Medical Record Classification: A Modified Genetic Algorithm for Feature Selection

Kamal Bakari Jillahi

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
September 2016
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Mustafa Tümer
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Ahmet Ünveren
Supervisor

Examining Committee

1. Asst. Prof. Dr. Adnan Acan

2. Asst. Prof. Dr. Yiltan Bitirim

3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

Medical record classification is the process of categorizing a patient's record as either having or not having a medical condition based on some given information (features) about the patient. Not all available features about a patient are both useful and relevant in the process of classification. As such, the need for selecting the relevant and useful features arises. Furthermore, the current growth in data dimensionality as a result of falling cost of data capture and storage also makes it necessary to feed the learning algorithm with only the required features about the patient. Over the years, the ML Community has used a number of algorithms for feature selection. One of such widely used algorithms is Genetic Algorithm (GA). Given that the performance of GA is depended on algorithm parameters and genetic operators used, this work modified the genetic operators (crossover and mutation) of the GA and used Extreme Learning Machine (ELM) which is a Single Layer Feedforward Neural Network (SLFN) with faster training time and least parameter tuning for the purpose of record classification. Furthermore, the work evaluated the performance of the proposed algorithm on 3 datasets from the UCI ML repository. The proposed algorithm showed a faster convergence, better classifier accuracy and fewer selected features than the traditional GA and other reported works. The proposed method is particularly useful in situation of time constraint, low computation power and high dimensional data.

Keywords: Feature Selection, Filter Methods, Wrapper Methods, Classification, Genetic Algorithms, Convergence, Extreme Learning Machine

ÖZ

Tıbbi kayıt sınıflandırma hasta hakkında bilinen tıbbi durum veya bazı verilen bilgilere (özellikler) dayalı olarak hastanın kaydını kategorize işlemidir. Sınıflandırma sürecinde hasta ile ilgili tüm bilgiler sınıflandırma için yararlı ve ilgili olmayabilir. Bu nedenle, yararlı ve ilgili bilgileri mevcut bilgiler arasından seçme ihtiyacı duymaktayız. Ayrıca, veri yakalama ve depolama maliyetini düşürme amaçlı hasta hakkında sadece gerekli özelliklere sahip olma ve bu özellikleri öğrenme algoritmalarında kullanmak için özellik seçimi önem kazanmaktadır. Yıllar geçtikçe, ML Topluluğu özellik seçimi için bir dizi algoritma kullanmıştır. Genetik Algoritma (GA) yaygın olarak kullanılan algoritmalardan biridir. GA algoritmasının performansı verilen papametreler ve genetic operatörlere bağlı olduğu gözönünde bulundurulduğundan bu çalışmada özellik seçimi için GA'nın genetic operatörleri (Çaprazlama ve Mutasyon) modifiye edilmiş ve kayıt sınıflandırma için hızlı öğrenme süresi ve az papametre kullanan Tek Katmanlı İleri Beslemeli Sinir Ağı (TKIBSA) ile Extreme Öğrenme Makinesi (EÖM) kulanılmıştır. Önerilen algoritma UCI ML deposunda bulunan 3 farklı dataset kullanılarak performansı test edildi. Önerilen algoritma geleneksel GA algoritmasından ve önerilen digger algoritmalardan daha hızlı yakınsama, daha iyi sınıflandırma doğruluğu ve daha az özellik kullanımı olduğu gösterildi. Önerilen yöntem, özellikle düşük hesaplama gücü ve yüksek boyutsal veriler durumunda yararlıdır.

Anahtar Kelimeler: Özellik Seçimi, Filtre Yöntemleri, Sarıcı Yöntemler, Sınıflandırma, Genetik Algoritmalar, Yakınsama, Aşırı Öğrenme Makinesi

To my Dad and Mum

ACKNOWLEDGMENT

Special appreciation and thanks to my supervisor Asst. Prof. Dr. Ahmet Ünveren for his guidance and relentless effort in ensuring the success of this work. His guidance at every step of this research has been a critical success factor to me and it is a great honor working with him.

I would like to use this opportunity to appreciate my parents for their prayer, moral and financial support, advice and care. I know I can't pay you back my plan is to show you that I appreciate.

My sincere appreciation to my wife and daughter: Maryam Ibrahim Jalo and Aishatu for their understanding and patience. I know my absence have cost you a lot. I appreciate every bit of sacrifice which directly or indirectly facilitated my success.

Thanks to all and sundry who have in one way or the other helped me towards this journey: to mention but a few all my brothers and sisters especially Aishatu Aliyu Ibrahim (Tani), friends, relatives and colleagues too numerous to mention your help is profound but my words are limited. Love you all and God bless you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

ADHD	Attention-Deficit Hyperactivity Disorder
AHP	Analytical Hierarchical Process
AI	Artificial Intelligence
ANN	Artificial Neural Network
BG	Bi-directional Generation
EA	Evolutionary Algorithm
EEG	Electro Encephalogram
ELM	Extreme Learning Machine
fMRI	Functional Magnetic Resonance Imaging
FPR	False Positive Rate
FS	Feature Selection
GA	Genetic Algorithm
GP	Genetic Programming
LBP	Local Binary Pattern
LDA	Linear Discriminant Analysis
LOO	Leave One Out
ML	Machine Learning
MRI	Magnetic Resonance Imaging
MRMR	Minimum Redundancy Maximum Relevance
MSE	Mean Squared Error
NP	Non Polynomial
OLS	Ordinary Least Squares
RBF	Radial Basis Function

RG	Random Generation
RGD	Regularized Gradient Descent
ROC	Receiver Operating Characteristic
RVM	Relevance Vector Machine
SA	Simulated Annealing
SBG	Sequential Backward Generation
SBS	Sequential Basic Search
SFG	Forward Feature Generation
SLFN	Single Layer Feedforward Network
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
TPR	True Positive Rate
UCI	University of California Irvine

Chapter 1

INTRODUCTION

1.1 Overview

Recently, the Machine Learning (ML) Community has seen a steady growth in both data dimensionality and sample size (see Figure 1.1 and Figure 1.2) part in due to the rise of fields like “the omics” [4], bioinformatics [6], natural language processing etc. and the falling cost of data capture and storage. This growth and the subsequent enormity pose a great scalability and performance issues to most of the prevalent learning algorithms. Concretely, highly dimensional data often contains a high degree of redundant (duplicate) and irrelevant (un-useful) attributes that remarkably degrade the efficiency of the learning algorithm used in the ML process. Therefore, attribute or feature subset selection proves to be an indispensable technique used in removing these irrelevant and redundant attributes in the ML pipeline. Especially when faced with a highly dimensional data.

Another appalling reason for feature subset selection is, since the main aim of the ML algorithms is to understand the underlining relationship that associate the feature (attribute) space and the class (target) space then, it is justifiable and rational to omit all those input attributes with less or no influence on the relationship between the attributes and the target in order to obtain a small and comprehendible predicting model. For example, [12] proposed many criteria for learning algorithm selection which consider the trade-offs between accuracy and size of the predicting model.

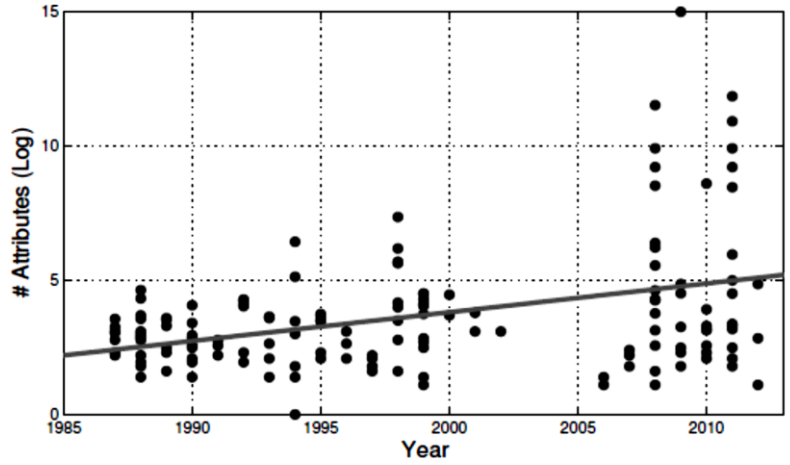


Figure 1.1: Growth in Number of attributes per dataset in UCI ML repository [18]

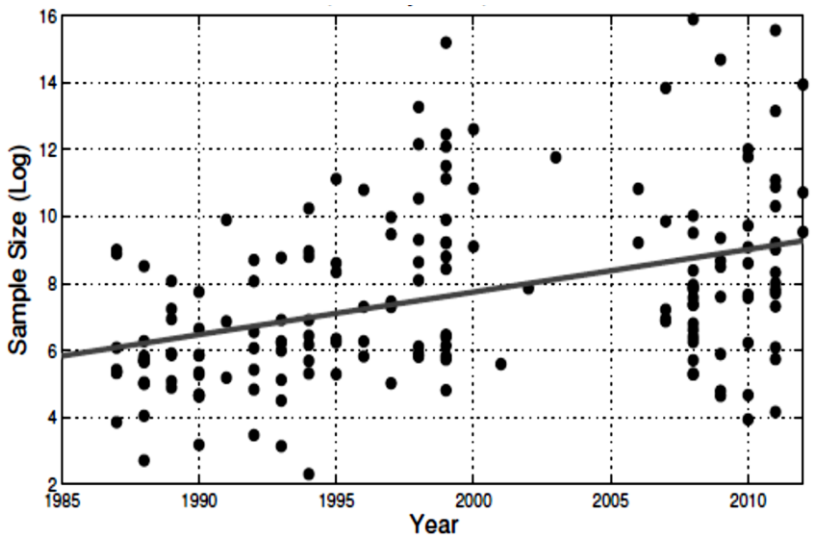


Figure 1.2: Growth in sample size of datasets in UCI ML repository [18]

Furthermore, data are collected for a vast array of uses other than ML purposes. For instance, data which is collected for accounting, audit or as legal requirement may end up been used for ML purpose. On the other hand, data collected for one ML algorithm might end up been used for a different algorithm. Thus, many irrelevant and redundant attributes in respect to the second algorithm might be obtainable in the dataset. More so, these irrelevant and redundant features will be unidentifiable by mere looking at the dataset even though the attributes are generated for our target

algorithm. Therefore, using intelligent procedures to extract those attributes which are important and useful to the learning algorithm is paramount.

In the same vein, when computational experiment is performed, we collect data about investigated entity. Often times, many other candidate attributes are incorporated even though some of these attributes are remotely associated to the entity been investigated; as a result, some of these incorporated attributes are inevitably irrelevant and or redundant. Therefore, to extract those relevant and useful attributes from these kinds of dataset, proven operations such as feature subset selection algorithm are required due to their objectivity and seeming accuracy.

In reality, there are two main problems which may be caused by irrelevant and redundant features in a dataset.

1. The irrelevant and redundant features induce more computational cost to the ML pipeline. For example, using a weighted linear regression [22] the computational expense is $O(m^2+n^2\ LogN)$ [22] for a single prediction where m is the number of attributes in the given dataset, n is the number of attributes to be selected with more features, the computational cost for predictions will increase polynomially. This is particularly true where there are a high number of such predictions; hence the computational cost will be immense.
2. The irrelevant and redundant features may cause overfitting. For example in a medical record classification problem where the Patient's ID is included as one of the attributes, an over tuned learning algorithm might conclude that the condition is fully or partially dependent on the patient's ID

1.2 Problem Statement

Given a dataset with *m* attributes, the task of feature subset selection is to find a set of *n* distinct features from *m* which provide the most accurate mapping of the input patterns (variables) to the target output (Class). This can be expressed mathematically as follows:

$$^{n}C_{m} = \frac{^{n}P_{m}}{m!} \tag{1.1}$$

Where the different permutations of *n* features selected from *m* is denoted by *P*, *m!* is the factorial of *m* that is $m \times (m-1) \times (m-2) \ldots \times 1$. This can also be expressed as

$$^{n}C_{m} = \frac{n!}{(n-m)!m!} \tag{1.2}$$

Subsequently, to obtain all possible combination of *n* features from *m* taking *n=1* to *n=m* at a time can be expressed as

$$\sum_{m=1}^{n} {}^{n}C_{m} = \sum_{m=1}^{n} \frac{n!}{(n-m)!m!} \tag{1.3}$$

Generally, the explicit combination of *n* features is 2^n i.e.

$$^{n}C_{0} + {}^{n}C_{1} + {}^{n}C_{2} + \cdots {}^{n}C_{n} = 2^{n} \tag{1.4}$$

Consequently, if a dataset has a total of 10 attributes then there are 2^{10} or *1024* possible combinations. For a larger dataset which contains 100 attributes, then that will be 2^{100} or 1.2677×10^{30} different subset combination. Apparently, this is time taking and computationally expensive even in a situation where the dataset is relatively small. This makes the used of heuristic or guided search inevitable as it avoids most of the less promising search space.

1.3 Motivation

Genetic Algorithm provides the required mechanism for a random search that compromises optimality for speed and less computational power requirement. This thereby increases the efficiency of the learning algorithm and subsequently improves classifier accuracy, model comprehensibility, faster convergence etc. Other advantages of using this approach include:

- Since GA is not exhaustive search this will lead to lower time requirement for FS and subsequently for the whole ML pipeline.
- Less computational requirement for FS since the whole dataset is encoded in a string of 0's and 1's
- Least requirement for parameter tuning as GA have few parameters.
- Easily comprehensible feature relevance metrics as a feature can either be selected or not selected as opposed to other methods which produce feature relevance metrics which are difficult to interpret e.g. regularized GD with a threshold of 0.5 which assigns a weight of 0.467 to an attributes is difficult to conclude to either select such a feature or discard it.

1.4 Thesis Objective

Unfortunately, even GA when faced with the task of FS for a highly dimensional dataset performs slowly. This is especially true where time (in online or real-time FS) is of great importance and computational power is limited. Therefore, enhancing the performance of GA in this situation is inevitable. To this end, this work modified the genetic operators (i.e. Crossover and Mutation) of the traditional binary GA in order to improve the classifier accuracy and convergence time of the learning algorithm. Furthermore, Extreme Learning Machine (ELM) which is a special Artificial Neural Network with a single hidden layer that requires low training time

and little parameter tuning was used to assess the efficiency of the proposed method using three (Pima Indians, Cleveland, Arrhythmia) datasets obtainable at the UCI ML Repository.

1.5 Structure of Thesis

This work is structured as follows: in Chapter 2 a comprehensive review of literature on FS is presented. Then, in chapter 3 the proposed genetic operators (crossover and mutation), datasets and methods used for testing and evaluating the proposed procedure are presented. Chapter 4 gives the experimental results. Finally, Chapter 5 offer conclusion and recommendations.

Chapter 2

LITERATURE REVIEW

2.1 Records Classification

Record classification is the task of categorizing a record into a class of known classes based on some known training dataset [6]. Most computational problems can be represented as record classification task. For example, medical record diagnosis can be modeled as the task of classifying a patient's record as having a condition or not, the task of email filtering can be modeled as classifying an email as "spam" or "non-spam", the task of News cataloging can be modeled as categorizing news items as one of many categories (e.g. "Politics", "Sports", Business", "Entertainment" etc.). This makes record classification a pivotal field in computational application. A generalized process flow of a record classification is shown below:

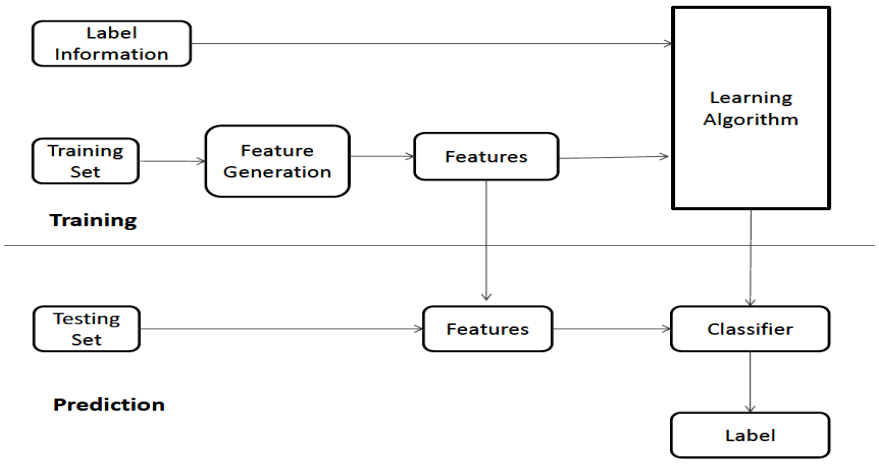


Figure 2.1: General outline of record classification [14]

From the above diagram, record classification process can be broken down into two major phases:

The Training Phase

In this phase, the dataset (training) is classified into classes based on the values of the attributes and classes conditional probabilities using either statistical, heuristics or other learning and induction algorithms. The attribute values and class tags can be categorical e.g. genotype information ("AA","AS","SS"), discrete age in years (e.g. 1, 2, 3), ordinal e.g. order of cardiac disorder ("First", "Second", "Third"), real e.g. heart beat measurement (103.22, 99.32, 100.23). Having the fact that some learning algorithms and classifiers require particular form of data (e.g. discrete, real, nominal etc.), all attributes and class values which do not conform to an algorithm's requirement need to be converted to enable the training phase learn a mapping from the attribute space to the class space as follows:

$$f\,(attributes) \rightarrow class$$

The major learning algorithms and classifiers can be categorized into:

- **Statistical Learning-** this set of algorithms use class conditional probability and training dataset distribution to learn a classification of the dataset based on likelihood of membership. These include algorithms such as Bayesian Models, Linear Regression Model, Logistic Regression Model etc.

