Predicting and Interpreting Business Failures with Supervised Information Geometric Algorithms

Caroline Ventura* Fred Célimène* Richard Nock* and Frank Nielsen†

Abstract

Business failure belongs to the most investigated topics in the business literature (Huang *et al.* (2008)). A plethora of works have addressed the problem using conventional statistical or machine learning techniques that are known to suffer from distributional assumptions, representational bias, statistical inconsistencies or over-fitting in generalization, and therefore bring results that have to be read with most caution for risk-free use (Crutzen and Van Caillie (2008); Ravi Kumar and Ravi (2007)). In this paper, we analyze corporate tax return from a thousand French companies using a powerful blend of wavelet-based data modeling and new classification techniques born from information geometry that do not suffer any of the former drawbacks (Nock and Nielsen (2009)). It is quite remarkable that particular cases of these techniques have been known for decades, yet researchers in the business failure field claim that they have remained under-exploited (Ravi Kumar and Ravi (2007)). Our results are a clear-cut advocacy for the usefulness of these techniques in this field, as they display the ability (i) to produce models accessible even to green users, whose interpretation sheds light on the statics and dynamics of business failure, (ii) to outperform in generalization classification algorithms traditionally used in the business failure field by orders of magnitude.

JEL Classification Numbers: C38 — Classification Methods; C53 — Forecasting Models.

Keywords: Business Failure; Boosting; Company Tax Return; Wavelet Expansion.

* CEREGMIA, Université des Antilles et de la Guyane, campus de Schoelcher, B.P. 7209, F-97275 Schoelcher Cédex, France.
E-mail: `venturac@yahoo.com`, {`fcelimen,rnock`}`@martinique.univ-ag.fr`.

* Sony Computer Science Laboratories, Inc., 3-14-13 Higashi Gotanda. Shinagawa-Ku. Tokyo 141-0022, Japan.
E-mail: `nielsen@csl.sony.co.jp`.

1. Introduction

"*We are drowning in information... and starving for knowledge*" : one would expect this sentence to be related in some way to the boom of the internet, but it is not the case. In fact, it dates back to 1985 — the prehistoric ages of personal computing — and is commonly attributed to a former Yale librarian. Today, this sentence would certainly fit to any prominent domain fueled with sizable quantities of data to analyze. Financial distress and failure prediction is certainly one such domain: there has been a treamount of works carried out so far to predict and explain business failure, with most of the techniques coming from statistics and intelligent systems tested to far. Ravi Kumar and Ravi (2007) divide failure prediction techniques into two sets: statistical and intelligent techniques. Statistical techniques contain *e.g.* logistic regression, factorial analyses. Intelligent techniques contain neural networks, nearest neighbor classifiers, operations research methods, decision tree induction methods, etc. . The conclusions of these works is that each of the standard algorithms have pros and cons that makes it both interesting and quite disappointing, as if each of them was representing a new way for the Sisyphus of failure prediction to roll his boulder up the hill, but with none he may succeed in making it to the peak. Speaking of failure prediction, the major problem for Sisyphus is that the hill is continuously growing with huge amounts of data gradually complicating the path upwards. A solution envisioned in various papers relies on combining techniques in so-called *hybrid approaches*, trying to keep the pros of all while leaving the cons of most of them (Huang *et al.* (2008); Li and Sun (2011); Ravi Kumar and Ravi (2007); Ravisankar *et al.* (2010)). Numerous desirable properties exist, falling into different categories: generalization abilities, theoretical approximation of the optimum, interpretability of the results for green users (*e.g.* if-then rules, ranking and/or filtering of variables), computational demand, parameters tuning, data requirements, resistance against over-fitting, etc. . While it is reasonable to imagine that powerful combinations of heterogeneous baseline algorithms indeed exist, they face the pitfall of getting desirable features while deteriorating further potential drawbacks, such as *over-fitting*, that is, an improvement of the system on the data at hand at the expense of its performances on unseen data.

But *birds of a feather flock together*: the first contribution of our paper is an attempt to display the fact that efficient hybrid systems may be developed from single parts with different features each *but* who are known to belong to the same theoretical breed. We do this using recent results (Nock and Nielsen (2009)) that unify popular classification algorithms under the same theory, showing for example that decision tree learning algorithms like CART, C45 (Hastie *et al.* (2002)) are essentially the *same* algorithms as AdaBoost (Freund and Schapire (1997); Hastie *et al.* (2002)); in particular, all these algorithms may be viewed as information geometric algorithms that perform exactly the same kind of search using the same generic toolbox. It is quite noticeable that AdaBoosting for failure prediction is still in embryonic stages in the business failure prediction literature, with experimental results that are very recent and still scratching the surface of the method's potential (Sun *et al.* (2011)). In this paper, our ambition is to drill down into this potential to start unveiling the results of a methodology claimed elsewhere to have been among the most important advances in classification in the late nineties (Friedman *et al.* (2000)). Our approach also contributes to fill the lack of use in business failure prediction of classification methods that are known to be extremely powerful, including algorithms that, to the best of our knowledge, have never been used in the field (Freund and Mason (1999)). Other algorithms we use are extremely popular in data mining, yet they have received quite a reduced coverage in failure prediction. For example, various surveys have recently stressed the need to drill down into the potential of decision tree induction for failure prediction (Balcaen and Ooghe (2006); Gepp *et al.* (2010); Ravi Kumar and Ravi (2007)). According to Ravi Kumar and Ravi (2007), "decision trees [...] are not employed as much as they deserve". There is a pragmatic explanation to the fact that decision trees have been so far less used than other classifiers like neural nets or nearest neighbor classifiers: a significant part of the learning methodology inherent to these latter classifiers has been known for quite a long time (*e.g.* the sixties for nearest neighbor classifiers (Cover and Hart (1967))), while the theory which has explained and developed further the efficient use of decision trees is much more recent (Freund and Mason (1999); Hastie *et al.* (2002); Kearns and Mansour (1999); Nock and Nielsen (2009)) — and thus has been out of reach of many works that could instead use neural nets or nearest neighbors at their "fullest potential". Decision trees, or many other classifiers whose induction may fall in the same theory (Nock and Nielsen (2008, 2009)), also offer visualization and interpretation capabilities that are far more convenient for the green user than *e.g.* neural networks, logistic regression, discriminant analysis, nearest neighbor classifiers, or even support vector machines (Sun *et al.* (2011)).

