Description: Iqra new logo

**COMPREHENSIVE RULE DISCOVERY USING**

**Differential Evaluation**

**FOR Breast Cancer Survival**

By

Zubair Farooq Piracha

(Registration No. 051-10-13067)

Supervised by

Dr. Hajira Jabeen

Assistant Professor, Iqra University, Islamabad Campus

Thesis Submitted as partial fulfillment of requirement for the Degree of

Master of Science in Computer Science (MSCS)

**Department of Computing and Technology**

**Iqra University Islamabad Campus**

February, 2014

ABSTRACT

Among all the cancer diseases breast cancer is more prevailing cancer throughout the world. The women of the world are most affected by this type of cancer. If the scenario of Pakistan is considered particularly, the same situation is observed throughout the country. Differential Evolution (DE), a new emerging Evolutionary Algorithm technique, is a simple, robust and direct search algorithm for global search optimization. Differential Evolution has the ability for searching the optimal solution and has been found efficient as compared to other evolutionary algorithms over many optimization tasks. A new attribute mapping technique is proposed for classification, which uses DE to discover if-then rules for real data for survival of breast cancer patients. The DE classifier improves the accuracy and provides coverage for both classes. The end result of DE classifier is to generate if-then rules which are comprehensible and offer better generalization abilities. The proposed classification algorithm has been tested on a dataset known as Haberman’s survival dataset downloaded from UCI Machine Learning Repository. This dataset has three attributes and two class labels. The proposed algorithm extracts the comprehensible rules for breast cancer survival data set. The proposed algorithm is compared to well-known classification algorithms and has been found efficient.

ACKNOWLEDGEMENTS

First and the foremost, extreme thanks to Almighty ALLAH, by whose grace I have been able to complete this thesis successfully.

I would also like to pay thanks to my supervisor Dr. Hajira Jabeen for her encouragement, advice and guidance throughout the thesis which enabled me to complete this thesis.

I owe success to my parents and family who are the real source of invaluable guidance that made me capable of all, what I am, today.

Last but not the least; I am grateful to my friends for encouraging and supporting me to embark on my research.

DECLARATION

I hereby declare that this research, neither as whole nor as part has been copied out from any source. It is further declare that I have prepared this report entirely on the basis of my personal efforts made under the sincere guidance of teachers especially my supervisor Dr. Hajira Jabeen. If any part of this thesis is proved to be copied out from any source or found to be reproduction of some other, I will stand by the consequences. No portion of the work presented has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Zubair Farooq Piracha

(MS (CS), 13057)

DEDICATION

I dedicate this thesis to my parents and my family who always inspired and encouraged me.

THESIS APPROVAL SHEET

It is certify that Zubair Farooq Piracha of MS (CS) Department of Computer & Technology, Student ID (13057) of IQRA University Islamabad, has submitted the final Thesis report on “**Comprehensive Rule Discovery Using Deferential Evaluation for Breast Cancer Survival**”. We have read the report and it fulfills the partial of Master of Science in Computer Science degree.

**INTERNAL EXAMINER:**

**Name:**

Designation:

Organization:

**Signature:** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EXTERNAL EXAMINER:**

**Name:**

Designation:

Organization:

**Signature:** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SUPERVISOR:**

**Name:** Dr. Hajira Jabeen

Designation: Assistant Professor

Organization: Iqra University, Islamabad Campus

**Signature:** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Table of Contents**

[Chapter 1 1](#_Toc428632254)

[Introduction 1](#_Toc428632255)

[1.1 Introduction to Machine Learning 1](#_Toc428632256)

[1.1.1 Supervised Learning 1](#_Toc428632257)

[1.1.2 Unsupervised Learning 2](#_Toc428632258)

[1.2 Introduction to Classification 2](#_Toc428632259)

[1.2.1 Classification Approaches 2](#_Toc428632260)

[1.2.2 Evolutionary Algorithm and Classification 3](#_Toc428632261)

[1.3 Introduction to Breast Cancer Disease 3](#_Toc428632262)

[1.3.1 Type of Breast Cancer 4](#_Toc428632263)

[1.3.2 Stages of Breast Cancer 4](#_Toc428632264)

[1.3.3 Treatment Methods in Breast Cancer 4](#_Toc428632265)

[1.3.4 Breast Cancer Facts 5](#_Toc428632266)

[1.4 Requirement of a Good Classifier 6](#_Toc428632267)

[1.5 Objectives 7](#_Toc428632268)

[1.6 Contributions 7](#_Toc428632269)

[1.7 Motivation 8](#_Toc428632270)

[1.8 Overview of the Thesis 8](#_Toc428632271)

[Chapter 2 10](#_Toc428632272)

[Literature review 10](#_Toc428632273)

[2.1 Introduction 10](#_Toc428632274)

[2.2 Decision Tree Based Algorithms 10](#_Toc428632275)

[2.2.1 C4.5 Decision Tree 13](#_Toc428632276)

[2.3 Rule Learning Classification 14](#_Toc428632277)

[2.3.1 RIPPER Algorithm 15](#_Toc428632278)

[2.4 Statistical Classification: 17](#_Toc428632279)

[2.4.1 Naïve Bayes Classifier: 18](#_Toc428632280)

[2.5 Neural Network 19](#_Toc428632281)

[2.5.1 Multilayer Perceptron 21](#_Toc428632282)

[2.6 Some Other Approaches in Classification 22](#_Toc428632283)

[Chapter 3 24](#_Toc428632284)

[Differential Evolution and Evolutionary Algorithms for Classification 24](#_Toc428632285)

[3.1 Introduction to Evolutionary Algorithms 24](#_Toc428632286)

[3.2 Differential Evolution 25](#_Toc428632287)

[3.3 Evolutionary Algorithm for Classification 27](#_Toc428632288)

[3.3.1 GA Based Approaches for classification 28](#_Toc428632289)

[3.3.1.1 GAssist as Classifier 29](#_Toc428632290)

[3.3.2 Other GA Approaches with Solution Representation and Fitness Functions 30](#_Toc428632291)

[3.3.3 PSO Based Approaches for Classification: 33](#_Toc428632292)

[3.3.3.1 Data mining using PSO 34](#_Toc428632293)

[CHAPTER 4 36](#_Toc428632294)

[Proposed Algorithm 36](#_Toc428632295)

[4.1 Haberman’s Survival Dataset: 36](#_Toc428632296)

[4.2 Rule Based Approach for Differential Evolution 36](#_Toc428632297)

[4.2.1 Differential Evolution as Classifier 37](#_Toc428632298)

[4.2.2 Solution Representation 37](#_Toc428632299)

[4.2.2.1 Attributes as Feature Vector 38](#_Toc428632300)

[4.2.2.2 If-Then Rule Representation 39](#_Toc428632301)

[4.2.2.3 Initialization and Ranges of Vector 39](#_Toc428632302)

[4.2.2.4 Mutation Algorithm 40](#_Toc428632303)

[4.2.2.5 Recombination 41](#_Toc428632304)

[4.2.3 Proposed Fitness Function 41](#_Toc428632305)

[4.3.4 Pseudo-code and Flow Chat 44](#_Toc428632306)

[Chapter 5 46](#_Toc428632307)

[Detail analysis of Habermen’s Survival Dataset 46](#_Toc428632308)

[5.1 Dataset Preparation 46](#_Toc428632309)

[5.2 Experimental Setup and Parameter Setting 47](#_Toc428632310)

[5.3 K-Fold Cross Validations 48](#_Toc428632311)

[5.4 Area under the ROC Curve (AUC) 48](#_Toc428632312)

[5.5 Weka an Keel Machine learning tools 49](#_Toc428632313)

[5.6 Analysis on Haberman’s Survival Dataset 49](#_Toc428632314)

[5.6.1 Training DE with Target Class Patient will Survive 50](#_Toc428632315)

[5.6.1.1 Accuracy Fitness Function 50](#_Toc428632316)

[5.6.1.1.1 Rule Representation with Detail Result using AND Operator 50](#_Toc428632317)

[5.6.1.1.2 Rule Representation with Detail Results using OR Operator 53](#_Toc428632318)

[5.6.1.2 AUC Fitness Function 55](#_Toc428632319)

[5.6.1.2.1 Rule Representation with Detail Results using AND Operator 55](#_Toc428632320)

[5.6.1.2.2 Rule Representation with Detail Results using OR Operator 57](#_Toc428632321)

[5.6.1.3 Comparison of Fitness Functions and Operators for Patient will survive as default class 59](#_Toc428632322)

[5.6.1.4 Comparison of DE with Traditional Algorithms 59](#_Toc428632323)

[5.6.1.5 Comparison of DE with Evolutionary Algorithms 60](#_Toc428632324)

[5.6.2 Training DE with Target Class Patient will not survived 60](#_Toc428632325)

[5.6.2.1 Accuracy Fitness Function 61](#_Toc428632326)

[5.6.2.1.1 Rule Representation with Detail Results using AND Operator 61](#_Toc428632327)

[5.6.2.1.2 Rule Representation with Detail Results using OR Operator 63](#_Toc428632328)

[5.6.2.2 AUC Fitness Function 65](#_Toc428632329)

[5.6.2.2.1 Rule Representation with Detail Results using AND Operator 65](#_Toc428632330)

[5.6.2.2.2 Rule Representation with detail Results using OR Operator 67](#_Toc428632331)

[5.6.2.3 Comparison of Fitness Functions and Operators 69](#_Toc428632332)

[5.6.2.4 Comparison of DE with Traditional Algorithms 69](#_Toc428632333)

[5.6.2.5 Comparison of DE with Evolutionary Algorithms 70](#_Toc428632334)

[5.6.3 Comparison of the DE-Classifier based on Class Selection 70](#_Toc428632335)

[5.6.4 Feature Selection 71](#_Toc428632336)

[5.6.5 Comparative Analysis of DE-Classifier on Other Biomedical Datasets 71](#_Toc428632337)

[5.6.5.1 Comparative Analysis of DE-Classifier on Heart Dataset 72](#_Toc428632338)

[5.6.5.2 Comparative Analysis of DE-Classifier of Pima Indians Diabetes Dataset 73](#_Toc428632339)

[5.6.5.3 Comparative Analysis of DE-Classifier of Mammographic Mass Dataset 74](#_Toc428632340)

[5.6.5.4 Comparative Analysis of DE-Classifier of Blood Transfusion Dataset 75](#_Toc428632341)

[5.6.6 Summary of the Experiment 76](#_Toc428632342)

[CHAPTER 6 78](#_Toc428632343)

[CONCLUSION AND FUTURE WORK 78](#_Toc428632344)

[6.1 Conclusion 78](#_Toc428632345)

[6.2 Future Work 79](#_Toc428632346)

[Works Cited 80](#_Toc428632347)

**List of Figures**

[Figure 1.1 Tumor size in Breast Cancer 4](#_Toc393199886)

[Figure 1.2: Breast Cancer in Pakistan 5](#_Toc393199887)

[Figure 1.3: Breast Cancer in Worldwide 6](#_Toc393199888)

[Figure 2.1: A Simple Decision Tree 11](#_Toc393199889)

[Figure 2.2: A Simple Neural Network Model 21](#_Toc393199890)

[Figure 2.3: Architecture of Feed-forward Multilayer Perceptron 21](#_Toc393199891)

[Figure 3.1: Main Steps in DE 25](#_Toc393199892)

[Figure 3.2: Simple Genetic Algorithms Pseudo Code 28](#_Toc393199893)

[Figure 3.3: Michigan Vs Pittsburgh Encoding Approaches 29](#_Toc393199894)

[Figure 3.4: GAssist Cycle 29](#_Toc393199895)

[Figure 3.5: Individual Representation in Genetic Algorithm for Discovering Rules 31](#_Toc393199896)

[Figure 3.6: Individual Representation in Discovering Comprehensible Rules with a GA 32](#_Toc393199897)

[Figure 3.7: PSO Algorithm 33](#_Toc393199898)

[Figure 3.8: Structure of a Particle in if-else Rule Discovery Using PSO 34](#_Toc393199899)

[Figure 4.1: Mapping of Attributes on Feature Vector 38](#_4.2.2.1__Attributes)

[Figure 4.2: Flow chart for Proposed Algorithm 45](#_Toc393199900)

[Figure 5.1: Hierarchy of the Experiment 50](#_Toc393199901)

[Figure 5.2: Comparison of Accuracy between Proposed Techniques, Traditional Algorithms and Evolutionary Algorithms 77](#_Toc393199902)

[Figure 5.3: Comparison of AUC between Proposed Techniques, Traditional Algorithms and Evolutionary Algorithms 77](#_Toc393199903)

**List of Tables**

[Table 3.1: Different Mutation Schemes in DE 26](#_Toc357977661)

[Table 4.1: Examples of attributes as Vector 38](#_Toc357977662)

[Table 4.2: Confusion Matrix for Two Classes 42](#_Toc357977663)

[Table 5.1: Details of Dataset 46](#_Toc357977664)

[Table 5.2: Class Distribution of Dataset 46](#_Toc357977665)

[Table 5.3: Ranges of Dataset 47](#_Toc357977666)

[Table 5.4: Particulars of Machines and software’s 48](#_Toc357977667)

[Table 5.5: 10 Best Rules using DE-Accuracy with AND Operator for “Patient will Survive” as True Class 51](#_Toc357977668)

[Table 5.6: DE-Accuracy Detail Results with AND Operator and “Patient will Survive” as True Class 52](#_Toc357977669)

[Table 5.7: Confusion Matrix for DE-Accuracy with AND Operator and “Patient will Survive” as True Class 52](#_Toc357977670)

[Table 5.8: 10 best Rules using DE-Accuracy with OR Operator for “Patient will Survive” as True Class 53](#_Toc357977671)

[Table 5.9: DE-Accuracy Detail Results with OR Operator and “Patient will Survive” as True Class 54](#_Toc357977672)

[Table 5.10: Confusion Matrix using DE-Accuracy with OR Operator for “Patient will Survive” as True Class 54](#_Toc357977673)

[Table 5.11: 10 best Rules using DE-AUC with AND Operator for “Patient will survive” as True Class 55](#_Toc357977674)

[Table 5.12: DE-AUC Detail Results with AND Operator and “Patient will Survive” as True Class 56](#_Toc357977675)

[Table 5.13: Confusion Matrix for DE-AUC with AND Operator and “Patient will Survive” as True Class 56](#_Toc357977676)

[Table 5.14: 10 best Rules using DE-AUC with OR Operator for “Patient will Survive” as True Class 57](#_Toc357977677)

[Table 5.15: DE-AUC Detail Results with OR Operator and “Patient will Survive” as True Class 58](#_Toc357977678)

[Table 5.16: Confusion Matrix for DE-AUC with OR Operator and “Patient will Survive” as True Class 58](#_Toc357977679)

[Table 5.17: Comparison of Operator and Fitness Function for Patient will Survive 59](#_Toc357977680)

[Table 5.18: Comparison of Proposed Technique (Patient will Survive as True Class) with Traditional Algorithms 59](#_Toc357977681)

[Table 5.19: Comparison of Proposed Technique (Patient will Survive as True Class) with Evolutionary Algorithms 60](#_Toc357977682)

[Table 5.20: 10 best Rules using DE-Accuracy with AND Operator for “Patient will not Survive” as True Class 61](#_Toc357977683)

[Table 5.21: DE-Accuracy Detail Results with AND Operator and “Patient will not Survive” as True Class 62](#_Toc357977684)

[Table 5.22: Confusion Matrix for DE-Accuracy with AND Operator and “Patient will not Survive” as True Class 62](#_Toc357977685)

[Table 5.23: 10 best Rules using DE-Accuracy with OR Operator for “Patient will not Survive” as True Class 63](#_Toc357977686)

[Table 5.24: DE-Accuracy Detail Results with OR Operator and “Patient will not Survive” as True Class 64](#_Toc357977687)

[Table 5.25: Confusion Matrix for DE-Accuracy with OR Operator and “Patient will not Survive” as True Class 64](#_Toc357977688)

[Table 5.26: 10 best Rules using DE-AUC with AND Operator for “Patient will not Survive” as True Class 65](#_Toc357977689)

[Table 5.27: DE-AUC Detail Results with AND Operator and “Patient will not Survive” as True Class 66](#_Toc357977690)

[Table 5.28: Confusion Matrix for DE-AUC with AND Operator and “Patient will not Survive” as True Class 66](#_Toc357977691)

[Table 5.29: 10 best Rules using DE-AUC with OR Operator for “Patient will not Survive” as True Class 67](#_Toc357977692)

[Table 5.30: DE-AUC Detail Results with OR Operator and “Patient will not Survive” as True Class 68](#_Toc357977693)

[Table 5.31: Confusion Matrix for DE-AUC with OR Operator and “Patient will not Survive” as True Class 68](#_Toc357977694)

[Table 5.32: Comparison of Operators and Fitness Functions for Patient will not Survive 69](#_Toc357977695)

[Table 5.33: Comparison of Proposed Technique (Patient will not Survive as True Class) with Traditional Algorithms 69](#_Toc357977696)

[Table 5.34: Comparison of Proposed Technique (Patient will not Survive as True Class) with Evolutionary Algorithms 70](#_Toc357977697)

[Table 5.35: Comparison of DE-Accuracy based on Class Selection 70](#_Toc357977698)

[Table 5.36: Comparison of DE-AUC based on Class Selection 71](#_Toc357977699)

Table 5.37: Comparison of DE-Classifier with Traditional Classification and Evolutionary Algorithms on Heart Dataset 72

Table 5.38: omparison of DE-Classifier with Traditional Classification and Evolutionary Algorithms on Pima Indians Diabetes Dataset …………………………………………………………………….73

Table 5.39: Comparison of DE-Classifier with Traditional Classification and Evolutionary Algorithms on Mammographic Mass Dataset.........................................................................74

Table 5.40: Comparison of DE-Classifier with Traditional Classification and Evolutionary Algorithms on Blood Transfusion Dataset ……….………………………………………………………………….75

**List of Algorithms**

[Algorithm 2.1: Pseudo-code for Decision Tree 12](#_Toc357977716)

[Algorithm 2.2: Pseudo code for C4.5 Algorithm 14](#_Toc357977717)

[Algorithm 2.3: Pseudo-code for Separate and Conquer Algorithm 15](#_Toc357977718)

[Algorithm 2.4: Pseudo-code of RIPPER for Binary Class 17](#_Toc357977719)

[Algorithm 2.5: Pseudo code for K-Nearest Neighbour Algorithm 23](#_Toc357977720)

[Algorithm 3.1: Pseudo code of simple EA 24](#_Toc357977721)

[Algorithm 4.1: Range of Vectors 39](#_Toc357977722)

[Algorithm 4.2: Initialization of Vector 40](#_Toc357977723)

[Algorithm 4.3: Mutation algorithm of Differential Evolution Classifier 40](#_Toc357977724)

[Algorithm 4.4: Recombination Algorithm for differential evolution 41](#_Toc357977725)

[Algorithm 4.5: Pseudo code for Proposed Algorithm 44](#_Toc357977726)

# Chapter 1

# Introduction

## 1.1 Introduction to Machine Learning

Machine Learning is an automatic computing procedure that learns from a series of examples and extracts information like rules, patterns etc. Machine learning systems try to reduce or eliminate the human interaction in data analysis, although the human interaction cannot be entirely eliminated because a system developer must indicate how the data is to be represented and which method is used to find the characteristics of data. Machine learning comes under the umbrella of artificial intelligence.

