Faculty of Applied Sciences

Department of Electrical Engineering and Computer Science

Improvement of randomized ensembles of trees for supervised learning in very high dimension

Author:
Arnaud Joly

Supervisors:
Prof. Louis Wehenkel
Dr. Pierre Geurts

A master thesis submitted in partial fulfilment of the requirements of the master degree in Electrical Engineering

Academic year 2010 - 2011

Abstract

Tree-based ensemble methods, such as random forests and extremely randomized trees, are methods of choice for handling high dimensional problems. One important drawback of these methods however is the complexity of the models (*i.e.* the large number and size of trees) they produce to achieve good performances.

In this work, several research directions are identified to address this problem. Among those, we have developed the following one.

From a tree ensemble, one can extract a set of binary features, each one associated to a leaf or a node of a tree and being true for a given object only if it reaches the corresponding leaf or node when propagated in this tree. Given this representation, the prediction of an ensemble can be simply retrieved by linearly combining these characteristic features with appropriate weights.

We apply a linear feature selection method, namely the monotone LASSO, on these features, in order to simplify the tree ensemble. A subtree will then be pruned as soon as the characteristic features corresponding to its constituting nodes are not selected in the linear model.

Empirical experiments show that the combination of the monotone LASSO with features extracted from tree ensembles leads at the same time to a drastic reduction of the number of features and can improve the accuracy with respect to unpruned ensembles of trees.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

I am grateful to my supervisors Prof. *Louis Wehenkel* and Dr. *Pierre Geurts*, who gave me the opportunity to discover Machine Learning, for their encouragements, guidance and support.

I also offer my regards and blessings to *my family, friends* and *all* of those who supported me in any respect during the completion of this project.

Chapter 1

Introduction

1.1 Context and motivation

Progress in information technology enables the acquisition and storage of growing amounts of rich data in science (biology, high-energy physics, astronomy, etc.), engineering (energy, transportation, production processes, etc.), and society (environment, commerce, etc.). The accumulating datasets come in various forms such as images, videos, time-series of measurements, recorded transactions, text, etc. WEB technology often allows one to share locally acquired datasets, and numerical simulation often allows one to generate low cost datasets on demand. Many opportunities exist thus for combining datasets from different sources to search for generic knowledge.

The analysis of such amounts of data is almost intractable without the help of computing technologies. *Machine Learning*, a field at the convergence of statistics, mathematics and computer science, provides automatic procedures to extract knowledge from data and summarises these through a model. *Supervised learning*, a machine learning task, works at finding a function to model the relationship between inputs and outputs of a *system*, and to *predict* its future outputs given its inputs.

Decision trees, a class of supervised learning methods, produce rather intuitive, fast and scalable models as a hierarchical set of questions. Today state-of-the-art decision tree techniques heavily use randomisation and ensemble methods to improve accuracy. Instead of having only one decision tree, an ensemble is constructed by introducing randomisation at the learning stage.

One of their important drawbacks is however the complexity of randomized tree models, *i.e.* the large number and the size of trees, to achieve good performance particularly in *high dimensional* problems. Even if the price of memory is decreasing, we always want to handle bigger problems. Furthermore, a new trend is to deploy supervised learning models or algorithms into mobile and embedded applications with tight hardware constraint, *e.g.* object (face, text, optical character, etc.) recognition in smart phone; interaction, navigation, etc of autonomous robots.

1.2 Objectives of the master thesis

The goal of this master thesis is to propose some pre- and/or post- processing methods in order to obtain compact and highly expressive models from ensembles of randomized trees. A selected approach will be studied experimentally.

The main idea of the developed method is to apply a feature selection method on features extracted from tree ensembles.

Empirical experiments are carried out on 3 datasets and they show that the combination of both the feature selection technique and ensemble of randomized trees leads to a drastic reduction of the number of features and can improve the accuracy with respect to regular ensembles.

1.3 Strategy and implementation

The work undertaken in this master thesis began with bibliographical researches about model reduction in high dimensional spaces. In parallel, a proof of concept was carried out by combining *extremely randomized trees* (see [Geurts et al., 2006a] or Section 2.3.3) and *glmnet* (see [Friedman, 2008]), a LASSO method (see Section 2.4.2) which is a feature selection method (see Section 2.5). Early experiments were carried out using existing libraries in a Matlab environment: a regression tree package [Geurts, 2011] and a glmnet package [Jiang, 2011].

Later on, it was decided to study in more details the combination of *extremely randomized trees* and *monotone LASSO* (see Section 2.4.3). At the same period, we moved to our own C++ implementation of every supervised learning algorithm using only Boost C++ libraries[1].

When the first implementation became stable, we deployed it on Nic3, the supercomputer of the University of Liège. We worked on its optimisation and some parallelisms were added with OpenMP (see [Chapman et al., 2007]). First steps were also made to use sparse matrix storage. Finally, we developed automatic procedures to perform our experiments.

1.4 Organisation of the document

We describe in this document the results of our research. We first begin by introducing in Section 2 core material about supervised learning and methods used in this master thesis. We present the developed methods to reduce the complexity of a randomized forest in Section 3 and their experimental evaluation in Section 4. Finally, we present our conclusions and the future perspectives at the end of this work in Section 5.

[1] Boost C++ is a set of general purpose free peer-reviewed portable C++ source libraries.[Boost, 2011]

Chapter 2

Supervised learning

The supervised learning framework is first introduced in Section 2.1. Model assessment and selection methods are detailed in Section 2.2. Next, we present decision trees (see Section 2.3) and regularisation with $L1$-norm (see Section 2.4), two approaches to solve supervised learning tasks. A short presentation of feature selection methods is made in Section 2.5.

2.1 Framework

Supervised learning algorithms try to find a function f that approximates at best the output attributes given the input attributes. Samples, also called objects, are provided to the learning algorithm during the learning task.

A set of samples forms a database or a dataset such as the one presented in Table 2.1. In the example, each row represents a sample with input attributes x_j and output attributes y_i. Attributes, also called variables or features, are of different nature *e.g.*: *symbolic e.g* "Yes", "No"; "$class_1$", "$class_2$", ... or *numerical e.g.* temperature, currency,...

A supervised learning problem is called a *classification* problem whenever the output is symbolic and a *regression* problem whenever the output is numerical.

x_1	x_2	x_3	x_4	...	y_1	y_2	...
0.51	1000	Yes	S	...	1	$class_1$	...
10.73	2000	No	M	...	3.5	$class_2$	...
0	-500	Yes	XXL	...	-5.2	$class_3$	...
...	...	...	...	...	...	...	...

Table 2.1 – A database example

The supervised learning task can be set as an optimisation problem. From a set of n learning samples $\{x_i, y_i\}_{i=1}^n$ with $x_i \in \mathscr{X}$ and $y_i \in \mathscr{Y}$, we will search for a function $f : \mathscr{X} \to \mathscr{Y}$ in a hypothesis space $\mathscr{H} \subset \mathscr{Y}^{\mathscr{X}}$ that minimizes the expectation of some loss function $l : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$ over the joint distribution of the input/output pairs:

$$\min_{f \in \mathscr{H}} E_{\mathscr{X},\mathscr{Y}}\{l(f(x),y)\}. \tag{2.1}$$

The minimal attainable error for a function $f \in \mathscr{Y}^{\mathscr{X}}$, called *residual error*, is obtained by the *Bayes model* f_{Bayes} the best function in $\mathscr{Y}^{\mathscr{X}}$ for a given loss function.

Common loss function In the present work, we will use a *quadratic error loss* in regression to estimate the error made on a sample $\{x,y\}$ with a learnt function $\hat{f}$:

$$l(\hat{f}(x),y) = (\hat{f}(x)-y)^2. \tag{2.2}$$

And when we deal with a classification problem, we will use a *0-1 loss* to quantify the error made on a sample $\{x,y\}$ with a classifier $\hat{f}$:

$$l(\hat{f}(x),y) = I(\hat{f}(x) \neq y) = \begin{cases} 1 & \text{if } \hat{f}(x) \neq y; \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

where $I(\bullet)$ is the indicator function.

2.2 Model assessment and selection

Model assessment

When a model $\hat{f}$ has been learnt, we want to assess the quality of this model. So a first idea would be to estimate the error on the learning samples *LS* using a loss function l:

$$Err_{LS} = \sum_{(x,y)\in LS} l(\hat{f}(x),y). \tag{2.4}$$

The quantified error Err_{LS} is called the *re-substitution error* and it is a poor estimator of the quality of a model. As illustrated on the bottom graph of Figure 2.1, the model $\hat{f}(x)$ may be unable to capture the relationship between the input and the output despite a re-substitution error equal to zero. The bottom of Figure 2.1 is indeed an example of *over-fitting*: the method has so many degrees of freedom that it fits the noise, and is thus unable to approximate well the Bayes model. Conversely, the top graph of Figure 2.1 shows a model that *under fits* the data: the model, here a linear function, is not complex enough to fit correctly the underlying function.

A better approach to assess the quality of a model is to estimate its *generalisation error* Err. The generalisation error is the error made on new samples drawn from the same distribution:

$$Err = E_{\mathscr{X},\mathscr{Y}}\{l(\hat{f}(x),y)\}. \tag{2.5}$$

In practice if there are enough samples, we can split the database into two sets: a *learning set* to build a model and a *testing set* to assess its quality. So the generalisation error is estimated on the test set TS:

$$Err_{TS} = \sum_{(x,y)\in TS} l(\hat{f}(x),y). \tag{2.6}$$

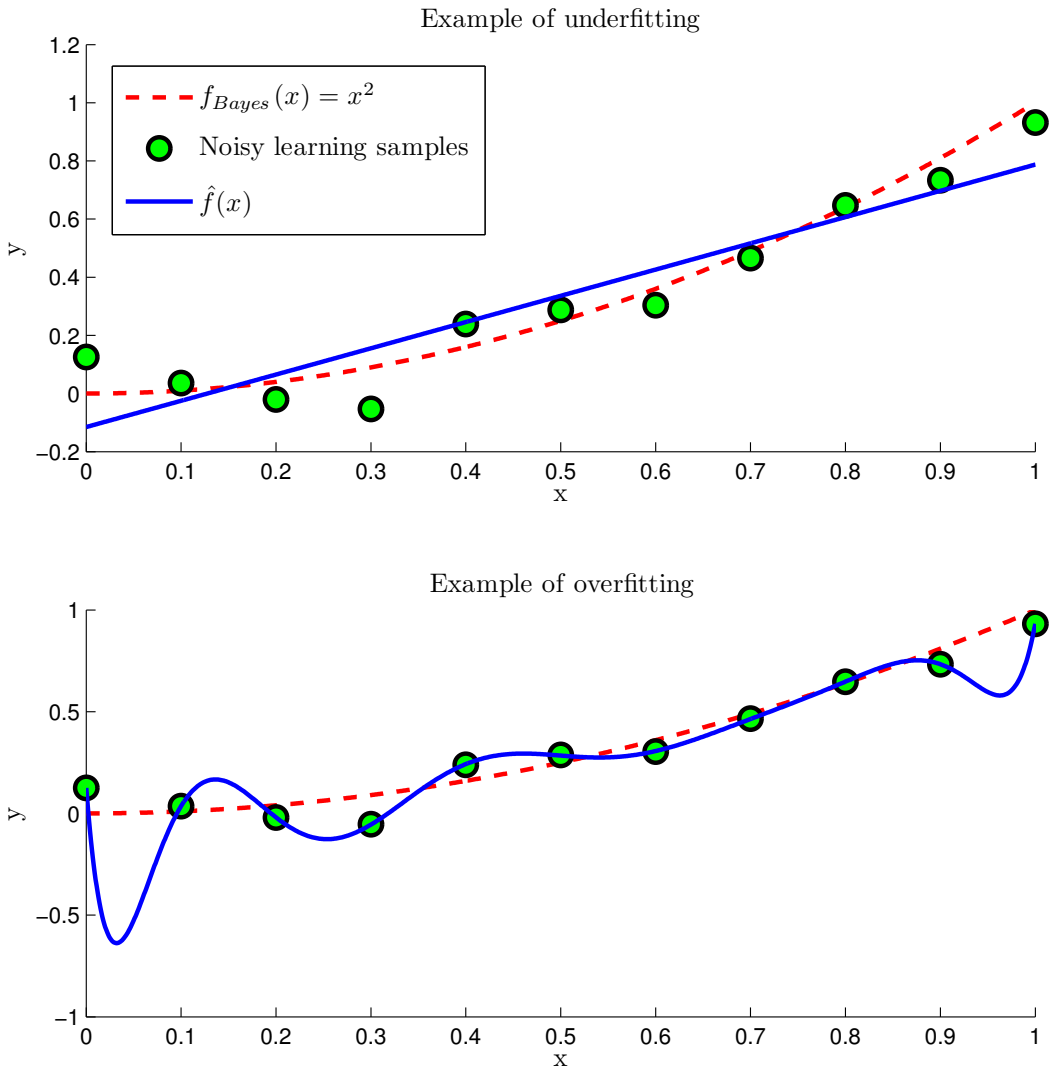

Figure 2.1 – An example of underfitting on *top* and an example of overfitting on *bottom*.

More advanced techniques exist such as cross-validation[1] or bootstrap (see [Kohavi, 1995]).

Error decomposition: bias/variance trade-off

The generalisation error Err of a supervised learning algorithm is a random variable depending on the randomly drawn learning set LS to build the model $\hat{f}$. The generalisation error can be averaged over randomly drawn learning sets LS of a given size n:

$$E_{LS}\{Err\} = E_{LS}\{E_{\mathscr{X},\mathscr{Y}}\{l(\hat{f}(x),y)\}\} \tag{2.7}$$
$$= E_{LS}\{E_{\mathscr{X}}\{E_{\mathscr{Y}|\mathscr{X}}\{l(\hat{f}(x),y)\}\}\} \tag{2.8}$$
$$= E_{\mathscr{X}}\{E_{LS}\{E_{\mathscr{Y}|\mathscr{X}}\{l(\hat{f}(x),y)\}\}\}. \tag{2.9}$$

Moreover, a model $\hat{f}$ can be built on every learning set LS and thus the *average model* $f_{avg}(x)$ is given by:

$$f_{avg}(x) = E_{LS}\{\hat{f}(x)\}. \tag{2.10}$$

In regression with a quadratic loss, the average generalisation error becomes:

$$E_{LS}\{Err\} = E_{\mathscr{X}}\{E_{LS}\{E_{\mathscr{Y}|\mathscr{X}}\{(\hat{f}(x)-y)^2\}\}\}. \tag{2.11}$$

And it can be decomposed into three terms (see Appendix A for the derivation of this decomposition):

$$E_{LS}\{Err\} = E_{\mathscr{X}}\{var_{LS}(\hat{f}(x)) + bias^2(x) + var_{\mathscr{Y}|\mathscr{X}}(y)\} \tag{2.12}$$
$$= E_{\mathscr{X}}\{var_{LS}(\hat{f}(x))\} + E_{\mathscr{X}}\{bias^2(x)\} + E_{\mathscr{X}}\{var_{\mathscr{Y}|\mathscr{X}}(y)\}. \tag{2.13}$$

The interpretation of the terms of the equation 2.13 are:

- The *variance of a supervised learning algorithm* $E_{\mathscr{X}}\{var_{LS}(\hat{f}(x))\}$ describes the variability of the model with randomly drawn learning sets LS:

  $$E_{\mathscr{X}}\{var_{LS}(\hat{f}(x))\} = E_{\mathscr{X}}\{E_{LS}\{(\hat{f}(x) - f_{avg}(x))^2\}\}. \tag{2.14}$$

  Algorithm producing too complex models have often a high variance as in the overfitting example of Figure 2.1.

- The *squared bias of a supervised learning algorithm* $bias^2(x)$ is the distance between the average learnt model f_{avg} and the Bayes model f_{Bayes}:

  $$E_{\mathscr{X}}\{bias^2(x)\} = E_{\mathscr{X}}\{(f_{avg}(x) - f_{Bayes}(x))^2\}. \tag{2.15}$$

  A biased model has too few parameters to fit correctly the data as in the underfitting example of Figure 2.1.

[1] Cross-validation is an error estimation method. In order to assess the quality of the model, the sample set is first divided randomly in K folds of equal size. Common values of K are 5 or 10. Then a model is built on $K-1$ folds and assessed on the last one. The resulting quality measure is obtained by averaging the estimated risk over the K folds.

- $E_{\mathscr{X}}\{var_{\mathscr{Y}|\mathscr{X}}(y)\}$ is the residual error which depends on the supervised learning problem.

In classification, similar concepts exist but the decomposition is not trivial. More information about bias/variance decomposition can be found in [Geurts, 2010].

Model selection

Supervised learning algorithms have usually several parameters, *e.g.* maximal degree of a polynomial to fit some data or the number of trees in an ensemble. And thus at the learning stage, a value of these parameters must be chosen in the parameter space.

Generally several models are built with different set of parameters and the best one is chosen. One approach to perform the *model selection and assessment* is to divide the database into three sets: a *learning set* in order to learn the models with the different set of parameters, a *validation set* to choose the best parameter and a *testing set* to assess properly the quality of the chosen parameter combination.

2.3 Randomized ensembles of trees

We first introduce *decision trees* theory (see Section 2.3.1) to help the reader understands more advanced material. We continue by presenting ensembles of decision trees (see Section 2.3.2). Then we develop *extremely randomized trees* (see Section 2.3.3) and *totally randomized trees* (see Section 2.3.4).

2.3.1 Decision trees

A *decision tree* is a hierarchical set of questions where every internal node tests an attribute and each leaf corresponds to an output value. Decision trees are widely used, because they are highly interpretable and versatile. Figure 2.2 presents a decision tree to interpret aerial photographs.

An optimal decision tree minimizes the expected number of tests to make a prediction. The search for an optimal decision tree is NP-complete[2] (see [Hyafil and Rivest, 1976]). So in practice, a greedy approach is taken instead to produce efficiently small, consistent and coherent trees with respect to learning samples.

First, we have to point out that each internal node of a decision tree partitions the input space into smaller sub-domains. A tree structure can be grown by splitting recursively a learning sample set until a particular *stopping criterion* is met (see algorithm 2.1). The predicted output at each leaf is obtained by averaging over the learning samples that reach the leaf in regression or a majority vote in classification.

"How to select the best splitting criterion at each internal node?". Let us first consider the *classification case*. We have to remember that we want small trees. A complexity constraint

[2]A NP-complete problem is NP (nondeterministic polynomial time) and NP-hard (any NP problem can be transformed into this problem) (see definition B.8 and theorem B.2 of [Bertsimas and Weismantel, 2005]).

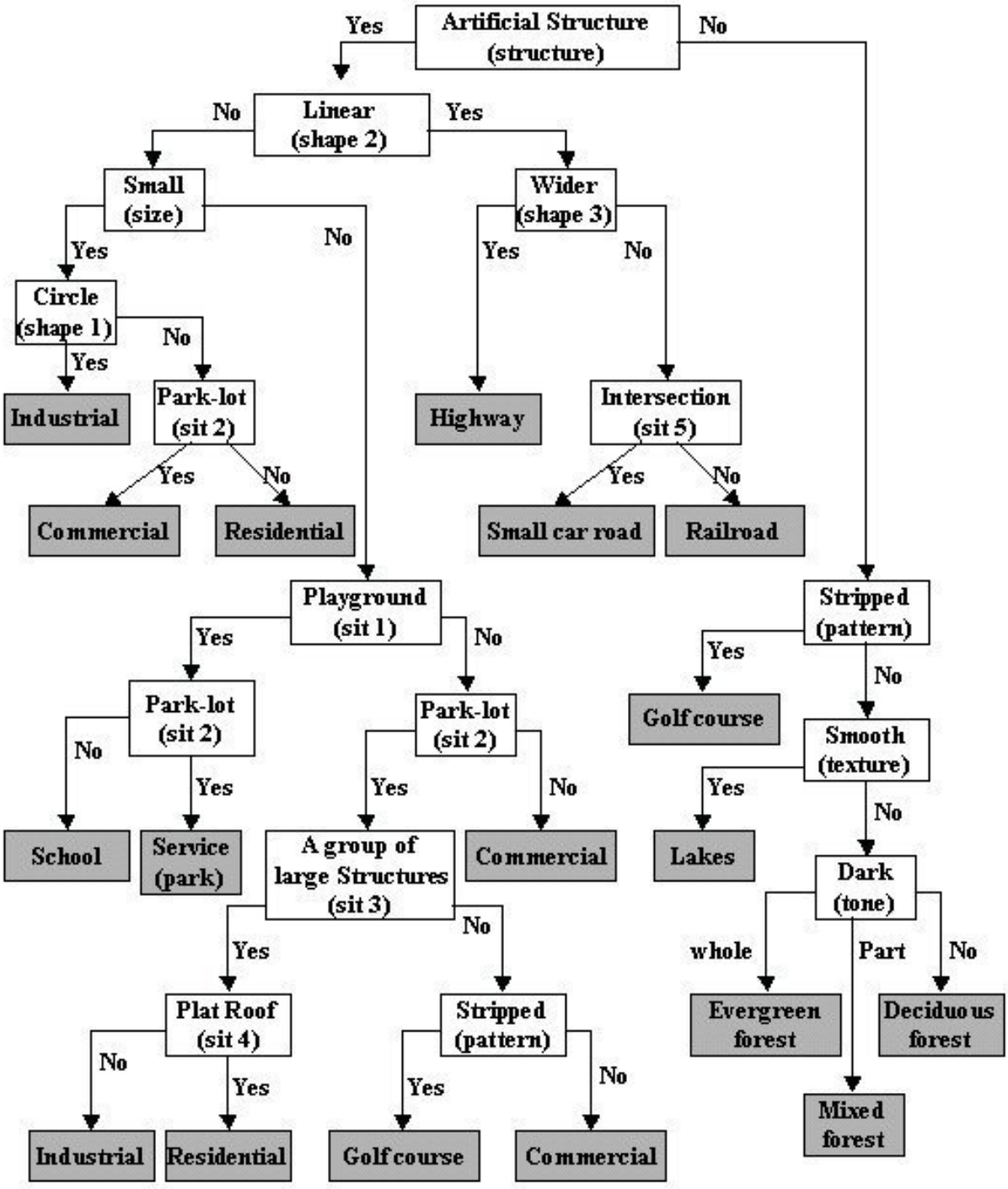

Figure 2.2 – A decision tree for interpretation of aerial photographs. This Figure is taken from [Choi, 2002], `http://www.ucgis.org/summer2002/choi/choi.htm`.

Algorithm 2.1: Top-down decision tree growing

Input: *LS*, learning samples
Output: A decision tree

```
learnDecisionTree(LS)
begin
    if stoppingCriterion(LS) then
        create a leaf ;
    else
        Select the "best" split;
        Create an internal node;
        Split LS according to the best split;
        Call learnDecisionTree on each of the resulting subsets;
end
```

can be expressed by maximising the class separation at each node. In other words, we want to minimize the impurity at each split.

Many measures of impurity $I(\bullet)$ exist (for an empirical comparison of splitting criterion see [Mingers, 1989a]). Some of these are entropy, Gini index or misclassification error rate presented in Table 2.2.

Impurity measure	
Entropy	$I(LS) = -\sum_j p_j \log p_j$
Gini index	$I(LS) = \sum_j p_j (1 - p_j)$
Misclassification error rate	$I(LS) = 1 - \max_j p_j$

Table 2.2 – Impurity measure in classification
p_j denotes the probability associated to the class j within the learning samples *LS*.

The best split will be the one that minimizes the expected reduction of impurity:

$$\Delta I(LS) = I(LS) - \sum_j \frac{|LS_j|}{|LS|} I(LS_j), \tag{2.16}$$

where LS_j with $j = \{1, \ldots\}$ are the disjoint subsets of the learning samples *LS* obtained by splitting on an attribute x_j.