- **Decision Trees-** these algorithms partition data into categories based on the closeness or similarity measure thereby assign membership to class based on closeness of data instance to the nearest class collective attribute such as class average. Examples of these algorithms include random tree, k-nearest neighbor, k-means etc.

- **Neural Network-** these are nature (based on biological nervous system) inspired models which are used to approximate or estimate a mapping between a large dimensional set of inputs to the target space (usually singular or multiple) classes. In essence, these are biologically inspire transformation models from one domain (attribute domain) to another (target domain).

- **Kernel Based Classifiers-** these classifiers construct hyperplane(s) on the training dataset by maximizing the class margin separability of the classes in the dataset. The best separability is attained by the hyperplane(s) with the maximum distance to the closest data item of any class. Examples include Support Vector Machines (SVM), Relevance Vector Machine (RVM) etc.

- **Ensemble Classifiers-** these are a combination of heuristic and other search algorithm which are used for the task of classification. Examples include gradient descent, recommender systems etc.

The Prediction Phase – In this phase, the mapping function or transformation learned in the training phase is used to categorize new data instances into the established classes based on the attribute values of the new data instance. Here, the features and class distribution of both the training and new data instances must be the same. This is because, both the prediction or transformation model are built on these premise.

2.2 Feature Selection

This is the most traditionally used approach for data dimensionality reduction among ML experts. The goal is to choose a subset of the original attribute set based on some metric that measures the relevance of the individual attributes selected or the quality

of the subset selected. This usually leads to better learning algorithm performance e.g. faster convergence, better accuracy, simpler model interpretability and cheaper computational cost. FS algorithms can be categorized depending on the training dataset been labeled or not. This is shown in the figure below:

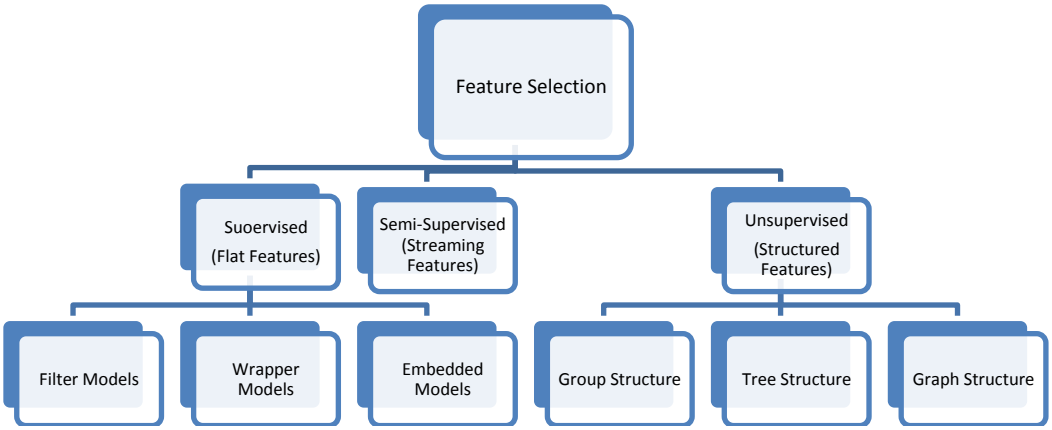


Figure 2.2: Feature Selection for Data Classification [18]

From the figure above, FS can be broadly categorized into three; Supervised (labeled training dataset), Unsupervised (un-labeled training dataset) and Semi-Supervised (uses both labeled and un-labeled training dataset). The supervised FS can be further sub-categorized into: filter which performs FS solely based on statistical and general properties of the dataset, wrapper which uses performance measurement of a predetermined classifier or learning algorithm to select features and embedded models which use in-built techniques for FS.

Unsupervised method on the other hand is an approach to FS where the training dataset is not labeled. These methods depend on intrinsic properties of the training dataset such as clustering quality [20]. Thus, many equally valid categorizations can be generated and subsequently the feature subset generated from this categorizations. Having a highly dimensional dataset, it is very impossible to get more useful subsets

without considering more constraints from an optimization point of view. Furthermore, objectively assessing the quality of generated subsets is another difficulty in unsupervised methods as opposed to supervised methods. This is because supervised FS has a basis of measurement (i.e. the class label) while unsupervised methods operate on un-labeled data thereby making performance assessment difficult.

In a situation where the data cannot be completely labeled, a sample thereof can be labeled and FS algorithm uses statistics from this sample to generalize on the population this is known as semi-supervised FS. Here, it should be noted that a sufficient sample size most be obtained to permit generalization and validate the performance of the FS algorithm or a Monte Carlo approach should be considered. Furthermore, the sample is supposed to be drawn randomly to preserve the population distribution. Typically, FS involve four major steps [5]; subset generation, subset evaluation, stopping criteria and validation. A number of candidate feature subsets are produced according to some search scheme in the subset generation step. Then, the generated subsets are evaluated based on some evaluation criteria in the evaluation stage. The subset with the best evaluation metrics is chosen after meeting the stopping criteria. Finally, the selected subset is validated using any validation mechanism or domain knowledge.

2.2.1 Feature selection for classification

Supervised FS is the widely applied technique to most of the real-world classification problems [25] because; the implicit class proportions and instance conditional probability of the data instances are known or can be modeled. Therefore, each instance can be categorized to a class. Unfortunately, we have little or no knowledge

about the features that produce the best learning model. For that, we initially include as much features as possible in the original dataset. These features may eventually be irrelevant or redundant to the target concept. Furthermore, it is practically impossible to extract the most reliable predictors (good features) before learning a model. Therefore, it is better to perform the FS before or while learning the induction model as this ensures the best features for the algorithm at hand are selected. Where the dataset has a very high dimension, then it may be a good idea to use all possible techniques to drastically reduce the feature set before applying supervised FS which works better on moderate to small datasets [25].

FS for classification aims to select the least sized subset at the same time meeting the following constrains:

- The accuracy of the classifier or learning algorithm does not diminish.
- The distribution of the resulting feature set is as similar to the class distribution of the original dataset as possible.

Because the search algorithms in FS explore through a very big space (2^m where m the number of features in the dataset), a stopping criteria is required for heuristic and random search algorithms in FS in order to prevent them from behaving as exhaustive search (i.e. run the search infinitely). Heuristic searches trade performance for accuracy while on the other hand exhaustive search trade computational complexity for optimality. Therefore, this calls for hybridization in other to strike a balance between efficiency and computational cost [17].

2.2.2 Approaches to Supervised Feature Selection

Supervised FS can be categorized into three main classes as shown in the figure below:

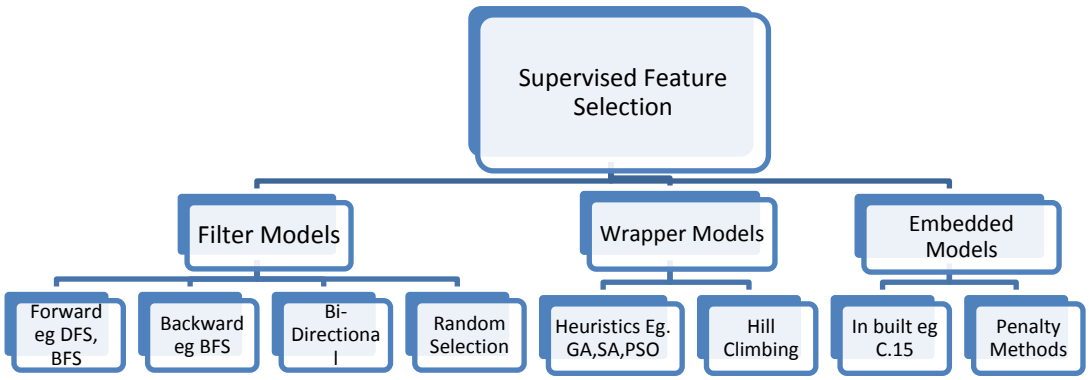


Figure 2.3: Figure 2.3: Approaches to Supervised Feature Selection [18]

2.2.2.1 Filter Models

The filter models use statistical and other intrinsic properties of the training dataset to extract the best feature subset without using any performance metrics of any induction algorithm to evaluate the goodness of the features generated or selected. This prevents interaction with any bias associated with the learning algorithm. Filter models rely on metrics like correlation, consistency, information, dependency or distance. Relief [30], Fisher Score [17], and information gain [22] are examples of filter based methods [18]. The major setback of these approaches is that the FS process does not consider the requirement and peculiarities of the learning algorithm to be used with the selected features. Filter models are preferred where the number of original attributes in the dataset is very large. The filter models have several advantages some of which are:

1. They do not consider learning algorithm's biases and peculiarities. That means features selected can be used with different classifiers or learning algorithms.
2. They generate subset faster than other methods because calculating data properties such as correlation, dependence, gain etc is usually cheaper than training and assessing the performance of a learning model.
3. In some situation (where classification cannot learned directly from original data), filter methods can be used to reduce the features before other FS algorithms are applied.

Below is a representation of filter model for FS

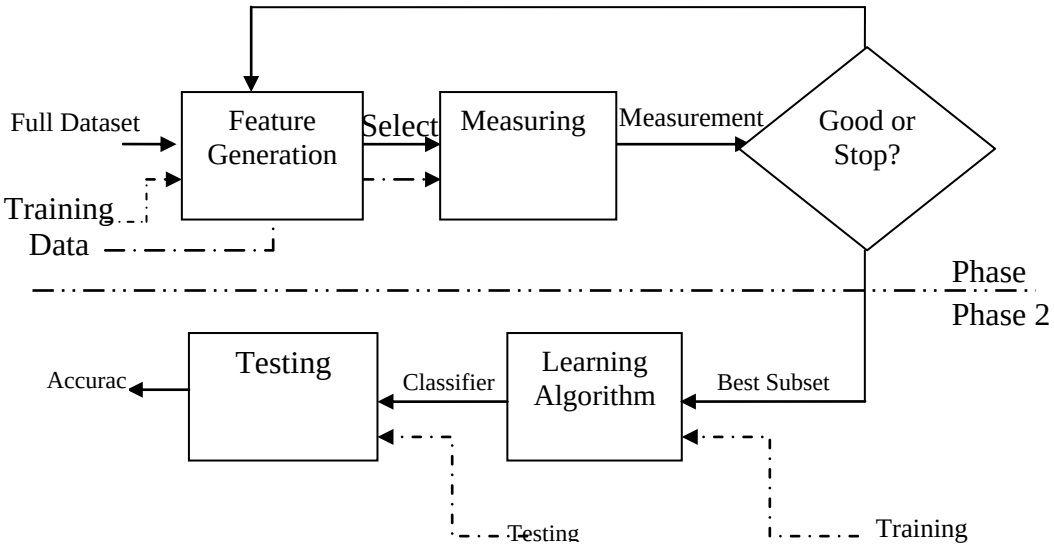


Figure 2.4: Filter Model Feature Selection [14]

Filter models can be broadly classified into four classes

- **Forward Selection-** these begin with an empty set of selected attributes. Attributes from the dataset are added to the selected feature list one after the other (sequentially) based on some measure of goodness. Usually, a feature with the best evaluation criteria is selected from the yet to be selected

features. The number of selected features increases until when the whole features from the original dataset are selected. Thereafter, the features are ranked based on how early they were added to the list of selected features. Form this list number of relevant features needed for the learning algorithm can be selected. The Sequential Feature Generation SFG is the most general form of Sequential Forward Generation. It starts with a subset of one feature then 2 features and so on. A generalized Pseudo code of SFG is given below:

```
Seq_Feat_Gen Scheme
Input: Features – Complete feat set, U – measure of
goodness
initialize: Subset = {} /* S selected features * /
repeat
feature = FindNext(Feature)
Subset = Subset u {feature}
Features = Features - {feature}
until Subset satisfies U or Features = {}
Output: Subset
```

Algorithm 2.1 Sequential Feature Generation [18]

- **Backward Elimination-** these begin with the complete attribute set of the original dataset as selected features. Thereafter, attributes are dropped one after the other based on some measurement metrics. The attribute with the least evaluation metric is dropped each time. Therefore, the list of selected attributes reduces until there is only one attribute at the end. Here, the relevance of an attribute is determined by how late it was removed from the original attribute set. Hence the most relevant attribute is the last dropped. Because it is easier to discover the most relevant attribute than the least relevant one and sometimes the other way round, SBG and SFG usually complement one another. A more generic form of SBG is the Backward Generation BG which begins with N (number of features in a dataset)

attributes subset then (*N-1*) and so on. A generalized pseudo code of SBG is given below:

```
Seq_Backward_Gen Scheme
Input: Features – complete feature set, U – measure of
goodness
initialize: Subset = {} /*keeps the dropped attributes *
/
repeat (1) feature = GetNext(Features)
       (2) Features = Features - {Dropped}
       (3) Subset = Subset u {Dropped}
Until: Features satisfy U or Features = {}
Output: Features U {Dropped}
```

Algorithm 2.2: Sequential Backward Generation [18]

- **Bi-directional Generation -** these start their subset generation from the two ends of the original dataset (i.e. two sequential searches are done in parallel; forward and backward). Both searches halt if either one search discovers the optimal attribute set or (based on supplied metrics) or both searches arrive at the middle of the dataset. Hence, we can say that BS leverages the advantage of both SFG and SBG. But it is worth noting that the attributes obtained by SFG and SBG may vary over a cause of experiments because their sequential selecting and dropping of attributes may not be deterministic. A generalized Pseudo code of BG is given below:

```
Feat_Bi_Gen Scheme
Input: Features_forward, Features_backward – full subset set, U
– measure of goodness
initialize:
Subset_forward = 0,   /* forward the added */
Subset_backward = 0          /* backward the dropped. */
repeat
(1) If = FindNext(Features_forward) I_backward =
GetNext(Features_backward)
(2) Subset_forward = Subset_forward U {features} Features_backward
= Features_backward - {f_dropped}
(3) Features_forward = Features_forward - {features Subset_backward
= Subset_backward U{l_backward}
until Subset_forward satisfy U or Features_forward = 0 or
Features_backward do not satisfy U or Features_backward = {}
Output: Subset_forward if (a) or Features_backward U {f_backward}
if (b)
```

Algorithm 2.3: Bi-directional Generation [18]

- **Random Generation –** these searches in random direction that is to say attributes are selected or dropped randomly based on some measurement metrics. The algorithm avoid been trapped in a local minima by changing their feature generation procedure. The size of the next generation subset cannot be determined unlike SFG or SBG. Although, the direction of feature generation (i.e. growing or shrinking) can be determined. A generalized pseudo code of RG is give below:

```
RAND_Gen Algorithm
Input: Features - full set, U – measure of goodness
initialize:
Subset = Subset_best = {}  /* Best subset set */
Card_best = #(Features)    /*# - cardinality of a set */
repeat
Subset = RandGen(Features)
Card = #(Subset)
if C ≤ Card_best ^ Subset satisfies U
Subset_best = Subset
Card_best = Card
print S_best
Until: stopping condition
Output: Subset_best  /*Best set Obtained* /
```

Algorithm 2.4: Random Generation [18]

2.2.2 Wrapper Models

These models base their decision of which attribute to select on the performance metric of a predetermined learning or induction algorithm and other factors such as the number of selected attributes and presence or absence of some required or disdained attributes. Hence for every generated subset, the wrapper models have to generate a learning model which makes them computationally prohibitive. But on the other hand, the wrapper models extract attributes which are more appropriate for the induction algorithm at hand. Hence, regardless of the induction algorithm the wrapper model is able to extract the best feature set. Given an induction algorithm or a classifier, the wrapper models proceed as thus:

1. *Step 1: Search the feature space and generate a subset*

2. *Step 2: Feed the selected subset to a learning or induction algorithm*
3. *Step 3: Measure the goodness of the generated subset based on the performance of the learning algorithm*
4. *Step 4: If desired quality is not achieved repeat Steps 1, 2 and 3 else Stop*

A diagrammatic representation of a wrapper FS is shown below:

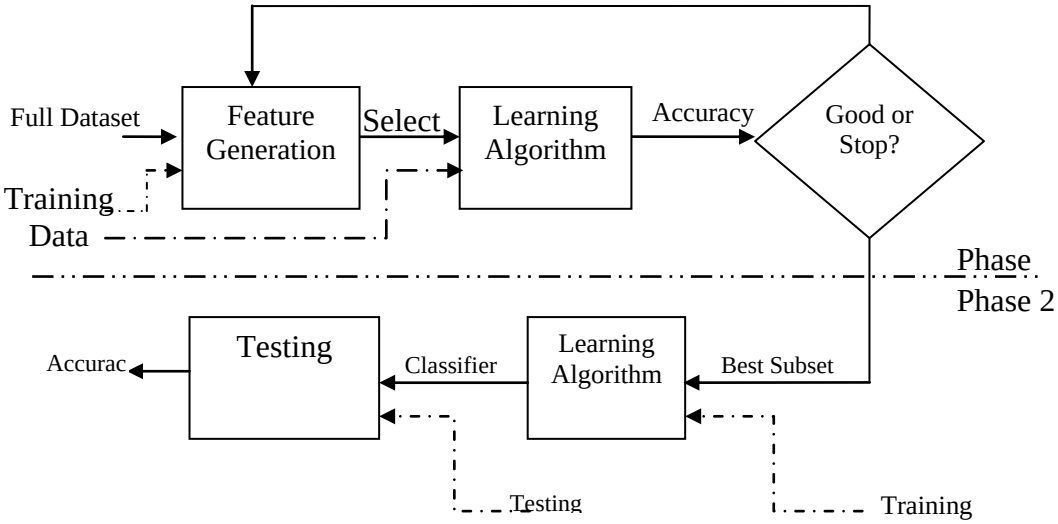


Figure 2.5: Wrapper Model Feature Selection [14]

Here, the feature generation module produces a subset of attributes; the evaluation module uses the classifier or learning algorithm's performance metrics (usually classifier accuracy) to measure the goodness of the generated attribute subset. This information will be fed back to the feature generation module for the next round which helps to enhance the quality of the generated attributes in the subsequent rounds. Finally, the subset with the best evaluation metric gets selected. The goodness of this subset is verified using an independent dataset. This is known as cross validation [5]. For a dataset with *m* attributes the computation time is $O(2^m)$. Therefore, applying an exhaustive approach may be impractical except in situation where *m* is relatively small. Many search approaches can be applied to overcome the

obstacles posed by exhaustive search. These include Best-First, Hill-climbing, Branch and Bound, Genetic Algorithm etc [8].

2.2.2.3 Embedded Models

Embedded models find which attributes are best predictors of the target class as the prediction model is learned. In essence, they perform feature FS as part of the learning procedure and are usually specific to given algorithms. Embedded models are broadly categorized into three:

- **Pruning Methods –** these methods use all data attributes to learn a model then try to remove some by making their coefficients 0 at the same time trying to maintain learned model performance where performance does not deteriorate these attributes are eliminated as irrelevant. An example is the recursive FS using SVM [31].

- **Built-in Mechanism –** these are embedded mechanism which use feature weight adjustment for FS and are specific to some learning algorithms.

- **Regularization Models**- these accomplish the task of FS with the use of a cost function which try to minimize the errors of model learning while penalizing the coefficients of the less relevant features. Finally, those features with 0 or close to 0 coefficients are removed as irrelevant. An example of this is Regularized Gradient Descend RGD.

2.3 Genetic Algorithm

This is an optimization algorithm that imitates the Darwin's process of natural selection. This algorithm (sometimes referred to as a meta-heuristic) is applied to solving combinatorial and other types of optimization problems where the objective

equations are complex to compute. GA form a part of the widely known Evolutionary Algorithm EA that use natural selection techniques such as crossover, mutation inheritance etc. to generate solutions to optimization and search problems.

2.3.1 Brief History of Genetic Algorithm

In 1950, Allan Turing suggested the idea of a "learning machine" to imitate the process of evolution of species [1]. In 1954, Nils Aall Barricelli started the simulation of Evolution in computing while working on the computer at the Institute of Advance Study in Princeton, New Jersey. In 1957, the Australian geneticist Alex Fraser reported a series of studies on "artificial selection of organism with multiple loci controlling measurable traits". The simulation of evolution process by biologist became rampant in 1960 based on these publications. Here, it is worth noting that most of the elements of the modern genetic algorithms were part of Fraser's initial simulation. 1960's Hans-Joachin Bremermman reported some studies wherein he suggested "a population of solutions which go through recombination and alteration to solve optimization problems". These works also had most of the properties of modern GA [8]. Some other pioneers on GA include John Holland, Richard Friedberg, George Friedman and Michael Conrad. Although the credit of simulating a simple evolutionary game is awarded to Barricelli, artificial evolution as a means of optimization method were broadly adopted due to the publications of Hans-paul Schwefel and Ingo Rechenberg in the 1960's and 1970's.

2.3.2 Genetic Algorithm Steps

In a GA, the process starts with a population of individuals (also known as solutions, chromosomes or phenotype) which strive to solve a search or optimization task. Then, these individuals develop through recombination and alteration to be better individuals (solutions). Every individual in the population is composed of traits

which are then recombined with those of others or altered to create better or fitter solutions. Normally, individuals are represented as a string of digits (i.e. 0's and 1's for binary encoding, 0-9 for permutations and real valued encodings) but other encodings are possible [8]. The general workings of GA can be represented as in the figure below:

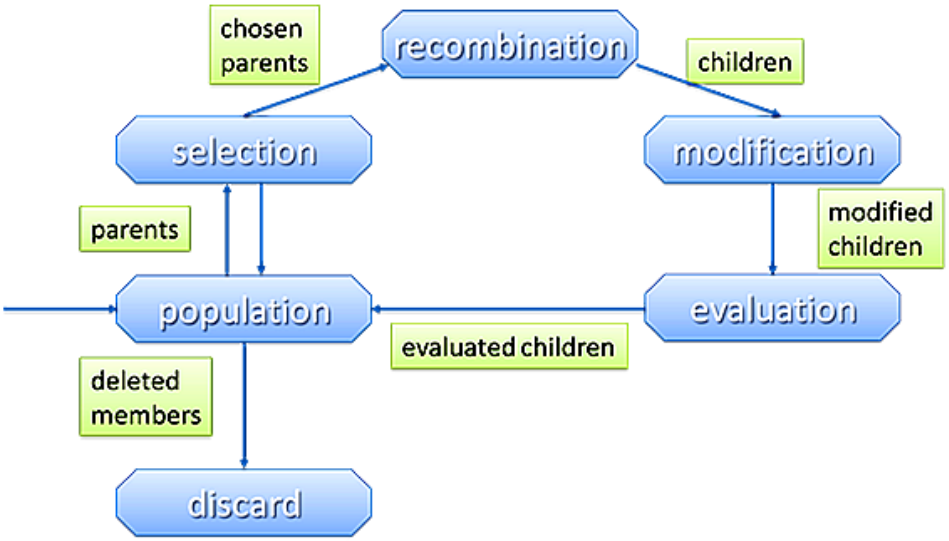


Figure 2.6: Genetic Algorithm Steps [16]

As shown above, the evolutionary process begins with a set of randomly created individuals (solutions), each set of individuals in an iteration are known as a generation. In every iteration, each solution is assessed as to how good it solves the problem at hand. This is known as fitness evaluation. Better solutions are chosen to proceed to the next generation and new solutions are created by recombining and altering other individuals which ensures new individuals and traits are introduced into the population at each iteration. Usually this stops when a maximum iteration is reached or an acceptable level of fitness is achieved or there is no improvement in the individuals over certain generations. Generally to function well a GA requires

1. A good encoding of individuals

2. A reliable measure of fitness to assess the quality of individuals.

In FS using GA, the generally accepted encoding of individuals is an array of bits (i.e. 0's and 1's) although other data structures and encodings are possible. The major reason behind this is there sizes are fixed therefore, they can be easily aligned. This facilitates simple crossover operation. Variable length encoding of solution is also applicable but this makes crossover more difficult and complex.

2.3.3 Outline of a Basic Genetic Algorithm

INPUT {pop_size =Population size, P_x= Probabilty of Crossover, P_m =Probability of mutation, Nbits = number of bits per individual, *f()* = fitness function }

1. [**Begin**] Create a population of *pop_size* randomly generated individuals with Nbits alleles
2. [**Fitness**] Measure the goodness of each individual in the population using *f()*
3. [**New population**] Generate new population by doing the following until the required number of individuals is obtained
    1. [**Selection**] choose two or more individuals from the population to serve as parents
    2. [**Crossover**] with a probability P_x apply crossover to parents to create new children
    3. [**Mutation**] alter the traits of a single individual to create a new individual with probability P_m
    4. [**Accepting**] accept or reject new individuals into population based on some measurement
4. [**Replace**] Use new population for next generation
5. [**Test**] test for termination condition
6. [**Loop]** go to 2

Algorithm 2.5: Outline of a Genetic Algorithm [1]

2.3.4 Components of a Genetic Algorithm

2.3.4.1 Encoding of a Chromosome

Each individual should encode information about how it solves the problem at hand. In FS, a string of binary is the simplest way of encoding solutions where a 1 means an attribute is chosen and a 0 means an attribute is not chosen e.g.

Chromosome One	1011001000100101
Chromosome Two	1100001000011100

Table 2.1: Example of Binary Chromosome Encoding

From the table above, each chromosome is a string of bits equal in size usually the total number of attributes in the dataset. As earlier mentioned, this makes other GA procedures easier.

2.3.4.2 Fitness Function

This is a summary of how price a particular individual solution arrives at the solution to the problem at hand. This is because each individual is simply a string of numbers therefore; a summarized expression of the goodness of each individual is needed to decide the surviving (thriving) and non-surviving (dying) individuals. Subsequently, after each round of testing, some individuals (worst) are deleted and replaced by better individuals. Thus, each individual requires to be assigned a measure of merit signifying how close it came to achieving the overall goal. One of the most endearing tasks in using GA for optimization is the design of fitness function because this requires the designer to figure out both workable and efficient design that measures every aspect of the problem been modeled. In essence, it requires more work from the human designer to come up with final design of the fitness function which is the main driver of the GA. This is because without an expression of how well a problem is solved, one does not know when to stop or continue or when to conclude if the problem is solvable or not. If the fitness function is bad, the GA will either arrive at a wrong solution or not solve the problem entirely. Moreover, the fitness function should not only model the aim of the optimization it should also be cheaply computable because the time of execution is equally valuable as the GA process must be repeated for a number of times in order to obtain a meaningful solution in a

nontrivial problem. There are instance where fitness approximation might how ever be appropriate. These include

- Where the time required to compute the fitness function of a single solution is extremely high.
- Where the accurate measurement of the fitness is not known
- Where the fitness is imprecise or un-deterministic.

2.3.4.3 Parent Selection

In GA, traits from individuals are put together in order to produce new and better individuals. Therefore, this raises the need for a technique of selecting those individuals to be used as parents for the purpose of children creation. The most popular parent selection procedures are as follows:

- **Roulette Wheel:** Each individual in the population is given an opportunity of been a parent equal to the value of its goodness. Then a number is drawn and the individual with fitness value less than the drawn value but greater than next individual in fitness is chosen as a parent. This process is iterated until we get the needed amount of parents. Consequently, individuals who have more fitness will dominate the selection process because they have more chances of been selected as they occupy more space on the roulette wheel.

- **Rank Based:** where there is high disparity between individual's fitness the roulette wheel method will be biased towards fitter individuals. As such, to avoid this, rank based method proves to be more applicable. Here, each individual is ranked accordingly to its fitness and each rank is given a place on the roulette wheel. Therefore, even weaker individuals are given an

opportunity of been selected as parents. The figure below gives a contrast between roulette wheel and rank based parent selection method

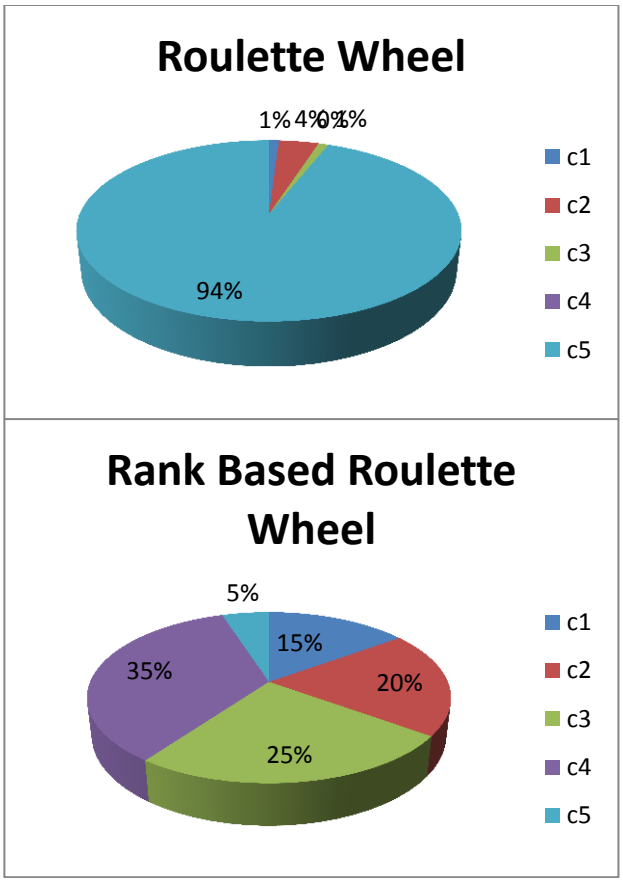


Figure 2.5: Comparison of Roulette and Rank Based Selection [8]

- **Tournament:** Here a tour size [8] is determined before the real selection process. The minimum tour size is two while the maximum is equal to the number of individuals in the population. Then, a random number is drawn over the population and a subset of the population equal to tour size is selected as mating pool. Then, the fittest individuals from the mating pool are selected as parents.

- **Genitor:** Here, individuals are chosen to be parents using linear regression. Thereafter the weaker parents are dropped and substituted by better children.

2.3.4.4 Genetic Operators

2.3.4.4.1 Crossover

Crossover selects genes from multiple parents in order to create new offspring. Alleles from the selected parents contain information that help the offspring solve the problem. Therefore, an offspring created from two good parents inherits some or all of their good traits and defects. For binary encoded GA, the four mostly used crossover procedures are:

- **One point crossover:** Here, a random point (known as crossover point) is determined then the selected parents are cut at these point and the new offspring are created by putting the various parts in a crossover arrangement [1]. The main problem with this scheme is there may be bias of positional arrangement due to consistency in sequential alleles. This is depicted in the figure below:

Figure 2.8: One Point Crossover [8]

- **N-point crossover:** here, two or more points are chosen as crossover points. Then, each parent is dissected into (*N+1*) points; the subsequent points are then arranged in an alternating arrangement from each of the parents [8]. This is depicted in the figure below:

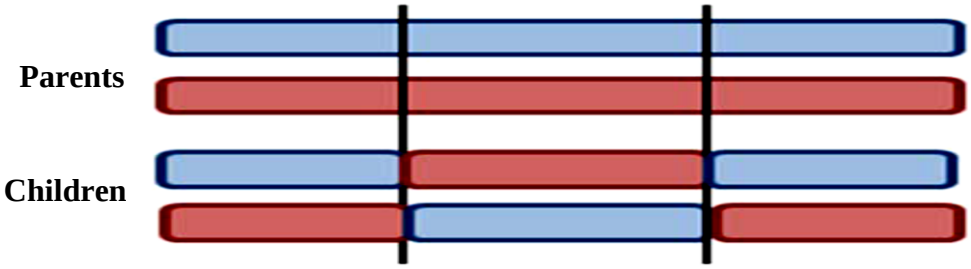


Figure 2.6: N-Point Crossover [8]

- **Uniform Crossover:** to tackle the bias inherent in both one point and N-point crossover, this scheme considers each allele independently. Each allele is alternated between the two parents to a child with a probability of 0.5. this is depicted as in the figure below:

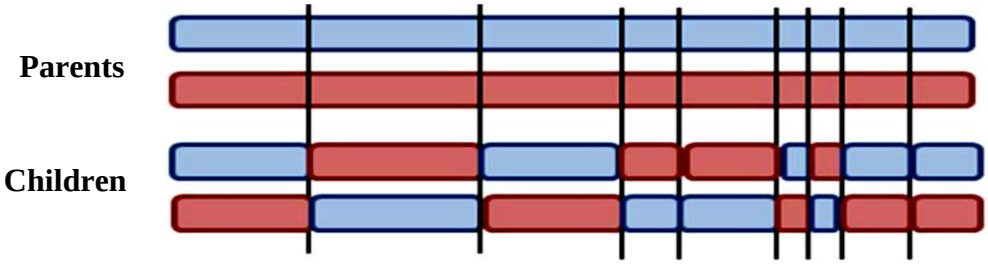


Figure 2.7: Uniform Cross Over [8]

- **Cut and Splice:** Here, the parents are cut at different points into different lengths and each child is formed by assembling different parts of the parent accordingly. This is depicted in the diagram below:

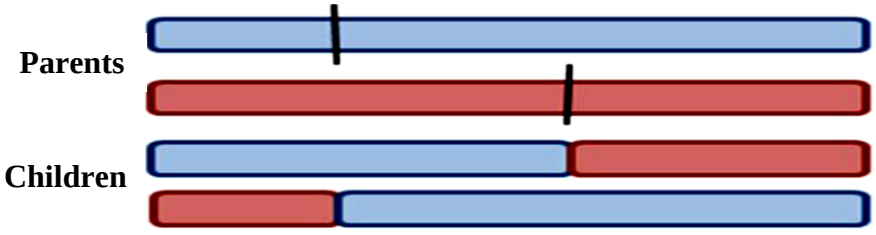


Figure 2.8: Cut and Splice Crossover [8]

2.3.4.4.2 Mutation

Because the crossover operation creates offspring from information inherited from multiple parents, offspring recycle traits contents in the population. Therefore, in mutation alleles in a single individual are altered this ensures new traits are introduced into the population pool. In binary encoded GA the four popularly used mutation operations are:

- **Bit Flip:** one allele (bit) at a particular point in the parent is flipped (altered from 0 to 1 or vice versa). This is depicted in the figure below:

Figure 2.9: Bit-Flip Mutation [14].

