection probability for each training example is estimated. This probability will play a deciding factor that for updating MLP, a considered example will be used or not. At the place of classification accuracy, the Multi-class area under the curve (MAUC) and G-mean is used to compare the performance. For comparing performance, 20 UCI data sets [43] are taken, and the proposed technique is applied. The result set shown in the paper concludes that DyS performed better than ROS, RUS, and no sampling (NoS) techniques on most of the data.

Another approach to handling unbalanced classification is to introduce synthetic samples for the minority class. However, in some scenario, the wrong sample may be generated, which make learning more difficult. *Barua et el.*[9] proposed Majority Weighted Minority Oversampling Technique (MWMOTE) for efficient handling of unbalanced data classification problems. The method first extracts samples that are hard to learn. These are the borderline samples as shown in Figure-2.8[9]. A weight is assigned to samples based on their importance. These weights are worked as selection probability. In last, synthetic samples are generated based on a customized clustering technique. The proposed technique is evaluated for 20 real world UCI data set, and results are compared with SMOTE and other techniques.

In the literature, the class imbalance ratio (IR) is used to measure the degree of class imbalance. IR is a suitable metric in binary classification, but in multi-class classification, it is

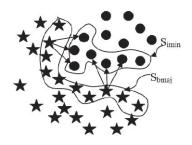


Figure 2.8: Illustration of Boundary Samples

not a good measure [122]. We can define entropy as an average amount of information present in a data set for multi-class classification. *Li et el.*[74] proposed entropy-based imbalance degree (EID) as a metric in their work and introduced three IR based techniques:

- 1. Entropy-based over-sampling (EOS) technique: over-sample other classes until their count reaches to largest majority class.
- 2. Entropy-based under-sampling (EUS) technique: under-sample other classes until their count reaches to the lowest minority class.
- 3. Entropy-based hybrid sampling (EHS) technique: merge the above two techniques. This technique under-samples the majority class and over-samples the minority classes until they are balanced between all classes.

The proposed techniques are implemented for 12 UCI data sets *ecoli*, *landsat*, *letter*, *msplice*, *page-bk0*, *penbased*, *segment0*, *shuttle*, *thyroid*, *vehicle1*, *wavefm3* & *yeast* and performance is compared with Adaptive Synthetic (ADASYN), SMOTE, MWMOTE and others. Figure-2.9[74], show how synthetic samples look like in different techniques.

Many advanced sampling-based techniques are also present in the literature [26, 69, 110, 135].

2.4.4 Algorithm level approaches

Although lots of works have been done under data level categories, but these external level techniques create a lot of computational overhead [141]. Algorithmic approaches concentrate on

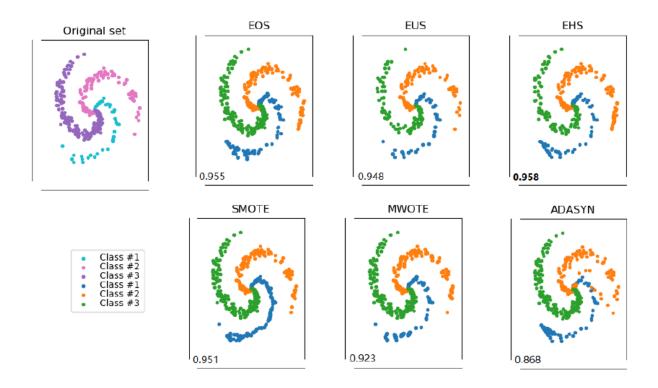


Figure 2.9: Illustration of Data Sampling Methods

proposing either new algorithms or modifying existing algorithms to improve the model's accuracy. GP is an evolutionary technique used to solve many problems by researchers. Bhowan et el.[15] proposed a cost adjustment approach in GP. The adjustment of cost is also called the internal approach. In GP, adjustment of cost can be performed by complying with appropriate fitness functions. This fitness function should be designed in such a way that a solution performed good for the minority as well as the majority classes, should be rewarded. Against this, a solution performing well only for the majority class should be penalties. Bhowan et el.[15] show that as the fitness function, neither accuracy (acc) nor average class accuracy (ave), fully solve the issues of unbalanced classification. Therefore, they proposed 3 new fitness functions: amse, incr and corr. The amse fitness function, is based on average mean square error. This work is based on an approach in which solutions with the smaller deviation between predicted and expected values will be rewarded. The *incr* fitness function assigns incremental rewards to solutions whose predicted output falls further away from the class boundary. The corr fitness function, is based on correlation ratio. As a benchmark, authors tested their proposed fitness function on 6 UCI data sets Ion, Spt, Ped, Bal, Yst1 and Yst2. In this work authors claim that for the binary classification, the newly introduced fitness functions performed better in the term of accuracy.

For solving multi-class classification problems, *Bhardwaj et el.*[13] have proposed a Genetically Optimized Neural Network (GONN) model. Here the complete classifier is composed of multiple GP tree, in which each tree represent one class. For each tree, if a tree output is 1, it belongs to that class; otherwise, it belongs to another class. For a given sample, only one tree's output will be one and other trees outputs will be 0. Tree generation is restricted so that it preserves the structure of the Artificial Neural Network (ANN) model. The function set is defined as $\{P,W,+,-,*,\%\}$, where P is sigmoid function, work as the activation function and W is the *weight* function. $\{+,-,*,\%\}$ are mathematical operators: add, subtract, multiplication, and division, respectively. Terminal set T is set of feature variables. The performance of the proposed technique is compared with ANN, GA, SVM, etc., and results are found comparably good than SVM, GA and ANN.

For Unbalanced classification, *Zhang et el.*[141], proposed an Evolutionary Cost-Sensitive Deep Belief Network (ECS-DBN), in which they focus on class-dependent miss-classification costs. To optimize the misclassification cost of Deep Belief Network (DBN), authors use an evolutionary algorithm variant named adaptive differential evaluation. For evaluation, 58 binary classification data sets are taken from KEEL (Knowledge Extraction based on Evolutionary Learning) data set repository [5]. Although the author took the data set of binary classification for evaluation, it may also be applied in multi-class classification.

For high dimensional unbalanced data classification, *Pei et el.*[97], proposed a new fitness function. This newly proposed fitness function is seen as an approximation AUC measure. The authors proposed a strategy to reuse previous good GP trees to save processing time and improve efficiency. *Tahir et el.*[120], proposed a GP fitness function based on Entropy and Information Gain to handle unbalanced data-set. Authors named this approach as Entropy and Information Gain based Genetic Algorithm (EIG-GA).

Numerous research works have been done to solve the unbalanced data classification are criteria based on assigning weight to the majority and the minority class classification performance. *Tran et el.*[126], assigned equal weight importance to the accuracy of both class samples. *Xong et el.*[151], proposed a Weighted Extreme Learning Machine (WELM) for unbalanced data classification. This work benefited from the concept of original extreme learning machine (ELM) which is simple and suitable to apply. *Lu et el.*[82] and *Raghuwanshi* [101], also proposed an improved version of a similar approach.

Several techniques are proposed in the literature, based on the distance between predicted and actual outcomes [39, 15, 21, 34]. In GP, these fitness functions calculate the difference between the real and the predicted output for each instance during training in different way.

Evolutionary Multi-objective Optimization (EMO) is also gaining popularity in research which facilitates solution to learning with multiple conflicted objectives. EMO based classifiers a Pareto optimal front based solutions are evolved to capture the performance trade-off between the different conflicting objectives [37]. This is different than single solution predictor classifier techniques where the fittest individual is returned from the training process. In this approach, the accuracies of the minority and majority classes will be considered separately in the learning process. EMO achieves this by treating the objectives independently in the learning procedure using the notion of the Pareto Dominance concept in fitness. Pareto Dominance concept establishes a rank based individual in the population according to how perfectly those solutions perform on all the objectives concerning each other [63, 16, 29].

2.4.5 Ensemble of classifiers

In this approach, new techniques are proposed by combining two or more existing techniques or classifiers to improve the accuracy of the model. *Bhowan et el.*[16] proposed an ensemble approach of multi-objective genetic programming to generate efficient classifiers for binary class classification. In this work of binary classification, the minority class accuracy and the majority class accuracy are treated as two objectives. For the cost adjustment, the authors utilized two existing Pareto-based fitness calculation techniques: Strength Pareto Evolutionary Algorithm 2 (SPEA2) by *Zitzler et el.*[148], and non dominated sorting genetic algorithm II (NSGA-II) by *Deb et el.*[37]. These techniques are based on dominance count and dominance rank. NSGA-II use dominance rank, which is the count of all solutions in the population, which dominate a given solution. SPEA2 uses both dominance rank and dominance count. Dominance count is the total count of solutions that were dominated by a particular solution. Again for diversity *crowding* metrics is used, which is nothing but Manhattan distance between solutions in objective space. Thus, solutions in the sparse region are preferred compared to solutions in the dense region if both solutions have the same fitness. Thus generated Pareto front classifiers work together by a voting mechanism. The authors use two ensemble selection strategies:

accuracy based approach and offline Ensemble selection algorithms (off-EEL). Also, the authors compared two voting schemes namely: traditional majority vote and a fitness-weighted vote. Six binary classification data sets Ion, Spt, Ped, Bal, Yst1 and Yst2 are taken, and performance is compared with Support Vector Machines (SVM) algorithm [123]. SVM supersedes on majority class accuracy whether as their proposed ensemble GP approach supersedes on minority class accuracy.

For multi-class classification, *Fernandes et el.* [46] proposed an Ensemble of classifiers based on the Evolutionary Multi-Objective genetic Sampling for Unbalanced Classification (E-MOSAIC) technique. The concept of E-MOSIAC is based on a multi-objective genetic algorithm named NSGA-II [37]. First a predefined *N*, balanced samples are generated from training data. The sample size is based on 90% of the total minority example count. For example, if there are 3 classes and the smallest minority class has 100 examples, then the sample size will be 3*100*0.90 i.e. 270 (90 from each class). In the genetic algorithm (GA), each sample is treated as an individual. The solution is represented as 1 if an example is selected and 0 if an example is not selected. This individual (sample) is used to train a base classifier, MLP. Two statistical approaches are used for measuring performance: Geometric Mean ()G-Mean) and moving Average Unit Cost (mAUC). The comparison shows that E-MOSAIC performed better than *DyS*, *ROS* and *RUS*.

Singh et el.[115], proposed a transfer learning approach to solve the Unbalanced classification of histopathological images. In this work, authors trained a VGG-19 model with ImageNet dataset [113, 38]. The learned knowledge is applied to the learning of histopathological images. From different layers of VGG19, the various number of features are extracted, and the Random Forests algorithm were used to classify these histopathological images [77]. The experiments were performed on this large scale dataset, which contains 277524 images, and the authors conclude that this approach handles the Unbalanced nature of data. Many ensemble based methods are also present in the literature [51, 61, 124, 146].

2.5 SUMMARY

This chapter presents the background and related work for the thesis. This background details cover machine learning (ML), evolutionary computing, and genetic programming. The related work discusses the literature review, including the challenges with unbalanced classification, various methods to handle it, the existing approach's limitations. The existing unbalanced classification methods are categorized into three groups: Data level, Algorithm level, and Ensemble of classifiers. The unbalanced data set has its complexity and the model's accuracy, in this case, is not straightforward. Lots of work have been done in the literature. But there is room for improvement of techniques to be used for unbalanced data classification.

CHAPTER 3

ASSESSMENT OF WEIGHT FACTOR IN GP's FITNESS FUNCTION FOR UNBALANCED DATA CLASSIFICATION

This chapter briefly introduces the custom weight assignment to the minority class samples during classifier training, suited for unbalanced data classification. After that, the proposed methodology for this weight assignment in fitness function is given, followed by experimental setup and results. The last section summarizes the finding of the custom weight assignment to the minority class while training a GP-based classifier.

