MULTIOBJECTIVE EVOLUTIONARY FEATURE SUBSET SELECTION ALGORITHM FOR BINARY CLASSIFICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FİRDEVSİ AYÇA DENİZ-KIZILÖZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2016

Approval of the thesis:

MULTIOBJECTIVE EVOLUTIONARY FEATURE SUBSET SELECTION ALGORITHM FOR BINARY CLASSIFICATION

submitted by **FİRDEVSİ AYÇA DENİZ-KIZILÖZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Prof. Dr. Ahmet Coşar
Supervisor, **Computer Engineering Department, METU** _____

Assist. Prof. Dr. Tansel Dökeroğlu
Co-supervisor, **Computer Engineering Department, UTAA** _____

Examining Committee Members:

Prof. Dr. Faruk Polat
Computer Engineering Department, METU _____

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Prof. Dr. Uğur Güdükbay
Computer Engineering Department, Bilkent University _____

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Assist. Prof. Dr. Yusuf Sahillioğlu
Computer Engineering Department, METU _____

Date: **25.08.2016**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: FİRDEVSİ AYÇA DENİZ-KIZILÖZ

Signature :

ABSTRACT

MULTIOBJECTIVE EVOLUTIONARY FEATURE SUBSET SELECTION ALGORITHM FOR BINARY CLASSIFICATION

Deniz-Kızılöz, Firdevsi Ayça
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. Ahmet Coşar
Co-Supervisor : Assist. Prof. Dr. Tansel Dökeroğlu

August 2016, 76 pages

This thesis investigates the performance of multiobjective feature subset selection (FSS) algorithms combined with the state-of-the-art machine learning techniques for binary classification problem. Recent studies try to improve the accuracy of classification by including all of the features in the dataset, neglecting to determine the best performing subset of features. However, for some problems, the number of features may reach thousands, which will cause too much computation power to be consumed during the feature evaluation and classification phases, also possibly reducing the accuracy of the results. Therefore, selecting the minimum number of features while preserving the accuracy of the results at a high level becomes an important issue for achieving fast and accurate binary classification. The multiobjective algorithms implemented in this thesis include two phases, selecting feature subsets and applying supervised/unsupervised machine learning techniques to these selected subsets. For the FSS part of the algorithms, first a brute-force approach is implemented. Since exhaustively investigating all of the feature subsets is unfeasible when

the number of features is larger than 20, secondly, a greedy algorithm implemented to find good-enough feature subsets. Finally, in order to select the most appropriate feature subsets intelligently, a genetic algorithm is proposed at the FSS part of the algorithms. Crossover and mutation operators are used to improve a population of individuals (each representing a selected feature subset) and obtain (near-)optimal solutions through generations. At the second phase of the algorithms, the performance of the selected feature subsets is evaluated by using five different machine learning techniques: Logistic Regression, Support Vector Machines, Extreme Learning Machine, K-means, and Affinity Propagation. The best performing multiobjective evolutionary algorithm is selected after comprehensive experiments and compared with the state-of-the-art algorithms in literature; Particle Swarm Optimization, Greedy Search, Tabu Search, and Scatter Search. 11 different datasets, mostly obtained from the well-known machine learning data repository of University of California UCI Machine Learning Repository, are used for the performance evaluation of the implemented algorithms. Experimental results show that the classification accuracy increases significantly with the most suitable subset of features and also execution time reduces greatly after applying proposed algorithm on the datasets.

Keywords: Multiobjective Feature Selection, Evolutionary Algorithm, Binary Classification, Supervised/Unsupervised Machine Learning

ÖZ

İKİLİ SINIFLANDIRMA İÇİN ÇOK AMAÇLI EVRİMSEL ÖZNİTELİK ALT KÜMESİ SEÇİMİ ALGORİTMASI

Deniz-Kızılöz, Firdevsi Ayça
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Ahmet Coşar
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Tansel Dökeroğlu

Ağustos 2016 , 76 sayfa

Bu çalışmada, ikili sınıflandırmada öznitelik alt küme seçimi problemi üzerine en yeni makine öğrenme algoritmalarıyla birlikte çok amaçlı algoritmaların performansı araştırılmıştır. Yakın zamandaki çalışmalar, en uygun özniteliklerin bulunduğu veri kümesinin ne olduğuna aldırmadan özniteliklerin tamamını kullanmakta ve ikili sınıflandırma problemlerinde doğruluk oranını bu şekilde artırmaya çalışmaktadır. Fakat bazı problemler için öznitelik sayısı binlere kadar ulaştığından karar verme sürecinde hesaplama için çok fazla güç harcanabilmekte ve sonucun doğruluğu azalırken problemi sınıflandırmak zorlaşabilmektedir. Bu nedenle, doğru ikili sınıflandırma sonuçlarına hızlı ulaşabilmek için sonuçların doğruluk oranlarını korurken öznitelik sayısını azaltmak oldukça önemlidir. Bu çalışmada geliştirilen çok amaçlı algoritmalar iki aşamadan oluşmaktadır. Bu aşamalar öznitelik alt kümesini seçmek ve sınıflandırma işlemi için bu küme üzerinde makine öğrenme tekniklerini uygulamaktır. Öznitelik kümelerini seçebilmek için geliştirilen ilk yöntem kaba kuvvet yaklaşımı

olmuştur. Kaba kuvvet yaklaşımı bütün öznitelik kümelerini incelemeyi gerektirir. Ancak 20'den fazla öznitelik bulunduğu durumlarda çözüme ulaşmak uygulanabilir bir işlem olmadığından ikinci yöntem olarak bir açgözlü algoritma geliştirilmiş ve yeterince iyi olan öznitelik alt kümeleri elde edilmeye çalışılmıştır. Son olarak, öznitelik alt kümesi seçimi işlemini daha akıllıca yapabilmek için bu aşamada bir evrimsel algoritma önerilmiştir. Çaprazlama ve mutasyon operatörleri seçilen bireylerden (öznitelik alt kümeleri) oluşan popülasyonu nesiller boyunca geliştirmekte ve ideale yakın çözümler elde etmektedir. Geliştirilen algoritmaların ikinci bölümünde, seçilen öznitelik kümelerinin performansı şu makine öğrenme algoritmaları ile hesaplanmıştır: Lojistik Regresyon, Destek Vektör Makineleri, Aşırı Öğrenme Makinesi, K-ortalama ve Benzeşim Yayılımı. En iyi performans gösteren çok amaçlı evrimsel algoritma seçilerek literatürdeki Parçacık Sürüsü Optimizasyonu, Aç Gözlü Arama, Tabu Arama ve Dağılım Arama algoritmaları ile karşılaştırılmıştır. Birçoğu tanınmış Kaliforniya Üniversitesi UCI Makine Öğrenme Deposu'ndan temin edilen 11 farklı veri kümesi, geliştirilen algoritmaların performans değerlendirmelerini yapmak için kullanılmıştır. Elde edilen sonuçlar göstermektedir ki, en uygun öznitelik alt kümesi seçimi ile sınıflandırma doğruluk oranı önemli ölçüde artmakta ve önerilen algoritma veri kümelerine uygulandığında çalışma zamanı oldukça azalmaktadır.

Anahtar Kelimeler: Çok Amaçlı Öznitelik Seçimi, Evrimsel Algoritma, İkili Sınıflandırma, Gözetimli/Gözetimsiz Makine Öğrenmesi

ACKNOWLEDGMENTS

First of all, I want to thank to my supervisor Prof. Dr. Ahmet Coşar for his guidance and imparting his knowledge and expertise in this study. I am also truly grateful to my co-supervisor, Assist. Prof. Dr. Tansel Dökeroğlu, for being always there to listen, read and comment on countless revisions of this manuscript.

Sincere thanks to all committee members, Prof. Dr. Faruk Polat, Prof. Dr. Uğur Güdükbay, Assoc. Prof. Dr. Pınar Karagöz, and Assist. Prof. Dr. Yusuf Sahillioğlu, for spending their valuable time and sharing their comments about this study.

Special thanks to my best friends, Gözde Kutayer and Seda Diker. With their support, I was able to stay sane through tough times and focus on my study. I deeply value their friendship and appreciate their belief in me.

I would like to thank TED University for giving me the chance to work as a research assistant and providing me their resources throughout this study.

I would also like to thank TÜBİTAK - BİDEB for the financial support that I received within the 2210-A bursary program throughout my M.Sc. study.

I must express my deepest gratitude to my parents for supporting me in every aspect for all my life. My mother, Emine Deniz, provided me with unconditional love and continuous encouragement. My father and my mentor, Hamdi Deniz, motivated me and never let me doubt myself. He is a constant source of inspiration and his advices made me who I am.

Last but not least, I am eternally greateful to my spouse and my source of happiness, Hakan Ezgi Kızılöz. He was always there to cheer me up since the day we met. He constantly supported me and shared his knowledge with me through the process of researching and writing this thesis. This accomplishment would not have been possible without him.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

AP	Affinity Propagation
ELM	Extreme Learning Machine
EA	Exhaustive Approach
FSS	Feature Subset Selection
GA	Genetic Approach
GR	Greedy Approach
IG	Information Gain
LR	Logistic Regression
NSGA-II	Non-dominated Sorting Genetic Algorithm II
SVM	Support Vector Machines

CHAPTER 1

INTRODUCTION

The process of making selection over many possible options is called decision-making. People have to make decisions in their daily activities every day, such as deciding whether or not carrying an umbrella on a cloudy day. In computer applications, algorithmic decision-making techniques are used for this process. For example, specialized advertisements targeting individual customers may be created by extracting the available customers' profile information.

An effective decision-making relies on the quality of information [32]. In general, previous experience/knowledge is used in making a decision. The amount of historical and statistical data increases massively as data collection technology and abilities improve. The speed of data accumulation is beyond manual processing capacity. In 2012, the amount of data generated in one day was around 2.5 exabytes (2.5 million terabytes), and this volume doubles in every 40 months [29]. As the amount of available information increases, it becomes much harder to extract meaningful information in terms of computational power and requires some advanced techniques like data preprocessing, data mining and/or machine learning. As a result, the use of historical data in the decision-making activities may be negatively affected due to the inability to properly filter and process this information.

Data mining, also known as knowledge discovery, identifies the existing patterns in the data which might help predict future behaviors. It is widely used in commercial business intelligence to provide insight and likely trends/expectations to decision-making experts. In addition to data mining techniques, machine learning techniques are also widely used in modern decision making processes. While data mining manip-

ulates data by filtering, formatting, etc., machine learning techniques help to build a smart model using past data and experience [1]. Machine learning techniques provide tools that can analyze large amounts of data in a limited time.

Given a set of items with their corresponding classes as (training) input; classification is a type of learning which can be done in a supervised or unsupervised way. It aims to identify which class a new item would fit into with respect to the similarity of its attributes (features) with the instances of the existing/known classes. For decision-making tasks, classification based on available information is a significant tool. For example, predicting, diagnosing or recognizing patterns are the frequently faced decision-making tasks. If a classification problem has two mutually exclusive classes, then this problem is called binary classification. When constructing a classifier, some of the available input class instances are used as the training set, so that the classifier can learn the patterns in the dataset which helps decide the class it belongs. Every item in the training set has its own set of attributes and these attributes are investigated to help determine how and which of them affect the determination of the item's class [9].

Researchers agree that data mining tools perform more effectively when data preprocessing is applied on the dataset before using it for mining [26]. One of the most commonly used data preprocessing techniques is feature selection. There are many cases that some of the attributes of a dataset can be irrelevant or redundant in making a certain decision [4]. These kinds of features do not affect or add anything to the objective concept and they have no contribution to the representation of the problem domain [21]. For example, the eye color of a person has no effect on his/her gender type. In real-world problems, irrelevant features are generally not known apriori and finding the optimal set of features is intractable [25]. Feature selection is the process of reducing the number of features by identifying these unnecessary features while keeping an optimization criterion fulfilled which minimizes any loss of information. Feature selection affects the result in two important ways [13]. First, machine learning algorithms run faster since the amount of data decreases. And second, accuracy results improve since the noisy data is removed.

As database sizes get bigger, the need for machine learning techniques increases.

Consequently, feature selection becomes indispensable to extract meaningful information from these huge databases. Feature selection algorithms are widely applied in various real-world problems such as text categorization [48], recommendation systems [38], gene analysis on microarray data [44], big data mining [16], and customer relationship management [31]. For example, a text categorization domain may contain a vocabulary with a size of hundreds of thousands. It is almost impossible to run a learning algorithm on that domain before selecting the most valuable features.

The reason for a multiobjective formulation is because there are two objectives of the feature selection problem. These objectives are minimizing the number of features and maximizing the accuracy. Therefore there might be more than one solution which serve to both objectives and cannot dominate each other. For example, a solution having an accuracy value of 0.85 by selecting 5 features and another solution having an accuracy value of 0.75 by selecting 3 features cannot overwhelm each other. Because one of them (former) provides a better solution for one objective (higher accuracy) and the other one (latter) for the other objective (lower number of features). Figure 1.1 shows sample solutions of a hypothetical problem set. The ideal point is $(1, 1)$ which has the value for maximum accuracy and minimum number of features. It is clear that solutions closest to the ideal point are more desirable. The solutions that

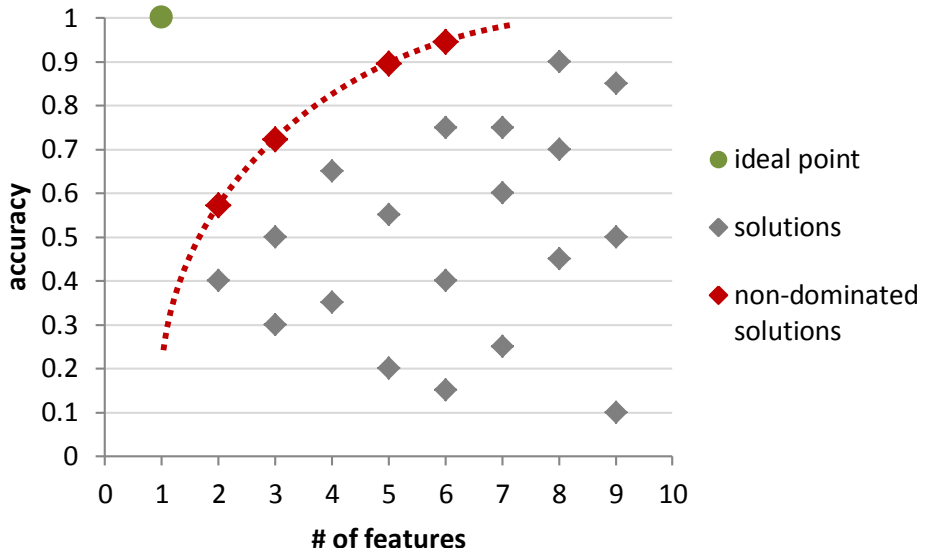


Figure 1.1: Non-dominated solutions fitting to a pareto curve in a multiobjective problem.

are not dominated by any other solution constitute a set of solutions and they fit to a pareto curve. In real world problems, however, the solutions may not form a curve, yet they will lie on a curve like shape.

The motivation of this thesis is providing a detailed analysis on three different feature selection strategies; namely, exhaustive, greedy and genetic. Furthermore, classification accuracies of these strategies are evaluated using five different machine learning algorithms: three supervised (Logistic Regression, Support Vector Machines, Extreme Learning Machine), and two unsupervised (K-means, Affinity Propagation). The most distinctive point of this study is giving the results of the mentioned techniques in a multiobjective way which is not discussed extensively in literature.

Experimental results of this thesis show that feature selection reduces the computational time and increases the classification accuracy noticeably. Exhaustive algorithm, one of the feature selection algorithms, searches over all possible combinations which makes it unfeasible to run in real-world examples. Greedy algorithm, on the other hand, gives better results than exhaustive one by decreasing the computational time tremendously. The last feature selection algorithm, genetic algorithm, has the best time and accuracy results on average. In Section 5.4.2, it is shown that how a randomly picked feature subset can be improved and approximated to the ideal point in terms of accuracy and feature size by employing evolutionary approach.

The rest of this thesis is organized as follows. In Chapter 2, related studies about the feature subset selection for the binary classification problem are given. In Chapter 3, feature subset selection problem is defined in detail and feature selection algorithms used in this study are explained. Applied machine learning techniques are introduced in Chapter 4. The setup of the experimental environment, obtained results, and performance comparison of the proposed algorithm and the state-of-the-art algorithms in literature are presented in Chapter 5. Concluding remarks along with possible future works are provided in the last chapter, Chapter 6. Finally, in Appendix A, two sample datasets used in this study are given with their attributes to visualize preprocessing and also to show how the classification result is affected after unnecessary features are detected.

CHAPTER 2

BACKGROUND

In this chapter, information about the history of feature subset selection (FSS), multiobjective studies about FSS, and the state-of-the-art methods for solving FSS for binary classification problem are given.

Dash et al. [9] conduct a survey on feature selection methods. The survey first gives a definition of feature selection by discussing previous definitions by many other authors. Afterwards, the procedure of a typical feature selection is explained with four steps: generation, evaluation, criteria to stop, and verification part. Consequently, different combinations of feature selection methods are revealed by splitting the generation procedures and the evaluation functions into categories. Some representative methods are analyzed to see the positive and negative parts of the methods. It is stated that the guideline given in the paper can be useful when choosing a particular method for the problem.

Narendra et al. [30] implement branch and bound algorithm to select the best feature subset. They report that, evaluating only 6000 subsets would yield the best 12 feature set among 24, where it would require thousands of evaluations in case of an exhaustive search. A strong drawback of the study is the need of a monotonic criterion function, which means adding new features to a subset does not decrease the value. The monotonic condition generally remains unsatisfied.

Kohavi et al. [25] investigate the performance of wrappers for FSS. After giving a comprehensive definition of the problem, they share their proposed methods and test results. They implement two greedy algorithms for feature selection part of the

study, *Hill-Climbing* and *Best-First Search*, and two classifiers for the testing part of the study, *Decision Tree* and *Naive-Bayes*. They use benchmark datasets to compare their results against same classifiers without the FSS process in means of accuracy and CPU times. They also compare the results against *Relief*, a filter based FSS approach. According to test results, both algorithms improve their results on the average. Yang et al. [47] propose the use of genetic algorithm for finding a suitable subset and a neural network algorithm, *DistAI*, for the classification process. They run tests on benchmark datasets and show that genetic algorithm combined with *DistAI* improves the results obtained from *DistAI* using all features (without subset selection). Finally, they compare their results against results from different studies, showing that their proposal generally achieves better. Inza et al. [20] give a state-of-the-art description of FSS problem and present *Feature Subset Selection by Estimation of Bayesian Network Algorithm*. Since it is derived from *Estimation of Distribution Algorithm*, it is an evolutionary and randomized search algorithm which can be useful when knowledge about domain is limited. In experiments, *Naive-Bayes* and *ID3* learning algorithms are used as classifiers. They compare both classifiers with and without FSS in terms of CPU time and accuracy. Results show that FSS process does not change accuracy significantly but decreases the CPU execution times dramatically.

