VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Using Genetic Algorithms to Perform Feature and Instance Selection for Classification

Inti Mateus Resende Albuquerque

Supervisors: Bing Xue, Mengjie Zhang

Submitted in partial fulfilment of the requirements for Bachelor of Engineering with Honours in Software Engineering.

Abstract

Feature and instance selection is a crucial part of pre-processing for machine learning classification problems involving a large amount of training data. This is because a significant volume of data will often include redundant features and instances, which do not contribute in a positive way to the outcome of the algorithm that are being used in the training process. There are many techniques used to complete this task; however, these mainly treat feature selection and instance selection as two distinct problems. The goal of this project is to create a solution for classification algorithms in the field of machine learning. Using Evolutionary Computing to encapsulate the tasks of feature selection and instance selection into a single algorithm.

Acknowledgments

I want to thank both of my supervisors; Bing Xue and, Mengjie Zhang for the opportunity to work on this project to complete my Honours. I would also like to thank Xiaoying Sharon Gao and Ying Bi for being present on the meetings and provide feedback on the work done. Furthermore, I would like to thank Ramesh Rayudu for being the course coordinator and providing support for the project. At last, but not less important, I would like the other engineering students in my year for giving me support and helping me proofread my work.

Contents

Chapter 1

Introduction

Data is a crucial part of any machine learning algorithm [1]. As the amount of data being produced grows [2], filtering out irrelevant values in data sets becomes more and more critical. Having redundant features and instances could decrease the accuracy of classification algorithms, increase their time complexity [3] and cause classification models to overfit[4].

There are several ways to achieve feature and instance reduction; one of the conventional approaches is through Evolutionary Algorithm[5]. However, most of the current solutions to the problem are done by dealing with features and instances separately [6]. This is because the search space is too large when these tasks are performed together, meaning there could be too many combinations of features and instances to search through. Nevertheless, combining feature and instance selection into one algorithm is still a task worth investigating, as it has the potential to achieve higher data reduction while maintaining the accuracy, as different combinations of features and combination of instances could correlate to each other. When performing these tasks together, these combinations are more likely to be maintained than when the tasks are performed separately, because the algorithm evaluates the combination of them together rather than evaluating combinations of features and then instances separately.

The goal of this research is to analyse further how combining these two tasks rather than having them separated may benefit the increase in data reduction and affect the accuracy of the classification models. The research will focus on developing a hybrid algorithm derived from Genetic Algorithms and Local Search techniques to achieve this goal.

1.1 Motivation

Classification methods are becoming more popular due to the increase of computational power in the last decades [7]. This allows computers to process more significant amounts of data in a feasible amount of time. The amount of data that can be stored, also increased, allowing the storage of data with many instances(rows in a data set) and features(columns in a data set).

Classification algorithms are computer programs that allow computers to classify instances [8]; these algorithms require a large amount of data to be accurate [9]. Hence the growth in data allows these algorithms to perform more accurately on average. However, some features and instances have none or negative impact on computational tasks, and removing them could improve the accuracy achieved by the algorithms [10].

For a data set with N features and M instances, the total size of the data set would be $N*M$. Because feature and instance selection decreases the dimensional of both, N and M, it has the potential achieve a far greater reduction of data dimensionality than only feature

selection or only instance selection.

Reducing the number of features and instances of a data set decreases the storage capacity needed to store that specific data set, and has the potential to improve the data set for classification purposes. The goal of this project is to explore a different approach to the problem of feature and instance selection, which will reduce the data used for classification.

1.2 Objectives

Genetic Algorithm(GA) has been used successfully to solve both tasks, feature selection [3], and instance selection [11]. To perform the task of feature selection, Local Search [12] has also been applied successfully in combination with GA [13]. The overall goal of this project is to reduce the size of a data set while increasing the accuracy of classification algorithms using the data set. To achieve this goal, there are four main objectives:

1. Create a simple feature and instance selection algorithm with a classifier accuracy as the fitness function;

2. Develop well-designed crossover and mutation operators;

3. Develop a fitness function that encapsulates both features and instances used in the solution as well as the accuracy achieved on the classifier.

4. Implement Local Search and merge it with the implementation of GA.

1.2.1 Objective 1

The representation of an individual and the generation of the initial population are crucial tasks for GA to achieve adequate results. Creating a basic version of GA to accomplish the task of feature and instance selection will allow us to explore a good way to specify both, a generation function and the representation of individuals. Because not much work has been done on the topic, this might be a difficult task to achieve [14].

1.2.2 Objective 2

To achieve relevant results, GA requires well-designed mutation and crossover operators [15]. Depending on the task, the design of these can be reasonably tricky. Because this project is merging two different tasks into one, the design of crossover and mutation operators will require innovative thinking. Crudely designed operators for this task will lead the algorithm to perform poorly when solving the classification task.

1.2.3 Objective 3

An adequate fitness function is what will direct the algorithm into a satisfying solution, rather than performing an utterly random search. The failure to design a suitable fitness function for the feature and instance selection will result in an algorithm with lousy performance. The proposed algorithm will use a wrapping approach, wrapping classification algorithms into a fitness function is a commonly used method to measure the fitness of an individual in GA [3].

1.2.4 Objective 4

Local Search has been successfully implemented with GA to perform the task of feature selection [13]. However, it has not been used for the task of feature and instance selection simultaneously. Incorporating this method into the proposed algorithm could increase its performance. This algorithm could be useful to further develop the best solutions created by GA.

1.3 Organisation

The remaining parts of this report are organised as follows. Chapter 2 is a literature survey describing related work on the topic and motivations for this project. Chapter 3 presents an overview of the algorithms used as the baseline and explains how the experiments were conducted to ac hive the results presented in the next sessions. The achievements so far, and the analysis of the results are covered in Chapters 4 and 5. Finally, Chapter 6 concludes the report providing a summary of all ideas discussed in the other sections.

Chapter 2

Literature Survey

As computation power increases, Machine Learning [16] has become more popular. The task of machine learning for classification requires clean data to be accurate. The term clean data, in machine learning, referrers to data with little to no noise. The data for the machine learning task is usually divided into two sets: training data set and test data set. The training data is used to teach the algorithm on how to solve a problem, while the test data is used to validate how well the algorithm solved the problem for previously unknown data. Given that the test set is supposed to be unseen, it can not be used in the evolutionary process of the algorithm. Therefore, when using Evolutionary Algorithms, the training set is usually split into two, one set used to train the algorithm and the other to calculate the fitness of the current solutions [17].

To assure that machine learning algorithms perform well on the test set, the instances in the training set need to have features that follow the same probability distribution to those in the test set, meaning they are similar. This will allow the algorithm to predict the instances in the test set correctly based on the training set. Feature and instance reduction are tasks that can improve that aspect of on the data.

The tasks of feature and instance selection, as crucial parts for machine learning, have been analysed by several individuals using several different measures. Already in 1999, there were surveys on methods used to solve the task at the time [18]. Even though these problems have been analysed for 20 years, there is no final solution only approximations, as these problems are in the subset of NP-Hard problems [19].

High dimensional data can contain many noisy features and instances, and when used to train a machine learning model can lead to overfitting [4]. Overfitting is when the trained model achieves high accuracy on the training set, but the accuracy achieved on the test set does not correlate. This happens when the model does not learn a generalised solution to solve the problem and is biased towards the training set. Feature and instance selection can remove noise in the data, preventing models from overfitting when trained on the training set.

2.1 Genetic Algorithm

Genetic Algorithm is a subset of evolutionary algorithms [20]. This set of algorithms is based on natural selection [21]. In nature, organisms are composed of chromosomes, which are composed of genomes. According to the theory of natural selection, chromosomes mutate over time, allowing organisms to evolve and better adapt to the environment they live in. Chromosomes can also be mixed trough reproduction. Reproduction allows two parents to generate a child which will have a mixture of both parents genomes. GA tries to simulate

this process in a computer algorithm to provide solutions to problems [22]. An example of the usage of GA can be found here [23]; in this example, GA is used to evolve cars to drive in different environments. In each generation, a new population of vehicles is generated with new individuals who could potentially outperform the previous generation in the same environment. GA is composed of the following aspects:

- **Individual:** an individual is a possible solution for the problem.
- **Chromosome:** chromosomes are parts that compose an individual.
- **Genomes:** These are parts that form a chromosome; genomes are domain-specific and vary depending on the problem.
- **Population:** a population is a group of individuals.
- **Fitness:** fitness is how well an individual solves the problem; it is evaluated using a fitness function.
- **selection:** this is the process that selects individuals of the population, based on their fitness values, to create a new population.
- **Crossover:** crossover mixes two different individuals to derive new solutions from the mixture.
- **mutation:** mutation changes the composition of an individual to try achieving a better solution.
- **Generation:** generation is a population at an iteration.

The genomes and chromosomes differ from one problem to another; the representation of these has to be decided by the developer depending on attributes of the problem. This means that generating the individuals and the fitness function will require a certain amount of domain-specific knowledge. However, the remaining parts of the algorithm are not domain-specific, allowing GA to solve a wide variety of problems.

The population is composed of diverse individuals; this allows it to have enough diversity enabling the algorithm to evaluate multiple solutions and cover a large search space [24]. In approximation algorithms such as GA, several solutions are developed for a problem. However, there are many reasonable solutions that are not-optimal which are called local solutions. The optimal solution is referred to as a global solution [25]; a global solution is the best possible solution for a problem. Having enough diversity can also help the algorithm develop a globally optimal solution rather than a local one, which is essential for search algorithms. Usually, global solutions are not found as it is tough to find them. Instead, a best local solution that approximates the problem well is developed. Often, the population size is consistent throughout the whole algorithm.

The fitness function varies from problem to problem; in general terms, it measures how well an individual solves a given task. The fitness function requires a certain level of domain-specific knowledge from the developer when being designed, as one needs to know how to measure the proximity of a solution to that specific problem. The design of the fitness function is essential, as it will guide the algorithm to provide a good solution. Poorly designed fitness functions will lead to a poorly performing algorithm [26].

Generally, the selection step selects individuals from the current population to generate a new one, made of individuals with good fitness values. However, there are other factors that need to be taken into account when selecting a new population; diverse potential solutions need to be preserved. To preserve both diversity and fitness, any individual in the

population has the chance to be selected, but the ones with lower fitness are less likely to be chosen to be taken into the next population. This process allows individuals to evolve, aiming for a better solution.

The crossover step generally mixes genomes of two individuals producing two offsprings. This generates two new solutions from two existing solutions that could potentially have a better fitness value than prior solutions. There are several ways the crossover can be done; however, the most typical one copies fifty per cent of one parent and fifty per cent of the other to each offspring. Mixing individuals often create a better solution or one that can contribute to a better solution in the future [27].