Machine learning is used in many applications particularly science and engineering. Classification, prediction, bioinformatics, image recognition, game playing, robotics, text categorization and global optimization problems are examples of application used in machine learning.

Machine learning algorithms mainly divided into two categories i.e. supervised learning and unsupervised learning [1] [2].

### 1.1.1 Supervised Learning

Supervised learning also known as classification is one of the tasks most often carried out by Intelligent Systems. In supervised learning, during the learning process, training data set provides both the input data and the desired target class (output). Machines use input data based on some target for learning. Supervised learning algorithms are usually fast, efficient and accurate. Examples of supervised learning models are Neural Networks, Multi-Layer Perceptron, Naïve bayes and Decision Trees, etc.

### 1.1.2 Unsupervised Learning

Unsupervised learning also known as clustering is a learning technique in which training data set does not have the target class. As unsupervised learning are without task oriented so the unsupervised models use some properties/characteristics of dataset during learning process. A lot of work been done for this technique; some examples of unsupervised learning are K-means, C-mean and Self Organizing Maps etc.

## 1.2 Introduction to Classification

Data classification is very common task in machine learning. Classification means that to categorize or group the things together in some manners, normally called characteristics.

In machine learning, classification is usually a particular type of learning in which we use the characteristic of data or examples of one or more classes having some class label to map the unknown examples. Classification helps people in the investigation and acquiring knowledge of their particular domains of interest. Classification is an interesting field due to its usage in many real life domains like fraud detection, diagnosis of disease, Network intrusion detection and decision making.

### 1.2.1 Classification Approaches

Classification task solves simplest as well as complex problems. In machine learning there is a lot of approaches used to perform classification. Each learning method is needed to train by using some examples called training dataset set to predict the class label for test data set with unknown labels. The common approaches for classification are defined below:-

* **Decision Tree based approach:** use the divide-and-conquer strategy e.g. C4.5
* **Rule based Approach:** use disjunctive normal form (DNF) e.g. RIPPER
* **Statistical Approach:** use statistical properties of Data e.g. Naïve Bayes Classifier
* **Artificial Neural Network:** inspired by biological Neural Network e.g. Multi Layer Perceptron
* **Evolutionary Algorithm Approach:** Natural behaviour of evolutionary process e.g. GIL

### 1.2.2 Evolutionary Algorithm and Classification

Evolutionary Algorithms (EAs) are stochastic optimization technique which inspired by the natural behaviour of evolutionary process inspired by ideas of Darwin and Lamarck [3]. The well-known EA algorithms are Simulated Annealing, Ant Colony Optimization, Genetic Algorithms, Genetic Programming, Particle Swam Optimization and Differential Evolution. Unlike the combinational algorithms EAs have been found efficient in performing data classification due to its robustness and global search optimization without searching entire search space.

There is lot of classification algorithms proposed with EAs. The most used techniques in EAs for classification are Genetic Algorithm and Genetic Programming but other techniques like Particle Swam Optimization and Ant Colony Optimization are also found in literature for classification tasks. Some of EAs are hybrid algorithms (combining two EAs techniques) to solve the classification problems like PSO/ACO [4].

The well-known EAs classification algorithms are GAssist, GIL, UCS, XCS, cAnt-Miner and SLAVE etc [5].

## 1.3 Introduction to Breast Cancer Disease

Breast cancer is type of cancer that starts in the breast cells. These cells grow in surrounding tissues and spread to distant areas of the body [6]. The source of traveling of breast cancer cells are blood/ lymph vessels. These cancer cells are attached with other tissues and make them tumors that damage other tissues. Breast cancer is found in both men and women but less case are found in men because the breast duct in men are less developed as compare to women.

### 1.3.1 Type of Breast Cancer

There are two most common types of breast cancer i.e. ductal carcinoma and lobular carcinoma [7]. Some women have a mixture of both type cancers. The types of breast cancer are defined below:

[**Ductal Carcinoma**](http://www.cancer.gov/Common/PopUps/popDefinition.aspx?id=CDR0000045085&version=Patient&language=English): Ductal carcinoma is the most existing type of breast cancer. This cancer begins from breast duct. About 70% women with breast cancer have this type of cancer.

**L**[**obular Carcinoma**](http://www.cancer.gov/Common/PopUps/popDefinition.aspx?id=CDR0000426416&version=Patient&language=English)**:** Lobular carcinoma is the next most common type of breast cancer. This cancer begins in a milk producing lobule of the breast. About 10% of women with breast cancer have lobular carcinoma.

### 1.3.2 Stages of Breast Cancer

Breast cancer has deferent stages, which depends on the size of tumor, whether the cancer is invasive or non-invasive, and spread of cancer to other parts of body etc. Breast cancers stages used Roman numbers i.e. 0, I, II, III and IV and some of the followed with alphabets i.e. A, B, C. if a patient is in in Stage I then we can say it is early-stage breast cancer patient and if it in stage IV it we can say it is in advance stage. Stage also shows the size of tumor like in Stage IA and Stage IB the size of tumor is no more than 2 centimeters across whereas in Stage IIIA tumor is no more than 5 centimeters across and in stage IV the tumor is of any size.



Figure 1.1: Tumor size in Breast Cancer [8]

### 1.3.3 Treatment Methods in Breast Cancer

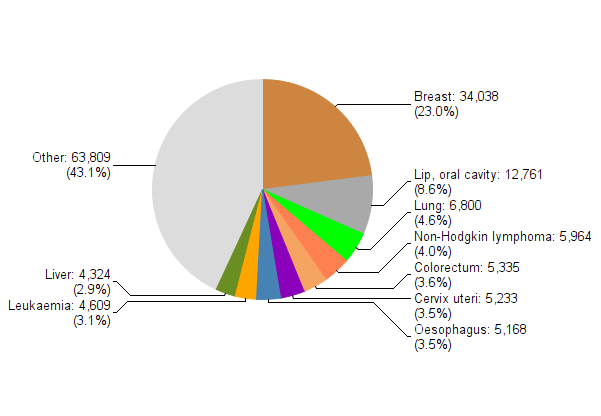
There are different treatment methods for breast cancer patents like surgery, Radiation therapy, hormone therapy, and Chemotherapy etc. The patient has option for selecting more than one treatment option. The treatments deepened on

* The general health of patient
* Which stage the breast cancer exist
* relation between the size of the tumor and the size of breast
* Whether tumor has hormone receptors etc.

### 1.3.4 Breast Cancer Facts

Breast cancer is most prevailing cancer in the world as well as in Pakistan. It is most leading cause of death in women in many countries [9].According to International Agency for Research on cancer (IARC)’s 2012 statistics, 25% of the total cancer case are found as breast cancer in female among them 14% are total death cases in breast cancer in female worldwide. In Pakistan 23% are estimated registered as breast cancer among them 16% are total death cases in breast cancer [10].

**Incidence**



**Mortality**

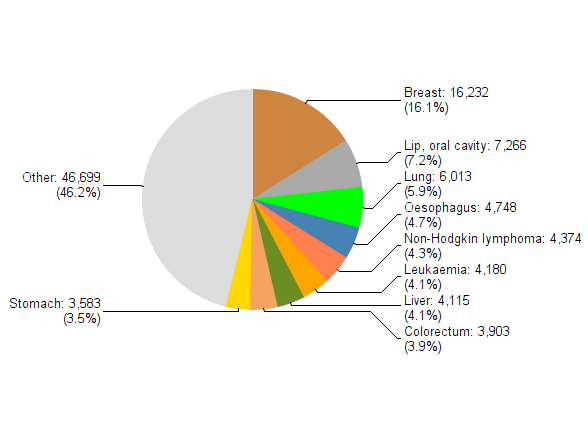
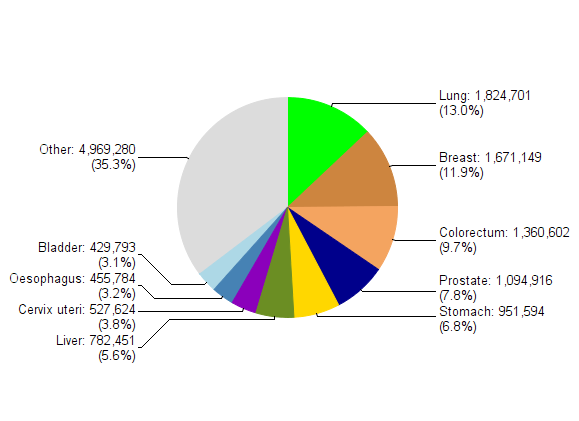


Figure 1.2: Breast Cancer in Pakistan (Both sexes) [10]

It is also mentioned that females of the most developed countries like America also suffer from breast cancer disease. According to Breast Cancer Facts & Figures 2013-2014 published by publication of the American Cancer Society, Atlanta, Georgia, indicates that, in 2013, around 39 thousand women are expected to die from breast cancer. [11].

**Incidence**



**Mortality**

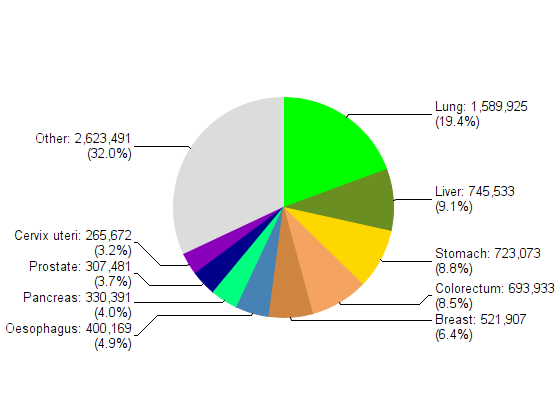


Figure 1.3: Breast Cancer in Worldwide (Both sexes) [10]

## 1.4 Requirement of a Good Classifier

Most often the requirement of a good classifier is accuracy and low error rate, which depends on the most covering examples i.e. if most example cover by the classifier then the classifier is more accurate, a good classifier also avoids the biasness of classes. Generalization is another characteristic of good classifier.

A classifier is easy to implement and applicable to any type of data (nominal, numeric, binary) and deals with unseen missing values so that less/no pre-processing is required. The result of the classifier is understandable so that it helps in future for decision making [12]. Robustness is another factor for measuring the performance of classifier.

## 1.5 Objectives

The objectives of this thesis are to develop a rule based classifier, which is easy to implement, more robust and enhance the accuracy of classifier by covering maximum examples. The other objectives which we cover in this thesis are given below.

* Try to avoid the biasness of class of unbalanced dataset by implementing different fitness function.
* Importance of selection of class.
* Provide the condition based solution according to the requirement.
* Features extraction from solution
* Compare with other EA Classifiers as well as other traditional classification techniques and show that DE-Classifier has the advantage of being more accurate and easy to implement.

## 1.6 Contributions

The contributions made in this thesis are:

* Development of DE-Classifier to improve accuracy and generate the if-then rules for decision makers that helps them in their practical life.
* Provide the solution for two class classification problem.
* Use different types of performance measuring function to improve the accuracy as well as provide maximum coverage for both classes.
* Performance evolution of dependency of Conditional Operator in classifier on numerical data types.
* Comparative evaluation of proposed approach with other classifiers on biomedical dataset.

## 1.7 Motivation

In supervised machine learning, there is large number of different classification algorithms and new algorithms are still appearing. All algorithms have their own pro and cons and have their importance for related groups and people. According to “no free lunch theorem” [13] there will never be a single best algorithm better than all others in terms of accuracy. That is way a new classification algorithm is proposed and has its own value [14].

Deferential Evolution (DE) is new Heuristic searching technique. Researchers have proved that DE is better than other EA algorithms, and as per current finding; no one proposed any algorithm for classification using DE to classify using if-then rules.

We use the Deferential Evolution for classification to generate the if-then-else rules. Although the traditional algorithms like C4.5 and CART are good classifier but the most traditional machine learning algorithms use local greedy search so we prefer DE as classifier due to its simplicity, robustness, and global search performance.

## 1.8 Overview of the Thesis

The thesis is divided into following parts:

* Chapter 2 provides the Literature review classical and combinational classification algorithms, including Artificial Neural Network and K-Nearest Neighbour.
* Chapter 3 focus on the Evolutionary Algorithms Techniques and their importance in classification with some examples of EA classification techniques discussed in detail. This chapter also discuss about DE.
* Chapter 4 Provide the detail for proposed algorithm with rule representation, proposed fitness function, pseudo-code and flow chart. This Chapter also tells about the importance of Haberman’s survival dataset.
* Chapter 5 represents the detail analysis and experiment results for proposed approach on Haberman’s survival dataset. This chapter also represents comparison of proposed classifier with other classification algorithms including EAs.
* Finally, conclusion of thesis and future work is discussed in chapter 6.

# 

# Chapter 2

# Literature review

## 2.1 Introduction

The most frequent use of intelligent systems is supervised classification. A lot of algorithms have been proposed and developed in supervised classification. These algorithms use different types of techniques. The most popular techniques are logical based (trees based techniques, rule base techniques), statistical, Artificial Neural Network, Genetic Algorithms and K-Nearest Neighbour etc.

All of the above techniques have their own advantages and disadvantages. Some of them are not good on different domains like tree based algorithms. Some of them can not represent the rule as human readable form like neural network. In this chapter we review some popular classical machine learning techniques with their advantages and disadvantages.

## 2.2 Decision Tree Based Algorithms

Decision trees are tree based classifier based on divide-and-conquer in which the data are sorted based on attribute values. Decision trees have nodes and branches like other trees. Nodes represent the attribute and branches represent the values. Root node (no incoming edge) is the starting node for decision tree and remaining nodes (known as terminal nodes or decision nodes) are called child.

Each node divides the attributes into two or -spaces more sub based on some function and each leaf represent to one class (Figure 2.1 shows the simple Decision tree). Construction of decision trees is an NP-Hard problem [15] so we need some heuristic method to construct a decision tree.

ID3, C4.5, CHAID, QUEST and CART are the well-known decision tree algorithms. C4.5 is the most popular algorithm to build up decision tree. C4.5 is the extension of ID3.

<30

FALSE

MALE

NO

NO

YES

FEMALE

YES

NO

TRUE

>50

<=50

Figure 2.1: A Simple Decision Tree

Best division of dataset based on selection of root node as well as decision nodes [16]. The division is based on two types of criteria univariate and multivariate. In univariate a single attribute is used to divide the dataset whereas in multivariate method several attributes are participated on division of dataset. Multivariate method is complicated and unpopular than univariate method. There are many proposed methods to select best nodes in decision tree e.g. information gain, gini index, distance measure, orthogonally criteria etc.

Overfitting is one of the problems while creating decision trees. Overfitting is defined as “A learned hypothesis h is overfit on training dataset if another hypothesis h’ exist and has large error then h while testing on entire dataset. To overcome overfitting on training dataset two common approaches are used i.e. either stop the training algorithm or prune the induced decision tree.

Prune has farther two types pre-prune and post-prone. Pre-prune do not allow the decision tree to grow its full size by using some termination criteria such as some threshold test. Post-prone is based on performance of decision trees by using some validation sets. There are many pruning methods proposed in literature like Minimum Description Length Pruning (MDL), Error-Based Pruning(used in C4.5), Minimum-Error Pruning, optimal pruning (OPT) etc.

* **Procedure decision\_tree (examples, attributes)**
  + **Best=choose\_best (attributes, examples)**
  + **tree= create\_tree (best);**
  + **for each value of vi in best**
    - **examples={elements of examples with best=vi}**
    - **remaining\_attributes=attributes-best;**
    - **subtree=decision\_tree(example, remaining\_attributes)**
  + **end for**
  + **add a branch of tree and subtree label vi**
  + **return tree**
* **end**

Algorithm 2.1: Pseudo-code for Decision Tree

Decision trees are self-explanatory and easily converted into set of rule. Decision trees are capable to handle nominal and numeric attributes, missing values and errors contains datasets. As decision tree are divide-and-conquer algorithm so they perform quickly but the sorting process consume most time especially for finding the threshold for numeric features. The performance of decision trees is based on interactions between attributes; if interaction is complex then performance is less. Sometimes the decision trees should contain duplicate subtrees to represent a classifier. Decision trees are over-sensitivity due to greedy characteristic to the training set for noise as well as the attributes that are not relevant.

### 2.2.1 C4.5 Decision Tree

C4.5 is well known algorithm for generating decision tree. It is an extension of ID3 and developed by [Ross Quinlan](http://en.wikipedia.org/wiki/Ross_Quinlan) [17]. C4.5 is used for supervised classification by using training dataset to calculate accuracy of rule.

C4.5 select one attribute at a time for splitting the dataset into subset based on information gain which is the difference of entropy. Entropy is defined as:

Let S be examples and f(C*i*,S) are the number of examples in S for class C*i* and |S| be the total number of examples then the equation becomes

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.1)

Information needed is defined as:

. . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . (2.2)

so information gained becomes

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . (2.3)

C4.5 is greedy algorithm and can easily be converted in to If-then rules. Although C4.5 is well established algorithm and good for small datasets but its efficiency becomes an issue when the dataset is very large [17]. Efficiency of C4.5 is decrease in continuous attributes [18]**.**Time complexity of C4.5 algorithm is O(n2.m) where n is number of attributes and m is training data [19].

Attribute selection is based on the highest value of information gain. To overcome the generalization C4.5 use the post-pruning in which some of the tree structure is removed by the accuracy criteria. Algorithm 2.2 defined the Pseudo code of C4.5 which is given below:

* **C45\_algorithm (Examples S, Attributes A)**
  + **for i=1 to A**
    - **gi =calcuate\_infogain(Ai)**
    - **if gi < Best\_gain**
      * **bestgain =gi**
      * **bestfeature=i**
    - **end if**
  + **end**
  + **create\_child(S,bestAttribute);**
  + **C45\_aglirhtm(S,A)//recusivly for each child**
* **end**

Algorithm 2.2: Pseudo code for C4.5 Algorithm

## 2.3 Rule Learning Classification

Rule learning algorithm is a type of supervised classification in which rule represent each class as propositional logic formulae known as disjunctive normal form (DNF). For k class mathematical representation of k-DNF is defined below:

...(2.4)

Where k is the number of disjunctions (ν) and n are number of conjunctions (Λ) of each disjunction.

Rule based algorithms use separate-and-conquer (also known as covering algorithms) strategy, in which a classifier searches rules from training dataset by covering positive example, if the examples are covered correctly then the rules are added into to rule set until the no positive examples left. After that the algorithm recursively discover rules that explain the remaining negative examples. Rule learning algorithms learns single rule at a time.

