Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

A study of feature selection algorithms for accuracy estimation

Kashif Javed Butt

Supervisor:

Lluís A. Belanche Muñoz

Barcelona, September 2012

Universitat Politècnica de Catalunya, BarcelonaTech

kashif.javed@lsi.upc.edu

Abstract

The main purpose of Feature Subset Selection is to find a reduced subset of attributes from a data set described by a feature set. The task of a *feature selection algorithm* (FSA) is to provide with a computational solution motivated by a certain definition of *relevance* or by a reliable evaluation measure.

Feature weighting is a technique used to approximate the optimal degree of influence of individual features using a training set. When successfully applied relevant features are attributed a high weight value, whereas irrelevant features are given a weight value close to zero. Feature weighting can be used not only to improve classification accuracy but also to discard features with weights below a certain threshold value and thereby increase the resource efficiency of the classifier.

In this work several fundamental *feature weighting algorithms* (FWAs) are studied to assess their performance in a controlled experimental scenario. A measure to evaluate FWAs *score* is devised that computes the degree of matching between the output given by a FWAs and the known optimal solutions. A study of relation between the *score* obtained from the different classifiers, variance of the score in the different sample size is carried out as well as the relation between the *score* and the estimated probability of error of the model (P_e) for the classification problems and the square error (e^2) for the regression problem.

Acknowledgements

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, my utmost gratitude to Dr. L.A. Belanche (Dept. de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain) whose sincerity and encouragement I will never forget. Dr. Belanche has been my inspiration as I hurdle all the obstacles in the completion this research work.

My colleagues in Artificial Intelligence to help me with my decision to use latex language to write the final report.

Last but not the least, my family for all their support, faith, interest and the one above all of us, the omnipresent God, for answering my prayers for giving me the strength to plod on despite my constitution wanting to give up and throw in the towel, thank you so much Dear Lord.

Contents

List of Figures

List of Tables

1

INTRODUCTION

1.1 Introduction

The *feature selection* problem is ubiquitous in an inductive machine learning or data mining and its importance is beyond doubt. The main benefit of a correct selection is the improvement of the inductive learner, either in terms of learning speed, generalization capacity or simplicity of the induced model.

The high dimensionality problems has brought an interesting challenge for machine learning researchers. Machine learning gets particularly difficult when there are many features and very few samples, since the search space will be sparsely populated and the model will not be able to distinguish correctly the relevant data and the noise.

In this work several fundamental algorithms are studied to assess their performance in a controlled experimental scenario. A measure to evaluate *feature weighting algorithms* (FWAs) '*scoring measure*' is devised that computes the degree of matching between the output given by a *feature weighting algorithm* (FWAs) and the known optimal solutions *score*. An extensive experimental study on synthetic problems is carried out to assess the behaviour of the algorithms in terms of solution accuracy and size as a function of the relevance, irrelevance and size of the data samples. The controlled experimental conditions facilitate the derivation of better-supported and meaningful conclusions.

1.2 Motivation and Related work

Previous experimental work on feature selection algorithms for comparative purposes include [Aha and Bankert, 1994], [Doak, 1992], [Jain and Zongker, 1997], [Kudo, 1997] and [Liu, 1998]. Some of these studies use artificially generated data sets, like the widespread *Parity*, *Led* or *Monks* problems [Thrun et al., 1991]. Demonstrating improvement on synthetic data sets can be more convincing that doing so in typical scenarios where the true solution is completely unknown. However, there is a consistent lack of systematical experimental work using a common benchmark suite and equal experimental conditions. This hinders a wider exploitation of the power inherent in fully controlled experimental environments: the knowledge of the (set of) optimal solution(s), the possibility of injecting a desired amount of relevance and irrelevance and the unlimited availability of data.

Another important issue is the way *feature weighting algorithms* (FWAs) ($\mathcal{A}$) performance is assessed. This is normally done by handing over the solution encountered by the FWAs to a specific inducer (during of after the feature selection process takes place). Leaving aside the dependence on the particular inducer chosen, there is a much more critical aspect, namely, the relation between the performance as reported by the inducer and the true merits of the subset being evaluated. In this sense, it is our hypothesis that *feature selection algorithms* (FSAs) are very affected by finite sample sizes, which distort reliable assessments of subset relevance, even in the presence of a very sophisticated search algorithm [Reunanen, 2003]. Therefore, sample size should also be a matter of study in a through experimental comparison. This problem is aggravated when using filter measures, since in this case the relation to true generalization ability (as expressed by the *Bayes error*) can be very loose [Ben-Bassat, 1982].

A further problem with traditional benchmarking data sets is the implicit assumption that the used data sets are actually *amenable* to feature selection. By this it is meant that performance benefits clearly from a good selection process (and less clearly or even worsens with a bad one). This criterion is not commonly found in similar experimental work. In summary, the rationale for using exclusively synthetic data sets is twofold:

1. Controlled studies can be developed by systematically varying chosen experimental conditions, thus facilitating the derivation of more meaningful conclusions.

2. Synthetic data sets allow full control of the experimental conditions, in terms of amount of relevance and irrelevance, as well as sample size and problem difficulty. An added advantage is the knowledge of the set of optimal solutions, in which case the degree of closeness to any of these solutions can thus be assessed in a confident and automated way.

The procedure followed in this work consists in generating sample datasets from synthetic functions of a number of discrete relevant features. These sample data sets are handed over to different FWAs to obtained a hypothesis. A *scoring measure* is used in order to compute the degree of matching between this hypothesis and the known optimal solution. The score takes into account the amount of relevance and irrelevance in each suboptimal solution as yielded by an algorithm.

1.3 Goals

Machine learning can take advantage of feature selection methods to be able to confront problems such as noise or large number of features. *Feature selection* (FS) is the process of detecting the relevant feature and discarding the irrelevant ones, with the goal of obtaining a subset of features that can give relatively the same performance without significant degradation.

Feature weighting is a technique used to approximate the optimal degree of influence of individual features using a training set. When successfully applied relevant features are attributed a high weight value, whereas irrelevant features are given a weight value close to zero. Feature weighting can be used not only to improve classification accuracy but also to discard features with weights below a certain threshold value and thereby increase the resource efficiency of the classifier.

The main goals of this study are expose the relation using *feature weighting algorithm* (FWAs) between '*Estimated probability of error of the model*' ($\hat{P}_e$), which we obtain predicting from the huge test set (TE) and how well the estimation of the relevance of the variables has been done by the FWAs ($\mathcal{A}$), called *score*. We also study the relation between the *score* and the square error (e^2) obtained from the regression type of problems. The aim is to evaluate the Linear correlation between these variables and finally analyse the influence of classifier in the dataset sample size.

For the classification and regression type of datasets we have used the feature weighting algorithms (FWAs) instead of purely features selection algorithms. In the next chapter details of differences between the two methods can be found, as well as the explanation of FWAs ($\mathcal{A}$) used for this approach.

1.4 Organization

The work is organized as follows. We first overview the methods in section 2.1 for feature selection and in section 2.2 for feature weighting, present them and explain how these method works to obtain the weight of each feature f_i. In section 3 we will describe the type of datasets used in this experiment and how they are constructed. Following the resampling process in section 4.1 will be describe along with the score measurement in section 4.2. We then present our empirical analysis of the methods on synthetic datasets 4.3. We conclude our study in Section 5.

2

METHODS

Variable and feature selection have become the focus of much research in areas of application for which datasets with tens or hundreds of thousands of variables are available. These areas include text processing of Internet documents, gene expression array analysis, and combinatorial chemistry. The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data. The contributions of this special issue cover a wide range of aspects of such problems: providing a better definition of the objective function, feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods.

2.1 Feature Selection

In machine learning and statistics, feature selection, also known as variable selection, feature reduction, attribute selection or variable subset selection, is the technique of selecting a subset of relevant features for building robust learning models. Feature selection is a particularly important step in analysing the data from many experimental techniques in biology, such as DNA microarrays, because they often entail a large number of measured variables (features) but a very low number of samples. By removing most irrelevant and redundant features from the data, feature selection helps improve the performance of learning models by:

- Alleviating the effect of the curse of dimensionality.
- Enhancing generalization capability.
- Speeding up learning process.
- Improving model interpretability.

Feature selection also helps people to acquire better understanding about their data by telling them which are the important features and how they are related with each other.

Feature selection has been an active research area in pattern recognition, statistics, and data mining communities. The main idea of feature selection is to choose a subset of input variables by eliminating features with little or no predictive information. Feature selection can significantly improve the comprehensibility of the resulting classifier models and often build a model that generalizes better to unseen points. Further, it is often the case that finding the correct subset of predictive features is an important problem in its own right. For example, physician may make a decision based on the selected features whether a dangerous surgery is necessary for treatment or not.

Feature selection in supervised learning has been well studied, where the main goal is to find a feature subset that produces higher classification accuracy. Recently, several researches (Dy and Brodley, 2000b, Devaney and Ram, 1997, Agrawal et al., 1998) have studied feature selection and clustering together with a single or unified criterion. For feature selection in unsupervised learning, learning algorithms are designed to find natural grouping of the examples in the feature space. Thus feature selection in unsupervised learning aims to find a good subset of features that forms high quality of clusters for a given number of clusters. However, the traditional approaches to feature selection with single evaluation criterion have shown limited capability in terms of knowledge discovery and decision support. This is because decision-makers should take into account multiple, conflicted objectives simultaneously. In particular no single criterion for unsupervised feature selection is best for every application (Dy and Brodley, 2000a) and only the decision-maker can determine the relative weights of criteria for her application. In order to provide a clear picture of the (possibly nonlinear) tradeoffs among the various objectives, feature selection has been formulated as a multi-objective or Pareto optimization.

Simple feature selection algorithms are ad hoc, but there are also more methodical approaches. From a theoretical perspective, it can be shown that optimal feature selection for supervised learning problems requires an exhaustive search of all possible subsets of features of the chosen cardinality. If large numbers of features are available, this is impractical. For practical supervised learning algorithms, the search is for a satisfactory set of features instead of an optimal set.

Feature selection algorithms typically fall into two categories: feature ranking and subset selection. Feature ranking ranks the features by a metric and eliminates all features that do not achieve an adequate score. Subset selection searches the set of possible features for the optimal subset.

In statistics, the most popular form of feature selection is stepwise regression. It is a greedy algorithm that adds the best feature (or deletes the worst feature) at each round. The main control issue is deciding when to stop the algorithm. In machine learning, this is typically done by cross-validation. In statistics, some criteria are optimized. This leads to the inherent problem of nesting. More robust methods have been explored, such as branch and bound and piecewise linear network.

2.1.1 Subset selection

Subset selection evaluates a subset of features as a group for suitability. Subset selection algorithms can be broken into Wrappers, Filters and Embedded. Wrappers use a search algorithm to search through the space of possible features and evaluate each subset by running a model on the subset. Wrappers can be computationally expensive and have a risk of over fitting to the model. Filters are similar to Wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. Embedded techniques are embedded in and specific to a model.

Many popular search approaches use greedy hill climbing, which iteratively evaluates a candidate subset of features, then modifies the subset and evaluates if the new subset is an improvement over the old. Evaluation of the subsets requires a scoring metric that grades a subset of features. Exhaustive search is generally impractical, so at some implementor (or operator) defined stopping point, the subset of features with the highest score discovered up to that point is selected as the satisfactory feature subset. The stopping criterion varies by algorithm; possible criteria include: a subset score exceeds a threshold, a program's maximum allowed run time has been surpassed, etc.

Alternative search-based techniques are based on targeted projection pursuit which finds low-dimensional projections of the data that score highly: the features that have the largest projections in the lower dimensional space are then selected.

Search approaches include:

- Exhaustive
- Best first
- Simulated annealing
- Genetic algorithm
- Greedy forward selection
- Greedy backward elimination
- Targeted projection pursuit
- Scatter Search
- Variable Neighbourhood Search

Two popular filter metrics for classification problems are correlation and mutual information, although neither are true metrics or 'distance measures' in the mathematical sense, since they fail to obey the triangle inequality and thus do not compute any actual 'distance' they should rather be regarded as 'scores'. These scores are computed between a candidate feature (or set of features) and the desired output category.

Other available filter metrics include:

- Class separability
  - Error probability
  - Inter-class distance
  - Probabilistic distance
  - Entropy
- Consistency-based feature selection
- Correlation-based feature selection

2.1.2 Wrapper and filter approach

Feature selection algorithms can be divided into two basic approaches. First is the wrapper approach, where the selection of features is 'wrapped' within a learning algorithm. The second method is called the lter approach, where the features are selected according to data intrinsic values, such as information, dependency or consistency measures. The differences between these approaches is in the way the quality of the feature subset is measured. For wrapper approaches, a common method is to measure the predictive accuracy for a given subset. Then, applying some forward or backward selection method, the subset is changed and the accuracy is re-evaluated. If it is better, the new subset is retained, otherwise the old one is kept. You can iterate this until you reach a given number of features, some accuracy threshold, after a xed number of iterations, or when you have explored the whole search space. For this method, the learning algorithm has to run for each subset, so it should not be too demanding.This is the main drawback of this approach. The other method is the lter method. As mentioned, this method uses the intrinsic properties of the data and therefore is not dependent on the learning algorithm. Information, dependency or consistency measures are usually less complex (in time). Therefore these algorithms can be used on large datasets, to reduce the input dimensionality. After that, you can use a classier or a wrapper to deal with the reduced dataset. Another advantage of this approach is that you can use any classier to evaluate the accuracy of the testset. In the wrapper approach, if we use a different classier for the testset and trainingset, results may be negatively affected.

The differences is in the evaluation method, where filter approaches use a evaluation measure independent of the learning algorithm. In the wrapper approach, the evaluation is performed by the learning algorithm itself, taking the predictive accuracy as a measure for feature quality.

2.2 Feature Weighting

Feature weighting is a technique used to approximate the optimal degree of influence of individual features using a training set. When successfully applied relevant features are attributed a high weight value, whereas irrelevant features are given a weight value close to zero. Feature weighting can be used not only to improve classification accuracy but also to discard features with weights below a certain threshold value and thereby increase the resource efficiency of the classifier. Feature weighting is a crucial process to identify an important subset of features from a data set. Due to fundamental differences between classification and ranking, feature weighting methods developed for classification cannot be readily applied to feature weighting for ranking. A state of the art feature selection method for ranking has been recently proposed, which exploits importance of each feature and similarity between every pair of features. However, this ranking methods must compute the similarity scores of all pairs of features, thus it is not scalable for high-dimensional data and its performance degrades on nonlinear ranking functions.

2.2.1 Random Forests

This section gives a brief overview of random forests and some comments about the features of the method.

2.2.1.1 Overview

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.

2. If there are M input variables, a number m¡¡M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

3. Each tree is grown to the largest extent possible. There is no pruning.

In the original paper on random forests, it was shown that the forest error rate depends on two things:

- The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.

- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Reducing m reduces both the correlation and the strength. Increasing it increases both. Somewhere in between is an "optimal" range of m - usually quite wide. Using the oob error rate (see below) a value of m in the range can quickly be found. This is the only adjustable parameter to which random forests is somewhat sensitive.

2.2.1.2 Features of Random Forests

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabelled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

2.2.1.3 Remarks

Random forests does not overfit. You can run as many trees as you want. It is fast. Running on a data set with 50,000 cases and 100 variables, it produced 100 trees in 11 minutes on a 800Mhz machine. For large data sets the major memory requirement is the storage of the data itself, and three integer arrays with the same dimensions as the data. If proximities are calculated, storage requirements grow as the number of cases times the number of trees.

2.2.1.4 The algorithm

The random forests algorithm (for both classification and regression) is as follows:

1. Draw ntree bootstrap samples from the original data.

2. For each of the bootstrap samples, grow an unpruned classification or regression tree, with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample mtry of the predictors and choose the best split from among those variables. (Bagging can be thought of as the special case of random forests obtained when mtry = p, the number of predictors.)

3. Predict new data by aggregating the predictions of the ntree trees (i.e., majority votes for classification, average for regression).

An estimate of the error rate can be obtained, based on the training data, by the following:

1. At each bootstrap iteration, predict the data not in the bootstrap sample (what Breiman calls out-of-bag, or OOB, data) using the tree grown with the bootstrap sample.

2. Aggregate the OOB predictions. (On the average, each data point would be out-of-bag around 36% of the times, so aggregate these predictions.) Calculate the error rate, and call it the OOB estimate of error rate.

Algorithm 1 Pseudo code for the random forest algorithm

```
1: To generate c classifiers:
2: for $i = 1 \to c$ do
3:     Randomly sample the training data $D$ with replacement to produce $D_i$
4:     Create a root node, $N_i$ containing $D_i$
5:     Call BuildTree($N_i$)
6: end for
7: BuildTree($N$):
8: if $N$ contains instances of only one class then return
9: else
10:    Randomly select $x$                    ▷ of the possible splitting features in N
11:    Select the feature $F$ with the highest information gain to split on
12:    Create $f$ child nodes of $N, N_1, ..., N_f$ , where $F$ has $f$ possible values $(F_1, ..., F_f)$
13:    for $i = 1 \to f$ do
14:        Set the contents of $N_i \to D_i$ , where $D_i$ is all instances in $N$ that match $F_i$
15:        Call BuildTree($N_i$)
16:    end for
17: end if
```

Figure 2.1: Algorithm: Random Forests

2.2.1.5 Extra information from Random Forests

The **randomForest** package optionally produces two additional pieces of information: a measure of the importance of the predictor variables, and a measure of the internal structure of the data (the proximity of different data points to one another).

Variable importance This is a difficult concept to define in general, because the importance of a variable may be due to its (possibly complex) interaction with other variables. The random forest algorithm estimates the importance of a variable by looking at how much prediction error increases when (OOB) data for that variable is permuted while all others are left unchanged. The necessary calculations are carried out tree by tree as the random forest is constructed. (There are actually four different measures of variable importance implemented in the classification code. The reader is referred to Breiman (2002) for their definitions.)

Proximity measure The (i, j) element of the proximity matrix produced by **randomForest** is the fraction of trees in which elements i and j fall in the same terminal node. The intuition is that similar observations should be in the same terminal nodes more often than dissimilar ones. The proximity matrix can be used to identify structure in the data (see Breiman,2002) or for unsupervised learning with random forests.

2.2.2 Relief

RELIEF is considered one of the most successful algorithms for assessing the quality of features due to its simplicity and effectiveness. It has been recently proved that RELIEF is an online algorithm that solves a convex optimization problem with a margin-based objective function.

Relief is a feature weight based algorithm inspired by instance-based learning. Given training data S, sample size m, and a threshold of relevancy τ, Relief detects those features which are statistically relevant to the target concept. τ encodes a relevance threshold ($0 \leq \tau \leq 1$). We assume the scale of every feature is either nominal (including boolean) or numerical (integer or real). Differences of feature values between two instances X and Y are defined by the following function diff.

When x_k and y_k are nominal,

$$diff(x_k, y_k) = \begin{cases} 0, & \text{if } x_k \text{ and } y_k \text{ are the same} \\ 1, & \text{if } x_k \text{ and } y_k \text{ are different} \end{cases}$$

When x_k and y_k are numerical,

$$diff(x_k, y_k) = (x_k - y_k)/nu_k$$

where nu_k is a normalization unit to normalize the values of $diff$ into the interval $[0, 1]$

In Relief, both nominal and numeric features can be used, and they can also be used simultaneously. However in that case, one should be aware that numerical features tend to be underestimated and compensate for this fact. I will not go into this, for I only have to deal with nominal features. Therefore the distance for a feature between two instances is either 0 (they are the same) or 1 (they are different). After each iteration, the weights for a feature are updated and normalized within the interval $[-1, 1]$ by dividing it with the number of samples and iterations. The output is a weight vector, with a weight W_i for each feature i. In the original paper, the authors propose a relevancy threshold τ. If you select all features with a weight $\geq \tau$ you get a subset selection algorithm. The authors describe a statistical mechanism to calculate τ , but for sequence alignments, a ranked list of all features is preferred in most cases. In this case, Relief operates as a feature ranking mechanism. By now, it should be obvious that the key to the success of Relief lays in the fact that it does a global and a local search. It does not rely on greedy heuristics like many other algorithms that often get stuck in local optima. This idea is nicely illustrated in [Liu and Motada, 1998]. The idea is that a relevant feature can separate two instances from opposite classes that are closely related. Therefore it takes the most closely related instance from an opposite class (the nearest miss) and from the same class (the nearest hit).

2.2.2.1 Algorithm

The algorithm finds weights of continuous and discrete attributes basing on a distance between instances. The algorithm samples instances and finds their nearest hits and misses. Considering that result, it evaluates weights of attributes.

Relief 2 picks a sample composed of m triplets of an instance X, its $Near-hit$ instance[1] and

[1]We call an instance a near-hit of X if it belongs to the close neighbourhood of X and also to the same category

$Near - miss$ instance. Relief uses the $p - dimensional$ Euclid distance for selecting $Near - hit$ and $Near - miss$. Relief calls a routine to update the feature weight vector W for every sample triplet and to determine the average feature weight vector **Relevance** (of all the features to the target concept). Finally, Relief selects those features whose average weight ('**relevance level**') is above the given threshold τ.

Algorithm 2 Relief algorithm

```
    function RELIEF($S, m, \tau$)
2:      Separate $S$ into $S^+ = \{$positive instances$\}$ and
        $S^- = \{$negative instances$\}$
4:      $W = (0, 0, ..., 0)$
        for $i = 1 \to m$ do
6:          Pick at random an instances $X \in S$
            Pick at random one of the positive instances closet to $X, Z^+ \in S^+$
8:          Pick at random one of the negative instances closet to $X, Z^- \in S^-$
            if $X$ is a positive instance then $Near - hit = Z^+$ $Near - miss = Z^-$
10:         else$Near - hit = Z^-$ $Near - miss = Z^+$
            end if
12:         update-weight($W, X, Near - hit, Near - Miss$)
        end for
14:     $Relevance = (1/m)W$
        for $i = 1 \to p$ do
16:         if $relevance_i \geq t$ then $f_i$ is a relevant feature
            else$f_i$ is an irrelevant feature
18:         end if
        end for
20: end function
    function UPDATE-WEIGHT($W, X, Near - hit, Near - Miss$)
22:     for $i = 1 \to p$ do
            $W_i = W_i - diff(x_i, near - hit_i)^2 + diff(x_i, near - miss_i)^2$
24:     end for
    end function
```

Figure 2.2: Algorithm: Relief

2.2.2.2 Theoretical Analysis

Relief has two critical components: the averaged weight vector **Relevance** and the threshold τ. **Relevance** is the averaged vector of the value $-(x_i - near - hit_i)^2 + (x_i - near - miss_i)^2$ for each feature f_i over **m** sample triplets. Each element of **Relevance** corresponding to a feature shows how relevant the feature is to the target concept. τ is a relevance threshold for determining whether the feature should be selected.

The complexity of Relief is θ(pmn). Since **m** is an arbitrarily chosen constant, the complexity is θ(pn). Thus the algorithm can select statistically relevant features in linear time in the number of features and the number of training instances.

as X. We call an instance a $near - miss$ when it belongs to the properly close neighbourhood of X but not to the same category as X.

Relief is valid only when the relevance level is huge for relevant features and small for irrelevant features, and τ retains relevant features and discards irrelevant features.

2.2.3 Linear Support Vector Machine (SVM-Linear)

Linear SVM is the newest extremely fast machine learning (data mining) algorithm for solving multi-class classification problems from ultra large data sets that implements an original proprietary version of a cutting plane algorithm for designing a linear support vector machine. LinearSVM is a linearly scalable routine meaning that it creates an SVM model in a CPU time which scales linearly with the size of the training data set.

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p1) - dimensional$ hyperplane. This is called a **linear classifier**. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier; or equivalently, the perceptron of optimal stability.

Given some training data $\mathcal{D}$, a set of n points of the form

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\, y_i \in \{-1, 1\}\}_{i=1}^{n}$$

where the y_i is either 1 or -1, indicating the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p-dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0,$$

where $\cdot$ denotes the dot product and $\mathbf{w}$ the normal vector to the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called "the margin". These hyperplanes can be described by the equations

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

By using geometry, we find the distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so we want to minimize $\|\mathbf{w}\|$. As we also have to prevent data points from falling into the margin, we add the following constraint: for each i either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i \text{ of the first class}$$

or

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i \text{ of the second.}$$

This can be rewritten as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n. \tag{1}$$

We can put this together to get the optimization problem: Minimize (in $\mathbf{w}, b$)

$$\|\mathbf{w}\|$$

subject to (for any $i = 1, \dots, n$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$

Below we present the popular feature selection algorithm (FSA) *SVM - Recursive Feature elimination* (SVM-RFE). However, we do not use this algorithm directly because this belong to the family of FSA and it is not purely a *Feature weighting algorithm.* Instead, we have adapted the SVM-RFE algorithm 3 for this approach. In this case we are not working with the returned list of features L. Instead, we return the *vector of weights* (ω) where each feature is assigned by his correspondent weight.

Algorithm 3 SVM-RFE algorithm

```
1: function SVM-RFE($T, fs$)
2: T: Set of training examples; each example is described by a vector of feature values ($x$) and
   its class ($y$)
3: fs: Set of features describing each example in T;
4: L: Ordered list of feature subsets; each subset contains the remaining features at every iteration
5:     $F_m = fs$                                          ▷ $m$ is the number of features
6:     $L = [F_m]$                                         ▷ Initially, one subset with all the features
7:     for $j = m \to 2$ do
8:         $\alpha = SVM(T)$                               ▷ Train SVM
9:         $\omega = \Sigma \alpha_k y_k x_k$              ▷ $\omega$: the hyperplane coefficients
10:        $r = argmin(\omega_i^2 : i = 1, \dots, |F_i|)$  ▷ The smallest ranking criterion
11:        $F_{j-1} = F_j \backslash f_r$                  ▷ Remove r-th feature from $F_j$
12:        $L = L + F_{j-1}$                               ▷ Add the subset of remaining features
13:        $T = \{x\prime_i : x\prime_i$ is $x_i \in T$ with $f_r$ removed$\}$   ▷ Remove r-th feature from examples in $T$
14:    end for
15:    $return(L)$
16: end function
```

Figure 2.3: Algorithm: SVM-RFE

2.2.3.1 Primal form

The optimization problem presented in the preceding section is difficult to solve because it depends on $||w||$, the norm of w, which involves a square root. Fortunately it is possible to alter the equation by substituting $||w||$ with $\frac{1}{2}\|\mathbf{w}\|^2$ (the factor of 1/2 being used for mathematical convenience) without changing the solution (the minimum of the original and the modified equation have the same w and b). This is a quadratic programming optimization problem. More clearly: Minimize (in $\mathbf{w}, b$)

$$\frac{1}{2}\|\mathbf{w}\|^2$$

subject to (for any $i = 1, \ldots, n$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$

By introducing Lagrange multipliers $\boldsymbol{\alpha}$, the previous constrained problem can be expressed as

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha} \geq 0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1] \right\}$$

that is we look for a saddle point. In doing so all the points which can be separated as $y_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1 > 0$ do not matter since we must set the corresponding α_i to zero. This problem can now be solved by standard quadratic programming techniques and programs. The "stationary" Karush-Kuhn-Tucker condition implies that the solution can be expressed as a linear combination of the training vectors

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}.$$

Only a few α_i will be greater than zero. The corresponding $\mathbf{x_i}$ are exactly the support vectors, which lie on the margin and satisfy $y_i(\mathbf{w} \cdot \mathbf{x_i} - b) = 1$. From this one can derive that the support vectors also satisfy

$$\mathbf{w} \cdot \mathbf{x_i} - b = 1/y_i = y_i \iff b = \mathbf{w} \cdot \mathbf{x_i} - y_i$$

which allows one to define the offset b. In practice, it is more robust to average over all N_{SV} support vectors:

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\mathbf{w} \cdot \mathbf{x_i} - y_i)$$

2.2.3.2 Dual form

Writing the classification rule in its unconstrained dual form reveals that the maximum margin hyperplane and therefore the classification task is only a function of the *support vectors*, the training data that lie on the margin. Using the fact that $\|\mathbf{w}\|^2 = w \cdot w$ and substituting $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}$, one can show that the dual of the SVM reduces to the following optimization problem: Maximize (in α_i)

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to (for any $i = 1, \ldots, n$)

$$\alpha_i \geq 0,$$

and to the constraint from the minimization in b

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

Here the kernel is defined by $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$.

W can be computed thanks to the α terms:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i.$$

2.2.4 Nonlinear classification

The original optimal hyperplane algorithm proposed by Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick (originally proposed by Aizerman et al. [Aizerman et al., 1964]) to maximum-margin hyperplanes. [Boser et al., 1992] The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimensions. Maximum margin classifiers are well regularized, so the infinite dimensions do not spoil the results. Some common kernels include:

- Polynomial (homogeneous): $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j})^d$
- Polynomial (inhomogeneous): $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$
- Gaussian radial basis function: $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$, for $\gamma > 0$. Sometimes parametrized using $\gamma = 1/2\sigma^2$
- Hyperbolic tangent: $k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c)$, for some (not every) $\kappa > 0$ and $c < 0$

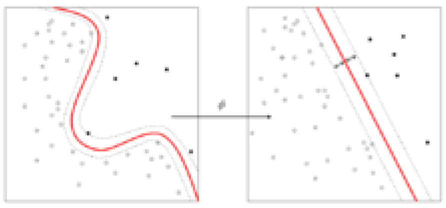
Figure 2.4: Kernel Machine

The kernel is related to the transform $\varphi(\mathbf{x_i})$ by the equation $k(\mathbf{x_i}, \mathbf{x_j}) = \varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x_j})$. The value w is also in the transformed space, with $\mathbf{w} = \sum_i \alpha_i y_i \varphi(\mathbf{x}_i)$. Dot products with w for classification can again be computed by the kernel trick, i.e. $\mathbf{w} \cdot \varphi(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})$. However, there does not in general exist a value w' such that $\mathbf{w} \cdot \varphi(\mathbf{x}) = k(\mathbf{w}', \mathbf{x})$.