- **Insert Mutation:** Two alleles (bits) are randomly selected from a single individual and the second allele (bit) is moved next to the first one as shown in the figure below:

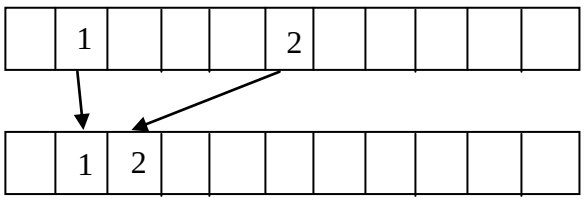


Figure 2.13: Insert Mutation [14].

- **Swap mutation:** Here, two random bits are selected then their respective positions are swapped. This is shown in the figure below:

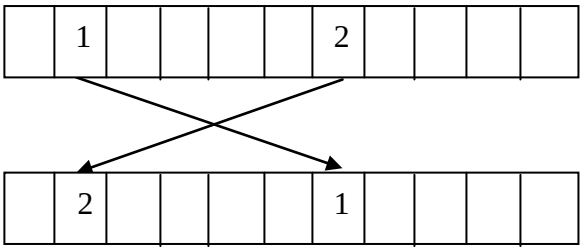


Figure 2.14: Swap Mutation [14].

- **Scramble Mutation:** *n* bits are chosen in an individual then each is randomly reassigned a new position in the resulting individual as shown in the figure below:

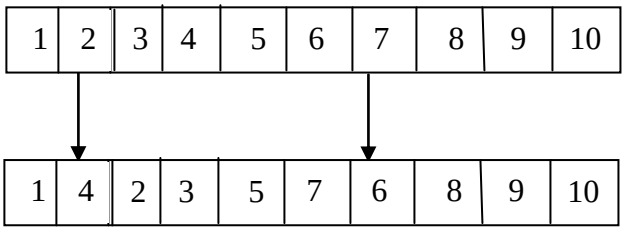


Figure 10: Scramble Mutation [14].

- **Inversion Mutation:** Here two points are chosen then the bits arrangement between those two points are reversed as shown in the figure below:

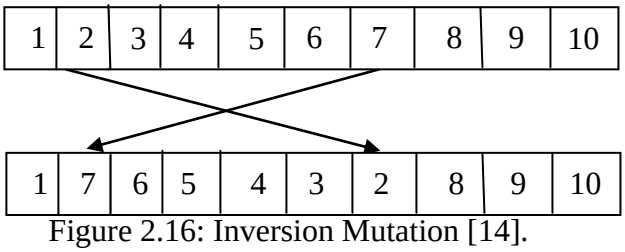


Figure 2.16: Inversion Mutation [14].

2.3.5 Related Works

[8] Studied an approach to improve AI and ML technique for generating classification rules for intricate real-world dataset. The study noted that standard rule inducing systems generate rules which are unacceptable due to two major reasons

- The need for minimal feature set coupled with cost of computing them.
- Computing time of the induction systems.

The study used Genetic Algorithm (GA) and AQ15 rule induction system wherein GENESIS [1] was used as the FS algorithm. The results show the potential use of FS techniques to improve rule induction systems. GA was shown to produce an impressive reduction in the amount of attributes needed for texture classification. The efficiency of the method proposed was compared with Sequential Basic Search (SBS) procedure. The study observed that the feature set extracted by the Relief [33] method were smaller than those obtained by heuristics algorithm (e.g. GA). However, GA showed a simultaneous improve in both number of discarded features and fitness accuracy as the number of iterations progressed while SBS only showed improvement in discarded features with little or no improvement in induction accuracy.

In [14] Shahamat et al used GA for FS of fMRI data used in the classification of patient records for schizophrenia. The selected features are classified using Euclidean distance based classifier and majority vote method using Leave One Out (LOO) cross validation procedure to assess the performance of the learned model. The study used the public available dataset NA-MIC to perform the classification. The study performed preprocessing including realignment, normalization, and smoothing before performing FS. After running GA algorithm the selected features were passed

to Linear Discriminant Analysis (LDA) to further extract features which maximizes the proportion of inter-class and intra-class variability. Finally, Local Binary Pattern (LBP) was used to classify the selected features as Schizophrenic or not. The result obtained was compared with the result obtained without applying GA for FS. The study noted that the result is comparable with other state of the art procedure.

Priyanka et al [33] investigated the performance of different classification methods before and after applying GA for FS. The study used the Ovarian Cancer dataset with a couple of classifiers. These classifiers include: Bayesnet [5], Sequential Minimal Optimization (SMO) [5], and simple logistic regression [5]. The performance of all algorithms improved dramatically after the introduction of GA for FS. Furthermore, the study noted that for any algorithm which is intended to be used with a large dataset, it has to be reduced reasonably to a subset which the learning algorithm can handle. Furthermore, GA been a stochastic random search provides the desired leverage for searching through the feature space. However, when a high rate of mutation is applied to the algorithm it tends to behave like other exhaustive search procedures. Finally, the study pointed out that GA as a tool for FS is indispensable in a situation where the relationship between features cannot be mathematically expressed or measured.

Bir et al [8] proposed a fitness function for GA algorithm to be used for FS. The proposed function in addition to classification accuracy penalizes any individual solution with higher number of selected features than individuals with less number of selected features. The proposed function is expressed as follows

$$fitness = \alpha.accuracy + \beta.num\_features \quad (2.1)$$

Where

accuracy = classifier accuracy

α = variable that reflects the influence of classifier accuracy

num_features = number of selected feature by an individual solution which is given by

β = variable that reflects the influence of number of selected feature

$$num\_featues \quad = num\_features\left(log\left(1+\frac{AES}{ES}\right)\right)$$

Where *AES* is the ensemble size and *ES* is the number of base classifiers.

Here the GA does not only consider the classifier accuracy but also the number of selected attributes in each individual. The proposed function was used together with Naïve Bayes, Nearest Neighbor, SMO classification algorithms with the following datasets; UCI hepatitis, UCI breast cancer, auto mpg where the result of GA FS which uses only accuracy as fitness was compared with the proposed fitness function. The result showed an increase of a factor of two in convergence speed. However, the study suggested the investigation of premature convergence of the proposed method in future studies.

Mitra et al [29] investigated ensemble FS approaches in comparison with heuristic approaches. The study tried to understand the factors that influence choice of a classifier in order to perform the task of FS. The study arrived at the conclusion that for ensemble methods; accuracy, information gain and voting methods are the determining factors of the success of FS algorithms while for heuristic; accuracy, fitness and diversity of individuals are more crucial in the performance of the algorithms. The study integrated accuracy, information gain, weighted and simple voting mechanism into ensemble algorithms (Naïve Bayes, Random Subspace, FSS and hill climbing) and compared their performance with the traditional algorithms.

While on the other hand; accuracy, fitness and diversity were integrated into heuristic algorithm (GA and SA) and the performance was compared with the traditional algorithm. Furthermore, the earlier integrated mechanisms were reversed for both ensemble and heuristic. The result of the reversal showed little or no gain in performance.

In [19] Riccardo investigated the use of GA for FS in spectral data. This study noted the peculiarity of FS in spectral data as features are spread out throughout the spectrum. Therefore, exhaustive searches find it difficult to find a subset in reasonable time. The study however noted that GA after suitable modification produce more interpretable result in shorter time since the wavelength are more dispersed. Furthermore, the study assumed that there is autocorrelation among variable in spectral-data. This makes the performance of a guided search easier to converge. On the other hand, the study noted the risk of applying GA as overfitting and this risk adds as the number of evaluated models increase because the chances of getting a model with good performance (due to random correlation) gets bigger. Finally, the study noted that the proposed GA modification did not consider autocorrelation between adjacent wavelength and variables that have never been used previously.

2.4 Extreme Learning Machine

These are Artificial Neural Networks composed of a single hidden node which are used for both classification and regression. ELM was proposed by Guang-Bin Huang [9]. ELM is different from other neural networks because the weight connecting the input nodes and the hidden layer are set once and never updated. This results in a faster learning process as opposed to the predominant back propagation algorithms.

Hence, ELM model produce superior generalization on fresh data and more comprehendible models than most other Neural Networks models using back propagation method for training. This is the reason why the model has attracted both academic research and practical adoption in recent years. Areas where ELM have recently been used include OP-ELM for evolving fuzzy systems [7], ELM for time series prediction [3, 9], regression with missing data [7], finding mislabeled samples using ELM [13], FS using ELM [9] classification for nominal data [9] etc. on the other hand, current areas of research on ELM include optimally pruned adaption of ELM [13], using GPM to accelerate ELM [7] etc. Training ELM is fast because the optimal output β is derived using mathematical procedure such as Ordinary Least Squares OLS and other regularized alternatives. A model of ELM training can be expressed as:

$$\hat{Y} = W_2\sigma(W_1.X) \tag{2.1}$$

Where σ is some activation function (usually sigmoid, radial basis, Gaussian, logistic or any order binary or bi-polar function in the case of classification and any linear function in the case of regression). W_1 is the vector of weights connecting the input and the hidden layer and W_2 is the vector of weights connecting hidden layer and the output layer. The sequential steps of the algorithm are as follows

1. W_2 is padded by some Gaussian noise
2. W_2 is estimated using least squares to fit a response vector.
3. Y is calculated by the pseudo inverse $^+$ having a design vector X:

$$W_2 = \sigma(W_1.X) + Y \tag{2.2}$$

2.4.1 Controversy

The purported invention of ELM by Guang-Bin Huang in 2008 provoked some debate where some researchers called the attention of the editor of IEEE transactions on neural network saying "the idea of using a connected hidden layer to the inputs by random untrained weights was already suggested in the original work on RBF networks in the late 1980's and experiments with multilayer perceptrons with similar randomness had appeared in about the same time frame". Subsequently, Guang-Bin replied in a paper in 2015 complaining about "a very negative and unhelpful comments on ELM in neither academic nor professional manner due to various reasons and intensions" [9]. Arguing that his work "provides a unifying learning platform" for various types of Neural Networks.

A diagrammatic representation of an ELM is shown below

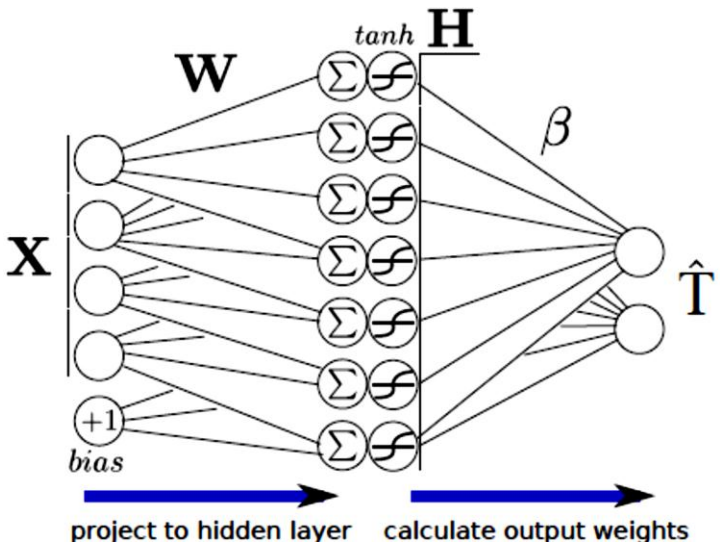


Figure 2.17: Representation of ELM with multiple outputs [16].

In the figure above, the model represents a multi-class categorization problem. A dataset of N observations with samples $\{x_i,y_i\}$ is assumed. Where $x_i \in R^d$ and $y_i \in \{1, 2,\ldots, c\}$ and c is the total number of different classes. A binary variable is used to

encode the target T. Here, T is the vector of class tags where T_{ij}=1 if and only if y_i = j (that is the instance 1 is a member of class j) Else T_{ij}=0. In the case of a bi-class classification problem, a single output variable is enough since membership to a class can be expressed using a threshold. An SLFN with d input node and M hidden nodes can therefore be represented as

$$f(x) = \sum_{k=1}^{M} \beta_k h(w_k . x_k) \quad (2.3)$$

Where β_k are the weights of the output layer, *h(.)* is a non-linear activation function, w_k is weights of the hidden layer, *x* is the input vector to the model, *f(.)* is a c-dimensional vector which represent the output of the model. Membership to a class is assigned based on the biggest element of the output vector. From a linear algebraic point of view, the problem is that of calculating the least square of the following matrix equation

$$H\beta = T \text{ where } H = h(w_k . x_i) \quad (2.4)$$

The model bias are represented by concatenating a 1 to each x_i or appending a column of 1's to the matrix H. for N different observations of $\{x_i, y_i\}$ where $x_i = [x_{i1}, x_{i2}, \ldots x_{in}]^T \in R^n$ and $t = [t_{i1}, t_{i2}, \ldots t_{in}]^T \in R^m$, a single layer feedforward neural network with Ñ hidden nodes and *g(x)* activation function can be written as

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g(w_i . x_j + b_i) = o_j \quad (2.5)$$

Where $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots \beta_{in}]$ is the vector of weights between the i^{th} hidden node and the output nodes $w_i = [w_{i1}, w_{i2}, \ldots w_{in}]^T$ is the vector of weights between the i^{th} hidden node to the output nodes. b_i is the threshold of the i^{th} hidden node. Hence, w_i, x_j represent the inner product of w_i and x_j. Subsequently, this single layer feedforward neural network with N hidden nodes and *g(x)* activation function can deduce N

samples with 0 error mean that is $\sum_{j=1}^{N}\|0 - t_j\| = 0$. This means there is a b_i,w_i such that

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i . w_j + b_i) = t_j, j = 1, .., N \quad (2.5)$$

The system above can be summarized as

$$H\beta = T \quad (2.6)$$

Where

$$H(w_i \ldots w_{\tilde{N}}, b_i \ldots b_{\tilde{N}}, x_i \ldots x_{\tilde{N}}) = \begin{bmatrix} h(w_1 . x_i + b_1) \ldots . h(w_1 . x_i + b_1) \\ . \\ . \\ . \\ h(w_1 . x_{\tilde{N}} + b_1) \ldots . h(w_{\tilde{N}} . x_{\tilde{N}} + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ . \\ . \\ . \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times M} \quad and\ T = \begin{bmatrix} t_1^T \\ . \\ . \\ . \\ t_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times M} \quad (2.7)$$

H is the matrix output of the hidden layer; the i^{th} column of H represent the hidden output associated to x_1,$x_{2,}\ldots x_N$. If the activation function g in the system above is infinitely differentiable then, we can show that the required hidden nodes of the model is $\tilde{N} \leq N$

2.4.2 Related Work

The applications of computational methods in medicine have shown a tremendous adoption of Artificial Intelligence (AI) and specifically Machine Learning (ML) approaches in the diagnosis of patient medical conditions. One remarkable area which has shown success is the application of soft computing methods such as ANN, pattern recognition, fuzzy principles, signal processing and image processing in the classification, clustering and regression of patient records. In this vein, a generation of expert systems which achieve a remarkable accuracy and efficiency has been reported by researchers in many different areas of medicine ranging from genomics,

personalized medicine, protein to protein interaction, disease-drug relation etc. Consequently, ELM been a specialized ANN have widely been used applied in the diagnosis and classification of medical records; results obtained from these studies have so far shown an incredible accuracy and speed of ELM in medical record classification.