Our second contribution goes upstream in the failure prediction process. It has been remarked that business failure comprises a time dimension that has been so far significantly neglected (Balcaen and Ooghe (2006)). In particular, failure prediction methods should integrate the timely nature of data collected, but very few of them actually try to do so (Balcaen and Ooghe (2006)). Furthermore, in the smaller subset of methods that meet this constraint, most of them actually do *not* take into account trends in failure patterns, but rather rely on fixed-prediction horizon: for example, data at time minus one year is used to predict failure at the current time. In order to really capture trends over failure patterns at various times *and* scales, we perform wavelet expansion of each of our time series using Haar wavelets prior to learning or predicting. Our experiments clearly display that business failure is a timely process with clearly identifiable and interpretable patterns, remembering the seminal works on failure patterns by Laitinen (1993). The big difference with this work, however, is that timely failure patterns are not expert-based: they are *learnt* by the machine. Furthermore, the machine is able to learn automatically the *scale* on which these patterns must hold over time to represent a risk of business failure, and also learn *combinations* of timely patterns over different variables that, brought together, increase the risk of business failure.

Our third contribution is to compare, in this whole framework, the performances in prediction of the whole data, to those of the data in which we restrict the set of time series to those of financial ratios known to bring accurate information about failure risks (Laitinen (1993)), or to subsets of time series in raw data that should be the most relevant to the task at hand (Crutzen and Van Caillie (2008)). The data we use is a set of a thousand French businesses for which we possess ten variables (time series) of the yearly company tax return, plus five time series for the annual financial ratios of Laitinen (1993), plus other variables (year of creation, legal form and the business section code from the French nomenclature). Each time series has time stamps in between 2003 and 2008.

The remaining of this paper is organized as follows. Section 2 presents the global prediction task, summarizes the data and presents wavelet expansion; Section 3 presents the various information geometric algorithms we consider; Section 4 details the experiments carried out, and a last Section (5) concludes the paper with avenues for further research. In order not to laden the

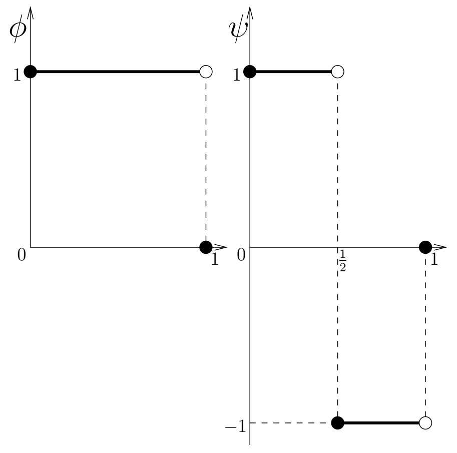

Figure 1: Haar scaling function $\phi(t)$ (left) and Haar mother wavelet function $\psi(t)$ (right).

paper's body, we have postponed to an Appendix (Section A) the complete description of data.

2. Dataset $\mathcal{S}$ and its wavelet expansions

We first begin by a slight abstraction of the task at hand, to fix some notations. We are given a set $\mathcal{S}$ of examples, where each example is an ordered pair (o, c), where o is an *observation* and c is a *class* (or *label*). The class takes on two values, -1 or 1. 1 refers to the existence of recovery proceedings or compulsory liquidation for the firm (hereafter simply called business failures), which basically means it disappears, and -1 refers to the absence of these two kinds of problems. Our basic objective is to build *classifiers*, that is, functions $g : \mathcal{O} \rightarrow \mathbb{R}$ taking as entry an observation $o \in \mathcal{O}$ and returning a real value $g(o)$ whose sign $\sigma(g(o)) \in \{-1, 1\}$ gives a prediction of risk failure for the firm matching observation o. Obviously, we want this prediction to be as accurate as possible, that is, we want $\sigma(g(o))$ to match the actual risk which faces the firm. There is another information, extremely useful, which gives $g(o)$: its absolute value, $|g(o)|$ may be translated as a *confidence* in the label predicted: the larger it is, the more confident we can be in the label predicted. Before drilling down into the accuracy requirement, we present the way we partially recode observations with wavelet expansions. This new coding is aimed at providing the algorithm which learns the classifiers the timely patterns of variation for variables, thus directly putting in data *prior* to learning a more accurate standpoint on time dimension, a feature of the utmost importance for the problem at hand (Balcaen and Ooghe (2006)).

Each observation comes first as a 18-uplet mixing both symbolic and numeric data (Section A). Notice the two specific subsets of time series, targeted to failure risk prediction: financial ratios (Laitinen (1993): ROI, QRA, TCF, SCA, TFD), and *blinkers* (CF, DX, DY). All but three of the uplet's coordinates are time series. Raw coding of time series does not capture their variations. For this reason, we preprocess each of these time series to code them in a wavelet domain (Chui (1992)), which makes is easy to capture these variations at multiple scales.

The wavelets we use are particularly convenient for histogram-based representation of regular time series like ours: Haar wavelets. We now present these wavelets and the way we process each of the time series of our dataset $\mathcal{S}$ to obtain their coordinates in the wavelet domain. Without loss of generality and for the sake of simplicity, assume a discrete time series $h(t)$ properly scaled so that it fits a regular piecewise constant function (a "histogram") in the interval $[0, 1]$, as displayed in Figure 2 (left) for an example with four time stamps. Assume furthermore that $h(t)$ contains a number of points which is a power of 2 — otherwise, replace $h(t)$ by an adequate interpolation which meets this constraint. The histogram corresponding to any such time series may be represented in a vector space $\mathcal{F}^j$ endowed with a particular scalar product $\langle ., . \rangle$, for $j \in \mathbb{N}$. To define $\mathcal{F}^j$, define the Haar scaling function ϕ and Haar mother wavelet function ψ, as follows:

$$\phi(t) \doteq \begin{cases} 1 & \text{if} \quad t \in [0, 1) \\ 0 & \text{otherwise} \end{cases} , \tag{2.1}$$

$$\psi(t) \doteq \begin{cases} 1 & \text{if} \quad t \in [0, 1/2) \\ -1 & \text{if} \quad t \in [1/2, 1) \\ 0 & \text{otherwise} \end{cases} . \tag{2.2}$$