In the *regression case*, we can still apply the previous ideas with an appropriate measure of impurity for every possible split of the output. A possible score measure is the variance reduction

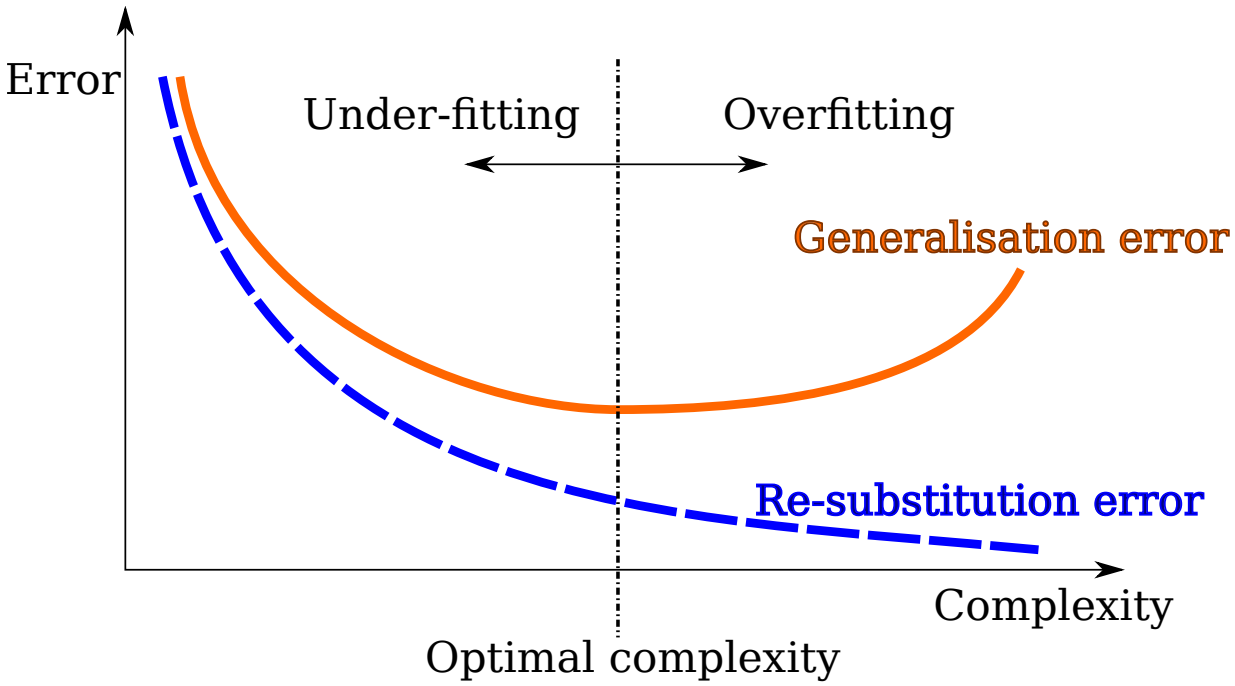

Figure 2.3 – The tradeoff between complexity and generalisation performance. This graph is a modified version of slide 36 [Geurts, 2009]

.

given a split LS_l, LS_r of LS:

$$score(LS, LS_l, LS_r) = \frac{var\{y|LS\} - \frac{|LS_l|}{|LS|} var\{y|LS_l\} - \frac{|LS_r|}{|LS|} var\{y|LS_r\}}{var\{y|LS\}}, \tag{2.17}$$

where $var\{y|LS\} = E\{(y - E\{y|LS\})^2|LS\}$.

One question still remains: "Which stopping criterion should we use?" A possible answer is to stop splitting a node as soon as the outputs of all samples reaching this node are similar or the number of samples in this node is below some predefined threshold. The graph of Figure 2.3 illustrates the trade-off between complexity, re-substitution error and generalisation error. In practice, the presented approach could lead to fully developed trees which have a poor generalisation error. The model overfits the data and has too many degrees of freedoms.

The performance in generalisation can be improved using pre- or post-pruning. *Pre-pruning* modifies the stopping criterion of the learning step, *e.g.* add the condition: stop to grow the tree if the number of samples is below a threshold. *Post-pruning* is a post processing step to decide a posteriori what is the best complexity. For a more detailed analysis of different pruning techniques, we refer the reader to [Esposito et al., 1997] and [Mingers, 1989b].

We have developed the core material about decision trees. Some extensions or generalisations have been left out, the interested reader can pursue with [Hastie et al., 2003] section 9.2.

2.3.2 Ensembles of decision trees

In the past two decades, research has been undertaken to improve decision trees with ensemble methods. The combination of ensemble techniques and decision trees allows to compete with other state of the art methods such as support vector machines (see [Schölkopf and Smola, 2002]).

Ensemble methods combine the prediction of several models to improve performance upon a single one (see [Dietterich, 2000] for a review of different ensemble methods). The idea is to exploit existing supervised learning algorithms and to alter their bias/variance trade-off. These techniques indeed try to shrink the variance (*resp.* bias) to zero while attempting to keep unchanged or to reduce the bias (*resp.* variance). Several ensemble paradigms exist such as boosting or perturb and combine methods:

- *Boosting methods* (see [Schapire, 2003]) learn sequentially a series of models using a weak learner[3]. At each step, a model is built on the weighted learning samples and added to the ensemble. The weight associated to each sample is updated and determined by the difficulty to fit this sample. The purpose is to reduce the bias by increasing sequentially the complexity of the model. A widely used instance of boosting is *Adaboost* (see [Freund and Schapire, 1995]).

- *Perturb and combine methods* first perturb the learning algorithm or the data to produce different models and the predictions are later combined. For a more in depth study of the perturb and combine paradigm applied to decision trees, see part 2 of [Geurts, 2002].

In the context of ensembles of decision trees, a successful approach is to randomize the learning procedure, *e.g. bagging* (see [Breiman, 1996a]) builds models on bootstrap[4] of the learning samples or *random forests* (see [Breiman, 2001]) which combine bagging and the best split selection in a random subset of attributes.

In the following Section, we will describe *extremely randomized trees* (see Section 2.3.3) and *totally randomized trees* (see Section 2.3.4) that have been used in the present master thesis. These are representative of perturb and combine techniques.

2.3.3 Extremely randomized trees

Extremely randomized trees (see [Geurts et al., 2006a]) build decision trees with randomization of attribute and cut point selection (see Algorithms 2.2, 2.3 and 2.4). *Extremely randomized trees* are also called *Extra Trees* and we will use both terms interchangeably. In the following paragraphs, we will describe the effect of the tuning parameters and our motivation to use this particular algorithm.

[3] A *weak learner* is a supervised learning algorithm that performs a little better than random guessing.

[4] *Bootstrap* is a re-sampling technique; a *bootstrap* is a copy of the learning samples obtained by drawing samples from the learning samples with replacement.

Algorithm 2.2: Learn an ensemble of extremely randomized trees

Input:

- LS, learning samples
- M, the number of terms in the ensemble
- K, the number of random splits drawn at each internal node
- n_{min}, the minimum number of reaching samples in a leaf

Output: An ensemble of extremely randomized trees

`extremelyRandomizedTrees` (LS, M, K, n_{min})
begin
  for $i \leftarrow 1$ **to** M **do**
    $t_i =$ `extraTree` (LS, K, n_{min}) ;
  return $\{t_1, t_2, \ldots, t_M\}$
end

Algorithm 2.3: Learn an extra tree

Input:

- LS, learning samples
- K, the number of attributes drawn at each internal node
- n_{min}, the minimum number of reaching sample in a leaf

Output: An extra tree

`extraTree` (LS, K, n_{min})
begin
  if $|LS| < n_{min}$ *OR all attributes are constant OR the output is constant* **then**
    Create a leaf;
  else
    Draw randomly $\{a_1, \ldots, a_K\}$ among non constant attributes and without replacement;
    Pick K splits $\{s_1, \ldots, s_K\}$ using `pickRandomSplit` (LS, a_i) ;
    Select $\{s*, a*\} = \max_{i=1,\ldots,K} score(s_i)$;
    Split LS given $\{s*, a*\}$ into two subsets LS_{left} and LS_{right};
    Grow left sub-tree $t_l =$ `extraTree` (LS_{left}, K, n_{min}) ;
    Grow right sub-tree $t_r =$ `extraTree` (LS_{right}, K, n_{min}) ;
end

Algorithm 2.4: Pick a random split given learning samples and an attribute

Input:

- *LS*, learning samples
- a, an input attribute

Output: A split of *LS* given the attribute a

```
pickRandomSplit(LS,a)
begin
    if a is a numerical attribute then
        Compute the minimal a_min and maximal a_max value of a in LS;
        Draw a random cut point a_c in [a_min, a_max] in LS;
        return the split [a < a_c];
    if a is a categorical attribute then
        Compute A_S the subset of A of values of a that appear in S;
        Randomly draw a proper non empty subset A_1 of A_S and a subset A_2 of A\A_S ;
        return the split [a ∈ A_1 ∪ A_2];
end
```

Effect of K. The parameter K determines the number of randomly selected attributes at each node, and controls the randomisation of the tree structure learning. Following the conjecture in [Geurts et al., 2006a], K allows to filter out irrelevant features and facilitates the selection of relevant ones. In fact, a suitable value of K reduces bias without increasing too much variance.

Effect of n_{min}. The parameter n_{min} controls the minimum number of samples in order to split a node. In fact, n_{min} manages the complexity of an extra tree and is an instance of a pre-pruning technique. A small value of n_{min} will allow an increase in the complexity of the model, hence a smaller bias and a greater variance, and *vice versa*.

Furthermore, the parameter n_{min} adds an averaging effect and an appropriate value will indeed improve performance in presence of noise.

Effect of M. The parameter M determines the size of the ensemble: prediction accuracy increases on average with higher ensemble size. A suitable M for a given problem depends on the rate of convergence and on the computational resources.

Motivation. We have decided to employ extremely randomized trees to implement our ideas for the following reasons:

1. The size of the learnt models is consequent. We will see if we are able with our methods to reduce consistently the complexity of the model.
2. The algorithm is fast and often accurate.

3. The various parameters allow to control easily the bias/variance trade-off.

2.3.4 Totally randomized trees

Totally randomized trees are decision trees built with a random split at each node. They are obtained through *extremely randomized trees* with only one randomly selected attribute at each internal node (*i.e.* $K = 1$).

They have two parameters n_{min}, the minimum number of objects in order to split, and M, the size of the forest. The use and interpretation of these parameters is similar to those presented in Section 2.3.3.

Totally randomized trees are particularly interesting in our application because the tree structure learning is not guided at all by the output values.

2.4 Regularisation with $L1$-norm

We first introduce the *regularisation framework* in the linear case in Section 2.4.1. *LASSO*, a $L1$-norm regularisation method, is then presented in Section 2.4.2. We continue in Section 2.4.3 with a more constraint version of the LASSO called *monotone LASSO*.

2.4.1 Linear regularisation framework

Given a set of n samples $\{(\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,p}), y_i)\}_{i=1}^{n}$, the goal of linear regression methods is to search for the parameters $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$ of the linear function to approximate at best the output y given an input $\boldsymbol{x}$

$$f(\boldsymbol{x}, \boldsymbol{\beta}) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j. \tag{2.18}$$

The learning problem can be stated as the minimisation of the risk R, the expectation of some loss function l over the joint distribution of input/output pairs

$$\min_{\boldsymbol{\beta}} R(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta}} E_{\boldsymbol{x},y}\{l(y, f(\boldsymbol{x}, \boldsymbol{\beta}))\}. \tag{2.19}$$

Due to the finite number of samples, the risk function is estimated with the learning samples

$$\hat{R}(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} l(f(\boldsymbol{x}_i, \boldsymbol{\beta}), y_i). \tag{2.20}$$

Thus an estimation of the optimal parameters is obtained by solving the following optimisation problem

$$\hat{\boldsymbol{\beta}} = \min_{\boldsymbol{\beta}} \hat{R}(\boldsymbol{\beta}). \tag{2.21}$$

With a quadratic loss, this would lead to the well known least square estimation. A complete treatment of least square regression can be found in [Hastie et al., 2003] Section 3.2.

However in the presence of over-fitting or of an ill-conditioned problem, *e.g.* when $p > n$, we need to add a criterion to find a unique solution. One approach, called regularisation, is to add a constraint on the complexity of the model or to add a term in the optimisation criterion that penalises the complexity of model, by using a complexity penalty function $P(\boldsymbol{\beta})$, namely

$$\min_{\boldsymbol{\beta}\, s.t.\, P(\boldsymbol{\beta}) \leq t} R(\boldsymbol{\beta}), \tag{2.22}$$

or

$$\min_{\boldsymbol{\beta}} \left(R(\boldsymbol{\beta}) + \lambda P(\boldsymbol{\beta}) \right), \tag{2.23}$$

with $\lambda, t \geq 0$. These two formulations are essentially equivalent, in the sense that solutions of eq. (2.22) found with smaller values of t will correspond to solutions found for eq. (2.23) with larger values of λ.

The tuning parameters, t or λ, manage the complexity of the model: the greater λ (*resp.* the smaller t), the more the coefficients are shrunk. The choice of the penalty function greatly influences the properties of the solution path which is found by solving the optimisation problem for different values of the parameter t or λ.

2.4.2 LASSO - Least Absolute Shrinkage and Selection Operator

Among all possible sets of penalty functions, one is the *power family*:

$$P(\boldsymbol{\beta}) = \sum_{j=1}^{p} ||\beta_j||^{\gamma} \text{ s.t. } \gamma \geq 0. \tag{2.24}$$

Different values of γ lead to commonly proposed penalty terms *e.g.*: $\gamma = 1$ leads to LASSO ([Tibshirani, 1996]) and $\gamma = 2$ to ridge regression ([Hoerl and Kennard, 1970]).

The LASSO penalty corresponds to the $L1$-norm of the weights associated to a feature. $L1$-norm regularisation has the property to lead to sparse models ([Tibshirani, 1996]), *i.e.* most coefficients β_j have zero weights. In this master thesis, this property is particularly searched.

Regularisation with a LASSO penalty and a quadratic error loss is a convex problem. Efficient algorithms compute the solution path *i.e.* the solution of the regularisation problem is computed for a large number λ or t. Some of these procedures are *least angle regression* (see [Efron et al., 2004]), *generalised path seeking* (see [Friedman, 2008]) or *pathwise coordinate optimisation* (see [Friedman et al., 2007]) such as *glmnet* (see [Friedman et al., 2009]).

These efficient algorithms have generally the same order of computation of a least square fitting (see Section 7 of [Efron et al., 2004] for a comparison of computational complexity between least angle regression and least square fitting; see Section 5 of [Friedman, 2008], Section 5 of [Friedman et al., 2009] and Section 5 of [Friedman et al., 2007] for some speed comparisons).

2.4.3 Monotone LASSO and incremental forward stagewise regression

Monotone LASSO is a more constrained version of the LASSO. The difference is that the first one restricts the evolution of the coefficients in the solution path to be monotonically increasing or decreasing.

Incremental forward stagewise regression also denoted FS_ε computes an approximation of the monotone LASSO solution path and is described in Algorithm 2.5. The tuning parameter ε controls the step size between each point of the path; the monotone LASSO path is obtained when $\varepsilon \to 0$.

Algorithm 2.5: Incremental forward stagewise regression

Input:

- LS, the learning samples
- ε, the step size

Output: A linear regression solution path

`incrementalForwardStagewiseRegression(`*LS,*ε`)`
begin
Normalise the attributes to have zero mean and unit variance;
Start with a first point $(\beta_0, \ldots, \beta_p) = (0, \ldots, 0)$ in the solution path;
Set the residuals r to $r \leftarrow 0$;
repeat
Find the most correlated attribute x_j with the residual r;
Set the increment δ to

$$\delta \leftarrow \varepsilon \, sign(corr(x_j, r)); \tag{2.25}$$

Add the previous point in the solution path updated with

$$\beta_j \leftarrow \beta_j + \delta; \tag{2.26}$$

Update the residuals r with

$$r \leftarrow r - \delta \, x_j; \tag{2.27}$$

until *the residuals are not correlated with the attributes* ;
end

According to the results obtained in [Hastie et al., 2007], the solution path obtained with the monotone LASSO is smoother and takes longer to over-fit in the presence of correlated variables.

For a better understanding, Figure 2.4 from [Hastie et al., 2007] depicts the coefficient path for LASSO and incremental forward stagewise regression for a simulated database of 50 groups of 20 variables which are strongly correlated ($\rho = 0.95$). The goal is to predict the output generated from the sum of one variable taken from each group with a weight obtained from a standard Gaussian and a Gaussian noise. Due to the high correlation between predictors, the coefficient

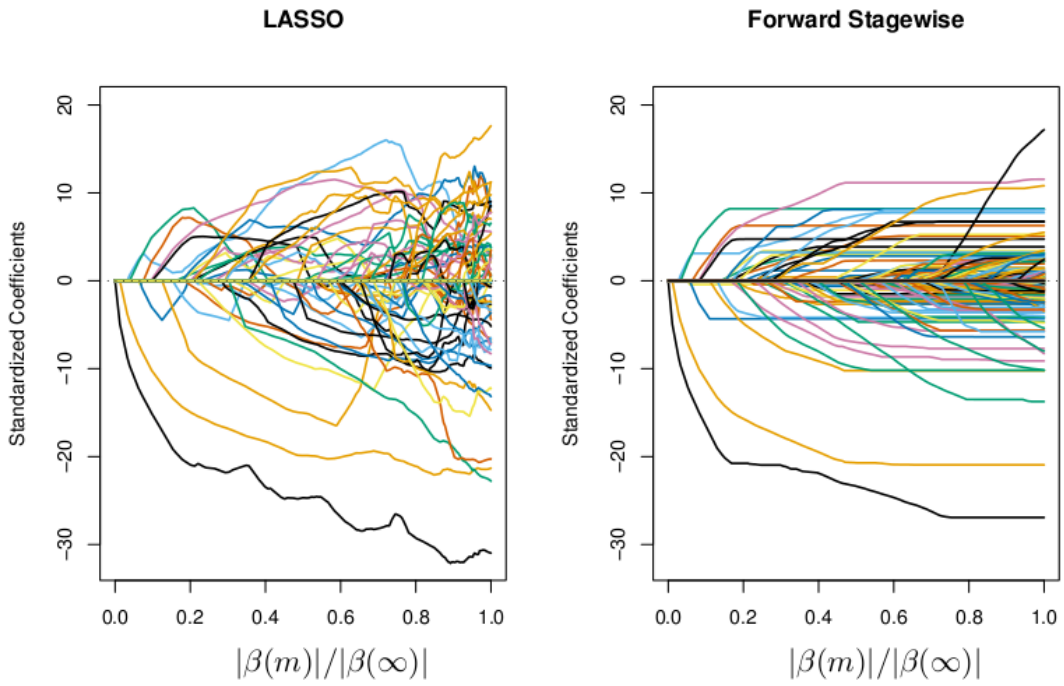

Figure 2.4 – Coefficient path for LASSO and monotone LASSO on an artificial database with 1000 attributes and 60 samples. This figure is taken from [Hastie et al., 2007].

path for the LASSO becomes erratic: small changes in the data may lead to erratic changes of the coefficient estimate.

2.5 Feature selection methods

Feature selection methods allow to select the most relevant subset of variables. These methods can be classified into several groups:

- *Filter methods* compute a relevance score such as correlation with the output to rank variables and select the most interesting ones.
- *Wrapper methods* try to find a subset of features that maximises the quality of the model with an output variable.
- *Embedded methods* are machine learning algorithms where the feature selection procedure is embedded in the learning phase, *e.g.* decision trees and regularisation in $L1$-norm are among these methods.

Feature selection techniques are particularly useful whenever there are many redundant and/or noisy[5] variables.

The interested reader can pursue with [Guyon, 2006] for a review of feature selection methods or [Guyon and Elisseeff, 2003] for a shorter introduction.

[5]Noisy variables are irrelevant variables that are not involved in the relationship between the input and the output.

Chapter 3

Compressing ensembles of randomized trees

3.1 Introduction and problem statement

The goal of this master thesis is to reduce the complexity of ensembles of randomized trees while trying to keep intact the original algorithm. In this perspective, we can either modify the data provided to the learning algorithm or either modify the learnt model.

An ensemble of randomized trees has a complexity linearly proportional to the size of the ensemble and in the worst case to the number of samples (see Appendix B). The randomisation of the learning stage brings redundancy in the ensemble and at the same time robustness. We will consider methods that are able to decrease the complexity while conserving the robustness and the accuracy of the model.

We would like to highlight the fact that post-processing methods applied on ensembles of decision trees have already been successfully developed to improve interpretability of decision trees [Friedman and Popescu, 2008, Meinshausen, 2009], accuracy [Pisetta et al., 2010]. But no research has been especially undertaken to our knowledge to find compact models of ensemble of randomized trees.

The identified research directions are presented in Section 3.2. And the developed ideas are presented in Sections 3.3 and 3.4.

3.2 Identified research directions

At the first stage of the research process, we have emitted the hypothesis that a sparse representation of the model induced by an ensemble of randomized trees exists.

Under ths perspective, the most interesting research directions identified during this master thesis are presented below :

- *Feature selection techniques applied on node characteristic functions*: From a tree, one can extract a set of binary features, each one associated to a leaf or a node of the tree and being

true for a given object only if it reaches the corresponding leaf or node when propagated in the tree. Given this representation, the prediction of an ensemble can be simply retrieved by linearly combining these characteristic features for all trees with appropriate weights.

A linear feature selection method, such as the LASSO, can be applied on these features; a subtree in the ensemble of trees will then be pruned as soon as the characteristic features corresponding to its constituting nodes are not selected in the linear model.

- *Exploiting the kernel view of tree-based ensembles with SVM*: An ensemble of trees can be used to define a kernel(*e.g.* [Geurts et al., 2006b]) between objects that measures the number of times these two objects reach the same leaf in a tree and is weighted by the specificity of the reached leaves. Using this kernel within support vector machines will help identifying the objects from the training sample that are really useful (the so-called support vectors). The ensemble of trees could then be pruned by keeping only paths traversed by these objects.

- *Random pruning*: Each tree in an ensemble is typically obtained by introducing some randomization at the learning stage. The robustness of an ensemble comes from the complementarity but also from the redundancy between the trees in the ensemble. Because of this redundancy, it is likely that the predictive accuracy of the ensemble would be robust to some extent to the random removal of some nodes or paths in the trees.

With a perturb and combine ensemble techniques such as *bagging*, *random forest* and *extremely randomized trees*, the variance is shrunk while trying to keep the bias unchanged. The variability of the method with the learning samples is reduced by allowing the original algorithm to express several times the same pattern *differently* through randomisation.

Common characteristics in the ensemble are then highlighted and relationships in the data are revealed. Within an ensemble of randomized trees, several nodes or subsets of node of each tree express the same pattern. So with appropriate techniques, one may try to discover a subset of representative nodes within the ensemble either on the generated feature space or the kernel view of trees. With "*feature selection techniques applied on node characteristic functions*" and "*exploiting the kernel view of tree-based ensembles with SVM*", there are high hopes to find a sparse solution, *i.e.* only a few features or support vectors would be selected, because of redundancy. However, some concerns have to be raised:

- the sparse representation of the randomized forest might not reside in the node feature space and a basis transformation would be needed;

- by selecting only a few node characteristic functions or support vectors, we may try to fit too strongly the learning samples or conversely we might be too selective and lose the representative power of the original model.

Random pruning is interesting from a theoretical point of view, because it would bring more insight on randomized trees. Furthermore, a straight application is possible *e.g.* random pre-pruning.

Finally, we have chosen to develop the feature selection technique applied on node characteristic functions because of an early proof of concept. Furthermore, this approach prunes directly the decision trees contrary to "*the kernel view of tree-based ensembles with SVM*".

Lastly, post-processing methods were favoured over pre-processing methods. Because, post-processing methods work on the tree structure and do not modify the provided data. The first reason is that decision trees are already a feature selection technique. And secondly, the versatility of this original method is kept; decision trees can indeed easily handle many types of data. Moreover nothing prevents us to use feature selection methods or projection techniques such as random projections[1] before learning decision trees.

3.3 Feature selection techniques applied on node characteristic functions

Feature selection techniques applied on node characteristic functions is the combination of ensembles of randomized trees and a feature selection technique in order to produce compact ensemble of decision trees.

From a tree, one can extract a set of *node characteristic functions*, each one associated to a leaf of the tree. Each function, denoted $I(x \in L_l)$, is a binary variable equal to 1 if x reaches the leaf L_l in the tree, 0 otherwise. Then the prediction of a decision tree with $|L|$ leaves $\{L_l\}_{l=1}^{|L|}$ is given by:

$$\hat{f}(x) = \sum_{l=1}^{|L|} w_l I(x \in L_l), \tag{3.1}$$

where w_l is the prediction in leaf L_l.

A node characteristic function can also be associated to internal node with weight w_l equal to zero. The prediction of a decision tree with $|N|$ nodes $\{N_l\}_{l=1}^{|N|}$ becomes:

$$\hat{f}(x) = \sum_{l=1}^{|N|} w_l I(x \in N_l). \tag{3.2}$$

The model $\hat{f}(x)$ associated to an ensemble of M decision trees can then be expressed as the sum of node characteristic functions:

$$\hat{f}(x) = \frac{1}{M} \sum_{m=1}^{M} \sum_{l=1}^{|N_m|} w_{ml} I(x \in N_{ml}). \tag{3.3}$$

[1]Random projection is a dimensionality reduction technique. High dimensional data are projected on a random subspace of lower dimension by multiplying with a random matrix. If the lower dimensional space has enough dimensions, the structure and distance in the data are preserved. Furthermore, it speeds up computation. The following paper presents a short introduction on the topic [Blum, 2006] and an example of application on text and image can be found in [Bingham and Mannila, 2001].