An efficient filter based feature selection algorithm is introduced in [49] to effectively handle high dimensional data. *Fast Correlation-Based Filter* identifies relevant features without pairwise analysis. They compare it with three well-known feature selection algorithms; *ReliefF*, *ConsSF*, and *CorrSF*. It is presented that the proposed algorithm runs faster than the other algorithms, and it increases the accuracy values for most of the datasets. Cervante et al. [6] combine *Particle Swarm Optimization (PSO)* with two information measures, namely *Mutual Information* and *Entropy*. Using each measure, they evaluate relevance and redundancy of the selected subsets, which are then used in fitness evaluation. They use *Decision Tree*s for classification. Analysis on benchmark datasets show that minimizing mutual information usually selects a smaller feature subset whereas maximizing group entropy achieves higher accuracy. Similarly for feature selection problem, Unler et al. [42] propose basically a *PSO* algorithm but with some modification. Features are chosen according to two characteristics: their independent likelihood and their predictive contribution to the

feature subset already selected. It is stated that the algorithm developed for binary classification problems and *Logistic Regression* model is used as classifier. Experiments show that performance of this adaptive feature selection algorithm is better than *Tabu* and *Scatter Search* algorithms.

Lopez et al. [28] propose a *Parallel Scatter Search* method for the feature selection problem. They use greedy approach to generate new feature subsets as solutions. It is presented that this parallelized algorithm performs better than *Sequential Scatter Search*. Pacheco et al. [33] propose a *Tabu Search* method for the problem of feature selection for *Logistic Regression* models. This new method is compared with the classic ones. The results supported by statistical tests show that the new method obtains better set of solutions than the others. However, it requires more computation time.

Khan et al. [23] present a multiobjective evolutionary algorithm for the feature selection problem. They use *NSGA-II*, one of the latest multiobjective genetic algorithm, in their study. They apply the algorithm on 4 datasets obtained from UCI database. The experiment results show that *NSGA-II* is a promising algorithm for the FSS problem. Unlike this thesis study, they use *ID3* as classifier and maximize both first class and second class accuracy values in their objectives. Sikdar et al. [39] propose a *Multiobjective Differential Evolution (MODE)* for feature selection and classifier altogether. Their objectives are set as minimizing the number of features, and maximizing the f-measure value. They use three biomedical datasets: *GENIA*, *GENETAG* and *AIMed*. F-measure values for these datasets are evaluated as 76.75%, 94.15% and 91.91%, respectively, achieving a similar score with the top score of other studies for *GENIA* and achieving higher than the top score for *GENETAG*. A recent study by Xue et al. [46] introduce multiobjective approach into *PSO* for the feature selection problem. In their study, they describe two *PSO* based algorithms and compare them against two existing single objective *PSO* algorithms. They also compare their proposal algorithms against three existing multiobjective evolutionary algorithms. According to the results, one of the proposed algorithms performs better than single objective methods and achieves comparable results against multiobjective algorithms; whereas, their other proposal performs better than all mentioned algorithms, including their first one.

Vafaie et al. [43] conduct a comparison between two feature selection methods, one of them is based on greedy search, and the other one is a genetic algorithm. They use classification performance as the evaluation function. Using real-world problems, they present that greedy search is more efficient on small datasets, while genetic algorithm results are more reliable without sacrificing too much computational effort. Talbi et al. [41] compare two population based metaheuristics; genetic algorithm and a new version of *PSO* which they name as *Geometric PSO (GPSO)*. *SVM* is used as classifier for both algorithms. First they prove that the *GPSO* achieves competitive results in feature selection. Afterwards they compare *GPSO* and genetic algorithm by running the algorithms on some popular high dimensional cancer datasets. Experiment results show that genetic algorithm is generally better at selecting features. However, *GPSO* performs better in terms of average accuracy of classification. A study by Yusta [50] proposes three metaheuristic strategies for the feature selection problem. These strategies involve *GRASP*, *Memetic Algorithm* and *Tabu Search*. After finding a suitable subset, *K-nearest Neighbour* algorithm is used for classification. The author compares the proposed methods using 6 benchmark datasets against three existing methods namely *Genetic Algorithms*, *Sequential Forward Floating Selection* and *Sequential Backward Floating Selection*. Comparison shows that *GRASP* and *Tabu Search* outperform every other method; whereas *Memetic Algorithm* shows better results than the remaining without significant difference.

Sarafrazi et al. [37] combine *Gravitational Search Algorithm (GSA)* with *SVM* for the classification of binary problems. They utilize *GSA* for two purposes: finding an optimal feature subset and optimizing SVM model parameters. *SVM*, on the other hand, is used for classifying the data using the selected subset. They report results obtained from 8 benchmark datasets and compare their proposed algorithm against results from other studies. The authors state that their proposal achieves similar results compared to other studies, if not better. Huang et al. [18] propose a genetic algorithm based strategy to optimize the process of feature selection and setting SVM parameters. This new approach is tested on 11 known real-world datasets and compared with the *Grid Algorithm* which is mostly used for parameter searching. The results show that this approach significantly affects the classification accuracy in a positive way. Kazemian et al. [22] compare both supervised and unsupervised types of machine

learning models for detecting malicious websites. They download all these websites with a web crawler and convert them into feature vectors. Finally, they apply different machine learning methods for classification purpose and compare the results. They state that, *Radial Basis Function Support Vector Machine* classifier reaches 97% accuracy, outperforming other supervised methods such as *K-nearest Neighbour*, *Linear Support Vector Machines* and *Naive-Bayes*. For unsupervised methods, all tested methods could separate malicious websites from safe websites perfectly with a silhouette coefficient of 0.963, 0.877 and 0.877 for *Affinity Propagation*, *K-means* and *Mini Batch K-means*, respectively.

Finally, a very recent survey by Xue et al. [45] gives a comprehensive analysis on the FSS problem. They investigate different evolutionary methods in literature by discussing how and which evaluation techniques are applied and their number of objectives. The study presents challenges and contributions of various FSS algorithms. Moreover, the authors state that feature selection improves classification performance by reducing the dimension of the data.

There is a wide range of studies about FSS for binary classification problem. Nevertheless, there is no certain results and it is still an open problem. Although there are promising experimental studies about this problem, an extensive analysis on combining different feature selection algorithms with different classification algorithms does not exist in literature. The aim of this thesis is filling this gap by giving an extensive analysis on three feature selection algorithms evaluated by five classification algorithms. Moreover, the most distinctive point of this study is presenting the experiment results and comparing them in a multiobjective way.

CHAPTER 3

FEATURE SUBSET SELECTION

In this chapter, feature subset selection (FSS) is described in detail, a formal definition of FSS is given, and finally, feature selection algorithms used in this study are introduced.

FSS is a process of selecting a subset of features from the original feature set. FSS process prevents the complicated calculations by shrinking the dataset which provides the classifiers run much faster. In literature, there are conceptually different definitions for FSS [9]. While preserving the structure of the original dataset, some of them care much about minimizing the size of subset whereas some other aim to improve prediction accuracy.

The most simple and comprehensive definition can be as follows. Feature subset selection is a process which aims to remove irrelevant or redundant features and create an effective subset that represents the dataset most informatively with minimum number of features. Therefore, there are two important criteria for FSS:

1. finding minimally sized feature subset that preserves the original structure,
2. not decreasing the classification accuracy remarkably, increasing if possible.

FSS is known as an NP-hard problem, since extracting the optimal feature subset is a challenging process and there is no exact way of solving it [27]. A typical FSS consists of four steps: generating feature subsets, evaluating these subsets, deciding on termination condition of the algorithm and validating the results. In the first step, candidate features are selected by a search strategy and subsets are generated. In the

second step, these subsets are evaluated and compared against each other in terms of the quality of the subset. First two steps are repeated until the third step, termination condition, is fulfilled. The final step is to validate the chosen subset by different tests or prior knowledge whether it is equal or close to the optimal feature subset.

The methods proposed for the problem of FSS basically split into three categories in literature: filter, wrapper and embedded [7]. In filter methods, the dataset is preprocessed and the subset is chosen by the instrinsic properties of the data without considering a classifier. Filter methods suppress the least interesting features by some quality metrics, such as a suitable ranking method like *Information Gain* or *Entropy*. Although they are effective in computation time, they may end up selecting more redundant features when compared to other methods. In wrapper methods, searching for the best feature subset is conducted by using a classifier. They generally obtain better accuracy results than the filter methods. Embedded methods aim to reduce the computation time spent in reclassifying different subsets by incorporating the selection process with the training process. The implemented algorithms in this study are examples of wrapper methods.

3.1 Problem Definition

There are two main phases of this study; selecting subsets of features and evaluating accuracy of these selected subsets. At the FSS phase of the algorithms, the subset of features are selected from the set of all features in the original dataset. Section 3.2 gives the details of these selection methods, and used classification methods for the evaluation part of the study are given in Chapter 4.

In this section, multiobjective FSS problem is formally defined. Multiobjective FSS can be described as selecting the minimum number of features with maximum classification accuracy acquired by the chosen feature subset.

Let D be a dataset with R instances and K features, that is $D = R \times K$. The objective of the FSS part is to obtain a subset, k, from the original dataset, where $k \subseteq K$, which optimizes both objectives. The multiobjective model used for optimizing both objectives is given below:

$$
\begin{aligned}
& \min(f_1) \\
& \max(f_2) \\
\textit{subject to} \quad & \\
& f_1 = |k| \\
& f_2 = accuracy(k) \quad \textit{where } k \subseteq K
\end{aligned}
\tag{3.1}
$$

There are two important decisions in this problem. First one is the number of features in the subset k. Generally, the optimal number of features is unknown in advance and incrementally searching over the sets of features is the common approach for this part of the problem. Even if the size of k is given, it is still a very intractable problem to extract the optimal subset. Second one is to choose a function to evaluate the quality of the selected subset of features. Most commonly used functions to compare the classification results are F_1-*measure* and *Accuracy*. In the beginning of this study, F_1-*measure* was chosen since it conveys a balance between precision (exactness) and recall (completeness). However, these metrics only measure the *true positive* ratios whereas *true negatives* are equally important in this study. Therefore, classification results, given in Chapter 5, are compared by using *Accuracy* which is defined as given below:

$$
Accuracy = \frac{true\ pos. + true\ neg.}{true\ pos. + false\ pos. + false\ neg. + true\ neg.} \tag{3.2}
$$

As seen in the formula, *Accuracy* is calculated by correctly classified instances divided by all instances. This function is utilized in the supervised classification part (see Section 4.1) of the experiments since the class label of the data is needed to calculate *Accuracy*.

In the unsupervised learning part (see Section 4.2) of the experiments, *Purity*, a basic metric to evaluate how good is the clustering when compared with the class labels of data, is utilized. To compute *Purity*, first, each cluster is assigned to a class by measuring the most frequent label in that cluster and then the ratio of correctly assigned instances in all clusters is calculated. Mathematical definition of purity is given below:

$$Purity = \frac{1}{|R|} \sum_{i=1}^{m} \max_{j} |w_j \cap c_i| \tag{3.3}$$

where m is the number of clusters which is 2 in this study, $\{c_1, c_2\}$ is the set of clusters and $\{w_1, w_2\}$ is the set of classes.

In experimental results (see Chapter 5), the term *accuracy* is used to express the performance of both supervised and unsupervised machine learning techniques.

3.2 Applied Algorithms

In this section, the details of implemented algorithms for the FSS part are described. Exhaustive, greedy, and genetic approaches are the three FSS algorithms applied in this study. Exhaustive approach is a brute force algorithm that is selected to compare its results with those of the greedy and genetic approaches.

3.2.1 Exhaustive Approach

Exhaustive Approach (EA), evaluating every possible combination of the set of features, can be used to find the optimal subsets of features. For every possible combination, the accuracy values are evaluated by testing the trained model. The implemented exhaustive based algorithm for FSS process can be seen in Algorithm 1. Since evaluating all possible feature subsets is unfeasible, the algorithm includes a time limit. The required time to evaluate the next feature subset is estimated and if the approximate total running time of the algorithm exceeds 3 hours, then the algorithm is terminated.

As the number of features in a dataset increases, complexity of this approach grows exponentially. Finding accuracy of every combination of a feature set in a dataset requires $\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$ iterations, and hence, the complexity of the EA becomes $O(2^n)$. For small datasets, a dataset having up to 10 features, it might be reasonable to evaluate every possible combinations; however, this is impossible for large datasets. As a result, EA gives certain results, yet it consumes too much time. Therefore, it is

Algorithm 1: Algorithm of the exhaustive based approach.

Input: $dataset$ = FSS to be applied on.

Output: S = set of examined solutions.

Function *ExhaustiveApproach (dataset)*

```
t_est ← 0 ;                          // estimated time to complete
S ← {};
n ← # of features in dataset;
i ← 0;
while t_est ≤ 10,000 and i < n do          // 10,000 sec ≈ 3 hrs
    i++;
    Q ← n choose i;
    t_initial ← timestamp;
    while Q is not empty do
        u ← Dequeue(Q);
        u.accuracy ← FindAccuracy(u, method);
        // method = LR, SVM, ELM, K-means, AP
        S ← S ∪ u;
    t_final ← timestamp;
    t_est ← t_est + ((t_final − t_initial) / (n choose i)) × (n choose i+1);
return S;
```

not feasible in many real world applications. For small number of subsets, however, it can be used to compare its exact results with those of the heuristic approaches.

3.2.2 Greedy Approach

A greedy algorithm is a process that searches for a solution which is easy-to-find. Greedy Approach (GR) makes decision by looking at the most promising solution at any moment and the key point of the algorithm is to never reconsider the decisions made. The implemented greedy based algorithm for FSS process can be seen in Algorithm 2.

Algorithm 2: Algorithm of the greedy based approach.

Input: $dataset$ = FSS to be applied on.

Output: S = set of best solutions at each iteration.

Function *GreedyApproach (dataset)*

```
S ← {};
n ← # of features in dataset;
Initialize SF with n features and set all values to false;
while SF has false features do
    Q ← FindNewSubsets(SF);
    iterationMax ← NULL;
    while Q is not empty do
        u ← Dequeue(Q);
        u.accuracy ← FindAccuracy(u, method);
        // method = LR, SVM, ELM, K-means, AP
        if u.accuracy > iterationMax.accuracy then
            iterationMax ← u;
    SF ← iterationMax;
    S ← S ∪ SF;
return S;
```

Function *FindNewSubsets (SF)*

```
Q ← {};
foreach feature in SF do
    if SF[feature] = false then
        u ← SF;
        u[feature] ← true;
        Enqueue(Q, u);
return Q;
```

As seen in Figure 3.1, having a dataset with four features, first the dataset is trained and tested for each feature separately and accuracies are calculated. Let f_i be the feature having the best accuracy result. Then the binary combinations of all fea-

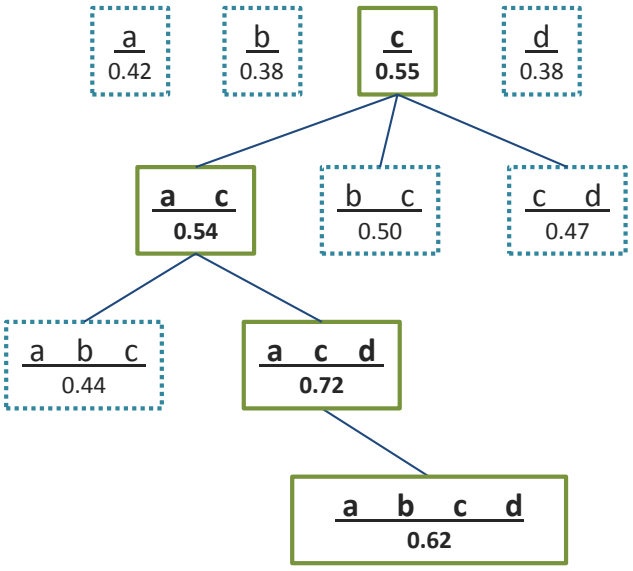


Figure 3.1: Algorithm steps of the greedy based approach.

tures including the feature f_i are trained and tested to calculate their accuracy values. Again, the best accuracy result is chosen and the related subset is selected to continue with. This procedure is performed iteratively until there are no features left to add to the subset of features.

The complexity of the GR is $O(n^2)$ since the evaluation of the feature sets explained in previous paragraph requires $\frac{n\times(n+1)}{2}$ iterations. In previous section, the complexity of EA was presented as $O(2^n)$. It is clear that the exhaustive based algorithm grows faster and requires more time to conclude than the greedy based algorithm.

GR may end up finding a local best solution rather than the global one, since it does not check all possible solutions as EA does. However, GR decreases the computational time drastically when compared with EA. Moreover, finding a good enough solution rather than trying to find the global best solution may be sufficient in real life problems.

3.2.3 Genetic Approach

A greedy algorithm may find a local best solution but it may fail finding the global solution. Therefore, an intelligent metaheuristic should be applied for better solutions.

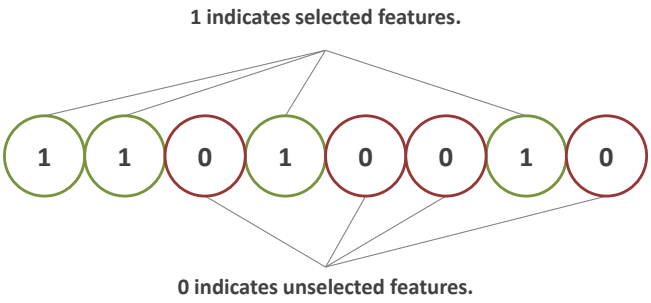


Figure 3.2: Chromosome structure.

Genetic Approach (GA) is a heuristic search method which is inspired by natural selection. Possible solutions of the problem are called chromosomes. A sample chromosome, which represents a combination of features, a subset, is given in Figure 3.2. A feature is included in the subset if its value in the chromosome is 1, whereas the value 0 shows unselected features. In this figure, the dataset has 8 features in total and the features 1, 2, 4 and 7 are the selected ones which will be evaluated.

Given or randomly chosen a set of chromosomes to populate upon are called the initial population. In single objective genetic algorithms, to determine how well a chromosome is, every chromosome's fitness value is evaluated. This value can be calculated using different types of fitness functions [2] such as statistical distributions, machine learning classifiers etc. Instead of using fitness function, objectives are set separately in this study, since there are two objectives. New generations are iteratively populated by pairing strong chromosomes with each other. The process of merging subsolutions of two pairs to generate a new chromosome is called crossover, see Figure 3.3. In order to prevent getting stuck at a local optima, mutation may be used in genetic algorithms. Mutation can be defined as randomly changing a part of a chromosome, a sample of which can be seen in Figure 3.4.

More specifically, in this study, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [10] algorithm has been employed. NSGA-II is a popular algorithm used for solving multiobjective optimization problems. This algorithm is already implemented in MOEA Framework [1], which is then utilized for solving the FSS for the

[1] MOEA Framework: a Free and Open Source Java Framework for Multiobjective Optimization, version 2.9, available at `http://www.moeaframework.org/`.