3.1 INTRODUCTION

Various ML techniques like K-nearest neighbour (KNN) [52, 137], Support Vector Machine (SVM) [86, 129], Multilayer Perceptron (MLP) [24, 138], Decision Tree (DT) [73, 87], fuzzy logic [83, 96], etc are used for data classification. Recent research works in the literature show that classifiers give biased results when data sets are unbalanced [46, 74, 141, 147]. This biasing is included because of higher sample counts of the majority class samples during training. As the minority class sample count is lower, its weighted impact on the learning ability of the classifier will also be lower. As a solution to this learning biasing, the data-level approach may be used, to create a trade-off between the majority and the minority class samples count. This trade-off is achieved either by over-sampling minority class data points or under-sampling of majority class data points. These data level approaches introduce a high computational cost. Also, over-sampling adds over-fitting, and under-sampling introduces the under-fitting of the classification model, respectively [74]. Also, These data-level techniques may suffer from loss of critical information, or the creation of noisy data contained in samples during the processing of unbalanced data [142]. Therefore, to handle unbalanced data classification issues, we have focused on the internal approach, i.e., the algorithm level approach. Genetic programming (GP) is a nature-inspired optimization algorithm introduced by Koza [71, 72]. In GP, many researchers focused attention on adapting the new fitness function to reward solutions that are accurate on both the minority and the majority classes [15, 100]. Classification accuracy, as the fitness function, doesn't solve the issues of unbalanced classification [15]. Therefore, numerous works have been done to solve the unbalanced data classification. Many of them are based on the criteria of assigning distinct weight to the majority, and minority class classification performance [9, 82, 88, 126]. Although the proper adjustment of weights during the training phase improves the model classification performance, assigning these weights can require a priori domain knowledge [15]. Therefore this chapter evaluates the minority class weightage during classifier training and concludes a generic value that should work for most of the classification problems.

3.2 PROPOSED METHODOLOGY

In GP, one approach to resolve the Unbalanced classification problem is to assign a custom high weight to the minority class during training. This custom weight assignment may nullify the impact of higher counts of the majority classes during the learning phase of the classifier. The GP fitness function may introduce the custom weight assignment for the minority class samples. This research work assesses the impact of weight factors in GP's fitness function for unbalanced data classification. For custom weight assignment to the minority class, we define fitness function as Eq. 3.1.

$$F_{weight} = W * \frac{TP}{TP + FN} + (1 - W) * \frac{TN}{TN + FP}$$

$$(3.1)$$

where TP indicates *true-positive*, TN indicates *true-negative*, FP indicates *false-positive*, and FN indicates *false-negative* counts for predictions.

In Eq. 3.1, the first part represents the minority class accuracy multiplied by a custom weight W and the second part describes the majority class accuracy by a custom weight (1-W).

3.3 EXPERIMENTAL SETUP

Our objective is to assess the impact of the weight factor in the GP fitness function for generating a classifier that gives accurate results over each class. For that, we have performed extensive experiments on five unbalanced problems with different weight factors. These data set are ABL-9-18, YEAST-2, YEAST-1, CKD, and WDBC. We have performed these experiments by assigning minority class weight values of 0.2, 0.4, 0.5, 0.6, and 0.8. Accuracy is always favourable toward the majority class due to its higher sample count. Against this, as the

AUC is non-biased since it treats the minority and majority class learning equally, we measure performance on the AUC values. In this research work, we use *StratifiedShuffleSplit*. For each weight value, 30 experiments are executed and mean AUC values are calculated for each Unbalanced problem. In each experiment, 80% data is used for training, and 20% data is used to evaluate the classifier's performance. This partitioning is done randomly for each experiment. GP is implemented in Python. For population initialization, we have used the "ramped half and half" method. This method generates a wide variety of individuals of various sizes, and shapes [71]. The tree initialization depth is taken as (5,15). The function set contains {+,-, *, /, and sigmoid} functions. The reproduction, crossover, and mutation probability is set to 0.10, 0.70, and 0.20, respectively. Population size is set to 200, and max generation is set to 100. For simplicity, sub-tree crossover, sub-tree mutation, and tournament selection with tournament size two is taken. For all these experiments GP parameter values are given in Table-3.1.

Table 3.1: GP Parameter Setting For Minority Class Weight Assessment

Parameter	Value
population initialization	ramp half and half
population size	200
maximum generation	100
max tree height	15
min tree height	5
function-set	+, -, *, /, sigmoid
tree terminals	feature variables & random constants
reproduction	0.10
mutation	0.20
crossover	0.70

3.4 RESULTS

Accuracy treats the performance of each class blindly and uses the total samples count irrespective of which class has how many instances. As majority class samples have higher sample counts, performance in terms of accuracy is always biased toward the majority class. Better performance measurement for Unbalanced classification is the area under the ROC (receiver operating characteristic curve) known as AUC. The AUC considers similar offerings to both minority and majority classes when evaluating the performance of an algorithm [91, 32]. As AUC is a non-biased performance measurement, we consider our results based on the AUC values.

Table 3.2: Custom Minority Class Weight vs Produced AUC Values

Data-Set	W=0.2	W=0.4	W=0.5	W=0.6	W=0.8
ABL-9-18	0.764 ± 0.078	0.753 ± 0.108	0.807 ±0.059	0.785 ± 0.08	0.761 ± 0.083
YEAST-2	0.881 ± 0.038	0.905 ± 0.025	0.901 ± 0.025	0.913 ±0.023	0.898 ± 0.023
YEAST-1	0.719 ± 0.031	0.747 ± 0.035	0.747 ±0.029	0.742 ± 0.035	0.639 ± 0.035
CKD	0.983 ± 0.025	0.989 ± 0.022	0.991 ±0.021	0.989 ± 0.022	0.989 ± 0.022
WDBC	0.921 ±0.029	0.922 ± 0.029	0.930 ± 0.025	0.932 ±0.021	0.906 ± 0.021

Experimental results are summarized in Table-3.2 and Figure 3.1, in which the mean value of AUC and the standard deviation is given. Out of five Unbalanced problems in this work, three benchmark Unbalanced problems produced the best result values when minority class weightage in fitness function is set to 0.5. These three Unbalanced problems include ABL-9-18, YEAST-1, and CKD. Two Unbalanced problems YEAST-2, and WDBC are slightly lower on W=0.5 than W=0.6 but better than other values of W. For YEAST-2, the performance difference is 0.012, and for WDBC, the difference is 0.002 compared to W=0.6. Setting the value of W in an apriori and domain-independent manner is required for designing any generic method. If we define a range, then the optimal values range of the minority class weightage is 0.50 ± 0.10 . Therefore we can conclude that as a generic approach, we can set the value of W equal to 0.50.



Figure 3.1: Custom Minority Class Weight vs Produced AUC Values

3.5 SUMMARY

The Unbalanced distribution of data across various classes generates intense challenges to classification algorithms. One way to solve these challenges is to assign a custom weight to the classes during model training. In GP, the learning process of classifiers is governed by the fitness function, and this fitness function can easily incorporate custom weight assignments. Setting the value of W in an *apriori* and domain-independent manner is required for designing any generic method. Therefore, we performed extensive experimentation on five UCI repository base Unbalanced problems in this work, with various values of minority class weight. Based on experimental results, we can conclude that for better handling of Unbalanced classification problems in a generic manner, we can set the weight value to 0.5. This custom weight assignment for the minority class samples during training generated better solutions, which tackle the challenges of the Unbalanced data classification problems. Thus custom weight factor in GP fitness function gives more balanced and accurate solutions for the Unbalanced data classification problems.

CHAPTER 4

AN EUCLIDEAN DISTANCE AND WEIGHT-BASED FITNESS FUNCTION IN GP FOR UNBALANCED DATA CLASSIFICATION

This chapter briefly introduces the custom weight assignment to the minority class samples and the distance between actual and predicted values during classifier training, suited for unbalanced data classification. After that, the proposed methodology for using weight and distance in fitness function is given, followed by experimental setup and results. The last section summarizes the finding of the proposed fitness function while training a GP-based classifier.

4.1 INTRODUCTION

Unbalanced data significantly challenges traditional classification algorithms. A variety of algorithms can do classification accurately, and their behaviour is accurate when the sample count distribution of objects or events is equal or nearly identical [55, 57]. These standard algorithms like Support Vector Machine (SVM), K-nearest neighbour (KNN), logistic regression, decision tree (DT), Naive Bayesian (NB), etc., may converge to a sub-optimal solution [36]. The higher sample count of the majority class significantly influences the classifier's learning capability during training. Due to this influence, classifiers give predictions biased toward the majority class [46, 141, 147]. Small sample size, lack of density, and high dimensionality also affect the learning process of algorithms [18, 134]. The learning algorithms guided by generic performance metrics like classification accuracy lead to a biased classification toward the majority classes [8, 105, 130].

To tackle Unbalanced classification issues, the data-level approaches focus on the *over-sampling* of the minority class examples or *under-sampling* of the majority class examples [4, 56, 70, 105]. However, oversampling the minority class sample may lead the classier to be over-fitted and create a noisy sample. On the other side, under-sampling of majority class examples may lead the classifier to under-fit or miss some crucial samples from the learning process [74, 142]. Also, these methods have drawbacks of high computational cost and a priori knowledge of the problem space [15, 141]. Another group of techniques calls *algorithm-centric techniques* either focus on proposing a new algorithm or modification to existing algorithms so that learning toward the minority class sample given much attention during training [57, 79, 142]. In this way, these approaches focus that the majority class data point's higher count does not supersede the minority class data points. Genetic programming (GP) is a nature-inspired

evolutionary computing technique [72]. GP, motivated by natural evolution, has been applied in various domains to solve a wide variety of problems [2, 14, 17]. The training ability of GP has also affected if the nature of the data is Unbalanced.

Fitness function is the primary criteria in GP for rewarding the right solution. However, in Chapter 2, section 2.3.2.1, we have seen that the standard fitness function has two significant drawbacks. These drawbacks may lead the GP framework to generate biased and under-fitted classifiers. Therefore GP with standard fitness function is unsuitable for the classification problems with the unbalanced nature. To resolve these issues, numerous works focus on assigning more weight to minority class samples during training for handling biased learning [9, 82, 88, 126]. The difficulty of such an approach is that assigning these weights may require problem-specific knowledge [15]. Different researchers also employed measuring distance how far the prediction point located, compared to the actual point in the solution space [21, 34, 15]. We utilized both these mechanisms and proposed a *Euclidean distance and weight-based (EDWB)* fitness function in GP to classify the Unbalanced data. The proposed fitness function focused on the cost adjustment of solutions during their fitness calculations. The proposed method doesn't require any domain-specific knowledge or data-prepossessing techniques when applied to an unbalanced dataset.

4.2 PROPOSED METHODOLOGY

This work combines the custom weight assignment to minority class and the distance between the predicted and the actual results. We proposed a new fitness function to handle unbalanced data classification. This proposed fitness function is given as Eq. 4.1. Based on our previous experiment in Chapter 3, we set the value of W as 0.50 in this fitness function. Thus, we give equal weightage to both the classes in the proposed fitness function.

$$F_{proposed} = \sum_{i=1}^{N_{major}} \frac{\left| dist_{major_i} \right|^2}{2 * N_{major}} + \sum_{i=1}^{N_{minor}} \frac{\left| dist_{minor_i} \right|^2}{2 * N_{minor}}$$
(4.1)

Where,

N_{major}: Total count of majority samples in training data.

N_{minor}: Total count of minority samples in training data.

dist_{major} i: Distance of predicted value from actual value in the ith majority class sample.

dist_{minor} i: Distance of predicted value from actual value in the ith minority class sample.