The association of the node characteristic functions $I(\bullet)$ from a set of M decision trees form a space $\mathscr{Z}$ derived from the original input space $\mathscr{X}$. The expression of the vector $x \in \mathscr{X}$ in the space $\mathscr{Z}$ is given by:

$$z = \{I(x \in N_{1,1}), \ldots, I(x \in N_{1,|N_1|}), \ldots, I(x \in N_{M,1}), \ldots, I(x \in N_{M,|N_M|})\} \tag{3.4}$$

We would like to point out that the vector z has only *binary values with many zero elements*.

The dataset $\{x_i, y_i\}_{i=1}^n$ can be expressed in the space induced by an ensemble of decision trees by projecting $\mathscr{X}$ on $\mathscr{Z}$ to form a new dataset $\{z_i, y_i\}_{i=1}^n$. A feature selection algorithm can then be applied to prune the ensemble of decision trees. Indeed, any subtree can be pruned as soon as none of the characteristic functions of its constituting nodes, except its root, has been selected by the feature selection method.

Instead of using every node characteristic functions, one can select only nodes that satisfy a particular condition *e.g.* every node, every leaf, randomly chosen nodes, top ranked nodes according to a "score" measure, etc. The space $\mathscr{Z}$ would contain only the selected node characteristic functions.

Our general approach is presented at Algorithm 3.1 and has several degrees of freedom :

- choice of node characteristic functions;
- choice of a method to build an ensemble of randomized trees;
- choice of a feature selection method.

3.4 Regularisation in $L1$-norm of the space induced by an ensemble of decision trees

Regularisation with $L1$-norm, a class of supervised learning algorithms and an embedded selection method, is a particularly successful in high dimensional space. It is a linear method which is also probably enough given that the node characteristic functions are non-linear functions of the original features. Regularisation with $L1$-norm sets an optimisation problem on the node characteristic function space, assumed of size p, given n samples $\{(z_i = (z_{i,1}, \ldots, z_{,i,p}), y_i)\}_{i=1}^n$ and a filtering parameter α_j associated to each node characteristic function:

$$\min_{\{\beta_0,\ldots,\beta_p\}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \alpha_j \beta_j z_{ij} \right)^2 \tag{3.5}$$

$$s.t. \qquad \sum_{j=1}^{p} |\beta_j| \leq t. \tag{3.6}$$

The optimal solution will be sparse (see [Tibshirani, 1996]), only a few weights β_j will be non zero. The ensemble of decision trees can be pruned using the weight β_j with $j = 1, \ldots, p$: a

Algorithm 3.1: Feature selection techniques applied on node characteristic functions of an ensemble of randomized trees

Input:

- $\{x_i, y_i\}_{i=1}^n$, a set of n learning samples with $x_i \in \mathscr{X}$ and $y_i \in \mathscr{Y}$;
- M, the number of terms in the ensemble of randomized decision trees.

Output: An ensemble of decision trees

begin

1. Learn an ensemble of randomized trees $\{t_1, t_2, \ldots, t_M\}$ using $\{x_i, y_i\}_{i=1}^n$;
2. Extract selected node characteristic functions from $\{t_1, t_2, \ldots, t_M\}$ to form $\mathscr{Z}$, the node characteristic function space;
3. Project $\mathscr{X}$ on $\mathscr{Z}$ to form samples $\{z_i, y_i\}_{i=1}^n \in \{\mathscr{Z}, \mathscr{Y}\}$ with $\{x_i, y_i\}_{i=1}^n \in \{\mathscr{X}, \mathscr{Y}\}$;
4. Apply a feature selection technique using $\{z_i, y_i\}_{i=1}^n \in \{\mathscr{Z}, \mathscr{Y}\}$ and prune $\{t_1, t_2, \ldots, t_M\}$;

return $\{t_1, t_2, \ldots, t_M\}$;

end

node j can be deleted if its weight β_j and the weights of all its siblings and descendants are equal to zero.

The coefficients α_j are user-defined filtering parameters that allow to define on which nodes the optimisation problem is carried out:

- every node characteristic functions can be selected if:

$$\alpha_j = 1 \quad \forall j. \tag{3.7}$$

- only leaves can be considered as node characteristic functions:

$$\alpha_j = \begin{cases} 1 & \text{if } N_j \text{ is a leaf;} \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

The step 4 of Algorithm 3.1 is modified by a regularisation in $L1$-norm step in the space $\mathscr{Z}$ (see Algorithm 3.2) as a feature selection technique.

The optimisation problem denoted by equations (3.5) and (3.6) can be re-stated into the original input space $\mathscr{X}$ using (3.3):

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{m=1}^{M} \sum_{l=1}^{|N_m|} \alpha_{ml} \beta_{ml} I(x_i \in N_{ml}) \right)^2 \tag{3.11}$$

$$s.t. \sum_{m=1}^{M} \sum_{l=1}^{|N_m|} |\beta_{ml}| \leq t, \tag{3.12}$$

Algorithm 3.2: Regularisation of the node characteristic function space in $L1$-norm.

4.

(a) Associate a coefficient β_j to each node characteristic function of filtering parameter α_j;

(b) Given the n samples $\{(z_i = (z_{i,1}, \ldots, z_{i,p}), y_i)\}_{i=1}^{n} \in \{\mathscr{X}, \mathscr{Y}\}$ and the p node characteristic functions, the following optimisation problem is set:

$$\min_{\{\beta_0,\ldots,\beta_p\}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \alpha_j \beta_j z_{ij} \right)^2 \tag{3.9}$$

$$s.t. \qquad \sum_{j=1}^{p} |\beta_j| \leq t. \tag{3.10}$$

(c) Suppress every node of $\{t_1, t_2, \ldots, t_M\}$ whose weight β_j is equal to zero, and the weights of all its siblings and descendants are equal to zero. Re-weigh remaining node with β_j;

with $\boldsymbol{\beta} = \{\beta_0, \beta_{1,1}, \ldots, \beta_{1,|N_1|}, \ldots, \beta_{M,1}, \ldots, \beta_{M,|N_M|}\}$ and $I(x_i \in N_{ml})$, the node characteristic function which is equal to one if the sample x_i reach the node N_{ml} of the m-th decision tree.

Interestingly when node characteristic functions of internal are selected ($\alpha_j, \alpha_{ml} > 0$), we are going to re-weight those. In the pruned and re-weighted ensemble, it means that an internal node will play a role in the propagation of a sample in the decision trees *and* may participate to the prediction with its descendants.

Chapter 4

Experimental evaluation

4.1 Goals of the experimental part

Our goals in the experimental part are the following:

- determine if it is possible to reduce the complexity of an ensemble of randomized trees with our approach;
- determine if we can improve accuracy over the original ensemble of randomized trees;
- evaluate the influence of the choice of characteristic node functions on the performance;
- evaluate the behaviour of the combination of the ensemble of randomized tree algorithm and feature selection methods with their parameters.

From this analysis, we hope to find new research directions and a better understanding of our approach.

We describe the studied datasets in Section 4.2, our experimental methodology in Section 4.3 and the notations in Section 4.4. Then, we present the results of some experiments: we study in Sections 4.5, 4.6, 4.7, 4.8, 4.9 the impact of each parameter of the combined algorithms. Next, we study the impact of the learning set size in Section 4.10. Finally, we draw conclusions of our experiments in Section 4.11.

4.2 Datasets

Artificial datasets are re-generated for each new experiment. Real datasets have only a limited number of samples, so the entire sample set is shuffled at each new experiment in order to increase the variability of the learning set.

Friedman1. Friedman1 is a regression benchmark problem described in [Friedman, 1991]. The goal is to estimate an output y with ten input variables taken in a uniform distribution $\mathscr{U}(0,1)$, half of them are irrelevant:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - \frac{1}{2})^2 + 10x_4 + 6x_5 + \varepsilon; \tag{4.1}$$

where ε is a Gaussian noise $\mathscr{N}(0,1)$.

There are 300 samples in the learning set and 2000 samples in the testing set.

Two-norm. The Two-norm problem is a classification benchmark problem described in [Breiman, 1996b]. There are twenty variables taken in a normal distribution: either in $\mathscr{N}(-a,1)$ if the class is 0 or in $\mathscr{N}(a,1)$ if the class is 1. The parameter a is equal to:

$$a = \frac{2}{\sqrt{20}} \tag{4.2}$$

There are 300 samples in the learning set and 2000 samples in the test set. The input variables are strongly correlated.

SEFTi. The SEFTi dataset is a regression problem described in [AA&YA Intel, 2008]. It is a simulated dataset which reproduces the tool level fault isolation in a semiconductor manufacturing. There are 600 input attributes, 4000 samples and one output. One quarter of the values are missing at random. Missing values were replaced by the median.

The dataset is split into a learning and a testing set of respectively 2000 samples.

Datasets summary. The table 4.1 recapitulates the main characteristics of the datasets.

Datasets	type	$\|LS\|$	$\|TS\|$	$\|attributes\|$	$\|classes\|$
Friedman1	regression	300	2000	10	n/a
Two-Norm	classification	300	2000	20	2
SEFTi	regression	2000	2000	600	n/a

Table 4.1 – Overview of benchmark datasets

$|\bullet|$ denotes the cardinality of a set

4.3 Experimental methodology

The ensemble of randomised trees are generated with extremely randomized trees as argued in Section 2.3.3. This ensemble is latter regularised with $L1$-norm with incremental forward stagewise regression algorithm which computes the monotone LASSO. We will evaluate the use of either all nodes or all leaves as node characteristics functions.

When monotone LASSO is applied, we have to choose a particular point of the path in order to obtain the final model. For every graph except those of Section 4.5, we have used the re-substitution error to pick the best one. However as we pointed out in the section 2.2, it is probably not the best way to select a model. The sparsest models have been favoured whenever several models have the same estimated risk $\hat{R}$, *i.e.* the average loss l of a learnt model $\hat{f}$ on a sample set S:

$$\hat{R} = \frac{1}{|S|} \sum_{(x,y) \in S} l(\hat{f}(x), y). \tag{4.3}$$

The presented results are averaged over 50 models, *i.e.* a measure is average over 50 independent experiments. The complexity of extra trees are measured by the sum of the number of leaves or nodes of each tree in the ensemble. For incremental forward stagewise regression and regularised forest, the complexity of the model is equal to the $L0$-norm of the vector formed by the coefficient associated to a feature.

In order to have a fixed magnitude order, we have decided to pre-whiten datasets before any learning operation: input and output data are pre-whitened to have unit variance and zero mean. Binary classification will be studied under a regression perspective, *i.e.* models are learned with a regression algorithm and hence a threshold is applied to the output of the model.

Note that the results shown for the incremental forward stagewise regression and extremely randomized tree algorithm are performed on the original datasets.

4.4 Notations

Table 4.2 presents the notations used to abbreviate captions and reminds the meaning of symbols used for algorithm parameters.

4.5 Understanding the path algorithm in randomized tree feature space

The optimisation path algorithm allows to retrieve the entire solution path of the regularisation problem. In this section, we take a closer look at the evolution of the solution path. Moreover, we will try to improve our understanding of the embedded feature selection process.

The diagrams presented at Figures 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6 show the evolution of the estimated risk given the number of selected nodes or t the sum of the absolute values of the node weights. Results are presented both on the learning set and on the testing set for extra trees, pruned extra trees and incremental forward stagewise regression. Parameters of the different algorithms are specified on the figure. *Note that the number of considered node characteristic functions is different when we project on all nodes or only on leaves.*

The incremental forward stagewise regression algorithm selects and weights features during its *learning procedure* (see Figure 4.1 for Friedman1, see Figure 4.3 for Two-Norm and see Figure 4.5 for SEFTi). The estimated risk falls until it is equal to zero or the path algorithm has converged

Notation	
ET	an abbreviation for extremely randomized trees (see Section 2.3)
rET	an abbreviation for regularized extra trees, which corresponds to the model obtained after having regularised the node characteristic functions as described in Sections 3.3 and 3.4
ifsr	an abbreviation for incremental forward stagewise regression (see Section 2.4.3)
M	the number of terms in an ensemble of randomized trees
K	the number of randomly selected attributes in the extremely randomized trees at the tree structure learning procedure
n_{min}	the parameter that controls pre-pruning: in order to split a node, there should be at least n_{min} samples
ε	is the size of a step made in the incremental forward stagewise regression algorithm, *i.e.* the incremented or decremented weight at each step
t_p	is the Lp-norm of the coefficients associated to a weight in the (monotone) LASSO or in the regularised extra trees

Table 4.2 – Notations and abbreviations used in the experimental Section.

with the given node characteristic features. Regularised extra trees are able to perfectly fit the learning samples as well as the unregularised version with $n_{min} = 1$.

For the Friedman1 and the Two-Norm problem, results obtained on the *testing set* are pretty similar to those obtained on the learning set (compare Figure 4.2 with 4.1 for Friedman1 and Figure 4.4 with 4.3 for Two-Norm). However in the SEFTi problem (compare Figure 4.6 with 4.5), the minimum of the estimated risk on the testing set is not located at the same place on the learning set. This means that a more advance technique such as K-fold cross-validation can improve performance. Note that we have limited the number of terms in the ensemble and pre-pruned each tree for practical reasons for the SEFTi dataset.

These first results on Friedman1 (see Figure 4.2), Two-norm (see Figure 4.4) and SEFTi (see Figure 4.6) datasets show that the *projection on all nodes* is better than the projection on all leaves. However with appropriate parameters, we are able to obtain similar accuracies with a projection on leaves (see Figure 4.7 for Friedman1 and Figure 4.8 for Two-Norm) where we have pre-pruned the ensemble.

The incremental forward stagewise regression algorithm selects sparsely the node characteristic features generated by an ensemble of extremely randomized trees. The combination of both leads to promising results that we will investigate more deeply in the next sections.

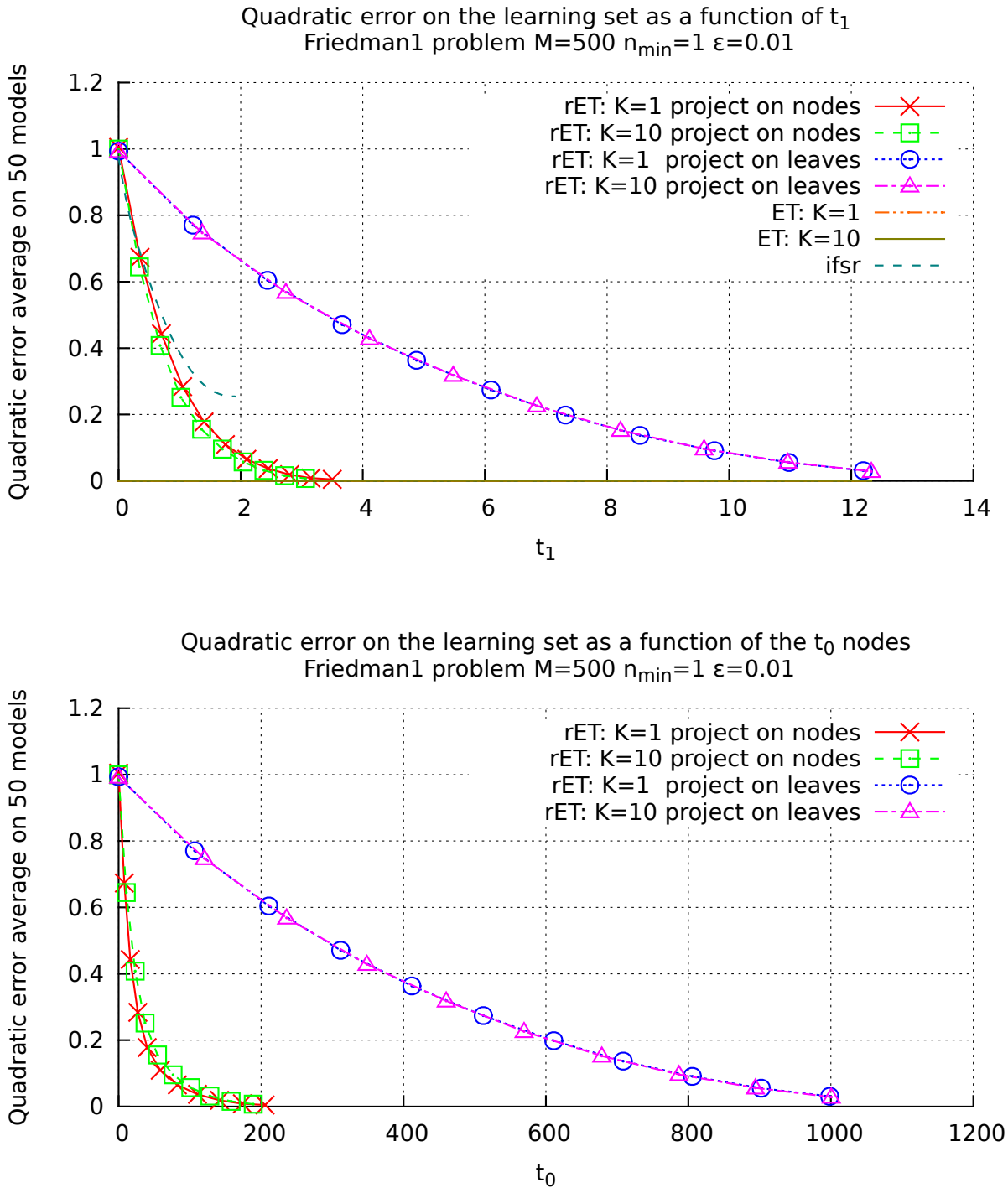

Figure 4.1 – Results for all figures are obtained on the **learning set** of the **Friedman1** dataset. For parameter values see the figure. On the *top*, the evolution of the quadratic error as a function of t_1 for extra trees, regularized extra trees and incremental forward stagewise regression. On the *bottom*, the evolution of the quadratic error as a function of t_0 for the regularised extra trees.

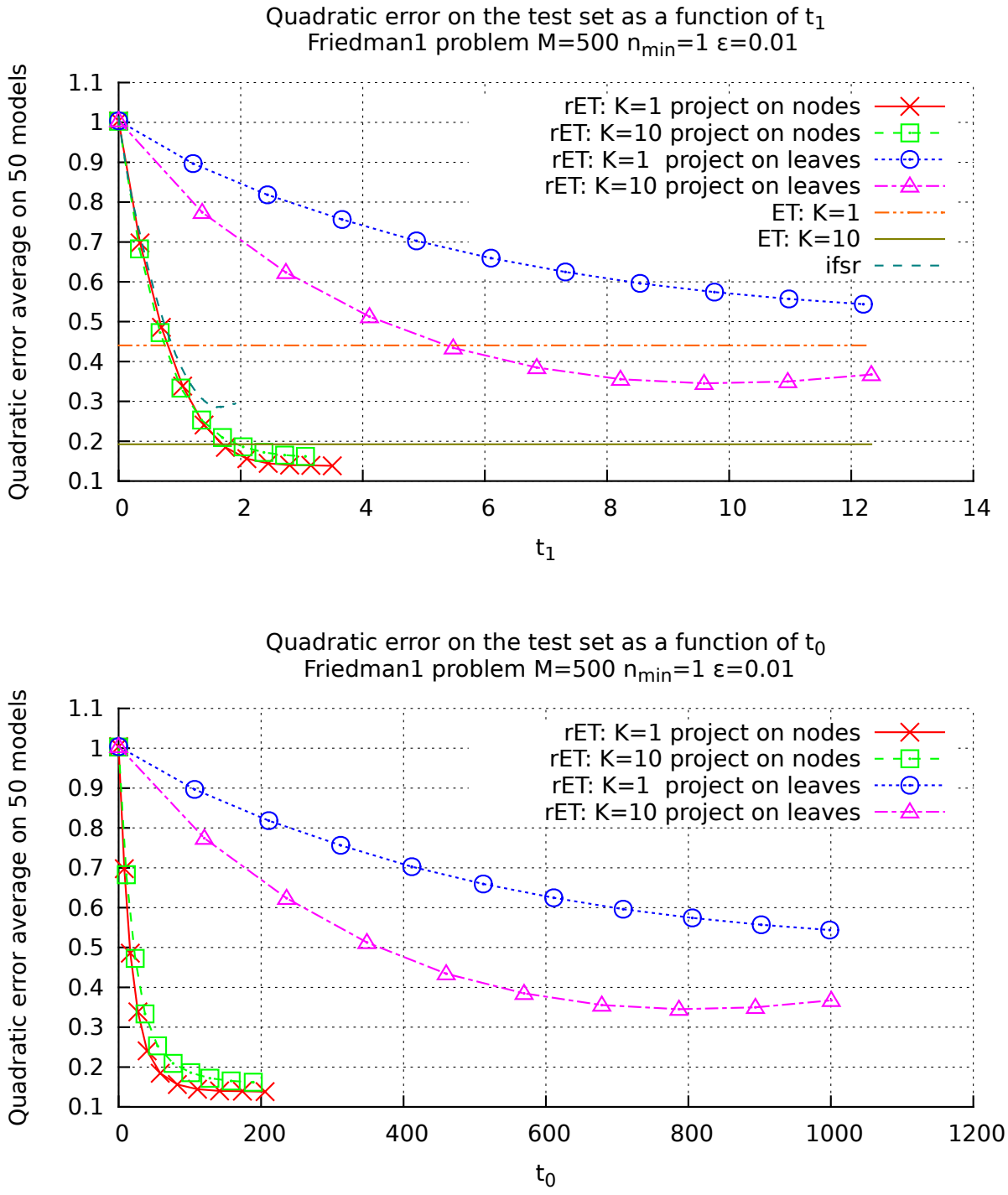

Figure 4.2 – Results for all figures are obtained on the **testing set** of the **Friedman1** dataset. For parameter values see the figure. On the *top*, the evolution of the quadratic error as a function of t_1 for extra trees, regularized extra trees and incremental forward stagewise regression. On the *bottom*, the evolution of the quadratic error as a function of t_0 for the regularised extra trees.

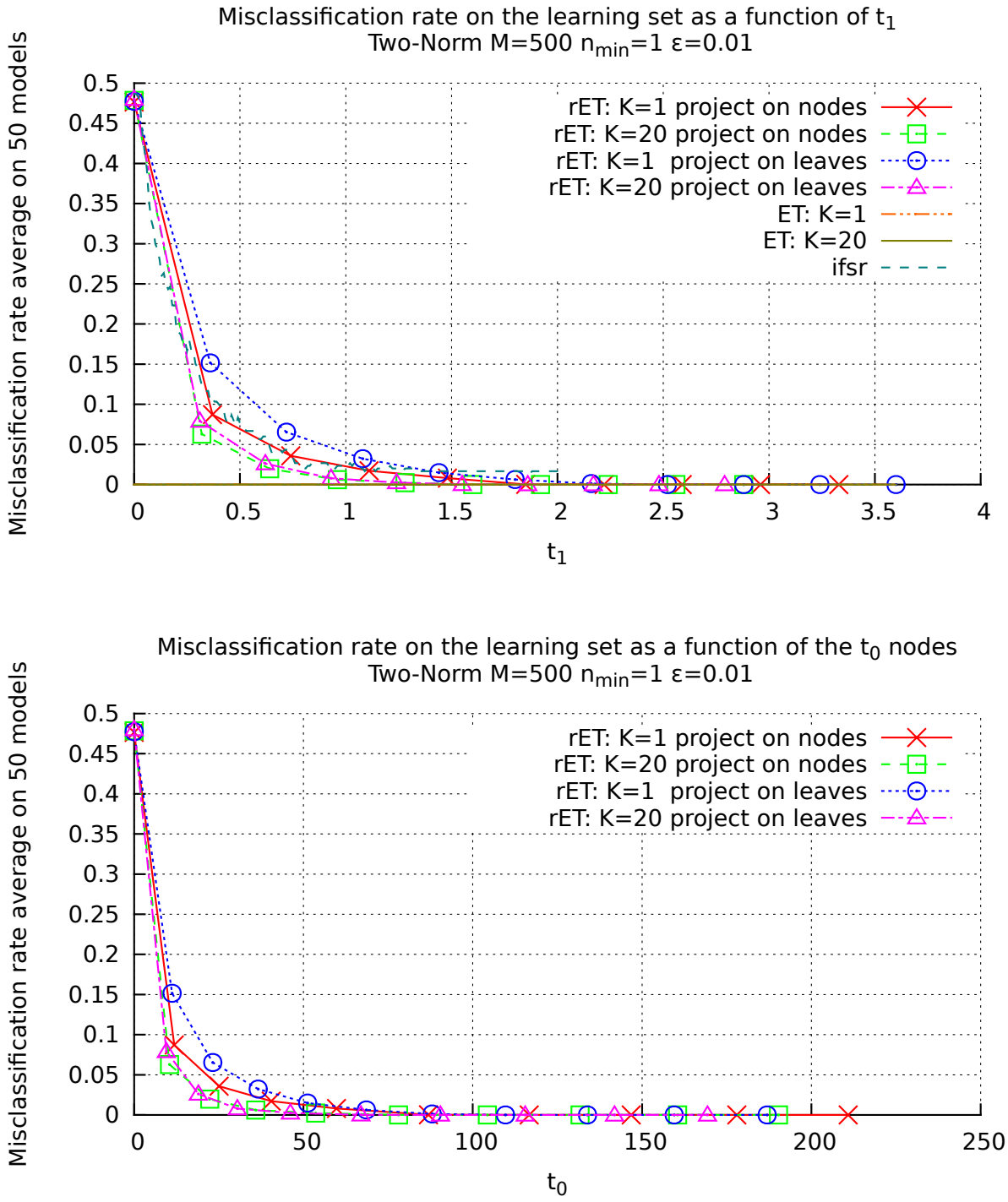

Figure 4.3 – Results for all figures are obtained on the **learning set** of the **Two-Norm** dataset. For parameter values see the figure. On the *top*, the evolution of the misclassification rate as a function of t_1 for extra trees, regularized extra trees and incremental forward stagewise regression. On the *bottom*, the evolution of the misclassification rate as a function of t_0 for the regularised extra trees.