2.2.4.1 Properties

SVMs belong to a family of generalized linear classifiers and can be interpreted as an extension of the perceptron. They can also be considered a special case of Tikhonov regularization. A special property is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

2.2.4.2 Parameter selection

The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C.

A common choice is a Gaussian kernel, which has a single parameter γ. Best combination of C and γ is often selected by a grid search with exponentially growing sequences of C and γ, for example, $C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\}$; $\gamma \in \{2^{-15}, 2^{-13}, \dots, 2^{1}, 2^{3}\}$. Typically, each combination of parameter choices is checked using cross validation, and the parameters with best cross-validation accuracy are picked. The final model, which is used for testing and for classifying new data, is then trained on the whole training set using the selected parameters. [Hsu et al., 2003]

2.2.4.3 Issues

Potential drawbacks of the SVM are the following three aspects:

- Uncalibrated class membership probabilities
- The SVM is only directly applicable for two-class tasks. Therefore, algorithms that reduce the multi-class task to several binary problems have to be applied.
- Parameters of a solved model are difficult to interpret.

2.2.5 Linear Regression

In statistics, **linear regression** is an approach to modelling the relationship between a scalar dependent variable y and one or more explanatory variables denoted X. The case of one explanatory variable is called simple regression. More than one explanatory variable is multiple regression. (This in turn should be distinguished from multivariate linear regression, where multiple correlated dependent variables are predicted,[citation needed] rather than a single scalar variable.)

In linear regression, data are modelled using linear predictor functions, and unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, linear regression refers to a model in which the conditional mean of y given the value of X is an affine function of X. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of y given X, rather than on the joint probability distribution of y and X, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications of linear regression fall into one of the following two broad categories:

- If the goal is prediction, or forecasting, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, if an additional value of X is then given without its accompanying value of y, the fitted model can be used to make a prediction of the value of y.
- Given a variable y and a number of variables $X_1, \ldots, X_p$ that may be related to y, linear regression analysis can be applied to quantify the strength of the relationship between y and the X_j, to assess which X_j may have no relationship with y at all, and to identify which subsets of the X_j contain redundant information about y.

Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the lack of fit in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares loss function as in ridge regression. Conversely, the least squares approach can be used to fit models that are not linear models. Thus, while the terms "least squares" and "linear model" are closely linked, they are not synonymous.

Given a data set $\{(y_i, x_i1, \ldots, x_ip)\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y_i and the p-vector of regressors x_i is linear. This relationship is modelled through a disturbance term or *error variable* ε an unobserved random variable that adds noise to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = \beta_1 x_i 1 + \cdots + \beta_p x_i p + \varepsilon_i = x_i^T \beta + \varepsilon_i, \quad i = 1, \ldots, n$$

where T denotes the transpose, so that $x_i^T \beta$ is the inner product between vectors x_i and β.

Often these n equations are stacked together and written in vector form as

$$y = X\beta + \varepsilon,$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_1 1 & \dots & x_1 p \\ x_2 1 & \dots & x_2 p \\ \vdots & \ddots & \vdots \\ x_n 1 & \dots & x_n p \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_p \end{pmatrix}$$

Some remarks on terminology and general use:

- y_i is called the regressand, exogenous variable, response variable, measured variable, or dependent variable (see dependent and independent variables.) The decision as to which variable in a data set is modelled as the dependent variable and which are modelled as the independent variables may be based on a presumption that the value of one of the variables is caused by, or directly influenced by the other variables.

- x_i are called regressors, endogenous variables, explanatory variables, covariates, input variables, predictor variables, or independent variables (see dependent and independent variables, but not to be confused with independent random variables). The matrix X is sometimes called the design matrix.

  - Usually a constant is included as one of the regressors. For example we can take $x_{i1} = 1$ for $i = 1, \dots, n$. The corresponding element of β is called the intercept. Many statistical inference procedures for linear models require an intercept to be present, so it is often included even if theoretical considerations suggest that its value should be zero.
  - Sometimes one of the regressors can be a non-linear function of another regressor or of the data, as in polynomial regression and segmented regression. The model remains linear as long as it is linear in the parameter vector β.
  - The regressors x_{ij} may be viewed either as random variables, which we simply observe, or they can be considered as predetermined fixed values which we can choose. Both interpretations may be appropriate in different cases, and they generally lead to the same estimation procedures; however different approaches to asymptotic analysis are used in these two situations.

- β is a p-dimensional parameter vector. Its elements are also called effects, or regression coefficients. Statistical estimation and inference in linear regression focuses on β.

- ε_i is called the error term, disturbance term, or noise. This variable captures all other factors which influence the dependent variable y_i other than the regressors x_i. The relationship between the error term and the regressors, for example whether they are correlated, is a crucial step in formulating a linear regression model, as it will determine the method to use for estimation.

2.2.6 Logistic Regression

Logistic regression is an approach to prediction, like Ordinary Least Squares (OLS) regression. However, with logistic regression, the researcher is predicting a dichotomous outcome. This situation poses problems for the assumptions of OLS that the error variances (residuals) are normally distributed. Instead, they are more likely to follow a logistic distribution. When using the logistic distribution, we need to make an algebraic conversion to arrive at our usual linear regression equation ($Y = B_0 + B_1X + e$).

With logistic regression, there is no standardized solution printed. And to make things more complicated, the unstandardised solution does not have the same straight-forward interpretation as it does with OLS regression.

One other difference between OLS and logistic regression is that there is no R^2 to gauge the variance accounted for in the overall model (at least not one that has been agreed upon by statisticians). Instead, a chi-square test is used to indicate how well the logistic regression model fits the data.

Probability that Y = 1

Because the dependent variable is not a continuous one, the goal of logistic regression is a bit different, because we are predicting the likelihood that Y is equal to 1 (rather than 0) given certain values of X. That is, if X and Y have a positive linear relationship, the probability that a person will have a score of $Y = 1$ will increase as values of X increase. So, we are stuck with thinking about predicting probabilities rather than the scores of dependent variable.

In logistic regression, a complex formula is required to convert back and forth from the logistic equation to the OLS-type equation. The logistic formulas are stated in terms of the probability that $Y = 1$, which is referred to as $\hat{p}$. The probability that Y is 0 is $1 - \hat{p}$

$$\ln\left(\frac{\hat{p}}{1-\hat{p}}\right) = B_0 + B_1X$$

The ln symbol refers to a natural logarithm and $B_0 + B_1X$ is our familiar equation for the regression line.

P can be computed from the regression equation also. So, if we know the regression equation, we could, theoretically, calculate the expected probability that $Y = 1$ for a given value of X.

$$\hat{p} = \frac{e^{B_0+B_1x}}{1 + e^{B_0+B_1x}}$$

Definition An explanation of logistic regression begins with an explanation of the logistic function, which, like probabilities, always takes on values between zero and one [Hosmer and Lemeshow, 2000]

Deviance. With logistic regression, instead of R^2 as the statistic for overall fit of the model, we have deviance instead. Chi-square was said to be a measure of "goodness of fit" of the observed and the expected values. We use chi-square as a measure of model fit here in a similar way. It is the fit of the observed values (Y) to the expected values ($\hat{Y}$). The bigger the difference (or "deviance") of the observed values from the expected values, the poorer the fit of the model. So, we want a small deviance if possible. As we add more variables to the equation the deviance should get smaller, indicating an improvement in fit.

Maximum Likelihood. Instead of finding the best fitting line by minimizing the squared residuals, as we did with OLS regression, we use a different approach with logistic Maximum Likelihood (ML). ML is a way of finding the smallest possible deviance between the observed and predicted values (kind of like finding the best fitting line) using calculus (derivatives specifically). With ML, the computer uses different "iterations" in which it tries different solutions until it gets the smallest possible deviance or best fit. Once it has found the best solution, it provides a final value for the deviance, which is usually referred to as "negative two log likelihood" (shown as "-2 Log Likelihood" in SPSS). The deviance statistic is called 2LL by Cohen et al. (2003) and Pedazur and D by some other authors (e.g., Hosmer and Lemeshow, 1989), and it can be thought of as a chi-square value.

2.3 R functions of the classifiers

2.3.1 Random Forests

We have used the function 'randomForest' defined in the package 'randomForest', which implements Breimans random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression.

2.3.2 Relief

We have used the function 'relief' defined in the package 'FSelector', this algorithm finds weights of continuous and discrete attributes basing on a distance between instances. The algorithm samples instances and finds their nearest hits and misses. Considering that result, it evaluates weights of attributes.

2.3.3 Linear-SVM

We have adapted the SVM-RFE algorithm so that we return the weights of each feature instead of the list of relevant features.

2.3.4 Linear and Logistic Regression

For both linear and logistic regression we use the glm function defined in the package 'glm'. The only difference is that if we want to perform linear regression we set the value of the parameter $family = "gaussian"$ and if we want to perform logistic regression then we define the parameter $family = "binomial"$ in the glm function.

3

DATASETS

Only Synthetic Datasets were used for this work. Following we will explain in details the type of datasets used and how they have generated.

Basically two type of synthetic datasets were used in this approach: classification and regression.

3.1 Datasets

We have two types of classification datasets. The differences between them are the way that covariance matrix and mean vector are defined. In the following sections we will describe the whole process of these datasets.

3.1.1 Classification dataset (I)

This type of dataset is also called 'probability of density'. Because it is generated following the next equation

$$P(x) = P(w_1)P(x|w_1) + P(w_2)P(x|w_2),$$

$P(x|w_1)$ is the density of $\mathcal{N}(\mu_0, \Sigma_0)$ and $P(x|w_2)$ is the density of $\mathcal{N}(\mu_1, \Sigma_1)$, where μ_0 and μ_1 are the mean vector of class 0 and 1, respectively, and Σ_0 and Σ_1 the covariance matrices. The distributions are multivariate gaussians.

In this model type, we let $\mu_0 = \left(\underbrace{\overbrace{m, \cdots, m}^{d\prime}, 0, \cdots, 0}_{\text{d}} \right)$ and $\mu_1 = -\mu_0$, where d is the total length of the mean vector and $d\prime$ is the length of the relevance features.

We use a block-based structure for the covariance matrices.In our distribution models we let Σ_0 and Σ_1 have the identical structure and we call them Σ, where Σ is of the form

$$\Sigma_{dxd} = \begin{bmatrix} \Sigma_1 & & & 0 \\ & \Sigma_2 & & \\ & & \ddots & \\ 0 & & & \Sigma_{\frac{d}{10}} \end{bmatrix}$$

The covariance matrix Σ_i used as the diagonal values for the big Σ_{dxd} is of the form

$$\Sigma_{kxk} = \begin{bmatrix} 1 & & & & 0.8 \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ 0.8 & & & & 1 \end{bmatrix}$$

Let's have a look on the parameters used to generate this type of dataset in the following table:

Parameters	Values/descriptions
$P(w_1)$	0.7
$P(w_2)$	0.3
N	sample size
	$N = 10^2$ (for small sample size)
	$N = 10^3$ (for medium sample size)
	$N = 10^4$ (for big sample size)
d	$d = 50$ (Total number of features)
$d\prime$	$d\prime = 30$ (# of relevant features)
m	value for the mean vector components
k	dimension of covariance sub matrix
a	$a = 0.8$ (upper and lower triangle values
	of the covariance sub matrix)
b	$b = 1$ (diagonal value of the covariance sub matrix)

Table 3.1: Classification (I): Distribution model parameters

Before starting with the dataset generation process we have to take into account the following restrictions on the parameters seen in the previous table:

Restrictions	Description
$P(w_1) + P(w_2) = 1$	The sum of both probabilities must be equal to one
$d\prime \leq d$	The total number of relevant feature should be lower
	or equal the total number of features
$d \div k = 0$	k must be divisor d
$a > 0$	The upper and lower triangular values of the
	covarianve matrix must be positive

Table 3.2: Classification (I): Parameters Restrictions

Now we want to distribute the data among two classes, which means we will work with binary class datasets. The next equation is used to divide data between two classes:

$$y = \text{sign}(\textstyle\sum_{i=1}^{d} r_i x_i)$$

where we choose $sign(z)$ to assign the sample to one class or another. Detailed description is in the following equation:

$$sign(z) = \begin{cases} 1, & z > 0 \text{ , if z is positive, sample belongs to class 1} \\ -1, & z < 1 \text{ , if z is negative, sample belongs to class 2} \end{cases}$$

Another important concept is the **relevance vector** (r_i^*) also known as optimal solution where we will have the relevance given to the features in the dataset generation process. To assign the relevance to the features we follow the next equation:

$$r_i^* = \frac{|r_i|}{\sum_{j=1}^{d} |r_j|}, \quad \text{where} \quad r_i = \begin{cases} \mathcal{N}(0, \sigma^2), & i = 1, \ldots, d\prime \\ 0, & i = d\prime + 1, \ldots, d \end{cases}$$

In the previous equation we can see the vector r_i is generated as a normal distribution of length $d\prime$ and then is filled with the value 0 from $d\prime + 1$ since d.