Here, our objective will be to minimize the fitness function values.

The above-proposed fitness function in equation-4.1 has two parts. The first part caters to the classification performance of the majority class, whether as second part caters to the classification performance of the minority class. In other words, it gives importance to the minority class training samples the same as the majority class training samples. In this way, the new fitness function resolves the first issue of the standard fitness function mentioned in the previous section.

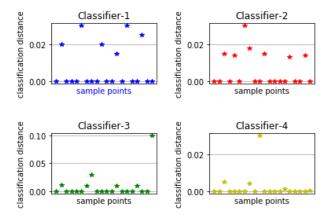


Figure 4.1: Illustration of Classifier Performance Evaluation

Also, the proposed fitness function calculates the distance between the predicted value and the expected value. In binary class classification, the expected output values are 0 & 1. As shown in equation-2.3, value 0.5 acts as a boundary value for binary classifiers. If the distance is less than 0.5, this prediction is treated as correct. In figure-4.1, four classifiers are shown. The *x-axis* denotes different training samples, and the *y-axis* shows the distance between the predicted value and the actual value. As in the case of all four classifiers, a distance of the predicted value from the actual value for each sample training point is less than 0.5, and all classifiers have a classification accuracy of 100%. But in reality, classifier 4 is better than classifier 3, and

classifier 3 is better than classifier 2, and classifier 2 is better than classifier 1. The proposed fitness function as equation-4.1, represents this fact mathematically. In this way, this newly proposed fitness function resolved the second issue of the standard fitness function. In summary, the above-proposed fitness function solves both the problem of the standard fitness function. Thus by assigning equal weightage to the minority and the majority class and considering the Euclidean distance between actual and predicted values, the proposed fitness function provides a better and more generalized solution toward unbalanced classification.

Summary of the proposed approach

The proposed GP approach is summarized as Algorithm-2. The proposed method uses a new fitness function to resolve the problem of the standard fitness function. Also, since this method resolves the issue of under-fitting, its side effect is possible if we train many iterations.

To avoid this, at every 5th generation, we calculate the classification accuracy of the best program in the population. If classification accuracy reaches to 100%, we decrease the *total generation* count. We set this *total generation* value to the minimum of *total generation* and K% more than the current generation. Multiple runs of the proposed approach were performed, and an excellent result was found at K=25. For example, let's assume that the total generation is defined as 100. Now, if at the 20th generation, the best program in the population shows 100% accuracy, then we will set the total generation value to (20+5), i.e., 25. This will avoid the problem of over-fitting the model. Therefore, this approach balances between underfitting and overfitting of model properties.

Thus the proposed approach solves the problem of biased classification by addressing both the issues with the standard GP approach.

4.3 EXPERIMENTAL SETUP

To evaluate the proposed method, we have performed extensive experiments on five unbalanced problems with varying levels of the Unbalanced factor. The five benchmark datasets are ABL-

```
Algorithm 2 Proposed Genetic Programming Algorithm
```

```
⊳ X is input variables and T is constant
 1: procedure GENPROG(X,T)
 2:
        N \leftarrow popSize
 3:
        gTotal \leftarrow 100
                                                                                    ▶ Total Generation
        pReprod \leftarrow 0.10
 4:
                                                                            > reproduction probability
        pCrssOvr \leftarrow 0.70
 5:
                                                                               > crossover probability
        pMutn \leftarrow 0.20
                                                                                6:
 7:
        bestFtnessGen \leftarrow 0
 8:
        P_{current} \leftarrow \text{generate N random programs}
         Calculate fitness of all programs using proposed fitness function.
 9:
        gen \leftarrow 0
10:
        bAcc100Achieved \leftarrow 0
11:
12:
        while gen \neq gTotal do
            P_{new} \leftarrow \phi
13:
            Copy top pReprod*N best fitness programs to P_{new}
14:
15:
            Generate pCrssOvr*N programs by crossover and add to P_{new}.
16:
            Generate pMutn*N programs by applying mutation and add to P_{new}
            P_{current} \leftarrow P_{new}
17:
            Calculate fitness of all programs in P_{current}.
18:
             // check stopping criteria
19:
            if (gen\%5 == 0) \&\& (bAcc100Achieved == 0) then
20:
                 trainAcc = training accuracy of best program
21:
                if (trainAcc == 100) then
22:
                     bAcc100Achieved \leftarrow 1
23:
                     gTotal \leftarrow \min(gTotal, 1.25*gen)
24:
25:
        return top program with best fitness
```

9-18, YEAST-2, YEAST-1, CKD, and WDBC. This Unbalanced factor varies from 6:94 to 37:63. The proposed method is also compared with standard ML algorithms. All approaches are implemented in Python. To make the comparison in the same environment and similar testing parameters, we also implemented SVM, KNN, and GP with standard fitness function. The k-nearest neighbour's algorithm is a non-parametric classification algorithm, and it is highly

used for classification and regression problems. KNN's performance depends on the value of K. Therefore, we performed a grid-test for all data sets separately to find the optimal value of K. In this grid-test, we have taken K values in the range of 1 to 30. This grid test result summary for five-fold cross-validation is shown in Table-4.3. For comparison, we have taken the KNN result with the best K values. In this research work, we use *StratifiedShuffleSplit*. For each benchmark problem, 30 experiments are executed and mean values of accuracy, recall, specificity, G-mean, and AUC, are calculated. In each experiment, 80% data is used for training, and 20% data is used to evaluate the classifier's performance. This partitioning is done randomly for each experiment.

For GP population initialization, we use the "ramped half and half" method. The tree initialization depth is taken as (5,15). The function set contains {+, -, *, %, sigmoid, log, sqrt, sin, absolute, max, and min} functions. We conducted extensive experiments to decide on different parameter settings. Based on these experiments, we set the reproduction, mutation, and crossover probabilities of 0.10, 0.20, and 0.70, respectively. We set population size as 1000 and max generations as 200. For simplicity, the tournament selection, the sub-tree mutation, and the sub-tree crossover are considered. These GP parameter values are given in Table-4.1.

Table 4.1: GP Parameter Setting for Proposed Method

Parameter	Value
population initialization	ramp half and half
population size	200
maximum generation	1000
max tree height	15
min tree height	5
function-set	+, -, *, %, sigmoid, log, sqrt, sin, absolute, max, and min
tree terminals	feature variables & random constants
reproduction	0.10
mutation	0.20
crossover	0.70

Table 4.2: Proposed GP Method Performance Evaluation

Data-Set	Imbalance	Algorithm	Accuracy	Recall	Specificity	G-mean	AUC
	Ratio	$(avg \pm std)$	$(avg \pm std)$	$(avg \pm std)$	$(avg \pm std)$	$(avg \pm std)$	$(avg \pm std)$
ABL-9-18	06 : 94	GP_{edwb}	76.640 ± 6.910	0.838 ± 0.144	0.762 ± 0.076	0.794 ± 0.070	0.800 ± 0.069
		GP_{std}	95.380 ± 1.000	0.329 ± 0.125	0.990 ± 0.008	0.560 ± 0.110	0.659 ± 0.062
		KNN	94.560 ± 0.000	0.000 ± 0.000	1.000 ± 0.000	0.000 ± 0.000	0.500 ± 0.000
		SVM	94.560 ± 0.000	0.000 ± 0.000	1.000 ± 0.000	0.000 ± 0.000	0.500 ± 0.000
YEAST-2	11:89	<i>GP_{edwb}</i>	85.420 ± 4.230	0.953 ± 0.036	0.842 ± 0.005	0.895 ± 0.024	0.897 ± 0.022
		GP_{std}	93.610 ± 1.750	0.630 ± 0.127	0.974 ± 0.011	0.780 ± 0.080	0.802 ± 0.064
		KNN	94.190 ± 1.190	0.573 ± 0.081	0.988 ± 0.005	0.750 ± 0.055	0.780 ± 0.042
		SVM	94.920 ± 1.260	0.694 ± 0.100	0.981 ± 0.007	0.823 ± 0.060	0.838 ± 0.050
YEAST-1	16 : 84	GP_{edwb}	51.820 ± 5.720	0.926 ± 0.045	0.438 ± 0.074	0.633 ± 0.044	0.682 ± 0.027
		GP_{std}	87.410 ± 1.280	0.392 ± 0.083	0.969 ± 0.015	0.613 ± 0.062	0.681 ± 0.038
		KNN	87.920 ± 1.540	0.494 ± 0.069	0.955 ± 0.014	0.685 ± 0.048	0.725 ± 0.034
		SVM	87.660 ± 1.340	0.366 ± 0.076	0.977 ± 0.012	0.595 ± 0.061	0.672 ± 0.037
CKD	27:73	GP_{edwb}	99.790 ± 0.790	0.993 ± 0.028	1.000 ± 0.000	0.996 ± 0.015	0.996 ± 0.014
		GP_{std}	99.380 ± 1.270	0.978 ± 0.045	1.000 ± 0.000	0.989 ± 0.023	0.989 ± 0.023
		KNN	83.020 ± 4.900	0.430 ± 0.148	0.987 ± 0.035	0.642 ± 0.110	0.708 ± 0.076
		SVM	78.120 ± 3.180	0.222 ± 0.113	1.000 ± 0.000	0.452 ± 0.138	0.611 ± 0.057
WDBC	37 : 63	GP_{edwb}	92.600 ± 2.920	0.955 ± 0.034	0.909 ± 0.044	0.931 ± 0.027	0.932 ± 0.026
		GP_{std}	93.650 ± 1.820	0.891 ± 0.037	0.963 ± 0.020	0.926 ± 0.021	0.927 ± 0.021
		KNN	91.400 ± 2.800	0.819 ± 0.076	0.969 ± 0.019	0.890 ± 0.041	0.894 ± 0.037
		SVM	90.640 ± 2.690	0.785 ± 0.082	0.977 ± 0.018	0.875 ± 0.043	0.881 ± 0.038

4.4 RESULTS

For each benchmark problem and each method, a total of 30 runs are performed. For these 30 runs, we calculated the mean values of classification specificity, sensitivity, accuracy, Gmean, and AUC values. Table-4.2 contains details of these experimental outcomes. In our experiment, we have taken results for best K values in the case of KNN. The experimental results are summarized in Table-4.2. This result table shows that even when the data set is highly unbalanced (ABL-9-18 & YEAST-2), the proposed approach gives a balanced performance. The proposed method performance difference is much higher when the Unbalanced ratio of the data set is high. When this Unbalanced ratio decreases, the difference between the proposed method with the standard GP method, SVM, and KNN decrease. However, in these cases, the proposed method also outperforms the other method on the scale of recall, G-mean and AUC.

Thus, the proposed method outperformed the SVM, KNN, and GP_{std} for four benchmark problems by generating better AUC, G-mean, and TPR values. On one Unbalanced problem, YEAST-1, the proposed method performs a bit lesser than KNN, but in this case, also, it outperforms the SVM and GP_{std} . This work aims to present the technique that gives unbiased and balanced performance over the majority and minority classes. We successfully achieved this, and our proposed method gives unbiased and balanced performance over the minority and majority classes. This balancing factor is the main reason behind a bit lower value for the 'specificity' of our proposed method. As the proposed fitness function gives the same emphasis to the minority class and the majority class, it nullifies the majority class sample's strong influence over the minority class samples during training. This identical weight assignment solves the issue of biasing during training and generates better classifiers. Again, by calculating the Euclidean distance, a classifier predicting close to the expected values gets more reward. This distance considered strategy resolves the issue of under-fitting. The outcome of this distance is that the proposed fitness function generates well-fitted classifiers. These are the main reasons which make the proposed technique outperform other techniques.