Algorithm 2.3: Pseudo-code for Separate and Conquer Algorithm

* **Separate\_and\_Conquer (Pos,Neg)**
  + **Rule\_Set= Null**
  + **While Pos ≠ 0**
    - **Rule=Genrate\_new\_Rule(Pos,Neg)**
    - **literal = null**
    - **While Neg ≠ 0 do**
      * **L=new\_literal (literal, Pos, Neg)**
      * **L = L U {literal}**
      * **N = x \ {x ε N and x violate literal}**
    - **End while**
    - **Add rule to rule\_set “if Λ literal ε L ‘literal’ then assign the positive class”**
    - **P = x \ {x ε P | x covered by Rule}**
  + **END while**
  + **Return VrεRr**
* **End**

In the above DNF separate and conquer algorithm Pos are set of Positive examples and Neg are set of Negative examples and assuming that Pos ∩ Neg = Null. To evaluate the new literals some heuristic method is used. New rule tells the incorrectly negative example of a rule.

AQ is considered as first separate-and-conquer algorithm proposed by Ryszard Michalski in 1960s in which top-down search method is used to find best rule [20]. Prism, Grow, Ripper, Foil, CN2 are well known rule learning algorithms. Most of separate-and-conquer algorithms work on DNF logic and use hill-climbing methods for greedy search. Some times DNF is not good for noisy data moreover hill-climbing strut in local optima.

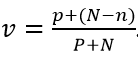
### 2.3.1 RIPPER Algorithm

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) is well known propositional rule learning algorithms proposed by William W. Cohen [21]. Ripper is a modified and optimized version of IREP proposed by Furnkranz and Widmer in 1994. Ripper performs well on large and noisy datasets. To understand the Ripper we first know about the RIP (Reduce Error Pruning) and IREP (Incremental Reduce Error Pruning).

In REP rule learning the training dataset is divided into two sets i.e. growing and pruning. RIP use some heuristic method to overfit the growing set and then repeatedly simplified the set by some pruning operator (e.g. deletion of single condition). This process is ends till operator will increase error on the pruning set.

IREP is modified version of REP [22]. Rule representation of IREP is conjunction of attributes and rule set is DNF formula. IREP generate single rule at a time in greedy way. When a single rule is found IREP delete all positive and negative examples which is covered and repeat this process till no further positive examples left over or an unacceptable error rate can be found by rule. RIPPER is produced by modified the following three steps in IREP:

1. Introduce new metric to guide pruning phase
2. New stopping condition when error rate is greater than 50%.
3. new technique to optimize the rule learning

**In Ripper growing phase is a proportional version of FOIL and in pruning rule is simplified by removing any final sequence of conditions by maximize the following function:

*.* . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . (2.5)

Where “P” represents total positive examples and “N” represents total Negative examples in pruning set. “p” is number of positive examples and “n” is number of negative examples covered by rule in pruning set This step is repeated until no change in value of “v” by deletion [21]. Ripper classifies multiple classes by using increasing order of occurrence of class for multiple classes RIPPER is known as RIPPERk. Algorithm 2.4 shows the pseudo-code of RIPPER for binary class.

Algorithm 2.4: Pseudo-code of RIPPER for Binary Class

* **procedure RIPPER (examples)**
  + **rule\_set = NULL**
  + **while Positive\_example ≠ NULL**
    - **Split dataset into Growing and pruning**
    - **rule = grow\_rule()**
    - **rule = prune\_rule()**
    - **If error\_rate> %50** 
      * **return rule\_set**
    - **else**
      * **rule\_set = {rule\_set ∪ rule}**
      * **Remove examples covered by rule**
    - **end if**
  + **end while**
  + **return rule\_set**
* **end**

## 2.4 Statistical Classification:

In statistical classification method, learning hypothesis works on the probability based model and/or statistical characteristics of data. Probability based model gives a probability for each class that an example belongs to.

A simple method in statistical classification is Linear Discriminant Analysis (LDA). LDA use the maximum likelihood method for finding linear combination between attributes and classify two or more than two classes. LDA assume that the attribute vector of a class are independent and follows some probability distribution with probability density function [23]. The unseen data is than assign a class for greater density function. LDA works on continuous attributes and for categorical attributes an equivalent technique to LDA called Discriminant Correspondence Analysis is used [16]. LDA works only in linear phenomena and can work on missing values [24].

The most popular and well known algorithm under statistical learning is Bayesian networks [16]. The two most popular Bayesian networks are Naïve Bayes Classifier (a simple Bayesian Network) and Bayesian Belief Network (graphical model of Bayesian network). The detail of Naïve Bayes classifier is in next section where Bayesian network is model in which the attributes are represented as relationship of probability in graphical model.

The main disadvantage of Bayesian Belief network is that when it use a dataset having numerous features. Numerous features need to make a large Bayesian network which is not practicable [16]. In most cases of Bayesian Belief network, numerical feature need to be discreteized.

### 2.4.1 Naïve Bayes Classifier:

A simple form of Bayesian network is Naïve Bayes in which classifier is represented as a direct acyclic graph in which one attribute represent as parent (unobserved node) and remaining are the children’s (observed nodes). This probability based model assuming independence of all attributes to the given class. This assumption is known as conditional independence although this assumption is not true in most of cases.

Naïve Bayes classifier works on the principle of Bayes’ theorem which is named after Thomas Bayes (18th century) [17]. Baye’s theorem is defined as follows.

Let X be a set or data example for any unknown class label, let we assume that C be that X belongs to; than baye’s theorem stats that

. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . (2.6)

Where P(C) is known as prior probability that shows the probability of occurrence of class C. p(X) is probability of occurring of a set X, p(X|C) is the X given that C also known as likelihood and p(C|X) probability of C given that X is observed also known as posterior probability

A simple naive Bayesian classifier work on the principle of naïve bayes rule, which work as follows:

1. Suppose we have a training set with n samples (A1,A2,…...An) and each X sample has d-dimension attributes , X=(X1,X2,….Xd)
2. Suppose that there are K classes C1, C2,….Ck .
3. The Classifier predict X (a data sample with unknown class) belongs to the class having the highest posterior probability. unknown sample X assign to a class Ci if and only if
4. . . . . . . . . . . . . . . . . . . . . . . . . .. (2.7)
5. Then we maximize the P (Ci | X) by using Bayesian theorem given below.
6. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.8)
7. P(X) is unchanged for all classes so P(X | Ci) P(Ci) is only required to maximize. For unknown prior probability we supposed that all classes are equally likely so there is only need to maximize P(X| Ci). Prior probability P(Ci) may be estimated by ratio of number of training examples of class Ci and the total number of training samples.
8. Computing P(X|Ci) is computationally expensive for many attributes so naïve bayes assume that the classes are conditionally independence and can be estimated from training examples so P(X| Ci) is defined as

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.9)

We already mention that Naïve bayes classifier assuming independency but practically the dependency exit between attributes and most of time it effect the accuracy.

## 2.5 Neural Network

Neural Network also called Artificial Neural Network is inspired by the biological Neural Network for solving the problems based on self-decision like human. It is an emerging technology popularized in 80’s after proposing multi-layered feed forward network. It can solve many problems like classification, clustering, function approximation, prediction, forecasting and optimization. Our focus on this mathematical model is supervised classification.

Artificial neural network play an important role for both supervised classification and unsupervised learning. The neural network as a tool has proven itself an alternative technique as a classifier due to its self-adaptive methods, nonlinear relations and complex communications between variables. Neural network can approximate any function with arbitrary accuracy, and nonlinear relations make them flexible moreover they have skilled to calculate the posterior probabilities. Neural network has many disadvantages. They can work only on real attribute in the range [0,1] for categorical and other real value attributes (except 0 and 1) a step pre-processing must be taken. Choosing a correct network is another problem because neural network has many parameters e.g. number of hidden layers, activation functions, number of units, learning rate etc. The major disadvantage of neural network in classification is that it seems like a black box, it gives us the weights instead of rules which are difficult to understandable for human being and have no meaning just usable for trained network.

In artificial neural network, larger number of simple process called neuron, are interconnected with each other. Each neuron has an input and desired output. There must be a relation between input and desired output.

Neural network is a connection of simple neuron. In a simple neuron model there is an input associated with weight and activation function and desired output. Mathematically it can be represented by the follows equations:

. . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . (2.10)

. . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.11)

Equation 2.10 represents the sum function and equation 2.11 represents the activation functions where Xi is input, Wi is weight associated with input and b represents the bias whereas y is output passes from the activation function ( Ф(y) ).

Figure2.2 shows a simple neural network model

**b**

**wi**

**y**

**∑**

**∫**

**Ф(y)**

**Xi**

Figure 2.2: A Simple Neural Network Model

### 2.5.1 Multilayer Perceptron

Perceptron is single layer model and can classify linearly separable dataset. In real word problem we do not know either dataset is linearly separable or not. Multilayer Perceptron (MLP) has an ability to classify linear as well as non-linear datasets.

MLP consists of many neurons connected with each other’s. In MLP neurons represents as layers. There are three types of layers in MLP i.e. input, output, and hidden layer. Feed-forward multilayer perceptron is most popular in MLP model. The figure 2.3 represents the architecture of Feed-forward MLP.

Output Vector

Output Layer

C1

1

n

n Hidden Layers

Input Layer

Input Vector

X1

Xn

Cn

Figure 2.3: Architecture of Feed-forward Multilayer Perceptron

In the above figure First layer is input layer, which take the examples as input. The next layers are hidden layers and last layer is output layer.

The feed-forward multilayer networks are the most widely used as neural network classifiers [25]. Feed-forward network are trained with some training algorithm like back propagation (bp), Linear Disarmament Analysis etc. back propagation is most popular training algorithm for the researchers [26].

Back-propagation works on the principle of minimizing the error. In back-propagation we fist initialize the weights on some random input example, propagate the signal forward and compute the error then back propagate the error to previous layer for adjusting the weights and this process repeats until error at the output layer below the threshold or maximum number of iteration reached.

## 2.6 Some Other Approaches in Classification

There are many other approaches for machine learning like evolutionary learning, Instance-based learning, case-based reasoning, [Learning Vector Quantization](http://en.wikipedia.org/wiki/Learning_Vector_Quantization) (LVQ) and Support Vector Machines (SVM) etc. In this section we discuss instance-based learning techniques especially k-Nearest Neighbour (kNN).

Instance base learning is a classification method which predicts a class based on some similarity between instance (examples) and also avoiding abstraction to specific instance. They store examples, delay the process till classify new examples. Instance based-learning are also known as lazy learning due to delay, so the main disadvantage of instance-based learning is large computational time.

K-Nearest Neighbour (kNN) is type of instance based-learning which classify the examples by identifying the nearest neighbour closed to instance belong to specific class. In kNN instance represent a point in distance space. There are different metrics to represent a distance like Minkowsky, Manbhattan, Chebychev, Euclidean, Camberra and Kendall’s Rank Correlation [16]. Here we defined Euclidean Distance.

The Euclidean distance D(p, q) between two points p and q is defined as:

. .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.12)

Although kNN is a simple algorithm but its classification speed is slow and it required large storage [27] and Chosen of correct k is another problem in kNN [16]. Algorithm 2.5 show the pseudo code for basic kNN which is defined below:

* **Procedure k\_nearst\_neighbour(examples X,Class C)**
  + **Y=(Y1,……Yn) //new example to be classified**
  + **For each (Xi,Ci)** 
    - **D(Xi,Y)=Claclutate\_distance(Xi,Y)**
  + **end**
  + **Order\_distance(D)//order distance from min to max**
  + **Select\_nearest\_K(Y,D)**
  + **Assign Y nearst class in K**
* **end**

Algorithm 2.5: Pseudo code for K-Nearest Neighbour Algorithm

# Chapter 3

# Differential Evolution and Evolutionary Algorithms for Classification

## 3.1 Introduction to Evolutionary Algorithms

Evolutionary Algorithms (EAs) are random search optimization algorithms inspired by idea of Darwin’s genetic evolution process. The working process of EA is simple. EA consist of population of individuals. In EA an individual works with other individual to create new generation if new generation is improved then we add this generation in existing population and this process ends on some conditions known as termination condition. The selection of new generation is based on some fitness function. Algorithm 3.1 shows the pseudo code of simple EA which is given below.

* **Procedure EA()**
  + **Initialize random population P with in search space**
  + **Evaluate (P)**
  + **While (termination condition is not satisfied)**
    - **P’= Selection(P); //select parents**
    - **P’’=Recombination(P’); P’’’=Mutation(P’’)**
    - **Evaluate( P’’’);**
    - **Best\_selection(P’’’)**
  + **End while**
* **end**

Algorithm 3.1: Pseudo code of simple EA

EAs contain many algorithms but most popular algorithms are Genetic Algorithms (GA), Genetic Programming (GP), and Differential Evolution (DE) etc. DE is the latest EA technique.

## 3.2 Differential Evolution

Differential Evolution (DE) is simple and fast population based random search algorithm proposed by Ston and Price in 1995 [28]. It is a real parameter optimization technique which comes under the Evolutionary algorithms.

DE algorithm involves four basic steps i.e. initialization, mutation, crossover and selection. Figure 3.1 show the main steps in DE. Working of DE algorithm is simple. Population of DE starts with an NP individuals and each individual represented as a D-dimension vector. Each NP individual is known as genotype of candidate solution.

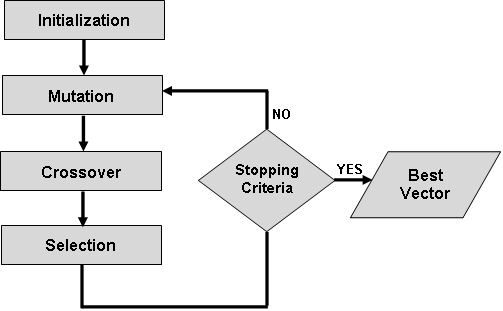


Figure 3.1: Main Steps in DE

DE generates new vector by combining of parent vector and other vectors with in population. A parent vector is replaced if and only if its fitness is better than child vector. There are three parameters in DE and the names of parameters are crossover rate (CR), population size (NP) and scaling factor (F).

Let the NP is total population of DE and each individual (vector) in NP has D dimension for generation G then individual is defined as

) ……. . . . . . . . . . . (3.1)

In literature different population size (NP) is defined by different researchers initially 10D populations is considered to be a good choice for finding the global optimum. Size of NP is very important to find the global optimum Different ranges of NP are suggested for reliability and balance the speed e.g. 5D to 10D, 3D to 8D and 20D to 40D [29].

After initialization of population, DE implements mutation operation by which a population vector becomes mutation vector which is defined below.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . (3.2)

Mutation vector is generated by different mutation scheme. There are many DE mutation schemes in literature these schemes use the convention “DE/a/b/c”. In this scheme “a” represents the construction of mutant vector; “b” represents that how many numbers involved in difference vector and “c” represents the crossover type [30]. The most commonly used scheme in DE is “DE/rand/1/bin” [29]. The different mutation schemes define in the table 3.1 [31].

|  |  |
| --- | --- |
| Mutation Scheme (DE/a/b/c) | Vector Generation Formula |
| DE/rand/1 |  |
| DE/best/1 |  |
| DE/current-to-best/2 |  |
| DE/best/2 |  |
| DE/rand/2 |  |

Table 3.1: Different Mutation Schemes in DE

In the above schemes r1, r2, r3, r4 and r5 are random selected vectors from current population at generation G. Xbest is global best vector and Xi is current best vector and F is scaling factor.

Performance of mutation depends on F. If F is larger than it increases the probability to escape from local optimum. Value of F must be greater than zero. Different researchers choose different values/ranges of F but typical value of F is between 0.5 and 1.0 [29].

After the mutation, crossover step becomes which control the convergence speed and tells how many population vectors are mutated. Crossover process use a parameter called crossover rate (CR). CR rate is a probability which must be in 0 ≤ CR ≤ 1 range. If the CR is larger than it means fast convergence. Initially the value of CR=0.1 is a good choice. To increase the convergence speed normally the value 0.9 or 1.0 is suggested for CR [29]. Crossover operation is defined below.

. . . . . . . . . . . . . . . .. . . . (3.3)

Where m=1,2,3 …….NP-1, NP and n=1,2,3…….D-1,D.

The above equation 3.3 is known as binomial cross over.

After crossover operations the selection operation becomes which is defined as:-

. . . . . .. . . . . . . . . . . . . . . . . . . . . . . (3.4)

## 3.3 Evolutionary Algorithm for Classification

Recently EAs prove itself as promising technique in data mining and classification. The main reason of using EAs is that it is robust, self-adaptive search technique and performs global search from collective learning [5] [32].

In machine learning and data mining most of EAs address the classification task and the popular among them is discovery of if-then rules [33]. In classification, Learning Classifier System (LCS), GAssist, Genetic based machine learning systems (GBML), cAnt-Miner etc. are well known classification algorithms which use EA technique.

### 3.3.1 GA Based Approaches for classification

Genetic Algorithm is a tpe of evolutionary algorithms inspired by biological evolution of Genes, initially invented by John Holland in 1960s [34]. GA provides efficient technique for optimization which is used in many applications including machine learning.

There are three types of operator involves in simple GA i.e. selection, crossover and mutation. Selection method chooses the chromosomes from the population for reproduction. Crossover method produces a child solution by combining parents’ solution. To escape from local optima mutation operator is used, which flips (0 as 1 and 1 as 0) some of the bits in a chromosome. Pseudo-code for simple genetic algorithm is given below (figure 3.2).

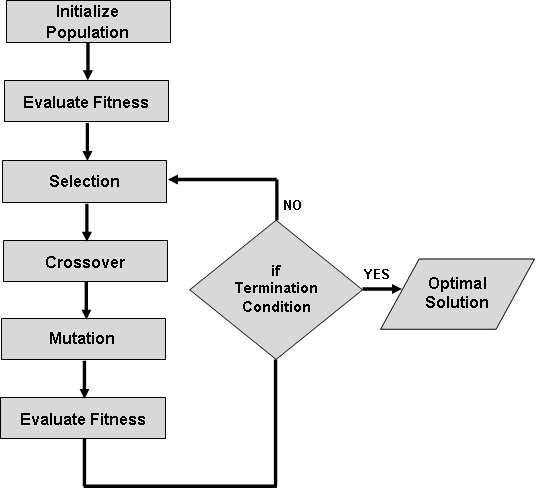


Figure 3.2: Simple Genetic Algorithms Pseudo Code

Genetic Algorithm as classifier is recognized as Genetic Based Machine Learning (GBML) and/or Learning Classifier System (LCS). There are two well-known Genetic Algorithm approaches based on how rules represent for classification i.e. Pittsburgh approach (Pittsburg LCS) and Michigan approach (Michigan LCS).

Michigan LCS approach represents individual encodes as a single rule, whereas Pittsburgh LCS represents set of rules as individual encodes [33] [35]**.** Figure 3.3 shows the difference between Michigan LCS and Pittsburgh LCS.