In [13] Electroencephalogram (EEG) was used to detect the presence of epilepsy seizures in participants. The study used sample entropy as a means of FS for performing the task of classification of EEG signals which are normal (ictal) or abnormal (intrical). Here, the value of the sample entropy plays the role of sample ceiling in the procedure. It was observed that the value of sample entropy falls suddenly in data with presence of epilepsy which delineates the occurrences or absence of epilepsy. The study used the Analytical Hierarchical Process (AHP) method to select the input weights and hidden biases for the ELM. The study observed that using sample entropy and hybridized ELM a better accuracy and speed was achieved.

Similarly, ELM was used in [2] for the identification of Erythematic Squamous Skin disease. Here, the researchers used ELM to classify a patient record into one of seven classes (psoriasis, seborrheic plegmatis, lichen planus, pityriasis rosea, chronic dermatitis and absence of the disease). Due to the close clinical features of the diseases, other classification algorithms perform poorly on this problem. In contrast, ELM performed astoundingly better with an accuracy of 84.74% as opposed to other methods such as the classical ANN which reported an accuracy of 77.26% on average based on the UCI Erythematic dataset. However, the study noted that ELM performs slightly better when the percentage of training sets decreases with an

increase in the testing sets which is a rare ability. More so, the study discovered that ELM was able to maintain a consistency in the face of missing entries in both the training and test sets.

Karpagavalli et al [9] used Electrocardiography (ECG) signal to detect cardiac disease using ELM classifier. The study compared the performance of ELM and Relevance Vector Machine (RVM) on MIT-BIH dataset. The result showed the superiority of the RVM on unprocessed dataset and vice versa on a processed dataset i.e. the ELM out performed RVM on a selected feature set in both accuracy and speed while the RVM performed better on the raw data. Furthermore, both approaches were compared with traditional classifiers such as ANN where the results indicate the superiority of the two approaches. However, the study noted the advantages of ELM over RVM as requiring less or no parameter tuning, learning speed and more comprehendible model due to the absence of hidden transformation. Finally, the research suggested the use of ELM in a situation where (1) data preprocessing can be performed, (2) where speed is of higher importance and learned model need to be understandable. While RVM is better applied in a situation where; preprocessing cannot be performed, speed is of low or no importance in the learning process and model comprehensibility is of low or no importance.

Muthanantha et al [7] proposed Optimized Extreme Learning Machine (OELM) for the classification of Encephalogram (EEG) with emphasis on epileptic seizure detection. The proposed method does not require selection of hidden neurons, adjustment of hidden weight and biases. The performance of the proposed classifier was compared against traditional classifiers such as Linear Regression, ANN, SVM and traditional ELM. The study showed that the speed and accuracy of the traditional

ELM was enhanced by ranking the neurons using Minimum Redundancy Maximum Relevance (MRMR) algorithm and selection of the most relevant neurons thereby reducing the size of the ELM and the computational requirement of the model in general. This reduction there by boost speed and accuracy of the learned model. The study observed that ELM needs more hidden nodes than Backpropagation methods but much less than SVM. Furthermore, ELM models tend to have problems when irrelevant (uncorrelated) and redundant variables are present in the training set. For this it is advised to perform pruning of the irrelevant and redundant variables using information gain approaches such as MRMR before applying ELM consequently their proposed method embed this algorithm making it a candidate in high performing ELM flavors.

[21] Used a distributed ELM with Mapreduce framework which can cover the shortage of the traditional ELM whose learning ability in huge dataset is weak. The study found out that most expensive computational part of the ELM is the Matrix Moore-Penrose generalized inverse operator in the output weight vector calculation. As the Matrix multiplication operator is decomposable, a distributed ELM based on the Mapreduce framework can calculate the matrix multiplication efficiently and accurately in parallel and then the corresponding output weight vector is calculated centrally. Consequently, the study outlined the importance of their proposed method in a situation where the data volume to be analyzed exceeds the computing capacity of a single machine. Due to generalization performance, rapid training speed and little parameter tuning, ELM has demonstrated it is indispensable in the area of online or real time classification of huge dataset. However, the study noted that the proposed method can only be applied in a situation where the matrix multiplication is decomposable and the corresponding output weight vector can be centrally

computed. In conclusion, the study proved the theoretically that most expensive computational part of the ELM can be distributed to any parallel computing pipeline to improve performance and speed. However, where the output weight cannot be summed or averaged over the entire distributed computing pipeline, this will pose a problem to the proposed Algorithm.

Xiaolong [3] et al used ELM for a study on the diagnosis Attention-Deficit/Hyperactivity Disorder (ADHD). They studied the effects of data volume on the performance of both SVM and ELM on the classification of which parts of the brain are associated with ADHD. The study employed a high resolution Magnetic Resonance Imaging (MRI) images from patient with and without ADHD. Multiple brain attributes e.g. cortical thickness was measured. The LOO cross-validation procedure was used to verifiy the performance of both classifiers. The result showed that ELM achieved 90.18% accuracy compared to SVM which achieved 86.55%. The study found a profound difference between patients with and without ADHD in the frontal, temporal and occipital lobes and in the insular parts of the brain. The research noted the speed and accuracy of the ELM algorithm and minimal interference requirement. In conclusion, the research proposed the use ELM in real-life examination of ADHD.

Chapter 3

DATA AND PROPOSED METHOD

3.1 Introduction

In this chapter, we will discuss the proposed algorithm, dataset and validation methods used in evaluating the proposed algorithm. Furthermore, we will discuss the measurement metrics employed to assess the performance of the proposed algorithm. This thesis selected three different datasets obtainable at the UCI ML repository. These include: Pima Indians dataset which has a total of 8 attributes including the class attribute, Cleveland dataset which contains 75 attributes including the class attributes and Arrhythmia dataset which has 279 attributes including the class attributes. This is in order to test the proposed algorithm against dataset with small, medium and high number of attributes.

3.2 Datasets

3.2.1 Heart disease Datasets

This dataset is made up of four different databases which were contributed to the repository by Andras Janosi of Hungarian Institute of Cardiology Budapest, William Steinbrunn of University Hospital Zurich, Matthias Pfisterer of University Hospital Basel Switzerland and Robert Detrano of V.A. Medical Center Long Beach and Cleveland Clinic Foundation. Each database contains 76 attributes although not all studies on the dataset refer to all the attributes. Rather, most studies are performed on a reduced attribute of 14 features in particular; the Cleveland dataset is the most widely used. In this dataset the attributes represent the presence or absence of heart

disease. Where 0 stands for absence and 1, 2, 3, 4 represent the absence of the disease. Although most researches (including this thesis) pay more attention on classifying only presence and absence, classes 1, 2, 3, 4 represent different classes of heart diseases. Recently, the names and social security numbers of patients were replaced with dummy values for privacy and security reasons. See Appendix A1 for attribute information on this dataset. This dataset is obtainable at: https://archive.ics.uci.edu/ml/datasets/Heart+Disease

3.2.1.2 Information Summary

Set Characteristics:	Multivariate	**Number of Instances:**	303	**Area:**	Life
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	75	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	343800

Table 3.1: Cleveland Heart Disease Dataset Information Summary

3.2.2 Pima Indians Diabetes Data Set

This dataset documents the relationship between the numbers of times women were pregnant and the BMIs of Pima Indian Women older than 21 years old, it has been used by researchers to predict diabetes pedigree of respondents. Variables such as whether the women have diabetes and their diabetes pedigree function are outline in this dataset. Several constraints were placed on the selection of these instances from a larger database by donors of the dataset. In particular, it is worth noting that all patients here are females at least 21 years old of Pima Indian heritage. See appendix A2 for attribute information on this dataset. This dataset is obtainable at: https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

3.2.2.2 Information Summary

Data Set Characteristics:	Multivariate	**Number of Instances:**	768	**Area:**	Life
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	8	**Date Donated**	1990-05-09
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	199047

Table 3.2: Pima Indians Diabetics Dataset Information Summary

3.2.3 Arrhythmia Data Set

This dataset is made up of 279 attributes including the class attribute. Of this attributes, 206 are linear while the remaining are nominal. The dataset was generated from a study by H. Altay Guvenir [27] to classify patient record as having or not having cardiac arrhythmia. In the target attribute, 01 means absence of disease, while 02-15 refer to different types of cardiac arrhythmia. The names and ID numbers of the patients were recently replaced with dummy values by the donor of the dataset for security and privacy issues. For attribute information on this dataset, see appendix A3. This dataset is obtainable at: https://archive.ics.uci.edu/ml/datasets/Arrhythmia

3.2.3.2 Information Summary

Data Set Characteristics:	Multivariate	**Number of Instances:**	452	**Area:**	Life
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	279	**Date Donated**	1998-01-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	116696

Table 3.3: Arrhythmia Dataset Information Summary

3.3 Proposed Crossover and Mutation

As in the traditional GA, the proposed method begins by creating a population of randomly generated individuals. Then these individuals are evaluated using a fitness function (this thesis used two different fitness functions to assess the performance of the algorithm), after the normal elitism, crossover and mutation; a special process of parent selection which uses the elite individuals is performed to select individuals into the mating pool. Then, the special crossover and mutation are then applied to these individuals to generate new offsprings. Finally, offspring created from the normal and special GA operations are put together and the best individuals are selected to next generation. This is repeated until a stopping criterion is met.

To ensure we retain the randomness of the GA, the special mutation generates a little number of offspring of the next generation. Furthermore, crossover and mutation are only applied to elite individuals to encourage greediness of the algorithm. In addition, the minimum requirement to serve as parent is averaged over the whole population. This procedure is suitable for FS because we are only interested in alleles which have higher relationship with target class. Therefore, only alleles agreed upon by elite individuals are considered as important.

3.3.1 Generating a New Population

A new population of chromosomes at every iteration is generated by three different recombination and alteration operations. These are

- Conventional Elitism
- Conventional Crossover and Mutation
- Special Crossover and Mutation

- **Conventional Elitism –** This is the process by which individual chromosomes are sent to the next generation without been altered. Usually, a percentage of the best individuals or individuals that meet some criteria are sent to the next generation in order to preserve good traits over generations hence, the name elitism. As in conventional GA, the rate of elitism used affects the convergence of the algorithm and the average fitness of individuals in a population. In the proposed algorithm, *N'* (a parameter pass to the algorithm or the conventional elitism rate might be used) individuals are selected as elite individuals to the next generation.

- **Conventional Crossover and Mutation –** next, the conventional crossover and mutation which is the process of individuals generation through recombination and alteration is used to generate *N''* number of individuals in the next generation. As in the conventional GA, the type of crossover and mutation operations used greatly affects the fitness of the generated individuals. Hence, a good crossover and mutation operations are required for a better performance.

- **Special Crossover and Mutation –** The remaining *(N''')* individuals in the population are generated using the proposed crossover and mutation operations which will is discussed in the next section. This operation produces fitter individuals because the elite individuals and voting mechanism are used to generate individuals at the same time average population fitness is used for performance measurement to prevent premature

convergence. Therefore, the composition of the next generation at any iteration can be diagrammatically represented as in the figure below:

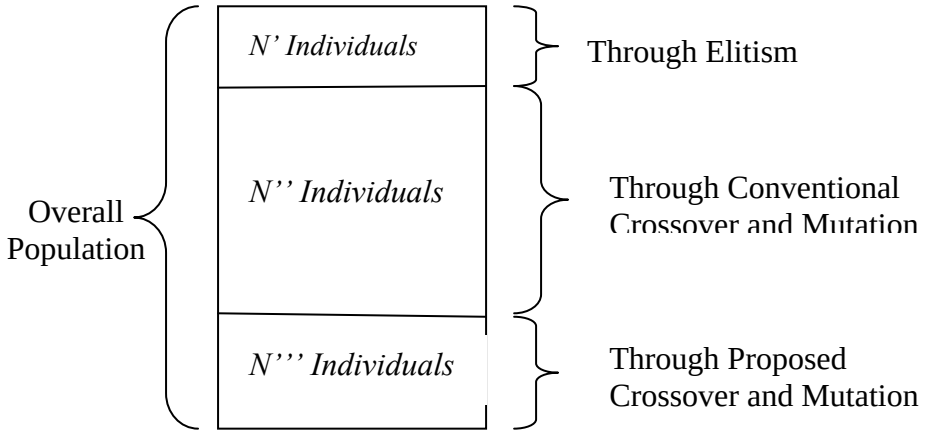


Figure 3.1: Generation of individuals in a population

3.3.2 Formulation of the New Individual

In a GA with a population of N individuals where each individual is composed of M alleles, then this population can be represented as a matrix of $N \times M$ dimension. The fitness of each individual is denoted by $f\ (indiv_n)$, the fitness required to be considered as a good parent is denoted as f_g, the average fitness in the population is denoted by F_{avg}. $indiv_n : f(indiv_n) \geq f_g$ are selected to undergo the special crossover and mutation to create a new individual. That is to say out of N individuals, N' good individuals (those with fitness $\geq f_g$) are selected for the proposed reproduction. The value of f_g is obtained using

$$f_g = g \times F_{avg} \tag{3.1}$$

Where g is a constant [0, 1] which signifies the relevance of the average fitness in the process and F_{avg} is the average fitness in the population and is given by

$$F_{avg} = \frac{\sum_1^N f(indiv_n)}{n} \tag{3.2}$$

If $g=1$ then, f_g will be equal to F_{avg}. The reason for selecting g [0, 1] is to ensure that the whole search space is been explored. After obtaining the minimum requirement to be selected as a parent (i.e. F_{avg} and f_g), the sum of 1's alleles across both horizontal and vertical directions of the matrix ($N \times M$) is obtained. The sum of alleles in the horizontal direction serves as indicator of the number of alleles which should be present in the new individual and is obtained as

$$L = h \times L_{avg} \tag{3.3}$$

Where L is the number of 1's in the parent h is a constant [0, 1] and L_{avg} is the average 1's alleles in the horizontal direction and is given as

$$L_{avg} = \frac{\sum_{n=1}^{N'} L_n}{N} \tag{3.4}$$

Where L_n is the sum of occurrences of 1's alleles in the horizontal direction which represents the number of attributes selected by an individual and is given by

$$L_n = \sum_{m=1}^{M} a_{nm} \tag{3.5}$$

And the sum of 1's in the vertical direction is the voting weight of a selected feature that determines which allele should be a 1 in the generated offspring and defined by

$$V_m = \sum_{m=1}^{M} a_{nm} \tag{3.6}$$

The created offspring will be composed of 1 alleles selected from the highest constant V_m $m=1$ *to* M. A single individual is considered for mutation using bit flip mutation where a single allele with a b_i value of one standard deviation below the mean (i.e. 1 value below L_{avg} is flipped from a zero to a 1 to generate another individual. More individuals are generated by repeating this procedure for all other alleles with one value below L_{avg} until the required number of N''' is obtained