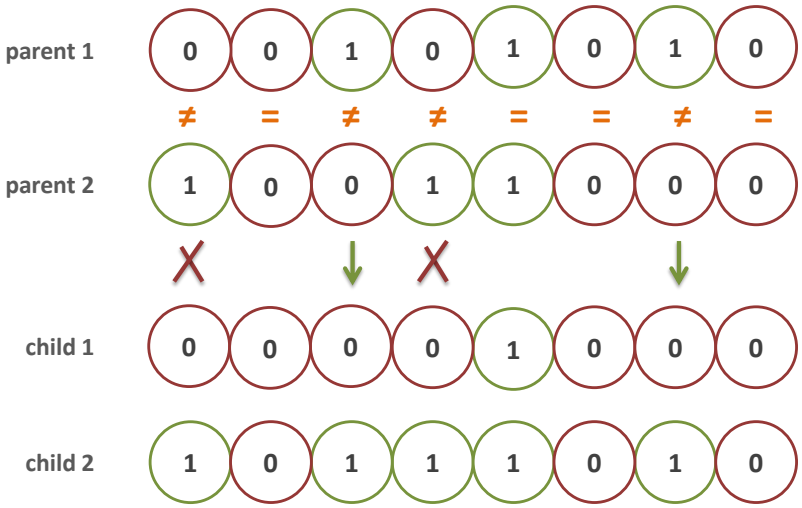


Figure 3.3: Half uniform crossover operation.

binary classification problem. Default parameters were used for the NSGA-II algorithm, as they were defined in the MOEA Framework; only population size and the number of generations were chosen manually (see Table 5.5). The selection of values of population size and the number of generations are described in Section 5.4.1 in detail.

The implemented genetic based algorithm for FSS process can be seen in Algorithm 3. A brief description of the algorithm is given as follows: First, the initial population is generated in a totally random fashion. For every chromosome in the initial population, the two objectives are calculated: the number of the selected features, and the accuracy value. Every unique chromosome with their objective values

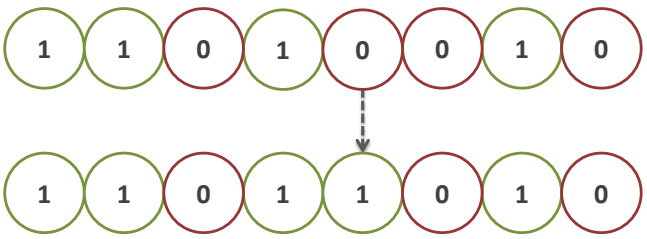


Figure 3.4: Bit flip mutation operation.

Algorithm 3: Algorithm of the genetic based approach.

Input: pop = population size, gen = number of generations.

Output: Non-dominated solutions for FSS problem.

Function *GeneticApproach (pop, gen)*

$P \leftarrow$ randomly generate initial population with the size of pop;

$S \leftarrow \{\}$; `// set of already examined individuals`

for $i \leftarrow 1$ **to** gen **do**

foreach u in P **do**

if u does not exist in S **then**

$u.objective_1 \leftarrow \#\ of\ selected\ features$;

$u.objective_2 \leftarrow FindAccuracy(u, method)$;

`// method = {LR, SVM, ELM, K-means, AP.}`

$S \leftarrow S \cup \{u\}$;

else

$u.objective \leftarrow S[u].objective$;

$P \leftarrow NSGA\text{-}II(P)$; `// generate new population [10]`

return $FindNonDominatedSolutions(P)$;

Function *NSGA-II (P)*

$size \leftarrow |P|$;

$P \leftarrow NonDominatedSort(P)$;

for $i \leftarrow 1$ **to** $size/2$ **do**

$p_1 \leftarrow BinaryTournament(P[rand(0, size)], P[rand(0, size)])$;

$p_2 \leftarrow BinaryTournament(P[rand(0, size)], P[rand(0, size)])$;

$c_1, c_2 \leftarrow HalfUniformCrossover(p_1, p_2)$;

`//` p_1, p_2 `= Parent 1 - 2,` c_1, c_2 `= Child 1 - 2`

$c_1 \leftarrow BitFlipMutation(c_1)$;

$c_2 \leftarrow BitFlipMutation(c_2)$;

$P \cup \{c_1\} \cup \{c_2\}$;

`// size of P is doubled`

$P \leftarrow NonDominatedSort(P)$;

return $P[0 \ldots size\text{-}1]$;

are recorded within a set in order not to evaluate objective values of the same chromosomes repeatedly. Finally the NSGA-II algorithm starts optimization with respect to both objectives.

According to NSGA-II algorithm; an individual, p , dominates another individual, q, only if at least one of p's objectives are better than of q's, while keeping all other objectives at least same. This may also be referred as q is dominated by p. If both p and q have at least one objective that is better than their opponent's, then p and q are non-dominated to each other.

Since this is a multiobjective optimization, all objectives need to be considered when individuals are compared. Selection of the stronger individual process depends on two metrics, namely fronts and crowding distance values. Non-dominated sort algorithm splits all individuals into their respective fronts according to their accuracy and selected feature size values. All individuals that are non-dominated by any other individuals are grouped in the first front. All individuals that are dominated by the first front group members, but non-dominated among each other are grouped into the second front, and so on. After all fronts are determined, the crowding distance values of the chromosomes are assigned. Crowding distance is an intra-front algorithm, and it measures the Euclidian distance between individuals according to their objective values. Once both metrics are calculated, comparing two individuals becomes straightforward. An individual assigned into a smaller front, i.e. first front, is stronger than of all individuals having bigger fronts. On the other hand, if two individuals are assigned into the same front, then individuals having larger crowding distance value are considered as stronger than of those having smaller crowding distance values.

Given a population, P, the NSGA-II algorithm applies a non-dominated sort over P and calculates crowding distance values. The algorithm generates new chromosomes by using the individuals in P. Generation starts with binary tournament selection. Two individuals are randomly chosen and the strongest individual becomes the first parent. The second parent is decided in the same way. In order to generate two children, half uniform crossover is applied on these parent individuals. Bit flip mutation may also be applied to the children individuals with some probability. The size of the population doubles at the end of the generation process. Finally, with a utilization of

the non-dominated sort and crowding distance values, weak individuals are discarded and only half of the individuals in the population remains as the new population.

Let pop be the population size and gen be the number of generations. The complexity of the GA is $O(pop \times gen)$.

3.3 Complexity Analysis

In previous section, the complexity of the greedy based approach is calculated as $O(n^2)$ and presented as a faster algorithm than the exhaustive based approach. As explained in detail in Section 5.4.1, population size is chosen as 40 and the number of generations is chosen as 60 for GA in the experimental part of this study. According to the complexity of GA, 2400 operations is needed in theory. Up to 11 features, which yields $2^{11} = 2048$ operations, it can be inferred that EA seems more reasonable to be used than GA. With a similar calculation, up to 49 features, GR is less complex than GA, see Figure 3.5. On the other hand, in GA, examined set of features are stored in a set and their objectives are not evaluated again. Moreover, the elitism property of the NSGA-II algorithm suggests a great decrease in practice which provides leverage

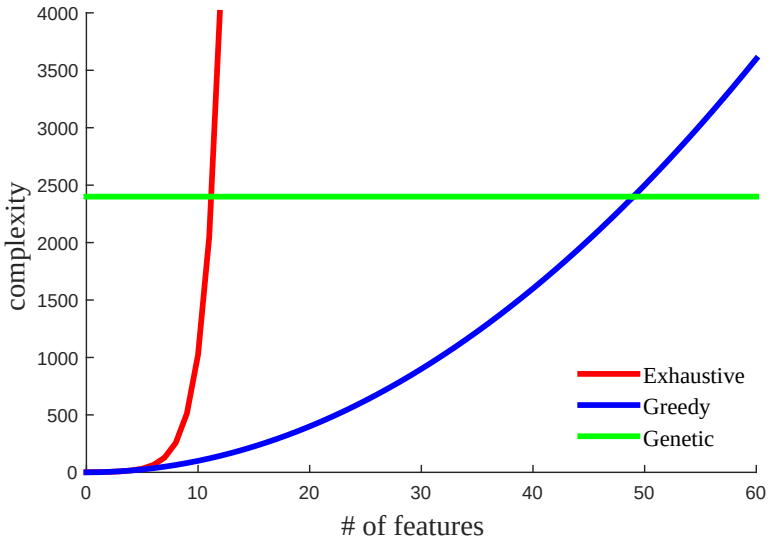


Figure 3.5: Complexity comparison of feature subset selection algorithms.

to GA against both EA and GR regardless of the feature size.

Note that, only subset selection part is considered in the complexity analysis. Since the classification complexity applies for all types of FSS algorithms, it is not given here for the sake of brevity. Moreover, all classification methods have different impacts on complexity. However, it should be mentioned that classification has a significant effect on complexity, since it operates as a multiplier in the complexity calculation.

CHAPTER 4

BINARY CLASSIFICATION USING MACHINE LEARNING

As mentioned in Section 3.1, second part of the study is to evaluate the accuracy of the selected subset of features. In this chapter, information about the machine learning techniques that are applied in this study is given. The datasets are trained with three supervised learning algorithms; Logistic Regression (LR), Support Vector Machines (SVM) and Extreme Learning Machine (ELM), and two unsupervised learning algorithms; K-means and Affinity Propagation (AP).

Datasets used in the experiments include either numerical data or categorical data which are converted to numerical values in the preprocess phase of the study. Therefore, the classifiers are chosen among those which are compatible to work with this kind of data. LR is easy, fast, and it does not require parameter tuning. SVM is well known as an effective classifier for binary classification of large datasets. ELM is a trending supervised classifier since it shortens training time considerably when compared to traditional neural networks. K-means is known as the most simple clustering algorithm. AP is a relatively new unsupervised learning algorithm which performs well in means of clustering accuracy.

4.1 Supervised Machine Learning Techniques

A supervised learning algorithm generates a model through a training process where the label or result of the data is known a priori. This model is used for classifying future instances in which the feature values are given as input, but the class label is unknown.

In this section, supervised machine learning techniques implemented in this study are explained in detail.

4.1.1 Logistic Regression

LR is a method which estimates the probability of occurrence of an event with respect to resemblance of its attributes to the training set. Distinctly from linear regression, which predicts a continuous value, LR is especially good for binary classifications since the used logarithm function reduces the value into the range between 0 and 1 [3]. Besides, experiments show that the performance of LR can result as good as some other more complicated classifiers [24]. LR uses Sigmoid function, see in Equation 4.1, to find the probability of an event to occur. The event is then predicted as 1 if its occurrence probability is greater than 0.5 and it is predicted as 0 otherwise.

$$P(y = 1 \mid X, \theta) = \frac{1}{1 + e^{-\theta X}} \tag{4.1}$$

where X is the matrix consisting feature sets to be examined, θ is a vector representing coefficient values for all features, and y is the probability of occurrence of the event.

A Matlab function for LR classification, *glmfit*, is utilized to train and test the datasets in the experimental part of the study.

4.1.2 Support Vector Machines

SVM performs classification by constructing a hyperplane to separate given data points [8]. The closest data points to the separating hyperplane are called support vectors. Figure 4.1 shows how a hyperplane is constructed between support vectors in a 2D space. It is seen that the separating line defines a boundary between two classes colored as green and red.

The optimal hyperplane is constructed in an iterative fashion by maximizing the margin between the hyperplane and the support vectors of the classes, due to the intuition that the generalization error decreases as the margin increases [40]. However, when

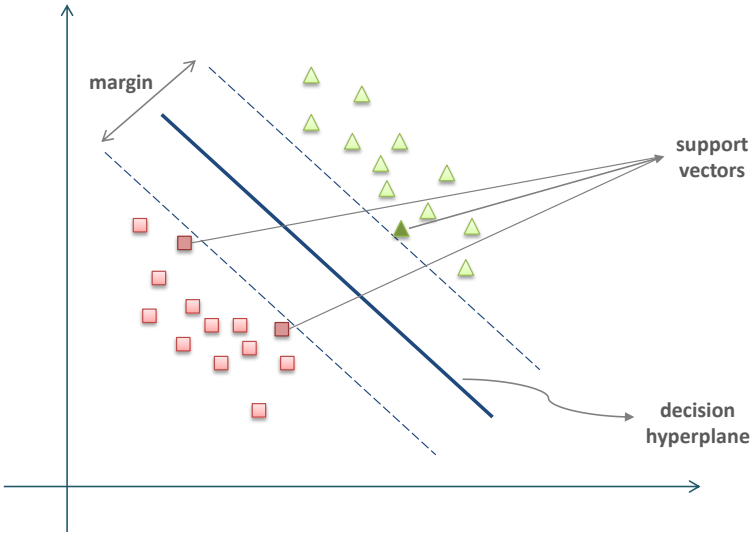


Figure 4.1: Separating two classes by constructing a hyperplane with Support Vector Machines.

classes are not linearly separable, this method would not suffice to classify successfully. In such case, data points can be mapped into a higher dimensional space with a transform function $\varphi(\vec{x_i})$, also known as the kernel function, expecting that the classes would be more discrete in that space that is separable by a hyperplane [15]. An example of classifying a nonlinear dataset by using kernel transformation can be seen in Figure 4.2.

A Matlab function for binary SVM classification, *fitcsvm*, is utilized to train and test the datasets in the experimental part of the study.

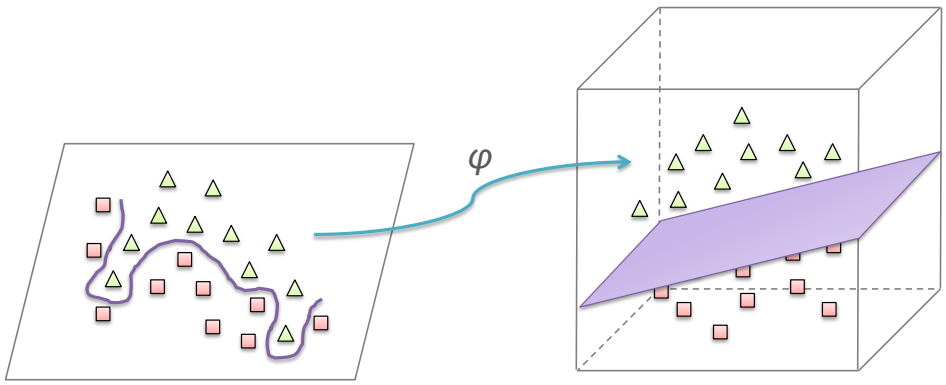


Figure 4.2: Mapping data points to a higher dimension.

4.1.3 Extreme Learning Machine

Neural networks are inspired by the brain, the biological nervous system. Similar to brain, neural networks consist of several layers. ELM is a type of feedforward neural network with a single hidden layer. This specific model consists of three layers called input, hidden and output. Input layer represents the training data information that is given to the network for the learning process. Inputs are weighted and transformed by a function and passed to the other layer. The activity of hidden layer is determined by the activities of the input units and the weights on the connections between the input and hidden layer. The behaviour of the output layer depends on the activity of the hidden layer and the weights between the hidden and output layer. An example of a feedforward neural network can be seen in Figure 4.3.

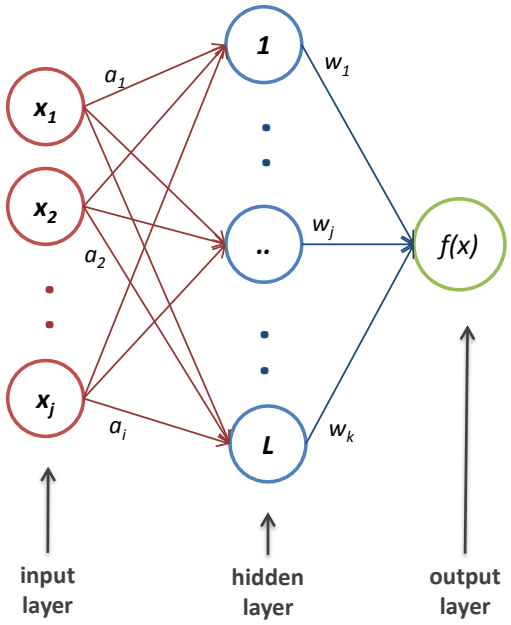


Figure 4.3: Feedforward neural network with a single hidden layer.

The output of such neural network having L number of hidden nodes can be represented with the formula below:

$$f(x) = \sum_{i=0}^{L} w_i G(a_i, b_i, x) \tag{4.2}$$

where w_i is the weight between the i-th hidden node and the output layer. $G(a_i, b_i, x)$ is the output of the activation function between the hidden layer and the input layer. Some commonly used activation functions are Sigmoid function, Gaussian function, and Fourier function. Sigmoid function, the most popular one, can be seen in Equation 4.3.

$$G(a, b, x) = \frac{1}{1 + e^{-(ax+b)}} \tag{4.3}$$

where x is the input, a is the contribution weight of the input to the hidden layer and b is the bias term.

Feedforward neural networks need to tune its parameters iteratively which increases the learning time. Since ELM does not require parameter tuning, learning time of the algorithm is much lower than the traditional feedforward network learning algorithms.

Huang et al. [19] analyze ELM theories and applications in detail. It is pointed out that ELM can overcome challenges which neural networks or SVMs cannot handle. In this survey, it is also stated that ELM performs better when compared to traditional computational intelligence techniques.

An ELM library which is available for download [1] is utilized in the experimental part of the study.

4.2 Unsupervised Machine Learning Techniques

An unsupervised learning algorithm divides data points into similarity groups called clusters. A cluster consists of a set of instances which are more similar to each other in some way than the other ones in other clusters. Since there is no need for output values in unsupervised learning, the class labels of the datasets used in this study are neglected for the training part of these techniques. In this section, unsupervised machine learning techniques implemented in this study are explained in detail.

[1] Extreme Learning Machine, `http://www.ntu.edu.sg/home/egbhuang/`.

4.2.1 K-means

K-means clustering is an iterative algorithm in which k clusters are partitioned by selecting k centroids at each iteration and assigning data points to the closest centroid's cluster [14]. Initially k is defined by user and the algorithm selects k centroids randomly. Afterwards data points are grouped according to their distance to each centroid. Different methods for calculating distance exist, e.g. Euclidian distance or Manhattan distance. Cluster centers are computed to find new centroids and then the steps above are repeated until cluster members stay same. In K-means algorithm, setting k is a hard decision to make. In general, different k values are tested and silhouette coefficient [35] is used for deciding which k value gives the best clusters. Since this study is on binary classification, k is set as 2 in the experiments.