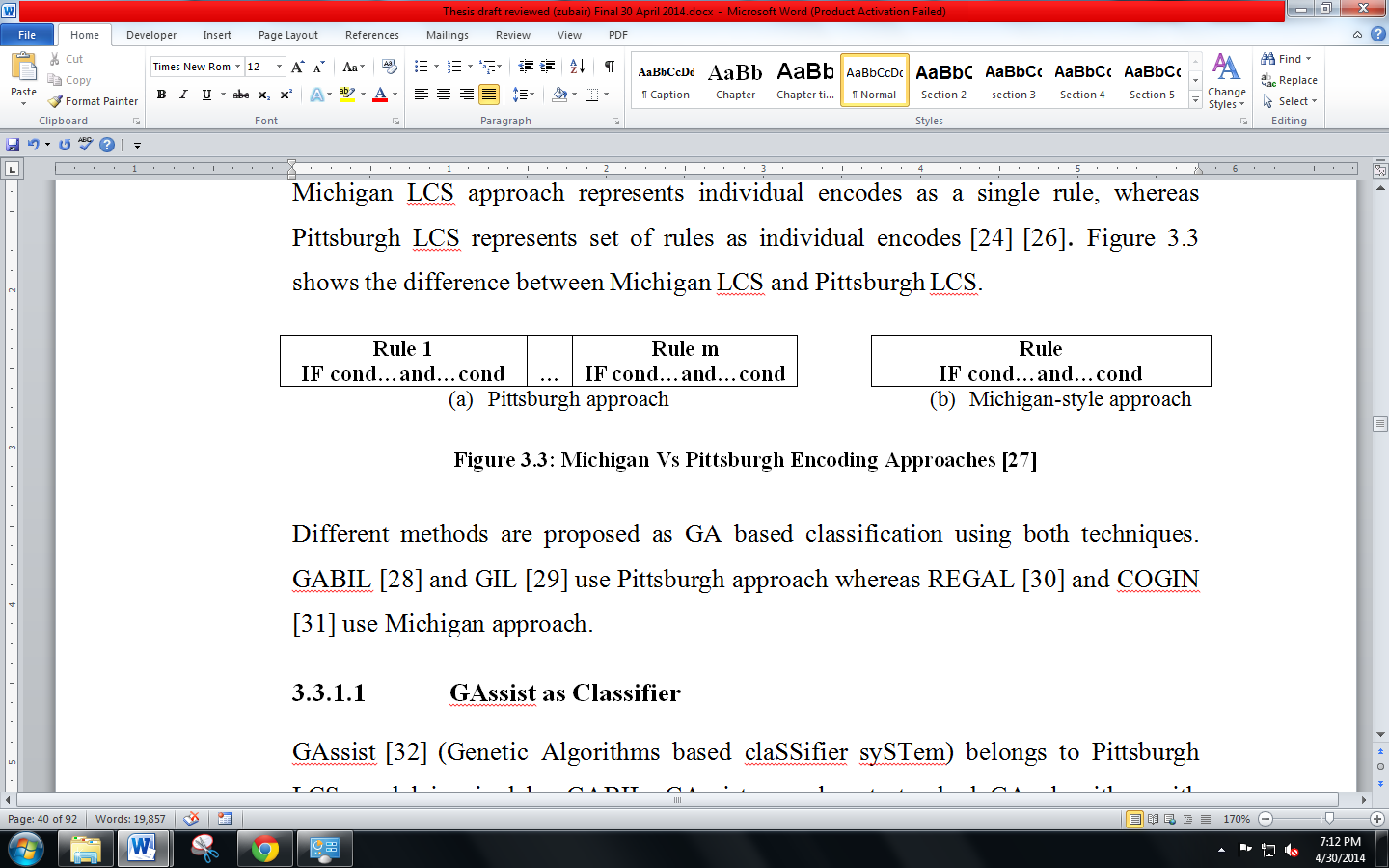


Figure 3.3: Michigan Vs Pittsburgh Encoding Approaches [36]

Different methods are proposed as GA based classification using both techniques. GABIL [37] and GIL [38] use Pittsburgh approach whereas REGAL [39] and COGIN [40] use Michigan approach.

#### 3.3.1.1 GAssist as Classifier

GAssist [41] (Genetic Algorithms based claSSifier sySTem) belongs to Pittsburgh LCS model inspired by GABIL. GAssist use almost standard GA algorithm with selection crossover and mutation operation, in which individuals represent complete problem solution. Figure3.4 show the GA cycle used in GAssist.



Figure 3.4: GAssist Cycle [41]

GAssist use ordered rule set, which are assembled as a decision list. Rules are represented as ADI (Adaptive Discretization Intervals), in which the continuous space is discretized into fixed intervals. The semantic of ADI rule consists of two parts i.e. conditional part and classification part, and represented as Conjunctive Normal Form (CNF). CNF represents the conjunction () of clauses where the clause is a disjunction () of literal. Mathematical representation of CNF is defined below.

. . . . . . . . . . . . . (3.5)

The crossover in GAssist takes place between the rules by selecting cut points across attribute boundaries. The default value of crossover operator is 0.6. Mutation is performed by bit-flipping same as GABIL. GAssist uses Tournament selection as Mutation strategy and the value of the tournament is 3. GAssist also has missing value policy in which missing values of attributes are replaced with substitution of same class (most frequent value or average value based on type of attribute). Fitness function of GAssist is squared accuracy fitness function which is defined below.

. . . . . . . . . . . . . . . . . . . (3.6)

### 3.3.2 Other GA Approaches with Solution Representation and Fitness Functions

There are lots of Data mining solutions proposed using Genetic algorithm, with differences like solution representations (e.g. Michigan LCS, Pittsburgh LCS), different mechanism for selection, mutation and crossover operator and fitness functions. Solution representation and fitness function of some GA based algorithm is defined below:

1. **GIL**

GIL [38] is Pittsburgh Learning Style Algorithm which represents rules as disjunction form and each rule represent in the form of VL1 logic. GIL use product of accuracy (correctness) and cost of individual for measuring fitness. The equation of fitness function is given below:-

. . . . . . . . . . . . . . . . . . . . . . …….. .. (3.7)

Correctness and cost is defined below

. . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . (3.8)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .(3.9)

and are positive and negative example covered by rule. and is total number of positive and negative examples. w1, w2 and w3 are weights with value of 0.5 for both w1 and w2 and w3 is smaller like 0.01-0.02.

1. **An Incremental Approach to Genetic Algorithms Based Classification (ILGA)**

ILGA [42]is based on incremental approach in which learning process is ever changing. ILGA used the standard GA approach for rule based classification. The encoding mechanism for discovering classification rules is given below.

*then y=C*. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .(3.10)

Fitness function of ILGA is simple and it is defined as the percentage of instances that can be correctly classified.

1. **A Genetic Algorithm for Discovering Classification Rules in Data Mining**

Hamid Shahbazkia Basheer and M. Al-Maqaleh [43] Proposed Michigan LCS system, in which individual is represented below.

|  |  |  |
| --- | --- | --- |
|  | …… |  |

Figure 3.5: Individual Representation in Genetic Algorithm for Discovering Rules [43]

In the above rule Ai represents i-th attribute, Valij represents the j-th value for attribute Ai and Opi is the comparison operator. For categorical attributes the “=” operator is used and for continuous value “<=” or “>” operators are used. Fitness function uses confidence (conf), coverage (cove) and comprehensibility.

Coverage is defined as the ratio of total examples that are correctly classified as decision class and total number of examples in decision class. Comprehensibility is defined as the simplicity of the rule which means the number of conditions effect the fitness i.e. less conditions reduce the fitness. The equation of fitness function is defined below.

. . . . . . . . . . . . . (3.11)

Where W1­­,W2 and W3 are user defined weights between 0 and 1.

1. **Discovering Comprehensible Classification Rules with a Genetic Algorithm** [44]

The proposed algorithm represents the if-then rules in which individual chromosome involves weight (Wi), Operator (Oi) and Variable (Vi). Wi is real valued variable between 0 and 1, which indicates the presence of the attribute in the rule. Oi are conditional operators and Vi is value belongs to attribute Ai.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Gene1 | | | … | … | Genen | | |
|  |  |  | … | … |  |  |  |

Figure 3.6: Individual Representation in Discovering Comprehensible Rules with a GA

Fitness function of above algorithm is product of Sensitivity and Specificity which is given below:

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . .(3.12)

Whereas the Se and Sp is defined as:

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (3.13)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (3.14)

tp is true positive, tn is true negative, fn is false negative and fp is false negative.

### 3.3.3 PSO Based Approaches for Classification:

In 1995, Eberhart and Kennedy proposed an algorithm known as Particle swarm optimization (PSO) [45]. PSO is self-adaptive random search algorithm which is inspired by the social behaviors of animals such as fish schooling and birds flocking etc.

In PSO system, multiple candidate solution called a ‘particle’ coexists and collaborates simultaneously. Each particle flies within search space for finding optimal position. A particle changes its position form own experience and experience of its neighbors particle. Tracking and remembering the best position encounter build particle’s experience.

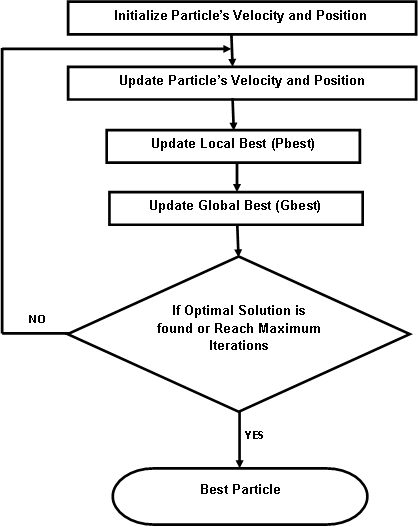
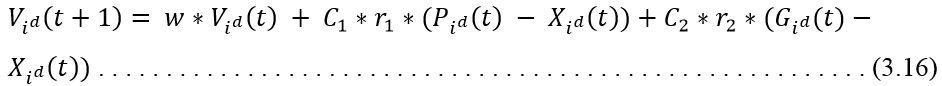


Figure 3.7: PSO Algorithm

PSO performance is based on Cognitive and social learning factors. To improve performance of population diversity many selection strategies have been proposed.

PSO is updating their generation using following formula.

. .. . . . . . . . . . . . . . . .. . . . (3.15)



Where and represents current position and velocity at time t for D dimension, represent current best, represent global best position. w is inertia weight whereas C1 cognitive factor and C2 is social factor. r1 and r2 are two random numbers between 0 and 1.

#### 3.3.3.1 Data mining using PSO

There are some work in rule discovery and classification tasks has been done using Particle swam Optimization (PSO). “Rule Discovery with Particle Swarm Optimization” [46] is proposed real encoded way for if-else rule discovery algorithm using PSO. The particle consist of real-valued attributes are in the range [0,1] and combination of Presence of attributes, attributes itself and operator. The structure of a particle is given below (Figure 3.8).

|  |  |  |
| --- | --- | --- |
| **Attribute-existence-array** | **Operator-array** | **Attribute-array** |

Figure 3.8: Structure of a Particle in if-else Rule Discovery Using PSO [46]

If the value of attribute-existence array is zero then the attribute is absence in the rule antecedent else attribute is present. If the data is continuous and operator-array is 0 then the operator is less than (<) else operator is greater-than-equal to (>=), and if the data is categorical and operator-array is 0 then the operator is not equal to (! =) else operator is equal to (=).

PSO also combined with Ant Colony Optimization to perform the classification task. PSO/ACO [4]and PSO/ACO2 [47] are the examples of two combined optimization approaches for classification. PSO/ACO performs tree-like hierarchical classification for predicted classes. PSO/ACO uses sequential covering approach for discovering if-else rules (one rule at a time). PSO/ACO2 is a modification of PSO/ACO.

# CHAPTER 4

# Proposed Algorithm

## 4.1 Haberman’s Survival Dataset:

Biomedical datasets are challenge tasks in machine learning due to its accuracy and comprehensible for decision makers [5]**.** Haberman’s survival dataset is one of them, which contain breast cancer data. Breast cancer is most prevailing cancer disease in women. It starts in the breast tissue.

Habermans’s Survival dataset is multivariate dataset which contains the records of patients with breast cancer who had undergone surgery. The dataset contains 3 attributes with class attribute detailed as follows [48]:

1. Age of patient at time of operation
2. Patient's year of operation
3. Number of positive axillary nodes detected
4. Survival status (class attribute)

* 1 = the patient survived 5 years or longer
* 2 = the patient died within 5 year

## 4.2 Rule Based Approach for Differential Evolution

In this thesis we proposed classification approach based on if-then rules. Differential Evolution is used in this research work to find out the rules and improves the accuracy and coverage for haberman’s survival data set. We chose the rule representation classification, because rules represent the unit of knowledge. Rules are understandable for human being and use in decision making process due to its natural format expression; Moreover if-then rules are popular in machine learning classification community.

### 4.2.1 Differential Evolution as Classifier

As mentioned in the chapter 3, Evolutionary algorithms have their importance in machine learning especially in classification due to its robustness, self-adaptive and global searching technique. Differential Evolution is one the most powerful and simple EA algorithm. Compare to most other EAs, DE is a simple and easily implementable. It has lot of variants as compare to other evolutionary algorithms. DE has few parameters i.e. Cr, F, NP [49].

DE is most robust than other EA algorithms. Although PSO is very simple, popular and robust algorithm; but performance of DE is better than the PSO as well as simulated annealing over many optimization functions [49] [50]. Jakob Vesterstrøm also proved that DE is efficient and robust then PSO and GA [51].

As per current findings; there is no classification algorithm which uses DE as a classification algorithm to generate if-then rules. By seeing the above points an experiment is needed to improve the classification accuracy by using DE. Experimental results indicate that the DE is performed better than many traditional classical algorithms as well as EAs.

### 4.2.2 Solution Representation

The solution representation of DE Classifier is similar to Michigan LCS approach in which each vector represents as a single rule. In our solution the conditional operators as well as the comparative operators are fixed through the search process. Population of DE algorithm generated within the domain such that the lower limit of domain is minimum value of attributes and the upper limit of domain contains the maximum values of attributes. Mutation is performed with randomly selected vectors with in domain. Recombination depends on Cr factor if Cr is greater than random number then it will mutate. Mutated vector is selected for calculating the fitness and if fitness of the selected vector is better than old vector then it replaces the old one.

#### 4.2.2.1 Attributes as Feature Vector

The attributes of the dataset represent each individual vector such that each attribute is mapped on feature vector and it is called if part of the rule; Class label represents then part of the rule.

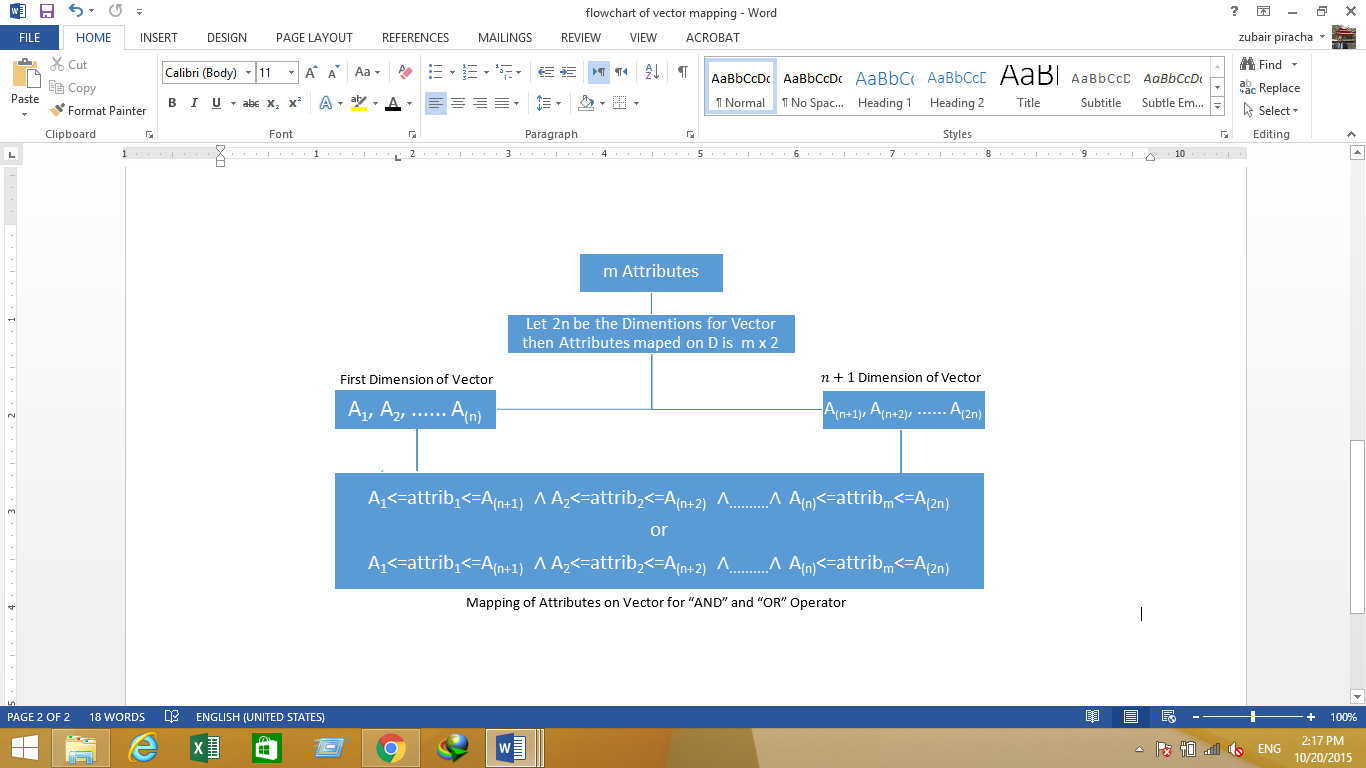
A vector is consisting of 2 times of attributes i.e. if we have n attributes then there are 2n dimension of a vector (i.e. for D=m x 2) such that if 1st dimension of vector represent the left side of condition then the (n+1)th dimension of the vector represents the right side the condition respectively. Figure 4.1 shows the mapping of attributes.

For example if we have 3 attributes than the vector can be consist of 6 dimensions. The first three dimensions of vector represent the left side of the condition for each attribute and the last three dimensions represent the right side of the condition for each attribute. Table 4.1 shows the example of vector for 6 dimensions.

|  |  |
| --- | --- |
| Type of Condition | Examples |
| AND | (D1<= Attribute1<= D4) AND (D2<=Attribute2<=D5) AND (D3<=attribute3<=D6) |
| OR | (D1<= Attribute1<= D4) OR (D2<=Attribute2<=D5) OR (D3<=attribute3<=D6) |

Table 4.1: Examples of Attributes as Vector

An Attribute is not a part of “if” statement if the left hand side of the condition is greater than the right hand site.

**Figure 4.1: Mapping of Attributes on Feature Vector**

#### 4.2.2.2 If-Then Rule Representation

As mentioned earlier, each individual vector consists of conditions to composing a given rule antecedent. An individual is a set of less than or equal to *n-1*operators, where *n* is the number of attributes. The equation below represents a rule with “AND” and “OR” operator respectively:

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.1)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.2)

A*j* represents the jth attribute; denotes the first column of jth dimension attribute for i-th vector, denotes the (n+1)th column of jth dimension for i-th vector and so on. We use fixed condition for the entire search, if algorithm use AND (**∧)** condition the all conditions are AND throughout the searching process and if the algorithm use OR (**∨)** condition then all condition are OR.

#### 4.2.2.3 Initialization and Ranges of Vector

Let d be the total dimensions and i be the total number of feature vectors to be initialized and j are total attributes in dataset then the algorithm 4.2 below shows how to initialize the feature vector within domain. The Algorithm 4.1 below shows the minimum and maximum range of attributes.