We also define translations and dilations of ψ, ψ_i^j, as follows:

$$\psi_i^j(t) \doteq \sqrt{2^j}\psi(2^j t - i) , \tag{2.3}$$

for $i = 0, 1, ..., 2^j - 1$ which refers to the translation, while j refers to the dilation of the mother wavelet. Letting $\langle f, g \rangle \doteq \int_0^1 f(t)g(t)\mathrm{d}t$, one may check that the set of functions in (2.1) and (2.3) define an orthonormal basis of vector space $\mathcal{F}^j$ built from their linear span, as for example $\langle \psi_i^j, \psi_{i'}^{j'} \rangle$ is 1 iff $i = i', j = j'$ and zero otherwise. Assuming $h(t)$ contains 2^k time stamps, it may be exactly represented in $\mathcal{F}^{k-1}$, as follows:

$$h(t) = \langle h, \phi \rangle \phi + \sum_{j=0}^{k-1} \sum_{i=0}^{2^j - 1} \langle h, \psi_i^j \rangle \psi_i^j . \tag{2.4}$$

For example, $k = 2$ for the time series depicted in Figure 2, and we would obtain:

$$\begin{aligned} h(t) &= \langle h, \phi \rangle \phi + \langle h, \psi_0^0 \rangle \psi_0^0 + \langle h, \psi_0^1 \rangle \psi_0^1 + \langle h, \psi_1^1 \rangle \psi_1^1 \\ &= 6\phi + 2\psi(t) + \psi(2t) - \psi(2t - 1) , \end{aligned}$$

as shown in Figure 2. Hence, in the wavelet domain, the coordinates of $h(t)$ would be $(6, 2, 1, -1)$. The leftmost coordinate is thus the average value of the time series, while the others capture both the increasing/decreasing dynamics of the time series (sign) at various scales, and their magnitude (absolute values). For example, in Figure 2, the coefficient of $\psi_0^0 = \psi(t)$ captures the overall decrease of the complete time series, while the coefficients of ψ_1^0 and ψ_1^1 capture the local decrease (resp. increase) of the time series during the first half (resp. second half) of time stamps.

To summarize, each example $(o, c) \in \mathcal{S}$ undergoes wavelet expansion of each of its fifteen time series, ending up with an observation in $3 + (15 \times 4) = 63$ dimensions. This step is schematized in the purple part in Figure 3. To fix the ideas of the nomenclature we adopt, Figure 2 provides the names of wavelet expansion's coefficients for variable CF. The next step consists in learning classifiers (purple g in Figure 3) out of the newly crafted $\mathcal{S}$.

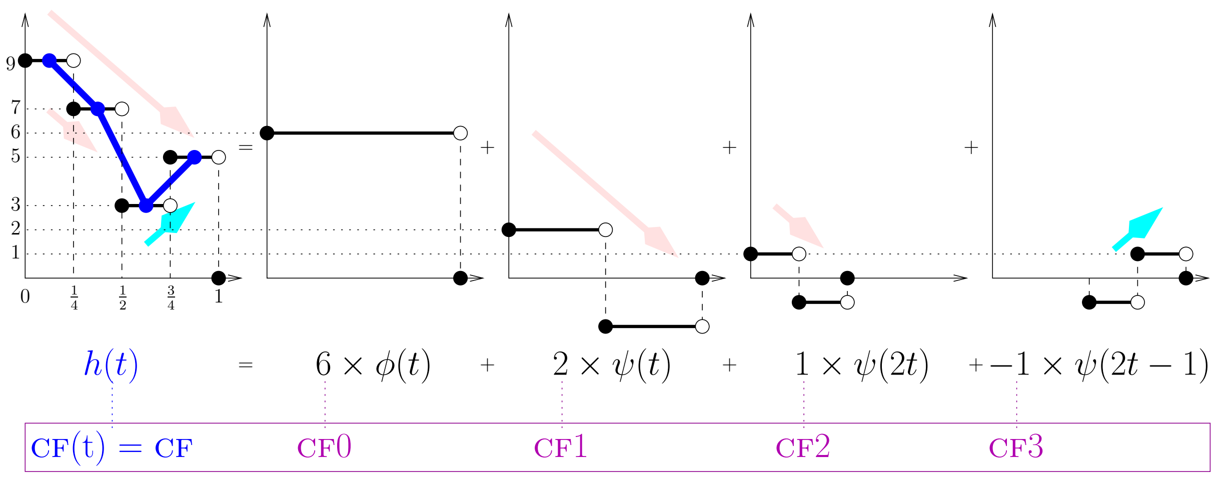
9
7
6
5
3
2
1
0
$\frac{1}{4}$
$\frac{1}{2}$
$\frac{3}{4}$
1
=
+
+
+
$h(t)$ = $6 \times \phi(t)$ + $2 \times \psi(t)$ + $1 \times \psi(2t)$ + $-1 \times \psi(2t-1)$
CF(t) = CF
CF0
CF1
CF2
CF3

Figure 2: A times series $h(t)$ with four regularly spaced time stamps and its matching histogram (left), and its decomposition as a linear combination of the scaling function $\phi(t)$ and mother wavelet function $\psi(t)$. Arrows depicted on h indicate perceptual trends either decreasing (pink) or increasing (cyan), at various scales. Notice that translations/dilations of $\psi(t)$ capture these trends at all scales. In the wavelet domain, the coordinates of $h(t)$ would be $(6, 2, 1, -1)$. We indicate in purple the wavelet expansion of variable CF (Section A) as coefficients CF0 (series average), CF1 (global variation of time series), CF2 (variation of time series over the first half of time stamps) and CF3 (variation over the second half of time stamps, see text for details).