The mutation step in the algorithm replaces a percentage of genomes of a given individual with different ones. This allows diversity to be maintained and could help the algorithm develop a global solution. The diversity of the population tends to decrease as the program runs, because of the selection step selecting the current best solutions which tend to be similar to each other. An adequately designed mutation function helps increase the coverage of the search space, allowing the algorithm to pursue different solutions[28].

In a generation, a new population is created from the old one. Typically, individuals in the population are crossed over, mutated and then selected into a new population. Usually, a number of generations are used as stopping criteria, where the algorithm runs for N generations until it stops and the results are analysed.

2.2 Related Work

Local Search

Local search is a search method used to solve problems by making small changes to current solutions of a problem trying to achieve a better result. This method aims to find a local best solution for the given problem. Local search is often used to approximate NP-Hard problems [19] and typically performs relatively well [29]. When trying to find a solution, the algorithm searches through the neighbourhood, a set of next possible solutions, and jumps to one of them if the result is better than the previous solution. The neighbourhood should be defined as close solutions, meaning that there is no drastic change between any of the neighbours and the current solution.

2.2.1 Feature Selection

Feature selection is the task of reducing the number of columns of a data set. Some features in a data set do not have a positive impact on classification algorithms, so minimising these features should not decrease the accuracy of the algorithm. However, removing them results in a smaller data set, taking less memory to store, and can increase the accuracy of the algorithm; as some features can contribute to noise in the data set. There are many different algorithms used to approximate the problems of feature and instance selection; this project will be focusing on two of them: GA and Local Search.

Both methods have been successful in approximating a solution for the problem individually [3][30]and together [13]. However, there is still work to be done in terms of improving the time complexity, accuracy and data reduction capability of the algorithms.

2.2.2 Instance Selection

Instance selection is the task of reducing the number of rows of a data set. Similarly to features, some instances can cause noise in a data set and do not contribute in a positive

way for the task of classification. Removing these instances reduces the amount of space need to store the data set and can also contribute to a higher accuracy on classification tasks. Although a substantial amount of work has been done on the topic [31], there is not as much work done as there is for feature selection. This might be because both are relatively similar, and concepts of feature selection also apply to the selection of instances. Similar to feature selection, there has been a substantial amount of research using GA for this task, and it again proved useful [11].

2.2.3 Genetic Algorithm for Feature and Instance Selection

GA is a good algorithm to approach NP-Hard problems [32]. For the tasks of feature and instance selection, GA can search an ample space and compute acceptable solutions with less computational power than other methods, and it does not require a large amount of domain-specific knowledge to solve the problem [33]. Because of these benefits, GA is a widely used algorithm for both of these tasks. However, there are few people who analysed both of these tasks together. Most times, when they are investigated together, one is performed after the other [6] and when performed together, the accuracy suffers due to the large search space [14]. However, finding a good solution for both of these problems together is still a problem worth analysing.

The approach of using a hybrid algorithm that merges GA and Local Search seems promising, as local search could search for a better solution around the solution found by GA. these methods used together have not yet been analysed for the task of feature and instance selection. When used for feature selection, the hybrid algorithm performed well for the task [13]. Hence this project will analyse the performance of both, GA and Local Search, for the task of feature and instance selection.

2.2.4 Limitations

One of the reasons why there is no substantial amount of work done on the tasks of feature and instance selection performed together is potentially the ample search space. When performed separately, the search space the algorithm has to cover is already vast; when performed together, this space becomes exponentially larger. This raises difficulties when navigating the space, searching for an acceptable solution.

These two tasks by themselves are computationally expensive, and when performed one after another, they can take a long time to complete. Performing them simultaneously is more expensive than running them individually. However, with processing power increases, this is a feasible task to be performed.

2.3 Summary

The tasks of feature and instance selection are crucial for the pre-processing of unclean data, as these reduce data dimensions and improve classification accuracy. A large number of methods have been developed to solve these problems separately; however, these methods rarely approach the tasks together. This is because the search space to find an optimal solution for both of these tasks together is very large. This project proposes a solution that will cover the search space creating a solution that will reduce the dimensions of the given data significantly, improving or maintaining the classification accuracy on the data containing all features and instances.

Chapter 3

The Baseline Algorithm

This chapter will analyse and describe an overview of the baseline algorithms and their methods. It will have diagrams illustrating the general algorithm and will provide a general overview of its methods. It will also offer insides on the methods used for testing and will describe the results achieved by the proposed algorithm on the data sets.

3.1 Chapter Goals

The goal of this chapter is to provide a general overview of the baseline algorithms and its methods. The diagrams will be as understandable and as straightforward as possible while describing the general outline in great detail. It will also describe the design of the experiment, explain the parameters used, describe the data sets used to test the algorithm and provide an overview of the baseline method as well as the results achieved by the methods.

3.2 Algorithm Overview

All algorithms in this report are based on GA. At the start, a new population of individuals is generated using the generation function. Once all individuals are created, the Genetic Algorithm process starts. Each individual in the population has the chance to be mutated, how much it is mutated by is decided by a variable specified before running the program. After the mutation process is done, individuals have a chance of being crossed over. After the crossover process, each new individual in the population has its fitness value evaluated, and the new population is selected using a selection method. Once the last generation is reached, the algorithm stops and displays the results.

The above-described methods are explained in detail in the next chapters for the algorithm developed in this project.

3.3 Algorithm Description

The algorithm developed in this project is an adapted version of GA for the task of feature and instance selection, it uses the standard genetic operators(mutation, crossover and selection) and the final version will also have a local search implementation.

3.3.1 Individual Representation

The individuals are represented as two binary lists, one representing the features and the other representing the instances. The Genomes can have a value of 0 or 1, representing

whether a feature or instance is present in the evaluation or not.

Figure 3.1: Individual Representation

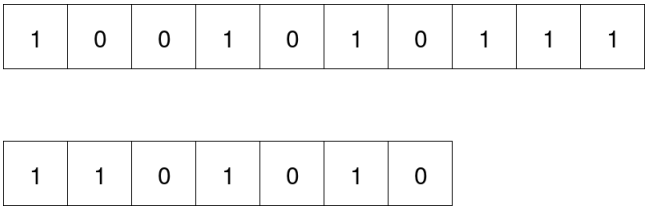


Figure 3.1 above shows the representation of an individual with six out of the ten original features and four out of seven initial instances.

3.4 Experiment Design

3.4.1 Feature Selection Baseline Description

The sklearn-genetic algorithm is an extension of the sklearn, a Python library for Machine Learning, for feature selection [34]. Using this algorithm as the baseline is acceptable as it performs GA for feature selection, a similar task to the one being evaluated on this project, successfully. The comparison to this baseline shows how the method tested in this project performs in comparison to other similar algorithms.

3.4.2 Feature And Instance Selection Baseline Description

As there are not many code sources available for a feature and instance selection Genetic algorithm, a baseline algorithm for this research was created. The algorithm implements the most straightforward genetic operators having a uniform crossover[35] and a mutation function that mutates one genome.

The selection method used is tournament selection [36]. Tournament selection selects a *tournmentSize* number of random individuals of the population and selects the best to be taken in to the next generation. This allows each individual in the population to have a chance of being taken in to the next generation.

The population initialisation function is more thought of than the other methods, and enforces data reduction, as explained in the next section. Comparing the final algorithm to this baseline will show how the methods proposed in this project contribute to the accuracy and data reduction of the final solution.

Generation Of The Population

The population is created by generating individuals with an increasing number of features and instances. The first 10% of the population is created with 10% of features, and 10% of instances allocated randomly, the remaining 90% of the instances and features are not used. The number of features and instances increase by 10%, of the possible amount, whenever a tenth of the population is generated. By the end of the generation, all features and instances should be represented in the individuals of the population. Each individual in the population holds 10-90% of all possible features and instances by the end of the generation.

Pseudo Code:

Algorithm 1 Initial Population Generation

```
1: function GENERATE(populationSize) m ← total number of features n ← total number of instancess
2:     ListfeaturePercent ← percentages evenly distributed values from 0.1 - 0.9 of length populationSize
3:     ListinstancePercent ← percentages evenly distributed values from 0.1 - 0.9 of length populationSize
4:     population ← empty list
5:     for i in populationSize do
6:         Fp ← select a random number from ListfeaturePercent
7:         remove Fp from ListfeaturePercent
8:         features ← (Fp * m) indices filled with 1s and the rest with 0s
9:         Ip ← select a random number from ListinstancePercent
10:        remove Ip from ListinstancePercent
11:        instances ← (Ip * n) indices filled with 1s and the rest with 0s
12:        add an individual with features and instances to population
13:    end for
14:    return population
15: end function
```

In the first version of this function, individuals could contain up to 100 % of possible features and instances. However, the results achieved with 10-90 % of the features present were better and provided a better reduction on the features and instances.

Fitness Function

To reduce bias in the accuracy achieved by the individual on the classifier, the algorithm uses K-Fold Cross-Validation. Dividing the data, used in the evolution process, into K batches, on each run one batch will be tested against the rest of the data and after K runs the average of the result of those runs is taken as the accuracy of the individual.

Mutation

The mutation function on for the Feature And Instance Selection Baseline mutates one random bit on the features and one on the instances of the given individual.

Crossover

The crossover function on for the Feature And Instance Selection Baseline copies half of one individual to the second one, creating two new offsprings.

Pseudo Code:

Algorithm 2 Crossover

```
function CROSSING(individual1, individual2) m ← total number of features n ← total number of instancess
    for Fi in m do
        rn ← number between 0-1
        if rn is smaller than 0.5 then
            values of individual1 and individual2 at Fix
        end if
    end for
    for Ii in n do
        rn ← number between 0-1
        if rn is smaller than 0.5 then
            swap values of individual1 and individual2 at Ii
        end if
    end for
    return individual1, individual2
end function
```

3.4.3 Data Sets

All data sets used to test the performance of both the baseline and the algorithm created in this project, had a medium to large numbers of features and instances. The data sets were retrieved from UCI, and Kaggle data sets [37][38] and are presented in Table 3.1.

Table 3.1: Data Sets

Data set	Number of Features	Number of Instances	Number of Classes
Libras Movement	90	360	15
arrhythmia(Ary)	279	452	12
Mobile KSD Data(Mobile) Set	71	2856	56
Human Freedom Index(Mobile)	118	880	4
First order theorem proving(ML)	56	6118	2
Multiple Features(Mfeat) 216	216	2000	10
Multiple Features(Mfeat) 240	240	2000	10
internet advertisements(Ad)	1558	3279	2
Online News Popularity Data Set(Popularity)	58	39644	11
Smartphone Dataset for Human Activity Recognition(UCI)	561	10299	6

Data sets containing a more significant amount of features and instances are more likely to have redundant values in them. These data set are suitable for feature and instance selection. More redundant features and instances can be removed to reduce the size of the data set and improve accuracy for classification.