4.5 **SUMMARY**

Unbalanced data significantly challenges traditional classification algorithms. The standard algorithms may converge to a sub-optimal solution. In this work, we proposed a new fitness function for GP for unbalanced data classification. We applied the proposed method to five unbalanced classification problems and calculated different performance evaluation metrics. The presented method results are compared with SVM, KNN, and GP with standard fitness function. We found that our approach gives better or more competitive results than the GP with the standard fitness function, KNN, and SVM methods.

Table 4.3: KNN Grid Test Result for AUC values

K	ABLN-9-18	YEAST2	YEAST1	CKD	WDBC
1	0.61972522	0.798027087	0.688048607	0.76455314	0.89296847
2	0.643254126	0.864921389	0.754099119	0.744625604	0.934767607
3	0.668395689	0.895435736	0.782166859	0.776086957	0.944920622
4	0.687061133	0.909969655	0.799635177	0.757004831	0.952889505
5	0.702301665	0.931918236	0.80684541	0.737379227	0.959625865
6	0.705321723	0.934918277	0.819424615	0.725422705	0.960220713
7	0.711054398	0.940635333	0.82475227	0.712198068	0.960813904
8	0.696933587	0.940081365	0.830299711	0.719323671	0.962979124
9	0.694648451	0.942192001	0.833442369	0.738647343	0.967061109
10	0.699974361	0.943701275	0.833709472	0.751690821	0.967691062
11	0.714175321	0.946207962	0.835264257	0.74692029	0.970849548
12	0.735336004	0.948926061	0.839388749	0.736835749	0.971913934
13	0.73067339	0.949452137	0.843094758	0.736231884	0.97280599
14	0.727487159	0.952968704	0.843888523	0.722101449	0.973072695
15	0.726851191	0.952112886	0.845286098	0.712983092	0.972583379
16	0.72849529	0.951975232	0.844965506	0.698369565	0.973053318
17	0.731355419	0.95443793	0.844837955	0.70205314	0.973347492
18	0.740822578	0.953815671	0.84489213	0.695471014	0.973020422
19	0.736306669	0.953579147	0.84595797	0.682065217	0.974224232
20	0.727957982	0.955842814	0.847653678	0.63955314	0.973928964
21	0.728431671	0.956424726	0.848489789	0.6647343	0.973432477
22	0.722823516	0.957252078	0.847330686	0.674154589	0.973372427
23	0.723421283	0.957512403	0.847187706	0.674698068	0.973105244
24	0.725616574	0.956647756	0.847832318	0.700241546	0.973000882
25	0.728799867	0.955681099	0.849753813	0.703804348	0.972742885
26	0.724306437	0.959143095	0.85207099	0.701388889	0.974065344
27	0.725017484	0.962106367	0.854487087	0.694806763	0.973574165
28	0.729128539	0.962952822	0.854063632	0.678562802	0.974000907
29	0.722415723	0.965028216	0.854481944	0.675362319	0.974578165

CHAPTER 5

MULTI-OBJECTIVE GP APPROACH FOR CLASSIFICATION OF UNBALANCED DATA

This chapter briefly introduces multi-objective optimization and a well-known multi-objective evolutionary optimization technique NSGA-II. After that, the proposed methodology is provided, followed by the experimental setup and results. The last section summarizes the finding of the proposed method while generating Pareto-front.

5.1 INTRODUCTION

Standard classification algorithms give biased results when data sets are Unbalanced. Genetic Programming, a machine learning algorithm based on the evolution of species in nature, also suffers from the same issue. In the previous chapter, we have seen that by assigning equal weight to the minority class samples, with respect to the majority class samples, it is possible to nullify the biasing issue of standard GP. Also, by adding the Euclidean distance between predicted and expected class values, we can generate a better-fitted GP classifier. However, in that proposed approach assigning weight is *a priori* process, and only a single solution is generated in this standard GP method. Also, the classifier performances toward the majority and minority classes are conflicted as boosting the accuracy of one class depletes the performance of the other class [85, 103].

Multi-objective problems (MOP) refer to the problem where the solution quality is measured by its capability, referencing more than one possible conflicting objective. [45]. In the case of unbalanced data classification, the majority class accuracy and the minority class accuracy may be considered two separate conflicting objectives. Thus the Unbalanced classification problems may be considered as MOP. In the evolutionary multi-objective optimization (EMO) approach, a set of solutions, having a good trade-off between both class accuracies, is evolved. These trade-off solutions generate a Pareto-front in a single execution by keeping both class performance objectives distinct during the evolution. EMO has been applied in various realworld problems [7, 121]. These applications include but are not limited to medical [41, 106], finance [99], supersonic wing design [95], electric vehicle control [28], oil production planning [114], Scheduling problems [94, 111] and routing [125, 140]. The main advantage of the EMO approach is that it can produce a set of good-performing classifiers in a single experimental execution. Against the MOGP approach, the canonical GP approach requires multiple experi-

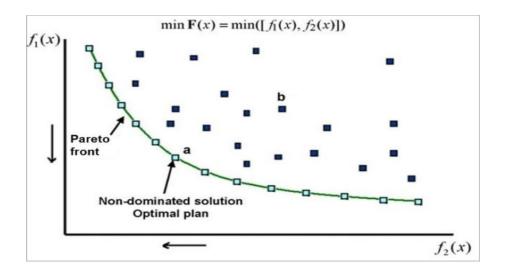


Figure 5.1: Pareto-front for a MOO Problem with 2 Objective

mental runs and a priori objective-based fitness function. Another advantage of MOGP is that it explicitly includes the learning bias into the algorithms.

5.2 MULTI-OBJECTIVE OPTIMIZATION

Multi-objective optimization refers to mathematical problems which involve more than one objective function to be optimized simultaneously [23]. This is also called multi-criteria optimization or Pareto-optimization. In the case of MOP, it is possible that a solution is better than other on one objective but lower or equal on other objectives. In this case, for comparing two solutions, the concept of 'dominance' comes in existence.

Dominance and Pareto Optimality

A solution dominates the other solution, if it is better than other for at least one objective and equal or greater than on other objectives. If we denote dominate symbol by \succeq . For *n* objectives, we define the solution *X* dominated to solution *Y* as following:

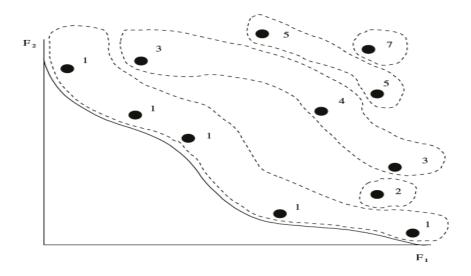


Figure 5.2: Dominance Rank

$$X \succeq Y \iff \exists i \in \{1, ..., n\}, x_i > y_i; \&\& \forall i \in \{1, ..., n\} \ x_i \ge y_i$$
 (5.1)

In case of conflicting objectives, no single solution exists which dominates all other solutions. Therefore, all non-dominated solutions have the property that their quality cannot be improved with respect to a given objective without affecting the other objective quality. Thus, in the search space, such solutions normally exist on the edge of the feasible regions, and the set of all these non-dominated solutions is known as the Pareto-front. Fig. 5.1, shows a Pareto-front for two objective minimization problem [40].

Based on this dominating property, below three important terms may be defined [29]:

- 1. dominance-rank: Number of solutions dominates the given solution.
- 2. dominance-count: Number of solutions dominated by the given solution.
- 3. dominance-depth: After performing dominance-based sorting, at which front the given solution belongs.

Fig. 5.2 and Fig. 5.3, show a graphical representation of these terms [29].

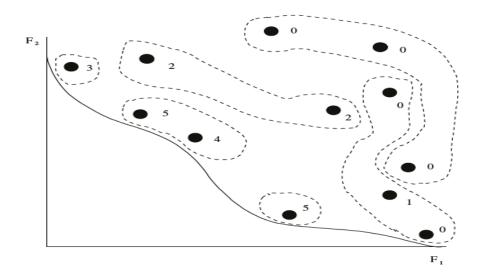


Figure 5.3: Dominance Count

Multi-objective Optimization Approaches

For Multi-objective Optimization problems various approaches based on evolutionary computation are presented by various researchers. In 1984, *Schaffer et el.* presented vector-evaluated genetic algorithm (VEGA), where the whole population is randomly divided into sub-populations, each assigned a fitness according to a distinct objective function. However selection and crossover are performed globally [107]. *Fonseca and Fleming*, proposed Multi-Objective Genetic Algorithm (MOGA) [49, 150] and *Horn et al.* proposed, Niche Pareto Genetic Algorithm (NPGA) [1, 60] based on dominance rank. *Zitzler et al.* [149, 148] introduced, Strength Pareto Evolutionary Algorithm (SPEA) based on dominance count and dominance rank. Again, *Srinivas and Deb* proposed Non-dominated Sorting Genetic Algorithm (NSGA), based the concept on dominance depth [117]. However, for M objectives and N population size, the complexity of NSGA becomes $O(M*N^3)$. Another drawback of this is its non-elitism approach and the requirement of defining a sharing parameter. To resolve these drawbacks, *Deb et al.* proposed NSGA-II, based on a fast non-dominated sorting approach [37]. The computational complexity of NSGA-II becomes $O(M*N^2)$. Authors show that, on most problems, NSGA-II can produce much better-diversified solutions and better convergence to the true optimal Pareto-front.

NSGA-II is the most prominent MOEA and is used for comparing any newly introduced MOEA technique. NSGA-II is applied to solve various research problems [64, 75, 90, 133, 136].

Bhowan et. el. (2013), proposed multi-objective Genetic Programming (MOGP) for the classification of an Unbalanced dataset [16]. In their work, the author uses the majority class accuracy and the minority class accuracy as two competing objectives during the training process. The authors utilized an existing Pareto-based optimization technique Non dominated sorting genetic algorithm II (NSGA-II) by Deb et al. (2002) [37]. For generating Pareto-front, NSGA-II, use dominance rank, which is the count of all solutions in population, dominating a given solution [139, 145]. Based on NSGA-II, Fernandes et al. (2019) proposed an ensemble of classifiers based on the multi-objective genetic sampling for Unbalanced classification (EMOSAIC) technique [46].

In this work, we presented a logarithmic distance-based MOGP approach to classifying Unbalanced data. The proposed approach utilizes the logarithmic value of the distance between predicted and expected values. This logarithmic values for the minority and the majority classes are treated as two separate objectives while learning. The aim is to minimize this logarithmic distance. In the final generation, the proposed approach generated a Pareto-front of classifiers with a balanced surface representing the majority and the minority class accuracies for binary classification.

In the next section, we will briefly discuss the NSGA-II technique and discuss our proposed work in the succeeding section.