The next figures gives us an idea of how is the composition of the dataframe

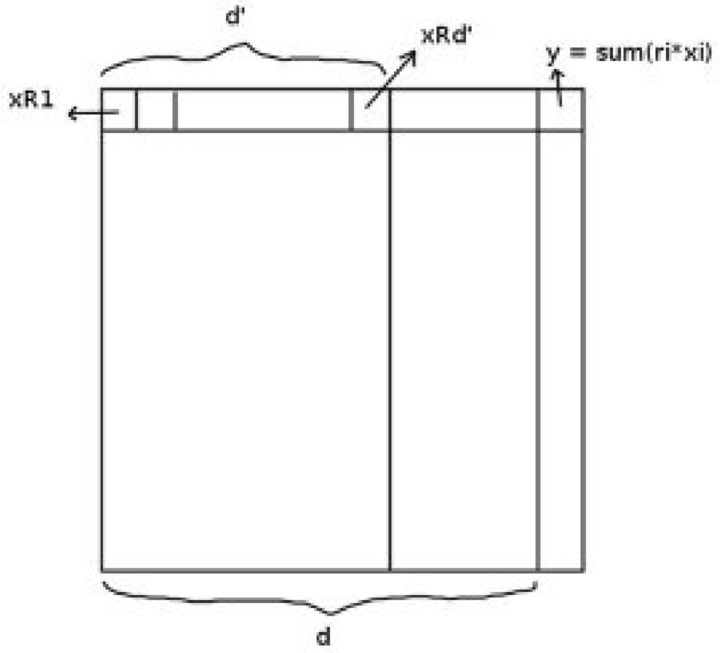

Figure 3.1: Dataframe

and the composition of the relevance vector (r_i^*)

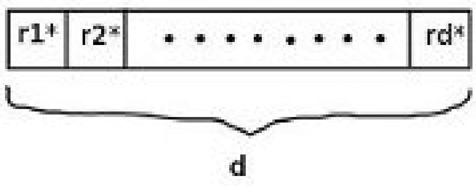

Figure 3.2: Relevance vector (r*)

The classification dataset generating process is explained in the follow algorithm:

Algorithm 4 Classification dataset (I)

```
function generateClasificationDataset($n, d, d\prime, m, k, a, b, priors, sdr$)
$n$: number of samples
$d$: total number of features
$d\prime$: number of relevant features
$m$: value of mean vector parameter
$k$: dimension of covariance matrix
$a, b$: diagonal and triangles values of covariance matrix
$priors$: $P(w_1), P(w_2)$
$sdr$: standard deviation for relevance vector

    if sum($priors$)$\neq$ 1 then
        $print$(”ERROR: The sum of probabilities should be 1.”)
        $return()$
    else
        $Covar = getSigma(d, k, a, b)$    ▷ Get covariance matrix $\Sigma_{dxd}$
        $xlist = list()$    ▷ Initialize the list that will contain the distribution
        for $i = 1 \to n$ do
            $pw1 = runif(1, min = 0, max = 1)$    ▷ Generate random number between 0 and 1
            if $pw1 \leq priors[1]$ then    ▷ check to which $P(w_i)$ belongs
                $mu.f = getMu(d, d\prime, m)$    ▷ Get the mean vector for probability $P(w_1)$
            else
                $mu.f = -getMu(d, d\prime, m)$    ▷ Get the mean vector for probability $P(w_2)$
            end if
            $xlist[[i]] = mvrnorm(1, mu = mu.f, Sigma = Covar)$    ▷ Generate one row of data
        end for
        $x = matrix(unlist(xlist), ncol = d, byrow = TRUE)$    ▷ Save the data matrix into x
        $r = GenerateRi(d, dR, sdr)$    ▷ Generate the vector $R_i$
        $classes = as.factor(apply(x, 1, function(x)sign(sum(x * r))))$    ▷ Assign each sample to one class or another
        $r_i^* = GenerateRiStar(r)$    ▷ Generate the relevance vector $r_i^*$
        $return(list(data.frame(classes, x), rStar))$ ▷ Return dataset and his Relevance vector
    end if
end function
```

Figure 3.3: Algorithm: Classification dataset (I)

3.1.2 Classification dataset (II)

Again we have two classes, two probabilities $P(w_1)$, $P(w_2)$ and $P(x)$ follows the equation

$$P(x) = P(w_1)P(x|w_1) + P(w_2)P(x|w_2),$$

In this case we let $\mu_0 = \left(\underbrace{\frac{1}{\sqrt{1}}, \frac{1}{\sqrt{2}}, \cdots, \frac{1}{\sqrt{d}}}_{\text{d}} \right)$ and $\mu_1 = -\mu_0$, as before we use a block-based structure for the covariance matrices.In our distribution models we let Σ_0 and Σ_1 have the identical structure and we call them Σ, where Σ is of the form

$$\mathbf{\Sigma} = \operatorname{diag}\left(\sigma_1^2, \sigma_2^2, \ldots, \sigma_d^2\right) \quad = \quad \begin{bmatrix} \sigma_1^2 & & & 0 \\ & \sigma_2^2 & & \\ & & \ddots & \\ 0 & & & \sigma_d^2 \end{bmatrix}$$

The **relevance vector** (r_i^*) is the same form as the previous type of dataset and follows the equation

$$r_i^* = \frac{|r_i|}{\sum_{j=1}^{d} |r_j|}$$

but the difference is that the relevance vector r_i is exactly the same as the mean vector. Meanwhile in the previous case the r_i was generated form a normal distribution of length $d\prime$ for the relevant features and zeros for the rest of the vector from $d\prime + 1$ to d.

Let's have a look on the parameters used to generate this type of dataset in the following table:

Parameters	Values/descriptions
$P(w_1)$	0.7
$P(w_2)$	0.3
N	sample size
	$N = 10^2$ (for small sample size)
	$N = 10^3$ (for medium sample size)
	$N = 10^4$ (for big sample size)
d	$d = 50$ (Total number of features)
$d\prime$	$d\prime = 30$ (# of relevant features)
v	diagonal value of the covariance matrix

Table 3.3: Classification (II): Distribution model parameters

We have to take into account the following restrictions on the parameters seen in the previous table while generating the distributions using these parameters. We can see the restrictions in the next table:

Restrictions	Description
$P(w_1) + P(w_2) = 1$	The sum of both probabilities must be equal to one
$d\prime \leq d$	The total number of relevant feature should be lower or equal the total number of features
$v > 0$	The diagonal values of the covariance matrix must be positive

Table 3.4: Classification (II): Parameters Restrictions

The classification dataset generating process is explained in the follow algorithm:

Algorithm 5 Classification dataset (II)

```
1: function GENERATECLASIFICATIONDATASET($n, d, d\prime, m, k, a, b, priors, sdr$)
2: $n$: number of samples
3: $d$: total number of features
4: $d\prime$: number of relevant features
5: $v$: diagonal values of covariance matrix
6: $priors$: $P(w_1), P(w_2)$
7: $sdr$: standard deviation for relevance vector
8:
9:     if sum($priors$)$\neq 1$ then
10:        $print$("ERROR: The sum of probabilities should be 1.")
11:        $return()$
12:    else
13:        $Covar = getSigma(d, k, a, b)$                    ▷ Get covariance matrix $\Sigma_{dxd}$
14:        $xlist = list()$                 ▷ Initialize the list that will contain the distribution
15:        for $i = 1 \to n$ do
16:            $pw1 = runif(1, min = 0, max = 1)$   ▷ Generate random number between 0 and 1
17:            if $pw1 \leq priors[1]$ then                    ▷ check to which $P(w_i)$ belongs
18:                $mu.f = getMu(d, d\prime, m)$          ▷ Get the mean vector for probability $P(w_1)$
19:            else
20:                $mu.f = -getMu(d, d\prime, m)$         ▷ Get the mean vector for probability $P(w_2)$
21:            end if
22:            $xlist[[i]] = mvrnorm(1, mu = mu.f, Sigma = Covar)$  ▷ Generate one row of data
23:        end for
24:        $x = matrix(unlist(xlist), ncol = d, byrow = TRUE)$     ▷ Save the data matrix into x
25:        $r = GenerateRi(d, dR, sdr)$                          ▷ Generate the vector $R_i$
26:        $classes = as.factor(apply(x, 1, function(x)sign(sum(x * r))))$   ▷ Assign each sample
   to one class or another
27:        $r_i^* = GenerateRiStar(r)$                     ▷ Generate the relevance vector $r_i^*$
28:        $return(list(data.frame(classes, x), rStar))$ ▷ Return dataset and his Relevance vector
29:    end if
30: end function
```

Figure 3.4: Algorithm: Classification dataset (II)

Now we want to distribute the data among two classes, which means we will work with binary class datasets. The next equation is used to divide data between two classes:

$$y = \mathrm{sign}(\textstyle\sum_{i=1}^{d} r_i x_i)$$

where we choose $sign(z)$ to assign the sample to one class or another. Detailed description is in the following equation:

$$sign(z) = \begin{cases} 1, & z > 0 \text{ , if z is positive, sample belongs to class 1} \\ -1, & z < 1 \text{ , if z is negative, sample belongs to class 2} \end{cases}$$

3.1.3 Regression dataset

In regression type of dataset we have a set of feature $X = (X_1, \ldots, X_d)$ of length d and the target is of the form

$$b = r_0 + \sum_{i=1}^{d} r_i x_i + \varepsilon$$

where r_i denotes the relevance of the feature x_i and it is generated using the normal distribution of the form

$$r_i \sim \mathcal{N}\left(0, \sigma_r^2\right), \qquad \text{where } \sigma_r^2 \text{ is the variance}$$

In the target equation b, x_i is generated from uniform distribution between $[-5, 5]$ of the form

$$x_i \sim \mathcal{U}\left(-5, 5\right)$$

and the independent term ε is also obtained from normal distribution of the form

$$\varepsilon \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right), \qquad \text{where } \sigma_\varepsilon^2 \text{ is the variance}$$

The **relevance vector** (r_i^*) is the same form as the previous types of datasets and follows the equation

$$r_i^* = \frac{|r_i|}{\sum_{j=1}^{d} |r_j|}$$

The regression dataset generating process is explained in the following algorithm:

Algorithm 6 Regression dataset

```
1: function GENERATEREGRESSIONDATASET($n, d, \sigma_x^2, \sigma_r^2, \sigma_{r_0}^2, \sigma_\varepsilon^2$)
2: $n$: number of samples
3: $d$: total number of features
4: $a$: Range for the uniform distribution
5: $\sigma_{r_0}^2$: Variance of the initial relevance vector
6: $\sigma_r^2$: Variance for relevance vector
7: $\sigma_\varepsilon^2$: Variance for the independent term
8: Initialize the target vector $b$
9:
10:     $x = \mathcal{U}(-a, a)$                          ▷ Generate the $nxd$ dimension data-matrix
11:     $r = \mathcal{N}(0, \sigma_r^2)$                  ▷ Generate the relevance vector for $d$ features
12:     $r_0 = \mathcal{N}(0, \sigma_{r0}^2)$             ▷ Generate the initial relevance vector
13:     $\varepsilon = \mathcal{N}(0, \sigma_\varepsilon^2)$   ▷ Generate the independent term
14:     $rx = r * x$              ▷ dot product of the feature vector $x$ and the relevance vector $r$
15:     $b = r0 + rx + \varepsilon$                       ▷ Calculate the target vector
16:     $dataframe = data.frame(cbind(x, b))$  ▷ Combine the data-matrix with the target vector
17:     $r^* = \frac{|r_i|}{\sum_{j=1}^{d}|r_j|}$         ▷ Calculate the relevance vector
18:     $return(list(dataframe, r^*))$        ▷ return the dataframe and the relevance vector
19: end function
```

Figure 3.5: Algorithm: Regression dataset

4

EXPERIMENTAL EVALUATION

Before we complete our description of the distribution model and move to the simulation setup, we comment that the models designed here are not intended to cover every possible real scenario.

4.1 Resampling

The main question arising in a feature selection experimental design is: what are the aspects that we would like to evaluate of a FWA ($\mathcal{A}$)[1] solution in a given data set? Certainly a *good* algorithm is one that maintains a well-balanced trade-off between small-sized and competitive solutions. To assess these two issues at the same time is a difficult undertaking in practice, given that their optimal relationship is user-dependent. In the present controlled experimental scenario, the task is greatly eased since the size and performance of the optimal solution is known in advance. The aim of the experiments is precisely to contrast the ability of the different FWAs to hit a solution with respect to relevance, irrelevance and sample size.

Relevance: Different families of problems are generated by varying the number of relevant features N_R. These are features that will have an influence on the output and whose role can not be assumed by any other subset.

Irrelevance: Irrelevant features are defined as those not having any influence on the output. Their values are generated at random for each example. For a problem with N_R relevant features, different numbers of irrelevant features N_I are added to the corresponding data sets (thus providing with several subproblems for each choice of N_R).

Sample Size: number of instances $|S|$ of a data sample S. In these experiments,

$$|S| = 10^k, \qquad \text{where } k = \begin{cases} 2, & \text{for small size dataset} \\ 3, & \text{for medium size dataset} \\ 4, & \text{for big size dataset} \end{cases}$$

[1]Feature weighting algorithms

4.1.1 Process

1. First of all we will generate independent data for the training set (TR), validation set (VA) and a huge test set (TE) of fixed size 10^5.

2. For each TR, VA and TE we have to adjust the learner ($\mathcal{L}$)[2] in $TR \cup VA$.

3. For each learner ($\mathcal{L}$) that applies, we have to readjust
   - **Relief**, is adjusted in the union of both ($TR \cup VA$)
   - **Regressions**, is adjusted also in $TR \cup VA$
   - **Random Forests**[3] and **SVM-Linear**[4] we adjust; $\forall$ value of the parameter we adjust in training (TR) and we choose the value which gives the minimum error value in validation (VA).

4. (SVM-Linear | RF) $\rightarrow$ readjust in ($TR \cup VA$) with the best parameter found in the previous step.

5. Extract the weights from the $\mathcal{L}$, normalize them applying the same procedure when we generate the relevance vector, the formula used for normalization is ($r_i^{\mathcal{L}} = \frac{r_i^{\mathcal{L}}}{\sum r}$) and finally compare the normalized weights with the known relevance applying the score formula.

6. Finally, predict the model in the huge TE generated previously, obtain the estimated probability of error in the classification problem, square error (e^2) in the regression problem and at last make the linear correlation between the score obtained in the previous step and the variables calculated in this step.

The whole process from 1 to 6 is performed N times and it's gives us three vectors of the form

$$\forall\,(\mathcal{L}, |S|, Dataset) \begin{cases} \hat{P}_e \rightarrow\ (C|R), \text{ errors in all sets of } (TE)\ (5) \\ e^2 \rightarrow\ \text{Square error from the prediction done} \\ \text{in the huge test set from step (5)} \\ \text{Scores} \rightarrow \text{weight comparison between extracted normalized} \\ \text{weights (4) and the real weights or known relevance} \end{cases}$$

where $\hat{P}_e$ is the estimated probability of error of the model.

[2]classifiers such as Relief, Random Forests, etc.
[3]In *RF* we adjust the parameter *mtry*
[4]In *SVM-Linear* we adjust the *cost* parameter

4.2 Score

In this section we will give and idea of having a *scoring measure* to compute the goodness of a solution obtained from the *Feature weighting algorithm* (FWA) ($\mathcal{A}$) with the known optimal solution (r^*). The idea is to check how much $r_{\mathcal{A}}$, solution or features relevance estimated by the $\mathcal{A}$, and r^*, theoretical relevance, have in common. To do this we compute the normalized distance between the relevance vector generated at beginning (known solution) and the relevance estimated by the *Feature weighting algorithm.* To sum up, the *score* $S_X(\mathcal{A}) : \mathcal{P}(X) \to [0,1]$ is defined in terms of the normalized distance between the two weights vectors ($r_{\mathcal{A}}, r^*$).