3. Information geometric algorithms $\mathcal{A}$ to induce classifiers g

It is becoming increasingly remarked that business failure prediction has to be tackled with sophisticated techniques, elsewhere called *hybrid* (Ravi Kumar and Ravi (2007)). While we support the idea that pinches of sophistication may prove in handy to specialize algorithms and better cope with business failure, we wonder whether a cooked algorithm mixing all sorts of classification ingredients (fuzzy logic, neural networks, decision trees, rough sets, etc., Ravi Kumar and Ravi (2007)) will taste anything better than the bitter taste of over-fitting. *God made food, the devil the cooks*, once wrote James Joyce. We do believe that we have to be extremely careful when elaborating recipes tailored for failure prediction, in particular by mixing carefully chosen algorithms not impeding each others. A solution relies on picking such algorithms from the *same* family of techniques, that is, built from the same tools and frameworks, that are proven to be able to boost each others. Belonging to the same family of algorithms does not prevent them to be heterogeneous in the sense used for hybrid systems, that is, building various kinds of models.

This is *not* a conjecture, but a major lesson from *boosting* (Freund and Schapire (1997); Hastie *et al.* (2002)), a theory which shows how to make careful combinations of classifiers only slightly different from random to obtain complex classifiers with empirical risk (see below) arbitrarily close to zero. This task however is not trivial: famed algorithms like decision tree induction algorithms, born in the early eighties (Breiman *et al.* (1984)), unveiled their core mechanisms no less than fifteen years later (Kearns and Mansour (1999)), and their parenthood with specific boosting algorithms like AdaBoost is even more recent (Nock and Nielsen (2009)). Yet, these results open fascinating avenues for research on crafting powerful complex algorithms relying on modules theoretically compatible to boost their performances. We now briefly summarize

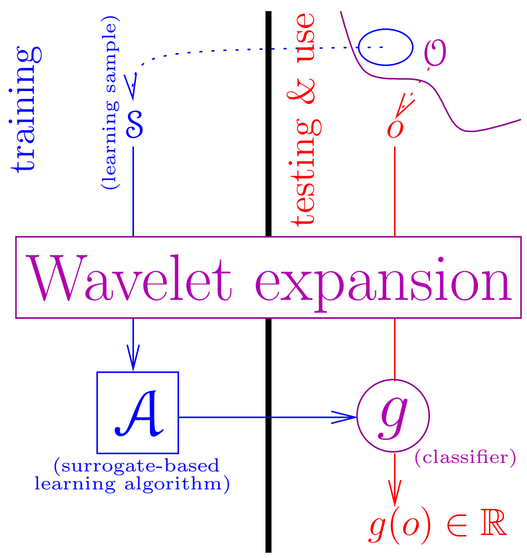

Figure 3: Classification as we carry out contains two essential phases: the induction (or training) of a classifier, shown in blue, and the testing (*i.e.* use) of this classifier on unseen observations, shown in red. Purple parts belong to both the training and testing phases.

the main steps in the discovery of the main bonds between the algorithms we use — to which belong these decision tree learning, AdaBoost and other boosting algorithms —, that define an homogeneous *information geometric* family of algorithms.

Over the last forty years, machine learning has undergone tremendous growth, both due to the boom in hardware technologies, and also to dramatic progresses from the theoretical standpoint, that have in part culminated in a theory elsewhere qualified as one of the most important developments in classification during the late XXth century (Friedman *et al.* (2000)): boosting. During the last decade, a significant part of this theory has been read and explained under an information geometric angle (Amari and Nagaoka (2000); Nock and Nielsen (2009)).

This theory basically deals with the *induction* of classifiers, that is, algorithms $\mathcal{A}$ that take as input $\mathcal{S}$ and return a classifier g (blue part in Figure 3). All the algorithms that we consider can be related in some way to this theory, and more particularly to a surprising and counter-intuitive feature of classification that this theory has emphasized, and so far partially explained. The accuracy of classifier g over $\mathcal{S}$ may be defined as one minus its *empirical* risk $\hat{\mathrm{E}}\textsc{rr}(g)$ (here, $\mathbf{E}$ is the mathematical expectation according to the density specified in index):

$$\hat{\mathrm{E}}\textsc{rr}(g) \quad \doteq \quad \mathbf{E}_{(o,c)\sim\mathcal{S}}[\sigma(g(o)) \neq c] \ . \tag{3.1}$$

The ideal objective is not to minimize the empirical risk, but rather the *true* risk $\mathrm{E}\textsc{rr}(g)$:

$$\mathrm{E}\textsc{rr}(g) \quad \doteq \quad \mathbf{E}_{o\sim\mathcal{O}}[\sigma(g(o)) \neq c(o)] \ , \tag{3.2}$$

where $c(o)$ is the actual class of observation o. Notice that (3.2) implicitly assumes a density over the set of all observations $\mathcal{O}$. It is not hard to see (and well known) that, provided we make the assumption that $\mathcal{S}$ is i.i.d. sampled according to this density, (3.1) becomes an estimator of (3.2), and the induction of g should probably be all the better as $\mathcal{S}$ gets larger. This indeed is true in general, *but* there is a significant caveat to inducing g on these sole bases: whenever the model g gets too complex, it gets more and more accurate on $\mathcal{S}$ at the expense of its goodness of fit on $\mathcal{O}$; this phenomenon is called *over-fitting* (Hastie *et al.* (2002)). This is not a contradiction with the fact that (3.1) still estimates (3.2), but rather that like the devil, *over-fitting is in the details* of the complex model.

It turns out that to better take control of and minimize (3.2), one should *not* rely on the minimization of its estimator (3.1), but on an convex upperbound with particular properties known as a *surrogate* (Nock and Nielsen (2009)). Clearly, this is counterintuitive, but it can be motivated by the fact that the surrogate is strictly convex and differentiable, properties that do not hold for the empirical risk (3.1). This goes for the computational advantages of surrogates. They have another interesting feature, as they define a rather non-conventional geometry (*i.e.* non Riemannian) over $\mathbb{R}^{\mathrm{card}(\mathcal{S})}$ which makes it possible to view classification as a geometric problem, and easily design new algorithms that possess the same desirable features, *and* that are "compatible" to be combined with some hybrid flavor (Ravi Kumar and Ravi (2007)) while bringing extremely strong guarantees. The main advantage is that the "elementary" modules may build classifiers with different syntaxes (*e.g.* decision trees, linear separators, etc.), but the algorithms that induce them work on exactly the same principles, gradually minimizing a particular surrogate of the empirical risk. The algorithms we use share a second common point: all start by inducing a large model which is later on pruned to yield a smaller final classifier. Roughly speaking, the pruning stage is aimed at reducing the true risk by reducing further the risk of over-fitting (Hastie *et al.* (2002)). We chose classification algorithms working on the same high-level schemata to reduce the *learning biases* and make it possible to compare classifiers obtained without caring too much about the algorithms used to induce the classifiers (Nock and Nielsen (2008)): indeed, differences in accuracies between models should stem more from inner properties of the models, rather than from differences between the corresponding induction algorithms.