3.4.4 Parameters

The parameter present in the algorithms were based on the parameters used in the following paper [3] and will be presented and discussed below.

Table 3.2: Parameters

Parameter Name	Parameter Value	FS-Baseline	FS-IS-Baseline
mutation rate	0.3	yes	yes
crossover rate	0.7	yes	yes
generations	50	yes	yes
population	100	yes	yes
tournament size	3	yes	yes
number of folds	10	yes	yes
crossover independent probability	0.5	yes	no
mutation independent probability	0.05	yes	no
scoring	accuracy	yes	no
max features	number of features present at the beginning	yes	no
flipProb	0.09	no	yes

Algorithm Runner

This function puts the algorithm together. It creates the initial population using the generation function; it then performs selection, crossover and mutation to generate a new population. This process is repeated for *numgen* which is the number of generations the algorithm is supposed to run for.

3.5 Testing Method

To ensure that results were not achieved on accident, both algorithms were run on ten different data sets with 30 random seeds. A large number of random samples provide results that can be represented as a normal distribution, allowing a confidence test to be performed in the results. In this experiment, the Wilcoxon rank-sum test was performed on the 30 samples of each data set.

Having ten different data sets ensures that the results achieved by the experiment are not restricted to one data set. While it is expected that the performance of each algorithm varies depending on the given data, taking the average of several data sets provides better inside of the overall performance of the algorithms.

Table 3.2 shown below contains the results of each algorithm on each detest. The data set column is populated with the name of the data set the experiment ran on. The algorithm column is populated with the name of the algorithm the experiment ran on. *Baseline + FS* represents the feature baseline, *Baseline +FS +IS* represents the feature and instance selection baseline, *New Algorithm +FS +IS* represents the methods that achieved the results and *Raw Classifier* represents the Classifier algorithm ran on the original data set. The number of Features and Number of Instances columns represent the number of features and instances after the algorithm has performed. The Average Accuracy column is populated with the average accuracy achieved by each algorithm with the 30 random seeds on each data set. The P-value is the result of the Wilcoxon rank-sum test on the results of the data set, achieved by each algorithm in comparisson to the raw classifier.

3.6 Results

Table 3.3: Baseline Results Table

data set	Algorithm	Number of Features	Number of Instances	Average Accuracy	significance test on accuray
Libras	Raw Classifier	90.0	240.0	78.33	
	Baseline Classifier + FS	38.0	240.0	82.67	+
	Baseline Classifier + FS + IS	52.0	202.0	76.86	+
Ary	Raw Classifier	279.0	306.0	60.27	
	Baseline Classifier + FS	131.0	306.0	63.13	+
	Baseline Classifier + FS + IS	145.0	216.0	63.95	+
Human	Raw Classifier	118.0	588.0	93.49	
	Baseline Classifier + FS	51.0	588.0	94.27	+
	Baseline Classifier + FS + IS	75.0	429.0	94.74	+
Mobile	Raw Classifier	71.0	1904.0	39.29	
	Baseline Classifier + FS	32.0	1904.0	46.17	+
	Baseline Classifier + FS + IS	48.0	1497.0	45.4	+
ML	Raw Classifier	56.0	4079.0	94.16	
	Baseline Classifier + FS	5.0	4079.0	100.0	+
	Baseline Classifier + FS + IS	10.0	2725.0	99.93	+
Mfeat 216	Raw Classifier	216.0	1340.0	93.64	
	Baseline Classifier + FS	98.0	1340.0	94.87	+
	Baseline Classifier + FS + IS	133.0	1085.0	94.46	+
Ad	Raw Classifier	1558.0	2186.0	95.88	
	Baseline Classifier + FS	739.0	2186.0	96.96	+
	Baseline Classifier + FS + IS	918.0	1742.0	97.17	+
Popularity	Raw Classifier	58.0	26432.0	40.86	
	Baseline Classifier + FS	18.0	26432.0	41.15	+
	Baseline Classifier + FS + IS	36.0	12733.0	42.16	+
Mfeat 240	Raw Classifier	240.0	1340.0	97.12	
	Baseline Classifier + FS	129.0	1340.0	96.97	+
	Baseline Classifier + FS + IS	179.0	1026.0	97.1	-
UIC	Raw Classifier	561.0	6868.0	96.68	
	Baseline Classifier + FS	278.0	6868.0	98.29	+
	Baseline Classifier + FS + IS	390.0	5488.0	97.64	+

As the table above shows, the feature selection baseline outperforms the feature and instance selection baseline in terms of accuracy and significantly in terms of feature reduction. However the feature and instance reduction baseline, reduces a fair amount of instances, making it possibly better in terms of overall data reduction depending on the data set.

The feature selection baseline performs exceptionally well on the Libras data set in comparison to the feature and instance selection classifier. This is due to the small number of instances in the data set compared to other the ones, which makes it less ideal for the task of instance selection.

3.7 Summary

This chapter describes the baseline algorithms and the tasks they perform. The parameters, data sets and test methods were also explained above.

In the results section it was established that the feature and instance selection is outperformed by the feature selection baseline in terms of accuracy and feature reduction. The overall reduction achieved by the feature and instance selection baseline is better, depending on the data set, as it reduces instances as well as features.

The rest of the report will be presenting the results of each new method and analyse its performance.

Chapter 4

The Proposed Algorithm with New Crossover and Mutation Operators

This chapter will present a description of the crossover and mutation operators used in the algorithm.

4.1 Chapter Goals

The goal of this chapter is to describe both operators and analyse how they affect the performance of the algorithm compared to the baseline methods. The operators will be described in detail and will be illustrated with diagrams and pseudocode.

4.2 Crossover

4.2.1 Crossover Description

The crossover function presented in this section takes two individuals as parents and generates two offsprings. The number of genomes taken from one individual is relative to the difference between the parents' fitness values. The parent with a higher fitness value will have a more significant number of genomes transferred into the next generation through the offsprings. The number of genomes used form the best parent has an upper-bound of 20 per cent. This parameter should be tuned and optimised to achieve the best results. The genomes selected from the best parent are randomly chosen until the percentage based on the difference in fitness between parents is reached. After the genomes from the parent, with a higher fitness value, are transferred, a uniform crossover is performed of the remaining genomes, having 50 per cent of each parent being added to each child.

Pseudo Code:

Algorithm 3 Novel Crossover

function CROSSING(*individual*1, *individual*2) $m \leftarrow$ total number of features $n \leftarrow$ total number of instancess
    $bestIndividual, worstIndividual \leftarrow$ idividuals with best and worst fitness from *individual*1 ad *individual*2
    $keep \leftarrow$ max(difference in fitness from $bestIndividual$ and $worstIndividual$,20 per cent)
    $ListpossibleFeatures \leftarrow$ from 0 to m
    $ListpossibleInstances \leftarrow$ from 0 to n
    for $feature$ in $keep * m$ **do**
        get random index from $ListpossibleFeatures$ and and replace value in
        $worstIndividual$ with the value of $bestIndividual$ at the index
        remove index from $ListpossibleFeatures$
    end for
    for *instance* in $keep * n$ **do**
        get random index from and replace value in $worstIndividual$ with the
        value of $bestIndividual$ at the index
        remove index from $ListpossibleInstances$
    end for
    perform uniform crossover on $bestIndividual$ and $worstIndividual$, as described in
    Chapter 3
    return $bestIndividual, worstIndividual$
end function

Figure 4.1: crossover

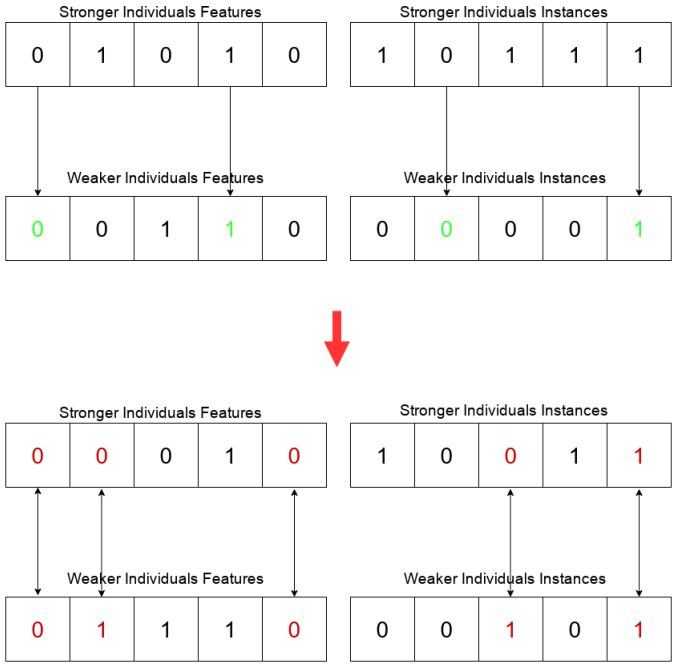


Figure 4.1 above represents the crossover function. The top binary strings represent the

features and instances of the individual with a higher fitness value, while the bottom one represents the one with the lower value. The genomes transferred are assured to be taken into the next generation by both offsprings. On the first part of the diagram, genomes of the better individual(represented in green) are inserted into the individual with a lower fitness value. On the second diagram, after the arrow, the crossover of genes is illustrated having the genes that were swapped in red.

4.2.2 Results

The table below shows the difference in results between the Classifier, the Feature Selection Baseline, the Feature and Instance Selection Baseline and the New Algorithm. The new Algorithm, in this case, is the baseline for feature and instance selection, with the novel crossover operator.