4.2.1 Construction of the score

The score function receives two weights vectors of the same length d, if length of r^* is different from length of $r_{\mathcal{A}}$ an error message will be produced. The score function follows the next equation

$$S_X(\mathcal{A}) = 1 - \frac{\|r_{\mathcal{A}} - r^*\|}{\sqrt{d}} \in [0,1]$$

where difference between the two weights vector is the norm and is defined as

$$\|r_{\mathcal{A}} - r^*\| = \sqrt{\sum_{i=1}^{d} (r_{i\mathcal{A}} - r_i^*)^2}$$

So the correct solution given by the $\mathcal{A}$ scoring measure must be near 1 instead of 0.

4.3 Experimental setup

The basic idea consists on generating sample data sets using synthetic functions f with known relevant features (X_R) and irrelevant features (X_I). Up to five *Feature weighting algorithms* (FWAs) were used in the experiments. These are RELIEF, RANDOM FORESTS, SVM-LINEAR for both classification and regression, LOGISTIC REGRESSION for classification type of problem and LINEAR REGRESSION for regression datasets. In this case I have adapted the SVM-LINEAR to obtain the weights from the model return by the SVM with linear kernel. Also adapted version of original RELIEF is used to extract the weights. As the FWAs, the idea is basically extract the relevance given to each attributes.

All the experiments are carried out fixing the total numbers of features. In this case I have used total number of features $d = 50$, where $X_R = 30$ are relevant features and $X_I = 20$ are irrelevant ones. A total number of three families of datasets were generated studying two different problems (*Classification* and *Regression*).

The experiment is divided in two main groups. The first group explores the relationship between *score* obtained from the FWAs and the known optimal solution. Meanwhile the second studies the relationship of the linear correlation between the *score*, the *estimated probability of error* (P_e) and the *square error* (e^2). Each group use the three families of problem (classification (I), classification (II) and regression). Also consider that more than three hundreds datasets were generated between training, validation and test set in this experiment, varying the number of sample size from 100 to 1000.

4.4 Summary of results

Details results of the experiments from the different problems can be found in the Appendix. In this chapter I present a brief summary of the results and discuss about them.

4.4.1 Score, estimated probability of error (Pe) and square error

The next table contains a summary of the results presented in the previous tables. Each row define the size of dataset. In the columns we have the average score, probability of error of the model (Pe) and square error (e^2) of fifty runs from all classifiers.

The square error (e^2) is calculated as:

$$e^2 = \frac{\sum_{i=1}^{n}(b_i - y_i)^2}{length(b)} \tag{4.1}$$

where y is the predicted vector in the huge test set and b is the target vector

SCORE vs P_e/e^2								
		Relief	RF		SVM		Log-R/Linear-R[5]	
	ss[6]	mean	mean	Pe/e^2	mean	Pe/e^2	mean	Pe/e^2
Classification(I)	100	0.89	0.88	0.52	0.88	0.51	0.88	0.44
	1000	0.89	0.88	0.49	0.88	0.49	0.88	0.47
Classification(II)	100	0.87	0.88	0.06	0.87	0.26	0.87	0.31
	1000	0.87	0.89	0.04	0.86	0.01	0.87	0.30
Regression	100	0.98	0.98	0.02	0.98	0.03	0.98	0.03
	1000	0.98	0.99	0.03	0.98	0.04	0.98	0.03

[1]Log-R: Logistic regression for datasets of type classification
[1]Linear-R: Linear regression for datasets of type regression
[2]Sample size

Table 4.1: Average Score, Pe/(e^2) after 50 runs - Classification (I - II) and Regression

SCORE vs SD									
		Relief		RF		SVM		Log-R/Linear-R[7]	
	ss[8]	mean	SD	mean	SD	mean	SD	mean	SD
Classif.(I)	100	0.89	0.0035	0.88	0.0028	0.88	0.0027	0.88	0.0030
	1000	0.89	0.0028	0.88	0.0022	0.88	0.0032	0.88	0.0033
Classif.(II)	100	0.87	0.0019	0.88	0.0029	0.87	0.0011	0.87	0.0015
	1000	0.87	0.0019	0.89	0.0014	0.86	0.0008	0.87	0.0010
Regression	100	0.98	0.0021	0.98	0.0015	0.98	0.0020	0.98	0.0020
	1000	0.98	0.0025	0.99	0.0015	0.98	0.0025	0.98	0.0022

[1]Log-R: Logistic regression for datasets of type classification
[1]Linear-R: Linear regression for datasets of type regression
[2]Sample size

Table 4.2: Average Score and SD after 50 runs - Classification (I - II) and Regression

The next table contains a summary of the both previous tables presented. In the columns we have the average score, SD, Pe and the square error (e^2) after fifty runs.

SCORE, SD & Pe/e^2												
		Relief		RF			SVM			Log-R/Linear-R[9]		
	ss[10]	mean	SD	mean	SD	Pe/e^2	mean	SD	Pe/e^2	mean	SD	Pe/e^2
Classification(I)	100	0.89	0.0035	0.88	0.0028	0.52	0.88	0.0027	0.51	0.88	0.0030	0.44
	1000	0.89	0.0028	0.88	0.0022	0.49	0.88	0.0032	0.49	0.88	0.0033	0.47
Classification(II)	100	0.87	0.0019	0.88	0.0029	0.06	0.87	0.0011	0.26	0.87	0.0015	0.31
	1000	0.87	0.0019	0.89	0.0014	0.04	0.86	0.0008	0.01	0.87	0.0010	0.30
Regression	100	0.98	0.0021	0.98	0.0015	0.02	0.98	0.0020	0.03	0.98	0.0020	0.03
	1000	0.98	0.0025	0.99	0.0015	0.03	0.98	0.0025	0.04	0.98	0.0022	0.03

[1]Log-R: Logistic regression for datasets of type classification

[1]Linear-R: Linear regression for datasets of type regression

[2]Sample size

Table 4.3: Average Score, SD and Pe/(e^2) after 50 runs - Classification (I - II) and Regression

In the previous table for the problems of type classification (I) and classification (II) for both sample size (100 and 1000) we can see the mean score is around 90% in all classifiers, the SD vary between the values 0.0008 and 0.0035; meanwhile, the probability of error of the model is around 45% in the classification (I) problem and between 1% and 30% for problems of type classification (II).

In regression type problems the mean score increase and goes around 98% in all classifiers for both sample size. The SD measure oscillates between 0.0015 and 0.0025; meanwhile the square error (e^2) present also small values that goes from 0.02 to 0.04 for all classifiers and sample size.

5 CONCLUSIONS

This work has presented various models working with *feature weighting algorithms.* This work differs slightly from existing approach in that it uses *feature weighting algorithms* instead of pure *feature selection algorithms.*

In the summary of results we can see that the scoring measure returns acceptable values between 88% and 98% for all type of problems and all sample size. For classification (I) we have obtained around 50% of probability of error (P_e) when we predict the models returned from the classifiers in the huge test-set of sample size ($ss = 10^5$). The SD measure is really small in all type of problems independently of sample size because the variance in the output vector given by the classifiers over fifty runs is inappreciable. The mean value is also stable in all cases and there is no dispersion or huge variation in the output.

For example, for classification (I) in the classifier *RF* for the sample size 100 we have got 0.52 of (P_e) while for sample size 1000 it decrease three points and register the value of 0.49, this fact indicates that increasing the sample size could improve the prediction of the model and reduce the probability of error. It is also the same case in the classifier *SVM.* In this particular case the model prediction done in the test-set is not as good as expected initially.

The (P_e) decrease significantly for problems of type classification (II). With ss=100 we have 0.06 of P_e in RF, 0.26 of P_e in SVM and 0.31 of P_e in Logistic regression; increasing the sample size to one thousand (ss=1000) the P_e values is improved in all classifiers, decreasing to 0.04 in RF, a huge improvement in the case of SVM[1] with the P_e value of 0.01 and slightly better value that falls to 0.30 in Logistic regression.

For the regression type of problems the mean score is around 98% in all cases independently of sample size, which indicate that the model fitted well the data. Also the square error (e^2) value is very low in all cases and is around 0.03. So the estimation done by the classifiers does not differs too much.

I would like to comment that I had execution problems with larger sample size of more than 1000 because apparently my computer was not able to produce results because of the memory allocation failure. For this reason I could not perform the experiment with large sample size as expected in the initial goals.

[1]Linear Support Vector Machines

5.1 Future work

Future work could include more *Feature Weighting Algorithms* and run the experiment by varying the sample size. Various cases can be analysed in the experimental part as analysing the behaviour of classifiers varying the number of features. Another option could be to include the concerns of working with different types of problems and datasets. This work has focused primarily on two types of synthetic datasets (classification and regression problems). One could look to work with real data.

Another interesting topic would be to extend the scoring measure and develop other scoring system. Since in this work we focus on computing the normalized distance between relevance vector obtained from the classifier and theoretical relevance vector.

In the synthetic datasets we worked with relevant and irrelevant features. It would be interesting to introduce features redundant, corrupt and include noise datasets to see the behaviour of classifiers in distorted environments. In the literature various works has been done using high-dimension data but primarily these works are focused on pure *feature selection.* So, the carried out experiments in this work are a small contribution in the works done in *feature weighting* area.

Bibliography

[Aha and Bankert, 1994] Aha, D. W. and Bankert, R. L. (1994). A comparative evaluation of sequential feature selection algorithms. 1

[Aizerman et al., 1964] Aizerman, M. A., Braverman, E. A., and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control,*, number 25, pages 821–837. 19

[Ben-Bassat, 1982] Ben-Bassat, M. (1982). Use of distance measures, information measures and error bounds in fuature evaluation. *In Krishnaiah, P. R. and Kanal, L. N., eds., Hand- book of Statistics, North Holland.*, 2:773–791. 2

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Workshop on Computational Learning Theory.* 19

[Doak, 1992] Doak, J. (1992). An evaluation of feature selectionmethods and their application to computer security. Technical report cse–92–18, davis, ca, University of California, Department of Computer Science. 1

[Hosmer and Lemeshow, 2000] Hosmer, D. W. and Lemeshow, S. (2000). *Applied logistic regression (Wiley Series in probability and statistics).* Wiley-Interscience Publication, 2 edition. 23

[Hsu et al., 2003] Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University. 20

[Jain and Zongker, 1997] Jain, A. K. and Zongker, D. E. (1997). Feature selection: Evaluation, application, and small sample performance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(2):153–158. 1

[Kudo, 1997] Kudo, M. (1997). A comparative evaluation of medium and large–scale feature selectors for pattern classifiers. *In Proc. of the 1st International Workshop on Statistical Techniques in Pattern Recognition, Czech Republic.*, pages 91–96. 1

[Liu, 1998] Liu, H. (1998). Scalable feature selection for large sized databases. *In Proc. of the 4th World Congress on Expert Systems*, pages 68–75. 1

[Liu and Motada, 1998] Liu, H. and Motada, H. (1998). *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers. 13

[Reunanen, 2003] Reunanen, J. (2003). Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3:1371–1382. 2

[Thrun et al., 1991] Thrun, S. B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., Jong, K. D., Džeroski, S., Fahlman, S. E., Fisher, D., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R. S., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., de Welde, W. V., Wenzel, W., Wnek, J., and Zhang, J. (1991). The MONK's problems: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. 1

Appendix A

The First Appendix

Detailed results of the experimental work can be found in this chapter.

A.1 Results of datasets classification (I)

In the next table we can find the score and the estimated probability of error ($\hat{P}_e$) obtained from fifty runs for the small size of datasets with sample size (**ss = 100**) from the different algorithm (Logistic Regression, Relief, Random Forest, SVM-Linear).

Score and Probability of error ($\hat{P}_e$)							
	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
1	0.88	0.89	0.88	0.89	0.44	0.61	0.59
2	0.89	0.88	0.88	0.89	0.44	0.34	0.32
3	0.89	0.88	0.89	0.88	0.44	0.31	0.33
4	0.88	0.88	0.88	0.88	0.44	0.78	0.71
5	0.88	0.88	0.88	0.89	0.44	0.53	0.48
6	0.89	0.88	0.89	0.88	0.44	0.39	0.39
7	0.88	0.88	0.88	0.88	0.44	0.84	0.78
8	0.88	0.88	0.88	0.89	0.44	0.53	0.57
9	0.89	0.89	0.89	0.89	0.44	0.88	0.85
10	0.89	0.88	0.89	0.89	0.44	0.66	0.62
							continued on next page

continued from previous page

	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
11	0.89	0.88	0.89	0.89	0.44	0.13	0.18
12	0.88	0.88	0.88	0.89	0.44	0.34	0.39
13	0.88	0.89	0.89	0.89	0.44	0.51	0.49
14	0.89	0.88	0.89	0.89	0.44	0.24	0.30
15	0.88	0.88	0.88	0.88	0.44	0.44	0.42
16	0.88	0.88	0.88	0.88	0.44	0.27	0.39
17	0.89	0.89	0.89	0.89	0.44	0.38	0.36
18	0.89	0.88	0.88	0.88	0.44	0.63	0.65
19	0.89	0.89	0.89	0.89	0.44	0.16	0.27
20	0.88	0.88	0.88	0.89	0.44	0.40	0.40
21	0.89	0.88	0.89	0.89	0.44	0.70	0.62
22	0.88	0.88	0.88	0.88	0.44	0.74	0.67
23	0.89	0.88	0.89	0.89	0.44	0.33	0.39
24	0.89	0.88	0.89	0.89	0.44	0.49	0.44
25	0.88	0.88	0.88	0.88	0.44	0.51	0.54
26	0.89	0.88	0.89	0.88	0.44	0.71	0.69
27	0.89	0.89	0.89	0.89	0.44	0.76	0.62
28	0.88	0.88	0.88	0.89	0.44	0.34	0.41
29	0.89	0.88	0.89	0.89	0.44	0.79	0.69
30	0.89	0.88	0.89	0.89	0.44	0.41	0.42
31	0.89	0.88	0.89	0.89	0.44	0.47	0.44
32	0.88	0.88	0.88	0.88	0.44	0.71	0.63
33	0.89	0.88	0.89	0.89	0.44	0.31	0.40
34	0.89	0.88	0.89	0.88	0.44	0.53	0.55
35	0.88	0.88	0.88	0.88	0.44	0.79	0.75
36	0.89	0.88	0.89	0.88	0.44	0.62	0.63
37	0.88	0.88	0.88	0.88	0.44	0.80	0.74
38	0.89	0.88	0.89	0.89	0.44	0.51	0.53
39	0.88	0.88	0.88	0.89	0.44	0.20	0.29
40	0.88	0.88	0.88	0.88	0.44	0.61	0.53
41	0.89	0.88	0.88	0.89	0.44	0.58	0.54

continued on next page

continued from previous page							
	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
42	0.89	0.88	0.89	0.89	0.44	0.67	0.63
43	0.89	0.89	0.89	0.90	0.44	0.54	0.49
44	0.88	0.88	0.88	0.89	0.44	0.51	0.54
45	0.89	0.89	0.89	0.89	0.44	0.61	0.56
46	0.89	0.88	0.89	0.89	0.44	0.35	0.38
47	0.89	0.88	0.88	0.89	0.44	0.53	0.51
48	0.89	0.89	0.89	0.89	0.44	0.80	0.76
49	0.89	0.88	0.89	0.89	0.44	0.35	0.36
50	0.89	0.89	0.89	0.89	0.44	0.34	0.43

Table A.1: Score and $\hat{P}_e$ of small datasets with $ss = 100$ - Classification (I)

The next table show the score obtained performing the experiment fifty times against a medium size dataset with sample size **ss** = **1000**.