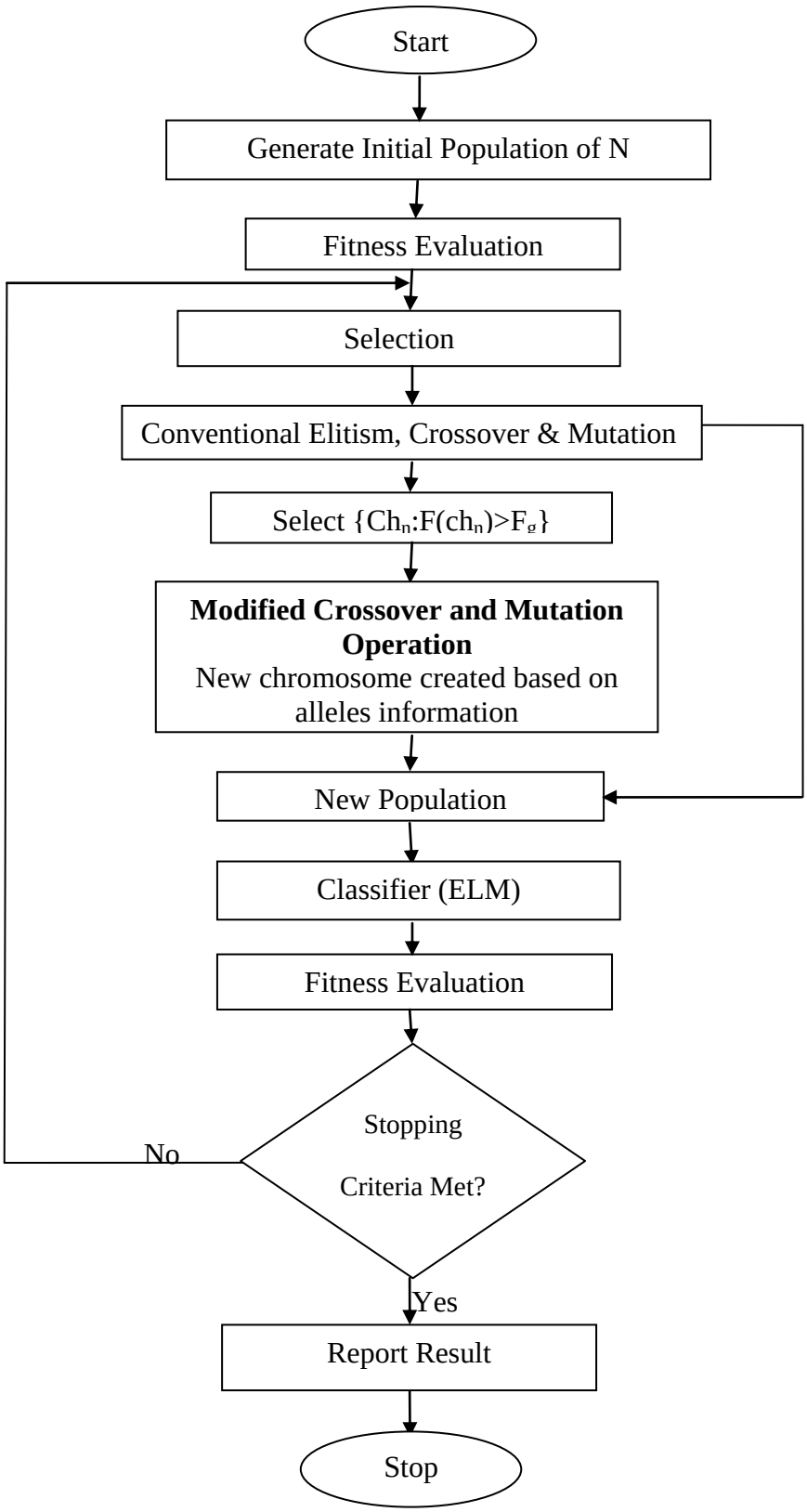


Figure 3.2: Flow chart of the proposed Procedure

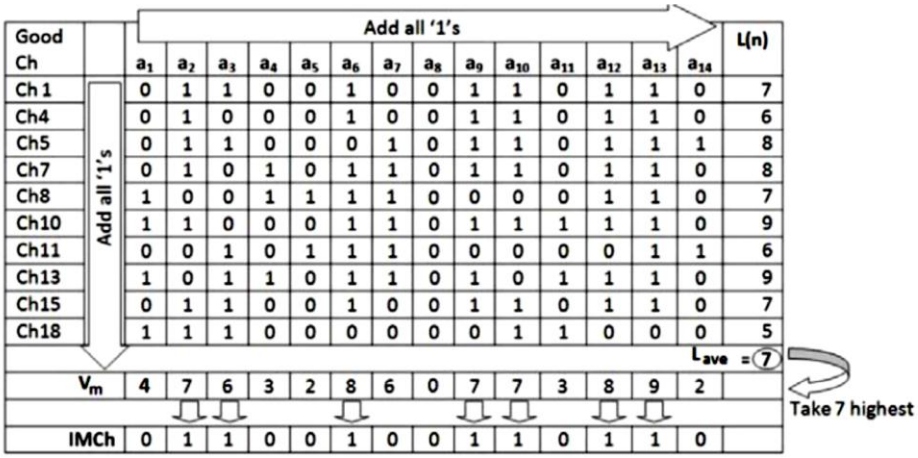


Figure 3.3: Example of Proposed Crossover and Mutation Technique

In the example above, L_{avg} =7 and $l = L_{avg.}$ Hence, the new individual will have 7 1's alleles. For us to get these alleles we use the biggest vertical alleles which are found at *m=13, 12, 6, 10, 9, 2, 3* with values 9,8, 8, 7, 7, 7, 6 respectively, these alleles are then ranked based on V_m and the top l ranked alleles are chosen. Here, we chose 6 which is the next rank allele because 7 cannot be chosen.

3.4 Data Partition

3.4.1 Training Set

In ML, a training set is composed of patterns and target vectors which are used by a learning algorithm to formulate a classification model (for example Neural Network or Naïve Bayes classifier). In these fields, emphasis are placed on avoiding to overfit, so as to achieve the best possible generalization performance on an independent dataset known as the test set that follows the same probability distribution as the training set. This thesis used 60% of each dataset as training set. This is the prevalent amount used for training by most researchers in the ML community. This amount ensures that the learning algorithm will see more than half (50%) example of data instances which is a precursor to a good generalization.

3.4.2 Validation Set

In order to avoid overfitting in any learning algorithm, it is necessary to have a validation set apart from the training and test sets. For example, in a situation where the best classifier for the problem is sought after, the training set is employed to train the various candidate algorithms, then the validation set is used to contrast their various performances and conclude which one to use, finally, the test set is used to obtain the performance of the selected algorithm. The validation set serves a myriad of functions: it is a training set used by testing the algorithm, but neither as part of the final testing, nor as part of the low-level training. A simple procedure is to set aside a part of the training set and it as validation set. This is referred to as the holdout method. Alternatively, this partitioning process can be repeated, where the original training set is partitioned into training and validation sets; this is referred to as the cross-validation. These repeated partitioning can be performed in a number of ways, such as splitting the dataset into two sets and using them as training-validation and then validation-training, or repeatedly selecting a random subset of the dataset as a validation set this is known as k-fold cross-validation.

3.4.3 Purpose of Cross-validation

In a learning process where the task is to fit a model with multiple parameters on a given dataset, the induction algorithm tends to optimize these parameters as best as possible on the given dataset. This is known as overfitting. This is more probable in a case where the data observations are few or the model parameters are many. To overcome this, an independent set of data is used to measure how the algorithm has fit the learning model. This can also be useful in a situation where we have to choose from among competing algorithms. Thus, a learning algorithm with the least fit algorithm is selected.

3.4.4 k-Fold Cross-validation

In this type of cross-validation, the dataset is randomly segmented into k equal partitions. A single partition is kept aside for cross-validation and the remaining k-1 partitions are used for training. This procedure is iterated for n (folds) times. Then, an average performance is obtained thereof which signifies the performance of the algorithm been evaluated. This process of cross-validation has a benefit over others because each data item is used for training and validation. Each observation is used for training n times for training and once for validation. In this thesis *n=10* is used and is the most commonly used regiment [6] but k is variable. Where k is equal to *n* (sample size), then this becomes a Leave One Out (LOO) cross-validation. Usually, k is chosen to make the mean response value over all the folds equal. However, for bi-class classification this means every fold will contain almost the same proportions of the two classes. A diagrammatic representation of k-fold cross validation is shown below:

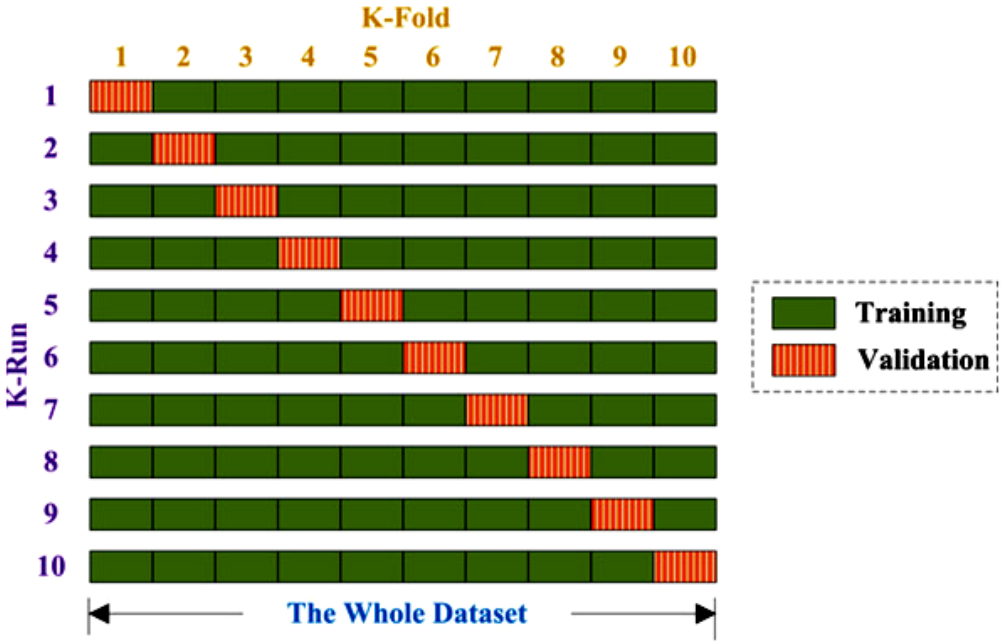


Figure 3.3: K-fold Cross validation [25]

Chapter 4

EXPERIMENTAL RESULTS

4.1 Introduction

In this chapter the results obtained from the computational experiment are presented. Here, Genetic Algorithm with the proposed crossover and mutation methods was used against the three selected datasets (Pima Indians, Cleveland and Arrhythmia) and the performance was compared with the performance of the traditional algorithm. Furthermore, the GA used two different fitness functions in this research (1) a fitness function which penalizes individual chromosomes which select more features as proposed by [5]. (2) a fitness function which uses only the classifier accuracy. The performance was evaluated and finally, the proposed GA operator was added to the GA and the performance was evaluated. The evaluation metrics used in this thesis included:

(1) Accuracy of the classifier

(2) Convergence of the algorithm

(3) Average individual fitness

(4) Diversity of individuals

(5) Number of features selected by best individual.

Other investigated issues include effect of parent pool size on the proposed algorithm, effect of population size on the convergence of the algorithm and stability of the GA using the proposed algorithm, the effect of probability of mutation and

crossover on the proposed method. Finally, from the Extreme Learning Machine (ELM) point this study reported specificity and sensitivity of the model trained using the selected features, ROC of the model and the confusion matrix. The experiment started with the traditional algorithm then a special fitness function was added to the GA. Thereafter, the proposed crossover and mutation was added with and without the fitness function. Here, an ELM (with 10 hidden neurons on both Pima Indians and Cleveland Datasets, 20 Hidden neurons on Arrhythmia datasets and a sigmoid activation function) was trained to access the performance of the proposed method. Finally, the accuracy, sensitivity, specificity and ROC of the trained model for Cleveland dataset were reported.

4.2 Results

This study started by investigating the performance of the traditional GA on the three datasets (Pima Indians, Cleveland, Arrhythmia). The GA used a population size of 50, maximum iteration of 100, elitism of 25%, and crossover rate of 20% and mutation rate of 2% as suggested by [21]. While in each dataset the number of bits is equal to the total number of features in the dataset. Thus at the end of an experiment, the number of alleles with a 1 signifies those features which are selected for example a chromosome with the following alleles 0011101 on the Pima Indians dataset means the 3^{rd},4^{th},5^{th} and 7^{th} were selected. The result of traditional GA is presented in the figure below:

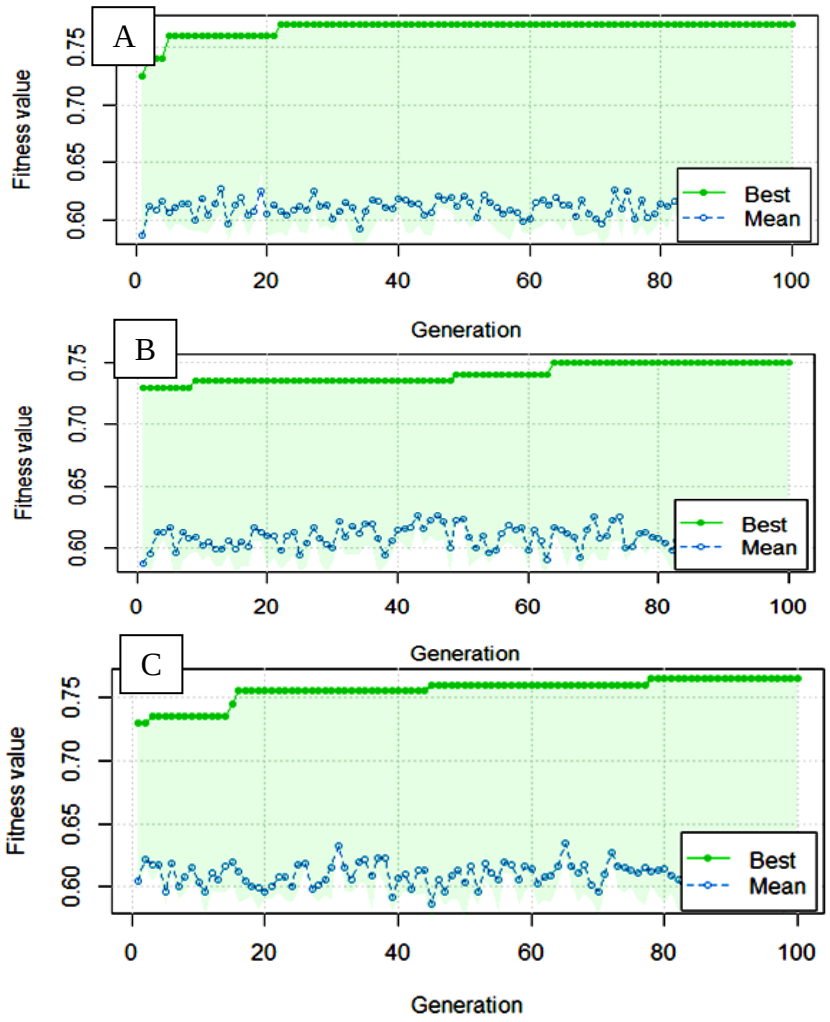


Figure 4.1: Result for Traditional GA without special crossover and mutation, without special fitness function. (A)Pima Indians (B) Cleveland (C) Arrhythmia

In the figure above, the convergence of the traditional GA on the three test datasets is presented. Here, the fitness function did not penalized individuals with higher number of selected features. It can be seen that the algorithm achieved an accuracy of 0.76 which is below the average reported accuracies [1, 8, 11, 14] and the algorithm converged at the 30th iteration in the case of Pima Indians dataset which has a small feature set of 8 features, 80th in the case of Arrhythmia dataset which has a large dataset of 279 features. Furthermore, the best fitness achieved here is 0.76 while the average fitness is 0.54 which points to the wide gap between individual fitness using this algorithm. Therefore, in situations where multiple equally fit individuals are

required this algorithm may not be of any use. Again the fact that the algorithm improved even at the 80th iteration can be seen as yet to converge because the achieved accuracy is below the baseline reported accuracies as earlier noted

Next the GA was integrated with a special fitness the function which penalizes an individual for selecting more features as discussed above. Theoretically, this is expected to force all individuals to conform to some environmental factors thereby making them more fit and so the population will be highly qualitative. The result of this experiment is presented below:

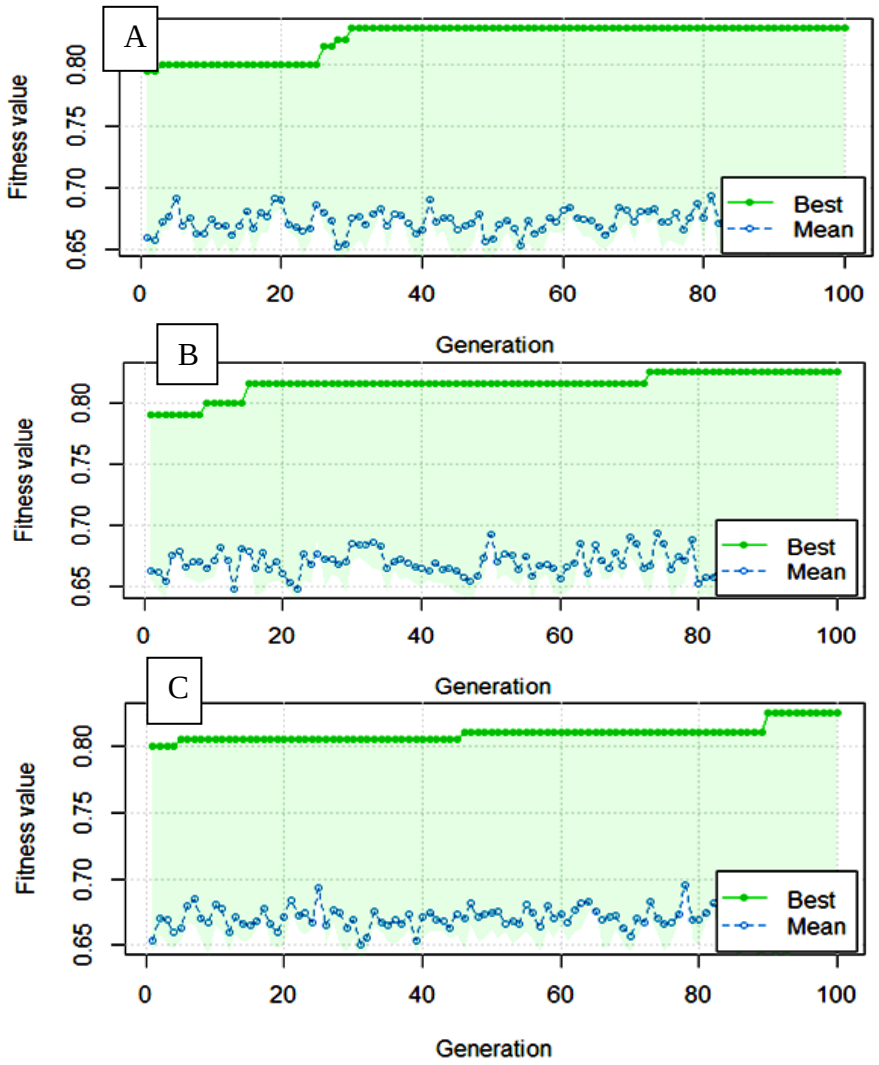
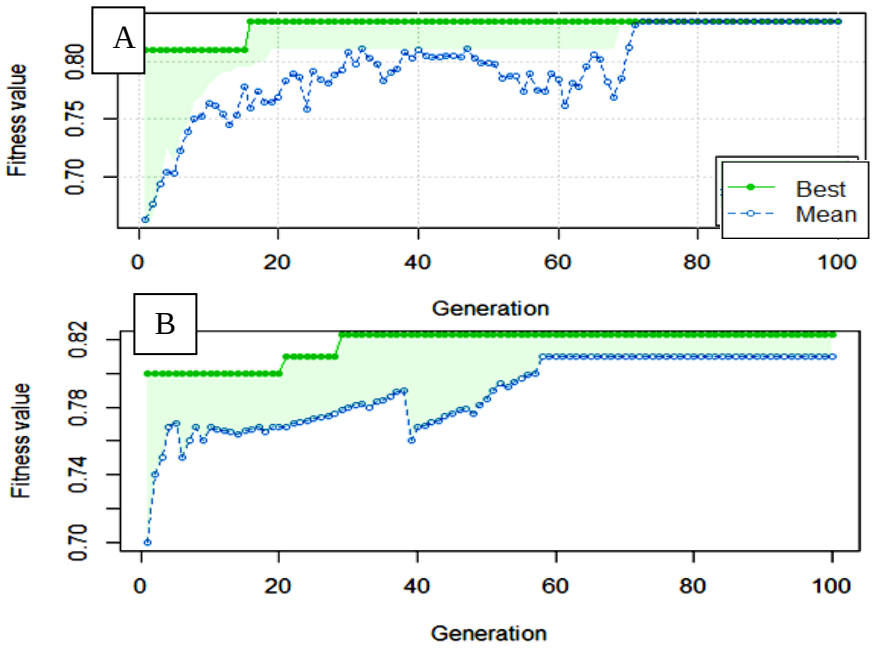


Figure 4.2: Result for Traditional GA without special crossover and mutation with special fitness function. (A)Pima Indians (B) Cleveland (C) Arrhythmia

In the figure above, all the other algorithm parameters and settings were kept same as in the first experiment only the special fitness function was introduced. The result indicates a relative improvement in the convergence time but the achieved accuracy remained the same. At the end of this experiment the algorithm selected 4 features (insulin, sugar, no of pregnancies and age) out of the 8 original features as the best indicators for the Pima Indians dataset. More so, it selected 34 features out of the original 76 features as the best indicators in the case of the Cleveland dataset and 126 features out of 276 features for Arrhythmia dataset. This indicates that the algorithm with this parameters and settings performed below the reported baseline performance [1, 8, 11, 14]



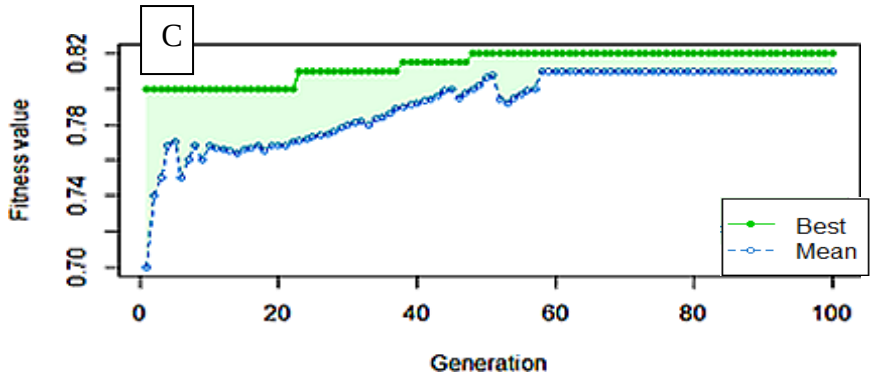


Figure 4.3: Result for GA with special crossover and mutation without Special fitness function. (A)Pima Indians (B) Cleveland (C) Arrhythmia

In the figure above, the proposed crossover and mutation were introduced to the GA. As in the two previous experiments, all other parameters and settings were unchanged. Furthermore, a simple fitness function which uses only the classifier accuracy (does not penalize individuals with higher selected features) was used. The result showed a leap in achieved accuracy of the classifier from 0.76 in the experiment two above to 0.84. Proportionately, the algorithm converged at the 20^{th} iteration for Pima Indians dataset, 25^{th} iteration for Cleveland and 50^{th} iteration for Arrhythmia dataset which is an improvement over the two earlier reported experiments. Furthermore, it can be seen that the difference between the best and average fitness in the population over have stabilized over the after the convergence of the algorithm for all the datasets:

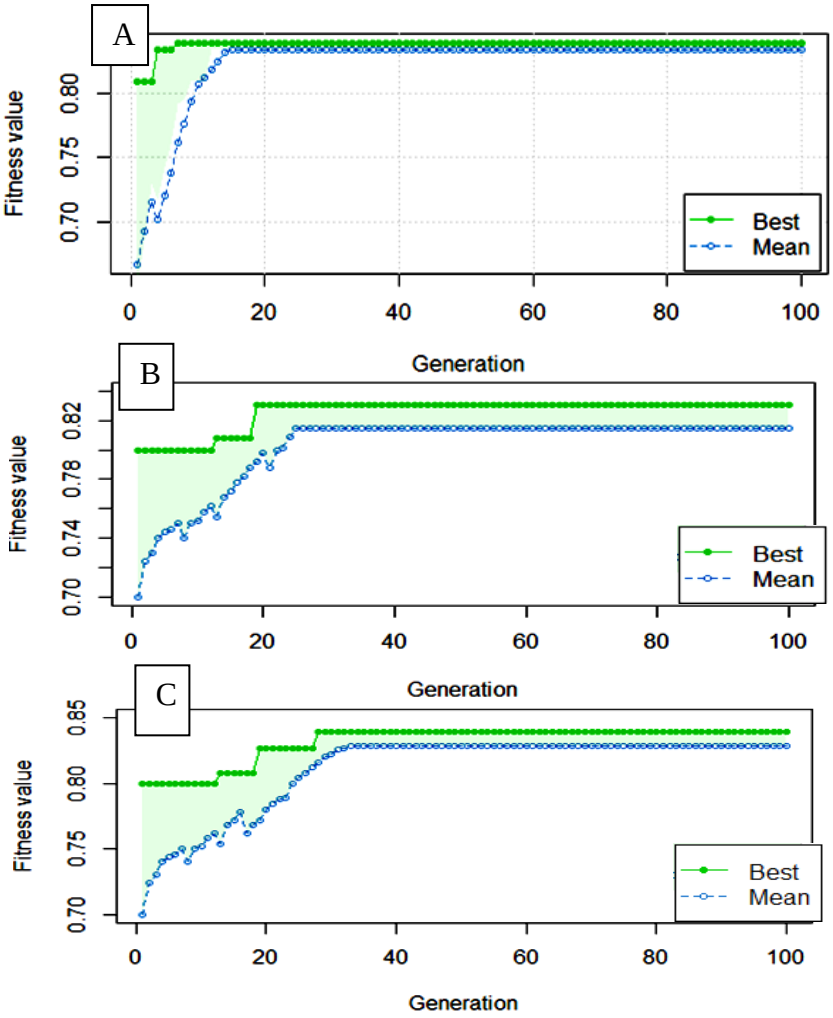


Figure 4.4: Result for Traditional GA with special crossover and mutation and fitness function. (A)Pima Indians (B) Cleveland (C) Arrhythmia

In the figure above, the proposed crossover and mutation was used in addition to a fitness function which uses both classifier accuracy and higher selected feature penalty. All other algorithm parameters and settings were kept constant. This is to maintain consistency in the test setup for all the three datasets. It can be seen that the difference between the best and average fitness was relatively low in the case of Cleveland and Arrhythmia datasets and much lower in the case of Pima Indians dataset. Furthermore, the achieved classifier accuracy improved tremendously from what was reported in all the other experimental setups. More so, the number of selected features for Pima Indians is 3 features (which is less than the baseline report [17] while in the case of Cleveland this experimental setup selected 8 features which

is also less than the 13 reported by most studies [1, 8, 11, 14]. Finally, it is worth noting that the average individual fitness in this setup is almost close to the best fitness which is an added advantage of the algorithm. This is useful in situations where more than one solution is required.

As discussed in section 3.3 one of the parameter settings of a GA is elitism size. This parameter determines which individuals go to the next generation without been changed thereby, preserving good traits in the population. This parameter was also used as a determinant of the parent pool size in the proposed algorithm, therefore, this thesis investigated the effect of elitism on the performance of the proposed algorithm using the Cleveland dataset and the result is presented in the table below:

Elitism	**No Iteration Before Convergence**	**Classifier Accuracy**	**No of Features Selected**
0	65	0.751	31
10	19	0.81	10
20	27	0.0.792	11
30	51	0.79	13
40	24	0.0.78	18
50	32	0.765	21
60	62	0.76	23
70	67	0.756	25

Table 4.1: Result of algorithm convergence for different elitism size on the proposed algorithm

In the table above, it can be seen that as the size of elitism increased the convergence performance of the algorithm, classifier accuracy dropped and the number of selected features increased. This is expectedly from the fact that as the parent pool size increase the sum of vertical important alleles will increase thereby making much

allele important. Therefore, the algorithm will be unable to drop as much unimportant alleles as possible and will require more iterations to converge. It is expected that using a very high rate of mutation will solve this problem as mutation will introduce new information into the population. In this case, a bit flipping mutation is the only applicable mutation. This is subject to further research.

Another parameter of GA which affects the behavior of the algorithm is population size. This parameter represents the number of solutions generated in each iteration and thus a variation in the population size will determine how long it takes to get the best individual. Hence, this thesis investigated the effect of population size on the performance of the proposed algorithm the Cleveland dataset and result is presented in the table below:

Population Size	**No Iteration Before Convergence**	**Classifier Accuracy**	**No of Features Selected**
20	62	0.74	24
25	58	0.76	21
30	52	0.765	21
35	44	0.78	19
40	32	0.786	17
45	21	0.792	16
50	18	0.818	12

Table 4.2: Result of algorithm convergence for different population size on the proposed algorithm

In the table above, it can be seen that as the size of the population increased the convergence performance of the algorithm improved. This is because at each iteration the number of individuals created or recombined is high. Therefore, the

probability of obtaining the best individual is increased as the number of individuals in an iteration increased.

In GA, the crossover rate is the probability of an individual to undergo the recombination process. This parameter controls the intensity of the search process in the GA. Been one of the proposed techniques in this thesis, we investigated how different values of the parameter affects the learning process. Whence, different values of crossover rate where set for the GA with the Cleveland dataset using the same parameters as in the other experiments. The result is presented below:

Crossover Rate	**No Iteration Before Convergence**	**Classifier Accuracy**	**No of Features Selected**
40	43	0.77	19
45	39	0.79	19
50	26	0.793	18
55	22	0.798	17
60	20	0.80	15
65	16	0.804	13
70	14	0.81	10

Table 4.3: Result of algorithm convergence for different probability of cross over on the proposed algorithm

From the table above, it can be seen that the number of iteration before convergence, classifier accuracy and number of selected features selected improved slowly as the crossover rate increased. This may be due to the high number of individuals that undergo recombination and the subsequent number of created individuals per iteration. Thus, as the intensity of the search increased, the required number of iteration reduced and more number of good individuals were generated which improved the classifier performance.

Mutation rate is the probability that an individual will undergo asexual reproduction i.e. new individuals will be generated from it without combining its traits with those of other individuals. This thesis investigated the effect of mutation rate on the proposed technique and the result obtained is presented below:

Mutation Rate	**No Iteration Before Convergence**	**Classifier Accuracy**	**No of Features Selected**
0.4	62	0.78	22
0.5	51	0.78	21
0.6	48	0.79	19
0.7	44	0.81	16
0.8	35	0.812	14
0.9	28	0.82	12
1	22	0.83	12

Table 4.4: Result of algorithm convergence for different probability of mutation on the proposed algorithm

Here, the mutation rate had more impact on the number of iteration before convergence but had little impact on the classifier performance. This can be from the fact that as mutation is increased the GA search diversity increases and the search tends to behave more like a random search. Thereby, requiring more iterations to converge but not affecting its ability to find better individuals faster.

One of the most desirable properties of an algorithm is its stability i.e. how it behaves when the experiment is repeated a number of times. In this study, the proposed algorithm was repeated for 10 times with Pima Indians dataset and the convergence iteration of the algorithm in each of the experiments was compared with that of the traditional GA. The result is presented in the figure below:

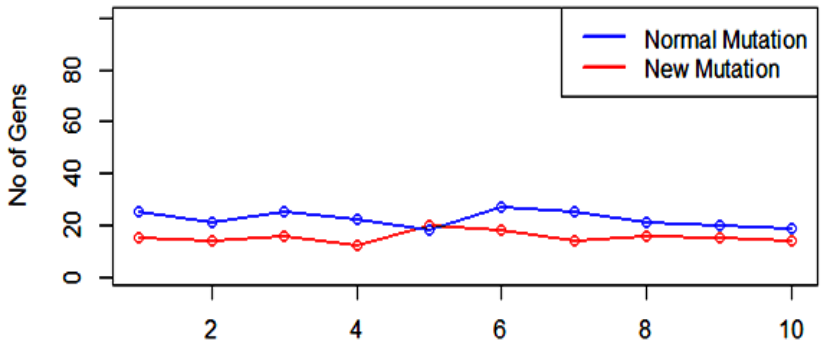


Figure 4.5: Convergence Comparison of Traditional and Modified GA

It can be seen from the table above the proposed algorithm has consistently converged before the 20th iteration. While, by comparison the traditional algorithm has always converged after the 20th iteration. In essence, the worst performance of the proposed method is the same as the best performance of the traditional algorithm in all the experiments.