Table 4.1: Crossover Results Table

data set	Algorithm	Number of Features	Number of Instances	Average Accuracy	significance test on accuray
Libras	Raw Classifier	90.0	240.0	78.33	
	Baseline Classifier + FS	38.0	240.0	82.67	+
	Baseline Classifier + FS + IS	52.0	202.0	76.86	+
	New Algorithm + FS + IS	52.0	203.0	77.31	+
Ary	Raw Classifier	279.0	306.0	60.27	
	Baseline Classifier + FS	131.0	306.0	63.13	+
	Baseline Classifier + FS + IS	145.0	216.0	63.95	+
	New Algorithm + FS + IS	144.0	215.0	64.29	+
Human	Raw Classifier	118.0	588.0	93.49	
	Baseline Classifier + FS	51.0	588.0	94.27	+
	Baseline Classifier + FS + IS	75.0	429.0	94.74	+
	New Algorithm + FS + IS	71.0	430.0	94.61	+
Mobile	Raw Classifier	71.0	1904.0	39.29	
	Baseline Classifier + FS	32.0	1904.0	46.17	+
	Baseline Classifier + FS + IS	48.0	1497.0	45.4	+
	New Algorithm + FS + IS	48.0	1509.0	45.5	+
ML	Raw Classifier	56.0	4079.0	94.16	
	Baseline Classifier + FS	5.0	4079.0	100.0	+
	Baseline Classifier + FS + IS	10.0	2725.0	99.93	+
	New Algorithm + FS + IS	10.0	2749.0	99.92	+
Mfeat 216	Raw Classifier	216.0	1340.0	93.64	
	Baseline Classifier + FS	98.0	1340.0	94.87	+
	Baseline Classifier + FS + IS	133.0	1085.0	94.46	+
	New Algorithm + FS + IS	135.0	1077.0	94.33	+
Ad	Raw Classifier	1558.0	2186.0	95.88	
	Baseline Classifier + FS	739.0	2186.0	96.96	+
	Baseline Classifier + FS + IS	918.0	1742.0	97.17	+
	New Algorithm + FS + IS	899.0	1745.0	97.16	+
Popularity	Raw Classifier	58.0	26432.0	40.86	
	Baseline Classifier + FS	18.0	26432.0	41.15	+
	Baseline Classifier + FS + IS	36.0	12726.0	42.17	+
	New Algorithm + FS + IS	36.0	12637.0	42.21	+
Mfeat 240	Raw Classifier	240.0	1340.0	97.12	
	Baseline Classifier + FS	129.0	1340.0	96.97	+
	Baseline Classifier + FS + IS	179.0	1026.0	97.1	-
	New Algorithm + FS + IS	178.0	1031.0	97.04	+
UIC	Raw Classifier	561.0	6868.0	96.68	
	Baseline Classifier + FS	278.0	6868.0	98.29	+
	Baseline Classifier + FS + IS	390.0	5488.0	97.64	+
	New Algorithm + FS + IS	387.0	5504.0	97.7	+

The results of the new crossover operator showed to improve the performance of the algorithm compared to the feature and instance selection baseline in terms of accuracy, feature reduction and in instance reduction. The results show that this novel method does not outperform the baseline for feature selection in terms of feature selection and average accuracy. However, the new methods have a higher data reduction depending on the dat set due to the instance selection capability.

The algorithm with the novel crossover operator outperforms the feature and instance selection baseline slightly in terms of average accuracy; it uses 2.6 fewer features on average and 3.6 fewer instances. The algorithms best performance was on the Popularity data set, reducing 89 more features on average while increasing the accuracy compared to the feature and instance selection baseline.

These results are achieved because features and instances that increase the accuracy have higher chances of being transferred into the next generation, while features and instances that do not contribute to the accuracy are more likely to be discarded. This due to the bias towards the fitter parent, as it assures that a certain amount of features and instances will be identical to the parent with a better fitness value.

The proposed algorithm does not outperform the feature selection baseline in terms of average accuracy and in terms of feature reduction. However, the average amount of instances reduced by the new algorithm is far more significant than the feature selection baseline as the baseline does not perform instance selection.

Due to the high computational cost of running this experiment, the tuning of this method was not performed. The number of genomes copied over has an upper bound of 20 per cent, tuning this parameter could have a positive impact on the performance of this method.

Overview

The algorithm described in this section has the crossover operator as described above all other functions are as described in Chapter 3.

Table 4.2: Parameters

Parameter Name	Parameter Value	FS-Baseline	FS-IS-Baseline	New Algorithm
mutation rate	0.3	yes	yes	yes
crossover rate	0.7	yes	yes	yes
generations	50	yes	yes	yes
population	100	yes	yes	yes
tournament size	3	yes	yes	yes
number of folds	10	yes	yes	yes
crossover independent probability	0.5	yes	no	no
mutation independent probability	0.05	yes	no	no
scoring	accuracy	yes	no	no
max features	number of features present at the beginning	yes	no	no
flipProb	0.09	no	yes	yes

4.3 Mutation

4.3.1 Description

The mutation operator is composed of two parts: soft mutation and hard mutation.

The soft mutation function mutates a low percentage of genomes from each individual. The amount that is mutated is specified as a parameter called *flipProb* representing the probability each genome has of mutating. This parameter is specified before running the program and should be small since this function is supposed only to mutate the individual slightly. Each time the function is invoked, it alternates on mutating the features and instances of the individual passed in. There is also a chance that both features and instances of an individual are mutated simultaneously; this probability is specified with the *feature and instance mutation chance*.

Figure 4.2: Soft Mutation

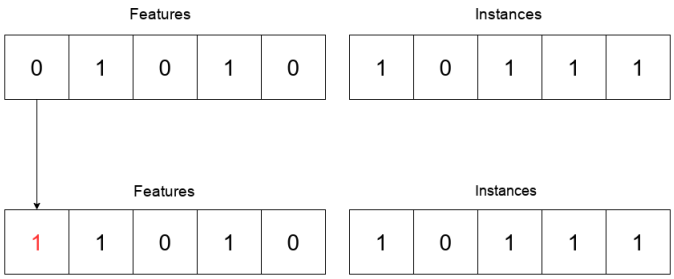


Figure 4.2 illustrates the soft mutation on the features of an individual. The same process occurs for the instances when they are supposed to mutate; the red genome has been mutated and flipped.

The hard mutation function mutates an individual according to its fitness values if the individual has a higher fitness value fewer genomes will be mutated whereas when the individual has a lower fitness value, it will have a higher amount of genomes mutated. This is because mutating a large number of genomes is likely to decrease the accuracy of an individual, so if the accuracy is already reasonably low the individual can have a more substantial amount of genomes mutating without having a significant effect on the best fitnesses of the population. The function will also alternate the mutation between features and instances as well as having a chance of mutating both, like the soft mutation.

Figure 4.3: Hard Mutation

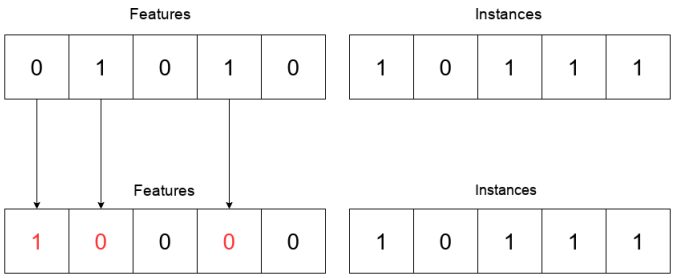


Figure 4.3 illustrates the hard mutation on the features of an individual; the same process occurs for the instances when they are supposed to mutate. The red genomes have been mutated and flipped. In this scenario, the hard mutation rate is four times higher than the soft mutation rate.

What mutation should be performed is decided by the *hardMutationRate* parameter. This parameter should be relatively low, as hard mutation should not occur too often. This is because mutating a large amount of an individual could lead to a substantial decrease in that individuals accuracy. However, this method increases the diversity of the population, allowing it to explore more diverse solutions.

The first implementation had the hard mutation and soft mutation in different functions with different methods and different implementations. However, after testing this turned out to be fragile and hard to tune, therefore in the new implementation, both of these functions are embedded into one method. When a hard mutation is supposed to occur, the *flipProb* is multiplied by the constant *hardMutationAmount*, meaning the individual will have a

higher number of chromosomes mutated. *hardMutationAmount* should not be too little since it would have to little effect on the mutation function and also not too large, as it can cause individuals to decrease their fitness significantly.

Pseudo Code:

Algorithm 4 Novel Mutation

```
1: function MUTATION(individual)fit ← individuals fitness
2:     f ← flipProb
3:     if randomNumber between (0,1) < hardMutationRate then
4:         f ← hardMutationAmount*(1 − fit)
5:     end if
6:     both ← false
7:     if randomNumber between (0,1) < chance of both mutating then
8:         both ← true
9:     end if
10:    if istances should mutate or both then
11:    for each instance in individual do
12:            if randomNumber between (0,1) < f then
13:                instance ←1-instance
14:            end if
15:    end for
16:    end if
17:    for each feature in individual do
18:            if randomNumber between (0,1) < f then
19:                feature ←1-feature
20:            end if
21:    end for
22:    end if
23:    return individual
24: end function
```

4.3.2 Results

The table below shows the difference in results between the Classifier, the Feature Selection Baseline, the Feature and Instance Selection Baseline and the New Algorithm. The new Algorithm, in this case, is the baseline for feature and instance selection, with the novel mutation operator.

Table 4.3: Mutation Results Table

data set	Algorithm	Number of Features	Number of Instances	Average Accuracy	significance test on accuray
Libras	Raw Classifier	90.0	240.0	78.33	
	Baseline Classifier + FS	38.0	240.0	82.67	+
	Baseline Classifier + FS + IS	52.0	202.0	76.86	+
	New Algorithm + FS + IS	53.0	199.0	77.39	+
Ary	Raw Classifier	279.0	306.0	60.27	
	Baseline Classifier + FS	131.0	306.0	63.13	+
	Baseline Classifier + FS + IS	145.0	216.0	63.95	+
	New Algorithm + FS + IS	140.0	207.0	64.95	+
Human	Raw Classifier	118.0	588.0	93.49	
	Baseline Classifier + FS	51.0	588.0	94.27	+
	Baseline Classifier + FS + IS	75.0	429.0	94.74	+
	New Algorithm + FS + IS	71.0	424.0	94.29	+
Mobile	Raw Classifier	71.0	1904.0	39.29	
	Baseline Classifier + FS	32.0	1904.0	46.17	+
	Baseline Classifier + FS + IS	48.0	1497.0	45.4	+
	New Algorithm + FS + IS	47.0	1477.0	45.23	+
ML	Raw Classifier	56.0	4079.0	94.16	
	Baseline Classifier + FS	5.0	4079.0	100.0	+
	Baseline Classifier + FS + IS	10.0	2725.0	99.93	+
	New Algorithm + FS + IS	10.0	2652.0	99.91	+
Mfeat 216	Raw Classifier	216.0	1340.0	93.64	
	Baseline Classifier + FS	98.0	1340.0	94.87	+
	Baseline Classifier + FS + IS	133.0	1085.0	94.46	+
	New Algorithm + FS + IS	136.0	1056.0	94.35	+
Ad	Raw Classifier	1558.0	2186.0	95.88	
	Baseline Classifier + FS	739.0	2186.0	96.96	+
	Baseline Classifier + FS + IS	918.0	1742.0	97.17	+
	New Algorithm + FS + IS	893.0	1731.0	97.16	+
Popularity	Raw Classifier	58.0	26432.0	40.86	
	Baseline Classifier + FS	18.0	26432.0	41.15	+
	Baseline Classifier + FS + IS	36.0	12733.0	42.16	+
	New Algorithm + FS + IS	36.0	13061.0	42.12	+
Mfeat 240	Raw Classifier	240.0	1340.0	97.12	
	Baseline Classifier + FS	129.0	1340.0	96.97	+
	Baseline Classifier + FS + IS	179.0	1026.0	97.1	-
	New Algorithm + FS + IS	176.0	1007.0	97.06	-
UIC	Raw Classifier	561.0	6868.0	96.68	
	Baseline Classifier + FS	278.0	6868.0	98.29	+
	Baseline Classifier + FS + IS	390.0	5488.0	97.64	+
	New Algorithm + FS + IS	387.0	5332.0	97.71	+

Using the novel mutation operator showed to improve the accuracy as well as feature reduction compared to the feature and instance selection baseline. The number of instances reduced by the baseline is slightly, but not significantly higher than the algorithm with the new method.