Score and Probability of error ($\hat{P}_e$)							
	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
1	0.88	0.89	0.88	0.89	0.47	0.40	0.46
2	0.88	0.88	0.88	0.89	0.47	0.71	0.66
3	0.88	0.88	0.88	0.89	0.47	0.47	0.52
4	0.88	0.88	0.88	0.89	0.47	0.58	0.56
5	0.87	0.88	0.88	0.88	0.47	0.23	0.29
6	0.88	0.88	0.88	0.89	0.47	0.50	0.51
7	0.88	0.88	0.88	0.89	0.47	0.63	0.62
8	0.88	0.88	0.88	0.89	0.47	0.59	0.56
9	0.88	0.88	0.88	0.89	0.47	0.45	0.47
10	0.88	0.88	0.88	0.89	0.47	0.48	0.49
11	0.88	0.88	0.88	0.89	0.47	0.56	0.53
12	0.88	0.88	0.88	0.89	0.47	0.50	0.50
13	0.88	0.88	0.88	0.89	0.47	0.64	0.61
14	0.88	0.88	0.88	0.89	0.47	0.43	0.48
15	0.88	0.88	0.88	0.89	0.47	0.48	0.47
16	0.88	0.88	0.88	0.89	0.47	0.25	0.29
17	0.88	0.88	0.88	0.89	0.47	0.39	0.39
18	0.87	0.88	0.87	0.89	0.47	0.32	0.32
19	0.88	0.88	0.88	0.89	0.47	0.64	0.58
20	0.88	0.88	0.88	0.88	0.47	0.43	0.42
21	0.88	0.88	0.88	0.89	0.47	0.74	0.61
22	0.88	0.88	0.88	0.89	0.47	0.62	0.61
23	0.88	0.88	0.88	0.89	0.47	0.60	0.60
24	0.88	0.88	0.88	0.89	0.47	0.54	0.53
25	0.88	0.88	0.88	0.89	0.47	0.19	0.23

continued on next page

continued from previous page

	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
26	0.88	0.88	0.88	0.89	0.47	0.41	0.43
27	0.87	0.88	0.87	0.89	0.47	0.33	0.35
28	0.88	0.88	0.88	0.88	0.47	0.63	0.58
29	0.88	0.88	0.88	0.88	0.47	0.23	0.26
30	0.88	0.88	0.88	0.89	0.47	0.73	0.65
31	0.89	0.88	0.88	0.89	0.47	0.66	0.60
32	0.87	0.88	0.87	0.89	0.47	0.79	0.72
33	0.88	0.88	0.88	0.89	0.47	0.41	0.42
34	0.88	0.88	0.88	0.89	0.47	0.24	0.31
35	0.88	0.88	0.88	0.89	0.47	0.53	0.54
36	0.88	0.88	0.88	0.88	0.47	0.48	0.48
37	0.88	0.88	0.88	0.89	0.47	0.38	0.42
38	0.89	0.88	0.89	0.89	0.47	0.42	0.46
39	0.88	0.88	0.88	0.88	0.47	0.58	0.55
40	0.88	0.88	0.88	0.88	0.47	0.24	0.28
41	0.89	0.88	0.89	0.89	0.47	0.51	0.51
42	0.88	0.88	0.88	0.88	0.47	0.73	0.71
43	0.88	0.88	0.88	0.89	0.47	0.38	0.38
44	0.89	0.88	0.88	0.89	0.47	0.46	0.48
45	0.88	0.88	0.88	0.89	0.47	0.44	0.48
46	0.88	0.88	0.88	0.89	0.47	0.51	0.50
47	0.88	0.88	0.88	0.89	0.47	0.45	0.49
48	0.89	0.89	0.89	0.89	0.47	0.71	0.69
49	0.88	0.88	0.88	0.89	0.47	0.43	0.45
50	0.88	0.88	0.88	0.89	0.47	0.42	0.43

Table A.2: Score and $\hat{P}_e$ of medium size datasets with $ss = 1000$ - Classification (I)

A.2 Results of datasets classification (II)

In the next table we can find the score obtained from fifty runs for the small size of datasets with sample size (**ss = 100**) from the different algorithm (Logistic Regression, Relief, Random Forest, SVM-Linear).

Score and Probability of error ($\hat{P}_e$)							
	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
1	0.87	0.88	0.87	0.87	0.31	0.07	0.31
2	0.87	0.88	0.87	0.87	0.31	0.06	0.31
3	0.88	0.88	0.87	0.88	0.31	0.07	0.31
4	0.87	0.88	0.87	0.87	0.31	0.04	0.31
5	0.87	0.88	0.87	0.87	0.31	0.05	0.31
6	0.87	0.88	0.87	0.87	0.31	0.04	0.02
7	0.87	0.88	0.87	0.88	0.31	0.05	0.31
8	0.87	0.88	0.87	0.87	0.31	0.05	0.31
9	0.87	0.88	0.87	0.87	0.31	0.06	0.31
10	0.87	0.88	0.87	0.87	0.31	0.05	0.01
11	0.87	0.88	0.87	0.88	0.31	0.05	0.31
12	0.87	0.88	0.87	0.87	0.31	0.06	0.31
13	0.87	0.88	0.87	0.87	0.31	0.05	0.31
14	0.87	0.87	0.87	0.87	0.31	0.06	0.31
15	0.87	0.88	0.87	0.87	0.31	0.06	0.31
16	0.87	0.88	0.87	0.87	0.31	0.05	0.31
17	0.87	0.88	0.87	0.87	0.31	0.04	0.31
18	0.87	0.88	0.87	0.87	0.31	0.06	0.31
19	0.87	0.88	0.87	0.88	0.31	0.04	0.31
20	0.87	0.87	0.87	0.87	0.31	0.06	0.02
21	0.87	0.88	0.87	0.88	0.31	0.05	0.31
22	0.87	0.88	0.87	0.87	0.31	0.06	0.31
23	0.87	0.88	0.87	0.87	0.31	0.08	0.31

continued on next page

continued from previous page

	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
24	0.87	0.88	0.87	0.87	0.31	0.06	0.31
25	0.87	0.88	0.87	0.88	0.31	0.06	0.02
26	0.87	0.88	0.87	0.87	0.31	0.07	0.31
27	0.87	0.88	0.87	0.88	0.31	0.08	0.31
28	0.87	0.87	0.87	0.87	0.31	0.08	0.31
29	0.87	0.88	0.87	0.88	0.31	0.06	0.31
30	0.87	0.87	0.87	0.87	0.31	0.09	0.31
31	0.87	0.88	0.87	0.87	0.31	0.06	0.31
32	0.87	0.88	0.87	0.87	0.31	0.06	0.31
33	0.87	0.87	0.87	0.88	0.31	0.05	0.02
34	0.87	0.88	0.87	0.88	0.31	0.06	0.31
35	0.87	0.88	0.87	0.87	0.31	0.05	0.31
36	0.87	0.88	0.87	0.87	0.31	0.06	0.31
37	0.87	0.88	0.87	0.87	0.31	0.08	0.31
38	0.87	0.88	0.87	0.87	0.31	0.06	0.31
39	0.87	0.87	0.87	0.87	0.31	0.06	0.31
40	0.88	0.88	0.87	0.87	0.31	0.05	0.02
41	0.87	0.88	0.87	0.87	0.31	0.07	0.31
42	0.87	0.88	0.87	0.87	0.31	0.04	0.31
43	0.87	0.88	0.87	0.88	0.31	0.06	0.31
44	0.87	0.88	0.87	0.87	0.31	0.04	0.31
45	0.87	0.88	0.87	0.88	0.31	0.07	0.31
46	0.87	0.88	0.87	0.87	0.31	0.06	0.31
47	0.87	0.87	0.87	0.88	0.31	0.05	0.02
48	0.87	0.88	0.87	0.87	0.31	0.05	0.31
49	0.87	0.87	0.87	0.87	0.31	0.06	0.03
50	0.87	0.87	0.87	0.87	0.31	0.05	0.31

Table A.3: Score and $\hat{P}_e$ of small datasets with sample size ($ss = 100$) - Classification (II)

The next table show the score obtained performing the experiment fifty times against a medium size of datasets with sample size **ss** = **1000**.

Score and Probability of error ($\hat{P}_e$)							
	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
1	0.86	0.89	0.86	0.88	0.30	0.04	0.01
2	0.87	0.89	0.87	0.88	0.30	0.04	0.01
3	0.86	0.89	0.86	0.87	0.30	0.04	0.02
4	0.86	0.89	0.86	0.87	0.30	0.04	0.01
5	0.86	0.88	0.86	0.87	0.30	0.04	0.01
6	0.87	0.89	0.87	0.87	0.30	0.03	0.01
7	0.86	0.88	0.86	0.87	0.30	0.04	0.01
8	0.87	0.89	0.86	0.87	0.30	0.04	0.02
9	0.86	0.89	0.86	0.88	0.30	0.04	0.01
10	0.87	0.89	0.86	0.87	0.30	0.04	0.02
11	0.87	0.88	0.87	0.87	0.30	0.04	0.01
12	0.87	0.89	0.86	0.87	0.30	0.04	0.02
13	0.86	0.88	0.86	0.88	0.30	0.03	0.01
14	0.86	0.88	0.87	0.87	0.30	0.04	0.02
15	0.87	0.88	0.86	0.87	0.30	0.04	0.02
16	0.87	0.89	0.87	0.87	0.30	0.04	0.02
17	0.87	0.89	0.87	0.88	0.30	0.04	0.02
18	0.86	0.88	0.86	0.87	0.30	0.04	0.02
19	0.87	0.89	0.87	0.87	0.30	0.04	0.02
20	0.86	0.89	0.87	0.87	0.30	0.04	0.01
21	0.87	0.88	0.86	0.87	0.30	0.04	0.02
22	0.87	0.88	0.86	0.88	0.30	0.04	0.02
23	0.87	0.89	0.87	0.87	0.30	0.04	0.01
24	0.86	0.89	0.86	0.87	0.30	0.04	0.01
25	0.87	0.88	0.86	0.87	0.30	0.04	0.02

continued on next page

continued from previous page							
	Score				Probability of error ($\hat{P}_e$)		
	Logistic-R	RF	SVM	Relief	Logistic-R	RF	SVM
26	0.87	0.89	0.86	0.87	0.30	0.04	0.01
27	0.86	0.88	0.86	0.87	0.30	0.04	0.01
28	0.87	0.88	0.87	0.87	0.30	0.04	0.01
29	0.87	0.88	0.87	0.88	0.30	0.04	0.01
30	0.86	0.89	0.86	0.87	0.30	0.04	0.02
31	0.86	0.88	0.86	0.87	0.30	0.04	0.01
32	0.87	0.89	0.87	0.87	0.30	0.04	0.01
33	0.87	0.88	0.87	0.87	0.30	0.04	0.02
34	0.86	0.88	0.86	0.87	0.30	0.04	0.01
35	0.87	0.89	0.87	0.87	0.30	0.04	0.02
36	0.87	0.89	0.87	0.87	0.30	0.04	0.02
37	0.87	0.88	0.87	0.87	0.30	0.04	0.02
38	0.87	0.89	0.87	0.87	0.30	0.04	0.02
39	0.87	0.89	0.87	0.87	0.30	0.04	0.01
40	0.87	0.89	0.86	0.88	0.30	0.04	0.02
41	0.87	0.88	0.87	0.87	0.30	0.04	0.01
42	0.86	0.89	0.87	0.87	0.30	0.04	0.01
43	0.87	0.88	0.86	0.87	0.30	0.04	0.02
44	0.86	0.89	0.86	0.87	0.30	0.04	0.02
45	0.87	0.89	0.87	0.87	0.30	0.04	0.02
46	0.87	0.89	0.86	0.87	0.30	0.04	0.02
47	0.86	0.89	0.86	0.87	0.30	0.04	0.01
48	0.87	0.89	0.87	0.87	0.30	0.04	0.01
49	0.87	0.88	0.86	0.87	0.30	0.03	0.01
50	0.86	0.88	0.86	0.87	0.30	0.04	0.02

Table A.4: Score and $\hat{P}_e$ of medium size datasets with $ss = 1000$ - Classification (II)

A.3 Results of datasets Regression

In the next table we can find the score and square error e^2 obtained from fifty runs for the small size of datasets with sample size (**ss = 100**) from the different algorithm (Linear Regression, Relief for regression, Random Forest for regression, SVM-Linear for regression).