* **Begin**
  + **Input dataset**
  + **Let i be the number of attributes in dataset**
  + **For (i)**
    - **Range [ai, bi] = min(attr[i] ), max(attr[i])**
  + **End**
* **end**

Algorithm 4.1: Range of Vectors

Algorithm 4.2: Initialization of Vector

* **d=2\*j; i=d\*10;**
* **for (i)**
  + **for (j)**
    - **Vector(i,j)=range(j,1)+(range(j,2)-range(j,1))\*rnd**
    - **Vector(i,j+j)=range(j,1)+(range(j,2)-range(j,1))\*rnd**
  + **end**
* **end**

In the above algorithm range (j,1) represent the minimum value of attribute j and range(j,2) represent the maximum value of j. The formula for random distribution within domain is define below

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.3)

#### 4.2.2.4 Mutation Algorithm

* **mutation(vector, F)**
  + **[row dim]=size(vector);**
  + **for (i=row)**
    - **while(r1==r2||r1==r3||r2==r3||r1==i||r2 ==i||r3==i)**
      * **r1=round(1 + (row-1)\*rand);**
      * **r2=round(1 + (row-1)\*rand);**
      * **r3=round(1 + (row-1)\*rand);**
    - **end;**
  + **m\_Vector(i)= round( Vector(r1)+ F\*( Vector(r2) -Vector(r3) ) );**
  + **r1=0;r2=0;r3=0;**
  + **end**
  + **return (m\_Vector)**
* **end**

Mutation strategy for differential evolution classifier is defined below.

Algorithm 4.3: Mutation Algorithm of Differential Evolution Classifier

In the above algorithm r1, r2, r3 are randomly selected index of population and F is scaling factor.

#### 4.2.2.5 Recombination

Algorithm 4.4 for recombination for DE classifier is defined below.

* **function recombination(m\_Vector,Vector,range,Cr)**
  + **[row dim]=size(Vector);**
  + **for (i=row)**
    - **for (j=dim)**
      * **if (rand <=Cr || j==round(rand\*dim))**
        + **r\_vector(i,j)=m\_vector(i,j);**
      * **else**
        + **r\_Vector(i,j)=vector(i,j);**
      * **end;**
    - **end;**
  + **end**
  + **if attribute of r\_Vector(i) is exceed from domain then set it with in domain**
  + **return r\_vector**
* **end**

Algorithm 4.4: Recombination Algorithm for Differential Evolution

### 4.2.3 Proposed Fitness Function

We proposed two fitness functions. One is accuracy based fitness function and other is AUC fitness function. Accuracy based fitness function is simple because it is the ratio of all correctly classified instance and total number of instance whereas AUC fitness function is the product of sensitivity and specificity. The reason for proposing two fitness function is that the accuracy fitness function covers correctly classified instance but it neglect the weight of the class but AUC fitness function has the ability to predict proportion of both correctly and incorrectly classified instance i.e. accuracy and coverage although sometimes it decreases accuracy.

To understand the fitness functions there is need to be explaining confusion matrix and some related terms. Table 4.2 shows the confusion matrix for two class classification.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **True Class** | |
|  |  | **Positive** | **Negative** |  |
| **Test Class** | **Positive** | **True Positive  (TP)** | **False Positive  (FP)** | **TP +FP** |
| **Negative** | **False Negative  (FN)** | **True Negative  (TN)** | **FN + TN** |
|  |  | **TP + FN**  **(P)** | **FP + TN**  **(N)** |  |

Table 4.2: Confusion Matrix for Two Classes

The Term related to confusion matrix is defined below:

* + TP= No. of correctly classified positive instance
  + FP= No. of Negative instance classified as positive
  + FN= No. of Positive instance classified as Negative
  + TN= No. of correctly classified negative instance

False Positive Rate (FP Rate) is the ratio between false positive and total negative instance of true class. FP Rate is known as type-I error. Mathematically it is defined as

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.4)

True Positive Rate (TP Rate) is the ratio between true positive and total positive instance of true class mathematically it is defined as:

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . (4.5)

OR

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.6)

False Negative Rate (FN Rate) is the ratio of false negative and total Instance. FP Rate is known as type-II error. Mathematically it is defined as

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.7)

Sensitivity also known as recall is the defined as the probability of test to find true class. The equation of sensitivity is given below

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.8)

Specificity is defined as

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . (4.9)

So we construct two fitness functions i.e. accuracy and AUC fitness. The accuracy fitness functions are defined below:

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . (4.10)

The AUC fitness function is described as

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.11)

We named AUC (Area Under Cover) because the AUC involved two terms i.e. sensitivity and specificity and the formula for AUC is defined as

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.12)

### 4.3.4 Pseudo-code and Flow Chat

The proposed algorithm us the standard methodology as a simple DE use. The Pseudo-code of proposed Algorithm is defined as follows:

* **DE(MaxIter,F,Cr,Dataset)**
  + **Count the Number Positive and Negative Class**
  + **Set the minimum and maximum range of training dataset**
  + **Initialize the total number of Vector uniformly distributed within search space.**
  + **fitness = Fitness(Vector)**
  + **Loop**
    - **for i=1:MaxIter**
      * **[m\_Vector] = mutation(Vector,F);**
      * **[R\_Vector]=recombination(m\_Vector,Vector, range,Cr);**
      * **If fitness(R\_Vector) > fitness(Vecotor)** 
        + **Vector =R\_vector**
      * **end**
    - **end;**
  + **end**
  + **Fitness = Fitness(Vector)**
  + **Best\_Vector=max(fitness);**
  + **Return best\_Vector as rule**
* **end**

Algorithm 4.5: Pseudo code for Proposed Algorithm

**Mutation**

**Recombination**

**Selection**

**No**

**Yes**

**If Termination Criteria is satisfied**

**Extract Best Rule**

**Test Dataset**

**Train Dataset**

**Initialize Random Population between Min Max of Dataset**

**Evaluate Fitness**

**Apply Rule**

**Accuracy**

**Dataset**>=30

Figure 4.2: Flow chart for Proposed Algorithm

# Chapter 5

# Detail analysis of Habermen’s Survival Dataset

## 5.1 Dataset Preparation

The dataset used in this experiment is Haberman’s Survival Dataset downloaded from UCI machine learning repository <http://www.ics.uci.edu/>. The detail of the dataset is given in table 5.1:

|  |  |
| --- | --- |
| Data Type | Multivariate |
| No. of Attributes | 3 |
| No. of Instances | 306 |
| No. of Classes | 2 |
| Missing values | No |
| Data Type of Attributes | Decimal |

Table 5.1: Details of Dataset

Table 5.2 represents the class distribution

|  |  |
| --- | --- |
| Class | No. of Instance |
| the patient survived 5 years or longer | 225 |
| the patient died within 5 year | 81 |

Table 5.2: Class Distribution of Dataset

Table 5.3 shows the minimum and maximum ranges of attributes

|  |  |
| --- | --- |
| Attribute Names | Range |
| Age of Patient at time of operation | 30 – 83 |
| Patient's year of operation | 58 – 69 |
| Number of positive axillary nodes detected | 0 – 52 |

Table 5.3: Ranges of Dataset

## 5.2 Experimental Setup and Parameter Setting

The proposed algorithm is written in Matlab. Matlab has been chosen as development tool because it is an easy to use and platform independent tool for implementing multi-dimensional arrays. The Particulars of machine and software version used for the experiment are given in table 5.4.

As we mentioned earlier DE has different mutation schemes but we used “DE/rand/1/bin” mutation scheme [29]. The said DE scheme is defined below.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (5.1)

We chose 0.9 as crossover rate (Cr) which increases the convergence speed [29] and value of f is 0.5. We choose number of population (NP) as 10D and it is chosen from classical DE [50]. The number of generations for DE is 200. As per current finding no other one propose classification using DE to discover if-then rules that is why parameters from classical DE is used as benchmark moreover “DE/rand/1/bin” mutation scheme is used in this thesis which is most widely used scheme in literature [29] . The Table 5.5 presents the DE parameters setting used in the proposed technique. The results have been gathered using k-fold cross-validation where the value of k is 10.

|  |  |
| --- | --- |
| Processor | Intel(R) Core(TM)2 CPU @ 1.66 GHz |
| Physical Memory (RAM) | 1.00 GB |
| Operating System | Windows Xp Professional |
| Development Tool | Matlab R2010a Version 7.10.0.499 |

Table 5.4: Particulars of Machines and software’s

## 5.3 K-Fold Cross Validations

Well known K-fold cross-validation procedure is used for prediction of accuracy where the value of k is chosen as 10, which is common choice in literature. In this procedure the dataset is divided into 10 mutually exclusive and exhaustive partitions such that the training dataset contains 90% of total dataset and remaining is test dataset. K-fold Cross validation is resemble to random sub-sampling. Its advantage is that all instance in the data eventually used for training and testing.

## 5.4 Area under the ROC Curve (AUC)

In machine learning, there are many ways for comparing performance of classifiers. Measuring the performance of learning algorithm is important because it tells us how and why a classifier is better. There are several techniques for performance evaluation for classifiers like accuracy, recall, specificity, precision, f-measure and AUC etc.

For biomedical domain the Receiver Operating Characteristic (ROC) curves are generally used. ROC is the curve between sensitivity (true positive Rates) and 1-Specificity (false positive rates) and shows a relation between them. The Area under the ROC curve (AUC) is the most common index in ROC and it is evaluation criteria for the classifier as well as comparing the analytical performance of two or more diagnostic classifiers [52] [53]. The advantage of the Area under the ROC curve (AUC) is that it has an ability to predict both correctly and incorrectly classified proportion [54].

Formula of AUC for one run [55] is defined as below:

The above formula is also known as balanced accuracy [56] and discriminant ability [57].

## 5.5 Weka an Keel Machine learning tools

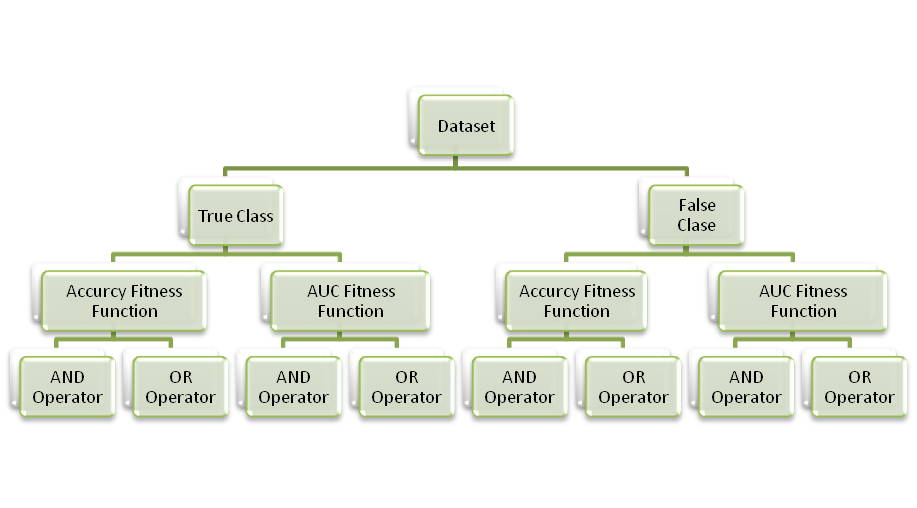
For comparative analysis we use two open source machine learning tools i.e. KEEL and Weka; which is defined bellow:

**KEEL (Knowledge Extraction based on Evolutionary Learning):** KEEL is a machine learning software developed in Java. It is an open source software. It supports the Data mining tasks like classification, regression and clustering etc. It pays special attention of evolutionary learning [58]. For comparisons of evolutionary algorithms with DE-classifier we use KEEL.

**WEKA (The Waikato Environment for Knowledge Analysis):** WEKA is an open source machine learning software which provides wide range of algorithm and data preprocessing tools for researchers. WEKA supports attribute selection, regression, classification, clustering and rule mining algorithms. It also provides visualization facilities and statistical results of algorithms etc [59].

## 5.6 Analysis on Haberman’s Survival Dataset

Analysis of Haberman’s Survival Dataset with DE-Classifier is divided into 2 Sessions. The first session use the ‘patient will survive’ as true class and second session uses the ‘patient will not survive’ as true class. Each session use two fitness functions (Accuracy fitness and AUC fitness function) with AND and OR operator. Figure 5.1 shows the hierarchy of the experiment.

Figure 5.1: Hierarchy of the Experiment

### 5.6.1 Training DE with Target Class Patient will Survive

In this Section the experiments are performed on “Patient will Survive 5 year or longer” as a default class/true class. This section is further divided into two subsection based on fitness functions i.e. Accuracy fitness function and AUC fitness function

#### 5.6.1.1 Accuracy Fitness Function

In this section DE use accuracy fitness function we named this classification method as DE-Accuracy. The experiment is performed for both AND and OR operator with “Patient will survive” as a true class. The Accuracy fitness function is defined as below.

##### 5.6.1.1.1 Rule Representation with Detail Result using AND Operator

The 10 best rules for DE-Accuracy for true class “Patient will survive” with AND operator is extracted by using 10 fold cross validation which is defined in table 5.5. All the rules depend on ‘axillary nodes detected’ and in most rules the value of axillary nodes detected is less than 9.

|  |  |  |  |
| --- | --- | --- | --- |
| S# | Rules | Accuracy (Training dataset) | Accuracy (Test dataset) |
| 1 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 75.72% | 76.67% |
| 2 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 76.36% | 70.97% |
| 3 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 77.09% | 64.52% |
| 4 | if ( axillary nodes detected<= 10) then patient will Survive within 5 year | 76.36% | 67.74% |
| 5 | if ( axillary nodes detected<= 16) then patient will Survive within 5 year | 73.45% | 83.87% |
| 6 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 75.27% | 80.65% |
| 7 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 77.09% | 64.52% |
| 8 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 76.09% | 73.33% |
| 9 | if ( axillary nodes detected<= 10) then patient will Survive within 5 year | 73.91% | 90.00% |
| 10 | if ( axillary nodes detected<= 10) then patient will Survive within 5 year | 75.00% | 80.00% |

Table 5.5: 10 Best Rules using DE-Accuracy with AND Operator for “Patient will Survive” as True Class

Table 5.6 shows the detailed 10-fold result of accuracy, true positive (survived) and true negative (not survived) for DE-Accuracy. The average accuracy for trainings and test data set is 75.64% and 75.23% respectively.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10-fold | Training Dataset Result | | | Test Dataset Result | | |
| **Accuracy** | **Survived** | **Not Survived** | **Accuracy** | **Survived** | **Not Survived** |
| 1 | 75.72% | 90.69% | 33.33% | 76.67% | 90.48% | 44.44% |
| 2 | 76.36% | 91.50% | 36.00% | 70.97% | 84.00% | 16.67% |
| 3 | 77.09% | 90.82% | 35.29% | 64.52% | 88.89% | 30.77% |
| 4 | 76.36% | 92.16% | 30.99% | 67.74% | 95.24% | 10.00% |
| 5 | 73.45% | 95.98% | 14.47% | 83.87% | 96.15% | 20.00% |
| 6 | 75.27% | 90.59% | 32.88% | 80.65% | 91.30% | 50.00% |
| 7 | 77.09% | 90.34% | 36.76% | 64.52% | 94.44% | 23.08% |
| 8 | 76.09% | 91.63% | 32.88% | 73.33% | 81.82% | 50.00% |
| 9 | 73.91% | 91.46% | 28.57% | 90.00% | 100.00% | 25.00% |
| 10 | 75.00% | 92.50% | 28.95% | 80.00% | 92.00% | 20.00% |
| AVG: | **75.64%** | **91.77%** | **31.01%** | **75.23%** | **91.43%** | **29.00%** |

Table 5.6: DE-Accuracy Detail Results with AND Operator and “Patient will Survive” as True Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion Matrix** | | | |  |
|  |  | **Predicted** | |  |
|  |  | **Survived** | **Not Survived** |  |
| **Actual** | **Survived** | **206** | 19 | 225 |
| **Not Survived** | 56 | **25** | 81 |
|  |  | 262 | 44 |  |

Table 5.7: Confusion Matrix for DE-Accuracy with AND Operator and “Patient will Survive” as True Class

##### 5.6.1.1.2 Rule Representation with Detail Results using OR Operator

In this experiment, the 10 best rules for DE-Accuracy for true class “Patient will survive” with OR operator are extracted with 10-fold cross validation. All the rules depend on axillary nodes detected with the combination of Age factor and year of operation. Here we observe that the value of axillary node detected is between less than 4 to less than 10 (Table 5.8).

|  |  |  |  |
| --- | --- | --- | --- |
| S# | Rules | Accuracy (Training dataset) | Accuracy (Test dataset) |
| 1 | if ( 36 <=AGE<= 43) OR ( 60 <=year of operation <= 61) OR ( axillary nodes detected<= 3) then patient will Survive within 5 year | 75.36% | 83.33% |
| 2 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 76.36% | 70.97% |
| 3 | if (year of operation = 61) OR ( axillary nodes detected<= 9) then patient will Survive within 5 year | 76.00% | 74.19% |
| 4 | if ( 61 <=AGE<= 64) OR ( axillary nodes detected<= 10) then patient will Survive within 5 year | 76.00% | 70.97% |
| 5 | if ( axillary nodes detected<= 10) then patient will Survive within 5 year | 76.00% | 70.97% |
| 6 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 77.09% | 64.52% |
| 7 | if ( axillary nodes detected<= 8) then patient will Survive within 5 year | 77.45% | 61.29% |
| 8 | if ( 66 <=year of operation <= 68) OR ( axillary nodes detected<= 10) then patient will Survive within 5 year | 76.45% | 80.00% |
| 9 | if (year of operation = 61) OR ( axillary nodes detected<= 8) then patient will Survive within 5 year | 76.09% | 83.33% |
| 10 | if ( 64 <=year of operation <= 67) OR ( axillary nodes detected<= 10) then patient will Survive within 5 year | 76.09% | 80.00% |

Table 5.8: 10 best Rules using DE-Accuracy with OR Operator for “Patient will Survive” as True Class

Table 5.9 shows the 10-fold result of accuracy, true positive (survived) and true negative (not survived) for DE-Accuracy with OR operator. The average accuracy for trainings and test data set is 76.29% and 73.96% respectively.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10-fold | Training dataset Result | | | Test Dataset Result | | |
| **Accuracy** | **Survived** | **Not Survived** | **Accuracy** | **Survived** | **Not Survived** |
| 1 | 75.36% | 87.94% | 42.86% | 83.33% | 88.46% | 50.00% |
| 2 | 76.36% | 91.58% | 34.25% | 70.97% | 82.61% | 37.50% |
| 3 | 76.00% | 92.54% | 31.08% | 74.19% | 95.83% | 0.00% |
| 4 | 76.00% | 94.06% | 26.03% | 70.97% | 86.96% | 25.00% |
| 5 | 76.00% | 92.65% | 28.17% | 70.97% | 90.48% | 30.00% |
| 6 | 77.09% | 90.29% | 37.68% | 64.52% | 94.74% | 16.67% |
| 7 | 77.45% | 92.16% | 35.21% | 61.29% | 76.19% | 30.00% |
| 8 | 76.45% | 94.55% | 27.03% | 80.00% | 100.00% | 14.29% |
| 9 | 76.09% | 91.58% | 33.78% | 83.33% | 100.00% | 28.57% |
| 10 | 76.09% | 96.06% | 20.55% | 80.00% | 95.45% | 37.50% |
| AVG: | **76.29%** | **92.34%** | **31.66%** | **73.96%** | **91.07%** | **26.95%** |