The results suggest that the mutation operator increased the average accuracy of the algorithm slightly in comparison to the feature and instance selection baseline, the number of features reduced is also lager having 3.7 more features reduced on average. The new algorithms best feature reduction occurred on the Ad data set, and it reduced 25 more features while keeping the accuracy.

The accuracy and feature reduction in the feature selection baseline is higher than the one achieved by the new method. The algorithm, including the new mutation function, performs a grater overall data reduction depending on the data set due to its instance reduction capability.

The results achieved by the new mutation method, are due to greater diversity in the population, as the mutation inserts a more considerable amount of random genes into the population than the previous methods. This allows the algorithm to search a broader range of possible solutions. The novel mutation function also takes into account the current fitness of the individual and does not fully randomise reasonable solutions.

The parameters for the mutation method could not be optimised due to the time required to run the algorithm. Tuning the parameters *feature and instance mutation chance* and *hard-*

MutationAmount could improve the performance of the algorithm, by allowing it to focus on instances and features simultaneously or more separated by changing the *feature and instance mutation chance* and also allowing the search space to be searched more thoroughly using a higher *hardMutationAmount*.

Overview

The algorithm described in this section contains the mutation operator as described above all other functions are as described in Chapter 3.

Table 4.4: Parameters

Parameter Name	Parameter Value	FS-Baseline	FS-IS-Baseline	New Algorithm
mutation rate	0.3	yes	yes	yes
crossover rate	0.7	yes	yes	yes
generations	50	yes	yes	yes
population	100	yes	yes	yes
tournament size	3	yes	yes	yes
number of folds	10	yes	yes	yes
crossover independent probability	0.5	yes	no	no
mutation independent probability	0.05	yes	no	no
scoring	accuracy	yes	no	no
max features	number of features present at the beginning	yes	no	no
feature and instance mutation chance	0.7	no	no	yes
flipProb	0.09	no	yes	yes
hardMutationRate	0.2	no	no	yes
hardMutationAmount	4	no	no	yes

4.4 Mutation and Crossover

4.4.1 Description

This section of the chapter will analyse the results of both above-described methods put together.

4.4.2 Results

The table below shows the results achieved by the new algorithm in comparison to the baselines.

Table 4.5: Mutation and Crossover Results Table

data set	Algorithm	Number of Features	Number of Instances	Average Accuracy	significance test on accuray
Libras	Raw Classifier	90.0	240.0	78.33	
	Baseline Classifier + FS	38.0	240.0	82.67	+
	Baseline Classifier + FS + IS	52.0	202.0	76.86	+
	New Algorithm + FS + IS	52.0	200.0	76.92	+
Ary	Raw Classifier	279.0	306.0	60.27	
	Baseline Classifier + FS	131.0	306.0	63.13	+
	Baseline Classifier + FS + IS	145.0	216.0	63.95	+
	New Algorithm + FS + IS	143.0	209.0	65.14	+
Human	Raw Classifier	118.0	588.0	93.49	
	Baseline Classifier + FS	51.0	588.0	94.27	+
	Baseline Classifier + FS + IS	75.0	429.0	94.74	+
	New Algorithm + FS + IS	67.0	418.0	94.54	+
Mobile	Raw Classifier	71.0	1904.0	39.29	
	Baseline Classifier + FS	32.0	1904.0	46.17	+
	Baseline Classifier + FS + IS	48.0	1497.0	45.4	+
	New Algorithm + FS + IS	48.0	1483.0	45.23	+
ML	Raw Classifier	56.0	4079.0	94.16	
	Baseline Classifier + FS	5.0	4079.0	100.0	+
	Baseline Classifier + FS + IS	10.0	2725.0	99.93	+
	New Algorithm + FS + IS	10.0	2728.0	99.9	+
Mfeat 216	Raw Classifier	216.0	1340.0	93.64	
	Baseline Classifier + FS	98.0	1340.0	94.87	+
	Baseline Classifier + FS + IS	133.0	1085.0	94.46	+
	New Algorithm + FS + IS	132.0	1056.0	94.26	+
Ad	Raw Classifier	1558.0	2186.0	95.88	
	Baseline Classifier + FS	739.0	2186.0	96.96	+
	Baseline Classifier + FS + IS	918.0	1742.0	97.17	+
	New Algorithm + FS + IS	882.0	1722.0	97.2	+
Popularity	Raw Classifier	58.0	26432.0	40.86	
	Baseline Classifier + FS	18.0	26432.0	41.15	+
	Baseline Classifier + FS + IS	36.0	12733.0	42.16	+
	New Algorithm + FS + IS	36.0	13025.0	42.15	+
Mfeat 240	Raw Classifier	240.0	1340.0	97.12	
	Baseline Classifier + FS	129.0	1340.0	96.97	+
	Baseline Classifier + FS + IS	179.0	1026.0	97.1	-
	New Algorithm + FS + IS	175.0	1005.0	97.03	-
UIC	Raw Classifier	561.0	6868.0	96.68	
	Baseline Classifier + FS	278.0	6868.0	98.29	+
	Baseline Classifier + FS + IS	390.0	5488.0	97.64	+
	New Algorithm + FS + IS	385.0	5362.0	97.72	+

The above-described methods performed together increased the number of features reduced on average compared to both algorithms run separately. It also shows improvement in the accuracy in comparison to the feature and instance selection baseline.

The new methods performed together achieved a slightly higher accuracy on the data sets than the feature and instance baseline classifier. It also reduces 5.6 more features on average, which is higher than any of the above-described methods run alone, the new method reduced on average 6.5 fewer instances of data. The new algorithm had the best improvement on the Ad data set; it reduced 36 more features and 20 more instances on average while increasing the accuracy on the test set.

The results achieved by this implementation of the algorithm is due to of both, mutation and crossover, methods making smaller changes to individuals with higher fitness than those done to individuals with lower fitness values. This preserves the fitness of the population while still allowing the algorithm to search the feature and instance space effectively.

In comparison to the feature selection baseline, the new algorithm still does not outperform it in terms of accuracy or feature selection. However, these methods performed together diminished the difference in accuracy between these two algorithms in comparison to the other methods.

4.4.3 Overview

The table below shows the parameters used by the algorithm.

Table 4.6: Parameters

Parameter Name	Parameter Value	FS-Baseline	FS-IS-Baseline	New Algorithm
mutation rate	0.3	yes	yes	yes
crossover rate	0.7	yes	yes	yes
generations	50	yes	yes	yes
population	100	yes	yes	yes
tournament size	3	yes	yes	yes
number of folds	10	yes	yes	yes
crossover independent probability	0.5	yes	no	no
mutation independent probability	0.05	yes	no	no
scoring	accuracy	yes	no	no
max features	number of features present at the beginning	yes	no	no
feature and instance mutation chance	0.7	no	no	yes
flipProb	0.09	no	yes	yes
hardMutationRate	0.2	no	no	yes
hardMutationAmount	4	no	no	yes

4.5 Summary

The results achieved by the novel method show positive results in the reduction of features and instances as well as in terms of accuracy when compared to the feature and instance baseline algorithm.

When compared to the feature selection baseline, the algorithm does not outperform it on average accuracy or feature reduction. This is because the feature selection baseline performs exceptionally well on the Libras data set and methods involving instance reduction do not. This is because the Libras movement data set lacks instances when compared to all other data sets, making it harder to perform instance selection on it without a loss in terms of accuracy. The number of features reduced by the new method is less than the one achieved by the feature selection baseline; this is due to the ample search space that needs to be searched by the feature and instance algorithm in comparison to only feature selection.

The tuning of both methods should be performed to achieve higher accuracy; however, due to the high computational power required to run these experiments and the lack of time, this was not possible.

In general, the novel operators improved the performance of the baseline and show promising results that can contribute well to the final implementation of the algorithm.

Chapter 5

The Proposed Algorithm with New Selection and Local Search Operators

This chapter will present a description of the local search and selection methods used in the algorithm. It will also give the final algorithm containing the new crossover, mutation, selection methods and local search.

5.1 Chapter Goals

The goal of this chapter is to describe the selection and local search methods and analyse how they affect the performance of the algorithm compared to both of the baseline methods. The chapter will describe in detail how the methods are performed and discussed the results achieved. The results of all methods embedded together will also be presented and discussed at the end of this chapter.

5.2 Selection

5.2.1 Description

The selection method used in this implementation is double tournament selection [39] with elitism [40]. Double tournament selection selects K number of individuals from the population according to the specified *tournament size* and from those individuals, it chooses the fittest one to be taken into the next torment. The second tournament selects the individuals in the same manner as the first one; however, it uses the number of features and instances as the selection criteria rather than the fitness, having the individuals with fewer features and instances selected. Elitism allows the best solution so far to be present in the next generation, providing it with a chance of being further developed. The double tournament allows the individuals with the least number of of features and instances and best fitness values to be taken into the next generation.

Pseudo Code:

Algorithm 5 Selection

```
1: function DOUBLETOURNAMENT(currentPop, tournmentSize)
2:     newPop ← []
3:     firstSel ← []
4:     for i in populationSize do
5:         t ← tournmentSize random individuals from currentPop
6:         firstSel gets the fittest out of t
7:     end for
8:     for i in populationSize do
9:         t ← tournmentSize random individuals from firstSel
10:        newPop gets the individial with the smallest number of fs and is out of t
11:    end for
12:    return newPop
13: end function
```

5.2.2 Results

The table below shows the difference in results between the Classifier, the Feature Selection Baseline, the Feature and Instance Selection Baseline and the New Algorithm. The new Algorithm, in this case, is the baseline for feature and instance selection, with the novel selection operator.