In K-means clustering, initial data points are generally chosen randomly. However, selection of initial data points may affect the results which can implicitly change the accuracy of clustering [5]. The importance of initial data point selection can be seen in Figure 4.4. Left side of the figure shows an example of inappropriate clustering of the sample data points. On the other hand, a drastic change is observed in the result when the algorithm selects some other initial data points, which can be seen at the right side of the figure. Therefore, it is clear that selection of the initial data points

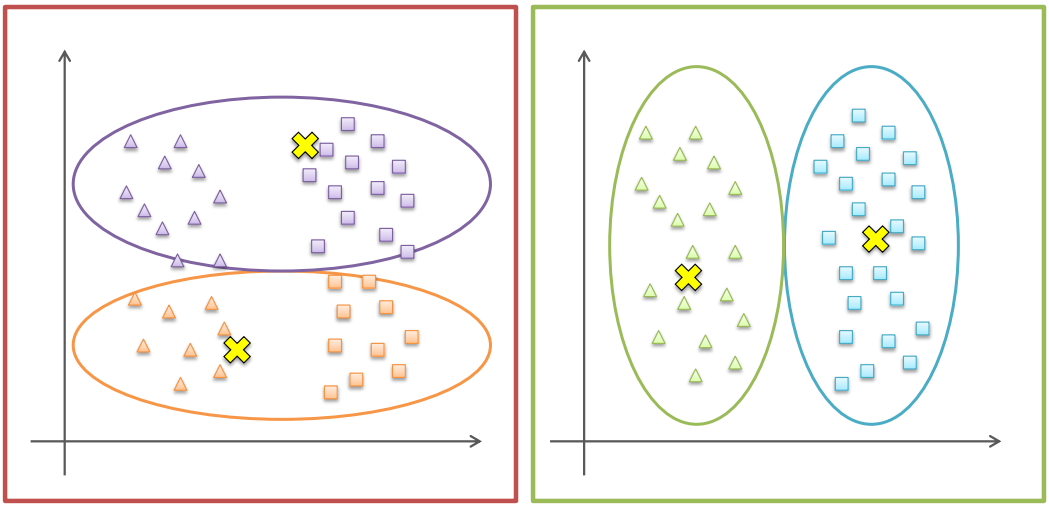


Figure 4.4: The significance of initial data points in K-means clustering.

correlate with the clustering result. Since the datasets used in this study already have a class field, this information is used in determination of initial data points. Average data points for each classes are calculated and used as initial data points in K-means clustering experiments to maximize the accuracy.

A Matlab function for K-means clustering, *kmeans*, is utilized in the experimental part of the study.

4.2.2 Affinity Propagation

AP is a clustering algorithm based on iterative messaging between data points [12]. Initially, all data points are considered as cluster representatives by the algorithm, in other words *exemplars*. Two types of messages, responsibility and availability, are sent between data points until a convergence is fulfilled. Responsibility messages are sent from the data points to their candidate exemplars. A responsibility message contains information about how well the data point serves to that candidate exemplar. Availability messages are sent from the candidate exemplars to the data points. An availability message represents how appropriate the candidate exemplar is for that data point.

Let s be a function which calculates the similarity between two data points. Using this similarity result, the algorithm performs by passing messages and updating the statuses of the data points. Let $r(i,k)$ be the responsibility function and $a(i,k)$ be the availability function. Equations 4.4 and 4.5 shows how responsibility and availability values of a data point are updated [11].

$$r(i,k) = s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\} \tag{4.4}$$

$$a(i,k) = \begin{cases} \min\{0, (r(k,k) + \sum_{i' \notin \{i,k\}} \max\{0, r(i',k)\})\}, & \text{for } i \neq k \\ \sum_{i' \neq k} \max\{0, r(i',k)\}, & \text{otherwise} \end{cases} \tag{4.5}$$

AP does not require the number of clusters beforehand which is a major distinction to K-means. Furthermore, AP does not randomly choose some data points as cluster representatives initially which gives an advantage in such case that initial choices do not conclude with a good solution.

An AP library which is available for download [2] is utilized in the experimental part of the study.

[2] Affinity Propagation, `www.psi.toronto.edu/affinitypropagation`.

CHAPTER 5

EXPERIMENTS AND RESULTS

In this chapter, first, the experimental environment is described and problem instances are introduced. Then, the results of the experiments performed to evaluate the effectiveness of the discussed methods in previous chapters are presented.

5.1 Experimental Environment and Problem Instances

Eleven datasets were used in the experimental part of the study. Ten of them were obtained from University of California UCI Machine Learning Repository [1], a well-known machine learning data repository. The eleventh dataset, Financial, can be reached from Pacheco et al. [33]. The feature size of the original datasets range from 8 to 93 and the instance counts range from 351 to $581,012$.

Table 5.1: Specification of the datasets used in the experiments.

Problem ID	Dataset	Number of features	Number of instances	Actual number of classes
CT	Covertype	54	581,012	7
MR	Mushrooms	22	8124	2
SB	Spambase	57	4601	2
NU	Nursery	8	12,960	5
C4	Connect-4 Opening	42	67,557	3
WF	Waveform	40	5000	3
FI	Financial	93	17,108	2
PM	Pima Indians Diabetes	8	768	2
BC	Breast Cancer	9	699	2
IO	Ionosphere	34	351	2
WBC	Wisconsin Breast Cancer	30	569	2

[1] UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml`

Total number of features, instances and classes of the datasets are shown in Table 5.1. The first column indicates the identifier (ID) of the dataset used in the forthcoming tables. The second column is the name of the dataset. The third column is the total number of features the dataset has. Total number of instances of the datasets are given in the next column. Finally, the last column gives information about the actual number of classes of the datasets.

Detailed description of the datasets presented in Table 5.1 is given below:

- Covertype (CT) dataset contains cartographic variables to predict forest cover type. Elevation, slope and soil type are the sample attributes of this dataset. There are both categorical and integer values in the dataset.
- Mushrooms (MR) dataset contains physical characteristics of different mushroom species in the Agaricus and Lepiota families. Class labels provide information about the mushroom being poisonous or edible. The dataset contains categorical data.
- Spambase (SB) dataset includes information about frequencies of different words and characters. Class labels denote whether the e-mail is spam or not. The dataset contains integer and real numbers.
- Nursery (NU) dataset was developed to decide whether an applicant should be recommended to a nursery school or not. The dataset provides information about the applicant's family structure in terms of social, health and financial statuses. The dataset contains categorical data.
- Connect-4 (C4) dataset consists of the players' position information of the Connect-4 game. The game board has $6 \times 7 = 42$ positions which can either be x (player x has taken), o (player o has taken) or b (blank). Class labels indicate win or loss status. The dataset contains categorical data.
- Waveform (WF) dataset provides attributes which include noise of different wave types. Class labels indicate which wave fits to the corresponding noise information. The dataset contains real valued numbers.
- Financial (FI) dataset contains financial ratios of various firms and class labels denote the status of the firm as failed or healthy. The dataset contains integer

and real numbers.

- Pima Indians Diabetes (PM) dataset keeps a patient's background information related to diabetes such as number of pregnancy, blood pressure, glucose tolerance test etc. Class labels indicate positive or negative result for diabetes. The dataset contains integer and real numbers.

- Breast Cancer (BC) dataset presents characteristics of various breast cells. Some attributes are clump thickness, uniformity of cell size and shape. The class labels show if the result is benign or malignant. Attribute values are given as integer values.

- Ionosphere (IO) dataset contains data gathered from a radar which detects free electrons in the ionosphere. Class labels show if there is an evidence of a structure in the ionosphere or not. The dataset contains integer and real numbers.

- Wisconsin Breast Cancer (WBC) dataset provides characteristics of breast cell nucleuses. Radius, perimeter and concavity are the sample attributes and the class labels are benign or malignant. The dataset contains real numbers.

Since the algorithms are implemented for binary classification problems, first two classes having the most number of instances are selected in the experiments for the datasets which include more than two actual classes. Another important issue was the existence of text based categorical data in the datasets. These values were converted to numerical data before applying machine learning algorithms in the experimental part of the study. A possible conversion would have been adding all distinct values as new features and setting only the related feature value as 1 while keeping all others as 0. This technique is often used in literature [17] since it prevents false prioritization of categorical data. However this conversion technique increases the number of features in the dataset, and hence it is unsuitable for the purpose of this study. Therefore, data types apart from real numbers were cast into integers by enumerating every distinct value in the dataset. A sample of original values of the NU dataset and its values after conversion process are given as an example in the Appendix A.

In the experiments, first, the classification model is trained using the training set, and afterwards, accuracy value of each model is calculated using the test set. Different

methods for generating these train and test sets exist. The most prominent method is k-fold cross-validation proposed by Salzberg [36]. K-fold cross-validation can simply be described as dividing the dataset into k subsets and utilizing a loop to select each subset as the test set for the respective iteration. At each iteration, remaining $k-1$ subsets are combined and labeled as the training set, and used for training the classification model. Trained model is then used for finding the accuracy of the test set. After k iterations, average accuracy is calculated and accepted as the accuracy of the classifier for that dataset. K-fold cross-validation is robust and less prone to errors in calculation of average accuracy since no data is discarded. On the other hand, using all data may increase the training time of large datasets having many instances. It gets even worse especially when the classification model iterates over every instance in the training set at each iteration of its optimization loop, e.g. SVM, K-means, etc.

A second method could be selecting the training and test sets randomly. In this method, a predefined number of random selected instances generate the training and test sets of a dataset. In this method, a total random selection of the training set could result in selecting instances from one class only. Training a model with instances of the same class could decrease accuracy. Moreover, some valuable data could be discarded during the random selection process. Hence, this method is more prone to errors in average accuracy calculation. Finally, the accuracy results may not be consistent due to complete randomness. Same dataset and classification model may produce distinct average accuracy values.

In this study, a specialized random selection method is applied. In this method, 10 different training sets, and 10 test sets for each training set (a total of 100 test sets) were generated and saved. Instances in the training set were randomly selected, however, selected in such fashion that training set involves the same proportion of each class existing in the original dataset. For example, consider that the original dataset includes 920 class0 instances, and 80 class1 instances. If the training set size is predefined as 100, all training sets have 92 randomly selected class0 instances, and 8 randomly selected class1 instances. Moreover, none of the instances existing in the training set is included in its test sets.

Table 5.2 shows selected training and test set sizes of the datasets and presents a pre-

Table 5.2: Size of the training and test sets used in the experiments and details of classification accuracy of the sets evaluated by Logistic Regression.

Dataset ID	Size of each training set	Size of each test set	Training				Test			
			Mean	Std. Dev.	Max.	Min.	Mean	Std. Dev.	Max.	Min.
CT	600	200	0.791	0.013	0.812	0.772	0.761	0.027	0.830	0.700
MR	1300	200	0.945	0.006	0.954	0.933	0.937	0.019	0.975	0.885
SB	600	200	0.940	0.020	0.945	0.913	0.893	0.022	0.935	0.825
NU	400	200	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000
C4	1200	200	0.837	0.033	0.865	0.748	0.820	0.042	0.885	0.655
WF	400	200	0.952	0.014	0.978	0.933	0.893	0.027	0.945	0.810
FI	1000	200	0.947	0.024	0.969	0.906	0.909	0.041	0.975	0.800
PM	268	200	0.793	0.017	0.817	0.769	0.762	0.026	0.815	0.705
BC	199	100	0.980	0.016	1.000	0.960	0.954	0.022	0.990	0.900
IO	101	50	1.000	0.000	1.000	1.000	0.812	0.055	0.960	0.660
WBC	169	80	1.000	0.000	1.000	1.000	0.924	0.034	1.000	0.825

liminary experimentation on all datasets by training the classifier having all features selected. The training and test set sizes are compatible with Unler et al. [42]. Accuracy statistics of all subsets in terms of mean, standard deviation, maximum and minimum values for each dataset are also given in the table, for both training and test sets. In order to obtain these statistics, LR is used for classification.

All experiments were conducted on an Intel Core i7-6700 processor with a CPU clock rate of 3.40 GHz and 16 GB main memory. The implemented algorithms mainly have two components, feature selection and machine learning. The FSS algorithms were implemented in the Java environment and MATLAB 2015a version was used for executing the machine learning techniques for testing purposes.

5.2 Exhaustive Algorithm Results

The results given here present the performance of machine learning techniques on feature subsets generated by Exhaustive Approach (EA) that evaluates every subset of features. This performance of the exhaustive selection is used as an evaluation criterion for other feature selection approaches. Table 5.3 gives the maximum number of features that could be fully examined by EA along with the achieved maximum accuracy and execution times of the algorithms.

Table 5.3: Classification results of feature subsets generated by exhaustive algorithm.

Dataset ID	Total # of features	Classifier	Max. examined # of features	Max. accuracy	# of features at max. accuracy	Exec. time (sec.)
CT	54	LR	3	0.764	3	1774.5
		SVM	3	0.763	3	4285.7
		ELM	3	0.657	3	6997.5
		K-means	3	0.725	1	2016.3
		AP	2	0.863	2	4517.4
MR	22	LR	6	0.952	6	8044.4
		SVM	4	0.946	4	3518.6
		ELM	4	0.992	4	1835.2
		K-means	5	0.931	5	5140.5
		AP	2	0.983	1	2811.1
SB	57	LR	3	0.857	3	1146.6
		SVM	3	0.865	3	5084.5
		ELM	3	0.866	3	7262.8
		K-means	3	0.744	3	2545.8
		AP	2	0.863	2	4923.5
NU	8	LR	8	1.000	1	15.7
		SVM	8	1.000	1	34.0
		ELM	8	1.000	1	50.7
		K-means	8	1.000	1	15.1
		AP	8	1.000	1	553.0
C4	42	LR	4	0.765	4	5962.9
		SVM	3	0.746	3	2978.3
		ELM	3	0.753	3	3889.9
		K-means	3	0.731	1	1632.5
		AP	2	0.908	2	5734.5
WF	40	LR	4	0.902	4	1940.3
		SVM	3	0.893	3	1743.9
		ELM	3	0.889	3	2541.8
		K-means	4	0.888	4	6548.3
		AP	2	0.859	2	1512.6
FI	93	LR	3	0.967	3	6085.1
		SVM	2	0.966	2	760.1
		ELM	2	0.966	2	985.5
		K-means	2	0.966	2	500.6
		AP	1	0.969	1	974.0
PM	8	LR	8	0.771	7	5.2
		SVM	8	0.769	6	40.2
		ELM	8	0.739	1	47.1
		K-means	8	0.736	4	12.7
		AP	8	0.792	5	287.4
BC	9	LR	9	0.963	5	10.3
		SVM	9	0.968	4	56.3
		ELM	9	0.963	3	80.5
		K-means	9	0.961	5	19.8
		AP	9	0.970	8	366.7
IO	34	LR	5	0.888	5	7771.7
		SVM	4	0.878	4	6000.0
		ELM	4	0.898	3	5844.0
		K-means	5	0.860	5	9998.8
		AP	3	0.894	3	1603.5
WBC	30	LR	5	0.975	5	3111.3
		SVM	4	0.975	4	3565.3
		ELM	4	0.954	3	3957.4
		K-means	5	0.917	4	6546.9
		AP	3	0.919	3	1881.1

As seen in Table 5.3, when the total number of features in a dataset is small, up to 9 for these datasets, it is possible to find the best subsets of features by using EA in a timely manner. For example, the *Nursery* dataset has 8 features in total and every combination of feature subsets can be examined by SVM in 34 seconds to find the best accuracy. On the other hand, datasets with higher number of features cannot be examined completely due to time limit. It can be seen from the table that EA could only examine the subsets with maximum 6 number of features for the *Mushrooms* dataset. As the feature size increases, required time to examine all possible subsets increases exponentially. For example, there exists 2^{54} possible combinations for *Covertype* and 2^{57} possible combinations for *Spambase* datasets.

It can be deduced from the results that, performance of EA in terms of required time to finish the algorithms is inversely correlated to feature size; meaning that, as the feature size increases, the number of fully evaluated subsets in the same amount of time decreases.

5.3 Greedy Algorithm Results

The results given here present the performance of machine learning techniques on feature subsets generated by Greedy Approach (GR) explained in Section 3.2.2. Table 5.4 gives the results of GR in terms of execution time and achieved maximum accuracy. This table gives promising results when compared to EA results. For example, SVM can classify *Nursery* dataset in 4.3 seconds which is much shorter than 34 seconds of EA. Also, EA worked for 19 minutes, yet it could only evaluate up to 0.857 accuracy value for *Spambase* dataset with LR; whereas GR achieved 0.923 accuracy value in about 7 minutes.

Selecting feature subsets exhaustively consumes much more time than GR. Moreover, GR achieves higher accuracy values in most cases since EA can only examine a limited proportion of subsets. Note that, the results of EA does not include the whole execution time to need to finish the algorithm since it is not feasible to examine all possible subset of features when the number of features is high.

Table 5.4: Classification results of feature subsets generated by greedy algorithm.

Dataset ID	Total # of features	Classifier	Max. accuracy	# of features at max. accuracy	Exec. time (sec.)
CT	54	LR	0.776	31	327.3
		SVM	0.775	32	507.7
		ELM	0.651	2	298.3
		K-means	0.725	1	133.0
		AP	0.860	3	4028.1
MR	22	LR	0.949	11	39.9
		SVM	0.954	16	93.5
		ELM	0.991	3	50.6
		K-means	0.836	15	38.4
		AP	0.979	2	4702.6
SB	57	LR	0.923	22	431.3
		SVM	0.923	31	513.3
		ELM	0.900	16	349.2
		K-means	0.818	37	151.8
		AP	0.915	31	10385.6
NU	8	LR	1.000	1	2.9
		SVM	1.000	1	4.3
		ELM	1.000	1	6.8
		K-means	1.000	1	2.1
		AP	1.000	1	61.7
C4	42	LR	0.835	28	215.7
		SVM	0.832	35	1423.5
		ELM	0.804	13	181.7
		K-means	0.731	1	132.1
		AP	0.922	10	10800.2
WF	40	LR	0.923	11	27.7
		SVM	0.923	14	184.4
		ELM	0.902	8	159.1
		K-means	0.896	17	52.8
		AP	0.923	38	2228.1
FI	93	LR	0.967	6	2249.9
		SVM	0.966	9	21463.7
		ELM	0.967	12	954.5
		K-means	0.966	2	575.3
		AP	0.970	18	51933.6
PM	8	LR	0.771	7	0.9
		SVM	0.769	6	7.0
		ELM	0.735	1	6.6
		K-means	0.736	4	1.8
		AP	0.792	5	41.2
BC	9	LR	0.962	5	1.0
		SVM	0.966	7	5.4
		ELM	0.964	3	6.9
		K-means	0.961	5	1.9
		AP	0.970	8	33.7
IO	34	LR	0.892	11	42.5
		SVM	0.895	20	69.6
		ELM	0.894	3	71.3
		K-means	0.856	2	19.6
		AP	0.913	10	177.1
WBC	30	LR	0.974	5	31.0
		SVM	0.977	6	58.4
		ELM	0.953	3	62.0
		K-means	0.916	3	19.3
		AP	0.924	13	228.4

5.4 Genetic Algorithm Results

In this section, first, selection steps of population size and the number of generations are given. Then, the performance of machine learning techniques on feature subsets generated by Genetic Approach (GA) are presented in terms of execution time and achieved maximum accuracy. Finally, the improvement of the solutions populated by GA through generations is shown in detail with graphics.