5.3 NON-DOMINATED SORTING GENETIC ALGORITHM II (NSGA-II)

NSGA-II, proposed by Deb et. el. [37], has been a highly used multi-objective evolutionary optimization technique since its inception in literature. Almost every newly proposed MO-based method is compared with NSGA-II for evaluating its performance [62]. For efficient computational complexity, NSGA-II introduces a fast non-dominated sorting procedure, briefly described in section-5.3.

```
Algorithm 3 Fast non-dominated sorting
```

```
1: procedure FASTNONDOMINATEDSORT( P)
                                                                                                \triangleright P is population
 2:
         for p \in P do
              S_p \leftarrow \Phi
 3:
              n_p \leftarrow 0
 4:
              for each q \in P do
 5:
                   if p \prec q then
 6:
                       S_p = S_p \cup \{q\}
 7:
                   else
 8:
                       if q \prec p then
 9:
                            n_p \leftarrow n_p + 1
10:
              if n_p = 0 then
11:
12:
                  p_{rank} = 1
                  F_1 = F_1 \cup \{p\}
13:
         i = 1
14:
         while F_i \neq \Phi do
15:
              Q = \Phi
16:
              for each p \in F_i do
17:
                   for each q \in S_p do
18:
                       n_q \leftarrow n_q - 1
19:
                  if n_q = 0 then
20:
                       q_{rank} = i + 1
21:
                       Q = Q \cup \{q\}
22:
              i = i + 1
23:
              F_i = Q
24:
         return generated fronts F
25:
```

Fast non-dominated sorting

Algo. 3 implements the fast non-dominated sorting procedure. Here, for each individual two entities are calculated: domination-count (n_p) , and a set of individuals that the individual dom-

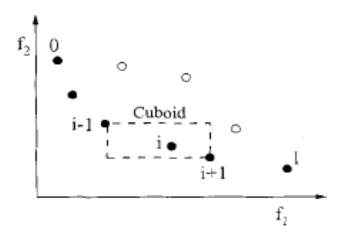


Figure 5.4: Calculation of Crowding-distance

inates (S_p) . All individuals, with $n_p = 0$, belongs to first non-dominated front. Now, for each individual p, belongs to this front, we visit each element q of its corresponding set S_p , and decrease its domination count by one. In this process, if for any element the domination count becomes zero, we assign this to the next non-dominated front (Q). Now, the same process is performed with each element of and this front (Q) and again next front is identified. We perform this procedure until all fronts are created.

Sustaining Diversity

Along with optimal front generation, maintaining diversity in Pareto-front is also a desired criteria [109]. For maintaining diversity, NSGA-II uses the concept of fitness sharing in the form of a crowded-comparison approach. For measuring crowded-factor (density factor) of an individual, the average distance of two individuals on either side of the given individual, is calculated. The overall distance is defined as the sum of individual crowding-distance, corresponding to each objective. Fig. 5.4, and Algo. 4 illustrate the calculation of the crowding distance process [37].

For diversified Pareto-front, the selection process is performed based on crowding-comparison (\prec). This process use non-dominance rank and crowding distance. The value of \prec is 1, if $[i_{rank} < j_{rank}]$, or $[(i_{rank} = j_{rank}) \& (i_{dist} > j_{dist})]$.

```
Algorithm 4 Crowding distance calculation
```

```
1: procedure CALCCROWDINGDIST(S_D)
                                                                                                \triangleright S_D is nondominated set
 2:
          n \leftarrow |S_D|
          for each i \in S_D do
 3:
               S_D[i]_{dist} \leftarrow 0
 4:
          for each o \in O do
                                                                                                        \triangleright O is objective set
 5:
               S_D \leftarrow sort(S_D, o)

    b do sorting w.r.f to each objectives

 6:
               S_D[1]_{dist} \leftarrow \infty
                                                                                           > set boundary values to infy
 7:
               S_D[n]_{dist} \leftarrow \infty
 8:
                                                    \Rightarrow f_o^{max}, f_0^{min} is max and min values of the o^{th} obj func
               for i \leftarrow 2 to (n-1) do
 9:
                    S_D[i]_{dist} \leftarrow S_D[i]_{dist} + (S_D[i+1]_{dist_m} + S_D[i-1]_{dist_m}) / (f_o^{max} - f_0^{min})
10:
```

NSGA-II Summary

The complete NSGA-II framework is presented as Algo. 5. GP parameters like population size, total generation, crossover, and mutation rates are defined at the start. The initial population (P) of programs is randomly generated. Each program, represented as a tree, contains various mathematical functions as intrinsic nodes and feature variables and random constants as a leaf node. The fitness of all programs with respect to each objective function is calculated, and non-dominance rank is defined based on this. A new offspring population (Q) of size N is generated using crossover and mutation. Selection is done based on the tournament selection process. In each generation, an intermediate population (R) is generated by combining P and Q. Thus, the size of R will become 2N. Here, all individuals are part of this intermediate population R, so elitism is ensured. Now R is sorted based on non-domination. Now all the solutions are filled in the next generation population as shown in Fig. 5.5. In this step, individual selections are performed based on the crowding-comparison operator.

```
Algorithm 5 NSGA-II Genetic Programming Algorithm
```

```
> X is input variables and T is constant
 1: procedure GENPROGNSGA-II(X,T)
         N \leftarrow popSize
                                                                                             ▶ Population size
 3:
         gTotal \leftarrow 100
                                                                                           ▶ Total Generation
        pCrssOvr \leftarrow 0.70
 4:
                                                                                      > crossover probability
         pMutn \leftarrow 0.30
 5:
                                                                                      P_{curr} \leftarrow generate N random programs
 7:
         gen \leftarrow 0
         while gen \neq gTotal do
 8:
             Calculate fitness of all programs in population, for each objective function.
 9:
                                                              \triangleright F = (F_1, F_2, ...) is set of non-dominated
             F \leftarrow FastNonDominatedSort(P_{curr})
10:
    fronts
             Q_{gen} \leftarrow Perform crossover and mutation, using tournament selection, and create a
11:
    offspring population of size N.
12:
             R_{gen} \leftarrow P_{curr} \cup Q_{gen}
             F \leftarrow FastNonDominatedSort(R_{gen})
13:
14:
             P_{new} \leftarrow \phi
             i \leftarrow 1
15:
             while |P_{new}| + |F_i| \le N do
16:
                  CalcCrowdingDist(F_i)
17:
                  P_{curr} \leftarrow P_{new} \cup F_i
18:
                  i \leftarrow i + 1
19:
             Sort(F_i, \prec)
20:
                             ⊳ sort in descending order using crowding-comparison operator.
             P_{new} \leftarrow P_{new} \cup F_i[1:N-|P_{new}|]
21:
22:
             P_{curr} \leftarrow P_{new}
             gen \leftarrow gen + 1
23:
         F \leftarrow FastNonDominatedSort(P_{curr})
24:
         return First Pareto-front F_1
25:
```

5.4 PROPOSED METHODOLOGY

In this section, we first briefly describe the Standard NSGA-II and introduce the proposed approach based on this.

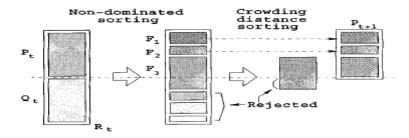


Figure 5.5: NSGA-II Procedure

Standard fitness function in NSGA-II

In the standard NSGA-II approach, the minority class accuracy $(F_{m_{std}})$ and the majority class accuracy $(F_{M_{std}})$ are taken as two objective fitness functions (Eq. 5.2 & Eq. 5.3).

$$F_{m_{std}} = \frac{TP}{TP + FN} \tag{5.2}$$

$$F_{M_{std}} = \frac{TN}{TN + FP} \tag{5.3}$$

where TN denotes *true negative*, TP denotes *true positive*, FN denotes *false negative* and FP denotes *false positive* prediction counts.

In binary classification, we choose a threshold value, generally 0.50. Any classifier output value greater than this is treated as one, and less than that is 0. In this case, if there are two classifiers, and for a given sample of the majority class (1), the first gives output 0.90, and the second gives output 0.60, both predictions will be treated as the majority class (1) and correct. As the first classifier gives forecast more closure to the expected value, in actuality, the 1^{st} one is better than the 2^{nd} one. In other words, the lower distance between the expected value and the predicted value is one inherent property of a good classifier. As the standard objective function considers only binary values, it does not satisfy this property. The ignorance of distance while calculating fitness is a drawback of this standard fitness function.

Proposed fitness function

To overcome the above shortcomings of the standard approach, we introduced a new fitness

function based on the distance among expected and predicted values. For that, let us define

classification distance for any prediction, as Eq. (5.4). We will use this classification distance

in our newly proposed fitness function.

 $dist = \left\| value_{expected} - value_{predicted} \right\|$ (5.4)

Where,

*value*_{expected}: binary value 0 or 1.

value_{predicted}: value predicted by classifier.

A GP program represents a mathematical formula that is nothing but a polynomial. This

polynomial consists of variables (input feature variables) and coefficients, which involves addi-

tion, subtraction, multiplication, and integer exponentiation of variables. For any data sample,

the value of the polynomial, represented by the GP program, is taken as the predicted value for

that sample. Depending on generated polynomial, its value may vary between $-\infty$ to $+\infty$. We

subtract this predicted value with the expected value and take the absolute value of this, called

classification-distance. Hence, the above classification distance values will range between 0

to ∞. Therefore, directly using this classification distance in the fitness function has an issue

illustrated in Fig. 5.6.

In this first plotted figure y-axis represent the classification distance for different data

samples. Some of the classification distance values are very high in this plot and supersede

the other classification distance of prediction. Consequently, when summing up these values,

it nullifies the even correctly classified sample's contribution. Therefore, these values need to

be bounded. To overcome this issue, we take logarithmic classification distance. Since the

value of classification distance is always zero or greater than zero, and the logarithmic value

of any number less than 1 is undefined, we add 1.00 to this classification distance value before

calculating the logarithmic value. We plot this logarithmic distance in the second part of this

63

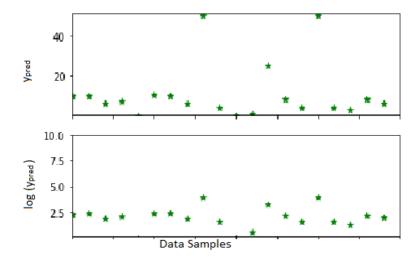


Figure 5.6: Euclidean-distance vs Logarithmic-distance for Different Data-samples

figure. Fig. 5.6 shows that even these larger classification distance contribution is still high, but it would not nullify the others correctly predicted or near correctly predicted values during summation.

Now, to resolve the standard approach issue, in this work, we proposed a logarithmic distance-based MOGP approach, in which we have defined two objective functions as Equ. 5.5 and Equ. 5.6.

$$F_{m_{prop}} = \sum_{i=1}^{N_{min}} \frac{\log(1 + |dist_{mini}|)}{N_{min}}$$
 (5.5)

$$F_{M_{prop}} = \sum_{i=1}^{N_{maj}} \frac{\log\left(1 + \left| dist_{maj_i} \right| \right)}{N_{maj}}$$
(5.6)

Where,

N_{min}: Minority samples count in training data.

N_{mai}: Majority samples count in training data.

 $\textit{dist}_{\text{maj }i}$: Distance of actual value from predicted value in i^{th} majority class sample.

 $\textit{dist}_{\min i}$: Distance between expected value and predicted value in i^{th} minority class sam-

ple.

Thus, the $F_{m_{prop}}$ replace the $F_{m_{std}}$ and $F_{M_{prop}}$ replace the $F_{M_{std}}$, in the standard MOGP framework. By introducing this distance factor in our proposed approach, we are targeting the better-fitted classifiers. After completing the GP run, we recalculate Pareto-front based on the minority class accuracy and the majority class accuracy. We will measure the hyper-are of the generated front using the Trapezoidal rule described in the next section.

Trapezoidal rule for the area under generated Pareto-front

The trapezoidal rule (TR) is utilized for calculating the hyper-area under generated Pareto-front curve. The TR divides the curve into small trapezoids and performs the summation of all the areas of the small trapezoids to calculate the total area under the curve [102].

Let y = f(x) be continuous function on interval [a, b]. We can divide the interval [a, b] into n equal sub-intervals, each of width h (Equ. 5.7 and Equ. 5.8).

$$a = x_0 < x_1 < x_2 < \dots < x_n = b \tag{5.7}$$

$$h = \frac{(b-a)}{n} \tag{5.8}$$

Now, the area under the Pareto-front curve is defined as Eq. 5.9.

$$\int_{a}^{b} f(x)dx \approx \frac{h}{2} * \left[f(x_0) + 2 * \sum_{i=1}^{n-1} f(x) + f(x_n) \right]$$
 (5.9)

5.5 EXPERIMENTAL SETUP

The proposed approach and the standard approach are implemented in Python. We use Python 3.6.6, installed on Microsoft Windows 10, powered by an Intel Core i7 processor with 32 GB RAM. In this work, we use *StratifiedShuffleSplit*. For each benchmark problem, 30 runs of standard MOGP and proposed MOGP approaches are executed. In every run, the hyper-area of generated Pareto-front is calculated [104]. We have evaluated the performance on four datasets: ABL-9-18, YEAST-2, YEAST-1, and WDBC.