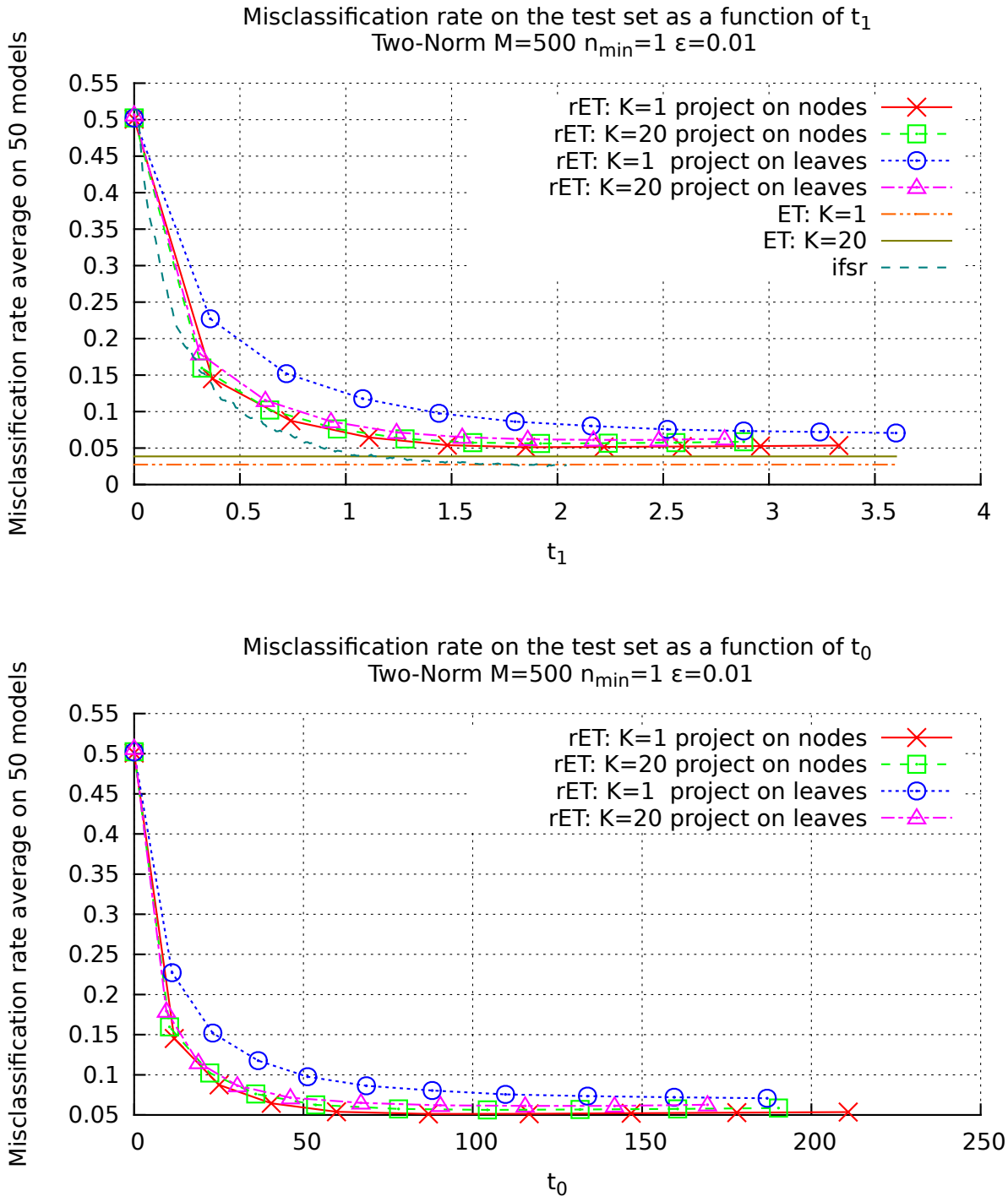

Figure 4.4 – Results for all figures are obtained on the **testing set** of the **Two-Norm** dataset. For parameter values see the figure. On the *top*, the evolution of the misclassification rate as a function of t_1 for extra trees, regularized extra trees and incremental forward stagewise regression. On the *bottom*, the evolution of the misclassification rate as a function of t_0 for the regularised extra trees.

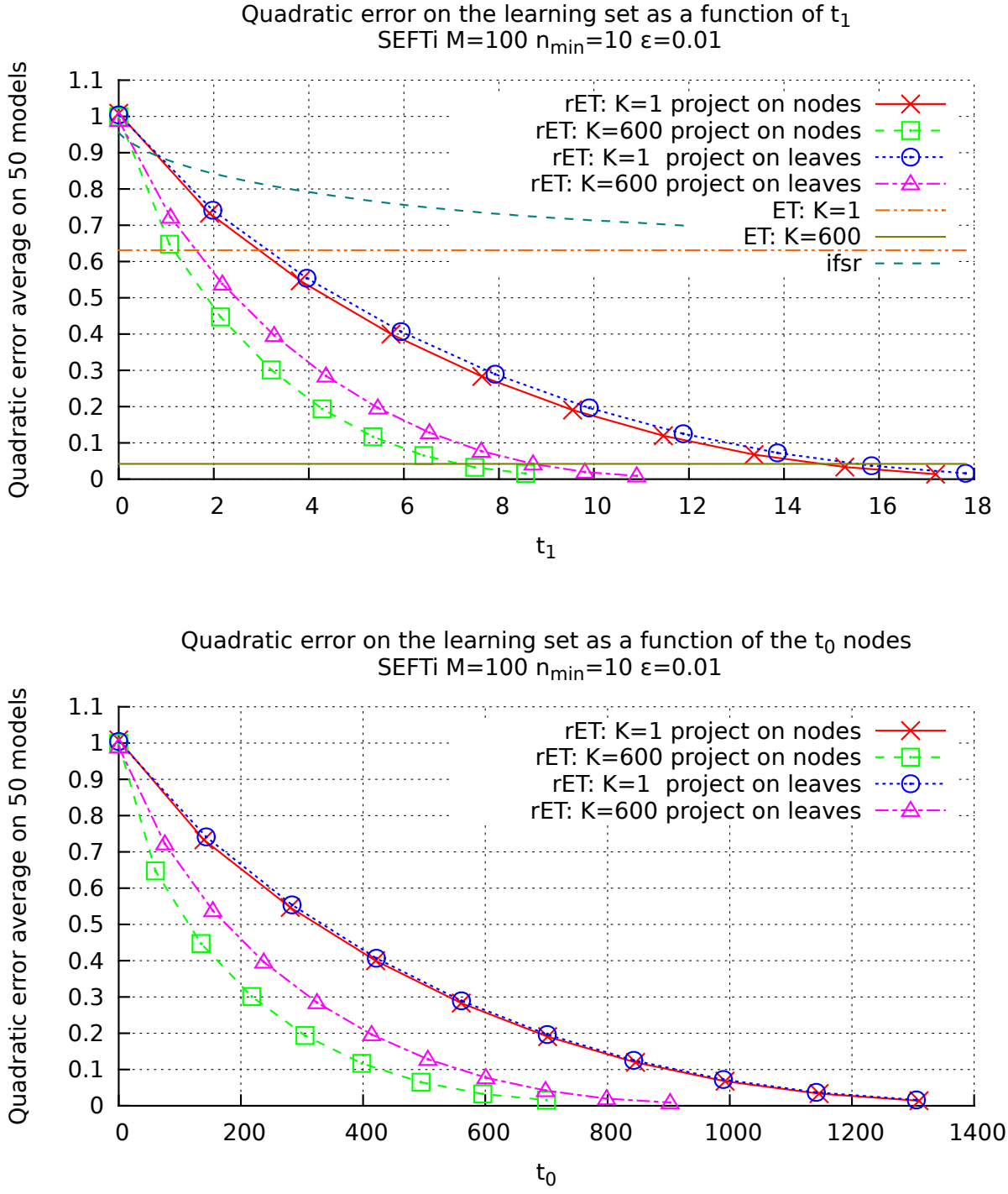

Figure 4.5 – Results for all figures are obtained on the **learning set** of the **SEFTi** dataset. For parameter values see the figure. On the *top*, the evolution of the quadratic error as a function of t_1 for extra trees, regularized extra trees and incremental forward stagewise regression. On the *bottom*, the evolution of the quadratic error as a function of t_0 for the regularised extra trees.

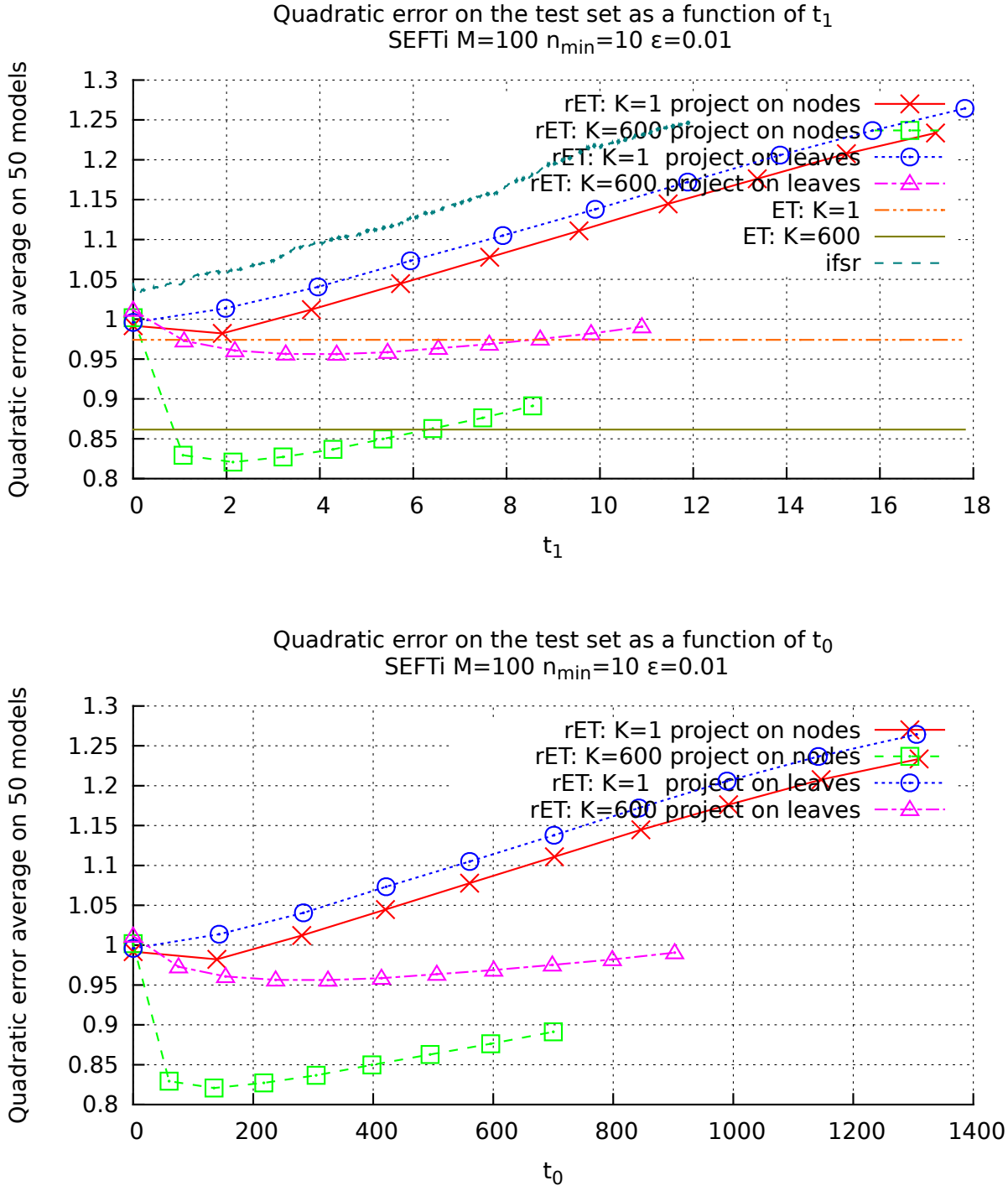

Figure 4.6 – Results for all figures are obtained on the **testing set** of the **SEFTi** dataset. For parameter values see the figure. On the *top*, the evolution of the quadratic error as a function of t_1 for extra trees, regularized extra trees and incremental forward stagewise regression. On the *bottom*, the evolution of the quadratic error as a function of t_0 for the regularised extra trees.

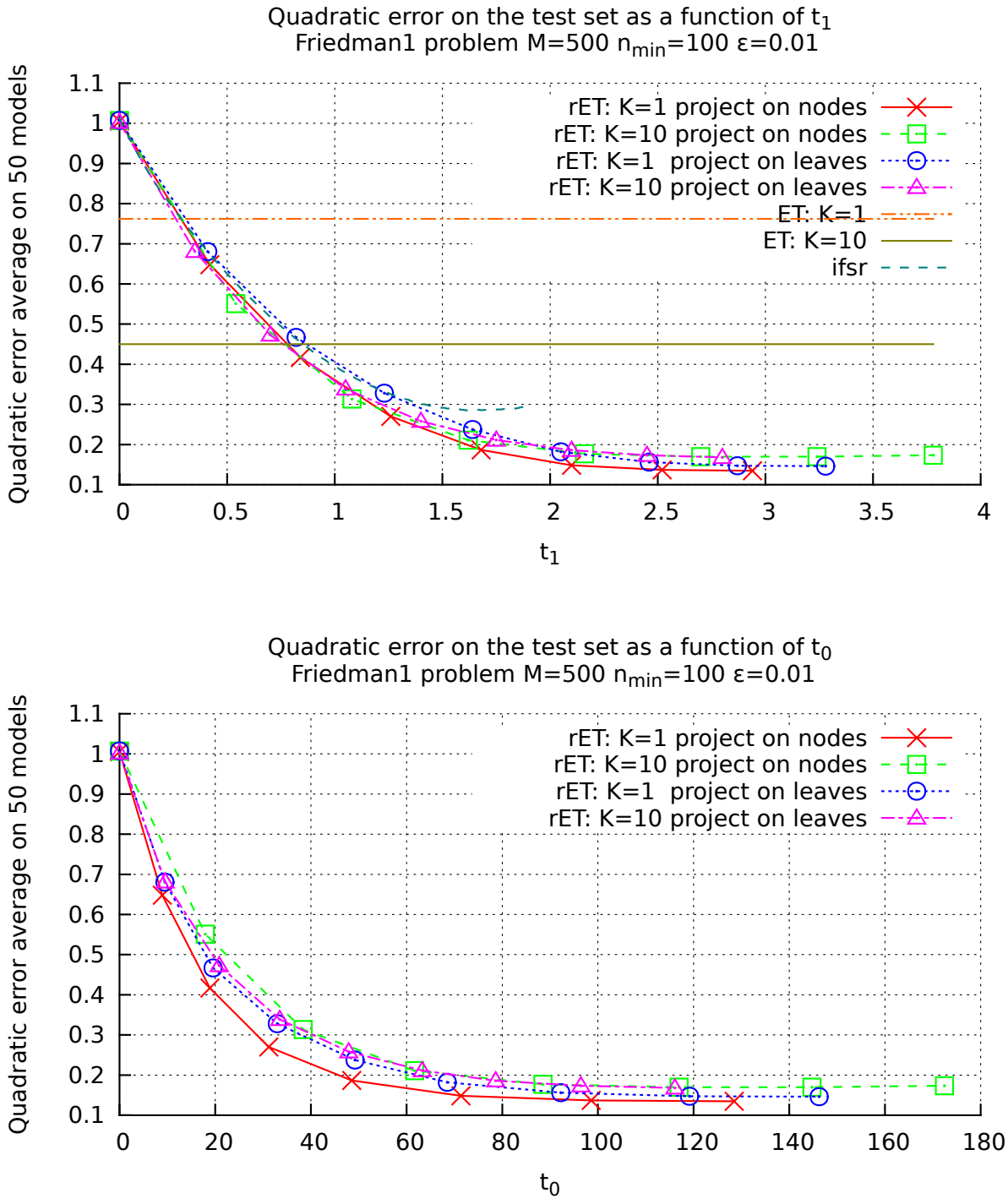

Figure 4.7 – Results for all figures are obtained on the **testing set** of the **Friedman1** dataset. For parameter values see the figure. On the *top*, the evolution of the quadratic error as a function of t_1 for extra trees, regularized extra trees and incremental forward stagewise regression. On the *bottom*, the evolution of the quadratic error as a function of t_0 for the regularised extra trees.

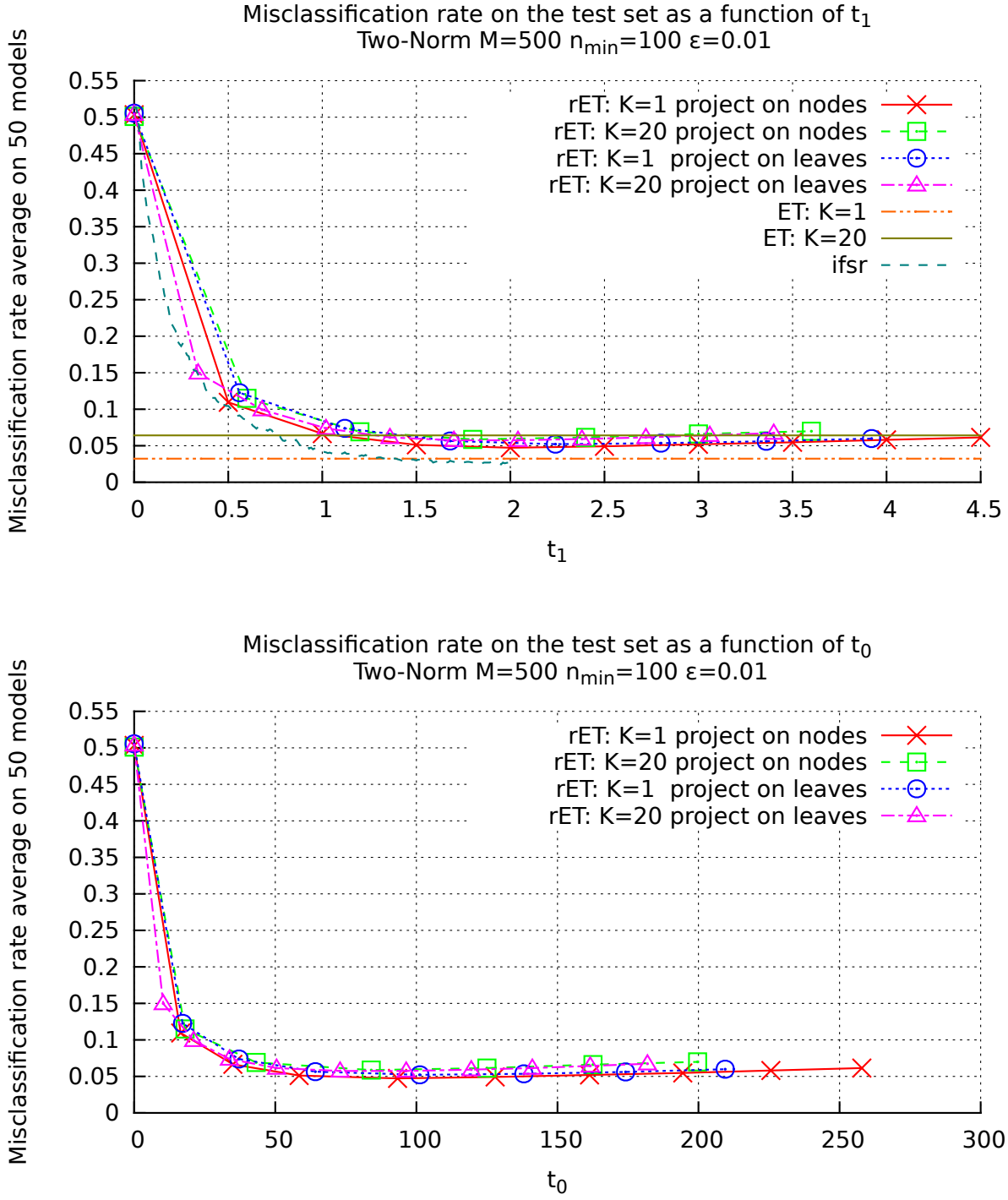

Figure 4.8 – Results for all figures are obtained on the **testing set** of the **Two-Norm** dataset. For parameter values see the figure. On the *top*, the evolution of the misclassification rate as a function of t_1 for extra trees, regularized extra trees and incremental forward stagewise regression. On the *bottom*, the evolution of the misclassification rate as a function of t_0 for the regularised extra trees.

4.6 Effect of n_{min} or pre-pruning

Effect on accuraccy

The parameter n_{min} allows to pre-prune during the learning phase. As we mentioned in section 2.3, pre-pruning reduces the variance by limiting the complexity of a decision tree.

The graphs of Figures 4.9, 4.10 and 4.11 show simultaneously the evolution of the estimated risk with respect to n_{min} and a number of terms M equal to 1 and 100 or 500 on each dataset. On each graph, we have six curves: the extra trees with K equal to 1 or the number of input attributes and the regularised forest with the same values for K where we project either on every node or only on the leaves.

Trends in Friedman1 and Two-norm problem. In the Friedman1 (see Figure 4.9) and Two-Norm (see Figure 4.10) problems, the regularized forest with $M = 100$ is *less sensible* to the parameter n_{min} compared with the extra trees. The comparison between regularized forests presents two trends:

1. When we project *only on the leaves*, the estimated risk falls until the forest is enough pre-pruned. An increase of K appears to diminish this tendency. When we project *on nodes*, the estimated risk is monotonically increasing with n_{min}, suggesting that the method is then able to automatically prune the tree.

2. Different projection techniques with a same value of K seem to converge to similar curves when n_{min} increases.

The first point suggests that generated features with only external nodes are too specific and over-fit the data. And also an increase of K may help to reject worse splits and filter irrelevant attributes. Nevertheless in the Friedman1 problem (see Figure 4.9), a high value of K leads to inferior accuracy. This indicates that rejecting irrelevant variables is not always necessary. High value of K could diminish the diversity of node characteristic functions.

The second trend can be explained by the fact that when n_{min} increases, it diminishes the height of each generated tree and this shrinks the space of possible generated features. So both projection techniques have same results when we restrict too much the size of the tree.

Trends in SEFTi. In the SEFTi problem (see Figure 4.11), we observe that the value of K is of major importance and helps greatly to reduce the error. The best results obtained with our approach seem inferior to extra trees. *But indeed, it mainly comes from a bad selection of the optimal value of regularisation parameter on the learning set.* In order to be convinced look at Figure 4.6 where regularisation leads to better results for the point $n_{min} = 100$ with the same number of terms.

The regularisations of the extra trees with $K = 1$ have an unusual shape compare to those with $K = 100$ (see the bottom Figure 4.11): the quadratic error increases until $n_{min} = 100$ and then diminishes. More research would be needed to explain this phenomenon.

Case of one decision tree. When there is only one term in the ensemble, the decision tree and its regularized version performs similarly (see top of Figures 4.9, 4.10 and 4.11). An interpretation would be that the $L1$-norm regularisation algorithm has not the opportunity to select better and alternative features, even if every node are provided.

Conclusion. The comparison between extra trees and its regularised counterparts show that most of the time for similar pre-pruning parameters, the regularised version performs better. Furthermore, the latter one gains *a relative insensitivity* in terms of accuracy to pre-pruning. And when the value of n_{min} is appropriately tuned, performance with its regularized counterpart is similar, or even better. When we compare the projection techniques, the projection *on all nodes* leads to higher accuracy than the projection on leaves.