5.4.1 Setting the Population Size and the Number of Generations

One of the most crucial point of a GA is determining the population size and the number of generations. The number of individuals in a population must be large enough to keep the diversity of the exploration space. Similarly, the number of generations must be large enough to allow the algorithm to converge. At the same time, both must be small enough to have a reasonable execution time during optimization. Therefore, in this part of the study, the experiments are carried out to find the most promising population size and number of generations. For that purpose, an interval of 10 to 100 with a step size of 10 is examined for both population size and number of generations. LR is utilized to calculate the accuracy of feature subsets. Figure 5.1 presents the results for three criteria; accuracy, number of features, and execution time of the algorithm during the optimization tests. Among three subfigures, the best case is colored with green and the worst case is colored with red. In-between values are colored with a transition color with respect to their goodness. A higher value is better for accuracy, whereas a lower value is better for both feature size and execution time. The results show the average of all 11 problem datasets.

Accuracy, number of selected features and execution times are the three criteria examined to decide population size and number of generations. Figure 5.1a which shows the progress of accuracy according to change in population size, provides a closer look to the effect of population size on the accuracy. If achieving the maximum accuracy value was the only objective, then selecting 100 would be the best solution for both population size and the number of generations, as easily be seen in Figure 5.1a. However, minimizing the number of selected features is another objective of

average accuracy	population size									
number of generations	10	20	30	40	50	60	70	80	90	100
10	0.9871	0.9903	0.9914	0.9929	0.9935	0.9919	0.9944	0.9940	0.9941	0.9943
20	0.9908	0.9920	0.9944	0.9951	0.9963	0.9978	0.9985	0.9965	0.9972	0.9962
30	0.9912	0.9943	0.9950	0.9955	0.9967	0.9982	0.9965	0.9972	0.9975	0.9966
40	0.9919	0.9949	0.9967	0.9959	0.9965	0.9986	0.9977	0.9982	0.9977	0.9981
50	0.9919	0.9964	0.9972	0.9959	0.9972	0.9981	0.9977	0.9981	0.9973	0.9985
60	0.9945	0.9956	0.9958	0.9963	0.9974	0.9984	0.9979	0.9975	0.9972	0.9986
70	0.9934	0.9964	0.9981	0.9970	0.9978	0.9973	0.9981	0.9986	0.9979	0.9981
80	0.9969	0.9971	0.9972	0.9971	0.9983	0.9986	0.9986	0.9980	0.9987	0.9976
90	0.9943	0.9969	0.9977	0.9975	0.9972	0.9969	0.9981	0.9987	0.9987	0.9982
100	0.9957	0.9977	0.9970	0.9979	0.9975	0.9978	0.9976	0.9982	0.9987	0.9988

(a) Average accuracy values.

average feature size	population size									
number of generations	10	20	30	40	50	60	70	80	90	100
10	4.8493	4.5343	4.6027	4.5558	4.4867	4.5573	4.6002	4.4983	4.9559	4.6291
20	3.7158	4.6736	3.9910	3.8249	3.8976	4.2750	4.0223	4.0946	3.8676	4.0076
30	3.6128	3.8063	3.8124	3.4479	4.0059	3.9338	3.8091	3.7813	3.8916	3.6582
40	3.9327	3.6642	3.8757	3.8135	3.4566	3.9843	3.6004	4.0295	3.5413	4.1986
50	3.5839	3.6633	3.6794	3.7502	3.4648	3.7642	3.7659	4.1467	3.8012	3.7490
60	3.5524	3.3840	3.6437	3.2256	4.0015	4.1042	3.8273	3.8739	3.5229	3.6367
70	3.4460	3.4602	3.6948	3.6308	3.5887	3.8308	3.7044	3.6588	3.9537	3.8007
80	3.6696	3.9515	3.6992	3.4956	3.7081	3.7431	3.8415	3.5591	3.5993	4.0732
90	3.4832	3.7229	3.5884	3.9165	3.3724	3.7939	3.8603	3.8134	3.9797	3.9834
100	4.0571	3.4846	3.7172	3.9868	3.8783	3.6988	3.4917	3.8294	3.8771	3.8869

(b) Average number of features.

average exec. time	population size									
number of generations	10	20	30	40	50	60	70	80	90	100
10	0.3741	0.6545	0.9652	1.2561	1.5584	1.7899	2.1307	2.4099	2.6123	3.1989
20	0.4544	0.8446	1.1826	1.6272	2.2269	2.1955	2.5989	3.1167	3.3974	4.1017
30	0.4422	0.9641	1.3907	1.8143	2.3165	2.9326	3.3037	3.5150	4.0993	4.3395
40	0.6533	1.1700	1.7077	2.1029	2.3892	3.1540	3.5492	3.9927	4.6459	5.4172
50	0.6777	1.1930	1.7166	2.3358	2.7689	3.3346	3.9718	4.7464	5.2018	6.1517
60	0.8259	1.2598	1.7761	2.3590	3.0762	4.1120	4.5547	4.9590	5.8034	7.2297
70	0.7487	1.4250	1.8989	2.5016	3.3245	4.1707	4.8512	6.0536	6.5200	7.6520
80	0.8942	1.5789	2.1598	3.0348	3.7306	4.4851	5.3548	6.1916	7.9519	8.4497
90	0.8728	1.6021	2.2648	3.0820	3.8577	4.9655	5.9662	7.4928	8.3107	9.6859
100	1.0319	1.7891	2.3987	3.6617	4.3096	5.5508	6.2921	7.8770	9.3527	10.4397

(c) Average execution times.

Figure 5.1: Average accuracy, number of features and execution times of all datasets with varying population size and number of generations.

the study. It can be seen from Figure 5.1b that, selected number of features is not minimized when the value of 100 is selected for population size and the number of generations. Moreover, execution time of the algorithm is an important aspect as well when deciding these sizes.

Figure 5.1c shows extremely high execution time for the mentioned values. A noteworthy observation is that the color in the top right cell (pop = 100, gen = 10) is more reddish than the bottom left cell (pop = 10, gen = 100). It is clear that increase in population size affects the execution time worse than increase in number of generations. Therefore, it is clear that population size should be chosen as small as possible.

A smarter method should be applied for selection of these values taking all these information into account. According to Figure 5.1a, accuracy values are monotonically increasing only when population size is 40. It can be deduced that only when the population size is selected as 40, an increase in accuracy can be expected as the number of generations increases. This is an important sign which suggests that the results are reliable. Similarly, according to the Figure 5.1b, the minimum number of selected features is found when population size is selected as 40. As a result, the value 40 was used as population size throughout this study.

Figure 5.1b shows promising results for the number of generations values of 30, 60 and 80, when population size is selected as 40. It is clear from Figure 5.1b that, the minimum number of features can be found when the number of generations is selected as 60. In addition, the value 60 may be considered as an optimal value when compared to the values 30 and 80, when both accuracy and time aspects are also considered. The accuracy value on selection of 60 is greater than the accuracy value on selection of 30. Similarly, execution time on selection of 60 is smaller than the execution time on selection of 80. As a result, the number of generations was selected as 60 for the study.

After choosing the population size and the number of generations, parameters for GA are set as given in Table 5.5. Other than the population size and the number of generations, default parameters were used in the algorithm, as they were defined in the MOEA Framework.

Table 5.5: Parameter configuration for genetic algorithm.

Parameter	Value
Population size	40
Number of generations	60
Crossover ratio	1.0
Mutation ratio	0.01

Table 5.6 presents the results of the machine learning techniques on feature subsets generated by GA. The maximum accuracy and the number of features at that maximum accuracy are given in this table. At first glance, the accuracy results of GA are better than the EA and are comparable with GR. Moreover, GA reduces the number of features as compared to GR. Execution times vary according to the total number of features of the dataset.

The experiment results in terms of execution times are compatible with the theoretical calculations given in Section 3.3, Figure 3.5. It is possible to examine all subsets of features of small datasets in an acceptable amount of time which makes the EA advantageous. For medium datasets, GR can be used to achieve a good enough solution in a reasonable amount of time. However, for large datasets, it is obvious that GA gives the best results in terms of execution time. A detailed comparison of all feature selection algorithms and machine learning algorithms with respect to accuracy, number of features and execution times is given in Section 5.5.

5.4.2 Improvement of Individuals in the Population through Generations

In this part of the experiments, to prove that evolutionary part of the GA works as intended, the solutions in the initial and final populations are compared. As explained in previous section, population size and number of generations were set as 40×60 respectively to be used in the experiments.

Datasets were categorized into three groups with respect to their number of features, as given in Table 5.7. One sample dataset from each group is randomly selected as representative: BC for small datasets, MR for medium datasets and SB for large datasets.

Table 5.6: Classification results of feature subsets generated by genetic algorithm.

Dataset ID	Total # of features	Classifier	Max. accuracy	# of features at max. accuracy	Exec. time (sec.)
CT	54	LR	0.775	14	341.2
		SVM	0.775	11	652.3
		ELM	0.683	8	343.8
		K-means	0.725	1	95.3
		AP	0.781	6	8047.2
MR	22	LR	0.949	9	102.5
		SVM	0.956	9	414.5
		ELM	0.990	3	112.6
		K-means	0.827	8	87.0
		AP	0.996	8	28562.7
SB	57	LR	0.921	20	356.3
		SVM	0.920	22	566.8
		ELM	0.901	12	409.3
		K-means	0.807	24	216.0
		AP	0.907	16	11979.4
NU	8	LR	1.000	1	6.0
		SVM	1.000	1	10.2
		ELM	1.000	1	17.6
		K-means	1.000	1	5.6
		AP	1.000	1	171.3
C4	42	LR	0.825	24	186.9
		SVM	0.826	24	2650.2
		ELM	0.800	16	425.6
		K-means	0.731	1	115.7
		AP	0.825	25	58104.0
WF	40	LR	0.923	11	42.3
		SVM	0.922	9	300.0
		ELM	0.904	5	224.4
		K-means	0.891	11	97.1
		AP	0.918	11	4560.1
FI	93	LR	0.966	3	606.6
		SVM	0.966	1	2307.2
		ELM	0.966	5	491.5
		K-means	0.966	2	264.9
		AP	0.968	18	24127.6
PM	8	LR	0.770	5	3.0
		SVM	0.769	6	20.9
		ELM	0.740	1	18.5
		K-means	0.736	4	5.7
		AP	0.792	5	142.1
BC	9	LR	0.963	5	4.0
		SVM	0.968	4	21.0
		ELM	0.962	3	22.6
		K-means	0.961	5	8.4
		AP	0.969	7	184.0
IO	34	LR	0.897	11	82.1
		SVM	0.897	10	179.2
		ELM	0.894	2	126.4
		K-means	0.856	2	28.0
		AP	0.915	9	500.9
WBC	30	LR	0.975	4	27.6
		SVM	0.978	10	147.3
		ELM	0.952	3	139.5
		K-means	0.917	4	31.4
		AP	0.929	5	548.9

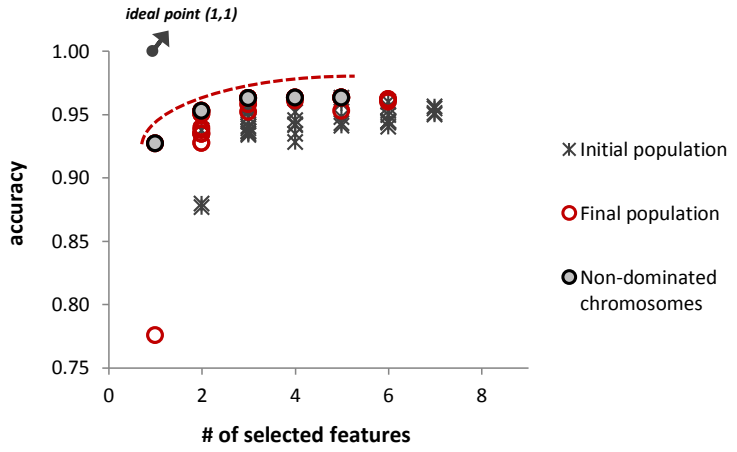
Table 5.7: Dataset categorization according to feature size.

Small	**Medium**	**Large**
NU	MR	CT
PM	WF	SB
BC	IO	C4
	WBC	FI

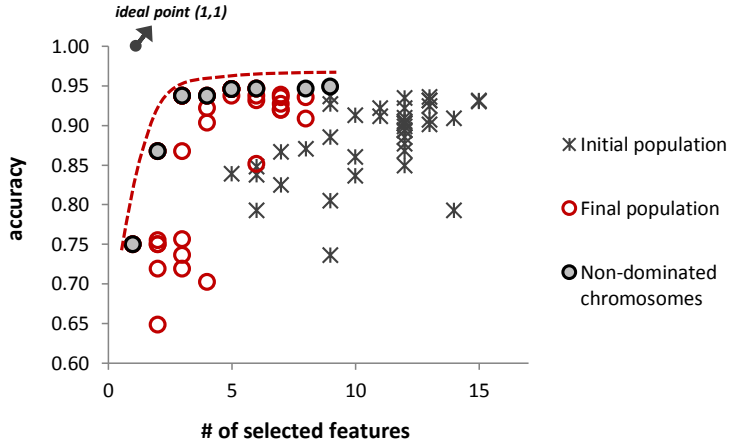
From Figure 5.2 to Figure 5.6, the distributions of initial and final solutions of GA solutions for the selected datasets evaluated by each classification technique are given. Non-dominated chromosomes shown in these figures are selected from the final population only. In order to prove that selection of machine learning technique is irrelevant in this process, figures of all three datasets evaluated by the same machine learning technique are given in one page each. It can be seen from the figures that the quality of solutions improves after all generations are populated when compared to the solutions of initial population. Initial population gives a picture of disorganized individuals but as the number of generations increases, the population tends to get closer to the hypothetical ideal point. In other words, the population evolve toward ideal point through generations. In the figures, the point $(1, 1)$ is shown as the hypothetical *ideal point* since the two objectives of this study are maximizing the accuracy and minimizing the number of features, both of which are 1 in their best cases.

Having less number of features, the number of non-dominated solutions is lower in the small dataset. Moreover, these solutions form a pareto curve like shape, as seen in the figures. On the other hand, the solutions tend to fit a pareto curve when the feature size of dataset gets larger.

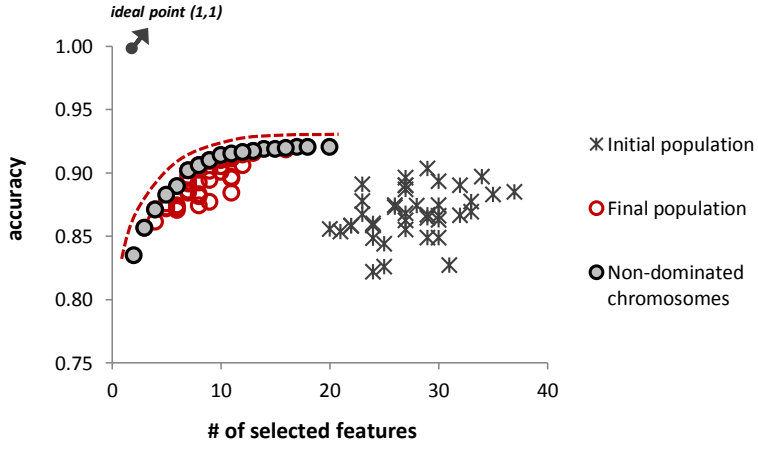
Even though the number of possible solutions is small for the small dataset and initial population is closer to the ideal point, the improvement of the solutions is still significant. This becomes even more apparent as the dataset gets larger.



(a) Distribution of solutions of GA on the BC dataset evaluated by LR.

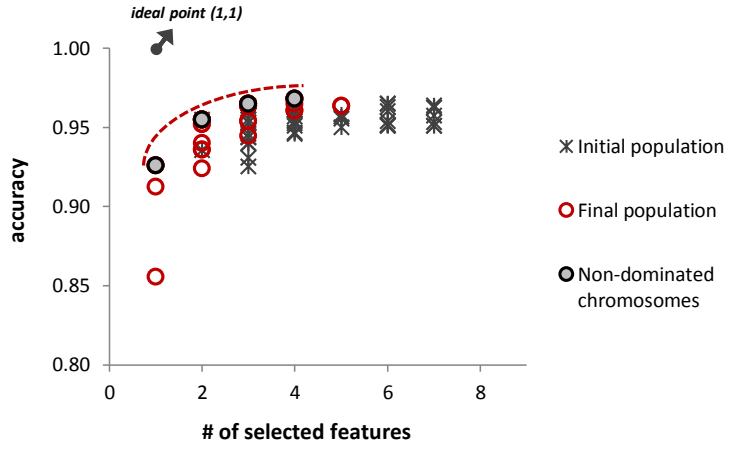


(b) Distribution of solutions of GA on the MR dataset evaluated by LR.

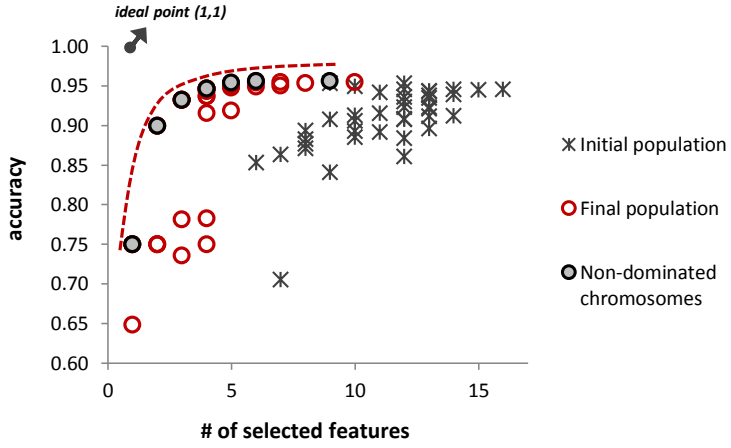


(c) Distribution of solutions of GA on the SB dataset evaluated by LR.

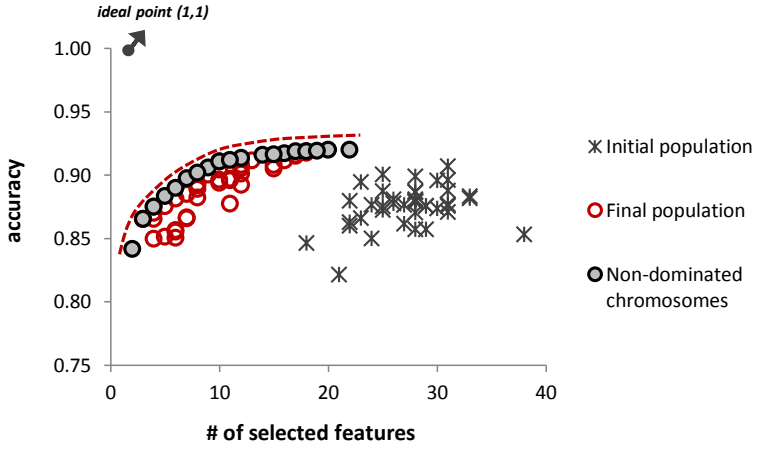
Figure 5.2: Distribution of genetic algorithm solutions evaluated by Logistic Regression.