We mainly consider four induction algorithms: J48 and CART, that induce *decision trees* (Hastie *et al.* (2002); Witten and Frank (2005)); RIPPER, which induces *disjunctive normal form formulas* (Cohen (1995)), and ADTree, which induces *alternating decision trees* (Freund and Mason (1999)). The main differences between CART and J48 relies on the surrogate they use, and the way they carry out pruning. We now briefly describe the classifiers.

A decision tree (DT) is a classifier which makes a simple recursive partitioning of $\mathcal{O}$ according to boolean tests on observation variables. Figure 4 presents a simple DT, in the case where observations contain two real-valued variables x_1, x_2 (they may be defined by more variables, but these are not present in the DT). Classes are predicted following the values at the *leaf* nodes, that we also call *prediction nodes* (Figure 4). To be classified, an observation follows a path from the root of the tree (the upmost node) to a leaf, according to the test it satisfies in the non-leaf nodes of the tree. For example, in Figure 4, an observation for which $x_1 = 0.5$ and $x_2 = 0.1$ would follow the leftmost path to the leftmost leaf, and thus be classified 1. The partition of $\mathcal{O}$, induced by the DT, may be simply represented as shown in Figure 4. In a DT, the most important node is naturally the root node, since it participates to classifying *each* possible observation. As we move bottom-wards, the nodes encountered participate to classifying a smaller number of observations, and are thus gradually devoted to *local* classification.

An alternating decision tree (ADT) is a powerful generalization of decision trees coined by

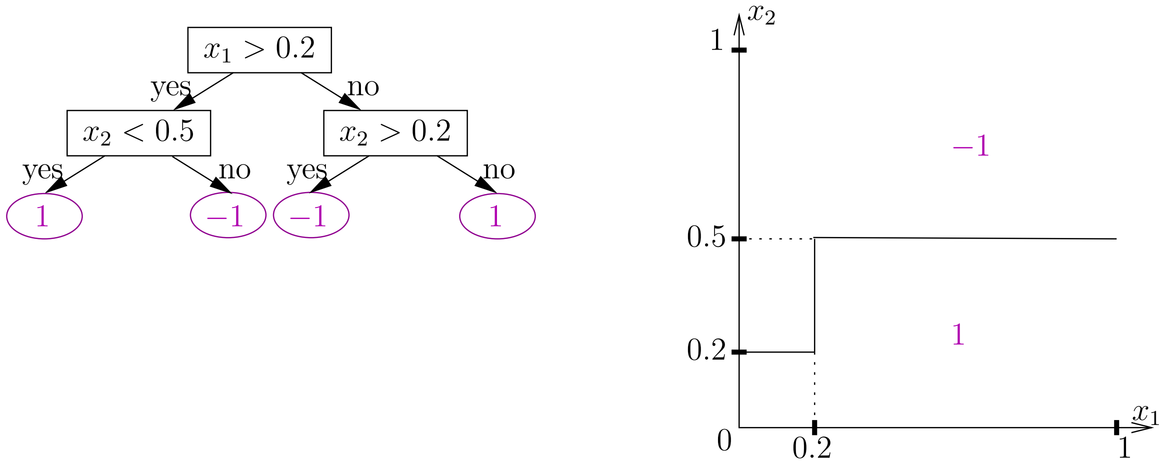

Figure 4: A decision tree (DT, left) and the partition of $\mathbb{R}^2_+$ it defines (right, see text for details). The topmost node of the tree is called the root of the tree, and the prediction nodes, labeled by classes (in purple), are called leaves.

Freund and Mason (1999), in which prediction nodes (in purple in Figure 5) may appear throughout the ADT. A second kind of nodes, called *decision nodes*, specify conditions on observation variables. An observation is classified by following not just one path (like in a DT), but *all* paths for which the associated decisions are true, and summing along each of these paths *all* prediction nodes encountered. For example, An observation for which $x_1 = 0.5$ and $x_2 = 0.1$ would sum the following prediction nodes (in a depth-first search manner): $0.5 - 0.2 + 0.4 + 0.5 = 1.2$ (See Figure 5). In an ADT, there is a subset of decision nodes of particular relevance that share the same importance as the root node in a DT, because they participate to classifying *each* possible observation. These *top* decision nodes are shown inside a dashed rectangle in Figure 5. More than simply deciding the class of any $o \in \mathcal{O}$, the prediction nodes attached to top decision nodes give an indication of the *confidence* in the classification of top decision nodes, hence providing us with a relevance indicator for top decision nodes.

This confidence stems from the fact that prediction nodes are *logit* (Hastie *et al.* (2002)). Let $p \in [0, 1]$; the (scaled) logit of p, $\ell(p) \in \mathbb{R}$, is defined as:

$$\ell(p) \doteq \frac{1}{2} \ln \frac{p}{1-p} \ . \tag{3.3}$$

From the information geometric standpoint, the logit is a natural lift of probabilities to reals in a space embedded with an entropic distortion (Nock and Nielsen (2009)). For example, in Figure 5, value -0.2 is a logit whose expression meets:

$$-0.2 = \ell(\hat{\mathrm{P}}[c(o) = +1 | x_2 < 0.5]) = \frac{1}{2} \ln \frac{\hat{\mathrm{P}}[c(o) = +1 | x_2 < 0.5]}{\hat{\mathrm{P}}[c(o) = -1 | x_2 < 0.5]} \ , \tag{3.4}$$

which yields $\hat{\mathrm{P}}[c(o) = +1 | x_2 < 0.5] \approx 0.4$, where $\hat{\mathrm{P}}$ is a probability estimator, which does *not* follow the sample's density, but provides us anyway with a relevance indicator of this top decision node among all top decision nodes. Remark thus that the variable in any top decision node may be represented in $\mathbb{R}^2$ according to the two logits of its prediction nodes. In this top