For GP population initialization, we used the "ramped half and half" method. This method generates a wide variety of individuals of various sizes, and shapes [71]. The tree initialization depth is taken as (2,6). The function set contains +,-, *, /, log, sin and cos functions. The crossover and mutation probability is set to 0.60 and 0.40, respectively. Population size is set to 512, and max generation is set to 50. We have use tournament selection, with tournament size 2. For simplicity, sub-tree crossover, sub-tree mutation, and tournament selection with tournament size two are taken. These GP parameter values are given in Table-5.1.

Table 5.1: MOGP Parameter Setting

Parameter	Value	
population initialization	ramp half and half	
population size	512	
maximum generation	50	
max tree height	6	
min tree height	2	
function-set	+,-, *, /, log, sin and cos	
tree terminals	feature variables & random constants	
mutation	0.20	
crossover	0.70	

Table 5.2: Generated Pareto-front Hyper-area by Proposed MOGP Method

Data-set name	Unbalanced	Front Hyper Area	Front Hyper Area
	Factor	GP_{prop}	GP_{std}
ABLN-9-18	06:94	0.844 ± 0.031	0.651 ± 0.033
Yeast-2	11:89	0.938 ± 0.009	0.833 ± 0.042
Yeast-1	16:84	0.816 ± 0.006	0.804 ± 0.032
WDBC	37:63	0.959 ± 0.008	0.120 ± 0.030

5.6 RESULTS

For each benchmark problems, a total of 30 runs of, standard MOGP, and proposed MOGP approaches, are executed. In the case of MOO algorithms, the diversification of solutions on generated Pareto-front and hyper area under the Pareto-front is the key measure of how a given algorithm is performing. In every run, the hyper-area of generated Pareto-front is calculated. For calculating hyper area under generated Pareto-front we have used trapezoidal rule [102]. The trapezoidal rule is a technique to approximate the integral of a curve (area), on a given interval [a, b]. The mean and standard deviation of these 30 runs for each benchmark problem is calculated. In the case of the standard MOGP method [16], for ABL-9-18, YEAST2, YEAST1, and WDBC datasets, the generated front hyper-area are 0.651, 0.833, 0.804, and 0.120, respectively. Against this, in the case of the proposed logarithmic-distance-based MOGP method, the generated front for ABL-9-18, YEAST2, YEAST1, and WDBC datasets hyper-area are 0.844, 0.938, 0.816, and 0.959, respectively. Generated Pareto-front for each Unbalanced problem are plotted in Fig. 5.7, Fig. 5.8, Fig. 5.9, and Fig. 5.10. The left part of the figure shows the generated front for the proposed MOGP approach, and the right-part of the figure shows the generated front for the standard MOGP approach.

For all the four benchmark problems, the generated front by the proposed method outperforms the generated front by the standard MOGP method. Fig. 5.7, Fig. 5.8, Fig. 5.9, and Fig. 5.10, clearly show the visual of this conclusion. The mean value of generated Pareto-front is summarized in Tab. 5.2. Thus the proposed logarithmic distance-based MOGP method beats the standard MOGP method [16] in all considered Unbalanced problems.

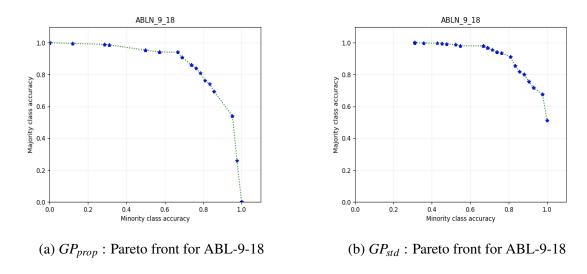


Figure 5.7: Generated Pareto-front by MOGP for ABL-9-18

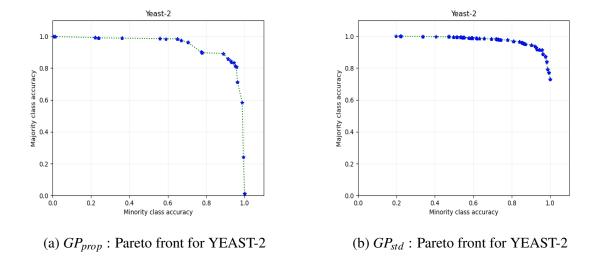


Figure 5.8: Generated Pareto-front by MOGP for YEAST-2

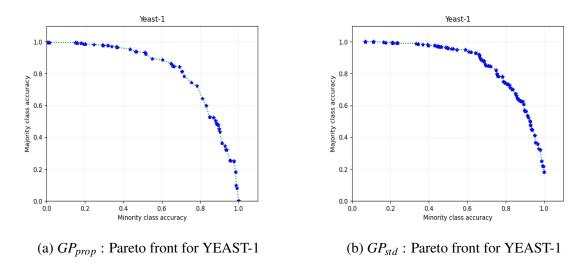


Figure 5.9: Generated Pareto-front by MOGP for YEAST-1

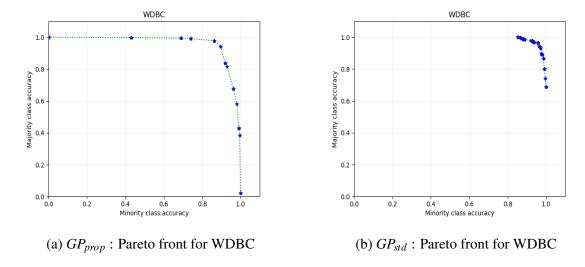


Figure 5.10: Generated Pareto-front by MOGP for WDBC

5.7 SUMMARY

Many of the real-world classification problems are Unbalanced, and this Unbalanced nature of data highly influences the performance of standard AI techniques. GP also suffers from a similar issue. This research work introduced a logarithmic distance-based multi-objective genetic programming approach for classifying Unbalanced data. In the case of MOO algorithms, the diversification of solutions on generated Pareto-front and hyper area under the Pareto-front is a key measure of how a given algorithm is performing. We plotted the generated Pareto-front and calculated its hyper-area for each imbalanced problem. Our proposed MOGP method generates a better Pareto-front, in the form of diversified solutions and higher hyper-area, compared to the standard MOGP method. Based on our extensive experimentation results, we can conclude that the proposed logarithmic distance-based MOGP approach outperforms the standard MOGP method.

CHAPTER 6 SUMMARY AND CONCLUSIONS

In this research work, we have briefly summarized the unbalanced data classification, challenges to tackle unbalanced data classification and review the literature related to unbalanced classification. Unbalanced data classification got significant attention in the field of data classification. Most classification algorithms have shown biased results inclined toward the majority class. The related literature for handling unbalanced data classification is categorized into three groups: data level, algorithm level, and the ensemble of classifiers. In the data level approach, the external method focuses on creating a trade-off between the majority and the minority class samples count. This trade-off is achieved either by over-sampling minority class data points or under-sampling of majority class data points. Ensemble-based techniques focus on adopting the technology of multiple classifiers to improve the performance of a single classifier and predict.

We have focused on the internal approach to handle unbalanced data classification issues, i.e., the algorithm level approach. Genetic programming (GP) is a nature-inspired optimization algorithm that also suffers from unbalanced data classification. In GP, we have focused on adapting the new fitness function to reward accurate solutions on both the minority and the majority classes.

One approach to resolve the unbalanced classification issue is to assign a custom high weight to the minority class during training. The GP fitness function may introduce the custom weight assignment for the minority class samples. We assess the impact of the weight factor on GP's fitness function for Unbalanced data classification. We performed intensive experimentation on the different weight factors of the minority class in GP's fitness function. Based on experimental results, we can conclude that for better handling of Unbalanced classification problems in a generic manner, we can set the weight value to 0.5. This custom weight assignment for the minority class samples during training generated better solutions, which tackle the challenges of the Unbalanced data classification problems.

GP with the standard fitness function does not emphasize how close the predicted values are compared to the expected values, leading to an under-fitted classifier. We focus on the distance between the predicted and the actual outcomes. We combined the custom weight and distance factors and proposed a new fitness function to handle unbalanced data classification. The proposed approach does not require any domain-specific knowledge when applied for unbalanced classification work. We found that the proposed fitness function gives better results than the standard GP's fitness function based on extensive experimentation.

Evolutionary multi-objective optimization (EMO) is also gaining popularity in research, facilitating learning with multiple conflicting objectives. EMO achieves this by treating the objectives independently in the learning procedure using the Pareto Dominance concept in fitness. Against the MOGP approach, the standard approach requires multiple experimental runs and a priori objective-based fitness function. Another advantage of multi-objective genetic programming (MOGP) is that it explicitly includes the learning bias into the algorithms. We proposed a logarithmic distance-based MOGP approach to classifying unbalanced data. The suggested logarithmic distance-based MOGP approach results have proved its superiority over the traditional MOGP method, in the form of better Pareto front and higher hyper area under generated Pareto front.

In this research work, we also compared the performance of proposed approaches with standard GP and other classification techniques, namely SVM and KNN. The comparison clearly shows that the proposed strategies either outperform or give comparative results. One positive aspect of the proposed method is that this outperforms other techniques even without prepossessing or feature extraction.

Future Work

In this work, our scope is to focus on binary classification only. However, in future work, we will focus on multi-class imbalanced classification. We found that the GP performs efficiently and very well for binary classification, But applying GP for multi-class classification is more complicated. There is one majority class in multi-class imbalanced classification while multiple minority classes (vice-versa is also possible). Therefore against other classification methods like KNN, SVM, and Decision-Trees (DT), in GP, a careful and appropriate approach needs to be taken to decide how a GP classifier's continuous numeric output is mapped to various classes labels. Thus, in the case of multi-class classification, the most critical aspect is designing boundary criteria that separate the various classes. This process may be done either by a priori defining fixed class boundaries for numeric GP classifier's output or by deciding class boundaries at run time on a solution-by-solution basis.

Thus, founding class boundaries for separating the various classes may be challenging,

mainly when the number of classes is high. Another possible approach is to also apply the proposed binary classification method to multi-class classification problems. In this approach, the multi-class classification problems may be converted to binary class classification problems by selecting one-versus-all. Therefore, N binary classifiers will be trained for an N-class multi-classification problem. However, in the final step, aggregation of these binary classifiers requires a thorough investigation.

Another future scope of this work includes the extension of the MOGP approach. In this research work, we considered NSGA-II as a base method. Another strategy may be to use SPEA-2 as the base method and compare the performance between NSGA-II and SPEA-2.

REFERENCES

- [1] M. Abido. A niched pareto genetic algorithm for multiobjective environmental/economic dispatch. *International journal of electrical power & energy systems*, 25(2):97–105, 2003.
- [2] D. Acharya, A. Billimoria, N. Srivastava, S. Goel, and A. Bhardwaj. Emotion recognition using fourier transform and genetic programming. *Applied Acoustics*, 164:107260, 2020.
- [3] D. Acharya, S. Goel, R. Asthana, and A. Bhardwaj. A novel fitness function in genetic programming to handle unbalanced emotion recognition data. *Pattern Recognition Letters*, 2020.
- [4] I. Albisua, O. Arbelaitz, I. Gurrutxaga, A. Lasarguren, J. Muguerza, and J. M. Pérez. The quest for the optimal class distribution: an approach for enhancing the effectiveness of learning via resampling methods for imbalanced data sets. *Progress in Artificial Intelligence*, 2(1):45–63, 2013.
- [5] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.
- [6] M. Awad and R. Khanna. Support vector machines for classification. In *Efficient Learning Machines*, pages 39–66. Springer, 2015.
- [7] R. Azzouz, S. Bechikh, and L. B. Said. Dynamic multi-objective optimization using evolutionary algorithms: a survey. In *Recent advances in evolutionary multi-objective optimization*, pages 31–70. Springer, 2017.
- [8] R. Barandela, J. S. Sánchez, V. Garca, and E. Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851, 2003.