(a) Distribution of solutions of GA on the BC dataset evaluated by SVM.

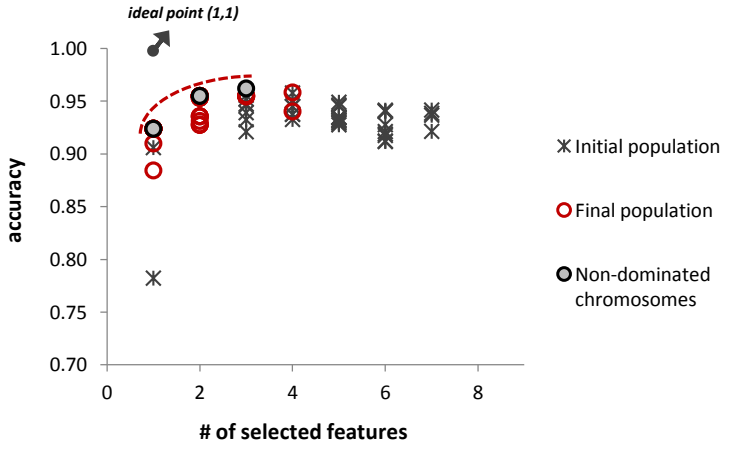


(b) Distribution of solutions of GA on the MR dataset evaluated by SVM.

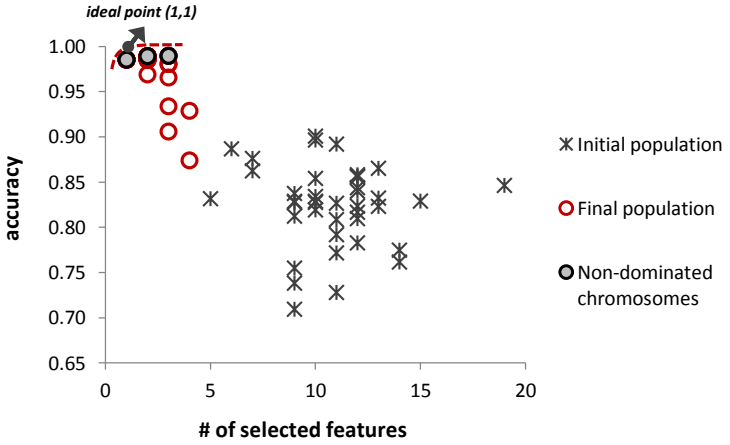


(c) Distribution of solutions of GA on the SB dataset evaluated by SVM.

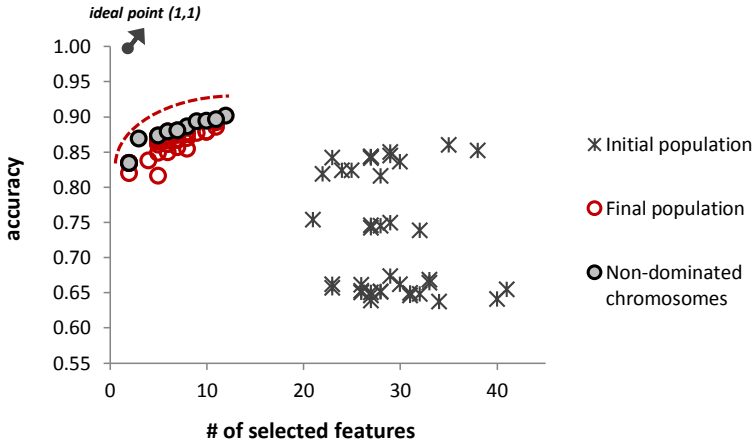
Figure 5.3: Distribution of genetic algorithm solutions evaluated by Support Vector Machines.



(a) Distribution of solutions of GA on the BC dataset evaluated by ELM.

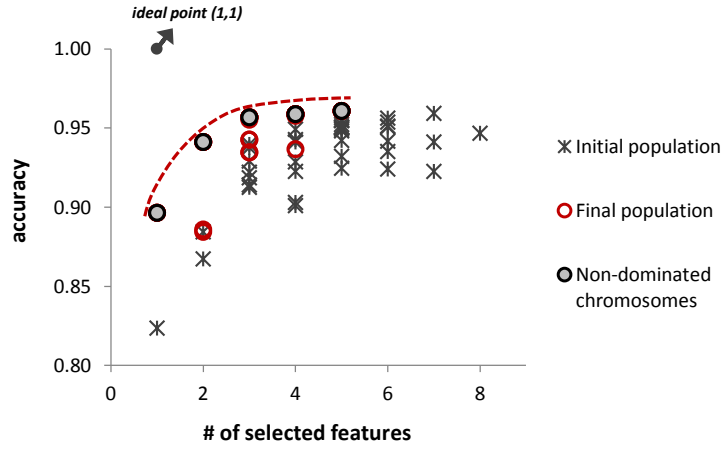


(b) Distribution of solutions of GA on the MR dataset evaluated by ELM.

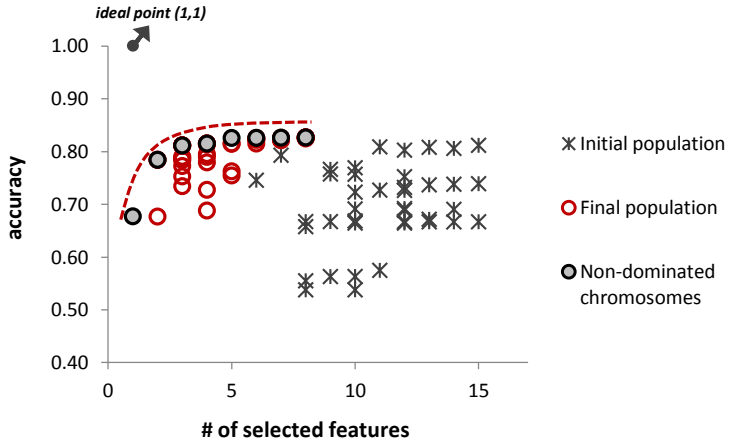


(c) Distribution of solutions of GA on the SB dataset evaluated by ELM.

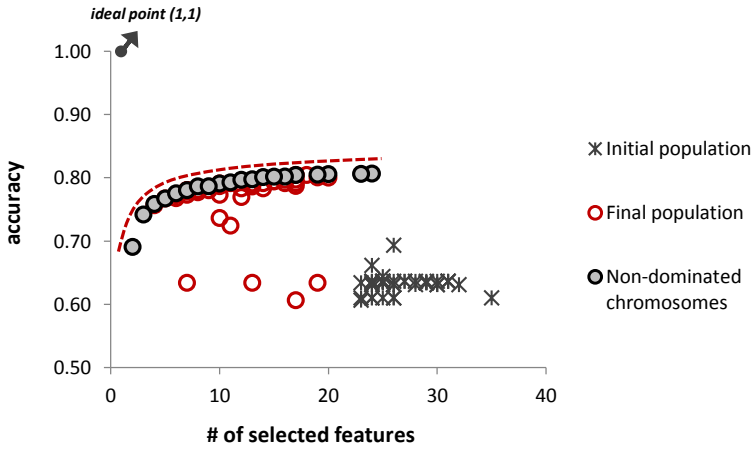
Figure 5.4: Distribution of genetic algorithm solutions evaluated by Extreme Learning Machine.



(a) Distribution of solutions of GA on the BC dataset evaluated by K-means.

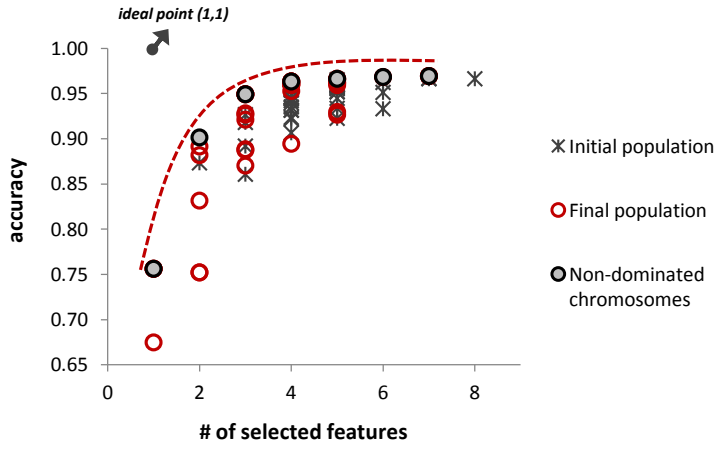


(b) Distribution of solutions of GA on the MR dataset evaluated by K-means.

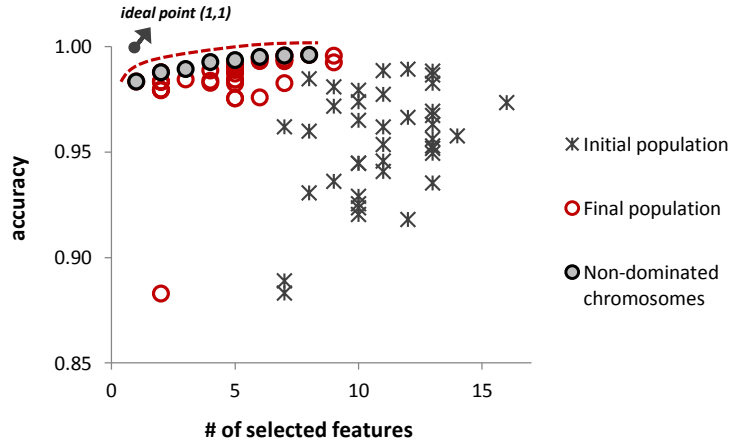


(c) Distribution of solutions of GA on the SB dataset evaluated by K-means.

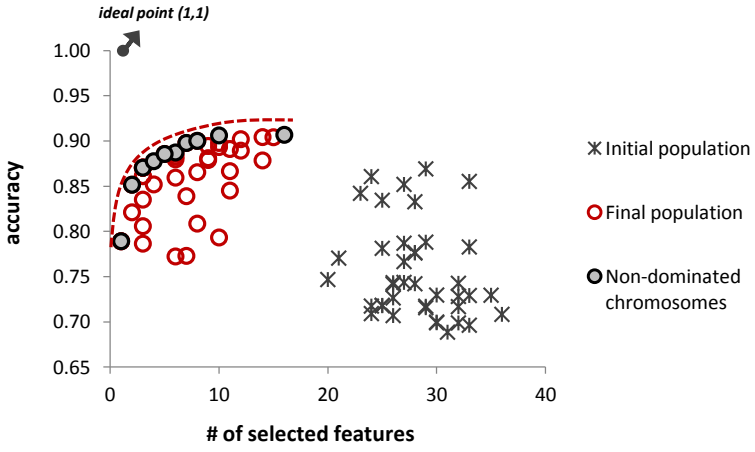
Figure 5.5: Distribution of genetic algorithm solutions evaluated by K-means.



(a) Distribution of solutions of GA on the BC dataset evaluated by AP.



(b) Distribution of solutions of GA on the MR dataset evaluated by AP.



(c) Distribution of solutions of GA on the SB dataset evaluated by AP.

Figure 5.6: Distribution of genetic algorithm solutions evaluated by Affinity Propagation.

5.5 Performance Comparison of the Algorithms

In this section, comparison of FSS algorithms and machine learning algorithms are discussed. Finally, the results are compared with a well-known filtering algorithm and other state-of-the-art algorithms in literature.

In Table 5.8, the classification accuracy values are given in a multiobjective fashion. Discussions of FSS and machine learning algorithms are given in following sections according to the values in this table. Distinct feature sizes (given as "F. size" in the tables) are extracted from the non-dominated solutions generated by GR and GA. Also, to give a baseline for each feature size, maximum EA values are added to the table, when applicable. Only accuracy values of feature sizes having both non-dominated GR and GA are given. Even though GR has accuracy values for every feature size, incomparable ones are omitted. Similarly, some of GA solutions that are not comparable with GR are not shown in the table. For every machine learning algorithm, GR and GA accuracy values are compared for every feature size and the higher ones are weighed with bold in the table. Note that LR, SVM and K-means are deterministic algorithms since they yield exact accuracy values in same conditions, whereas, ELM and AP are non-deterministic and accuracy of the same condition may vary. That is why GR or GA could find higher accuracy values than EA for ELM and AP.

Table 5.8: Solution sets of all feature selection algorithms evaluated by all machine learning algorithms for all datasets.

(**bold**: dominant solution)

(a) Solution sets of the CT dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	**GR**	**GA**	**EA**	**GR**	**GA**	**EA**	**GR**	**GA**	**EA**	**GR**	**GA**	**EA**	**GR**	**GA**
1	0.743	0.743	0.743	0.743	0.743	0.743	0.609	0.609	0.609	0.725	0.725	0.725	0.856	**0.855**	0.722
2	0.753	0.753	0.753	0.754	0.754	0.754	0.640	**0.651**	0.640	-	-	-	0.863	**0.858**	0.770
3	0.764	0.764	0.764	0.763	**0.763**	0.759	-	-	-	-	-	-	-	**0.860**	0.775
4	-	0.767	0.767	-	0.767	0.767	-	-	-	-	-	-	-	-	-
5	-	0.770	0.770	-	0.771	0.771	-	-	-	-	-	-	-	-	-
6	-	0.772	0.772	-	0.772	0.772	-	-	-	-	-	-	-	-	-
7	-	**0.773**	0.772	-	0.773	0.773	-	-	-	-	-	-	-	-	-
8	-	0.773	0.773	-	0.774	0.774	-	-	-	-	-	-	-	-	-
9	-	0.774	0.774	-	-	-	-	-	-	-	-	-	-	-	-
10	-	0.774	0.774	-	**0.775**	0.774	-	-	-	-	-	-	-	-	-
11	-	-	-	-	0.775	0.775	-	-	-	-	-	-	-	-	-
13	-	**0.775**	0.774	-	-	-	-	-	-	-	-	-	-	-	-
14	-	0.775	0.775	-	-	-	-	-	-	-	-	-	-	-	-

(b) Solution sets of the MR dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	0.763	**0.763**	0.750	0.750	0.750	0.750	0.985	0.985	0.985	0.773	**0.773**	0.677	0.983	0.971	**0.983**
2	0.905	0.864	**0.867**	0.899	0.826	**0.899**	0.990	0.989	0.989	0.821	**0.812**	0.784	0.979	0.979	**0.988**
3	0.937	0.891	**0.937**	0.937	0.861	**0.932**	0.990	**0.991**	0.990	-	-	-	-	-	-
4	0.940	0.937	0.937	0.946	0.894	**0.946**	-	-	-	-	-	-	-	-	-
5	0.949	0.937	**0.945**	-	0.937	**0.954**	-	-	-	-	-	-	-	-	-
6	0.952	0.938	**0.946**	-	0.937	**0.956**	-	-	-	-	-	-	-	-	-
8	-	0.940	**0.946**	-	-	-	-	-	-	-	0.825	**0.827**	-	-	-
9	-	0.945	**0.949**	-	-	-	-	-	-	-	-	-	-	-	-

(c) Solution sets of the SB dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	-	-	-	-	-	-	-	-	-	-	-	-	0.818	**0.820**	0.789
2	0.835	0.835	0.835	0.842	0.842	0.842	0.846	**0.846**	0.834	0.728	**0.728**	0.691	0.863	**0.863**	0.852
3	0.857	0.857	0.857	0.865	0.865	0.865	0.866	0.868	0.868	0.744	**0.744**	0.742	-	**0.874**	0.871
4	0.871	0.871	0.871	-	0.875	0.875	-	-	-	-	0.759	0.759	-	0.878	0.878
5	-	0.883	0.883	-	0.883	0.883	-	0.870	**0.873**	-	0.767	0.767	-	0.881	**0.886**
6	-	0.889	**0.890**	-	0.890	0.890	-	0.877	**0.879**	-	0.776	0.776	-	0.883	**0.887**
7	-	0.898	**0.902**	-	0.897	0.897	-	0.878	**0.880**	-	0.781	0.781	-	0.890	**0.898**
8	-	0.905	**0.906**	-	0.902	0.902	-	0.884	**0.886**	-	0.787	0.787	-	0.894	**0.900**
9	-	0.910	0.910	-	0.906	0.906	-	0.889	**0.893**	-	**0.790**	0.787	-	-	-
10	-	0.914	0.914	-	0.911	0.911	-	0.891	**0.894**	-	**0.793**	0.791	-	0.899	**0.906**
11	-	0.915	0.915	-	0.912	0.912	-	0.894	**0.896**	-	**0.796**	0.793	-	-	-
12	-	**0.917**	0.916	-	0.913	0.913	-	0.897	**0.901**	-	**0.798**	0.797	-	-	-
13	-	**0.918**	0.917	-	-	-	-	-	-	-	**0.799**	0.798	-	-	-
14	-	0.919	0.919	-	0.916	0.916	-	-	-	-	**0.802**	0.801	-	-	-
15	-	0.919	0.919	-	**0.918**	0.916	-	-	-	-	**0.803**	0.802	-	-	-
16	-	0.920	0.920	-	**0.918**	0.917	-	-	-	-	**0.804**	0.802	-	-	-
17	-	0.920	0.920	-	**0.920**	0.918	-	-	-	-	**0.806**	0.805	-	-	-
18	-	**0.921**	0.920	-	**0.921**	0.919	-	-	-	-	-	-	-	-	-
19	-	-	-	-	**0.921**	0.919	-	-	-	-	**0.809**	0.805	-	-	-
20	-	-	-	-	**0.921**	0.920	-	-	-	-	**0.810**	0.806	-	-	-
22	-	-	-	-	**0.921**	0.920	-	-	-	-	-	-	-	-	-
23	-	-	-	-	-	-	-	-	-	-	**0.812**	0.806	-	-	-
24	-	-	-	-	-	-	-	-	-	-	**0.813**	0.807	-	-	-

(d) Solution sets of the NU dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