Table 5.9: DE-Accuracy Detail Results with OR Operator and “Patient will Survive” as True Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion Matrix** | | | |  |
|  |  | **Predicted** | |  |
|  |  | **Survived** | **Not Survived** |  |
| **Actual** | **Survived** | **208** | 17 | **225** |
| **Not Survived** | 55 | **26** | **81** |
|  |  | 263 | 43 |  |

Table 5.10: Confusion Matrix using DE-Accuracy with OR Operator for “Patient will Survive” as True Class

#### 5.6.1.2 AUC Fitness Function

In this section DE use AUC fitness function we named this classification method as DE-AUC. The experiment is performed for both AND and OR operator with “Patient will survive” as true class. The AUC Fitness function is given below:

##### 5.6.1.2.1 Rule Representation with Detail Results using AND Operator

The 10 best rules for DE-AUC with AND operator and “Patient will survive” as true class are extracted. All the rules depend on axillary nodes detected where only one rule shows age factor also. In most rules all rues, the value of axillary node detected is between less than 2 or less then and equal to 2 (Table 5.11).

|  |  |  |  |
| --- | --- | --- | --- |
| S# | Rules | Accuracy (Training dataset) | Accuracy (Test dataset) |
| 1 | if ( axillary nodes detected<= 1) then patient will Survive within 5 year | 64.86% | 83.33% |
| 2 | if (axillary nodes detected<= 2) then patient will Survive within 5 year | 69.82% | 70.97% |
| 3 | if ( axillary nodes detected<= 3) then patient will Survive within 5 year | 72.00% | 70.97% |
| 4 | if ( axillary nodes detected<= 2) then patient will Survive within 5 year | 71.64% | 54.84% |
| 5 | if ( axillary nodes detected<= 1) then patient will Survive within 5 year | 66.55% | 67.74% |
| 6 | if ( 31 <=AGE<= 74) AND ( axillary nodes detected<= 3) then patient will Survive within 5 year | 70.55% | 70.97% |
| 7 | if ( axillary nodes detected<= 2) then patient will Survive within 5 year | 70.91% | 61.29% |
| 8 | if ( axillary nodes detected<= 4) then patient will Survive within 5 year | 73.55% | 80.00% |
| 9 | if ( axillary nodes detected<= 1) then patient will Survive within 5 year | 67.03% | 63.33% |
| 10 | if ( axillary nodes detected<= 1) then patient will Survive within 5 year | 67.39% | 60.00% |

Table 5.11: 10 best Rules using DE-AUC with AND Operator for “Patient will Survive” as True Class

Table 5.12 shows the 10-fold result of accuracy, true positive (survived) and true negative (not survived) for DE-AUC. The average accuracy for training dataset is 69.43% whereas average of test dataset is and 68.34%.After the analysis of the result, it is found that by changing the fitness function (AUC instead of Accuracy) overall accuracy will decrease but the coverage of both classes is increased (e.g. the accuracy of false class in the table 5.6is 31.01% whereas in the above experiment the accuracy of false class is 60.40%)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10-fold | Training dataset Result | | | Test Dataset Result | | |
| **Accuracy** | **Survived** | **Not Survived** | **Accuracy** | **Survived** | **Not Survived** |
| 1 | 64.86% | 64.88% | 64.79% | 83.33% | 85.00% | 80.00% |
| 2 | 69.82% | 72.77% | 61.64% | 70.97% | 78.26% | 50.00% |
| 3 | 72.00% | 79.70% | 50.68% | 70.97% | 73.91% | 62.50% |
| 4 | 71.64% | 75.62% | 60.81% | 54.84% | 54.17% | 57.14% |
| 5 | 66.55% | 65.84% | 68.49% | 67.74% | 73.91% | 50.00% |
| 6 | 70.55% | 77.11% | 52.70% | 70.97% | 70.83% | 71.43% |
| 7 | 70.91% | 73.66% | 62.86% | 61.29% | 70.00% | 45.45% |
| 8 | 73.55% | 82.18% | 50.00% | 80.00% | 95.65% | 28.57% |
| 9 | 67.03% | 68.32% | 63.51% | 63.33% | 52.17% | 100.00% |
| 10 | 67.39% | 67.00% | 68.49% | 60.00% | 63.64% | 50.00% |
| AVG: | **69.43%** | **72.71%** | **60.40%** | **68.34%** | **71.75%** | **59.51%** |

Table 5.12: DE-AUC Detail Results with AND Operator and “Patient will Survive” as True Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion Matrix** | | | |  |
|  |  | **Predicted** | |  |
|  |  | **Survived** | **Not Survived** |  |
| **Actual** | **Survived** | 164 | 61 | 225 |
| **Not Survived** | 32 | 49 | 81 |
|  |  | 196 | 110 |  |

Table 5.13: Confusion Matrix for DE-AUC with AND Operator and “Patient will Survive” as True Class

##### 5.6.1.2.2 Rule Representation with Detail Results using OR Operator

10 best rules are extracted with DE-AUC whereas OR operator and “Patient will survive” as true class is used in this experiment. All the extracted rules in table 5.14 depend on axillary nodes detected and its value is between less than 2 to less than 4.

|  |  |  |  |
| --- | --- | --- | --- |
| S# | Rules | Accuracy (Training dataset) | Accuracy (Test dataset) |
| 1 | if ( axillary nodes detected<= 1) then patient will Survive within 5 year | 66.67% | 66.67% |
| 2 | if ( axillary nodes detected<= 3) then patient will Survive within 5 year | 73.45% | 58.06% |
| 3 | if ( axillary nodes detected<= 2) then patient will Survive within 5 year | 69.82% | 70.97% |
| 4 | if ( axillary nodes detected<= 2) then patient will Survive within 5 year | 69.09% | 77.42% |
| 5 | if ( axillary nodes detected<= 2) then patient will Survive within 5 year | 69.82% | 70.97% |
| 6 | if ( axillary nodes detected<= 2) then patient will Survive within 5 year | 70.18% | 67.74% |
| 7 | if ( AGE<= 43) OR ( axillary nodes detected<= 2) then patient will Survive within 5 year | 76.36% | 64.52% |
| 8 | if ( axillary nodes detected<= 1) then patient will Survive within 5 year | 65.94% | 73.33% |
| 9 | if ( axillary nodes detected<= 2) then patient will Survive within 5 year | 70.65% | 63.33% |
| 10 | if ( axillary nodes detected<= 1) then patient will Survive within 5 year | 66.30% | 70.00% |

Table 5.14: 10 best Rules using DE-AUC with OR Operator for “Patient will Survive” as True Class

Table 5.15 shows detail results of accuracy, true positive (survived) and true negative (not survived) for DE-AUC with OR operator for “patient will survive” as true class the average accuracy for training and test data set is 69.83% and 68.30% respectively. The results also shows that the AUC covers both Classes i.e. instance that correctly classified for class survived and not survived are 72.94% and 61.36% respectively for training dataset.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10-fold | Training dataset Result | | | Test Dataset Result | | |
| **Accuracy** | **Survived** | **Not Survived** | **Accuracy** | **Survived** | **Not Survived** |
| 1 | 66.67% | 67.80% | 63.38% | 66.67% | 55.00% | 90.00% |
| 2 | 73.45% | 79.41% | 56.34% | 58.06% | 76.19% | 20.00% |
| 3 | 69.82% | 73.60% | 60.26% | 70.97% | 71.43% | 66.67% |
| 4 | 69.09% | 71.64% | 62.16% | 77.42% | 87.50% | 42.86% |
| 5 | 69.82% | 73.13% | 60.81% | 70.97% | 75.00% | 57.14% |
| 6 | 70.18% | 73.50% | 61.33% | 67.74% | 72.00% | 50.00% |
| 7 | 76.36% | 82.91% | 59.21% | 64.52% | 76.92% | 0.00% |
| 8 | 65.94% | 66.67% | 63.77% | 73.33% | 66.67% | 83.33% |
| 9 | 70.65% | 74.02% | 61.11% | 63.33% | 66.67% | 55.56% |
| 10 | 66.30% | 66.67% | 65.22% | 70.00% | 66.67% | 75.00% |
| AVG: | **69.83%** | **72.94%** | **61.36%** | **68.30%** | **71.40%** | **54.06%** |

Table 5.15: DE-AUC detail results with OR Operator and “Patient will Survive” as True Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion Matrix** | | | |  |
|  |  | **Predicted** | |  |
|  |  | **Survived** | **Not Survived** |  |
| **Actual** | **Survived** | **164** | 61 | 225 |
| **Not Survived** | 31 | **50** | 81 |
|  |  | 195 | 111 |  |

Table 5.16: Confusion Matrix for DE-AUC with OR Operator and “Patient will Survive” as True Class

#### 5.6.1.3 Comparison of Fitness Functions and Operators for Patient will survive as default class

The table 5.17 shows comparison of Operators (i.e. AND and OR Operator) and fitness functions (i.e. Accuracy and AUC). The results shows that results of OR with operator (76.28%) dominate the AND operator for accuracy fitness function and AUC fitness function in term of accuracy and AUC point.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | AND Operator | | OR Operator | |
| Classifier | **DE-Accuracy** | **DE-AUC** | **DE-Accuracy** | **DE-AUC** |
| Accuracy (%) | 75.68% | 69.45% | **76.28%** | **69.93%** |
| AUC Point | 0.614 | 0.666 | **0.620** | **0.673** |

Table 5.17: Comparison of Operator and Fitness Function for Patient will Survive

#### 5.6.1.4 Comparison of DE with Traditional Algorithms

|  |  |  |
| --- | --- | --- |
| Algorithm | Accuracy | AUC Point |
| DE-Accuracy(OR) | **76.28%** | **0.620** |
| DE-AUC(OR) | 69.93% | **0.673** |
| JRip (RIPPER) | 73.90% | 0.605 |
| Decision Table | 72.50% | 0.521 |
| J48 Tree (C4.5) | 71.90% | 0.584 |
| Naive Bayes Classifier | 74.80% | 0.576 |
| Multilayer Perceptron | 72.90% | 0.590 |

Table 5.18: Comparison of Proposed Technique (Patient will Survive as True Class) with Traditional algorithms

When we compare the proposed algorithms (the best result chosen from AND and OR Operator) with other traditional algorithms then it is found that the DE-Accuracy dominated the traditional algorithms for both accuracy and AUC point where as the DE-AUC dominated the traditional algorithms for AUC point. DE-Accuracy also dominate the DE-AUC for accuracy but dominated by the DE-AUC in term of AUC points. This shows that the DE-AUC cover maximum examples for both classes and try to remove the biasness of unbalanced classes.

#### 5.6.1.5 Comparison of DE with Evolutionary Algorithms

|  |  |  |
| --- | --- | --- |
| Algorithm | Accuracy | AUC |
| DE –Accuracy (OR) | **76.28%** | **0.620** |
| DE –AUC (OR) | 69.93% | **0.673** |
| ILGA | 73.86% | 0.506 |
| UCS | 74.43% | 0.548 |
| Falco GP | 74.41% | 0.575 |
| Advance Ant Miner | 74.65% | 0.539 |
| CPSO | 76.11% | 0.590 |
| PSO/ACO | 75.27% | 0.580 |

Table 5.19: Comparison of Proposed Technique (Patient will Survive as True Class) with Evolutionary algorithms

Comparison of Proposed algorithms with other evolutionary algorithms are also performed and found that the DE-Accuracy and DE-AUC both dominated the evolutionary algorithms in term of AUC whereas DE-Accuracy also dominate the evolutionary algorithms as well as DE-AUC in term of accuracy.

### 5.6.2 Training DE with Target Class Patient will not survived

In this section the experiments are performed on “Patient will not Survive 5 year or longer” as a default class/true class. This Section is further divided into two subsection based on fitness functions i.e. Accuracy fitness function and AUC fitness function

#### 5.6.2.1 Accuracy Fitness Function

In this sub-section DE use Accuracy fitness function. The experiment is performed for both AND and OR operator using “Patient will not survive” as true class.

##### 5.6.2.1.1 Rule Representation with Detail Results using AND Operator

The 10 best rules for DE-Accuracy with AND operator and “Patient will not survive” as true class are extracted. In this experiment, the results show that the rules depend on all three attributes i.e. age, year of operation and axillary nodes detected. If we observer the value of axillary nodes detected that we can say the critical value of axillary nodes detected is grater then 8 for the true class patient will not survive the 10 best rules for this experiment are defined below.

|  |  |  |  |
| --- | --- | --- | --- |
| S# | Rules | Accuracy (Training dataset) | Accuracy (Test dataset) |
| 1 | if ( 44 <=AGE<= 67) AND ( year of operation <= 67) AND ( 9 <=axillary nodes detected<= 25) then patient will Not Survive within 5 year | 77.17% | 83.33% |
| 2 | if ( 40 <=AGE<= 80) AND ( year of operation <= 65) AND ( 9 <=axillary nodes detected<= 28) then patient will Not Survive within 5 year | 78.18% | 74.19% |
| 3 | if ( 47 <=AGE<= 71) AND ( 9 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 76.73% | 83.87% |
| 4 | if ( 37 <=AGE<= 67) AND ( 9 <=axillary nodes detected<= 39) then patient will Not Survive within 5 year | 77.45% | 74.19% |
| 5 | if ( 45 <=AGE<= 67) AND ( 10 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 78.55% | 70.97% |
| 6 | if ( 44 <=AGE<= 68) AND ( 5 <=axillary nodes detected<= 32) then patient will Not Survive within 5 year | 78.18% | 70.97% |
| 7 | if ( 44 <=AGE<= 71) AND ( 5 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 78.18% | 74.19% |
| 8 | if ( 44 <=AGE<= 67) AND ( 9 <=axillary nodes detected<= 26) then patient will Not Survive within 5 year | 78.62% | 70.00% |
| 9 | if ( 47 <=AGE<= 69) AND ( 8 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 77.17% | 83.33% |
| 10 | if ( 42 <=AGE<= 67) AND ( 5 <=axillary nodes detected<= 41) then patient will Not Survive within 5 year | 77.90% | 70.00% |

Table 5.20: 10 best Rules using DE-Accuracy with AND Operator for “Patient will not Survive” as True Class

Table 5.21 shows the 10-fold result of accuracy, true positive (survived) and true negative (not survived) for DE-Accuracy with AND operator for “patient will not survive” as true class the average accuracy for test and training dataset are 77.81% and 75.51% respectively.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10-fold | Training Dataset Result | | | Test Dataset Result | | |
| **Accuracy** | **Not Survived** | **Survived** | **Accuracy** | **Not Survived** | **Survived** |
| 1 | 77.17% | 28.00% | 95.52% | 83.33% | 16.67% | 100.00% |
| 2 | 78.18% | 28.17% | 95.59% | 74.19% | 20.00% | 100.00% |
| 3 | 76.73% | 24.00% | 96.50% | 83.87% | 33.33% | 96.00% |
| 4 | 77.45% | 31.08% | 94.53% | 74.19% | 42.86% | 83.33% |
| 5 | 78.55% | 24.29% | 97.07% | 70.97% | 18.18% | 100.00% |
| 6 | 78.18% | 43.84% | 90.59% | 70.97% | 12.50% | 91.30% |
| 7 | 78.18% | 41.89% | 91.54% | 74.19% | 42.86% | 83.33% |
| 8 | 78.62% | 30.56% | 95.59% | 70.00% | 11.11% | 95.24% |
| 9 | 77.17% | 25.00% | 95.59% | 83.33% | 44.44% | 100.00% |
| 10 | 77.90% | 41.10% | 91.13% | 70.00% | 50.00% | 77.27% |
| AVG: | **77.81%** | **31.79%** | **94.37%** | **75.51%** | **29.20%** | **92.65%** |

Table 5.21: DE-Accuracy Detail Results with AND Operator and “Patient will not Survive” as True Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion Matrix** | | | |  |
|  |  | **Predicted** | |  |
|  |  | **Not Survived** | **Survived** |  |
| **Actual** | **Not Survived** | **26** | 55 | 81 |
| **Survived** | 13 | **212** | 225 |
|  |  | 38 | 268 |  |

Table 5.22: Confusion Matrix for DE-Accuracy with AND Operator and “Patient will not Survive” as True Class

##### 5.6.2.1.2 Rule Representation with Detail Results using OR Operator

The 10 best rules for DE-Accuracy with OR operator and “Patient will not survive” as true class are taken out from the experiment. The extracted rules shown in table 5.23 are also depending on axillary nodes detected and the most rules shows that the value of axillary nodes detected is between 9 and 24.

|  |  |  |  |
| --- | --- | --- | --- |
| S# | Rules | Accuracy (Training dataset) | Accuracy (Test dataset) |
| 1 | if ( 9 <=axillary nodes detected<= 26) then patient will Not Survive within 5 year | 75.72% | 80.00% |
| 2 | if ( 9 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 76.73% | 74.19% |
| 3 | if ( 9 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 76.36% | 77.42% |
| 4 | if ( 11 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 77.09% | 67.74% |
| 5 | if ( 9 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 77.09% | 70.97% |
| 6 | if ( 11 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 78.18% | 58.06% |
| 7 | if ( 9 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 76.00% | 80.65% |
| 8 | if ( 9 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 75.36% | 86.67% |
| 9 | if ( 9 <=axillary nodes detected<= 23) then patient will Not Survive within 5 year | 75.36% | 83.33% |
| 10 | if ( 11 <=axillary nodes detected<= 24) then patient will Not Survive within 5 year | 76.09% | 76.67% |

Table 5.23: 10 best Rules using DE-Accuracy with OR Operator for “Patient will not Survive” as True Class

Table 5.24 shows the 10-fold result of accuracy, true positive (survived) and true negative (not survived) for DE-Accuracy with OR operator for “patient will not survive” as true class. The average result for 10-fold cross validation for training data set is 76.40% and for test data set it is 75.57%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10-fold | Training Dataset Result | | | Test Dataset Result | | |
| **Accuracy** | **Not Survived** | **Survived** | **Accuracy** | **Not Survived** | **Survived** |
| 1 | 75.72% | 31.08% | 92.08% | 80.00% | 42.86% | 91.30% |
| 2 | 76.73% | 32.43% | 93.03% | 74.19% | 28.57% | 87.50% |
| 3 | 76.36% | 33.78% | 92.04% | 77.42% | 14.29% | 95.83% |
| 4 | 77.09% | 28.17% | 94.12% | 67.74% | 10.00% | 95.24% |
| 5 | 77.09% | 31.51% | 93.56% | 70.97% | 37.50% | 82.61% |
| 6 | 78.18% | 28.36% | 94.23% | 58.06% | 14.29% | 94.12% |
| 7 | 76.00% | 31.51% | 92.08% | 80.65% | 37.50% | 95.65% |
| 8 | 75.36% | 30.67% | 92.04% | 86.67% | 50.00% | 95.83% |
| 9 | 75.36% | 29.73% | 92.08% | 83.33% | 42.86% | 95.65% |
| 10 | 76.09% | 27.03% | 94.06% | 76.67% | 14.29% | 95.65% |
| AVG: | **76.40%** | **30.43%** | **92.93%** | **75.57%** | **29.21%** | **92.94%** |

Table 5.24: DE-Accuracy Detail Results with OR Operator and “Patient will not Survive” as True Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion Matrix** | | | |  |
|  |  | **Predicted** | |  |
|  |  | **Not Survived** | **Survived** |  |
| **Actual** | **Not Survived** | **25** | 56 | 81 |
| **Survived** | 16 | **209** | 225 |
|  |  | 41 | 265 |  |

Table 5.25: Confusion Matrix for DE-Accuracy with OR Operator and “Patient will not Survive” as True Class

#### 5.6.2.2 AUC Fitness Function

In this sub-section DE use AUC fitness function. The experiment is performed for both AND and OR operator with “Patient will not survive” as true class.