- [9] S. Barua, M. M. Islam, X. Yao, and K. Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2014.
- [10] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.
- [11] C. Beyan and R. Fisher. Classifying imbalanced data sets using similarity based hierarchical decomposition. *Pattern Recognition*, 48(5):1653–1672, 2015.
- [12] H.-G. Beyer and H.-P. Schwefel. Evolution strategies—a comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.
- [13] A. Bhardwaj, A. Tiwari, H. Bhardwaj, and A. Bhardwaj. A genetically optimized neural network model for multi-class classification. *Expert Systems with Applications*, 60:211–221, 2016.
- [14] A. Bhardwaj, A. Tiwari, R. Krishna, and V. Varma. A novel genetic programming approach for epileptic seizure detection. *Computer methods and programs in biomedicine*, 124:2–18, 2016.
- [15] U. Bhowan, M. Johnston, and M. Zhang. Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):406–421, 2012.
- [16] U. Bhowan, M. Johnston, M. Zhang, and X. Yao. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation*, 17(3):368–386, 2013.
- [17] U. Bhowan, M. Zhang, and M. Johnston. Genetic programming for image classification with unbalanced data. In 2009 24th International Conference Image and Vision Computing New Zealand, pages 316–321. IEEE, 2009.
- [18] P. Branco, L. Torgo, and R. P. Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):31, 2016.
- [19] H. J. Bremermann et al. Optimization through evolution and recombination. *Self-organizing systems*, 93:106, 1962.

- [20] L. A. Bugnon, C. Yones, D. H. Milone, and G. Stegmayer. Deep neural architectures for highly imbalanced data in bioinformatics. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.
- [21] M. J. Cavaretta and K. Chellapilla. Data mining using genetic programming: The implications of parsimony on generalization error. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1330–1337. IEEE, 1999.
- [22] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.
- [23] K.-H. Chang. Chapter 5 multiobjective optimization and advanced topics. In K.-H. Chang, editor, *Design Theory and Methods Using CAD/CAE*, pages 325–406. Academic Press, Boston, 2015.
- [24] S. Chatterjee, S. Dzitac, S. Sen, N. C. Rohatinovici, N. Dey, A. S. Ashour, and V. E. Balas. Hybrid modified cuckoo search-neural network in chronic kidney disease classification. In 2017 14th International Conference on Engineering of Modern Electric Systems (EMES), pages 164–167. IEEE, 2017.
- [25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
- [26] B. Chen, S. Xia, Z. Chen, B. Wang, and G. Wang. Rsmote: A self-adaptive robust smote for imbalanced problems with label noise. *Information Sciences*, 553:397–428, 2021.
- [27] G. Cheng, J. Han, and X. Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.
- [28] R. Cheng, T. Rodemann, M. Fischer, M. Olhofer, and Y. Jin. Evolutionary many-objective optimization of hybrid electric vehicle control: From general optimization to preference articulation. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(2):97–111, 2017.
- [29] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

- [30] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.
- [31] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, EC-14(3):326–334, 1965.
- [32] L. Cuadros-Rodríguez, E. Pérez-Castaño, and C. Ruiz-Samblás. Quality performance metrics in multivariate classification methods for qualitative analysis. *TrAC Trends in Analytical Chemistry*, 80:612–624, 2016.
- [33] P. Cunningham, M. Cord, and S. J. Delany. *Supervised learning*, pages 21–49. Springer, 2008.
- [34] R. Curry, P. Lichodzijewski, and M. I. Heywood. Scaling genetic programming to large datasets using hierarchical dynamic subset selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(4):1065–1073, 2007.
- [35] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi. Credit card fraud detection: a realistic modeling and a novel learning strategy. *IEEE transactions on neural networks and learning systems*, 29(8):3784–3797, 2017.
- [36] S. Das, S. Datta, and B. B. Chaudhuri. Handling data irregularities in classification: Foundations, trends, and future challenges. *Pattern Recognition*, 81:674–693, 2018.
- [37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.
- [38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.
- [39] D. Devarriya, C. Gulati, V. Mansharamani, A. Sakalle, and A. Bhardwaj. Unbalanced breast cancer data classification using novel fitness functions in genetic programming. *Expert Systems with Applications*, 140:112866, 2020.

- [40] M. Di Somma. Optimal operation planning of distributed energy systems through multiobjective approach: A new sustainability-oriented pathway. PhD thesis, Università di Napoli Federico II, 2016.
- [41] L. Dioşan and A. Andreica. Multi-objective breast cancer classification by using multi-expression programming. *Applied Intelligence*, 43(3):499–511, 2015.
- [42] D. Dua and C. Graff. UCI machine learning repository, 2017.
- [43] D. Dua and C. Graff. Uci machine learning repository. "https://archive.ics.uci.edu/ml/datasets.php", 2017.
- [44] A. Duan, L. Guo, H. Gao, X. Wu, and X. Dong. Deep focus parallel convolutional neural network for imbalanced classification of machinery fault diagnostics. *IEEE Transactions on Instrumentation and Measurement*, 69(11):8680–8689, 2020.
- [45] A. E. Eiben, J. E. Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.
- [46] E. Fernandes, A. C. P. de Leon Ferreira, D. Carvalho, and X. Yao. Ensemble of classifiers based on multiobjective genetic sampling for imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 2019.
- [47] L. J. Fogel. Artificial intelligence through a simulation of evolution. In *Proc. of the 2nd Cybernetics Science Symp.*, 1965, 1965.
- [48] B. Fogel David. Evolutionary computation: The fossil record, 1998.
- [49] C. M. Fonseca, P. J. Fleming, et al. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Icga*, volume 93, pages 416–423. Citeseer, 1993.
- [50] E. Galván-López, L. Vázquez-Mendoza, and L. Trujillo. Stochastic semantic-based multi-objective genetic programming optimisation for classification of imbalanced data. In *Mexican International Conference on Artificial Intelligence*, pages 261–272. Springer, 2016.

- [51] X. Gao, B. Ren, H. Zhang, B. Sun, J. Li, J. Xu, Y. He, and K. Li. An ensemble imbalanced classification method based on model dynamic selection driven by data partition hybrid sampling. *Expert Systems with Applications*, 160:113660, 2020.
- [52] Z. Geler, V. Kurbalija, M. Ivanović, and M. Radovanović. Weighted knn and constrained elastic distances for time-series classification. *Expert Systems with Applications*, 162:113829, 2020.
- [53] X. Gu, T. Ni, and H. Wang. New fuzzy support vector machine for the class imbalance problem in medical datasets classification. *The scientific world journal*, 2014, 2014.
- [54] D. Guan and W. Yuan. A survey of mislabeled training data detection techniques for pattern classification. *IETE Technical Review*, 30(6):524–530, 2013.
- [55] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.
- [56] W. Han, Z. Huang, S. Li, and Y. Jia. Distribution-sensitive unbalanced data oversampling method for medical diagnosis. *Journal of medical Systems*, 43(2):39, 2019.
- [57] E. M. Hassib, A. I. El-Desouky, L. M. Labib, and E.-S. M. El-kenawy. Woa+ brnn: An imbalanced big data classification framework using whale optimization and deep neural network. *soft computing*, 24(8):5573–5592, 2020.
- [58] T. Hastie, R. Tibshirani, and J. Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.
- [59] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM journal on computing*, 2(2):88–105, 1973.
- [60] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, pages 82–87. Ieee, 1994.

- [61] W.-h. Hou, X.-k. Wang, H.-y. Zhang, J.-q. Wang, and L. Li. A novel dynamic ensemble selection classifier for an imbalanced data set: An application for credit risk assessment. *Knowledge-Based Systems*, 208:106462, 2020.
- [62] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima. Performance comparison of nsgaii and nsga-iii on various many-objective test problems. In 2016 IEEE Congress on Evolutionary Computation (CEC), pages 3045–3052. IEEE, 2016.
- [63] A. Jain, S. Ratnoo, and D. Kumar. A novel multi-objective genetic algorithm approach to address class imbalance for disease diagnosis. *International Journal of Information Technology*, May 2020.
- [64] J. Jemai, M. Zekri, and K. Mellouli. An nsga-ii algorithm for the green vehicle routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 37–48. Springer, 2012.
- [65] R. Johansson. NSGA-II design for feature selection in EEG classification related to motor imagery. PhD thesis, School Of Innovation, Design And Engineering Västerås, Mälardalen University, Sweden, 2020.
- [66] A. Kalantari, A. Kamsin, S. Shamshirband, A. Gani, H. Alinejad-Rokny, and A. T. Chronopoulos. Computational intelligence approaches for classification of medical data: State-of-the-art, future challenges and research directions. *Neurocomputing*, 276:2–22, 2018.
- [67] Q. Kang, L. Shi, M. Zhou, X. Wang, Q. Wu, and Z. Wei. A distance-based weighted undersampling scheme for support vector machines and its application to imbalanced classification. *IEEE transactions on neural networks and learning systems*, 29(9):4152–4165, 2017.
- [68] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587, 2017.
- [69] J. Kim, J. Jeong, and J. Shin. M2m: Imbalanced classification via major-to-minor translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13896–13905, 2020.

- [70] G. Kovács. An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, 83:105662, 2019.
- [71] J. Koza. On the programming of computers by means of natural selection. *Genetic Programming*, 1992.
- [72] J. R. Koza. Human-competitive results produced by genetic programming. *Genetic programming and evolvable machines*, 11(3-4):251–284, 2010.
- [73] F. Li, X. Zhang, X. Zhang, C. Du, Y. Xu, and Y.-C. Tian. Cost-sensitive and hybrid-attribute measure multi-decision tree over imbalanced data sets. *Information Sciences*, 422:242–256, 2018.
- [74] L. Li, H. He, and J. Li. Entropy-based sampling approaches for multi-class imbalanced problems. *IEEE Transactions on Knowledge and Data Engineering*, 2019.
- [75] Y. Li, S. Wang, X. Duan, S. Liu, J. Liu, and S. Hu. Multi-objective energy management for atkinson cycle engine and series hybrid electric vehicle based on evolutionary nsga-ii algorithm using digital twins. *Energy Conversion and Management*, 230:113788, 2021.
- [76] Z. Li, T. Zheng, Y. Wang, Z. Cao, Z. Guo, and H. Fu. A novel method for imbalanced fault diagnosis of rotating machinery based on generative adversarial networks. *IEEE Transactions on Instrumentation and Measurement*, 70:1–17, 2020.
- [77] A. Liaw, M. Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.
- [78] M. Lin, K. Tang, and X. Yao. Dynamic sampling approach to training neural networks for multiclass imbalance classification. *IEEE Transactions on Neural Networks and Learn*ing Systems, 24(4):647–660, 2013.
- [79] V. López, S. Del Río, J. M. Benítez, and F. Herrera. Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data. *Fuzzy Sets and Systems*, 258:5–38, 2015.