Score							
	Score				Square error (e^2)		
	Linear-R	RF	SVM	Relief	Linear-R	RF	SVM
1	0.98	0.98	0.98	0.98	0.03	0.02	0.03
2	0.98	0.98	0.98	0.98	0.03	0.02	0.04
3	0.98	0.98	0.98	0.98	0.03	0.02	0.03
4	0.98	0.98	0.98	0.98	0.03	0.02	0.04
5	0.98	0.98	0.98	0.98	0.03	0.02	0.04
6	0.98	0.98	0.98	0.98	0.03	0.02	0.03
7	0.98	0.99	0.98	0.98	0.04	0.03	0.04
8	0.98	0.98	0.98	0.98	0.03	0.02	0.03
9	0.98	0.99	0.98	0.98	0.03	0.02	0.03
10	0.98	0.98	0.98	0.98	0.03	0.02	0.03
11	0.98	0.98	0.98	0.98	0.03	0.02	0.04
12	0.98	0.98	0.98	0.98	0.03	0.02	0.03
13	0.98	0.98	0.98	0.98	0.03	0.02	0.03
14	0.98	0.99	0.98	0.98	0.03	0.02	0.04
15	0.98	0.98	0.98	0.98	0.03	0.03	0.03
16	0.98	0.98	0.98	0.98	0.03	0.02	0.03
17	0.98	0.99	0.98	0.98	0.03	0.02	0.03
18	0.98	0.98	0.98	0.98	0.03	0.02	0.03
19	0.98	0.98	0.98	0.98	0.03	0.02	0.03
20	0.98	0.98	0.98	0.98	0.03	0.03	0.04
21	0.98	0.98	0.98	0.98	0.03	0.02	0.04
22	0.98	0.98	0.98	0.98	0.04	0.02	0.04
23	0.98	0.99	0.98	0.98	0.03	0.02	0.04

continued on next page

continued from previous page

	Score				Square error (e^2)		
	Linear-R	RF	SVM	Relief	Linear-R	RF	SVM
24	0.98	0.99	0.98	0.98	0.03	0.02	0.04
25	0.97	0.98	0.98	0.98	0.03	0.02	0.04
26	0.98	0.99	0.98	0.98	0.03	0.02	0.03
27	0.98	0.99	0.98	0.98	0.03	0.02	0.03
28	0.98	0.98	0.97	0.98	0.03	0.03	0.04
29	0.98	0.99	0.98	0.98	0.03	0.02	0.03
30	0.98	0.99	0.98	0.98	0.03	0.02	0.03
31	0.98	0.98	0.98	0.98	0.03	0.02	0.03
32	0.98	0.99	0.98	0.98	0.03	0.02	0.04
33	0.98	0.98	0.98	0.98	0.03	0.02	0.03
34	0.98	0.98	0.98	0.98	0.03	0.03	0.04
35	0.98	0.98	0.98	0.98	0.03	0.02	0.03
36	0.98	0.98	0.98	0.98	0.03	0.02	0.03
37	0.98	0.98	0.98	0.98	0.03	0.02	0.03
38	0.98	0.99	0.98	0.98	0.03	0.02	0.03
39	0.98	0.98	0.98	0.98	0.03	0.03	0.03
40	0.98	0.99	0.98	0.98	0.03	0.02	0.03
41	0.98	0.98	0.98	0.98	0.03	0.02	0.03
42	0.98	0.98	0.98	0.98	0.03	0.02	0.04
43	0.98	0.99	0.98	0.98	0.03	0.02	0.03
44	0.98	0.98	0.98	0.98	0.03	0.02	0.03
45	0.98	0.99	0.98	0.98	0.03	0.02	0.04
46	0.98	0.98	0.98	0.98	0.03	0.02	0.03
47	0.98	0.98	0.98	0.98	0.03	0.02	0.04
48	0.98	0.98	0.98	0.98	0.03	0.02	0.03
49	0.98	0.98	0.98	0.98	0.03	0.02	0.03
50	0.98	0.99	0.98	0.98	0.03	0.03	0.03

Table A.5: Score of small size of datasets with $ss = 100$ - Regression

In the next table we can find the score obtained from fifty runs for the medium size of datasets with sample size (**ss = 1000**) from the different algorithm (Linear Regression, Relief for regression, Random Forest for regression, SVM-Linear for regression).

Score							
	Score				Square error (e^2)		
	Linear-R	RF	SVM	Relief	Linear-R	RF	SVM
1	0.98	0.98	0.98	0.98	0.04	0.04	0.04
2	0.98	0.99	0.98	0.98	0.03	0.03	0.03
3	0.98	0.98	0.98	0.97	0.03	0.03	0.03
4	0.98	0.99	0.98	0.98	0.03	0.03	0.04
5	0.98	0.99	0.98	0.98	0.03	0.03	0.03
6	0.98	0.99	0.99	0.97	0.04	0.03	0.04
7	0.98	0.99	0.98	0.98	0.04	0.03	0.04
8	0.98	0.99	0.98	0.98	0.04	0.03	0.04
9	0.98	0.99	0.98	0.98	0.03	0.03	0.03
10	0.98	0.98	0.98	0.98	0.03	0.03	0.04
11	0.98	0.98	0.98	0.97	0.04	0.03	0.04
12	0.98	0.98	0.98	0.99	0.03	0.03	0.03
13	0.98	0.99	0.98	0.98	0.04	0.03	0.04
14	0.98	0.98	0.98	0.98	0.04	0.03	0.04
15	0.98	0.98	0.98	0.98	0.03	0.03	0.03
16	0.99	0.99	0.98	0.98	0.04	0.03	0.04
17	0.98	0.98	0.98	0.98	0.03	0.03	0.03
18	0.98	0.99	0.98	0.98	0.04	0.03	0.04
19	0.98	0.98	0.98	0.98	0.03	0.03	0.03
20	0.98	0.99	0.98	0.98	0.04	0.03	0.04
21	0.98	0.98	0.98	0.97	0.03	0.03	0.03
22	0.98	0.99	0.98	0.98	0.03	0.03	0.03
23	0.98	0.99	0.98	0.98	0.03	0.03	0.03
24	0.99	0.99	0.99	0.98	0.04	0.03	0.04
25	0.99	0.99	0.99	0.98	0.04	0.03	0.04

continued on next page

continued from previous page							
	Score				Square error (e^2)		
	Linear-R	RF	SVM	Relief	Linear-R	RF	SVM
26	0.98	0.98	0.98	0.98	0.03	0.03	0.04
27	0.98	0.99	0.98	0.98	0.03	0.03	0.04
28	0.98	0.99	0.98	0.98	0.03	0.03	0.04
29	0.98	0.98	0.98	0.98	0.04	0.03	0.04
30	0.98	0.98	0.98	0.98	0.03	0.03	0.04
31	0.98	0.99	0.98	0.98	0.04	0.03	0.04
32	0.98	0.98	0.98	0.98	0.03	0.03	0.04
33	0.98	0.98	0.98	0.98	0.04	0.03	0.04
34	0.98	0.99	0.98	0.98	0.04	0.03	0.04
35	0.98	0.98	0.98	0.98	0.04	0.03	0.04
36	0.98	0.99	0.98	0.98	0.03	0.03	0.03
37	0.98	0.99	0.98	0.98	0.04	0.03	0.04
38	0.98	0.98	0.98	0.98	0.03	0.03	0.03
39	0.98	0.99	0.98	0.98	0.03	0.03	0.03
40	0.98	0.98	0.98	0.98	0.04	0.03	0.04
41	0.98	0.98	0.98	0.98	0.04	0.03	0.03
42	0.98	0.99	0.98	0.98	0.03	0.03	0.04
43	0.98	0.99	0.99	0.98	0.04	0.03	0.04
44	0.98	0.99	0.98	0.98	0.03	0.03	0.03
45	0.98	0.99	0.98	0.98	0.03	0.03	0.03
46	0.98	0.99	0.98	0.98	0.03	0.03	0.04
47	0.98	0.98	0.98	0.98	0.04	0.04	0.04
48	0.98	0.98	0.98	0.98	0.03	0.03	0.04
49	0.98	0.98	0.98	0.98	0.03	0.03	0.04
50	0.98	0.99	0.98	0.98	0.04	0.03	0.04

Table A.6: Score and e^2 of medium size of datasets with $ss = 1000$ - Regression

Appendix B

The Second Appendix

Here we present some graphics figures of the scores obtained in the experimental process.

B.1 Datasets classification (I)

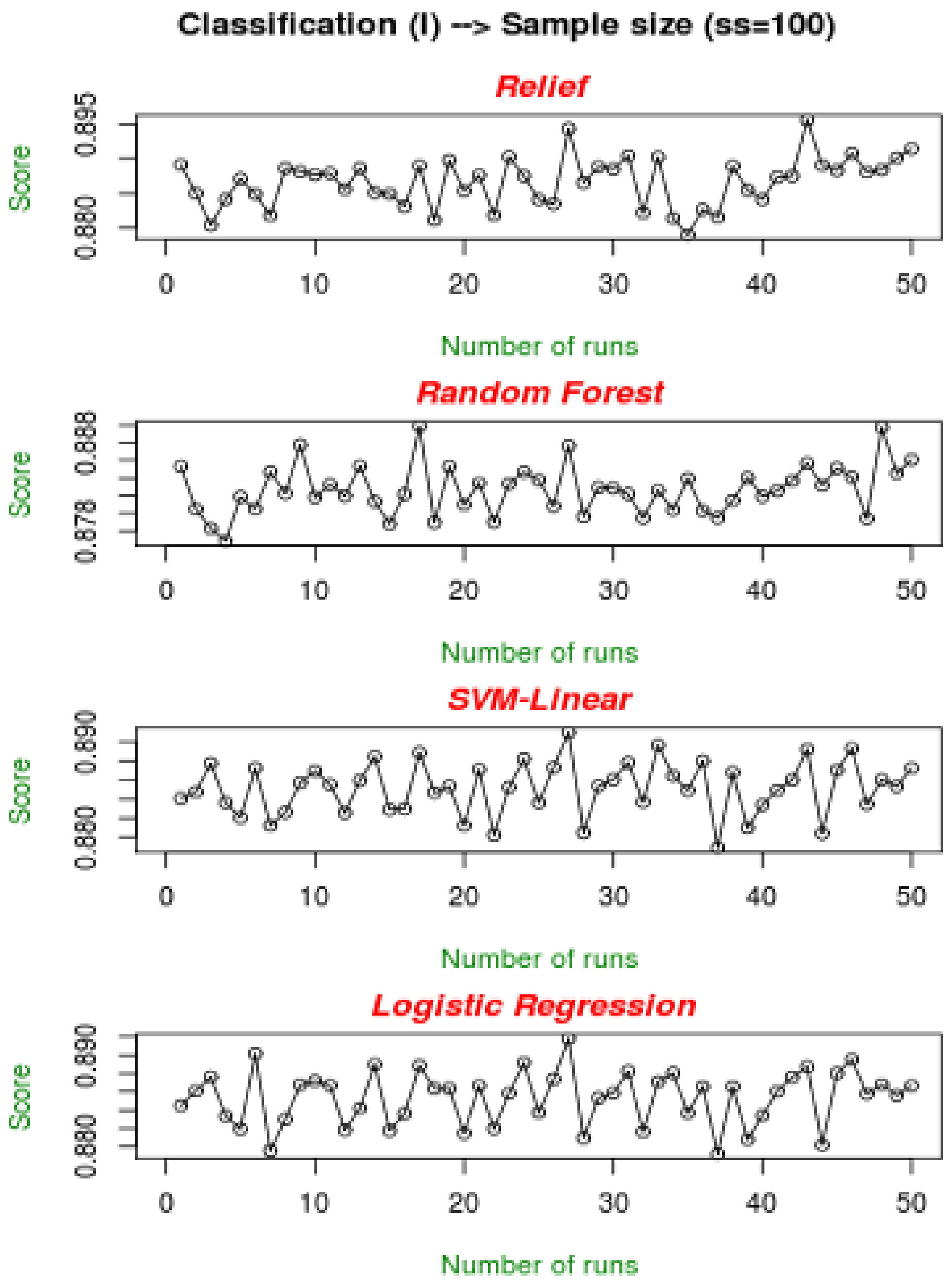

Figure B.1: Classification (I) - Sample size (ss=100)

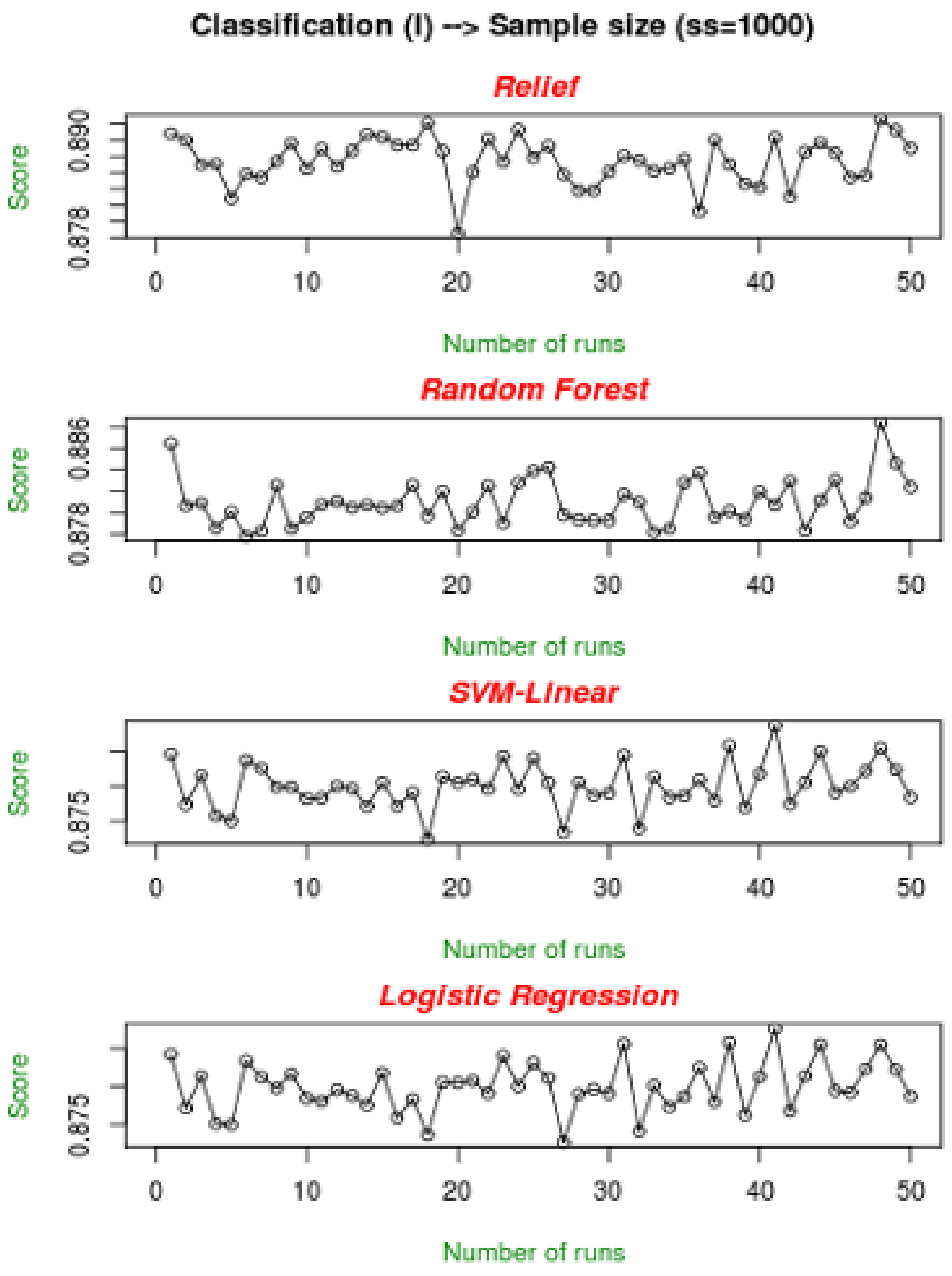

Figure B.2: Classification (I) - Sample size (ss=1000)

B.2 Datasets classification (II)