Table 5.1: Double Tournament Selection Results Table

data set	Algorithm	Number of Features	Number of Instances	Average Accuracy	significance test on accuray
Libras	Raw Classifier	90.0	240.0	78.33	
	Baseline Classifier + FS	38.0	240.0	82.67	+
	Baseline Classifier + FS + IS	52.0	202.0	76.86	+
	New Algorithm + FS + IS	36.0	193.0	76.22	+
Ary	Raw Classifier	279.0	306.0	60.27	
	Baseline Classifier + FS	131.0	306.0	63.13	+
	Baseline Classifier + FS + IS	145.0	216.0	63.95	+
	New Algorithm + FS + IS	107.0	178.0	65.02	+
Human	Raw Classifier	118.0	588.0	93.49	
	Baseline Classifier + FS	51.0	588.0	94.27	+
	Baseline Classifier + FS + IS	75.0	429.0	94.74	+
	New Algorithm + FS + IS	41.0	370.0	95.18	+
Mobile	Raw Classifier	71.0	1904.0	39.29	
	Baseline Classifier + FS	32.0	1904.0	46.17	+
	Baseline Classifier + FS + IS	48.0	1497.0	45.4	+
	New Algorithm + FS + IS	36.0	1411.0	44.71	+
ML	Raw Classifier	56.0	4079.0	94.16	
	Baseline Classifier + FS	5.0	4079.0	100.0	+
	Baseline Classifier + FS + IS	10.0	2725.0	99.93	+
	New Algorithm + FS + IS	9.0	2453.0	99.94	+
Mfeat 216	Raw Classifier	216.0	1340.0	93.64	
	Baseline Classifier + FS	98.0	1340.0	94.87	+
	Baseline Classifier + FS + IS	133.0	1085.0	94.46	+
	New Algorithm + FS + IS	104.0	1013.0	94.52	+
Ad	Raw Classifier	1558.0	2186.0	95.88	
	Baseline Classifier + FS	739.0	2186.0	96.96	+
	Baseline Classifier + FS + IS	918.0	1742.0	97.17	+
	New Algorithm + FS + IS	717.0	1643.0	97.05	+
Popularity	Raw Classifier	58.0	26432.0	40.86	
	Baseline Classifier + FS	18.0	26432.0	41.15	+
	Baseline Classifier + FS + IS	36.0	12726.0	42.17	+
	New Algorithm + FS + IS	21.0	9520.0	42.5	+
Mfeat 240	Raw Classifier	240.0	1340.0	97.12	
	Baseline Classifier + FS	129.0	1340.0	96.97	+
	Baseline Classifier + FS + IS	179.0	1026.0	97.1	-
	New Algorithm + FS + IS	153.0	938.0	97.02	-
UIC	Raw Classifier	561.0	6868.0	96.68	
	Baseline Classifier + FS	278.0	6868.0	98.29	+
	Baseline Classifier + FS + IS	390.0	5488.0	97.64	+
	New Algorithm + FS + IS	316.0	5092.0	97.74	+

The results suggest that the reduction, of both features and instances, and accuracy achieved by the double tournament selection is significantly betters than the baseline for feature and instance selection.

The accuracy achieved by the implementation of double tournament selection into the algorithm is slightly better than the one produced by the feature and instance baseline. However, the number of features instances reduced is far more significant. The new algorithm reduces 44.6 more features and 433.2 more instances on average than the feature and instance selection baseline. The algorithms best performance was on the Human data set, improving the accuracy while reducing ten more features than the feature selection baseline and 34 more features than the feature and instance selection baseline. It also reduced 59 more instances than the feature and instance selection baseline.

The overall data reduction achieved by the new method is higher than the one produced by the feature selection baseline, due to the number of instances reduced as well as features. The new algorithm does not yet perform better feature selection than the feature selection baseline; however, it narrowed the gap between the two significantly.

Using double tournament selection showed to improve the algorithm in feature and instance reduction as well as in terms of accuracy and will contribute well to the final implementation of the algorithm.

Overview

The algorithm described in this section contains selection function as described above all other functions are as described in Chapter 3.

Table 5.2: Parameters

Parameter Name	Parameter Value	FS-Baseline	FS-IS-Baseline	New Algorithm
mutation rate	0.3	yes	yes	yes
crossover rate	0.7	yes	yes	yes
generations	50	yes	yes	yes
population	100	yes	yes	yes
tournament size	3	yes	yes	yes
number of folds	10	yes	yes	yes
crossover independent probability	0.5	yes	no	no
mutation independent probability	0.05	yes	no	no
scoring	accuracy	yes	no	no
max features	number of features present at the beginning	yes	no	no
flipProb	0.09	no	yes	no

5.3 Local Search

At first, the idea was to implement local search as one of the genetic operators. However, this would give an outstanding advantage to the individuals who had local search performed on, creating biases towards a local solution in the selection process of the algorithm. In this version of the algorithm, local search runs on the best solution after the evolutionary process is completed.

5.3.1 Description

One of the goals of this project was to implement a local search algorithm to run as part of my final algorithm. Local search is a powerful search method used to search for a locally optimal solution. The implementation of this search method varies depending on the problem; however, the main structure of the search is the same. The local search algorithm is composed of three steps:

1. creating a neighbourhood. Neighbouring solutions define the neighbourhood; these solutions depend on the nature of the problem.
2. selecting the best solution. To achieve this, there needs to be a method that measures how good a solution is.
3. repeat the search on the new best solution until the stopping criteria is reached. The stopping criteria can be a certain number of iterations or when the new neighbourhood does not have a better solution than the current one.

The neighbourhood for my algorithm is composed of each individual where one feature or one instance differs from the current best solution. Because the size of the neighbourhood is exponentially large, instead of searching for the best solution in the neighbourhood, the algorithm searches for a better solution than the current one. On each iteration, the algorithm searches through a part of the neighbourhood, declared by the *searchPart* parameter, and if a better solution is found, the solution is used in the next iteration. If a better solution is not found in the current section of the neighbourhood, the algorithms searchers a new part of the same neighbourhood trying to find a better solution, this continues until a certain number of runs is achieved or the whole neighbourhood was searched, and no better solution was found in the entire neighbourhood.

Figure 5.1: Local Search Diagram

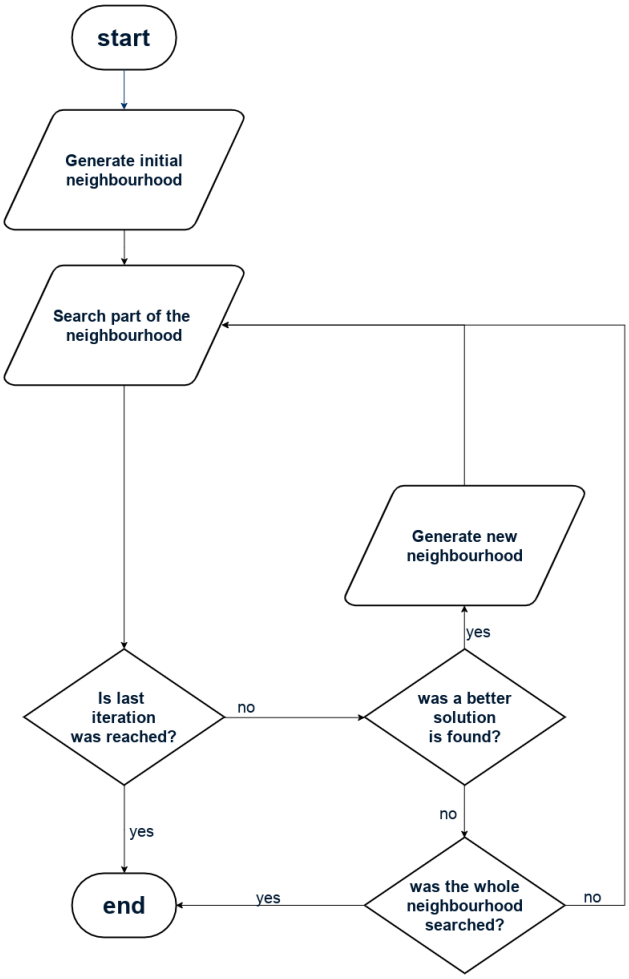


Figure 5.1 above shows the steps of the implemented version local search in a visual way.

Pseudo Code:

Algorithm 6 Local

```
1: function GENERATENEIGHBOURHOOD(individual)
2:     neig ← []
3:     features ← []
4:     instances ← []
5:     for instance in individual do
6:         c ← copy of individual
7:         c[instance] ← 1 − c[instance]
8:         add c to instances
9:     end for
10:    for feature in individual do
11:        c ← copy of individual
12:        c[feature] ← 1 − c[feature]
13:        add c to features
14:    end for
15:    for instance in instances do
16:        for feature in features do
17:            add individual with features and instances to neig
18:        end for
19:    end for
20:    return neig
21: end function
22: function LOCAL SEACH(individual, partSize, iterations)
23:    best ← individual
24:    currentPlace ← 0
25:    for epoch in iterations do
26:        if currentPlace = 0 then
27:            neig ←generateNeighbourhood(best)
28:        end if
29:        for i = currentPlace;i < currentPlace + partSize and i < neighbourhood size do
30:            if neig[i]> best then
31:                best ← neig[i], currentPlace = 0
32:                break
33:            end if
34:            currentPlace = i
35:        end for
36:        if currentPlace >= neighbourhoodsize − 1 then
37:            break
38:        end if
39:    end for
40:    return best
41: end function
```

5.3.2 Results

The table below shows the difference in results between the Classifier, the Feature Selection Baseline, the Feature and Instance Selection Baseline and the New Algorithm. The new Algorithm, in this case, is the baseline for feature and instance selection, with the novel local search operator.

Table 5.3: Local Search Results Table

data set	Algorithm	Number of Features	Number of Instances	Average Accuracy	significance test on accuray
Libras	Raw Classifier	90.0	240.0	78.33	
	Baseline Classifier + FS	38.0	240.0	82.67	+
	Baseline Classifier + FS + IS	52.0	202.0	76.86	+
	New Algorithm + FS + IS	51.0	203.0	76.97	+
Ary	Raw Classifier	279.0	306.0	60.27	
	Baseline Classifier + FS	131.0	306.0	63.13	+
	Baseline Classifier + FS + IS	145.0	216.0	63.95	+
	New Algorithm + FS + IS	139.0	216.0	64.63	+
Human	Raw Classifier	118.0	588.0	93.49	
	Baseline Classifier + FS	51.0	588.0	94.27	+
	Baseline Classifier + FS + IS	75.0	429.0	94.74	+
	New Algorithm + FS + IS	67.0	426.0	94.58	+
Mobile	Raw Classifier	71.0	1904.0	39.29	
	Baseline Classifier + FS	32.0	1904.0	46.17	+
	Baseline Classifier + FS + IS	48.0	1497.0	45.4	+
	New Algorithm + FS + IS	45.0	1490.0	45.47	+
ML	Raw Classifier	56.0	4079.0	94.16	
	Baseline Classifier + FS	5.0	4079.0	100.0	+
	Baseline Classifier + FS + IS	10.0	2725.0	99.93	+
	New Algorithm + FS + IS	9.0	2770.0	99.96	+
Ad	Raw Classifier	1558.0	2186.0	95.88	
	Baseline Classifier + FS	739.0	2186.0	96.96	+
	Baseline Classifier + FS + IS	918.0	1742.0	97.17	+
	New Algorithm + FS + IS	909.0	1767.0	97.14	+
Mfeat 240	Raw Classifier	240.0	1340.0	97.12	
	Baseline Classifier + FS	129.0	1340.0	96.97	+
	Baseline Classifier + FS + IS	179.0	1026.0	97.1	-
	New Algorithm + FS + IS	184.0	1032.0	96.99	-
Popularity	Raw Classifier	58.0	26432.0	40.86	
	Baseline Classifier + FS	18.0	26432.0	41.15	+
	Baseline Classifier + FS + IS	36.0	12733.0	42.16	+
	New Algorithm + FS + IS	34.0	13245.0	42.19	+
Mfeat 216	Raw Classifier	216.0	1340.0	93.64	
	Baseline Classifier + FS	98.0	1340.0	94.87	+
	Baseline Classifier + FS + IS	133.0	1085.0	94.46	+
	New Algorithm + FS + IS	135.0	1084.0	94.21	+
UIC	Raw Classifier	561.0	6868.0	96.68	
	Baseline Classifier + FS	278.0	6868.0	98.29	+
	Baseline Classifier + FS + IS	390.0	5488.0	97.64	+
	New Algorithm + FS + IS	392.0	5525.0	97.59	+

The results suggest that the incorporation of local search into the algorithm improved the accuracy and the number features reduced in comparison to the feature and instance selection baseline.