- [80] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585–6608, 2012.
- [81] O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto. Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases. *Neurocomputing*, 175:935–947, 2016.
- [82] C. Lu, H. Ke, G. Zhang, Y. Mei, and H. Xu. An improved weighted extreme learning machine for imbalanced data classification. *Memetic Computing*, 11(1):27–34, 2019.
- [83] P. Luukka. A new nonlinear fuzzy robust pca algorithm and similarity classifier in classification of medical data sets. *International Journal of Fuzzy Systems*, 13(3), 2011.
- [84] Y. Ma and G. Guo. Support vector machines applications, volume 649. Springer, 2014.
- [85] H. H. Maheta and V. K. Dabhi. Classification of imbalanced data sets using multi objective genetic programming. In 2015 International Conference on Computer Communication and Informatics (ICCCI), pages 1–6. IEEE, 2015.
- [86] S. Maldonado and J. López. Dealing with high-dimensional class-imbalanced datasets: Embedded feature selection for svm classification. *Applied Soft Computing*, 67:94–105, 2018.
- [87] C. J. Mantas and J. Abellan. Credal-c4. 5: Decision tree based on imprecise probabilities to classify noisy data. *Expert Systems with Applications*, 41(10):4625–4637, 2014.
- [88] D. Mateos-García, J. García-Gutiérrez, and J. C. Riquelme-Santos. On the evolutionary optimization of k-nn by label-dependent feature weighting. *Pattern Recognition Letters*, 33(16):2232–2238, 2012.
- [89] M. Matheny, S. T. Israni, M. Ahmed, and D. Whicher. Artificial intelligence in health care: the hope, the hype, the promise, the peril. *NAM Special Publication. Washington, DC: National Academy of Medicine*, page 154, 2019.
- [90] H. Morais, T. Sousa, R. Castro, and Z. Vale. Multi-objective electric vehicles scheduling using elitist non-dominated sorting genetic algorithm. *Applied Sciences*, 10(22):7978, 2020.

- [91] S. S. Mullick, S. Datta, S. G. Dhekane, and S. Das. Appropriateness of performance indices for imbalanced data classification: An analysis. *Pattern Recognition*, 102:107197, 2020.
- [92] D. P. Muni, N. R. Pal, and J. Das. A novel approach to design classifiers using genetic programming. *IEEE transactions on evolutionary computation*, 8(2):183–196, 2004.
- [93] K. Nag and N. R. Pal. A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification. *IEEE transactions on cybernetics*, 46(2):499–510, 2016.
- [94] S. Nguyen, M. Zhang, and K. C. Tan. Enhancing genetic programming based hyperheuristics for dynamic multi-objective job shop scheduling problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2781–2788. IEEE, 2015.
- [95] S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose. Multiobjective evolutionary computation for supersonic wing-shape optimization. *IEEE transactions on evolutionary computation*, 4(2):182–187, 2000.
- [96] H. Patel and G. S. Thakur. Classification of imbalanced data using a modified fuzzy-neighbor weighted approach. *International Journal of Intelligent Engineering and Systems*, 10(1):56–64, 2017.
- [97] W. Pei, B. Xue, L. Shang, and M. Zhang. Genetic programming for high-dimensional imbalanced classification with a new fitness function and program reuse mechanism. *Soft Computing*, pages 1–18, 2020.
- [98] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza. *A field guide to genetic programming*. Lulu. com, 2008.
- [99] A. Ponsich, A. L. Jaimes, and C. A. C. Coello. A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications. *IEEE Transactions on Evolutionary Computation*, 17(3):321–344, 2012.
- [100] A. Ragalo and N. Pillay. Evolving dynamic fitness measures for genetic programming. *Expert Systems with Applications*, 109:162–187, 2018.

- [101] B. S. Raghuwanshi and S. Shukla. Minimum class variance class-specific extreme learning machine for imbalanced classification. *Expert Systems with Applications*, 178:114994, 2021.
- [102] Q. I. Rahman and G. Schmeisser. Characterization of the speed of convergence of the trapezoidal rule. *Numerische Mathematik*, 57(1):123–138, 1990.
- [103] V. H. A. Ribeiro and G. Reynoso-Meza. Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets. *Expert Systems with Applications*, 147:113232, 2020.
- [104] N. Riquelme, C. Von Lücken, and B. Baran. Performance metrics in multi-objective optimization. In 2015 Latin American computing conference (CLEI), pages 1–11. IEEE, 2015.
- [105] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera. Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, 291:184–203, 2015.
- [106] Ç. Sarvan and N. Özkurt. Multi-objective advisory system for arrhytmia classification. Biomedical Signal Processing and Control, 69:102838, 2021.
- [107] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the first international conference on genetic algorithms and their applications*, 1985. Lawrence Erlbaum Associates. Inc., Publishers, 1985.
- [108] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* Adaptive Computation and Machine Learning Series, 2018.
- [109] O. Schütze and C. Hernández. Computing the entire pareto front. In *Archiving Strategies* for Evolutionary Multi-objective Optimization Algorithms, pages 27–39. Springer, 2021.
- [110] Z. Seng, S. A. Kareem, and K. D. Varathan. A neighborhood undersampling stacked ensemble (nus-se) in imbalanced classification. *Expert Systems with Applications*, 168:114246, 2021.

- [111] X.-N. Shen and X. Yao. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Information Sciences*, 298:198–224, 2015.
- [112] Q. Shi and H. Zhang. Fault diagnosis of an autonomous vehicle with an improved sym algorithm subject to unbalanced datasets. *IEEE Transactions on Industrial Electronics*, 2020.
- [113] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [114] H. K. Singh, T. Ray, and R. Sarker. Optimum oil production planning using infeasibility driven evolutionary algorithm. *Evolutionary Computation*, 21(1):65–82, 2013.
- [115] R. Singh, T. Ahmed, A. Kumar, A. K. Singh, A. K. Pandey, and S. K. Singh. Imbalanced breast cancer classification using transfer learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.
- [116] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. A linear genetic programming approach to intrusion detection. In *Genetic and Evolutionary Computation Conference*, pages 2325–2336. Springer, 2003.
- [117] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.
- [118] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv. Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE transactions on cybernetics*, 50(9):3840–3854, 2020.
- [119] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. MIT press, 2018.
- [120] M. A. U. H. Tahir, S. Asghar, A. Manzoor, and M. A. Noor. A classification model for class imbalance dataset using genetic programming. *IEEE Access*, 7:71013–71037, 2019.
- [121] R. Tanabe and H. Ishibuchi. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing*, 89:106078, 2020.

- [122] B. Tang and H. He. Gir-based ensemble sampling approaches for imbalanced learning. *Pattern Recognition*, 71:306–319, 2017.
- [123] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* (*Cybernetics*), 39(1):281–288, 2009.
- [124] X. Tao, W. Chen, X. Li, X. Zhang, Y. Li, and J. Guo. The ensemble of density-sensitive svdd classifier based on maximum soft margin for imbalanced datasets. *Knowledge-Based Systems*, 219:106897, 2021.
- [125] E. B. Tirkolaee, A. Goli, A. Faridnia, M. Soltani, and G.-W. Weber. Multi-objective optimization for the reliable pollution-routing problem with cross-dock selection using pareto-based algorithms. *Journal of Cleaner Production*, 276:122927, 2020.
- [126] B. Tran, B. Xue, and M. Zhang. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing*, 8(1):3–15, 2016.
- [127] A. Turing. Intelligent machinery: a report. London: National Physical Laboratory, 1948.
- [128] J. Van Hulse and T. Khoshgoftaar. Knowledge discovery from imbalanced and noisy data. *Data & Knowledge Engineering*, 68(12):1513–1542, 2009.
- [129] S. Vijayarani, S. Dhayanand, et al. Data mining classification algorithms for kidney disease prediction. *International Journal on Cybernetics & Informatics (IJCI)*, 4(4):13–25, 2015.
- [130] P. Vuttipittayamongkol, E. Elyan, and A. Petrovski. On the class overlap problem in imbalanced data classification. *Knowledge-based systems*, page 106631, 2020.
- [131] P. Vuttipittayamongkol, E. Elyan, and A. Petrovski. On the class overlap problem in imbalanced data classification. *Knowledge-based systems*, 212:106631, 2021.
- [132] S. Wang and X. Yao. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130, 2012.

- [133] S. Wang, D. Zhao, J. Yuan, H. Li, and Y. Gao. Application of nsga-ii algorithm for fault diagnosis in power system. *Electric Power Systems Research*, 175:105893, 2019.
- [134] M. Wasikowski and X.-w. Chen. Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on knowledge and data engineering*, 22(10):1388–1400, 2009.
- [135] J. Wei, H. Huang, L. Yao, Y. Hu, Q. Fan, and D. Huang. Ni-mwmote: An improving noise-immunity majority weighted minority oversampling technique for imbalanced classification problems. *Expert Systems with Applications*, 158:113504, 2020.
- [136] X. Xu, S. Fu, W. Li, F. Dai, H. Gao, and V. Chang. Multi-objective data placement for workflow management in cloud infrastructure using nsga-ii. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(5):605–615, 2020.
- [137] Y. Xu, Y. Zhang, J. Zhao, Z. Yang, and X. Pan. Knn-based maximum margin and minimum volume hyper-sphere machine for imbalanced data classification. *International Journal of Machine Learning and Cybernetics*, 10(2):357–368, 2019.
- [138] P. Yildirim. Chronic kidney disease prediction on imbalanced data by multilayer perceptron: Chronic kidney disease prediction. In 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), volume 2, pages 193–198. IEEE, 2017.
- [139] Y. Yusoff, M. S. Ngadiman, and A. M. Zain. Overview of nsga-ii for optimizing machining process parameters. *Procedia Engineering*, 15:3978–3983, 2011.
- [140] S. Zajac and S. Huber. Objectives and methods in multi-objective routing problems: a survey and classification scheme. *European Journal of Operational Research*, 290(1):1–25, 2021.
- [141] C. Zhang, K. C. Tan, H. Li, and G. S. Hong. A cost-sensitive deep belief network for imbalanced classification. *IEEE Transactions on Neural Networks and Learning Systems*, 30(1):109–122, 2019.
- [142] H. Zhang, H. Zhang, S. Pirbhulal, W. Wu, and V. H. C. D. Albuquerque. Active balancing mechanism for imbalanced medical data in deep learning–based classification models. *ACM Transactions on Multimedia Computing, Communications, and Applications* (*TOMM*), 16(1s):1–15, 2020.

- [143] L. Zhang, S. K. Shah, and I. A. Kakadiaris. Hierarchical multi-label classification using fully associative ensemble learning. *Pattern Recognition*, 70:89–103, 2017.
- [144] S. Zhang. Cost-sensitive knn classification. *Neurocomputing*, 391:234–242, 2019.
- [145] B. Zhao, Y. Xue, B. Xu, T. Ma, and J. Liu. Multi-objective classification based on nsga-ii. *International Journal of Computing Science and Mathematics*, 9(6):539–546, 2018.
- [146] J. Zhao, J. Jin, S. Chen, R. Zhang, B. Yu, and Q. Liu. A weighted hybrid ensemble method for classifying imbalanced data. *Knowledge-Based Systems*, 203:106087, 2020.
- [147] M. Zhu, J. Xia, X. Jin, M. Yan, G. Cai, J. Yan, and G. Ning. Class weights random forest algorithm for processing class imbalanced medical data. *IEEE Access*, 6:4641–4652, 2018.
- [148] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.
- [149] E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK-report*, 43, 1998.
- [150] N. A. Zolpakar, S. S. Lodhi, S. Pathak, and M. A. Sharma. Application of multi-objective genetic algorithm (moga) optimization in machining processes. In *Optimization of Manufacturing Processes*, pages 185–199. Springer, 2020.
- [151] W. Zong, G.-B. Huang, and Y. Chen. Weighted extreme learning machine for imbalance learning. *Neurocomputing*, 101:229–242, 2013.