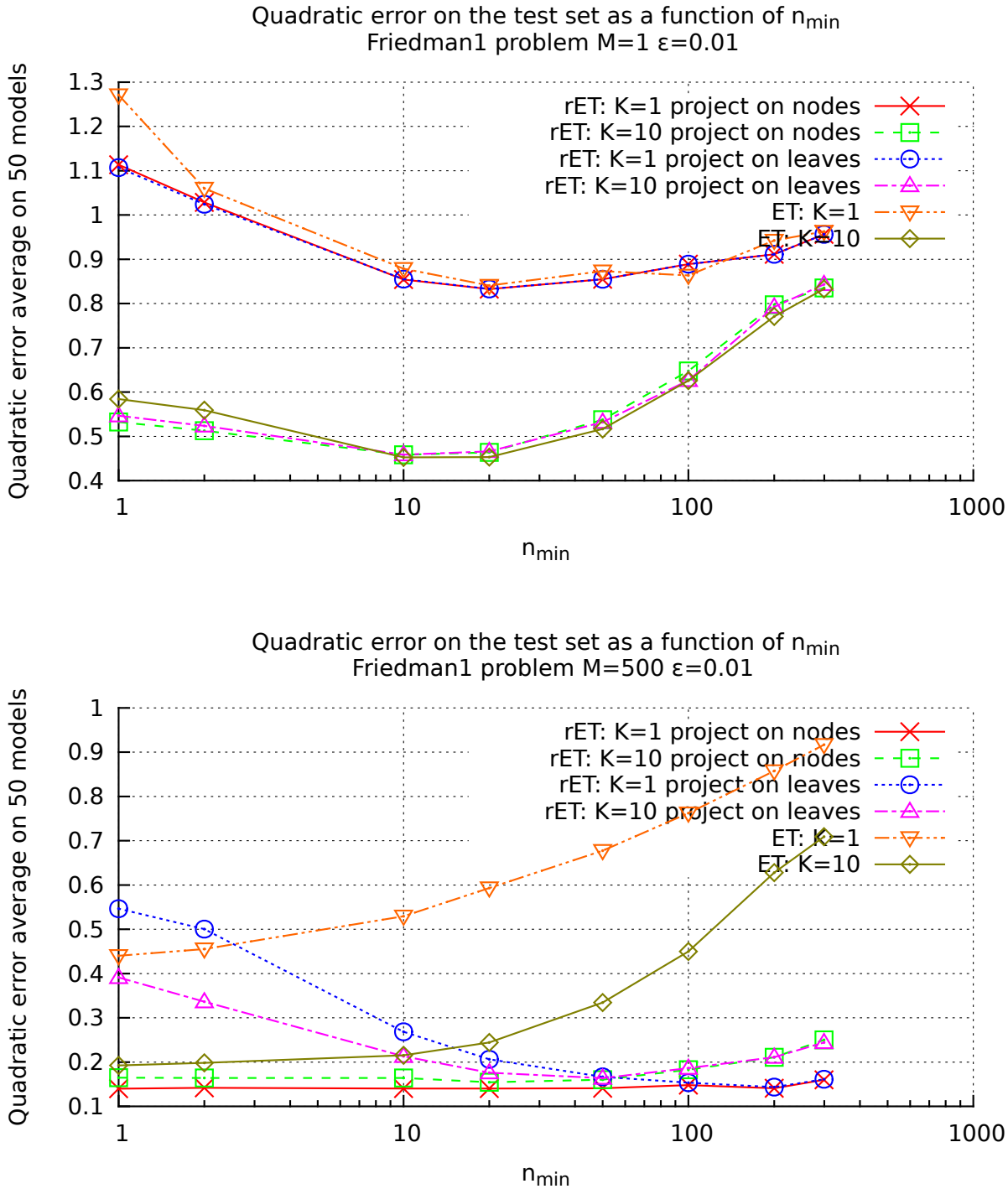

Figure 4.9 – Evolution of the quadratic error as a function of the parameter n_{min} on the Friedman database.

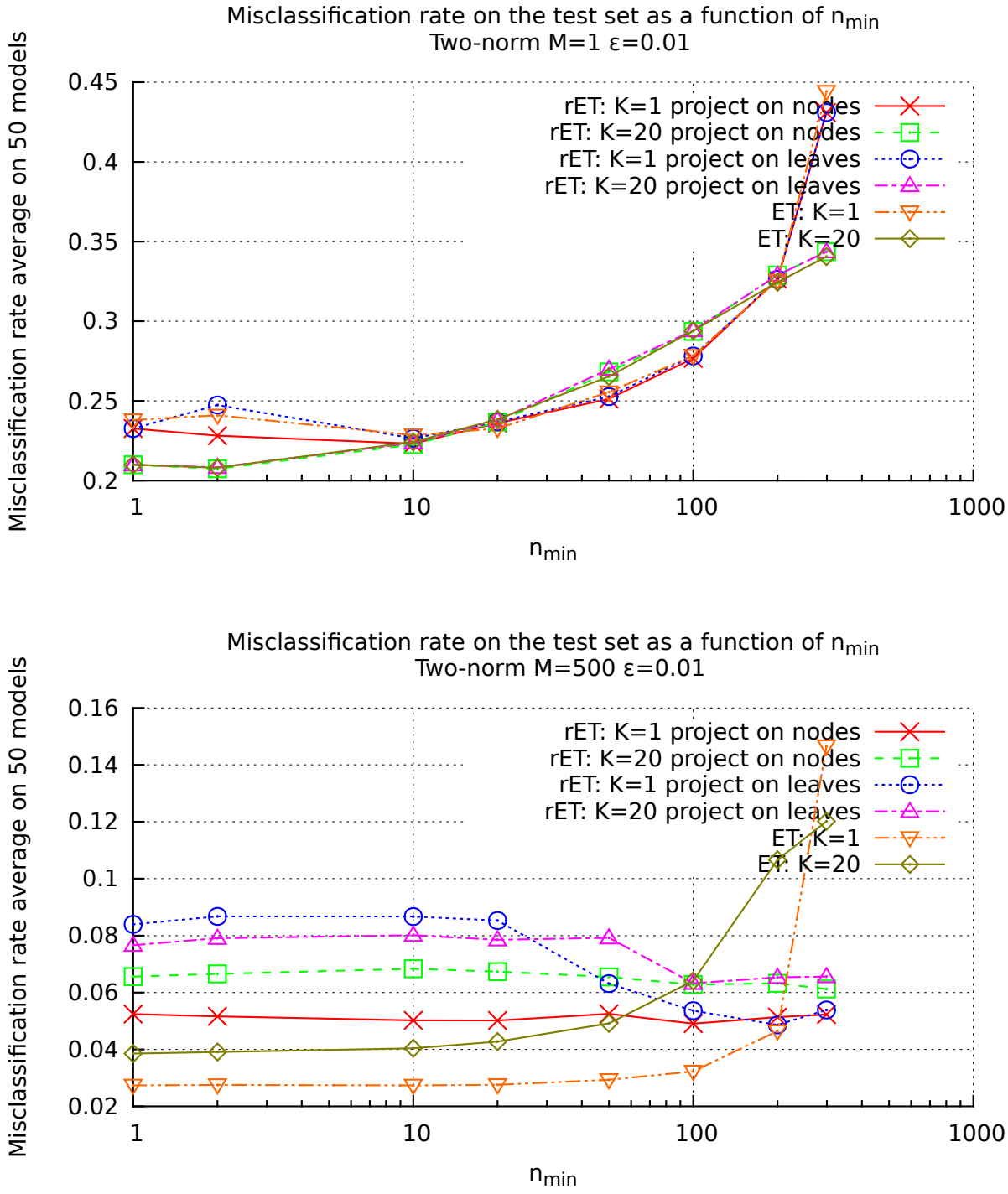

Figure 4.10 – Evolution of the misclassification rate as a function of the parameter n_{min} on the Two-norm database.

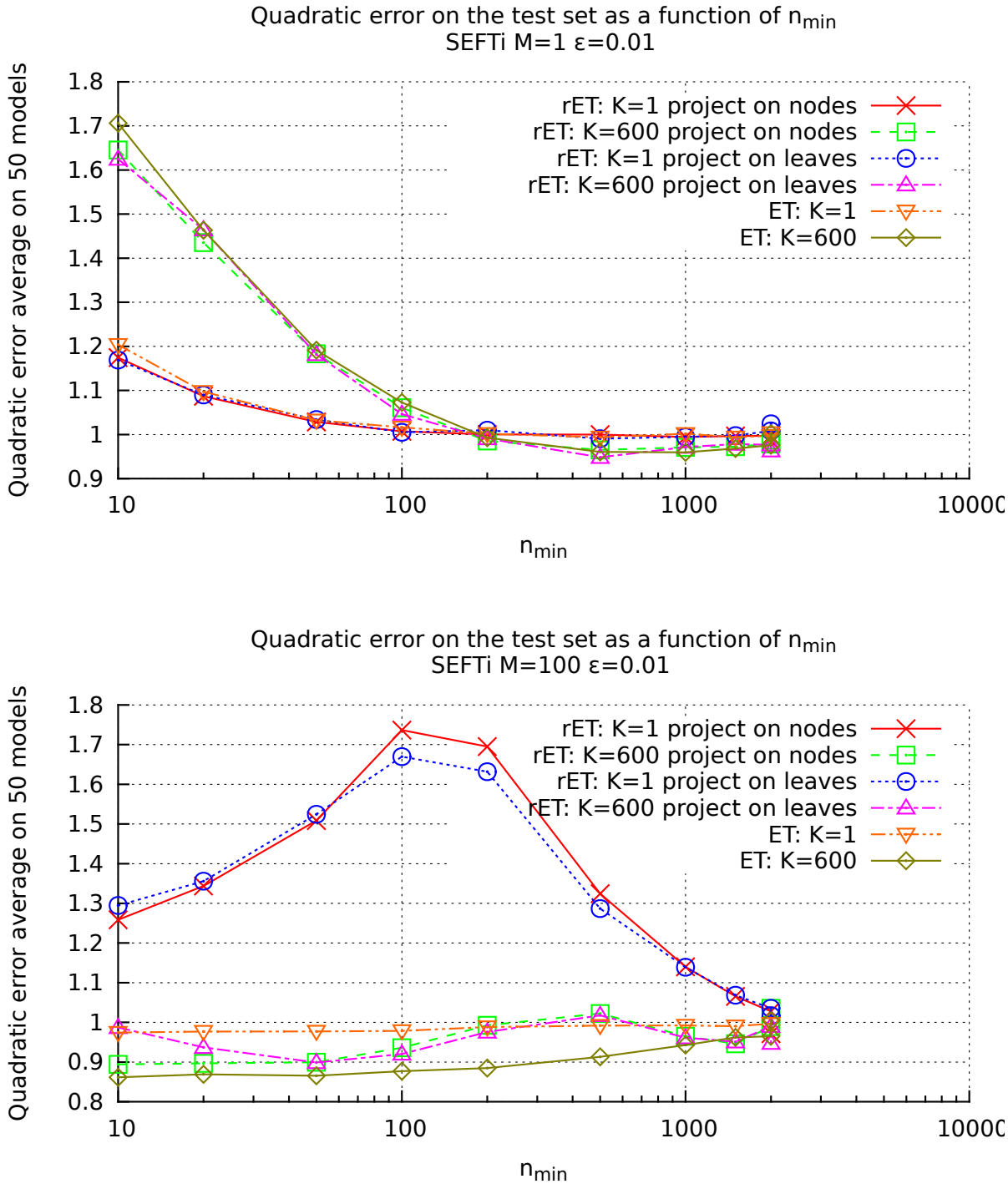

Figure 4.11 – Evolution of the quadratic error as a function of the parameter n_{min} on the SEFTI database.

Complexity reduction

One of the two main goals of this master thesis is to find a robust way to prune ensembles of randomized trees. Due to the randomisation, many nodes are redundant and we hope that we will be able to select a subset of the most relevant nodes.

Figures 4.12, 4.13 and 4.14 present the evolution of the complexity of the forest in function of n_{min} with M equal to 1 and 100 or 500 on the benchmark datasets. The complexity of the forest is measured by the sum of the number of external and internal nodes of each tree. These graphs also show the number of selected characteristic functions associated to the solution of the regularisation problem by four variants of our algorithm: we project only on leaves or on all nodes with K equal to one or the number of attributes of the dataset. Axes are in logarithmic scale.

In the *Two-norm* problem (see Figure 4.13), curves are almost flat and rise slowly until the size of the tree is too much constrained. In the *Friedman1* problem (see Figure 4.14), the complexity of the regularized model has similar behaviour to *Two-Norm* problem (see Figure 4.13), except when we project on leaves. In the *SEFTi* problem (see Figure 4.14), we have a decrease of the complexity of the regularized model with n_{min}. It might come from our model selection strategy for the incremental forward stagewise regression algorithm.

If we do not pre-prune too much the ensemble, we have a *relative insensitivity* in terms of complexity of the model to pre-pruning when projecting to nodes (see bottom of Figure 4.12, 4.13 and 4.14). This is the same behaviour that we have already observed with the evolution of the estimated risk.

With only one randomized tree, the complexity of the regularized model follows the complexity of the decision tree and in particular the number of leaves in extra tree. This might mean that with only one decision tree, the best features are the most complex.

Finally, one goal of this thesis is met. We have a *drastic reduction* of the size of the regularised forest. If we project only on leaves, we have already a pretty good shrinkage in the number of nodes. Only the last layer is indeed pruned, but it is the densest one. *Even better* results are obtained if we project on all nodes of the randomized forest.

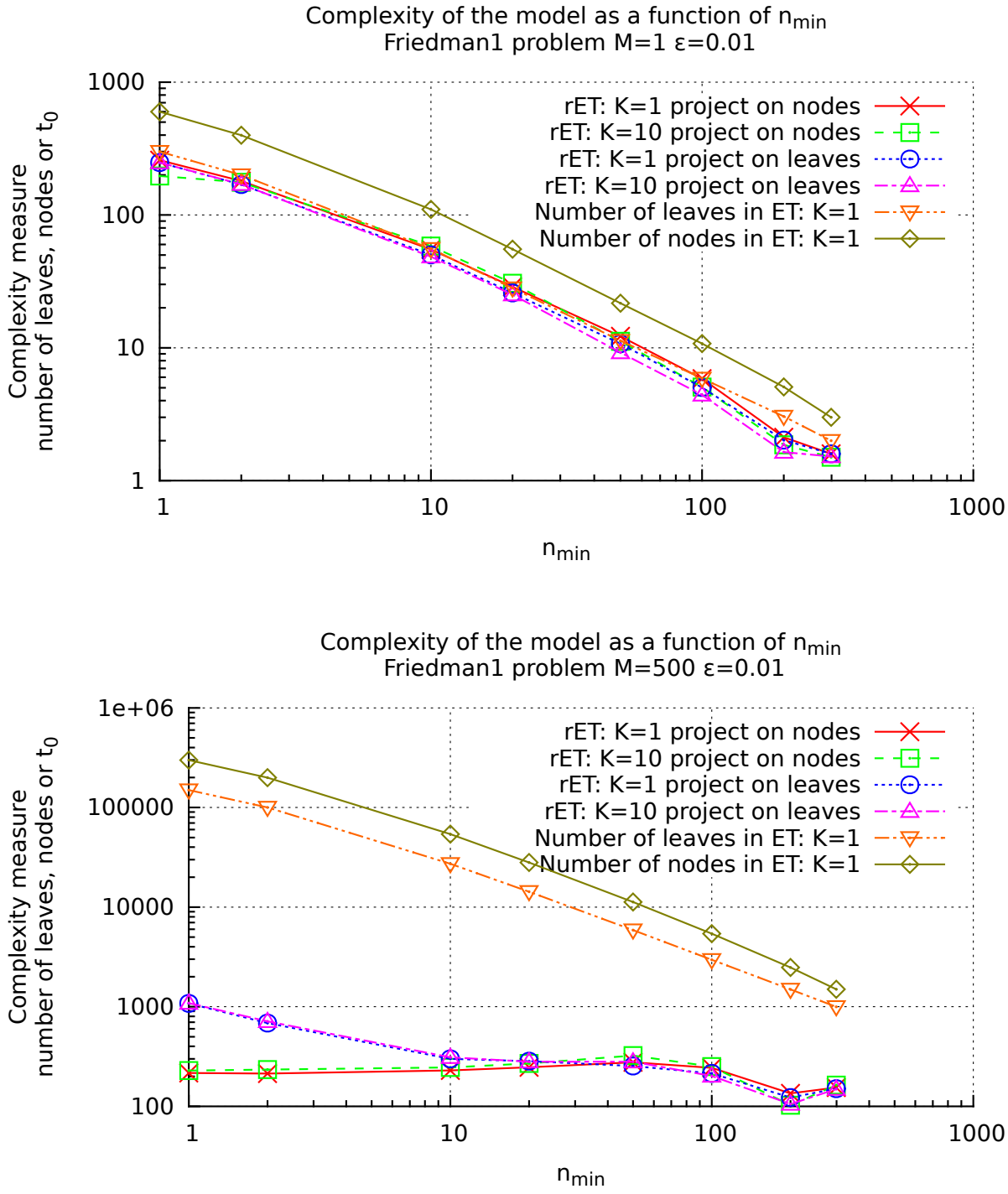

Figure 4.12 – Evolution of complexity of the model with respect to n_{min} on Friedman1 database for ensemble of size M 1 and 500.

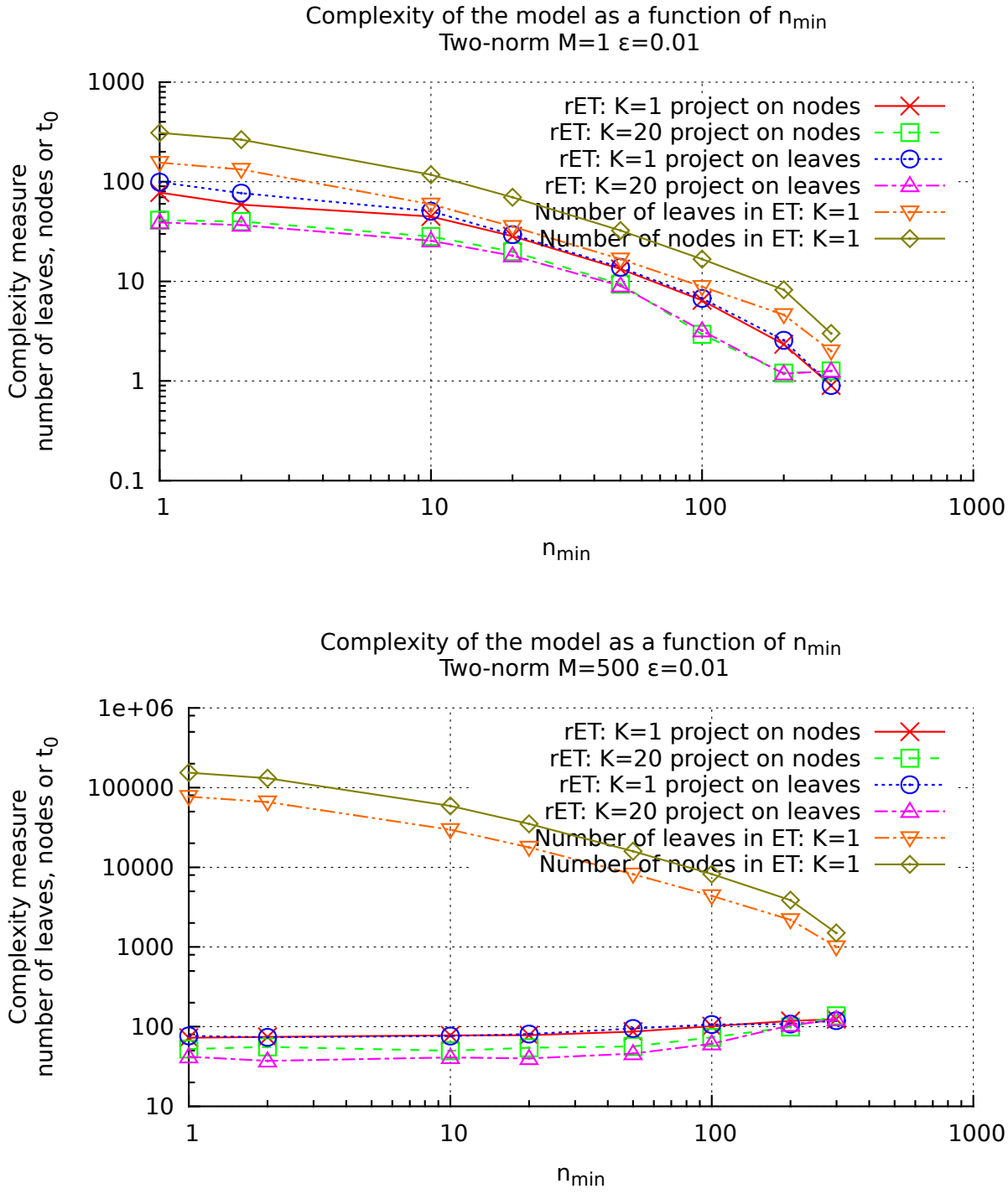

Figure 4.13 – Evolution of complexity of the model with respect to n_{min} on Two-Norm database for ensemble of size M 1 and 500.

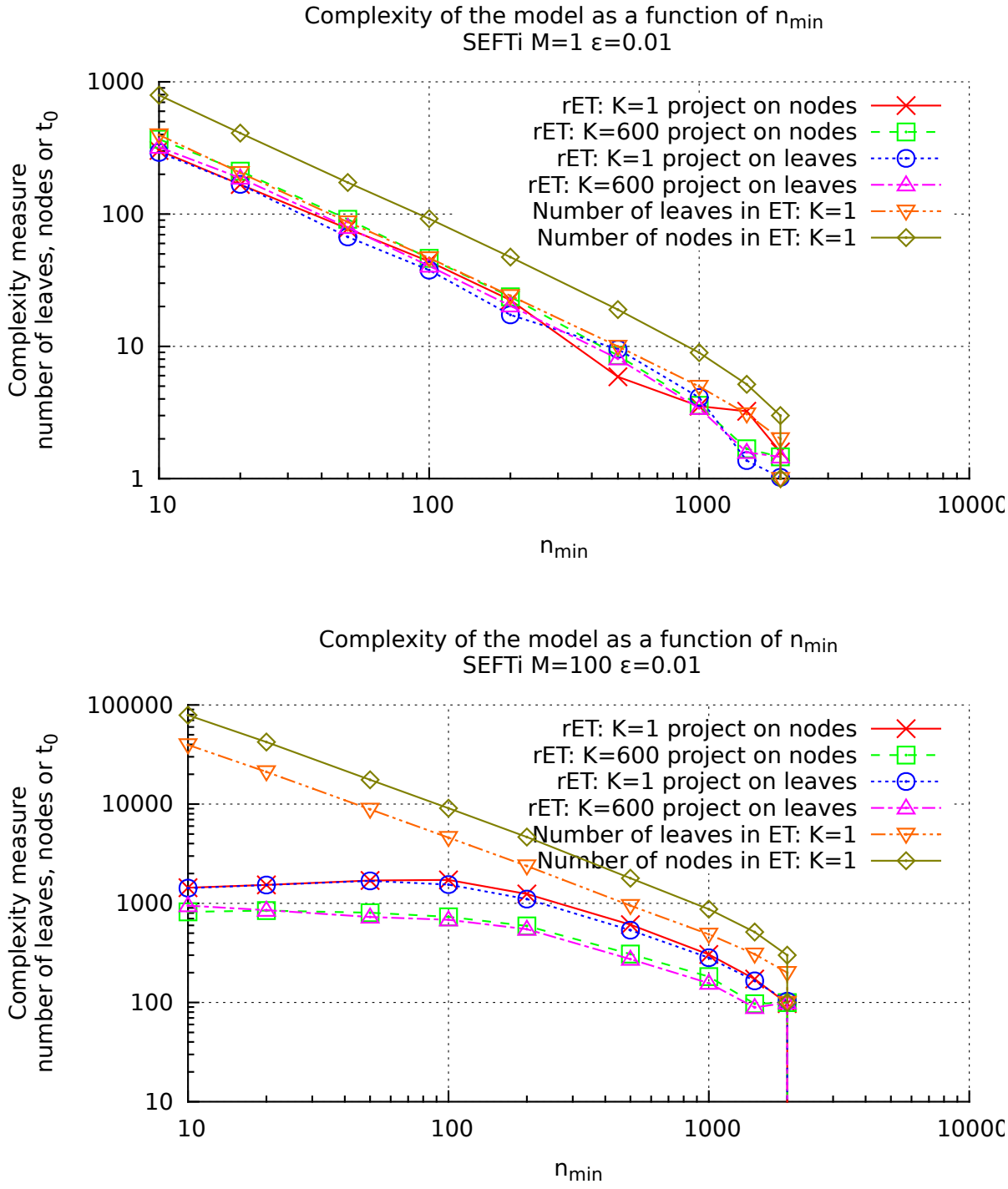

Figure 4.14 – Evolution of complexity of the model with respect to n_{min} on SEFTi database for ensemble of size M 1 and 100.