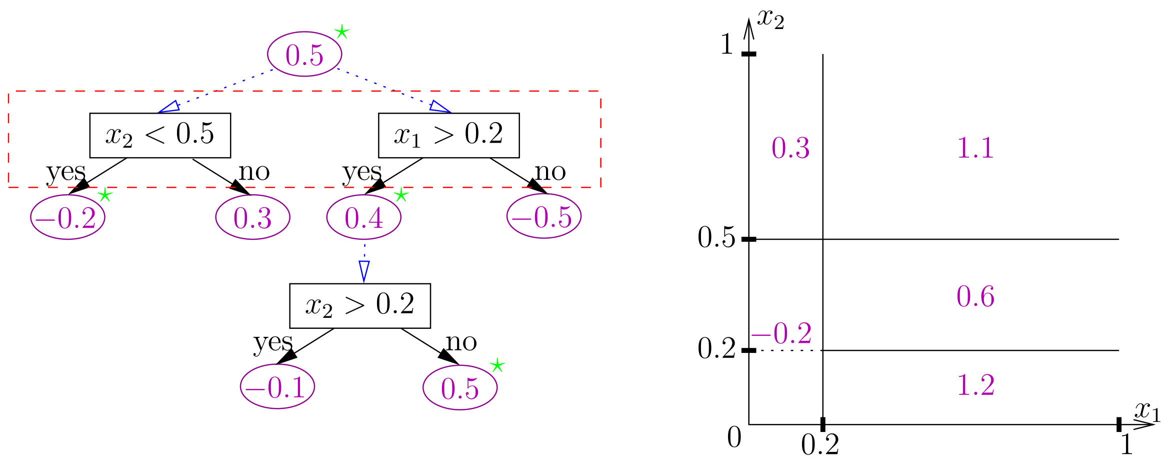

Figure 5: An alternating decision tree (ADT, left) and the partition of $\mathbb{R}^2_+$ it defines (right). Green stars ($\star$) show the prediction nodes that would be summed up for an observation for which $x_1 = 0.5$ and $x_2 = 0.1$. The dashed red rectangle shows the decision nodes that would be used for any observation (see text for details).

decision nodes' *relevance diagram*, the "better" this variable, the farthest from the origin its point in the relevance diagram (Figure 6 provides a relevance diagram over our data, see Section 4). Furthermore, the proximity between two variables in the relevance diagram indicates similarities in their classification. In the same way as a DT does, an ADT performs a partition of $\mathcal{O}$, displayed in Figure 5 for the ADT at hand.

To finish up with classifiers, a disjunctive normal form formula (DNF) is simply a set of conjunctions of tests over description variables, associated to one of the two classes. Table 1 gives examples of DNFs obtained on one of our datasets; consider for example the DNF obtained on W4P07. All logical rules are associated to the same class (here, 1), and to be classified in this class, an observation has to meet the conditions of at least one of these rules. Otherwise, it is classified in the other class. For example, a firm for which $\text{ROI0} = -2, \text{GW3} = 14, \text{QRA0} = 419$ would trigger the first rule (regardless of the values of its other observation variables), and thus be classified as risky. What is interesting in Table 1 is that all rules are predicting the failure risk. DNFs are much easier to interpret than DT or ADT (hence our choice to include them), yet they bring sometimes large classifiers, on which many rules explain a tiny part of the data, making the interpretation less obvious than the one which follows from our example in Table 1 (See Section 4).

Finally, we have also tested a very powerful boosting algorithm, ADABOOST (Freund and Schapire (1997)). Roughly speaking, ADABOOST works by repeatedly calling (T times) another learning algorithm (called a *weak learner*: *e.g.* J48, CART, etc...), and then making a linear combination of all classifiers obtained to form a *meta* classifier which dramatically improves the accuracy of the single models in general. Thus, what is lost in interpretability is in general gained in accuracy, and helps somehow to get an idea of the smallest risks achievable (sometimes, they even approach Bayes rule's). A notable feature of ADABOOST is that it is known *not* to overfit in general, with very large models that achieve extremely small risk (Hastie *et al.* (2002)). Such very large and accurate models may be used as black boxes to predict risk default, in a much more convenient way than *e.g.* neural networks.

W4P07 (1.36%)	
If $(\textsc{roi0} \leq -1.62) \wedge (\textsc{gw3} \geq 13.29) \wedge (\textsc{qra0} \leq 418.81)$	Then 1
Or if $(\textsc{sca2} \geq 1.00) \wedge (\textsc{roi0} \leq 2.42) \wedge (\textsc{dy0} \geq 24.24) \wedge (\textsc{bx3} \geq -0.43) \wedge (\textsc{dv1} \leq 9.70)$	Then 1
Else	-1
W6P09 (1.14%)	
If $(\textsc{qra0} \leq 461.39) \wedge (\textsc{caf13} \geq -0.93) \wedge (\textsc{tcf1} \geq 1.41) \wedge (\textsc{cf0} \leq 21.62) \wedge (\textsc{year} \geq 1982)$	Then 1
Or if $(\textsc{gw2} \geq 6.71) \wedge (\textsc{bx2} \geq 0.21) \wedge (\textsc{dx3} \leq -4.14)$	Then 1
Or if $(\textsc{dx0} \in [81.42, 90.61]) \wedge (\textsc{caf13} \geq 5.07)$	Then 1
Else	-1
W6P10 (1.76%)	
If $(\textsc{dy1} \leq -2.78) \wedge (\textsc{caf10} \leq 49.19) \wedge (\textsc{qra1} \geq 37.78) \wedge (\textsc{caf11} \geq 1.46)$	Then 1
Or if $(\textsc{gw0} \leq 6.97) \wedge (\textsc{dv0} \leq 8.13) \wedge (\textsc{qra2} \leq -12.57) \wedge (\textsc{year} \in [1989, 1999])$	Then 1
Or if $(\textsc{dy3} \leq -1.71) \wedge (\textsc{roi0} \leq 0.51) \wedge (\textsc{cf1} \leq -3.08)$	Then 1
Else	-1

Table 1: Examples of DNFs obtained with RIPPER on three datasets (see Section A). We also indicate the average true risk estimated from stratified cross-validation (See Section 4; see text for details).