(e) Solution sets of the C4 dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	-	-	-	0.730	**0.730**	0.729	0.730	**0.731**	0.730	0.731	0.731	0.731	-	-	-
2	-	-	-	0.737	0.730	**0.733**	0.746	0.738	**0.744**	-	-	-	0.908	**0.902**	0.738
3	-	-	-	-	-	-	0.753	0.748	**0.753**	-	-	-	-	-	-
4	0.765	0.747	**0.765**	-	0.730	**0.757**	-	0.761	**0.763**	-	-	-	-	-	-
5	-	0.756	**0.772**	-	0.730	**0.765**	-	0.769	0.769	-	-	-	-	-	-
6	-	0.766	**0.778**	-	-	-	-	0.774	**0.775**	-	-	-	-	-	-
7	-	0.772	**0.785**	-	-	-	-	0.780	**0.781**	-	-	-	-	-	-
8	-	0.778	**0.791**	-	-	-	-	0.783	**0.786**	-	-	-	-	-	-
9	-	0.784	**0.796**	-	0.730	**0.794**	-	**0.791**	0.789	-	-	-	-	**0.903**	0.774
10	-	0.791	**0.800**	-	0.738	**0.800**	-	0.794	**0.795**	-	-	-	-	**0.922**	0.784
11	-	0.796	**0.802**	-	0.742	**0.801**	-	-	-	-	-	-	-	-	-
12	-	0.802	**0.807**	-	0.753	**0.804**	-	**0.797**	0.796	-	-	-	-	-	-
13	-	0.805	**0.810**	-	0.769	**0.808**	-	**0.804**	0.796	-	-	-	-	-	-
14	-	0.810	**0.812**	-	0.781	**0.811**	-	-	-	-	-	-	-	-	-
15	-	0.814	**0.815**	-	0.794	**0.816**	-	-	-	-	-	-	-	-	-
16	-	0.817	**0.819**	-	0.800	**0.819**	-	-	-	-	-	-	-	-	-
17	-	0.821	0.821	-	0.802	**0.821**	-	-	-	-	-	-	-	-	-
18	-	**0.825**	0.822	-	-	-	-	-	-	-	-	-	-	-	-
19	-	**0.827**	0.822	-	0.812	**0.822**	-	-	-	-	-	-	-	-	-
20	-	**0.829**	0.823	-	0.816	**0.822**	-	-	-	-	-	-	-	-	-
21	-	**0.830**	0.823	-	-	-	-	-	-	-	-	-	-	-	-
22	-	**0.831**	0.824	-	0.823	**0.825**	-	-	-	-	-	-	-	-	-
23	-	**0.833**	0.824	-	0.823	**0.826**	-	-	-	-	-	-	-	-	-
24	-	**0.834**	0.825	-	0.825	**0.826**	-	-	-	-	-	-	-	-	-

(f) Solution sets of the WF dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	-	-	-	0.806	**0.806**	0.791	0.798	0.796	**0.797**	0.804	**0.804**	0.793	0.798	0.798	0.798
2	0.868	0.857	**0.868**	0.869	0.855	**0.869**	0.866	0.854	**0.869**	0.852	0.840	**0.852**	0.859	0.859	0.859
3	0.893	0.887	**0.893**	0.893	0.887	**0.893**	0.889	0.882	**0.890**	0.875	0.875	0.875	-	0.882	0.882
4	0.902	0.901	**0.902**	0.904	0.904	0.904	0.900	0.898	**0.901**	0.888	0.882	0.882	-	0.900	0.900
5	0.915	0.910	**0.915**	-	0.910	0.914	-	0.902	**0.904**	-	**0.889**	0.887	-	0.906	0.906
6	-	0.917	0.917	-	0.917	0.917	-	-	-	-	**0.890**	0.889	-	0.910	0.910
7	-	0.919	0.919	-	0.918	0.918	-	-	-	-	**0.891**	0.890	-	0.912	**0.914**
8	-	0.921	0.921	-	0.921	0.921	-	-	-	-	**0.891**	0.890	-	0.912	**0.916**
9	-	0.921	**0.922**	-	0.922	0.922	-	-	-	-	-	-	-	-	-
10	-	0.922	**0.923**	-	-	-	-	-	-	-	**0.893**	0.890	-	-	-
11	-	0.923	0.923	-	-	-	-	-	-	-	**0.894**	0.891	-	0.913	**0.918**

(g) Solution sets of the FI dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	0.966	0.966	0.966	0.966	0.966	0.966	0.966	0.966	0.966	0.966	0.966	0.966	-	-	-
2	-	-	-	-	-	-	-	-	-	0.966	0.966	0.966	-	**0.970**	0.966
3	0.967	0.966	0.966	-	-	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.970**	0.968
18	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.970**	0.968

(h) Solution sets of the PM dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	0.747	0.747	0.747	0.747	0.747	0.747	0.739	0.735	**0.740**	0.728	0.728	0.728	0.714	**0.714**	0.712
2	0.760	0.760	0.760	0.760	0.760	0.760	-	-	-	0.732	0.732	0.732	0.777	0.776	**0.777**
3	0.766	0.766	0.766	0.765	0.765	0.765	-	-	-	0.735	0.735	0.735	0.784	0.781	**0.784**
4	0.768	0.768	0.768	0.766	0.766	0.766	-	-	-	0.736	0.736	0.736	-	-	-
5	0.771	0.770	0.770	0.768	0.768	0.768	-	-	-	-	-	-	0.792	0.792	0.792
6	-	-	-	0.769	0.769	0.769	-	-	-	-	-	-	-	-	-

(i) Solution sets of the BC dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	0.927	0.927	0.927	0.926	0.926	0.926	0.925	0.924	0.924	0.896	0.896	0.896	0.792	**0.818**	0.756
2	0.953	0.953	0.953	0.955	0.955	0.955	0.955	0.954	0.955	0.941	0.941	0.941	0.899	0.845	**0.902**
3	0.963	0.958	**0.963**	0.965	0.959	**0.965**	0.963	**0.964**	0.962	0.957	0.957	0.957	0.951	0.933	**0.949**
4	0.963	0.961	**0.963**	0.968	0.965	**0.968**	-	-	-	0.959	0.959	0.959	0.963	0.944	**0.963**
5	0.963	0.962	**0.963**	-	-		-	-	-	0.961	0.961	0.961	0.966	0.961	**0.966**
6	-	-	-	-	-	-	-	-	-	-	-	-	0.968	0.963	**0.968**
7	-	-	-	-	-	-	-	-	-	-	-	-	0.968	0.966	**0.969**

(j) Solution sets of the IO dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	-	-	-	0.811	0.811	0.811	0.818	0.816	**0.819**	0.835	0.835	0.835	0.782	**0.782**	0.689
2	0.872	0.872	0.872	0.864	0.864	0.864	0.892	0.889	**0.894**	0.856	0.856	0.856	0.846	**0.839**	0.773
3	0.876	0.876	0.876	0.873	0.873	0.873	-	-	-	-	-	-	0.894	**0.894**	0.858
4	0.883	0.878	0.878	0.878	0.877	0.877	-	-	-	-	-	-	-	**0.896**	0.890
5	0.888	0.883	**0.886**	-	0.879	**0.883**	-	-	-	-	-	-	-	-	-
6	-	0.889	0.889	-	0.88	**0.888**	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-	-	-	-	0.905	**0.906**
8	-	0.891	**0.892**	-	0.885	**0.892**	-	-	-	-	-	-	-	**0.911**	0.909
9	-	-	-	-	0.887	**0.894**	-	-	-	-	-	-	-	-	-
11	-	0.892	**0.897**	-	-	-	-	-	-	-	-	-	-	-	-

(k) Solution sets of the WBC dataset.

F. size	LR			SVM			ELM			K-means			AP		
	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA	EA	GR	GA
1	0.920	0.920	0.920	0.921	**0.921**	0.919	0.918	**0.917**	0.915	0.901	0.901	0.901	0.891	**0.891**	0.875
2	0.961	0.961	0.961	0.960	0.960	0.960	0.947	**0.947**	0.946	0.915	0.915	0.915	0.911	**0.911**	0.908
3	0.972	0.968	**0.972**	0.970	**0.970**	0.969	0.954	**0.953**	0.952	0.916	0.916	0.916	0.919	**0.919**	0.914
4	0.975	0.974	**0.975**	0.975	0.974	**0.975**	-	-	-	-	-	-	-	0.921	**0.928**
5	-	-	-	-	-	-	-	-	-	-	-	-	-	0.923	**0.929**
6	-	-	-	-	**0.977**	0.975	-	-	-	-	-	-	-	-	-

5.5.1 Comparison of Feature Selection Algorithms

In this section, the performance of GR and GA are compared in a multiobjective fashion, while EA provides a baseline. With the use of EA, 152 baseline subsets were found in the above given subtables of Table 5.8. GR could only find 95 (62.5%) of these subsets, whereas GA could reach up to 107 (70.4%). GR performs well, yet GA performs even better.

When GR and GA are compared one-by-one for each machine learning technique at each feature size, there exists 326 direct comparisons. Among these 326 comparisons, GA dominates GR for 115 times (35%), and GR dominates GA only for 84 times (26%). GA and GR tie on the remaining 127 comparisons (39%).

A more detailed analysis shows that, GR dominates GA for only 38 times when supervised techniques are employed, whereas it is dominated by GA for 91 times. Moreover, among these dominations, GR cannot even reach up to half of GA for any of these methods. These results indicate that GR is no match for GA when supervised techniques are used, regardless of which method is selected. On the other hand, the use of unsupervised techniques tells another story. When K-means is used for classification, GR dominates GA for 24 times and is dominated for 2 times only. The use of AP is more balanced with 22 and 21 dominations in favor of GR and GA, respectively. Nevertheless, GR is better with both unsupervised techniques. On the other hand, accuracy values achieved by GR with K-means are generally dominated by of those achieved by GA with supervised techniques, only 6 of them remained non-dominated. This number is higher for GR with AP, 20 out of 52 are non-dominated by GA with supervised techniques. However, execution time of AP is much larger than all other machine learning techniques, which will be discussed in Section 5.5.2, and hence it is not a preferable method. As a result, it is proved that GA is more preferable than other FSS algorithms.

5.5.2 Comparison of Machine Learning Algorithms

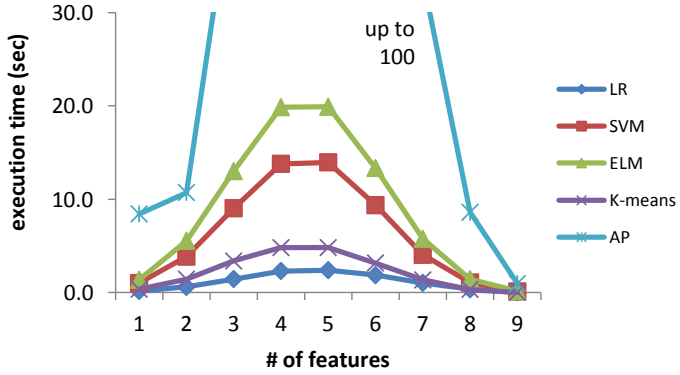
In this section, the performance of machine learning techniques are compared in a multiobjective fashion. In previous section, GA is proved to be the best FSS algo-

rithm, and hence its results are used in performance comparisons of machine learning techniques.

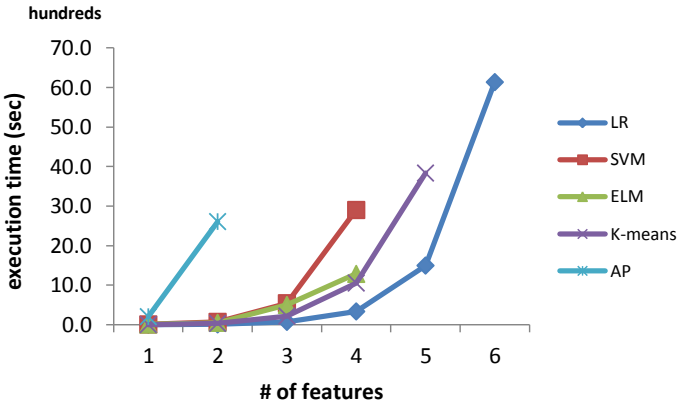
In previous section, baseline results obtained from EA, were given in a cumulative form. Here, they are broken down to show the performance of each machine learning technique. Of the 152 baseline subsets; LR could find 29 out of 38 (76.3%), SVM could find 27 out of 35 (77.1%), ELM could find 16 out of 25 (64%), K-means could find 21 out of 27 (77.8%), and AP could find 13 out of 27 (48.1%). It can be seen that all techniques achieve similar success ratios in finding baseline subsets, only ELM and AP are lower. There may be several reasons for that. First, even though the number of times ELM could not find baseline subsets (9) is comparable with LR (9) and SVM (8), its ratio drops more than others since ELM could find less number of non-dominated solutions in total (25). Second, non-deterministic nature of both techniques could harm this ratio by evaluating a lower accuracy for the same baseline subset generated by different feature selection algorithms.

In order to give a more detailed analysis on machine learning techniques, they are compared one-by-one for each feature size when there exists at least one comparison for that feature size. Methods are compared separately as supervised and unsupervised. According to comparison results of supervised methods, LR, with a total domination of 42 out of 69 (60.9%), outperformed both SVM and ELM, 24 (34.8%) and 11 (15.9%) out of 69 respectively. Total count of dominations (77) is higher than total number of comparisons (69). The reason for this, two methods non-dominated to each other, dominated the third method in some cases. Since the domination is significant, it was counted for each method in calculation of domination results. According to comparison results of unsupervised methods, AP, with a total domination of 23 out of 33 (69.7%), outperformed K-means having 10 dominations out of 33 (30.3%).

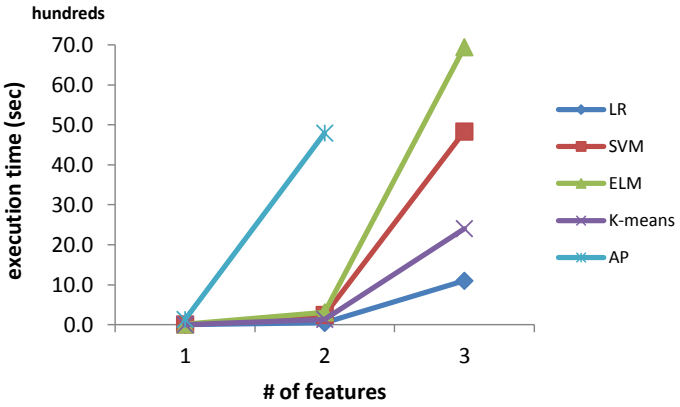
Figure 5.7 compares execution times of machine learning algorithms on the three selected datasets when EA is employed. Since EA works on a timely manner and it can only explore a small search space, execution times of GR and GA are not given in the same figure. According to all three subfigures, LR is the most time efficient technique, whereas AP is the most time consuming one. K-means, SVM and ELM lay in-between. Note that LR as a supervised technique and K-means as a unsupervised



(a) Execution times of machine learning algorithms for the BC dataset.



(b) Execution times of machine learning algorithms for the MR dataset.



(c) Execution times of machine learning algorithms for the SB dataset.

Figure 5.7: Execution times of machine learning algorithms applied on three datasets with exhaustive algorithm.

technique could explore more feature subsets than other techniques could explore in the same amount of time.

Figure 5.8 shows total execution time of each machine learning algorithm working with GR and GA on all 11 datasets. Similarly, LR and K-means are the most time efficient algorithms among supervised and unsupervised techniques, respectively. In total, GA works faster than GR when LR, SVM and K-means are employed. On the other hand, GR works slightly and significantly faster than GA when ELM and AP are employed, separately. This, again, may be the result of non-deterministic nature of these algorithms, since GA adopts elitism which relies on reliability of the results. AP, even though achieving higher accuracy values than K-means, is shown to be an unpreferable machine learning technique with its enormous execution time values in all FSS algorithms.

To sum up, GA was found as a superior FSS algorithm as compared to GR in the previous section. Also it was found that, GA performs better with supervised techniques. In this section, LR was found as the best supervised technique in terms of both accuracy and execution time.

As a result, GA with LR is called as the proposed algorithm in the following sections of this study.

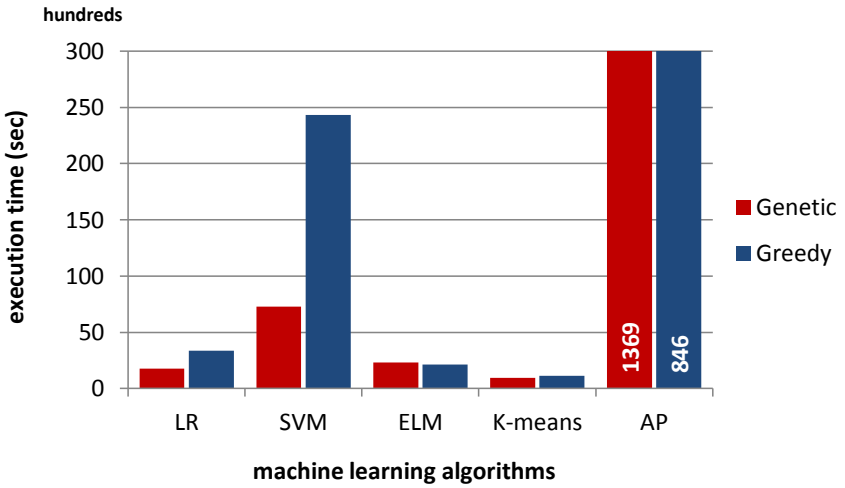


Figure 5.8: Comparison of total execution times of machine learning algorithms applied on all datasets with the greedy and genetic algorithms.

5.5.3 Comparison with a Filtering Algorithm

In this section, the performance of the proposed algorithm (GA with LR) is compared against Information Gain (IG), a well-known filtering algorithm [34]. As described in Chapter 3, wrapper methods utilize accuracy results when searching to find the best subset and the proposed algorithm is an example of them. IG, on the other hand, is an example of filter method which ranks features of a dataset by their contribution to the structure of dataset.

WEKA library [2] is utilized to apply IG on the datasets. For every dataset, first, the features are ranked with IG by their importance. Then, for every feature size, feature subsets are generated by choosing the most valuable features up to that size. Finally, generated subsets are evaluated by using LR. Table 5.9 compares accuracy results of IG subsets having same number of features with non-dominated GA solutions, which are both evaluated by LR.

In almost all cases, GA performs better than IG. In a total of 95 comparisons, GA outperforms IG for 88 times and they tie on the remaining 7 comparisons.

As a result, it is obvious that selecting the most valuable k features may not constitute the best feature subset having k features for this problem. IG may have worked better if FSS were a monotonic increasing problem in which increasing feature size would not decrease the accuracy value.

5.5.4 Comparison with State-Of-The-Art Algorithms

Table 5.10 presents the results of the proposed algorithm (GA with LR) and other state-of-the-art algorithms in literature that utilized same datasets with this study. Unler et al. [42] propose a Particle Swarm Optimization (PSO) based algorithm and evaluate the algorithm with all 11 datasets used in this study. Pacheco et al. [33] propose a Tabu Search (TS) method for the feature selection part and compares its results with two different greedy based algorithms, Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS), using 7 of the datasets used in this study.

[2] WEKA, `http://www.cs.waikato.ac.nz/ml/weka/`.

Table 5.9: Multiobjective comparison of the proposed algorithm and a filtering algorithm (Information Gain).