4.7 Effect of the ensemble size M

The parameter M controls the size of the ensemble. The graphs of Figures 4.15, 4.16 and 4.17 show simultaneously the evolution of the estimated risk and the complexity of the model with respect to M. On each graph, we have six curves: the extra trees with K equals to 1 or the number of input attribute and regularised forests with same value for K where we project either on all nodes or only on the leaves.

Accuracy. As depicted on the top of Figures 4.15, 4.16 and 4.17, a high value of M allows to improve accuracy for every supervised learning algorithm until convergence to a minimum.

In the Friedman1 problem (see Figure 4.15), the parameter K and the projection technique have a high influence on the curves at extreme values of M: *with low values of* M, K is preponderant and conversely with *high values of* M, the projection technique becomes preponderant. This phenomenon also appears Two-Norm problem with less emphasis (see Figure 4.16). In the SEFTi problem (see Figure 4.17) when we increase M with $K = 1$, the accuracy curve raises until the point $M = 64$ and then decreases. It is the opposite behaviour of the extra trees or when $K = 600$ and when we regularised the forest of extra trees. We are not able to explain this strange behaviour yet.

Indeed, K allows to filter out variable and improve performance when we are not able to construct very large ensemble. However, it becomes less obvious when the ensemble size is huge or when there are many irrelevant variables.

Complexity. The complexity values are shown at the bottom of Figures 4.15, 4.16 and 4.17.

The complexity of the ensemble of randomized trees grows linearly with the size of ensemble. When we compare with the number of selected nodes by the monotone LASSO, it is *drastically less*. Whenever there are a few terms in the ensemble, the feature selection algorithm selects a large number of nodes until the complexity of the ensemble is high enough. After this point, the complexity decreases even more with M. The feature selection algorithm has indeed even more possibility to pick the best nodes.

Conclusion. So practically, the regularisation of randomized trees scales very well with the number of terms in the ensemble both in terms of model performance and model compression. And indeed very high ensemble sizes allow to retrieve very compact models for a given sample set.

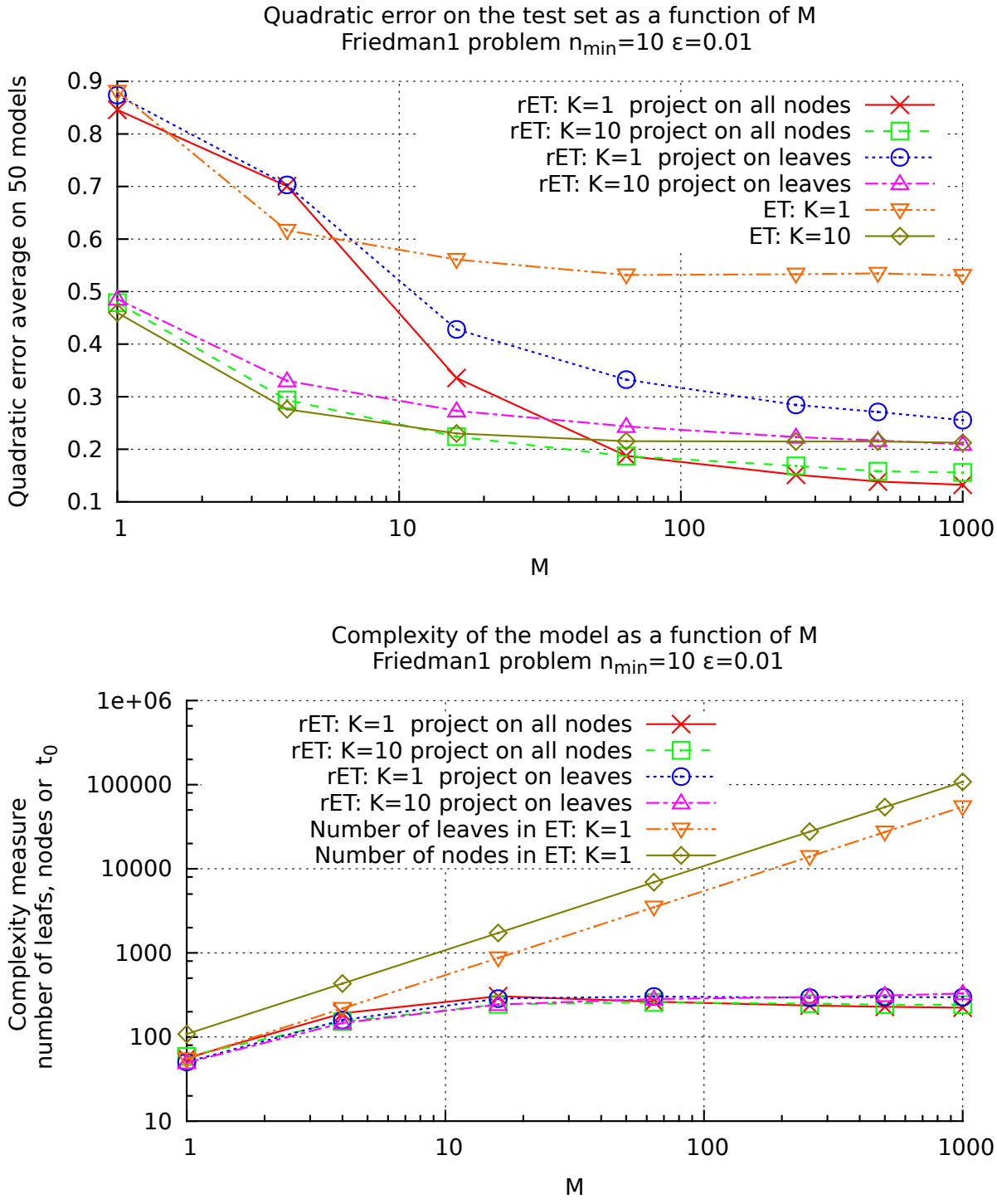

Figure 4.15 – Evolution of the quadratic error and complexity of the model as a function of M, the ensemble size, on the Friedman1 database.

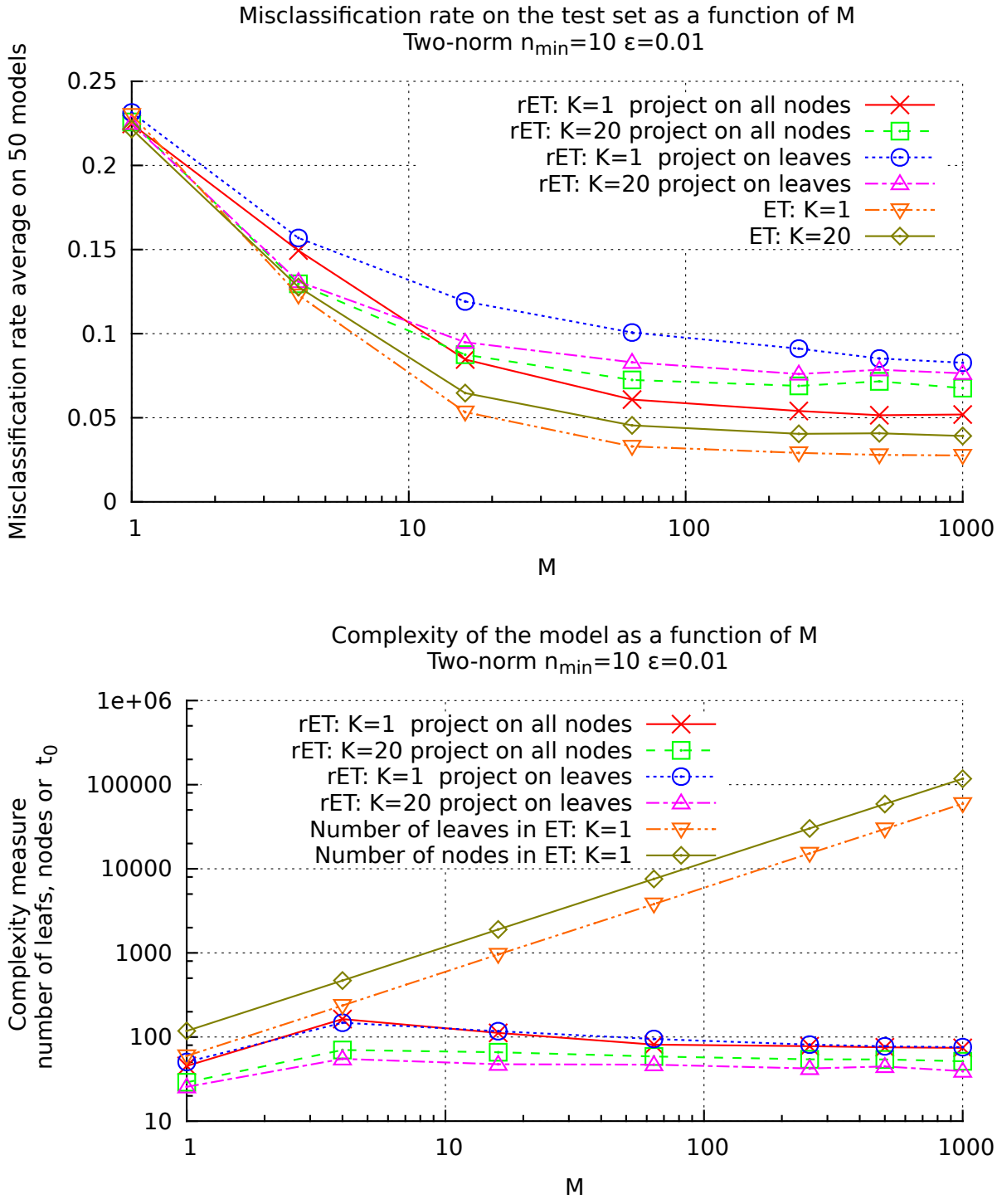

Figure 4.16 – Evolution of the misclassification rate and the complexity of as a function of M, the ensemble size, on the Two-Norm database.

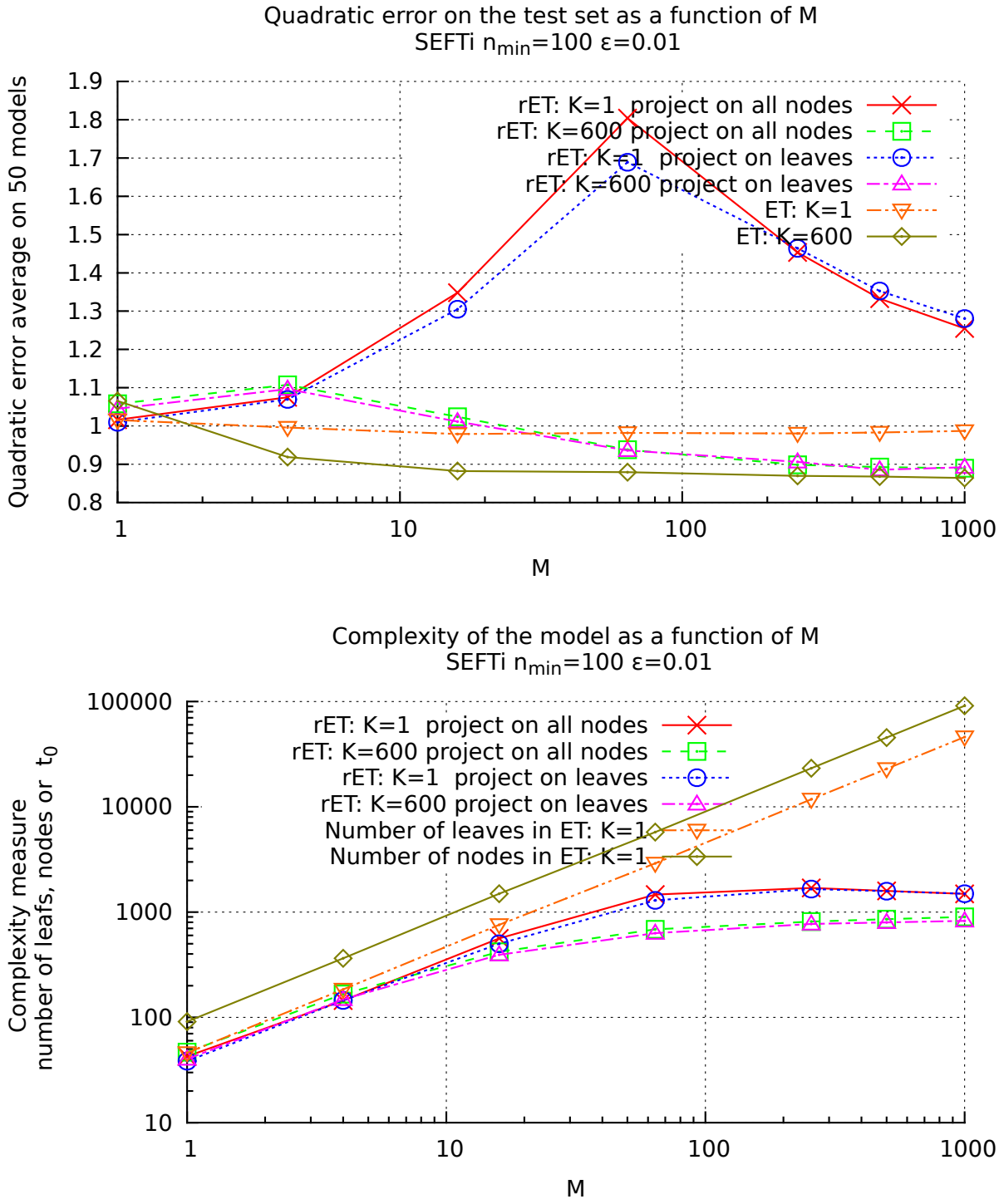

Figure 4.17 – Evolution of the quadratic error and the complexity of as a function of M, the ensemble size, on the SEFTi database.

4.8 Effect of K

The parameter K controls the number of splits drawn without replacement when splitting a node while learning the extremely randomized trees. In this section, we try to study its impact when we apply the feature selection technique.

Figures 4.18, 4.19 and 4.20 depict the evolution of the estimated risk and complexity as a function of K with 100 terms in the ensemble and with n_{min} equal to 1 or 100. We analyse this effect for a forest of extra trees and its regularized counterparts when we project either on leaves or all nodes.

Friedman1 problem. The accuracy curves (see top of Figure 4.18) of the forest of randomized trees have a similar behaviour to the one obtained through the regularisation of leaf characteristic functions. The parameter K helps to improve the performance by rejecting irrelevant variables: the higher K, the greater the performance. Nevertheless when we project on every node, the parameter K helps at first to improve accuracy and then accuracy is degraded.

Two-Norm problem. In the Two-Norm problem (see Figure 4.19), the accuracy decreases when K increases for extra trees and regularized with projection on nodes, but increase with the regularisation with projection on leaves. Note that in this problem all variables are of equal importance.

SEFTi problem. In the SEFTi problem (see Figure 4.20), the performance improves when K increases. The SEFTi problem may contain a large number of irrelevant variables.

Complexity. The complexity of the extra trees and regularised forest is almost insensitive to K, except for the SEFTi problem (see bottom of Figure 4.20). When there are many irrelevant variables, a higher K help to produce more compact model. The node characteristic functions would be of higher quality with high values of K in presence of irrelevant features.

Conclusion Our conclusion is that the parameter K helps to filter out irrelevant variables and helps when we prune the ensemble. However when a representative subset of every possible node is generated as in the Friedman problem, accuracy is degraded.

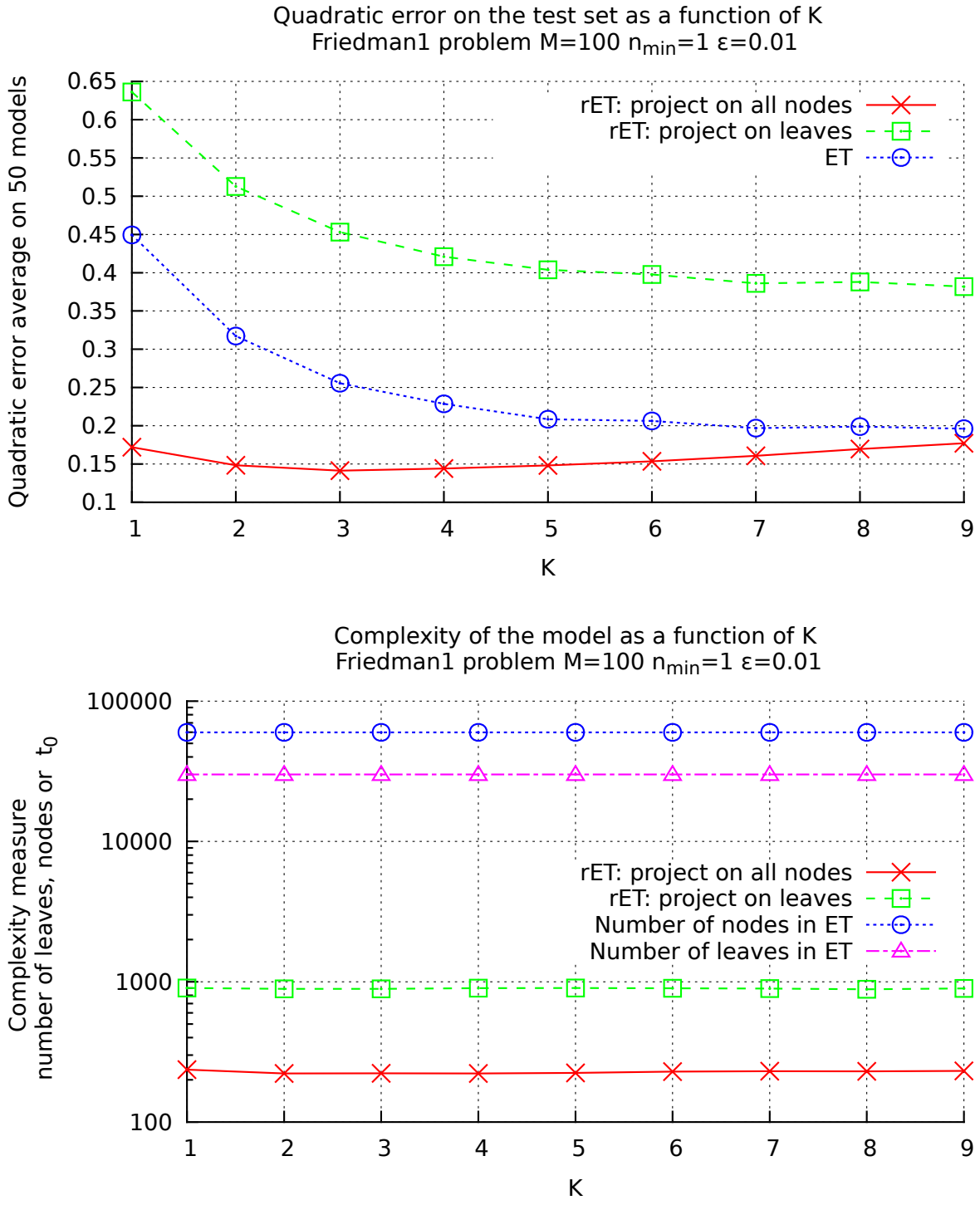

Figure 4.18 – Evolution of the quadratic error and the complexity as a function of K on the Friedman1 problem.

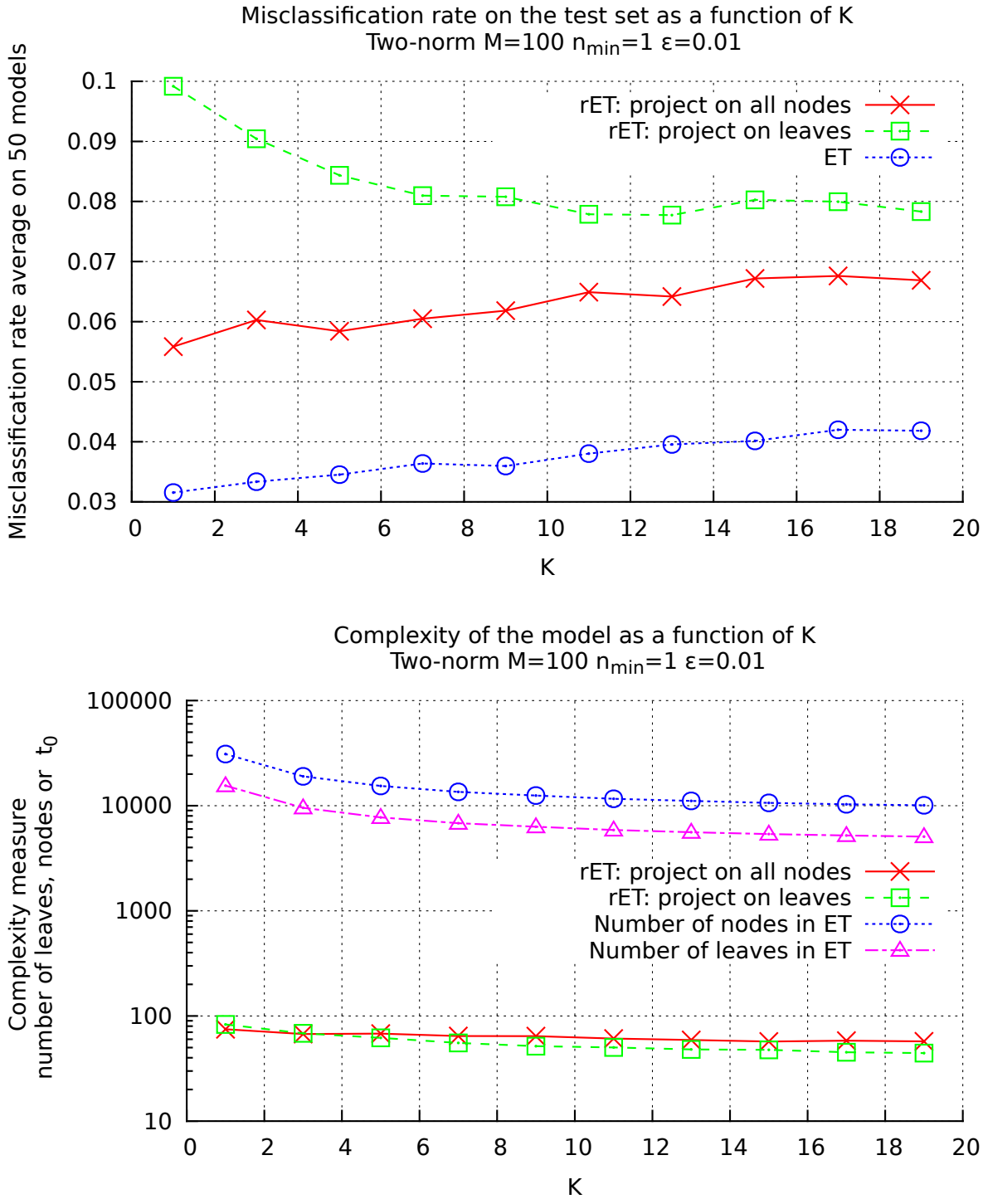

Figure 4.19 – Evolution of the misclassification rate and the complexity as a function of K on the Two-Norm problem.