The algorithm with the implementation of local search performs slightly better on average than the feature and instance selection baseline in terms of accuracy. It also reduces 2.1 more instances on average. The algorithm performed best on the Mobile data set, reducing three more features and seven more instances than the feature and instance selection baseline on average while increasing the accuracy on the test set.

The feature selection baseline outperforms the new algorithm with local search in accuracy and feature selection. The new algorithm has a higher overall data reduction depending on the data set due to the instance selection process.

The implementation of local search shows to have the potential to benefit the algorithm. However, the search space to be covered by local search is extremely large and requires a lot of computational power. The implementation of local search in this algorithm does not perform the search to find the best possible solution in the search space. Nevertheless, it searches for a better solution than the current one and improves the final solution achieved by GA and will contribute to the final implementation of the algorithm.

Overview

The algorithm described in this section contains local search function as described above all other functions are as described in Chapter 3.

Table 5.4: Data Sets

Parameter Name	Parameter Value	FS-Baseline	FS-IS-Baseline	New Algorithm
mutation rate	0.3	yes	yes	yes
crossover rate	0.7	yes	yes	yes
generations	50	yes	yes	yes
population	100	yes	yes	yes
tournament size	3	yes	yes	yes
number of folds	10	yes	yes	yes
crossover independent probability	0.5	yes	no	no
mutation independent probability	0.05	yes	no	no
scoring	accuracy	yes	no	no
max features	number of features present at the beginning	yes	no	no
flipProb	0.09	no	yes	yes
local	True	no	no	yes
partSize	400	no	no	yes

5.4 The Final Algorithm

5.4.1 Description

The final algorithm is a version of the feature and instance selection baseline with all novel methods discussed in this research embedded into it. The algorithm has the initial population generated, as described in Chapter 3. This is because the initialisation method used enforces diversity, feature reduction and instance reduction in a way that allows individuals in the population to start with acceptable fitness values.

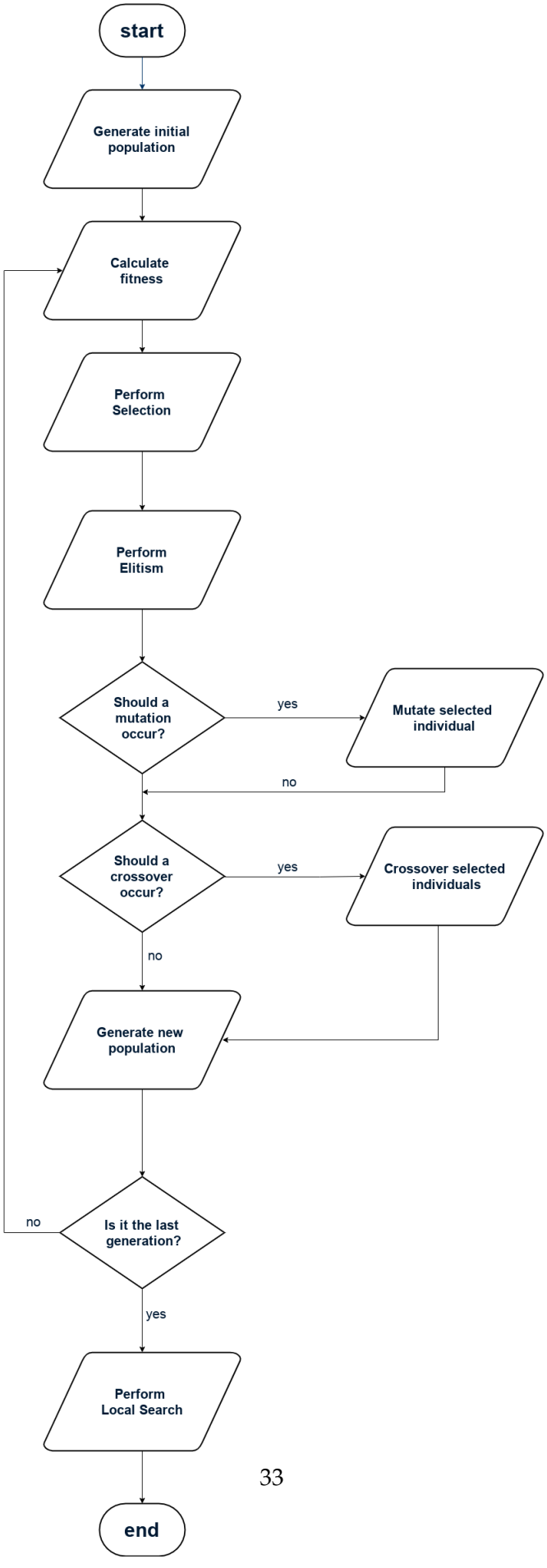
The genetic operators used are as described in Chapter 4. This is because both methods showed to slightly improve average accuracy and feature reduction on the data sets. These methods could not be tuned. However, they still perform well and contribute to the performance of the algorithm.

The selection methods used is double tournament selection first on fitness than on the size of individuals. This selection methods showed to significantly increase the average data reduction of the algorithm while increasing the accuracy compared to the feature and instance selection baseline.

The local search operator is performed after the evolutionary process is finished and found a solution. The best solution achieved by GA could have better neighbouring solutions that could be found by local search. The local search algorithm is restricted due to its sizeable computational power and is not performed to its full potential. However, it showed to have the potential to increase the performance of the algorithm and therefore, it is used in the final solution.

Figure 5.2 below illustrates an overview of the algorithm. It follows a standard evolutionary process with local search performed on the best solution found by GA.

Figure 5.2: Final Algorithm



5.4.2 Results

Table 5.5: Final Algorithm Results Table

data set	Algorithm	Number of Features	Number of Instances	Average Accuracy	significance test on accuray
Ary	Raw Classifier	279.0	306.0	60.27	
	Baseline Classifier + FS	131.0	306.0	63.13	+
	Baseline Classifier + FS + IS	145.0	216.0	63.95	+
	New Algorithm + FS + IS	106.0	174.0	64.63	+
Human	Raw Classifier	118.0	588.0	93.49	
	Baseline Classifier + FS	51.0	588.0	94.27	+
	Baseline Classifier + FS + IS	75.0	429.0	94.74	+
	New Algorithm + FS + IS	38.0	368.0	95.22	+
Libras	Raw Classifier	90.0	240.0	78.33	
	Baseline Classifier + FS	38.0	240.0	82.67	+
	Baseline Classifier + FS + IS	52.0	202.0	76.86	+
	New Algorithm + FS + IS	38.0	190.0	76.35	+
Mobile	Raw Classifier	71.0	1904.0	39.29	
	Baseline Classifier + FS	32.0	1904.0	46.17	+
	Baseline Classifier + FS + IS	48.0	1497.0	45.4	+
	New Algorithm + FS + IS	41.0	1394.0	45.31	+
ML	Raw Classifier	56.0	4079.0	94.16	
	Baseline Classifier + FS	5.0	4079.0	100.0	+
	Baseline Classifier + FS + IS	10.0	2725.0	99.93	+
	New Algorithm + FS + IS	9.0	2472.0	99.97	+
Mfeat 216	Raw Classifier	216.0	1340.0	93.64	
	Baseline Classifier + FS	98.0	1340.0	94.87	+
	Baseline Classifier + FS + IS	133.0	1085.0	94.46	+
	New Algorithm + FS + IS	99.0	1013.0	94.37	+
Ad	Raw Classifier	1558.0	2186.0	95.88	
	Baseline Classifier + FS	739.0	2186.0	96.96	+
	Baseline Classifier + FS + IS	918.0	1742.0	97.17	+
	New Algorithm + FS + IS	708.0	1642.0	96.96	+
Mfeat 240	Raw Classifier	240.0	1340.0	97.12	
	Baseline Classifier + FS	129.0	1340.0	96.97	+
	Baseline Classifier + FS + IS	179.0	1026.0	97.1	-
	New Algorithm + FS + IS	155.0	935.0	96.98	+
UIC	Raw Classifier	561.0	6868.0	96.68	
	Baseline Classifier + FS	278.0	6868.0	98.29	+
	Baseline Classifier + FS + IS	390.0	5488.0	97.64	+
	New Algorithm + FS + IS	278.0	5284.0	97.95	-

Due to the lack of time and computational resources, the final experiment could not be completely finalised. However, for the data sets, where it was possible to complete the runs, the final algorithm showed significant improvement in data reduction.

The results show that the final algorithm outperforms both baselines in terms of data reduction significantly given that the size of the data is represented as *features*instances*. As an example, using the Libras data set, the final solution of the feature selection baseline used

$$\frac{38 * 240}{90 * 240} = \frac{9120}{21600} = 0.42$$

fraction of the initial data, meaning the reduction is of 58 per cent on the overall data size. While for the feature and instance selection baseline, the final solution used

$$\frac{52 * 202}{90 * 240} = \frac{11312}{21600} = 0.52$$

fraction of the data, having 48 per cent of the initial data reduced. And the final algorithm used

$$\frac{38 * 190}{90 * 240} = \frac{7220}{21600} = 0.33$$

fraction of the data, which means 77 per cent of the initial data was reduced.

The final algorithm also outperforms the feature and instance selection baseline in terms of accuracy, and when compared to the feature selection baseline the difference in accuracy is low while reducing a significantly larger amount of the initial data.

The results achieved by the algorithm are due to the performance of all methods together. While the initialisation and selection methods focus on data reduction, the crossover, mutation and local search methods focus on the accuracy. Having the accuracy being lower than the feature selection baseline indicates that these methods require further tuning and improvement. Nevertheless, they performed well and led the algorithm to outperform the feature and instance selection baseline slightly.