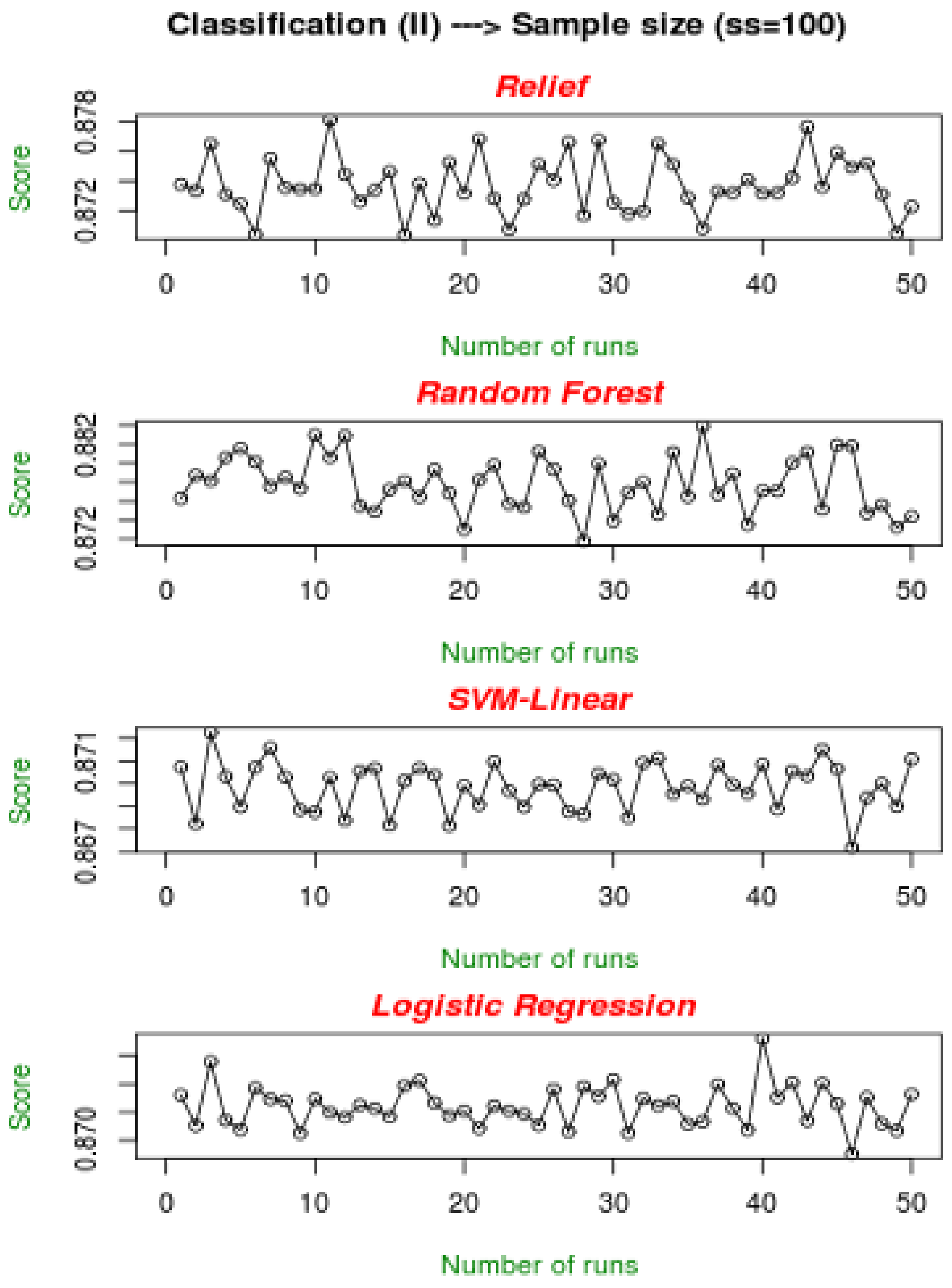

Figure B.3: Classification (II) - Sample size (ss=100)

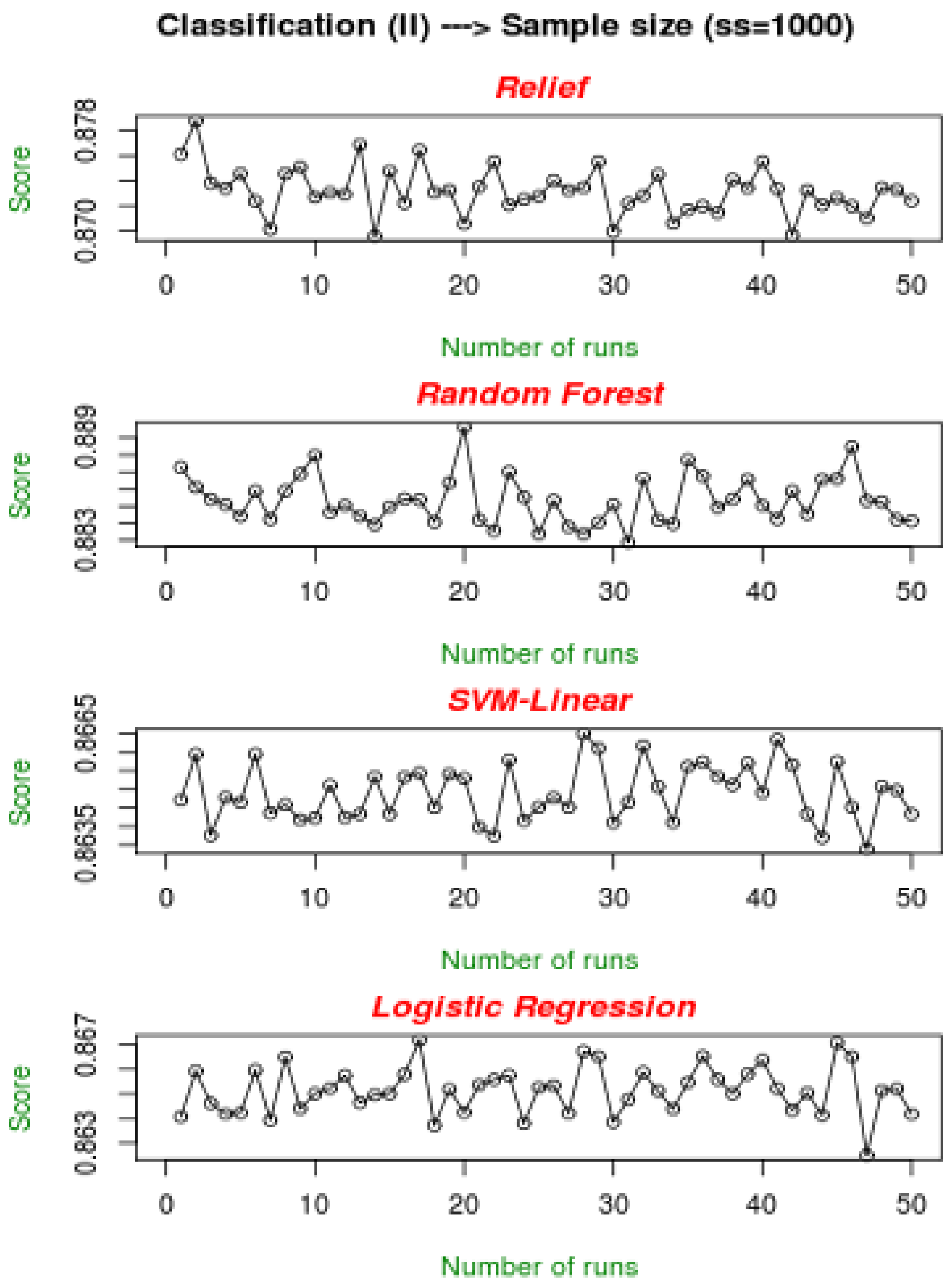

Figure B.4: Classification (II) - Sample size (ss=1000)

B.3 Datasets Regression

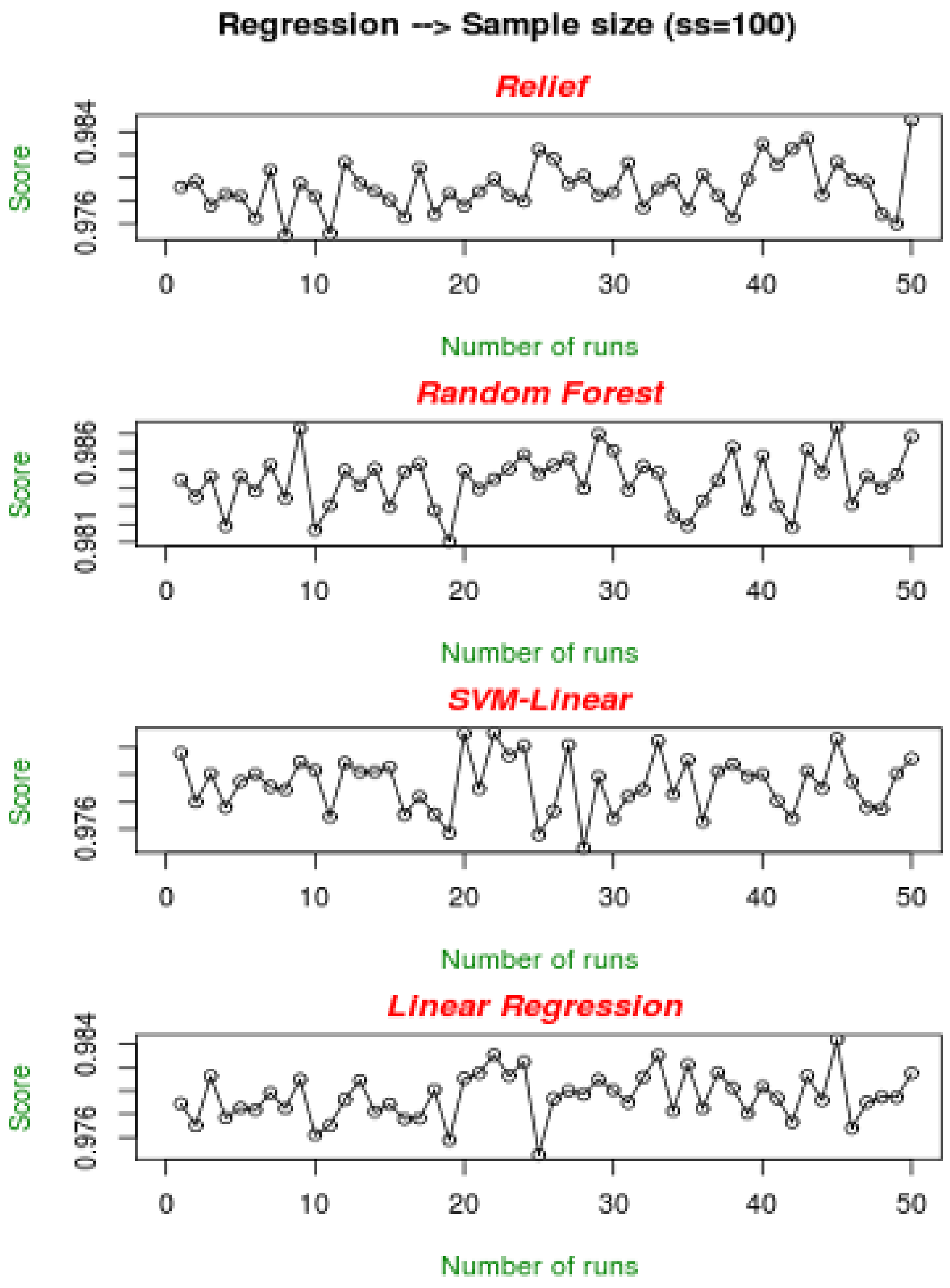

Figure B.5: Regression - Sample size (ss=100)

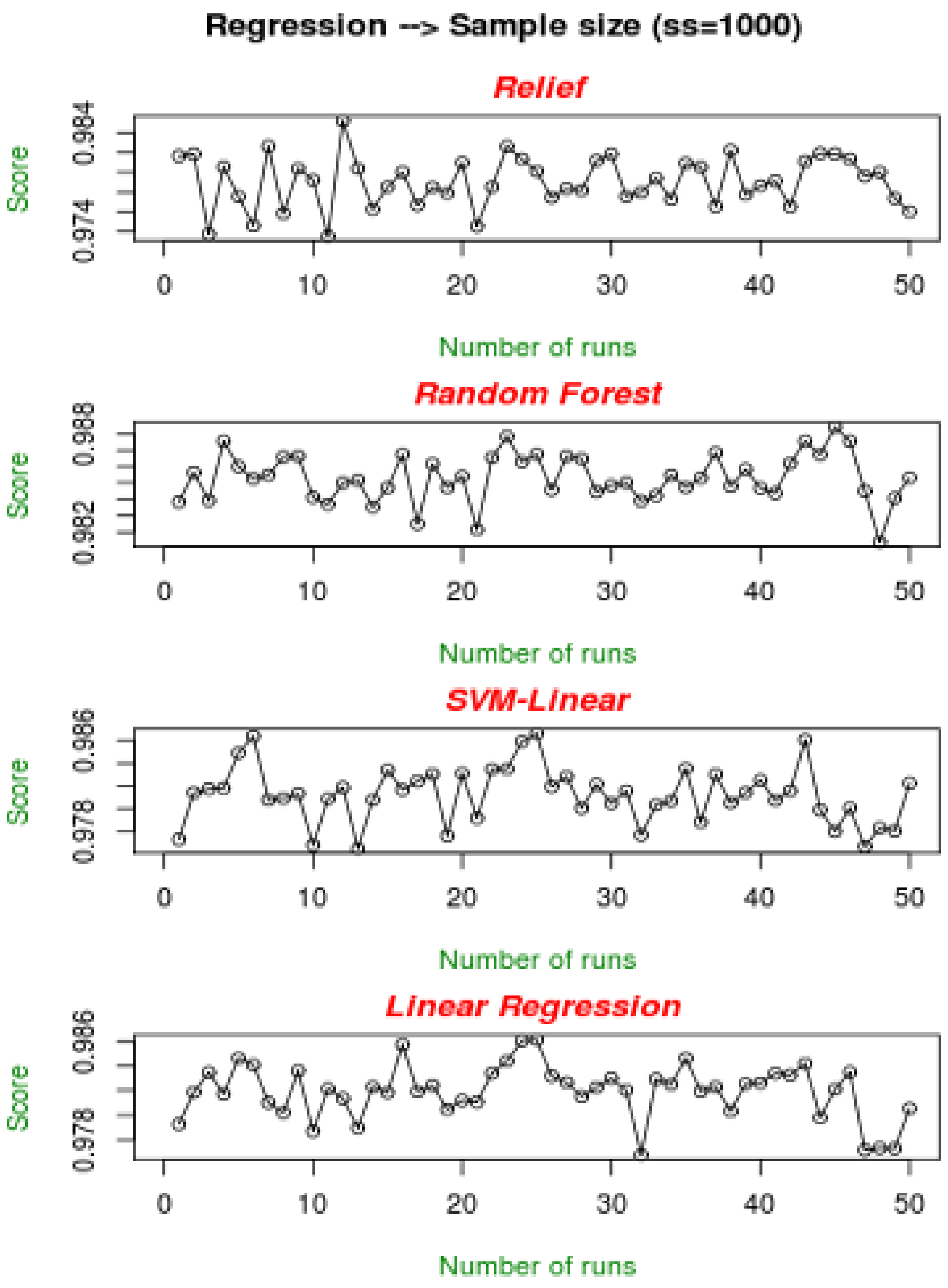

Figure B.6: Regression - Sample size (ss=1000)