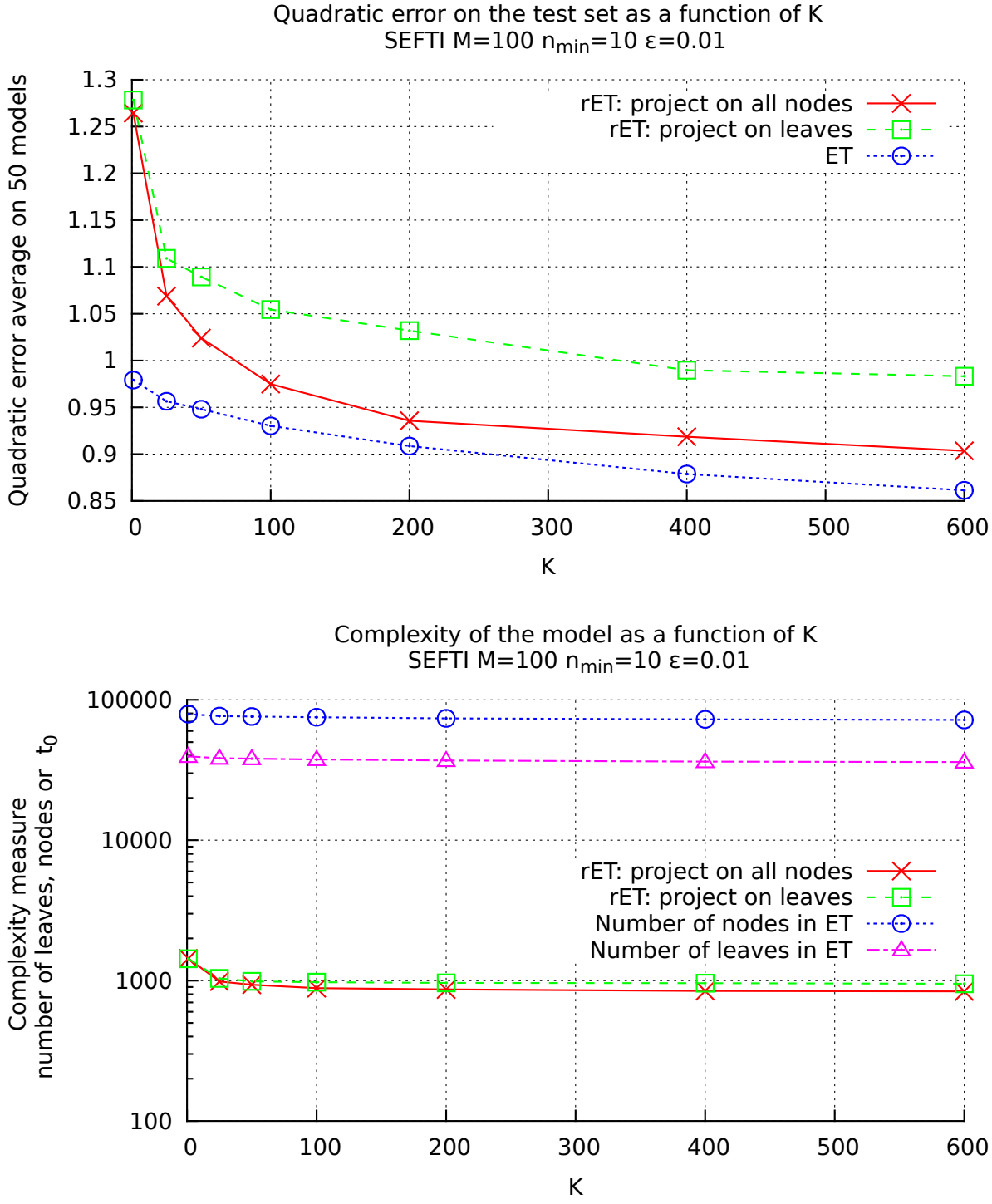

Figure 4.20 – Evolution of the quadratic error and the complexity as a function of K on the SEFTi dataset.

4.9 Effect of the step size ε in path algorithm

The parameter ε allows to control the size of a step made in the incremental forward stagewise regression algorithm. We are going to investigate in this section its effect on accuracy and complexity. In general, ε is as small as possible.

The graph of Figures 4.21 , 4.22, 4.23 show the evolution of the estimated risk and complexity as a function of ε on incremental forward stagewise regression and its combination with randomized trees for the each dataset. We refer to the figures for parameter values.

Impact on accuracy When we decrease ε, we are able to observe three phases on the curves of accuracy:

1. A diminution of ε *first* leads to an improvement of the performance: the incremental forward stagewise regression algorithm has the possibility more finely tune the weight associated to a feature, *e.g* see Figure 4.21 on Friedman1 or Figure 4.22 on Two-Norm.

2. Then when we further decrease ε, performance does not improve emphe.g. see Figure 4.21 on Friedman1 . It means that the algorithm has converged or is near convergence. The accuracy gain becomes negligible.

3. When we decrease ε even more (see *e.g.* Figure 4.22 on Two-Norm or Figure 4.23 on SEFTi), accuracy is degraded. Indeed, we begin to over-fit the data at the selection model stage, *i.e.* when we have to select one solution in the path.

Impact on complexity A decrease of ε leads to an increase of the complexity of the model (see Figures 4.21, 4.22 and 4.23). However it tends to converge: fewer nodes are added in the regularized model at each decrement of ε.

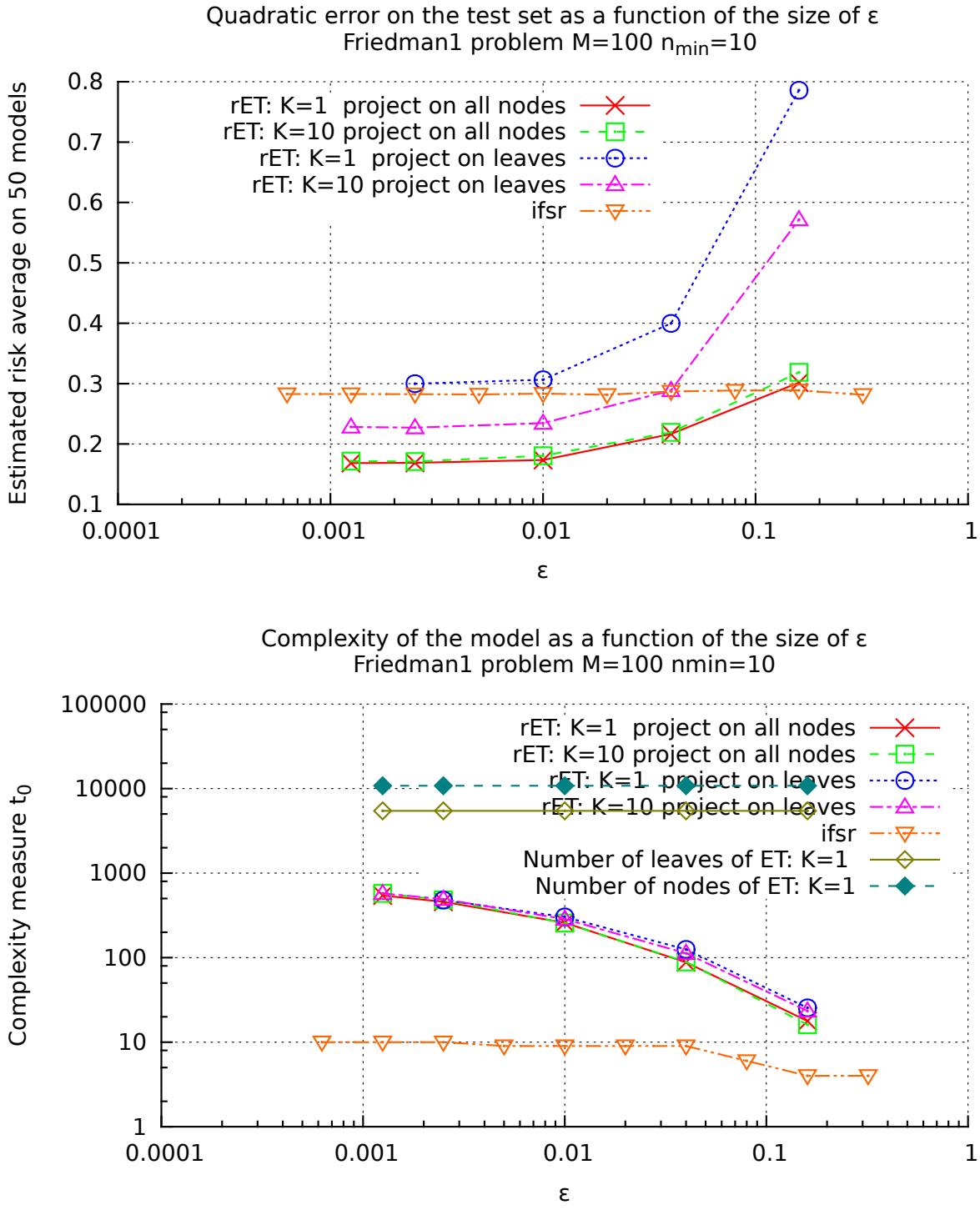

Figure 4.21 – Evolution of the quadratic error and the complexity of the model as a function of ε on the Friedman1 database.

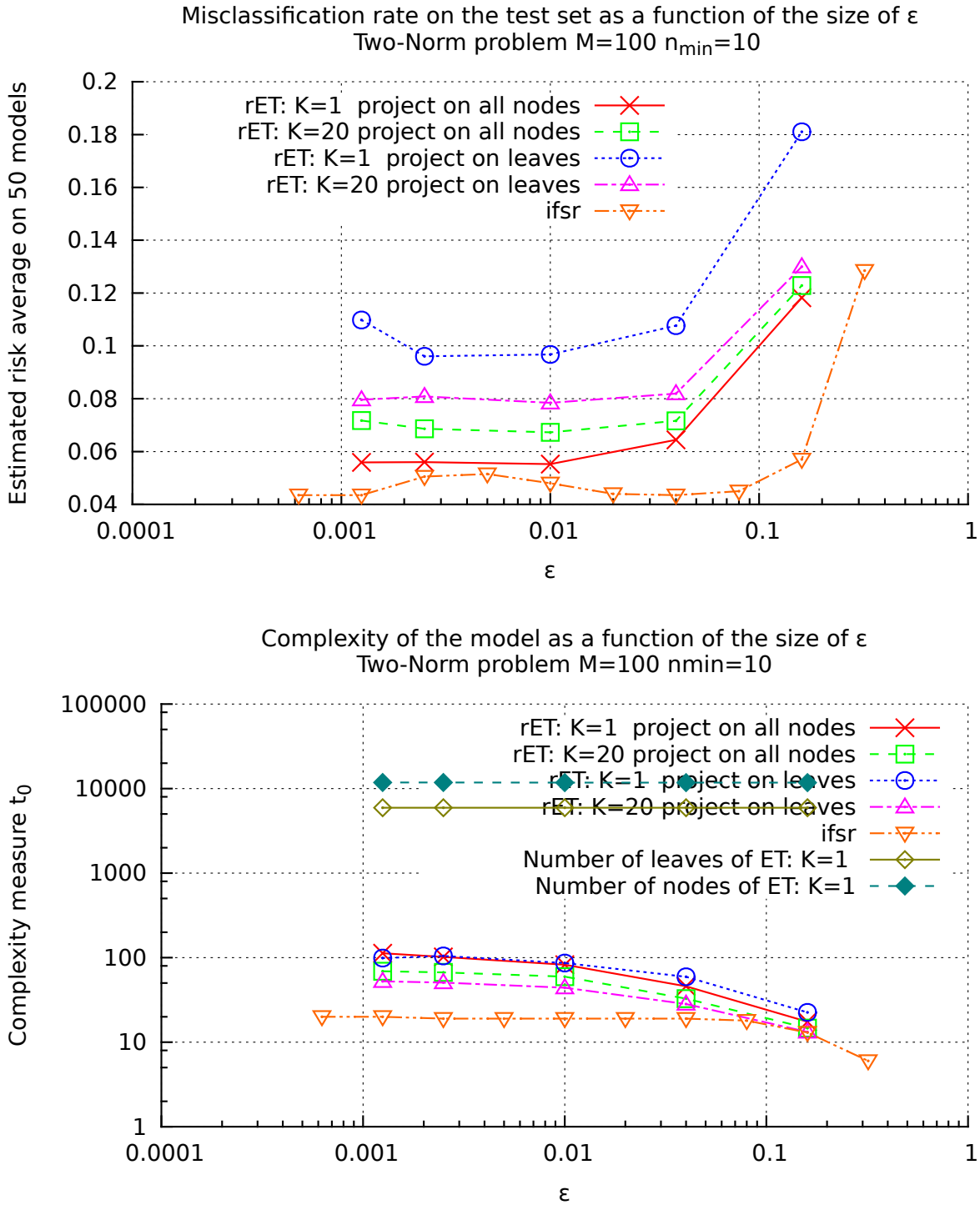

Figure 4.22 – Evolution of the misclassification rate and the complexity of the model as a function of ε on the Two-Norm database.

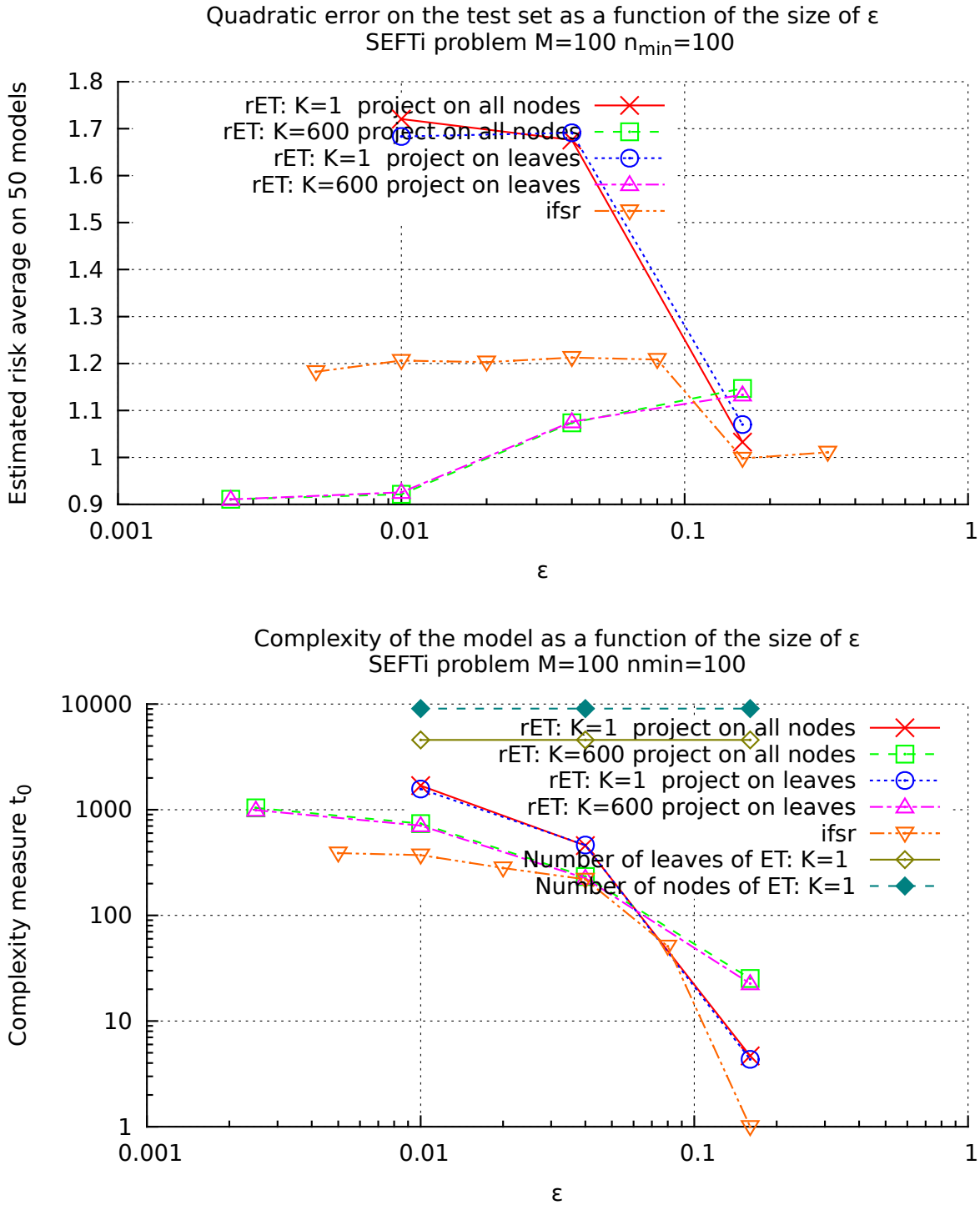

Figure 4.23 – Evolution of the quadratic error and the complexity of the model as a function of ε on the SEFTi database.

4.10 Effect of the learning set size on model complexity

The size of the ensemble M and the pre-pruning parameter n_{min} allow to control the complexity of the model and have a huge impact on the performance of extra trees. However the regularisation of an ensemble of randomized trees is relatively insensible to those parameters in term of complexity. There is still one parameter that may influence the complexity of the model and that we have not studied: the size of the learning set.

The graphs of Figures 4.24, 4.25 and 4.26 depict experiments where we quantify the evolution of the performance as a function of the size of the learning set for extra trees and its regularised version on each dataset. For the values of parameters, we refer the reader to the figures.

For the three datasets, the accuracy (see top of Figures 4.24, 4.25 and 4.26) and the complexity (see bottom of Figures 4.24, 4.25 and 4.26) of every algorithm increase with the size of the learning set. The complexity of the regularised model is directly related to the number of learning samples and not to the parameters of the algorithm such as M or n_{min}. The accuracy increase comes from a larger number of points in the input/output space.

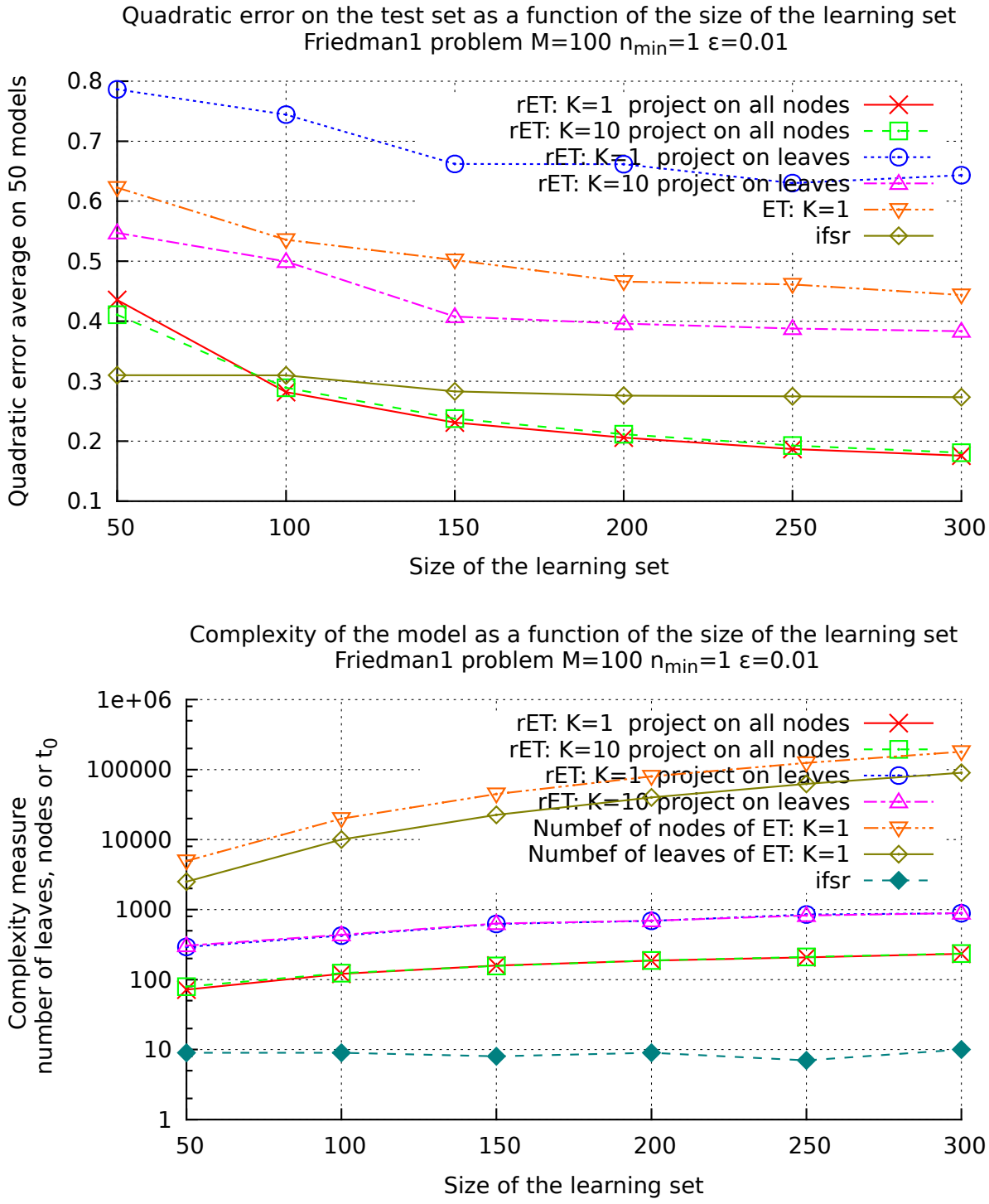

Figure 4.24 – Evolution of the quadratic error and the complexity of the model as a function of the size of the learning set on the Friedman1 problem.

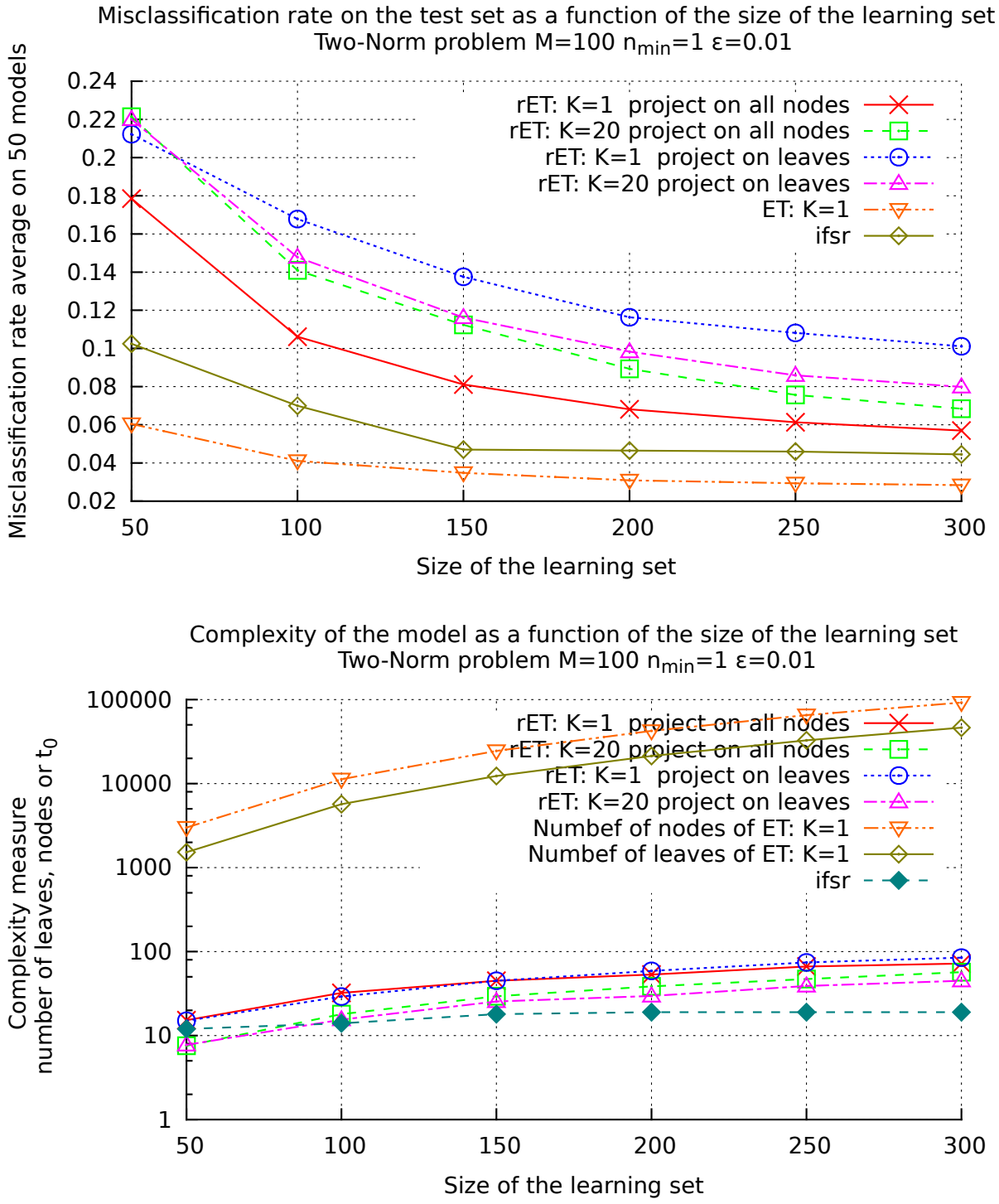

Figure 4.25 – Evolution of the misclassification rate and the complexity of the model as a function of the size of the learning set on the Two-Norm problem.