Although tuning is required to ensure the best performance of the algorithm, the final version of the algorithm presented in this research outperformed both baselines and is a viable approach to feature and instance selection.

5.4.3 Overview

The parameters used to run this version of the program are stated in the table below:

Table 5.6: Parameters

Parameter Name	Parameter Value	FS-Baseline	FS-IS-Baseline	New Algorithm
mutation rate	0.3	yes	yes	yes
crossover rate	0.7	yes	yes	yes
generations	50	yes	yes	yes
population	100	yes	yes	yes
tournament size	3	yes	yes	yes
number of folds	10	yes	yes	yes
crossover independent probability	0.5	yes	no	no
mutation independent probability	0.05	yes	no	no
scoring	accuracy	yes	no	no
max features	number of features present at the beginning	yes	no	no
feature and instance mutation chance	0.7	no	no	yes
flipProb	0.09	no	yes	yes
hardMutationRate	0.2	no	no	yes
hardMutationAmount	4	no	no	yes
local	True	no	no	yes
partSize	400	no	no	yes

5.5 Summary

Both methods discussed in this chapter have improved the performance of the algorithm. And the final algorithm outperformed both, the feature and instance selection baseline as well as the feature selection baseline, in terms of overall data reduction, feature and instance selection.

Local search is not used to its full potential due to its high computational power in large search spaces, such as the one explored in this project. Instead of searching for the local best solution, it is searching for a local best solution, limiting the increase in accuracy achieved by the algorithm. However, it is still performing well and contributes to the final version of the algorithm.

The high performance of the feature selection baseline in the Libras Movement data set leads to higher average accuracy in comparison to other methods. However, the implementation of double tournament selection helps the new algorithm to perform almost at the same level as the baseline in terms of feature selection. In terms of overall data reduction, the implementation of double tournament outperforms both baselines significantly, and the implementation with the methods previously described improved further the reduction of data achieved by the algorithm.

The final algorithm performs very well in terms of feature, instance and overall data reduction. The primary goal of this research was to achieve this result without having the accuracy of the classifier suffer from the reduction of the data. This goal was completed successfully, as the algorithm reduced the size of the data while increasing the accuracy in comparison to the raw classifier. This result was possible because of the selection and initialisation methods, encouraging data reduction without penalising the accuracy achieved by the machine learning model. It was also possible due to the genetic operators' tendencies to not punish reasonable solutions in terms of accuracy.

The accuracy achieved by the algorithm still needs to be improved. This could be achieved by fine-tuning the methods described throughout this project or creating novel ways to enforce accuracy increase for the algorithm.

Chapter 6

Conclusions

All main objectives of this project were achieved, except for objective 4 were the results were obtained in a different way than initially planned. The algorithm performs well and increases the accuracy of the raw classifier by reducing features and instances.

The algorithm presented in this project outperforms both baseline algorithms in terms of overall data reduction, instance reduction and feature reduction. In terms of accuracy increase, the feature and instance selection baseline was outperformed. The feature selection baseline was not exceeded in terms of accuracy. However, the difference in performance was low, in both feature reduction and average accuracy, and dependent on the data set it ran on. This is because data sets with a small number of instances tend to be fragile to instance selection, as each instance individually has a more significant impact on the outcome of the classification in comparison to more substantial data set. Another reason for the better performance of the feature selection baseline is due to the smaller search space it covers. While the search space is still ample, it is exponentially lower than the one that has to be covered by feature and instance selection.

The final algorithm developed has proven to be a viable solution for the tasks of feature and instance selection performed simultaneously and reduced the size of the original data set significantly.

6.1 Main conclusions

The project had four objectives to be achieved, as stated in Chapter 1.

- **Create a simple feature and instance selection:** This objective was achieved early on the project and then used as one of the baseline methods to compare the improved algorithm. The algorithm has a simple mutation and cross over function and used tournament selection as the selection function. This algorithm achieved a reasonable accuracy and also outperformed the feature selection baseline classifier in terms of average data reduction.

- **Develop well-designed crossover and mutation operators:** This objective was achieved halfway throughout the project; however, they were slightly modified and improved later on. Both these operators take into account the current fitness of the individuals on which the operation will be performed. This prevents individuals that are converging towards a good solution from being penalised by the operators, while stills allowing the algorithm to jump out of local solutions and allowing individuals with a lower fitness to explore several solutions.

- **Develop a fitness function that encapsulates features, instances and accuracy:** The desired outcome of this objective was achieved in a different way than initially planned. The original goal was meant to enforce feature and instance reduction without decreasing the accuracy. Encapsulating this into the fitness function turned out to be fragile and hard to tune. Therefore, instead of having the fitness function enforce reduction and accuracy, it only enforces accuracy while the selection function enforces both, reduction and accuracy.
- **Implement Local Search:** This objective was achieved towards the end of the project. The running of this method was more time consuming than expected; therefore, some restrictions had to be implemented to ensure that it ran at a reasonable time. However, these restrictions affected the performance of the local search operator, having it not find the best possible solution.

6.2 Future Work

This project explored different methods to approach the task of feature and instance selection and achieved promising results. However, there is still a task worth further research. Finding ways to search the space presented by this task in an efficient manner is an essential task as the amount of data grows, and more data cleaning techniques are required. The methods presented in this project also have the potential to be refined and further optimised to achieve better results in the performance of this task.

Having the local search as a genetic operator without compromising the evolution process is also a topic worth researching and analysing, as it has the potential to improve the performance of GA for this task.

Another idea would be to run local search on a few of the best solutions achieved by GA and use the one that produced the highest accuracy as the final solution. This could lead to a better overall solution, as the best GA solution is not necessarily the best solution with the best neighbouring solutions.

The proposed algorithm achieves higher accuracy than the raw classifier; however, there is still room for improve met on its accuracy performance. Novel methods that assure higher accuracy should be developed and tested to ensure that the results are optimal.

To summarise, this task still requires a lot of research and focus. The current methods to search the space are computationally expensive and although the results achieved by this project were good. There is still room for improvement and new ways to approach this task.

Bibliography

[1] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, p. 67, 2016.

[2] G. Press, "A very short history of big data," Dec 2013.

[3] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, pp. 606–626, Aug 2016.

[4] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017.

[5] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[6] C.-F. Tsai, W. Eberle, and C.-Y. Chu, "Genetic algorithms in feature and instance selection," *Know.-Based Syst.*, vol. 39, pp. 240–247, Feb. 2013.

[7] I. A. Contributor, "The rise in computing power: Why ubiquitous artificial intelligence is now a reality," May 2019.

[8] M. Sidana and M. Sidana, "Types of classification algorithms in machine learning," Feb 2017.

[9] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," 2009.

[10] M. R. Smith and T. Martinez, "Improving classification accuracy by identifying and removing instances that should be misclassified," *The 2011 International Joint Conference on Neural Networks*, 2011.

[11] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study," *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 561–575, Dec 2003.

[12] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How easy is local search?," *Journal of computer and system sciences*, vol. 37, no. 1, pp. 79–100, 1988.

[13] M. K. Md., S. Md., and M. Kazuyuki, "A new local search based hybrid genetic algorithm for feature selection author links open overlay panel,"

[14] J. Wang, B. Xue, X. Gao, and M. Zhang, *A Differential Evolution Approach to Feature Selection and Instance Selection*, pp. 588–602. Springer International Publishing, 2016.

[15] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, and H. De Garis, "An overview of evolutionary computation," in *European Conference on Machine Learning*, pp. 442–459, Springer, 1993.

[16] M. KUBAT, *INTRODUCTION TO MACHINE LEARNING*. SPRINGER INTERNATIONAL PU, 2018.

[17] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[18] M. J. Martin-Bautista and M. . Vila, "A survey of genetic feature selection in mining issues," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 2, pp. 1314–1321 Vol. 2, July 1999.

[19] S. Davies and S. Russell, "Np-completeness of searches for smallest possible feature sets," *NP-Completeness of Searches for Smallest Possible Feature Sets*.

[20] mathworks, "What is the genetic algorithm?."

[21] T. E. o. E. Britannica, "Natural selection," Dec 2018.

[22] M. Mitchell, "Genetic algorithms: An overview," *Complexity*, vol. 1, no. 1, pp. 31–39, 1995.

[23] "https://rednuht.org/genetic_cars_2/."

[24] C. Segura, A. H. Aguirre, S. I. V. Peña, and S. B. Rionda, *The Importance of Proper Diversity Management in Evolutionary Algorithms for Combinatorial Optimization*, pp. 121–148. Cham: Springer International Publishing, 2017.

[25] J. Shapiro, "Genetic algorithms in machine learning," in *Advanced Course on Artificial Intelligence*, pp. 146–168, Springer, 1999.

[26] S. M. Shorman and S. A. Pitchay, "Significance of parameters in genetic algorithm, the strengths, its limitations and challenges in image recovery," *ARPN Journal of Engineering and Applied Sciences*.

[27] Tutorialspoint.com, "Genetic algorithms crossover."

[28] M. D. Vose, "A closer look at mutation in genetic algorithms," *Annals of Mathematics and Artificial Intelligence*, vol. 10, no. 4, pp. 423–434, 1994.

[29] J. Kleinberg and T. Eva, *12. LOCAL SEARCH*. Pearson, 2014.

[30] D. Boughaci and A. A.-s. Alkhawaldeh, "Three local search-based methods for feature selection in credit scoring," *Vietnam Journal of Computer Science*, vol. 5, pp. 107–121, May 2018.

[31] N. Jankowski and M. Grochowski, "Comparison of instances seletion algorithms i. algorithms survey," *Lecture Notes in Computer Science Artificial Intelligence and Soft Computing - ICAISC 2004*, p. 598603, 2004.

[32] K. A. De Jong and W. M. Spears, "Using genetic algorithms to solve np-complete problems,"

[33] F. Gomez and A. Quesada, "Genetic algorithms for feature selection."

[34] M. Calzolari, "sklearn-genetic," Apr 2019.

[35] D. A. Umbarkar and P. Sheth, "Crossover operators in genetic algorithms: A review," *ICTACT Journal on Soft Computing (Volume: 6 , Issue: 1)*, vol. 6, 10 2015.

[36] T. Blickle and L. Thiele, "A mathematical analysis of tournament selection.," in *ICGA*, vol. 95, pp. 9–15, Citeseer, 1995.

[37] "https://archive.ics.uci.edu/ml/index.php."

[38] "https://www.kaggle.com/."

[39] S. Bleuler, M. Brack, L. Thiele, and E. Zitzler, "Multiobjective genetic programming: Reducing bloat using spea2," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, vol. 1, pp. 536–543, IEEE, 2001.

[40] C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 367–385, 2003.