A Receiver Operating Characteristic (ROC) curve shows the accuracy performance of a classification or induction model as its discrimination threshold is varied. The curve is created by plotting the False Positive Rate (FPR) against the True Positive Rate (TPR) at various threshold settings. The ROC curve is thus the sensitivity as a function of fall-out. The ROC curve for the trained model on Pima Indians dataset with proposed techniques is shown below:

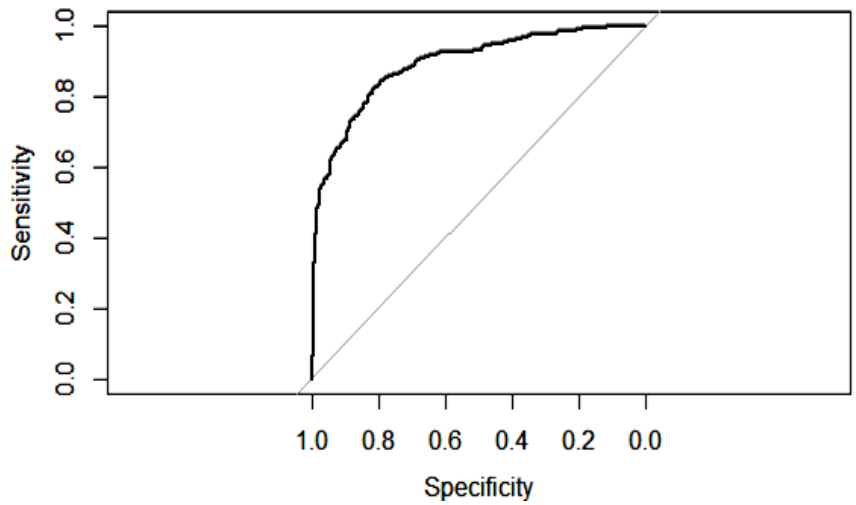


Figure 4.6: ROC Curve for the trained Model on Pima Indians Dataset

In general, the probability distributions for both detection and false alarm rate of the learned model are high. The Area Under the curve AUC of the learned model is 0.782 which points to its accuracy.

In statistical classification, a confusion or error matrix is a table that shows the performance of a classifier or induction algorithm, normally a supervised learning algorithm. Each column of the table shows the instances of predicted classes while each row shows the instances of the actual class (or vice-versa). The confusion matrix of the trained model on Pima Indians dataset with the proposed techniques is shown below:

	Diabetic	**Non-Diabetic**
Diabetic	408	72
Non-Diabetic	51	237

Table 4.5: Trained Model Confusion Matrix on Cleveland dataset

Chapter 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

This work reviewed the problem of data dimensionality growth in ML in current times. Furthermore, it reviewed the current approaches used for the task of feature subset selection in a situation where speed and computational cost are of high importance.

Subsequently, a new crossover and mutation technique was introduced to the traditional GA which is used in the task of FS for medical record classification. Pima Indians, Cleveland and Arrhythmia datasets of UCI ML Repository were used to test the performance of the proposed method. In addition, a fitness function which penalizes individuals for selecting more features as proposed by [33] was used to enhance the performance of the proposed method.

The main aim of the study is to provide a recombination procedure for GA which facilitates the convergence of the algorithm and improves the accuracy of the classifier by reducing the number of redundant and irrelevant features from the dataset. Here, Extreme Learning Machine (ELM) was used as a classifier because it requires less or no parameter tuning and is faster than most backpropagation learning algorithms. The obtained result was verified using a stratified 10-fold cross

validation on all the three datasets where 60-40 data partition was used for training and testing respectively.

The result obtained suggest the superiority of the proposed method over the traditional algorithm in convergence, classifier accuracy and population diversity. To this end, the proposed method also performed better than some of the reported performances [2, 3, 8].

Additionally, issues that affect the performance of the proposed method such as probability of crossover and mutation, elitism and population size were also investigated. The result obtained showed that the proposed algorithm performed better when elitism and crossover rate were high and population size and mutation rate are low.

Finally, it is worth noting that in a situation where all individuals in the mating pool give the same importance to all features in the dataset (i.e. equal vote for all alleles) this algorithm may not be applicable.

5.2 Recommendation for future work

Based on the received results from this study, an investigation of the proposed method for premature convergence is highly recommended. This is because of the seeming fast convergence of the algorithm. Furthermore, the compatibility of the algorithm with other proposed fitness function such as [22, 23, 23] is a subject of research as the classifier accuracy is greatly suppressed by some of this proposed fitness function when they are intended to be used with the proposed algorithm.

REFERENCE

[1] Haleh Vafaie, Kenneth De Jong (2012). *Genetic Algorithms as a Tool for Feature Selection in Machine Learning*, Springer.

[2] Tong Liu, Liang Hu, Chao Ma, Zhi-Yan Wang, Hui-Ling Chen (2015). A Fast Approach for Detection of Erythemato-Squamous Diseases Based On Extreme Learning Machine With Maximum Relevance Minimum Redundancy Feature Selection *International Journal of Systems Science* Volume 46, Issue 5.

[3] Xiaolong Peng, Pan Lin , Tongsheng Zhang, Jue Wang (2013). Extreme Learning Machine-Based Classification of ADHD Using Brain Structural MRI Data Published: November 19, 2013.

[4] S. Kim and E. Xing (2009). Statistical Estimation of Correlated Genome Associations To A Quantitative Trait Network. PLoS genetics, 5(8):e1000587.

[5] M. Dash and H. Liu (1997). Feature selection for classification. Intelligent Data Analysis.

[6] S. Ma and J. Huang (2008). Penalized Feature Selection and Classification in Bioinformatics. *Briefings in Bioinformatics*, 9(5):392–403.

[7] A. S. Muthanantha Murugavel, S. Ramakrishnan (2014). An Optimized Extreme Learning Machine for Epileptic Seizure Detection *International Journal of Computer Science*, 41:4, 30 November 2014.

[8] Bir Bhanu, Yingqiang Lin (2003). Genetic Algorithm Based Feature Selection for Target Detection in SAR Images, *Image and Vision Computing Journal* 21 (2003) 591–608

[9] C. V. Banupriya, S. Karpagavalli (2015). Electrocardiogram Beat Classification Using Support Vector Machine and Extreme Learning Machine ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol I Volume 248 of The Series Advances in Intelligent Systems and Computing pp 187-193

[10] D. Wettschereck, D. Aha & T. Mohri (1997). A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.

[11] X. Wu, K. Yu, H. Wang, & W. Ding (2010). Online Streaming Feature Selection. In Proceedings of the 27th International Conference on Machine Learning, pages 1159–1166, 2010.

[12] Z. Xu, R. Jin, J. Ye, M. Lyu, & I. King (2009). Discriminative Semi-supervised Feature Selection via Manifold Regularization. In IJCAI' 09: Proceedings of the 21th International Joint Conference on Artificial Intelligence, 2009.

[13] G. Geetha, S. N. Geethalakshmi (2011). Detecting Epileptic Seizures Using Electroencephalogram: A New and Optimized Method for Seizure Classification Using Hybrid Extreme Learning Machine International

Conference on Process Automation, Control and Computing (PACC), 2011 , Pages 1 – 6, Coimbatore.

[14] H. Shahamat, A. A. Pouyan (2012). Feature Selection Using Genetic Algorithm for Classification of Schizophrenia Using fMRI Data *Journal of AI and Data Mining* Vol 3, No 1, 2015, 30-37.

[15] J.G. Dy and C.E. Brodley (2004). Feature Selection for Unsupervised Learning. *The Journal of Machine Learning Research*, 5:845–889, 2004.

[16] Q. Gu, Z. Li, and J. Han (2012). Generalized Fisher Score for Feature Selection. arXiv preprint arXiv:1202.3725, 2012.

[17] M.A. Hall & L.A. Smith (1999). Feature Selection for Machine Learning: Comparing A Correlation-Based Filter Approach To The Wrapper. In Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference, volume 235, page 239, 1999.

[18] I. Inza, P. Larranaga, R. Blanco, and A. J. Cerrolaza (2004). Filter Versus Wrapper Gene Selection Approaches in DNA Microarray Domains. Artificial Intelligence in Medicine, 31(2):91–103, 2004.

[19] Riccardo Leardi (2000). Application of Genetic Algorithm–PLS for Feature Selection in Spectral Datasets *Journal Of Chemometrics* 2000; Vol 14: 643–655

[20] H. Liu and L. Yu (2005). Toward Integrating Feature Selection Algorithms for Classification and Clustering. *IEEE Transactions on Knowledge and Data Engineering,* 17(4):491, 2005.

[21] Junchang Xin, Zhiqiong Wang, Chen Chen, Linlin Ding, Guoren Wang, Yuhai Zhao (2013). *Distributed Extreme Learning Machine with MapReduce* , 29 June 2013, springer

[22] L. Yu and H. Liu (2003). Feature Selection for High-dimensional data: A Fast Correlation-based Filter Solution. In International Conference on Machine Learning, volume 20, page 856, 2003.

[23] M. Yuan and Y. Lin (2006). Model Selection and Estimation in Regression With Grouped Variables. *Journal of the Royal Statistical Society*: Series B (Statistical Methodology), 68(1):49–67.

[24] Z. Zhao and H. Liu (2007). Semi-supervised Feature Selection via Spectral Analysis. In Proceedings of SIAM International Conference on Data Mining.

[25] D. Zhou, J. Huang, and B Scholkopf (2005). Learning from Labeled and Unlabeled data on a directed graph. In International Conference on Machine Learning, pages 1036–1043.

[26] H. Zou and T. Hastie (2005). Regularization and Variable Selection via The Elastic net. *Journal of the Royal Statistical Society*: Series B (Statistical Methodology), 67(2):301–320.

[27] J. Wang, P. Zhao, S. Hoi, and R. Jin (2013). Online Feature Selection and its Applications. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14.

[28] P. Mitra, C. A. Murthy, and S. Pal (2002). Unsupervised Feature Selection Using Feature Similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:301–312.

[29] M. Robnik-˘Sikonja and I. Kononenko (2003). *Theoretical and Empirical Analysis of Relieff And RRelieff. Machine learning*, 53(1-2):23–69.

[30] Cheng-Lung Huang , Chieh-Jen Wang (2006). A GA-based Feature Selection and Parameters Optimization for Support Vector Machines *Journal of Expert Systems with Applications* 31 (2006) 231–240

[31] Priyanka khare, D. Kavita (2016). Burse Feature Selection Using Genetic Algorithm and Classification using Weka for Ovarian Cancer *International Journal of Computer Science and Information Technologies*, Vol. 7 (1), 194-196

APPENDIX

Appendix A: Dataset Attributes Documentation

A1: Cleveland Dataset

1 id: patient identification number

2 ccf: social security number (I replaced this with a dummy value of 0)

3 age: age in years

4 sex: sex (1 = male; 0 = female)

5 painloc: chest pain location (1 = substernal; 0 = otherwise)

6 painexer (1 = provoked by exertion; 0 = otherwise)

7 relrest (1 = relieved after rest; 0 = otherwise)

8 pncaden (sum of 5, 6, and 7)

9 cp: chest pain type

-- Value 1: typical angina

-- Value 2: atypical angina

-- Value 3: non-anginal pain

-- Value 4: asymptomatic

10 trestbps: resting blood pressure (in mm Hg on admission to the hospital)

11 htn

12 chol: serum cholestoral in mg/dl

13 smoke: I believe this is 1 = yes; 0 = no (is or is not a smoker)

14 cigs (cigarettes per day)

15 years (number of years as a smoker)

16 fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

17 dm (1 = history of diabetes; 0 = no such history)

18 famhist: family history of coronary artery disease (1 = yes; 0 = no)

19 restecg: resting electrocardiographic results

-- Value 0: normal

-- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

-- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

20 ekgmo (month of exercise ECG reading)

21 ekgday(day of exercise ECG reading)

22 ekgyr (year of exercise ECG reading)

23 dig (digitalis used furing exercise ECG: 1 = yes; 0 = no)

24 prop (Beta blocker used during exercise ECG: 1 = yes; 0 = no)

25 nitr (nitrates used during exercise ECG: 1 = yes; 0 = no)

26 pro (calcium channel blocker used during exercise ECG: 1 = yes; 0 = no)

27 diuretic (diuretic used used during exercise ECG: 1 = yes; 0 = no)

28 proto: exercise protocol

1 = Bruce

2 = Kottus

3 = McHenry

4 = fast Balke

5 = Balke

6 = Noughton

7 = bike 150 kpa min/min (Not sure if "kpa min/min" is what was written!)

8 = bike 125 kpa min/min

9 = bike 100 kpa min/min

10 = bike 75 kpa min/min

11 = bike 50 kpa min/min

12 = arm ergometer

29 thaldur: duration of exercise test in minutes

30 thaltime: time when ST measure depression was noted

31 met: mets achieved

32 thalach: maximum heart rate achieved

33 thalrest: resting heart rate

34 tpeakbps: peak exercise blood pressure (first of 2 parts)

35 tpeakbpd: peak exercise blood pressure (second of 2 parts)

36 dummy

37 trestbpd: resting blood pressure

38 exang: exercise induced angina (1 = yes; 0 = no)

39 xhypo: (1 = yes; 0 = no)

40 oldpeak = ST depression induced by exercise relative to rest

41 slope: the slope of the peak exercise ST segment

-- Value 1: upsloping

-- Value 2: flat

-- Value 3: downsloping

42 rldv5: height at rest

43 rldv5e: height at peak exercise

44 ca: number of major vessels (0-3) colored by flourosopy

45 restckm: irrelevant

46 exerckm: irrelevant

47 restef: rest raidonuclid (sp?) ejection fraction

48 restwm: rest wall (sp?) motion abnormality

0 = none

1 = mild or moderate

2 = moderate or severe

3 = akinesis or dyskmem (sp?)

49 exeref: exercise radinalid (sp?) ejection fraction

50 exerwm: exercise wall (sp?) motion

51 thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

52 thalsev: not used

53 thalpul: not used

54 earlobe: not used

55 cmo: month of cardiac cath (sp?) (perhaps "call")

56 cday: day of cardiac cath (sp?)

57 cyr: year of cardiac cath (sp?)

58 num: diagnosis of heart disease (angiographic disease status)

-- Value 0: < 50% diameter narrowing

-- Value 1: > 50% diameter narrowing

(in any major vessel: attributes 59 through 68 are vessels)

59 lmt

60 ladprox

61 laddist

62 diag

63 cxmain

64 ramus

65 om1

66 om2

67 rcaprox

68 rcadist

69 lvx1: not used

70 lvx2: not used

71 lvx3: not used

72 lvx4: not used

73 lvf: not used

74 cathef: not used

75 junk: not used

76 name: last name of patient (I replaced this with the dummy string "name")

A2: Pima Indians Attributes Documentation

1. Number of times pregnant

2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test

3. Diastolic blood pressure (mm Hg)

4. Triceps skin fold thickness (mm)

5. 2-Hour serum insulin (mu U/ml)

6. Body mass index (weight in kg/(height in m)^2)

7. Diabetes pedigree function

8. Age (years)

9. Class variable (0 or 1)

A3: Arrhythmia Attribute Documentation:

1 Age: Age in years , linear

2 Sex: Sex (0 = male; 1 = female) , nominal

3 Height: Height in centimeters , linear

4 Weight: Weight in kilograms , linear

5 QRS duration: Average of QRS duration in msec., linear

6 P-R interval: Average duration between onset of P and Q waves in msec., linear

7 Q-T interval: Average duration between onset of Q and offset of T waves in msec., linear

8 T interval: Average duration of T wave in msec., linear

9 P interval: Average duration of P wave in msec., linear Vector angles in degrees on front plane of:, linear

10 QRS

11 T

12 P

13 QRST

14 J

15 Heart rate: Number of heart beats per minute ,linear Of channel DI: Average width, in msec., of: linear

16 Q wave

17 R wave

18 S wave

19 R' wave, small peak just after R

20 S' wave

21 Number of intrinsic deflections, linear

22 Existence of ragged R wave, nominal

23 Existence of diphasic derivation of R wave, nominal

24 Existence of ragged P wave, nominal

25 Existence of diphasic derivation of P wave, nominal

26 Existence of ragged T wave, nominal

27 Existence of diphasic derivation of T wave, nominal Of channel DII:

28 .. 39 (similar to 16 .. 27 of channel DI) Of channels DIII:

40 .. 51 Of channel AVR:

52 .. 63 Of channel AVL:

64 .. 75 Of channel AVF:

76 .. 87 Of channel V1:

88 .. 99 Of channel V2:

100 .. 111 Of channel V3:

112 .. 123 Of channel V4:

124 .. 135 Of channel V5:

136 .. 147 Of channel V6:

148 .. 159 Of channel DI: Amplitude , * 0.1 milivolt, of

160 JJ wave, linear

161 Q wave, linear

162 R wave, linear

163 S wave, linear

164 R' wave, linear

165 S' wave, linear

166 P wave, linear

167 T wave, linear

168 QRSA , Sum of areas of all segments divided by 10, (Area= width * height / 2), linear

169 QRSTA = QRSA + 0.5 * width of T wave * 0.1 * height of T wave. (If T is diphasic then the bigger segment is considered), linear Of channel DII:

170 .. 179 Of channel DIII:

180 .. 189 Of channel AVR:

190 .. 199 Of channel AVL:

200 .. 209 Of channel AVF:

210 .. 219 Of channel V1:

220 .. 229 Of channel V2:

230 .. 239 Of channel V3:

240 .. 249 Of channel V4:

250 .. 259 Of channel V5:

260 .. 269 Of channel V6:

270 .. 279