4. Experiments

Our general objectives are mainly twofold : (i) evaluate the best coding of observations from the wavelet standpoint and evaluate the reliable prediction horizons of timely patterns — *i.e.* to what extent we may predict failure after the last time stamp observed, (ii) evaluate the potential of the boosting methodology on the task (*e.g.* which kind of models are the best to explain the data, and whether we may obtain classifiers with very small true risk). On top of these objectives, we wish to make a fair comparison between financial ratios that are used as indicators of failure (Laitinen (1993)) and the available variables available in company tax return which could be used as first primers to spot the risk of failure, in particular our blinkers (CF, DX, DY, see Section A, Crutzen and Van Caillie (2008)). For this objective, we create, from each of our primary datasets, three datasets based on filtering variables created from time series: (i) the first, referred to as "Full", contains all variables; (ii) the second one, referred to as "|L", restricts time series to the ratios of (Laitinen (1993)), thus giving $3 + (5 \times 4) = 23$ variables; (iii) the third one, referred to as "|B", restricts time series to blinkers, thus reducing even further the number of observation variables to $3 + (3 \times 4) = 15$. We use the terminology "primary" datasets, because we actually craft more than one dataset, depending on the prediction horizon of data *and* the number of coefficients in the wavelet expansion (recall objective (i) above). We generate seven datasets, each of which having name with pattern "WXPY": $X \in \{4, 6\}$ gives the number of time stamps used to compute wavelet expansion, and Y gives the year(s) on which failure is detected. When y is a range, *e.g.* 06-10, it means that all failures in the range are detected (in the former example, from 2006 to 2010) — hence, the dataset becomes intuitively harder as the yearly patterns of failure may vary. Section A presents in complete details these datasets.

To make fair comparisons, we have used the Java platform WEKA to run all our algorithms. Each of them is run with WEKA's default parameters — thus, results do not stem from careful

	AdaBoost+J48									Logistic regression			MLP		
	Full			|L			|B			Full	|L	|B	Full	|L	|B
$T =$	1	10	100	1	10	100	1	10	100	N/A	N/A	N/A	N/A	N/A	N/A
W4P06-10	7.92	2.16	0.81	8.47	2.79	1.08	9.72	3.97	1.89	19.55	25.05	28.37	7.21	25. 41	16.12
W4P07	1.28	0.32	0.08	1.29	**0.16**	**0.00**	1.85	0.40	0.32	3.62	5.79	9.65	1.37	3.45	2.41
W4P08	2.56	0.08	0.00	2.64	0.32	0.08	3.20	0.64	0.24	5.20	14.65	18.73	2.16	7.84	2.88
W4P09	4.97	0.89	0.16	**4.56**	1.06	0.16	4.72	0.98	0.16	16.29	27.87	23.39	4.48	17.85	8.15
W6P07-10	3.38	0.64	0.00	**2.86**	0.84	0.11	3.91	0.85	0.21	4.87	10.49	19.60	3.18	5.93	3.39
W6P09	2.39	0.41	0.21	**1.97**	**0.31**	**0.10**	3.11	0.42	0.31	3.84	7.69	6.96	2.08	4.26	2.49
W6P10	2.39	0.10	0.00	**2.18**	0.31	0.21	**1.76**	0.10	0.00	4.57	9.67	7.59	1.76	2.59	2.08

Table 2: Average true risk of AdaBoost using J48 as weak learner, for a number of boosting rounds T equal to 1 (hence, only one DT is built), 10 or 100. **Bold faces** in the columns of "|L" and "|B" emphasize values that are smaller than for "Full". To compare with other popular techniques (Laitinen (1993); Ravi Kumar and Ravi (2007); Sun *et al.* (2011)), we also indicate, in the last couples of three columns each, the average true risk of multinomial logistic regression and neural networks (multilayer perceptron, MLP; see text for details).

tunings of whichever algorithm's parameters —. True risks are estimated by ten folds stratified cross-validation: each dataset is split into ten parts, the classifier is learnt on nine, and tested on the tenth; averages over the ten possible runs give the estimated true risk.

AdaBoosting does not overfit, captures the nonlinearity of failure prediction, and produces powerful blackboxes Table 2 presents the results of AdaBoost using J48 as weak learners, when the number of boosting rounds ranges between 1 and 100. In this latter case, AdaBoost requests 100 DT and combine them in a single linear combination, which accounts for a large black box classifier. Applying AdaBoost thus amounts to performing a regression with potentially highly nonlinear features. AdaBoost was ran on all datasets and their filterings (Full, |L, |B). Since linear methods and neural networks are appreciated in the field (Laitinen (1993); Ravi Kumar and Ravi (2007)), we also compare the results with (multinomial) logistic regression and multilayer perceptron in WEKA. Remark that, while logistic regression brings models whose size (number of variables in the model) compares to AdaBoost's with $T \leq 10$, neural networks build *very large* models, with sometimes more than thirty hidden layers — for a learning time which can be huge, exceeding one hour on a Mac Pro with eight processors to complete the cross-validation, while AdaBoost's results are obtained in less than a minute even for $T = 100$.

The most important conclusion, obvious from Table 2, has quite general scope: failure prediction is *highly* nonlinear, as displayed by the extremely poor logistic regression results provided by WEKA, with average true risk higher by order of magnitudes than *any* of AdaBoost's results — regression's true risks are smaller than those of Laitinen (1993), but the magnitude in the differences is *much* smaller than compared to AdaBoost's —. A second conclusion is also quite general: predicting with steroids (huge classifiers, huge running time, etc.) does not help in failure prediction, as the results of neural networks are extremely poor compared to AdaBoost. This clearly sounds like a warning for hybrid methods crafted without caution. Neural networks also display a singular pattern, as their performances on |B is systematically (much) better than on |L, thus tending to prove that blinkers may prove in handy for failure prediction, or similarly that one has to carefully select his/her observation variables prior to learning a predictor for failure.