##### 5.6.2.2.1 Rule Representation with Detail Results using AND Operator

The 10 best rules for DE-AUC with AND operator and “Patient will not survive” as true class are defined in table 5.26. The rules depend on two attributes and in most rules the value of age is greater than 40 and the value of axillary nodes detected is greater than 2 or 3.

|  |  |  |  |
| --- | --- | --- | --- |
| S# | Rules | Accuracy (Training dataset) | Accuracy (Test dataset) |
| 1 | if ( 37 <=AGE<= 75) AND ( 3 <=axillary nodes detected<= 37) then patient will not Survive within 5 year | 72.46% | 70.00% |
| 2 | if ( 44 <=AGE<= 82) AND ( 2 <=axillary nodes detected<= 39) then patient will not Survive within 5 year | 73.09% | 80.65% |
| 3 | if ( 44 <=AGE<= 75) AND ( 2 <=axillary nodes detected<= 40) then patient will not Survive within 5 year | 73.82% | 77.42% |
| 4 | if ( 40 <=AGE<= 80) AND ( 3 <=axillary nodes detected<= 35) then patient will not Survive within 5 year | 73.09% | 77.42% |
| 5 | if ( 38 <=AGE<= 71) AND ( 3 <=axillary nodes detected<= 38) then patient will not Survive within 5 year | 73.82% | 64.52% |
| 6 | if ( 43 <=AGE<= 74) AND ( 2 <=axillary nodes detected<= 50) then patient will not Survive within 5 year | 72.00% | 77.42% |
| 7 | if ( 44 <=AGE<= 79) AND ( 2 <=axillary nodes detected<= 37) then patient will not Survive within 5 year | 76.00% | 54.84% |
| 8 | if ( 43 <=AGE<= 76) AND ( 2 <=axillary nodes detected<= 26) then patient will not Survive within 5 year | 73.19% | 70.00% |
| 9 | if ( 44 <=AGE<= 74) AND ( 2 <=axillary nodes detected<= 35) then patient will not Survive within 5 year | 74.64% | 70.00% |
| 10 | if ( 44 <=AGE<= 75) AND ( 2 <=axillary nodes detected<= 36) then patient will not Survive within 5 year | 74.64% | 70.00% |

Table 5.26: 10 best Rules using DE-AUC with AND Operator for “Patient will not Survive” as True Class

Table 5.27 shows the 10-fold result of accuracy, true positive (survived) and true negative (not survived) for DE-AUC with AND operator for “patient will not survive” as true class. The average accuracy for training and test dataset are 73.67% and 71.23% respectively. As discussed in section 5.6.1.2.1 that using AUC fitness function the accuracy will decrease but coverage will increase. The same scenario exist here if we see the average result for training dataset individually for class survived (59.12%) and not survived (78.91%).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10-fold | Training Dataset Result | | | Test Dataset Result | | |
| **Accuracy** | **Not Survived** | **Survived** | **Accuracy** | **Not Survived** | **Survived** |
| 1 | 72.46% | 57.14% | 78.39% | 70.00% | 75.00% | 69.23% |
| 2 | 73.09% | 55.71% | 79.02% | 80.65% | 81.82% | 80.00% |
| 3 | 73.82% | 58.11% | 79.60% | 77.42% | 71.43% | 79.17% |
| 4 | 73.09% | 58.57% | 78.05% | 77.42% | 45.45% | 95.00% |
| 5 | 73.82% | 56.34% | 79.90% | 64.52% | 60.00% | 66.67% |
| 6 | 72.00% | 57.53% | 77.23% | 77.42% | 87.50% | 73.91% |
| 7 | 76.00% | 63.38% | 80.39% | 54.84% | 30.00% | 66.67% |
| 8 | 73.19% | 63.51% | 76.73% | 70.00% | 14.29% | 86.96% |
| 9 | 74.64% | 61.11% | 79.41% | 70.00% | 44.44% | 80.95% |
| 10 | 74.64% | 59.74% | 80.40% | 70.00% | 50.00% | 73.08% |
| AVG: | **73.67%** | **59.12%** | **78.91%** | **71.23%** | **55.99%** | **77.16%** |

Table 5.27: DE-AUC Detail Results with AND Operator and “Patient will not Survive” as True Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion Matrix** | | | |  |
|  |  | **Predicted** | |  |
|  |  | **Not Survived** | **Survived** |  |
| **Actual** | **Not Survived** | **48** | 33 | 81 |
| **Survived** | 47 | **178** | 225 |
|  |  | 95 | 211 |  |

Table 5.28: Confusion Matrix for DE-AUC with AND Operator and “Patient will not Survive” as True Class

##### 5.6.2.2.2 Rule Representation with detail Results using OR Operator

The 10 best rules for DE-AUC with AND operator and “Patient will not survive” as true class are defined below. The extracted rules show that most of cases axillary nodes detected are greater than 3. Rules also show that age factor is also not ignorable.

|  |  |  |  |
| --- | --- | --- | --- |
| S# | Rules | Accuracy (Training dataset) | Accuracy (Test dataset) |
| 1 | if ( 44 <=AGE<= 75) OR ( 2 <=axillary nodes detected<= 41) then patient will Not Survive within 5 year | 74.28% | 73.33% |
| 2 | if ( 40 <=AGE<= 81) OR ( 2 <=axillary nodes detected then patient will Not Survive within 5 year | 71.64% | 64.52% |
| 3 | if ( 44 <=AGE<= 80) OR ( 2 <=axillary nodes detected<= 49) then patient will Not Survive within 5 year | 74.18% | 67.74% |
| 4 | if ( 41 <=AGE<= 68) OR ( 3 <=axillary nodes detected<= 28) then patient will Not Survive within 5 year | 74.18% | 67.74% |
| 5 | if ( 2 <=axillary nodes detected then patient will Not Survive within 5 year | 66.91% | 64.52% |
| 6 | if ( 44 <=AGE<= 80) OR ( 1 <=axillary nodes detected<= 39) then patient will Not Survive within 5 year | 68.36% | 61.29% |
| 7 | if ( 44 <=AGE<= 71) OR ( 2 <=axillary nodes detected<= 49) then patient will Not Survive within 5 year | 74.18% | 70.97% |
| 8 | if ( 41 <=AGE<= 76) OR ( 2 <=axillary nodes detected<= 25) then patient will Not Survive within 5 year | 71.38% | 73.33% |
| 9 | if ( 44 <=AGE<= 76) OR ( 2 <=axillary nodes detected<= 31) then patient will Not Survive within 5 year | 76.09% | 53.33% |
| 10 | if ( 43 <=AGE<= 74) OR ( 2 <=axillary nodes detected<= 35) then patient will Not Survive within 5 year | 71.74% | 83.33% |

Table 5.29: 10 best Rules using DE-AUC with OR Operator for “Patient will not Survive” as True Class

Table 5.30 shows the 10-fold result of accuracy, true positive (survived) and true negative (not survived) for DE-AUC with OR operator for “patient will not survive” as true class. The results show that DE-AUC with OR operator also covers both classes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10-fold | Training Dataset Result | | | Test Dataset Result | | |
| **Accuracy** | **Not Survived** | **Survived** | **Accuracy** | **Not Survived** | **Survived** |
| 1 | 74.28% | 62.67% | 78.61% | 73.33% | 16.67% | 87.50% |
| 2 | 71.64% | 59.72% | 75.86% | 64.52% | 88.89% | 54.55% |
| 3 | 74.18% | 62.50% | 78.33% | 67.74% | 33.33% | 81.82% |
| 4 | 74.18% | 54.29% | 80.98% | 67.74% | 36.36% | 85.00% |
| 5 | 66.91% | 65.28% | 67.49% | 64.52% | 77.78% | 59.09% |
| 6 | 68.36% | 69.01% | 68.14% | 61.29% | 60.00% | 61.90% |
| 7 | 74.18% | 60.00% | 79.02% | 70.97% | 45.45% | 85.00% |
| 8 | 71.38% | 59.46% | 75.74% | 73.33% | 71.43% | 73.91% |
| 9 | 76.09% | 60.27% | 81.77% | 53.33% | 37.50% | 59.09% |
| 10 | 71.74% | 60.00% | 76.53% | 83.33% | 100.00% | 82.76% |
| AVG: | **72.29%** | **61.32%** | **76.25%** | **68.01%** | **56.74%** | **73.06%** |

Table 5.30: DE-AUC Detail Results with OR Operator and “Patient will not Survive” as True Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion Matrix** | | | |  |
|  |  | **Predicted** | |  |
|  |  | **Not Survived** | **Survived** |  |
| **Actual** | **Not Survived** | **50** | 31 | 81 |
| **Survived** | 53 | **172** | 225 |
|  |  | 103 | 203 |  |

Table 5.31: Confusion Matrix for DE-AUC with OR Operator and “Patient will not Survive” as True Class

#### 5.6.2.3 Comparison of Fitness Functions and Operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | AND Operator | | OR Operator | |
| Classifier | **DE-Accuracy** | **DE-AUC** | **DE-Accuracy** | **DE-AUC** |
| Accuracy (%) | **77.80%** | **73.67%** | 76.39% | 72.30% |
| AUC Point | **0.631** | **0.690** | 0.617 | 0.688 |

Table 5.32: Comparison of Operators and Fitness Functions for Patient will not Survive

The above table 5.32 shows that AND operator dominate the OR operator for both fitness functions in term of accuracy and AUC point. If we compare the AND Operator with their Accuracy and AUC fitness functions then we can found that DE-AUC (0.631) dominate the DE-Accuracy (0.690) in terms of AUC points but DE-Accuracy (77.80%) dominate the DE-AUC (73.67%) in terms of accuracy.

#### 5.6.2.4 Comparison of DE with Traditional Algorithms

|  |  |  |
| --- | --- | --- |
| Algorithm | Accuracy | AUC |
| DE-Accuracy (AND) | **77.80%** | **0.631** |
| DE-AUC (AND) | 73.67% | **0.690** |
| JRip (RIPPER) | 73.90% | 0.605 |
| Decision Table | 72.50% | 0.521 |
| J48 Tree (C4.5) | 71.90% | 0.584 |
| Naive Bayes Classifier | 74.80% | 0.576 |
| Multilayer Perceptron | 72.90% | 0.590 |

Table 5.33: Comparison of Proposed Technique (Patient will not Survive as True Class) with Traditional Algorithms

DE-Accuracy dominated the traditional algorithms in both accuracy and AUC. The DE-AUC fitness function dominated the Decision Table, J48 and Multilayer Perceptron in term of accuracy, if we compare the AUC points then all traditional algorithms as well as DE-Accuracy are dominated by DE-AUC.

#### 5.6.2.5 Comparison of DE with Evolutionary Algorithms

|  |  |  |
| --- | --- | --- |
| Algorithm | Accuracy | AUC |
| DE –Accuracy (AND) | **77.80%** | **0.631** |
| DE-AUC (AND) | 73.67% | **0.690** |
| ILGA | 73.86% | 0.506 |
| UCS | 74.43% | 0.548 |
| Falco GP | 74.41% | 0.575 |
| Advance Ant Miner | 74.65% | 0.539 |
| CPSO | 76.11% | 0.590 |
| PSO/ACO | 75.27% | 0.580 |

Table 5.34: Comparison of Proposed Technique (Patient will not Survive as True Class) with Evolutionary Algorithms

DE-AUC dominated the evolutionary algorithms as well as DE-Accuracy in term of AUC whereas DE-Accuracy dominated the all evolutionary algorithms in term of Accuracy as well as AUC points.

### 5.6.3 Comparison of the DE-Classifier based on Class Selection

We perform the class wise comparison for both fitness functions. The best results are chosen from AND and OR Operators. The result shows that the performance of Classifier is improved with default class patient are not survived for both fitness functions. Table 5.35 shows the class wise comparison for Accuracy fitness function.

|  |  |  |
| --- | --- | --- |
| Class wise Comparison of DE-Accuracy (accuracy fitness function) | | |
| Class | **Accuracy (%)** | **AUC Point** |
| Survived | 76.28% | 0.620 |
| Not survived | **77.80%** | **0.631** |

Table 5.35: Comparison of DE-Accuracy based on Class Selection

The class wise comparison is also performs DE-AUC (table 5.36) and found that the selection of class also impact on the performance of classifier as shown in table 5.35 and 5.36. In our experiment the results show that the performance for default class not survived is better than the class survived as default class.

|  |  |  |
| --- | --- | --- |
| Class wise Comparison for DE-AUC (AUC Fitness function) | | |
| Class | **Accuracy (%)** | **AUC Point** |
| Survived | 69.93% | 0.673 |
| Not survived | **73.67%** | **0.690** |

Table 5.36: Comparison of DE-AUC based on Class Selection

### 5.6.4 Feature Selection

DE classifier also helps us for selection of important/useable features. The above all experiments shows that the most useable feature for haberman’s survival dataset (breast cancer dataset) is axillary nodes detected although some result also represent the age factor in the rules.

### 5.6.5 Comparative Analysis of DE-Classifier on Other Biomedical Datasets

DE-Classifier is implemented on other biomedical datasets like Heat, Pima Indians Diabetes, Mammographic Mass and Blood Transfusions. These dataset is downloaded from UCI Machine Learning Repository. We compare results with traditional classification algorithms as well as Evolutionary classification algorithms. The results are based on training DE using AND operator with default class having less number of instance. Comparative analysis of each dataset is given in next sections. The comparative result shows that the performance of proposed algorithm on said datasets are also better than most of the algorithms.

#### 5.6.5.1 Comparative Analysis of DE-Classifier on Heart Dataset

DE-Classifier is implemented on heart dataset having 270 records and each record contains 13 attributes. The results are compared with traditional classification algorithms and evolutionary algorithms which are shown in table 5.37. The results show that accuracy and coverage of proposed algorithm is better than other algorithms except Decision Table and Multilayer Perceptron. If we considered only coverage then DE-Accuracy (AND) is better than all algorithms. This shows that the performance of purposed algorithm is better for Heart Dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Algorithm** | **Accuracy** | **AUC Point** |
| **Proposed Technique** | **DE-Accuracy (AND)** | **77.04%** | **0.783** |
| **DE-AUC (AND)** | **77.08%** | **0.772** |
| **Traditional Classification Algorithms** | **JRip (RIPPER)** | 71.11% | 0.729 |
| **Decision Table** | **77.78%** | **0.775** |
| **J48 Tree (C4.5)** | 75.19% | 0.749 |
| **Naive Bayes Classifier** | 68.52% | 0.683 |
| **Multilayer Perceptron** | **77.04%** | 0.768 |
| **Evolutionary Algorithms** | **ILGA** | 73.50% | 0.748 |
| **UCS** | 76.77% | 0.766 |
| **Falco GP** | 63.64% | 0.632 |
| **Advance Ant Miner** | 54.81% | 0.401 |
| **CPSO** | 75.56% | 0.753 |
| **PSO/ACO** | 70.37% | 0.703 |

Table 5.37: Comparison of DE-Classifier with Traditional Classification and Evolutionary Algorithms on Heart Dataset

#### 5.6.5.2 Comparative Analysis of DE-Classifier of Pima Indians Diabetes Dataset

The DE-Classifier is implemented on Pima dataset Pima having 768 records and each record contains 8 attributes. DE-Classifier is compared with traditional classification algorithms and evolutionary algorithms which are shown in table 5.38. The results show that accuracy and coverage of proposed algorithm is better than other algorithms except Multilayer Perceptron, Advance Ant Miner and PSO/ACO. If we considered only coverage then DE-Accuracy (AND) is better than all algorithms except Advance Ant Miner and PSO/ACO. This shows that the overall performance of DE-Classifier is better for Heart Dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Algorithm** | **Accuracy** | **AUC Point** |
| **Proposed Technique** | **DE-Accuracy (AND)** | **74.68%** | **0.733** |
| DE-AUC (AND) | 72.51% | 0.705 |
| **Traditional Classification Algorithms** | **JRip (RIPPER)** | 70.77% | 0.698 |
| **Decision Table** | 72.40% | 0.695 |
| **J48 Tree (C4.5)** | 71.95% | 0.692 |
| **Naive Bayes Classifier** | 73.61% | 0.713 |
| **Multilayer Perceptron** | 74.09% | 0.714 |
| **Evolutionary Algorithms** | **ILGA** | 72.76% | 0.719 |
| **UCS** | 70.05% | 0.666 |
| **Falco GP** | 65.89% | 0.604 |
| **Advance Ant Miner** | 73.25% | **0.740** |
| **CPSO** | 65.92% | 0.618 |
| **PSO/ACO** | **75.00%** | **0.742** |

Table 5.38: Comparison of DE-Classifier with Traditional Classification and Evolutionary Algorithms on Pima Indians Diabetes Dataset

#### 5.6.5.3 Comparative Analysis of DE-Classifier of Mammographic Mass Dataset

DE-Classifier is implemented on Mammographic Mass Dataset for comparative analysis. The said dataset is multivariate dataset having 961 records and each record contains 6 attributes. The results show that the accuracy of DE-Classifier is better than other algorithms except Decision Table. If comparison is performed on coverage then DE-Classifier dominates all algorithms. The results shown in table 5.39.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Algorithm** | **Accuracy** | **AUC Point** |
| **Proposed Technique** | **DE-Accuracy (AND)** | **82.09%** | **0.838** |
| **DE-AUC (AND)** | **82.06%** | **0.836** |
| **Traditional Classification Algorithms** | **JRip (RIPPER)** | 76.09% | 0.793 |
| **Decision Table** | **82.52%** | 0.824 |
| **J48 Tree (C4.5)** | 77.76% | 0.686 |
| **Naive Bayes Classifier** | 59.26% | 0.694 |
| **Multilayer Perceptron** | 81.37% | 0.813 |
| **Evolutionary Algorithms** | **ILGA** | 73.66% | 0.757 |
| **UCS** | 78.07% | 0.780 |
| **Falco GP** | 68.24% | 0.686 |
| **Advance Ant Miner** | 54.63% | 0.771 |
| **CPSO** | 74.76% | 0.747 |
| **PSO/ACO** | 81.89% | 0.818 |

Table 5.39: Comparison of DE-Classifier with Traditional Classification and Evolutionary Algorithms on Mammographic Mass Dataset

#### 5.6.5.4 Comparative Analysis of DE-Classifier of Blood Transfusion Dataset

DE-Classifier is implemented on Blood Transfusion dataset (748 records with 5 attributes) which is downloaded from UCI. The results show that the performance of DE-Classifier on this dataset is better than other algorithms. DE-Accuracy (AND) dominate all algorithms in term of accuracy i.e. 78.72% and if we compare the coverage of the algorithm then only UCS dominates DE-Accuracy (AND). The detail results shows in Table 5.40.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Algorithm** | **Accuracy** | **AUC Point** |
| **Proposed Technique** | **DE-Accuracy (AND)** | **78.72%** | **0.714** |
| **DE-AUC (AND)** | 65.31% | **0.634** |
| **Traditional Classification Algorithms** | **JRip (RIPPER)** | 52.61% | 0.580 |
| **Decision Table** | 76.20% | 0.381 |
| **J48 Tree (C4.5)** | 77.76% | 0.686 |
| **Naive Bayes Classifier** | 76.79% | 0.695 |
| **Multilayer Perceptron** | 76.20% | 0.381 |
| **Evolutionary Algorithms** | **ILGA** | 75.94% | 0.381 |
| **UCS** | 77.64% | **0.734** |
| **Falco GP** | 75.94% | 0.622 |
| **Advance Ant Miner** | 55.41% | 0.540 |
| **CPSO** | 74.73% | 0.612 |
| **PSO/ACO** | 73.66% | 0.651 |

Table 5.40: Comparison of DE-Classifier with Traditional Classification and Evolutionary Algorithms on Blood Transfusion Dataset

### 5.6.6 Summary of the Experiment

The experimental results show that the overall performance of the DE classifier is good on breast cancer dataset. While comparing on fitness functions (i.e. Accuracy fitness and AUC fitness), the DE-AUC classifier decrease the accuracy of the classifier but it increase the coverage to both classes i.e. If we compare the fitness function with AND operator and default class died, the results show that the true positive rates, true negative rates and accuracy are 31.79%, 94.37% and 77.81% for accuracy fitness function and 59.13%, 78.91% and 73.67% for AUC fitness function respectively. The result shows that the DE-AUC provides coverage to both classes and improves the results even when classes are unbalanced.