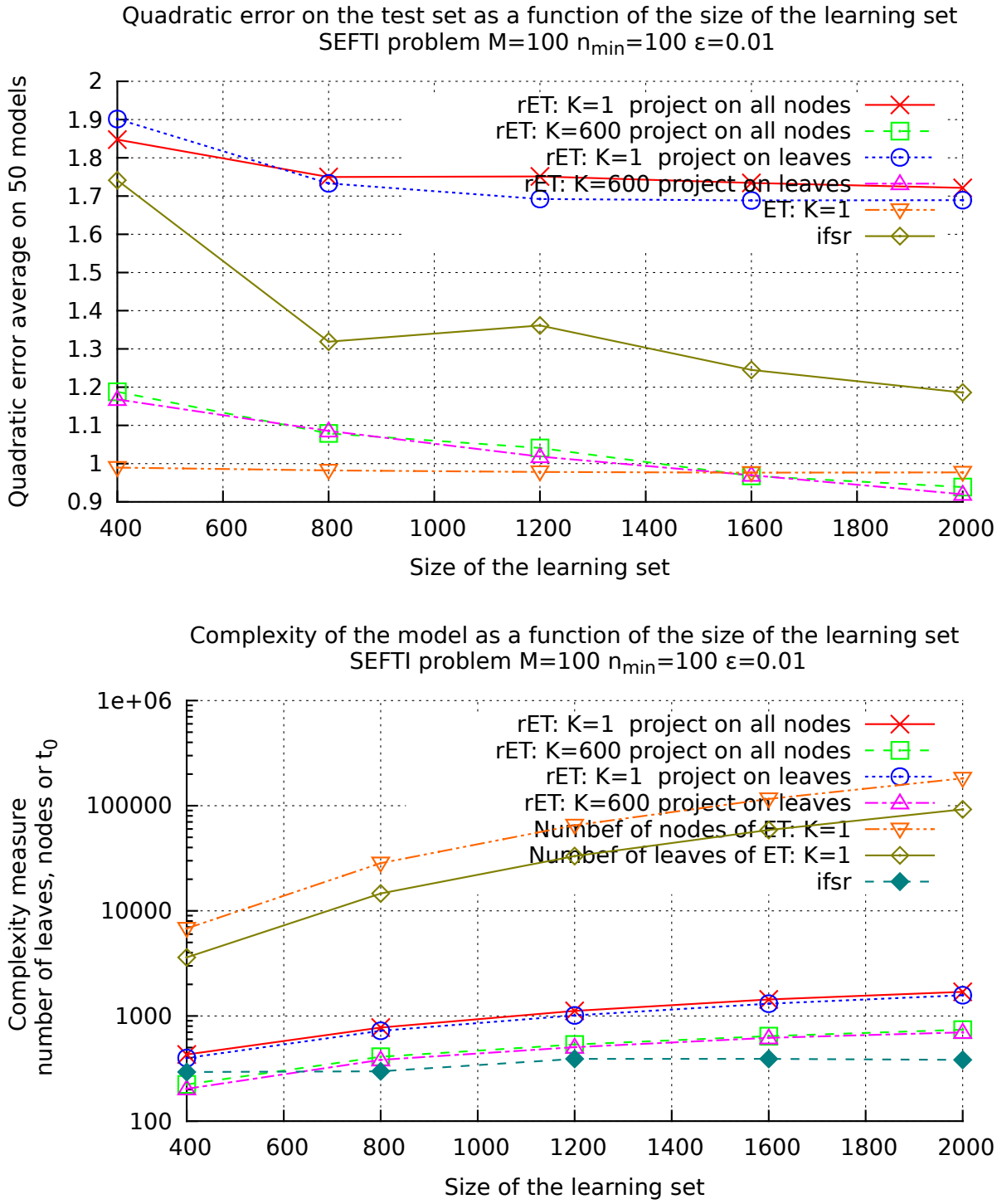

Figure 4.26 – Evolution of the quadratic error and the complexity of the model as a function of the size of the learning set on the SEFTi problem.

4.11 Overall conclusion of the experimental part

In this experimental part, we have studied the performance in terms of complexity and accuracy. Some of our results for the regularisation with $L1$-norm on ensembles of extra trees can be summarised as follow:

- A greater size of the ensemble M often improves accuracy by improving tree diversity.
- An appropriate value of K[1] allows to filter out irrelevant variables. However, it might degrade accuracy when we are able to build large ensembles of characteristic node functions as in the Friedman1 problem.
- Pre-pruning might not be necessary because not using it produces complex node characteristic functions. Furthermore, we are going to prune ensemble of randomized tree with the regularisation in $L1$-norm. Nevertheless, the algorithm seems relatively insensitive to pre-pruning if the height of a trees is high enough. Pre-pruning however speeds up greatly the learning process.
- Accuracy increases with small step size ε. The over-fitting problem could be solved by modifying the selection process *e.g.* with K-fold cross validation.
- Projection on all nodes leads to better results than projection only on leaves on all experiments.
- The complexity of a regularized ensemble of extremely randomized depends mostly on the number of samples.

Finally, the regularisation in $L1$-norm of ensembles of randomized trees produces compact and expressive models. Accuracy can be improved by the post-processing step. More research is still needed to understand the results obtained with the regularisation in $L1$-norm of totally randomized trees on the SEFTi problem.

[1] K is the parameter that controls the number of randomly selected attributes at the learning stage.

Chapter 5

Conclusions and future works

5.1 Conclusions

Tree-based ensemble methods, such as random forests and extremely randomized trees, are methods of choice for handling high dimensional problems. One important drawback of these methods however is the complexity, i.e. the large number and the size of trees, of the models to achieve good performance. In this master thesis, several research directions have been proposed in order to address this problem and prune the ensemble of randomized trees. The following has been developed:

- *Feature selection techniques applied on node characteristic functions*: a set of nodes are selected in an ensemble of randomized trees. These nodes will define a new space. Each node represents a binary function which is equal to one if the sample reaches that node and zero otherwise. The resulting feature space is *sparse* and contains only binary values.

  Once the database has been projected on this new space, a feature selection technique such as regularisation with $L1$-norm is applied on the node feature space. The ensemble of randomized trees is pruned and re-weighted with new coefficients.

Empirical evaluation has been undertaken on several databases with the combination of extremely randomized trees and monotone LASSO. From the experimental evaluation, we show that:

- The proposed approach allows to prune drastically the ensemble of randomized trees and re-weigh the decision trees consistently.
- The accuracy of the pruned ensemble is similar or better despite the naive model selection procedure. We conjecture that with a less biased selection method, we would always obtain better or equal results.
- On several databases, we have shown that the performance of the re-weighted and pruned ensembles of randomized trees is insensitive to pre-pruning to some extend with projection on nodes.

- The use of every node characteristic functions performs better than every leaf characteristic functions.
- The complexity of the pruned model seems to depend mainly on the size of the learning set.

The regularisation with $L1$-norm of the space generated by node characteristic functions of an ensemble of randomized tree leads to compact and expressive models. The accuracy can be improved over the original ensemble of randomized trees.

5.2 Future works

A more comprehensive set of experiments has to be undertaken in order to propose a set of default parameters to our approach/algorithms. Moreover, it would be interesting to compare these parameters with the default ones of individual algorithm. In parallel, a detailed bias/variance analysis might be carried out to better understand the effect of $L1$-norm regularization in terms of bias and variance.

In the developed approach, we have already obtained good results with ensemble of randomized trees. Indeed our approach consists first in creating many complementary and highly redundant sub-domains. The feature selection technique then selects and re-weighs only a few of them. An interesting research direction would be to generate fewer and more diverse sub-domains without enforcing redundancy. A first step would be to generate random tree branches to break the symmetry. And in a second step, we could investigate several search strategy to construct wisely these branches.

Techniques have been developed to prune drastically the ensemble of randomized tree. We do not indeed control the size of the generated model. For applications with tight hardware constraints, we should be able to define a maximal complexity of the generated and pruned model, and stay within this limit.

Two research directions remain to be evaluated:

- *Exploiting the kernel view of tree-based ensembles with SVM*: an ensemble of trees can be used to define a kernel between objects that measure the number of times these two objects reach the same leaf in a tree and is weighted by the specificity of the reached leaves. Using this kernel within support vector machines will help identifying the objects from the training sample that are really useful (the so-called support vectors). The ensemble of trees could then be pruned by keeping only paths traversed by these objects.
- *Random pruning*: nodes of the ensemble can be randomly pruned. Due to the redundancy introduce at the learning step by randomisation, the generated forest is robust to some extent at random node removal.

Another perspective would be to consider our approach under the light of compressed sensing[1] which is a method in information theory to compress signal by exploiting random projections and

[1]Compressed sensing is a technique to solve high dimensional undetermined linear systems whose solution is sparse. Compressed Sensing (CS) (see [Candès and Wakin, 2008] and [Donoho, 2006]) exploits random projections

then the signal can be retrieved by setting an optimisation problem in L1-norm. Ensembles of randomized trees can be interpreted as projecting the data on a randomly generated subspace (see [Dasgupta and Freund, 2009]).

in order to reduce measurement bandwidth for compressible data. Suppose that some data vector $x \in \mathbb{R}^n$ may be represented (almost) exactly with only s non zero components in some orthonormal basis Ψ, and that instead of measuring the individual n components of x or the s non-zero components of Φx, we measure some $m \leq n$ components of Φx, where Φ is another orthonormal basis. CS theory shows that x can be reconstructed (almost) exactly from the m measurements (by minimizing the number of non-zero components in Φ) as long as the number m of measurements is larger than $C\mu^2_{\Phi,\Psi} s \log n$, where C is a constant (typically $C \leq 10$) and where $\mu^2_{\Phi,\Psi} = \sqrt{n} \max \{ \left| \psi_i^T \phi_j \right| : 1 \leq i, j \leq n \}$ measures the coherence of the two bases Φ and Ψ ($\mu \in [1; \sqrt{n}]$). This approach leads to impressive results in information theory and data compression.

Appendix A

Bias/variance decomposition with a quadratic loss

The following bias/variance decomposition is taken from [Geman et al., 1992] Sections 2.1 and 3.1. In regression with a quadratic loss, the average generalisation error over randomly learning set *LS* of a given size n drawn in a distribution *i.i.d.* in a conjoint law $P(x,y)$:

$$E_{LS}\{Err\} = E_{\mathscr{X}}\{E_{LS}\{E_{\mathscr{Y}|\mathscr{X}}\{(\hat{f}(x)-y)^2\}\}\}. \tag{A.1}$$

We can decompose $E_{LS}\{E_{\mathscr{Y}|\mathscr{X}}\{(\hat{f}(x)-y)^2\}\}$:

$$\begin{aligned}
&E_{LS}\{E_{\mathscr{Y}|\mathscr{X}}\{(\hat{f}(x)-y)^2\}\} \\
=&E_{LS}\{E_{\mathscr{Y}|\mathscr{X}}\{(\hat{f}(x)-E_{\mathscr{Y}|\mathscr{X}}\{y\}+E_{\mathscr{Y}|\mathscr{X}}\{y\}-y)^2\}\} && \text{(A.2)}\\
=&E_{LS}\{E_{\mathscr{Y}|\mathscr{X}}\{(\hat{f}(x)-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2\}\}+E_{LS}\{(E_{\mathscr{Y}|\mathscr{X}}\{(E_{\mathscr{Y}|\mathscr{X}}\{y\}-y)^2\}\} \\
&+E_{LS}\{E_{\mathscr{Y}|\mathscr{X}}\{2(\hat{f}(x)-E_{\mathscr{Y}|\mathscr{X}}\{y\})(E_{\mathscr{Y}|\mathscr{X}}\{E_{\mathscr{Y}|\mathscr{X}}\{y\}-y)\}\} && \text{(A.3)}\\
=&E_{LS}\{(\hat{f}(x)-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2\}+E_{\mathscr{Y}|\mathscr{X}}\{(E_{\mathscr{Y}|\mathscr{X}}\{y\}-y)^2\} \\
&+E_{LS}\{2\,(\hat{f}(x)-E_{\mathscr{Y}|\mathscr{X}}\{y\})(E_{\mathscr{Y}|\mathscr{X}}\{y\}-E_{\mathscr{Y}|\mathscr{X}}\{y\})\} && \text{(A.4)}\\
=&E_{LS}\{(\hat{f}(x)-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2\}+E_{\mathscr{Y}|\mathscr{X}}\{(E_{\mathscr{Y}|\mathscr{X}}\{y\}-y)^2\}. && \text{(A.5)}
\end{aligned}$$

The term $E_{LS}\{(\hat{f}(x)-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2\}$ can be further decomposed:

$$\begin{aligned}
&E_{LS}\{(\hat{f}(x)-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2\} \\
=&E_{LS}\{(\hat{f}(x)-E_{LS}\{\hat{f}(x)\}+E_{LS}\{\hat{f}(x)\}-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2\} && \text{(A.6)}\\
=&E_{LS}\{(\hat{f}(x)-E_{LS}\{\hat{f}(x)\})^2\}+E_{LS}\{(E_{LS}\{\hat{f}(x)\}-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2\} \\
&+E_{LS}\{2(\hat{f}(x)-E_{LS}\{\hat{f}(x)\})(E_{LS}\{\hat{f}(x)\}-E_{\mathscr{Y}|\mathscr{X}}\{y\})\} && \text{(A.7)}\\
=&E_{LS}\{(\hat{f}(x)-E_{LS}\{\hat{f}(x)\})^2\}+(E_{LS}\{\hat{f}(x)\}-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2 \\
&+2(E_{LS}\{\hat{f}(x)\}-E_{LS}\{\hat{f}(x)\})(E_{LS}\{\hat{f}(x)\}-E_{\mathscr{Y}|\mathscr{X}}\{y\}) && \text{(A.8)}\\
=&E_{LS}\{(\hat{f}(x)-E_{LS}\{\hat{f}(x)\})^2\}+(E_{LS}\{\hat{f}(x)\}-E_{\mathscr{Y}|\mathscr{X}}\{y\})^2. && \text{(A.9)}
\end{aligned}$$

Then, we set:

$$var_{LS}(\hat{f}(x))\} = E_{LS}\{(\hat{f}(x) - E_{LS}\{\hat{f}(x)\})^2\} \tag{A.10}$$

$$bias^2(x) = (E_{LS}\{\hat{f}(x)\} - E_{\mathscr{Y}|\mathscr{X}}\{y\})^2 \tag{A.11}$$

$$var_{\mathscr{Y}|\mathscr{X}}(y) = E_{\mathscr{Y}|\mathscr{X}}\{(E_{\mathscr{Y}|\mathscr{X}}\{y\} - y)^2\}. \tag{A.12}$$

Finally, we obtain the bias/variance decomposition:

$$E_{LS}\{Err\} = E_{\mathscr{X}}\{var_{LS}(\hat{f}(x)) + bias^2(x) + var_{\mathscr{Y}|\mathscr{X}}(y)\} \tag{A.13}$$

$$= E_{\mathscr{X}}\{var_{LS}(\hat{f}(x))\} + E_{\mathscr{X}}\{bias^2(x)\} + E_{\mathscr{X}}\{var_{\mathscr{Y}|\mathscr{X}}(y)\}. \tag{A.14}$$

Appendix B

Estimation of decision tree complexity

The complexity of a decision tree can be measured by the number of leaves, the number of nodes or the number of splitting nodes. In order to fix the ideas, we can compute the number of nodes in function of n_{min}, the minimal number of samples in order to split a node. If we base our computation only on this criterion, we have two different cases:

- *Median splitting*: at each node, the sample set is split perfectly in two sets.
- *Degenerate splitting*: at each node, the split of n samples is degenerated and leads to a set of 1 sample and an other one of $n-1$ samples. The number of nodes in this case is given by $2(n-n_{min})+1$.

In regression, the two presented splitting give best and worst bound on complexity of a decision tree. In classification, it is a pessimistic approximation of the number of nodes, because we stop splitting when every sample are of the same class. The distinction between both splitting cases with 100 learning samples is illustrated on Figure B.1. In order to obtain small decision trees, we should favour *median splitting*.

The complexity of a decision tree is indeed linear with the number of samples and can be largely reduced with a pre-pruning criterion.

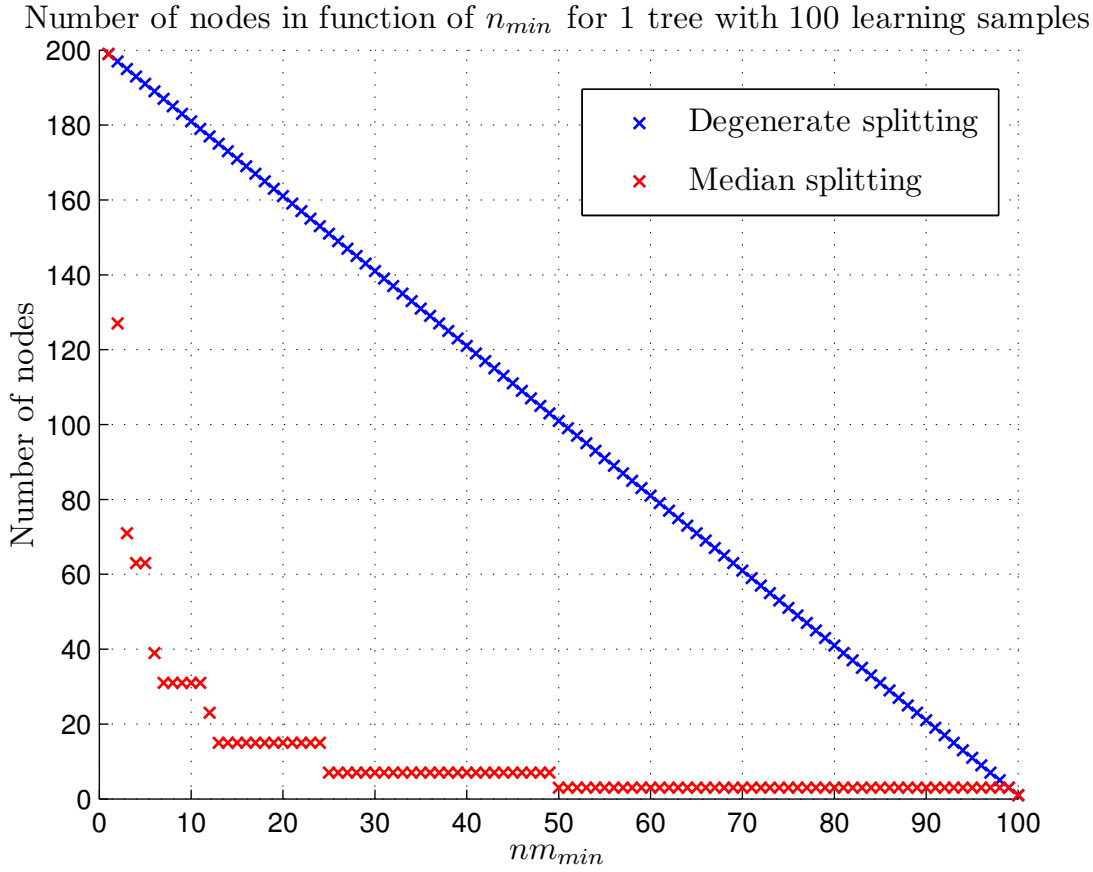

Figure B.1 – Evolution of minimal and maximal decision tree complexity as a function of n_{min}, the minimal number of samples in order to split a node, based only this criterion.

Bibliography

[AA&YA Intel, 2008] AA&YA Intel (2008). Manufacturing data: Semiconductor tool fault isolation. 26

[Bertsimas and Weismantel, 2005] Bertsimas, D. and Weismantel, R. (2005). *Optimization over integers*, volume 13. Dynamic Ideas. 7

[Bingham and Mannila, 2001] Bingham, E. and Mannila, H. (2001). Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM. 21

[Blum, 2006] Blum, A. (2006). Random projection, margins, kernels, and feature-selection. *Subspace, Latent Structure and Feature Selection*, pages 52–68. 21

[Boost, 2011] Boost (2011). Boost C++ Libraries provides free peer-reviewed portable C++ source libraries. `http://www.boost.org`. 2

[Breiman, 1996a] Breiman, L. (1996a). Bagging predictors. *Machine learning*, 24(2):123–140. 11

[Breiman, 1996b] Breiman, L. (1996b). Bias, variance, and arcing classifiers. *STATISTICS*. 26

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32. 11, 72

[Candès and Wakin, 2008] Candès, E. and Wakin, M. (2008). An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21–30. 64

[Chapman et al., 2007] Chapman, B., Jost, G., and Pas, R. v. d. (2007). *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press. 2

[Choi, 2002] Choi, J. (2002). A rule-based expert system using an interactive question-and-answer sequence. *University Consortium for Geographic Information Science (UCGIS) Summer Assembly Graduate Papers, Athens, Georgia*. iv, 8

[Dasgupta and Freund, 2009] Dasgupta, S. and Freund, Y. (2009). Random projection trees for vector quantization. *Information Theory, IEEE Transactions on*, 55(7):3229–3242. 65

[Dietterich, 2000] Dietterich, T. (2000). Ensemble methods in machine learning. *Multiple classifier systems*, pages 1–15. 11

[Donoho, 2006] Donoho, D. (2006). Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306. 64

[Efron et al., 2004] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499. 15

[Esposito et al., 1997] Esposito, F., Malerba, D., Semeraro, G., and Kay, J. (1997). A comparative analysis of methods for pruning decision trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(5):476–491. 10

[Freund and Schapire, 1995] Freund, Y. and Schapire, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer. 11

[Friedman, 1991] Friedman, J. (1991). Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67. 26

[Friedman, 2008] Friedman, J. (2008). Fast sparse regression and classification. *URL http://wwwstat. stanford. edu/jhf/ftp/GPSpaper. pdf*. 2, 15

[Friedman et al., 2007] Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332. 15

[Friedman and Popescu, 2008] Friedman, J. and Popescu, B. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954. 19

[Friedman et al., 2009] Friedman, J. H., Hastie, T., and Tibshirani, R. (2009). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22. 15

[Geman et al., 1992] Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58. 66

[Geurts, 2002] Geurts, P. (2002). Contributions to decision tree induction: bias/variance tradeoff and time series classification. *Liege, Belgium: University of Liege*. 11

[Geurts, 2009] Geurts, P. (2009). An introduction to machine learning. Slide from Ulg Applied Inductive Learning lectures. iv, 10

[Geurts, 2010] Geurts, P. (2010). Bias vs variance decomposition for regression and classification. *Data Mining and Knowledge Discovery Handbook*, pages 733–746. 7

[Geurts, 2011] Geurts, P. (2011). Regression tree package: contains a c implementation of (multiple output) regression trees and various ensemble methods thereof, including extremely randomized trees [Geurts et al., 2006a], Random Forests [Breiman, 2001], and multiple additive regression trees (Friedman et al.). The c code is not standalone but a matlab interface is also included. `http://www.montefiore.ulg.ac.be/~geurts/Software.html`. 2

[Geurts et al., 2006a] Geurts, P., Ernst, D., and Wehenkel, L. (2006a). Extremely randomized trees. *Machine Learning*, 36(1):3–42. 2, 11, 13, 72

[Geurts et al., 2006b] Geurts, P., Wehenkel, L., and D'Alché-Buc, F. (2006b). Kernelizing the output of tree-based methods. In *Proceedings of the 23rd international conference on Machine learning*, pages 345–352. Acm. 20

[Guyon, 2006] Guyon, I. (2006). *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*, volume 207. Springer, 1 edition. 18

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182. 18

[Hastie et al., 2007] Hastie, T., Taylor, J., Tibshirani, R., and Walther, G. (2007). Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1. iv, 16, 17

[Hastie et al., 2003] Hastie, T., Tibshirani, R., and Friedman, J. H. (2003). *The Elements of Statistical Learning*. Springer, corrected edition. 10, 15

[Hoerl and Kennard, 1970] Hoerl, A. and Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67. 15

[Hyafil and Rivest, 1976] Hyafil, L. and Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5:15–17. 7

[Jiang, 2011] Jiang, H. (2011). Glmnet for Matlab. Lasso (L1) and elastic-net regularized generalized linear models. `http://www-stat.stanford.edu/~tibs/glmnet-matlab/`. 2

[Kohavi, 1995] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145. Citeseer. 6

[Meinshausen, 2009] Meinshausen, N. (2009). Node harvest: simple and interpretable regression and classification. *Arxiv preprint arXiv*, 910. 19

[Mingers, 1989a] Mingers, J. (1989a). An empirical comparison of selection measures for decision-tree induction. *Machine learning*, 3(4):319–342. 9

[Mingers, 1989b] Mingers, J. (1989b). An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243. 10

[Pisetta et al., 2010] Pisetta, V., Jouve, P., and Zighed, D. (2010). Learning with ensembles of randomized trees: new insights. *Machine Learning and Knowledge Discovery in Databases*, pages 67–82. 19

[Schapire, 2003] Schapire, R. (2003). The boosting approach to machine learning: An overview. *LECTURE NOTES IN STATISTICS-NEW YORK-SPRINGER VERLAG-*, pages 149–172. 11

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. the MIT Press. 11

[Tibshirani, 1996] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288. 15, 22