(**bold**: dominant solution)

(a) C4

F. size	IG	GA
4	0.753	**0.765**
5	0.763	**0.772**
6	0.776	**0.778**
7	0.785	0.785
8	0.784	**0.791**
9	0.790	**0.796**
10	0.794	**0.800**
11	0.793	**0.802**
12	0.796	**0.807**
13	0.798	**0.810**
14	0.802	**0.812**
15	0.808	**0.815**
16	0.809	**0.819**
17	0.811	**0.821**
18	0.813	**0.822**
19	0.814	**0.822**
20	0.813	**0.823**
21	0.813	**0.823**
22	0.814	**0.824**
23	0.817	**0.824**
24	0.817	**0.825**

(b) SB

F. size	IG	GA
2	0.819	**0.835**
3	0.831	**0.857**
4	0.860	**0.871**
5	0.865	**0.883**
6	0.868	**0.890**
7	0.870	**0.902**
8	0.872	**0.906**
9	0.873	**0.910**
10	0.885	**0.914**
11	0.890	**0.915**
12	0.901	**0.916**
13	0.903	**0.917**
14	0.903	**0.919**
15	0.902	**0.919**
16	0.903	**0.920**
17	0.903	**0.920**
18	0.904	**0.920**
20	0.907	**0.921**

(c) CT

F. size	IG	GA
1	0.743	0.743
2	0.747	**0.753**
3	0.748	**0.764**
4	0.747	**0.767**
5	0.754	**0.770**
6	0.753	**0.772**
7	0.755	**0.772**
8	0.756	**0.773**
9	0.756	**0.774**
10	0.756	**0.774**
13	0.754	**0.774**
14	0.755	**0.775**

(d) WF

F. size	IG	GA
2	0.857	**0.868**
3	0.875	**0.893**
4	0.895	**0.902**
5	0.906	**0.915**
6	0.911	**0.917**
7	0.914	**0.919**
8	0.914	**0.921**
9	0.914	**0.922**
10	0.912	**0.923**
11	0.920	**0.923**

(e) IO

F. size	IG	GA
2	0.812	**0.872**
3	0.813	**0.876**
4	0.811	**0.878**
5	0.810	**0.886**
6	0.799	**0.889**
8	0.822	**0.892**
9	0.815	**0.895**
10	0.812	**0.896**
11	0.808	**0.897**

(f) MR

F. size	IG	GA
1	0.589	**0.750**
2	0.848	**0.867**
3	0.747	**0.937**
4	0.744	**0.937**
5	0.744	**0.945**
6	0.747	**0.946**
8	0.825	**0.946**
9	0.890	**0.949**

(g) BC

F. size	IG	GA
1	0.927	0.927
2	0.938	**0.953**
3	0.954	**0.963**
4	0.955	**0.963**
5	0.954	**0.963**

(h) PM

F. size	IG	GA
1	0.747	0.747
2	0.760	0.760
3	0.759	**0.766**
4	0.758	**0.768**
5	0.757	**0.770**

(i) WBC

F. size	IG	GA
1	0.918	**0.920**
2	0.918	**0.961**
3	0.919	**0.972**
4	0.948	**0.975**

(j) FI

F. size	IG	GA
1	0.966	0.966
3	0.965	**0.966**

(k) NU

F. size	IG	GA
1	1.000	1.000

Lopez et al. [28] utilize Scatter Search (SS) algorithm to propose three different heuristics, Sequential SS with Greedy Combination (SSS-GC), Sequential SS with Reduced Greedy Combination (SSS-RGC), and Parallel SS (PSS). Unler et al. and Pacheco et al. use LR as classifier, whereas Lopez et al. use three different classifiers, IB1, Naive Bayes, and C4.5. The values of Lopez et al. in Table 5.10 are obtained from the IB1 classifier. Since they all use supervised machine learning algorithms for classification and proposed algorithm is evaluated by LR, which is also supervised machine learning algorithm, comparing the results is reasonable.

The comparison between the proposed algorithm and the other state-of-the-art algorithms is given in a multiobjective fashion in Table 5.10. The values of the proposed algorithm are chosen from its set of non-dominated solutions. The values of the other algorithms are chosen from the solution sets having maximum accuracy. The fonts of the results having both higher accuracy and lower number of features are weighted with bold as a sign of domination. However, if a dominant result does not exist for a dataset, all the non-dominated results are underlined. The proposed algorithm outperforms all other algorithms providing the best multiobjective solutions in 8 datasets out of 11. In remaining three datasets, MR, C4, and PM, the proposed algorithm is not dominated by any other algorithm. The proposed algorithm is non-dominated with both PSO and TS in MR, both PSO and SBS in C4, and PSO in PM.

As a result, proposed algorithm generates comparable solutions, if not better, when compared to mentioned state-of-the-art algorithms.

Table 5.10: Multiobjective comparison of the proposed algorithm and state-of-the-art algorithms in literature.

(**bold**: dominant solution, underline: non-dominated solution)

Dataset ID	Proposed Algorithm		Unler et al. [42]		Pacheco et al. [33]						Lopez et al. [28]					
			PSO		TS		SFS		SBS		SSS-GC		SSS-RGC		PSS	
	accuracy	# of features	accuracy	# of features	accuracy	# of features	accuracy	# of features	accuracy	# of features	accuracy	# of features	accuracy	# of features	accuracy	# of features
CT	**0.770**	**5**	0.770	7	0.755	7	0.764	5	0.761	7	-	-	-	-	-	-
MR	0.867	2	1.000	3	1.000	5	0.860	3	0.869	3	-	-	-	-	-	-
SB	**0.906**	**8**	0.902	8	0.900	8	0.879	8	0.876	8	-	-	-	-	-	-
NU	**1.000**	**1**	1.000	3	1.000	3	1.000	3	1.000	3	-	-	-	-	-	-
C4	0.802	11	0.813	12	0.791	12	0.782	11	0.749	7	-	-	-	-	-	-
WF	**0.915**	**5**	0.906	7	0.903	7	0.899	5	0.899	5	-	-	-	-	-	-
FI	**0.966**	**1**	0.882	8	0.879	3	0.873	3	0.873	5	-	-	-	-	-	-
PM	0.768	4	0.774	6	-	-	-	-	-	-	0.679	4.1	0.677	4.0	0.681	4.2
BC	**0.963**	**3**	0.962	4	-	-	-	-	-	-	0.952	5.2	0.949	4.8	0.951	5.4
IO	**0.878**	**4**	0.862	4	-	-	-	-	-	-	0.878	6.1	0.871	5.7	0.874	3.9
WBC	**0.975**	**4**	0.963	7	-	-	-	-	-	-	0.947	6.8	0.936	5.5	0.937	6.0

CHAPTER 6

CONCLUSION AND FUTURE WORK

The use of previous knowledge is crucial when making a decision. The amount of data generated in one day increases massively. This increase could lead to vital improvement in decision-making when in use by expert decision-makers. However an effective decision-making relies on the quality of information and extracting valuable information from huge databases is an intractable process. Machine learning techniques provide tools that can analyze large amounts of data in a reasonable time. On the other hand, these tools may be insufficient especially when the number of features in the data reaches to very large numbers, e.g. hundreds of thousands for text categorization domain. Moreover, having too much features could cause overfitting problem in classification, which is more memorizing than learning. Therefore, selecting most valuable features, feature subset selection (FSS), becomes indispensable in such cases. In FSS, keeping the classification accuracy at a comparable level with the original data is as much important as decreasing the number of features. With respect to this information, a multiobjective approach should be employed for this problem. These objectives are minimizing the number of features and maximizing the classification accuracy.

In this study, three different FSS algorithms, exhaustive (EA), greedy (GR) and genetic (GA), are implemented for the feature selection part and their performances are evaluated by five different machine learning algorithms, Logistic Regression (LR), Support Vector Machines (SVM), Extreme Learning Machine (ELM), K-means, and Affinity Propagation (AP). In the experimental part of the study, 11 publicly available datasets are used in the performance evaluation of the mentioned algorithms.

Table 6.1: The effect of feature subset selection on classification performance.

Dataset ID	Before FSS		After FSS	
	accuracy	**# of features**	**accuracy**	**# of features**
CT	0.761	54	0.775	14
MR	0.937	22	0.949	9
SB	0.893	57	0.921	20
NU	1.000	8	1.000	1
C4	0.820	42	0.825	24
WF	0.893	40	0.923	11
FI	0.909	93	0.966	3
PM	0.762	8	0.770	5
BC	0.954	9	0.963	5
IO	0.812	34	0.897	11
WBC	0.924	30	0.975	4

Experiment results show that FSS improves classification performance by finding the most valuable subset of features and hence reducing the dimension of the data. For all datasets, the classification accuracy values having all features selected were obtained using LR and the results were given in Table 5.2 in previous chapter. The accuracy values of the feature subsets obtained from GA using LR were also given in Table 5.6. The values before FSS and after FSS are combined in Table 6.1 to show the effectiveness of FSS in terms of accuracy and feature size of the dataset. As seen in the table, FSS reduces the number of features tremendously for all datasets. At the same time, for all datasets, accuracy values are increased considerably after FSS. For example, the dataset FI has 93 features and a classification accuracy of 0.909 before FSS. After finding the most valuable 3 features with FSS, classification accuracy reaches up to 0.963. Since the required time to get a classification result is highly correlated with the number of features, FSS decreases the execution time by discarding 90 unnecessary features.

In more comprehensive comparisons, GA outperformed GR when supervised machine learning techniques are employed, by dominating GR in 91 of the comparisons and being dominated in only 38 of them. Among the supervised techniques, LR was found by far the best supervised machine learning technique, as compared to both SVM and ELM. LR dominated for a total of 42 times, whereas domination numbers for SVM and ELM are 24 and 11, respectively. Moreover, LR was found as the most

efficient machine learning technique in execution time comparisons. As a result, best performing algorithm is proved as GA with LR in this study.

To verify efficiency of the multiobjective evolutionary algorithm (GA), the best performing machine learning algorithm (LR) results are compared with the state-of-the-art Particle Swarm Optimization, Tabu Search, Greedy Search and Scatter Search algorithms in literature. GA outperforms all other algorithms in 8 of the datasets, and generates solutions that are non-dominated for the remaining 3 datasets.

To sum up, EA is preferable for small datasets, since it may check every possible feature subsets in a reasonable time. However, it is impossible to employ a complete search of EA in medium to large datasets. In medium sized datasets, GR can achieve good enough solutions in a small amount of time. GA can achieve better solutions, yet it requires more time. In large datasets, however, GA outperforms GR with respect to both accuracy values and execution times.

A possible future work can be training and testing different datasets and comparing their results with other state-of-the-art algorithms. Furthermore, different machine learning algorithms can be utilized to evaluate the performance of the feature selection algorithms. Another future work can be using parallel programming on this problem. It would most likely decrease the execution time of both feature selection and machine learning algorithms. Finally, initial population of GA can be organized intelligently instead of randomization. For example, a short explorization with exhaustive approach can reveal some valuable feature subsets to be used as the initial population.

REFERENCES

[1] E. Alpaydin. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2014.

[2] M. W. Aslam. Selection of fitness function in genetic programming for binary classification. In *Science and Information Conference (SAI), 2015*, pages 489–493. IEEE, 2015.

[3] C. M. Bishop. Pattern recognition. *Machine Learning*, 2006.

[4] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997.

[5] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99. Citeseer, 1998.

[6] L. Cervante, B. Xue, M. Zhang, and L. Shang. Binary particle swarm optimisation for feature selection: A filter based approach. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[7] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[8] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[9] M. Dash and H. Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[11] D. Dueck and B. J. Frey. Non-metric affinity propagation for unsupervised image categorization. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[12] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[13] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[14] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[15] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28, 1998.

[16] S. C. Hoi, J. Wang, P. Zhao, and R. Jin. Online feature selection for mining big data. In *Proceedings of the 1st international workshop on big data, streams and heterogeneous source mining: Algorithms, systems, programming models and applications*, pages 93–100. ACM, 2012.

[17] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. 2003.

[18] C.-L. Huang and C.-J. Wang. A ga-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with applications*, 31(2):231–240, 2006.

[19] G.-B. Huang, D. H. Wang, and Y. Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107–122, 2011.

[20] I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra. Feature subset selection by bayesian network-based optimization. *Artificial intelligence*, 123(1):157–184, 2000.

[21] G. H. John, R. Kohavi, K. Pfleger, et al. Irrelevant features and the subset selection problem. In *Machine learning: proceedings of the eleventh international conference*, pages 121–129, 1994.

[22] H. Kazemian and S. Ahmed. Comparisons of machine learning techniques for detecting malicious webpages. *Expert Systems with Applications*, 42(3):1166–1177, 2015.

[23] A. Khan and A. R. Baig. Multi-objective feature subset selection using non-dominated sorting genetic algorithm. *Journal of applied research and technology*, 13(1):145–159, 2015.

[24] M. Y. Kiang. A comparative assessment of classification methods. *Decision Support Systems*, 35(4):441–454, 2003.

[25] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

[26] H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012.

[27] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502, 2005.

[28] F. G. López, M. G. Torres, B. M. Batista, J. A. M. Pérez, and J. M. Moreno-Vega. Solving feature subset selection problem by a parallel scatter search. *European Journal of Operational Research*, 169(2):477–489, 2006.

[29] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton. Big data. *The management revolution. Harvard Bus Rev*, 90(10):61–67, 2012.

[30] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *Computers, IEEE Transactions on*, 100(9):917–922, 1977.

[31] K. Ng and H. Liu. Customer retention via data mining. *Artificial Intelligence Review*, 14(6):569–590, 2000.

[32] C. A. O'Reilly. Variations in decision makers' use of information sources: The impact of quality and accessibility of information. *Academy of Management journal*, 25(4):756–771, 1982.

[33] J. Pacheco, S. Casado, and L. Núñez. A variable selection method based on tabu search for logistic regression models. *European Journal of Operational Research*, 199(2):506–511, 2009.

[34] J. R. Quinlan. *C4. 5: Programs for machine learning*. Morgan Kaufmann, 1993.

[35] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[36] S. L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[37] S. Sarafrazi and H. Nezamabadi-pour. Facing the classification of binary problems with a gsa-svm hybrid system. *Mathematical and Computer Modelling*, 57(1):270–278, 2013.

[38] I. Schwab, A. Kobsa, and I. Koychev. Learning about users from observation. In *Adaptive user interfaces: Papers from the 2000 AAAI spring symposium*, pages 102–106, 2000.

[39] U. K. Sikdar, A. Ekbal, and S. Saha. Mode: multiobjective differential evolution for feature selection and classifier ensemble. *Soft Computing*, 19(12):3529–3549, 2015.

[40] I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[41] E.-G. Talbi, L. Jourdan, J. Garcia-Nieto, and E. Alba. Comparison of population based metaheuristics for feature selection: Application to microarray data classification. In *Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on*, pages 45–52. IEEE, 2008.

[42] A. Unler and A. Murat. A discrete particle swarm optimization method for feature selection in binary classification problems. *European Journal of Operational Research*, 206(3):528–539, 2010.

[43] H. Vafaie and I. F. Imam. Feature selection methods: genetic algorithms vs. greedy-like search. In *Proceedings of International Conference on Fuzzy and Intelligent Control Systems*, pages 39–43, 1994.

[44] E. P. Xing, M. I. Jordan, R. M. Karp, et al. Feature selection for high-dimensional genomic microarray data. In *ICML*, volume 1, pages 601–608. Citeseer, 2001.

[45] B. Xue, M. Zhang, W. Browne, and X. Yao. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, Aug 2016.

[46] B. Xue, M. Zhang, and W. N. Browne. Particle swarm optimization for feature selection in classification: A multi-objective approach. *Cybernetics, IEEE Transactions on*, 43(6):1656–1671, 2013.

[47] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*, pages 117–136. Springer, 1998.

[48] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

[49] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*, volume 3, pages 856–863, 2003.

[50] S. C. Yusta. Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters*, 30(5):525–534, 2009.

APPENDIX A

DETAILS OF SAMPLE DATASETS

Datasets used in this study include various attributes which have different data types. Breast Cancer dataset is selected as a sample dataset and Table A.1 gives information about the attributes of this dataset.

Table A.1: Attributes, domain ranges and sample instances of Breast Cancer dataset.

	clump thickness	**uniformity of cell size**	**uniformity of cell shape**	**marginal adhesion**	**single epithelial cell size**	**bare nuclei**	**bland chromatin**	**normal nucleoli**	**mitoses**	**class**
domain range	1 - 10	1 - 10	1 - 10	1 - 10	1 - 10	1 - 10	1 - 10	1 - 10	1 - 10	2, 4
5 sample instances	8	7	5	10	7	9	5	5	4	4
	7	4	6	4	6	1	4	3	1	4
	4	1	1	1	2	1	2	1	1	2
	4	1	1	1	2	1	3	1	1	2
	10	7	7	6	4	10	4	1	2	4

Table A.2 shows the classification accuracy with and without feature selection. Training the classifier with all 9 features has the accuracy value of 0.954. Accuracy increases significantly after the feature selection process and training the model with only 5 features of the datasets gives an accuracy value of 0.963. These five features are *clump thickness*, *uniformity of cell shape*, *bare nuclei*, *normal nucleoli* and *mitoses*.

Table A.2: Classification accuracies of Logistic Regression with & without feature selection on Breast Cancer dataset.

selected (1) / unselected (0) features									**accuracy**
1	1	1	1	1	1	1	1	1	0.954
1	0	1	0	0	1	0	1	1	0.963

It is obvious that applying feature selection is highly effective. Because, as seen in this example, only 5 features represent this dataset better than all 9 features.

A second example is Nursery dataset. Nursery differs from Breast Cancer since it has text based categorical data along with real-valued data. In Table A.3, sample instances from original Nursery dataset are given. Table A.4 shows the same data as given in Table A.3, only after preprocessing techniques are applied on the dataset. As seen in the table, all text-based values are converted to numerical values before machine learning techniques are applied.

Table A.3: Attributes and sample instances of Nursery dataset.

parents	**has_nurs**	**form**	**children**	**housing**	**finance**	**social**	**health**	**class**
usual	proper	complete	1	convenient	convenient	slightly_prob	recommended	recommend
usual	proper	complete	3	convenient	convenient	slightly_prob	priority	priority
usual	proper	complete	3	convenient	convenient	slightly_prob	not_recom	not_recom
usual	proper	complete	3	convenient	convenient	problematic	recommended	priority
usual	proper	complete	3	convenient	convenient	problematic	priority	priority
usual	proper	complete	3	convenient	convenient	problematic	not_recom	not_recom
pretentious	less_proper	completed	1	convenient	inconv	slightly_prob	recommended	very_recom
pretentious	improper	foster	2	critical	convenient	nonprob	recommended	priority
pretentious	improper	foster	2	critical	convenient	nonprob	priority	spec_prior
pretentious	improper	foster	2	critical	convenient	nonprob	not_recom	not_recom

Table A.4: Attributes and sample instances of Nursery dataset after preprocessing.

parents	**has_nurs**	**form**	**children**	**housing**	**finance**	**social**	**health**	**class**
1	1	1	1	1	1	2	1	2
1	1	1	3	1	1	2	2	4
1	1	1	3	1	1	2	3	1
1	1	1	3	1	1	3	1	4
1	1	1	3	1	1	3	2	4
1	1	1	3	1	1	3	3	1
2	2	2	1	1	2	2	1	3
2	3	4	2	3	1	1	1	4
2	3	4	2	3	1	1	2	5
2	3	4	2	3	1	1	3	1