The results also show the selection of the class selection has its importance. In over results the class having minimum instance (i.e. patient will not survived within 5 years) has better results for both AND and OR operator as well as for both fitness functions.

All rules extracted from DE-Classifier depend on the feature axillary nodes detected but in some rules age factor also involves. This shows that feature axillary nodes detected is the most important feature for breast cancer dataset and the second one is age of patient. If we analyze the rules in detail we found that, in most rules, the patient will survive if the axillary node detected is less than 9 for accuracy fitness function and less than 3 for AUC fitness function.

When comparison is concerned with other classifiers for both traditional and evolutionary algorithms, the result of the proposed technique is better in coverage i.e. AUC point as well as in accuracy. Figure 5.2 shows that the result for DE-Accuracy for class patient will not survive dominate the all algorithms in term of accuracy and if we concern with AUC point the proposed techniques dominate the all algorithms (figure 5.3).

The performance of DE-Classifier is also good on other biomedical datasets i.e. Heart Dataset, Pima Indians Diabetes, Mammographic Mass and Blood Transfusion. Most of algorithms implemented on above said datasets are dominated by DE-Classifier.

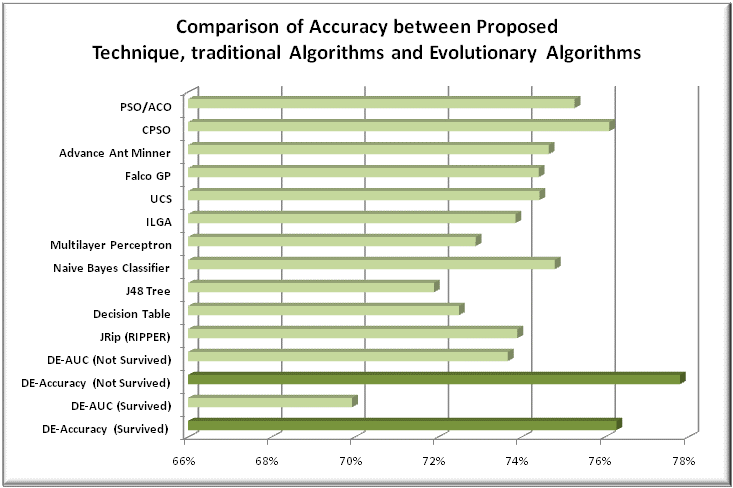


Figure 5.2: Comparison of Accuracy between Proposed Techniques, Traditional Algorithms and Evolutionary Algorithms

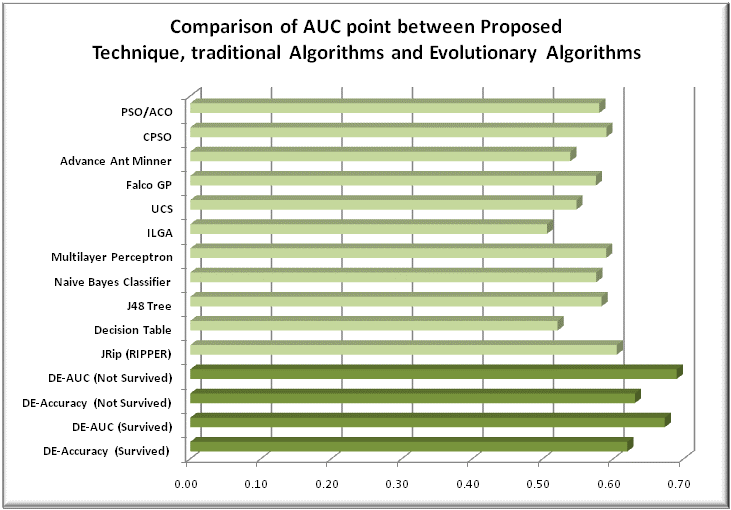


Figure 5.3: Comparison of AUC between Proposed Techniques, Traditional Algorithms and Evolutionary Algorithms

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

First part of this chapter represents the conclusion and findings about the research and the next part is about the future work and guideline for researchers.

## 6.1 Conclusion

We have presented if-then rule based classification which is extracted through Differential Evolution. The research shows that the proposed technique performs good classification results. The rules extracted from DE are comprehensible and easy to understand and improve accuracy as well as coverage. We proposed two fitness functions i.e. **Accuracy fitness function** which Improve the accuracy of classifier and **AUC fitness function** which Improve the coverage for both positive and negative class.

The DE-Accuracy use the accuracy fitness function whereas DE-AUC use the AUC fitness function. If we concern accuracy as evaluation parameter the performance of DE-Accuracy is better than DE-AUC, and if we concern the Area Under the Curve as evolutionary parameter then the performance of DE-AUC is better than DE-Accuracy.

In this thesis, Haberman’s survival data set is used for experiment which comes under the category of biomedical dataset and performance measuring of biomedical data set is AUC in most cases. For the performance of DE-AUC is batter then DE-Accuracy because it takes into account the correctly classified instances of both classes.

We also focus on the selection of class. The results show that the selection of class has its own importance. In our results the selection of negative class(less number of instances) improves the accuracy as well as coverage for both fitness functions.

Selection of Operators also has its importance. We use two operator variants i.e. AND and OR. Our results shows for the positive class, the results of OR variants improves the accuracy as well as coverage, whereas the AND variant improves the accuracy and coverage for negative class. The results are for both fitness functions respectively.

This algorithm also help us for feature selection as we mentioned in previous chapter and we concluded that “no of positive auxiliary node” is most important feature for breast cancer patients.

The proposed work is also compared with traditional as well as evolutionary algorithms on Haberman’s survival dataset and other biomedical datasets and finds that the performance of our technique is encouraging in term of accuracy and coverage.

## 6.2 Future Work

This work has light up a new horizon for future research in the field of rule based classification through differential evolution. It can be extended by using combination of both AND and OR operator in rule encoding. Another option is open for researches by using this technique i.e. multi class classification. Variety of dataset can be used for classification by this technique.

This work has demonstrated that discovery of classification rules for breast cancer survival patient performed successfully by using Differential Evolution and it can also be implemented on different other diseases/biomedical datasets.

# Works Cited

|  |  |
| --- | --- |
| [1] | D. Shi, Z. DingDing and D. Wei, "The Study of Network Traffic Identification Based on Machine Learning Algorithm," in *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on. IEEE,*, 2012. |
| [2] | S. B. Kotsiantis, I. D. Zaharakis and P. E. Pintelas, "Machine learning: a review of classification and combining techniques," *Artif Intell Rev,* vol. 26, no. 3, p. 159–190, 2006. |
| [3] | K. Holzinger, V. Palade, R. Rabadan and A. Holzinger, "Darwin or Lamarck? Future Challenges in Evolutionary Algorithms for Knowledge Discovery and Data Mining," in *Interactive knowledge Discovery and Data Minin in Biomedical Informatics*, Springer, 2014, pp. 35-56. |
| [4] | N. Holden and A. A. Freitas, "A hybrid Particle Swarm/Ant Colony Algorithm for the Calssificaiton of Hierarchical Biological Data," in *IEEE Swarm Intelligence Symposium*, Pasadena, California, USA, 8-10 June, 2005. |
| [5] | A. K. Tanwani and M. Farooq, "Performance Evaluation of Evolutionary Algorithms in Classification of Biomedical Datasets," in *GECCO’09*, July 8–12, 2009. |
| [6] | "http://www.cancer.org/cancer/breastcancer/detailedguide/breast-cancer-what-is-breast-cancer," [Online]. [Accessed 12 July 2014]. |
| [7] | "http://www.cancer.gov/cancertopics/wyntk/breast/page4," [Online]. [Accessed 12 july 2014]. |
| [8] | "http://www.cancer.gov/cancertopics/wyntk/breast/page5," [Online]. [Accessed 11 July 2014]. |
| [9] | S. Gupta, D. Kumar and A. Sharma, "Data Mining Classification Techniques Applied for Breast Cancer Diagnosis and Prognosis," *Indian Journal of Computer Science and Engineering (IJCSE),* vol. 2, no. 2, Apr-May 2011. |
| [10] | GLOBOCAN 2012: Country Fast Stat, International Agency for Research on Cancer (IARC). |
| [11] | Breast Cancer Facts & Figures 2013-2014, Atlanta, Georgia: publication of the American Cancer Society, 2013. |
| [12] | H. Jabeen, "Advancements in Genetic Programming for Data Classification," Islamabad, Pakistan, 2010. |
| [13] | D. H. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation,* vol. I, no. 1, pp. 67-82, Apr 1997. |
| [14] | C. Igel, "No Free Lunch Theorems: Limitations and perspectives of metaheuristics," in *Theory and Principled Methods for the Design of Metaheuristics*, Springer, 2014, pp. 1-23. |
| [15] | L. Rokach and O. Maimon, "Top-Down Induction of Decision Trees Classifiers—A Survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 35, no. 4, pp. 476-487, November 2005. |
| [16] | S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica,* vol. 31, no. 3, pp. 249-268, 2007. |
| [17] | J. Han and M. Kamber, Data Mining: Concepts and Techniques, Second Edition ed., ELSEVIER, 2006. |
| [18] | J. R. Quinlan, "Improved Use of Continuous Attributes in C4.5," *Journal of Artificial Intelligence Research,* vol. 4, no. 1, pp. 77-90, 1996. |
| [19] | J. Su and H. Zhang, "Fast Decision Tree Learning Algorithm," *In Proceedings of the National Conference on Artificial Intelligence,* vol. 21, no. 1, 2006. |
| [20] | Sammut, Claude and G. I. Webb, Eds., Encyclopedia of Machine Learning, New York: Springer-Verlag, 2011. |
| [21] | W. W. Cohen, "Fast Effective Rule Induction," in *Machine Learning: Proceedings of the Twelfth International Conference*, 1995. |
| [22] | J. Fürnkranz and G. Widmer, "Incremental reduced error pruning," in *International Conference on Machine Learning*, 1994. |
| [23] | D. Michie, D. Spiegelhalter and C. Taylor, Machine Learning, Neural and Statistical, D. Michie, D. Spiegelhalter and C. Taylor, Eds., 1994. |
| [24] | S. Tufféry, Data Mining and Statistics for Decision Making, First ed., John Wiley & Sons, Ltd, 2011. |
| [25] | G. Zhang, "Neural Networks for Classification: A Survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 30, no. 4, pp. 451-462, November 2000. |
| [26] | A. K. Jain, J. Mao and K. Mohiudddin, "Artificial Neural Networks: A Tutorial," *IEEE Computational Science & Engineering,* vol. 29, no. 3, pp. 31-44, March 1996. |
| [27] | D. R. Wilson, "Advances in Instance-Based Learning Algorithms," Provo, UT, 1997. |
| [28] | R. Storn and K. Price, "Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces," Berkeley, CA, Tech. Rep. TR-95-012, 1995. |
| [29] | R. Mallipeddi and P. N. Suganthan, "Differential Evolution Algorithm with Ensemble of Parameters and Mutation and Crossover Strategies," in *Proc. Swarm Evol. Memetic Comput. Conf*, Chennai, India, 2010. |
| [30] | D. Zaharie, "A Comparative Analysis of Crossover Variants in Differential Evolution," in *Proceedings of IMCSIT*, 2007. |
| [31] | Q. Bo-Yang and P. Suganthan, "Novel multimodal problems and differential evolution with ensemble of restricted tournament selection," in *IEEE, Evolutionary Computation (CEC)*, 18-23 July, 2010. |
| [32] | J. Alcalá-Fdez, L. Sánchez, S. García, M. J. d. Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández and F. Herrera, "KEEL: a software tool to assess evolutionary algorithms," *Soft Computing,* vol. 13, no. 3, pp. 307-318, February 2009. |
| [33] | A. A. Freitas, "A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery," *Advances in Evolutionary Computing,* p. 819–845, 2003. |
| [34] | M. Mitchell, An Introduction to Genetic Algorithms, Fifth ed., Massachusetts, London: The MIT Press Cambridge, 1999. |
| [35] | J. Bacardit and J. Garrell, "Evolving multiple discretizations with adaptive intervals for a Pittsburgh Rule-Based Learning Classifier System," in *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO2003, Springer*, Heidelberg, 2003. |
| [36] | A. A. Freitas, "A Review of Evolutionary Algorithms for Data Mining," in *Soft Computing for Knowledge Discovery and Data Mining*, University of Kent, UK, Computing Laboratory, 2008. |
| [37] | K. A. D. Jong, W. M. Spears and D. F. Gordon, "Using genetic algorithms for concept learning," *Machine Learning ,* vol. 13, no. 2-3, pp. 161-188, 1993. |
| [38] | C. Janikow, "A knowledge-intensive genetic algorithm for supervised learning," *Machine Learning,* vol. 13, pp. 189-228, 1993. |
| [39] | A. Giordana and F. Neri, "Search-intensive Concept Induction," *Evolutionary Computation,* vol. 3, no. 4, pp. 375-416, 1995. |
| [40] | D. P. Greene and S. F. Smith, "Competition-based induction of decision models from examples," *Machine Learning,* vol. 13, pp. 229-257, 1993. |
| [41] | J. Bacardit, "Pittsburgh Genetic-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time," Barcelona, October 18, 2004. |
| [42] | S.-U. Guan and F. Zhu, "An Incremental Approach to Genetic-Algorithms-Based Classification," *IEEE Transactions on Systems,man, and Cybernetics—Part B,* vol. 35, no. 2, pp. 227-239, April 2005. |
| [43] | H. S. Basheer and M. Al-Maqaleh, "A Genetic Algorithm for Discovering Classification Rules in Data Mining," *International Journal of Computer Applications,* vol. 41, no. 18, p. 0975 – 8887, March 2012. |
| [44] | M. Fidelis, H. Lopes and A. Freitas, "Discovering Comprehensible Classification Rules with a Genetic Algorithm," in *Proc. Congress on Evolutionary Computation-2000 (CEC’2000)*, La Jolla, CA, USA, 2000. |
| [45] | R. Eberhart and J. Kennedy, "Particle Swarm Optimization," in *In Proceedings of IEEE International Conference on Neural Networks*, 1995. |
| [46] | Y. Liu, Z. Qin, Z. Shi and J. Chen, "Rule Discovery with Particle Swarm Optimization," *AWCC 2004, LNCS 3309,* p. 291–296, 2004. |
| [47] | N. Holden and A. A. Freitas, "A Hybrid PSO/ACO Algorithm for Classification," in *GECCO’07*, London, England, United Kingdom, July 7–11, 2007. |
| [48] | "http://archive.ics.uci.edu/ml/datasets/Haberman's+Survival," [Online]. [Accessed 14 March 2012]. |
| [49] | S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation,* vol. 15, no. 1, pp. 4-31, February 2011. |
| [50] | R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization,* vol. 11, pp. 341-359, 1997. |
| [51] | J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," *In Evolutionary Computation, CEC2004,* vol. 2, pp. 1980-1987, 2004. |
| [52] | T. A. Lasko, J. G. Bhagwat, K. H. Zou and L. Ohno-Machado, "The use of receiver operating characteristic curves in biomedical informatics," *Journal of Biomedical Informatics,* vol. 38, no. 5, p. 404–415, 2008. |
| [53] | H. Cheng-Lung and W. Chieh-Jen, "A GA-based feature selection and parameters optimization for support vector machines," *Expert Systems with Applications,* vol. 31, p. 231–240, 2006. |
| [54] | M. S. Boyce, P. R. Vernier, S. E. Nielsen and F. K. Schmiegelow, "Evaluating resource selection functions," *Ecological Modelling,* vol. 157, no. 2-3, pp. 281-300, 2002 . |
| [55] | H. Iba and C. C. Aranha, Practical Applications of Evolutionary Computation to Financial Engineering Robust Techniques for Forecasting, Trading and Hedging, Springer, 2012. |
| [56] | M. Sokolova, N. Japkowicz and S. Szpakowicz, "Beyond Accuracy, F-score and ROC: a Family of Discriminant Measures for Performance Evaluation," *AI'06 Proceedings of the 19th Australian joint conference on Artificial Intelligence: advances in Artificial Intelligence,* pp. 1015-1021, 2006. |
| [57] | L. Williams and Wilkins, Studying a Study & Testing a Test: How to Read the Medical Evidence, 5th ed., R. K. Riegelman, Ed., 2005. |
| [58] | 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," *Mult.-Valued Logic & Soft Computing,* vol. 17, no. 2-3, p. 255–287, 2010. |
| [59] | M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations,* vol. 11, no. 1, pp. 10-18, 2009. |
| [60] | J. Kennedy, "Small worlds and mega-minds: Effects of Neighborhood Topology on Particle Swarm Performance," in *In Proceedings of Congress on Evolutionary Computation*, 1931–1938. |
| [61] | R. Poli, J. Kennedy and T. Blackwell, "Particle Swarm Optimization : An Overview," *Swarm Intelligence,* pp. 33-57, 2007. |