The second most important conclusion is that AdaBoost does not overfit on *any* of the

	Full	|L	|B
w4p06-10	tcf2(-1,tcf0(gw0,dy3))	tcf2(-1,tcf2(roi0,sca3))	dx1(cf0(legal form,dx1),legal form(· · ·))
w4p07	roi0(gu0(-1,sca2),-1)	roi0(tcf0(sca2,-1),-1)	cf3(-1,dy3(cf2,legal form))
w4p08	fl0(-1,cf0(gw0,-1))	qra0(tfd0(qra1,sca3),roi0(sca0,-1))	cf0(dx0(cf1,legal form),-1)
w4p09	tcf2(-1,dv0(cf3,-1))	tcf2(-1,qra2(-1,roi1))	dx0(cf0(cf1,cf1),legal form(· · ·))
w6p07-10	qra3(-1,roi2(-1,bx0))	qra3(-1,roi2(-1,qra1))	dy1(dy3(dx0,-1),dy0(dx0,dx3))
w6p09	qra3(-1,dv0(dv0,-1))	qra3(-1,qra2(qra2,-1))	dy0(-1,dx0(-1,dx1))
w6p10	qra1(-1,tcf0(dv1,-1))	qra1(-1,tcf0(tcf0,-1))	dx1(cf2(-1,dy1),-1)

Table 3: Compact notation of the dts obtained with j48 on each of our datasets (the dt is built over the complete dataset). To save space, we replace the twelve children of node legal form by "· · ·" (see text for details).

datasets, and regardless of the variables kept. This is important because it opens the possibility to craft a very powerful black box to predict failure, even for the most difficult dataset in which failures are recorded at various time stamps (w4p06-10, Section A): for this dataset, inducing a hundred dts buys more than 7% decrease in the estimated true risk compared to the case $T = 1$. The fact that the true risk obtained for w4p06-10 exceeds by far the ones of w4p07, w4p08, w4p09 tend to display the fact that the determinants of failure actually vary depending on the prediction horizon, or, similarly, that failure has timely patterns that vary depending on the year. The second conclusion brought by these experiments is that when we use four time stamps instead of the more informative framework of six time stamps (w4 vs w6, see Section A), the task becomes significantly harder. Still, the results appear pretty good for w4p09, with less than 5% estimated true risk regardless of the observation variables used. The third conclusion is that Laitinen (1993)'s ratios (and, even though less pronounced, blinkers) achieve an excellent performance compared to keeping all variables, *but* this is considerably dampened as the number of boosting rounds (T) increases. This tends to prove that the information captured by ratios, which is not present in raw data, is important to build small models but becomes less crucial as the models get larger, and may be captured by the linear combinations of dts. Blinkers, on the other hand, display the fact that keeping a very small number of (carefully selected) observation variables may yield performances quite comparable to those achieved with all variables, or even the financial ratios. This is obviously good news, as it indeed proves (or confirms) that the company tax return is an indicator of the firm's financial health.

Decision trees reveal hierarchies among ratios and blinkers To simplify notations of dt, we adopt the *compact* recursive notation which consists in indicating, for each internal node of a dt (thus, excluding leaves), the variable at the node followed, in parentheses, by the notations of its left- and right-subtrees. For example, the dt in Figure 4 would be written $x_1(x_2(1, -1), x_2(-1, 1))$. While it loses the tests made on variables, this notation has the benefits to provide an extremely compact representation of the tree shape and variables. To simplify further and catch only the upmost variables, we shall limit ourselves to nodes at depth 0 (the root), 1 (the nodes immediately below the root) and 2 (the nodes immediately below depth-1 nodes). Remark that by keeping upmost variables, we keep the most important variables from the prediction standpoint: indeed, the variable at the root node classifies *all* observations, while its immediate children still participate to classify significant portions of $\mathcal{O}$ in general. Table 3 allows to draw the following conclusions. Laitinen (1993)'s ratios are extremely handy to build dts: in the "Full" datasets, a ratio is at the root in all but one case. It is also very

	J48			CART			ADTREE$_{100}$			RIPPER		
	Full	|L	|B	Full	|L	|B	Full	|L	|B	Full	|L	|B
W4P06-10	7.92_{67}	8.47_{80}	9.72_{127}	7.84_{36}	8.28_{50}	8.82_{57}	5.31	6.58	6.84	9.27_{12}	$\mathbf{9.01}_{13}$	9.72_{16}
W4P07	1.28_{30}	1.29_{10}	1.85_{52}	2.09_{12}	2.25_{10}	2.82_{18}	0.71	**0.48**	**0.64**	1.36_{2}	$\mathbf{1.21}_{4}$	1.45_{4}
W4P08	2.56_{29}	2.64_{29}	3.20_{70}	3.44_{18}	4.96_{29}	4.64_{34}	1.28	**1.28**	1.60	2.80_{6}	3.28_{7}	3.36_{10}
W4P09	4.97_{56}	$\mathbf{4.56}_{70}$	$\mathbf{4.72}_{87}$	4.81_{65}	6.35_{31}	4.97_{36}	2.77	**2.20**	3.99	3.99_{16}	$\mathbf{3.75}_{20}$	4.23_{20}
W6P07-10	3.38_{26}	$\mathbf{2.86}_{38}$	3.91_{53}	2.86_{16}	4.45_{19}	4.44_{25}	1.27	2.01	1.69	2.96_{13}	3.18_{12}	3.18_{14}
W6P09	2.39_{23}	$\mathbf{1.97}_{23}$	3.11_{27}	2.39_{12}	2.81_{20}	$\mathbf{2.18}_{17}$	0.52	**0.52**	0.62	1.14_{3}	1.45_{4}	1.55_{5}
W6P10	2.39_{11}	$\mathbf{2.18}_{11}$	$\mathbf{1.76}_{23}$	2.18_{11}	2.60_{13}	3.95_{16}	0.31	0.72	0.72	1.76_{3}	$\mathbf{1.35}_{3}$	3.01_{4}

Table 4: Average true risks of four algorithms that build decision trees (DT: J48, CART), alternating decision trees (ADT: ADTREE) and disjunctive normal form formulas (DNF: RIPPER). The small numbers in indices are the number of leaves of the DTs or the number of rules of the DNFs. ADTREE is run with a fixed number of 100 variables in the ADTs. **bold faces** denote errors no greater than for the Full dataset (see text for details).

interesting to see that QRA is much more important for prediction in datasets with six time stamps, while TCF (and to a lesser extent, ROI) turn out to be better for datasets with four time stamps. It thus comes that liquidity ratios (TCF, QRA) are the most important to predict failure, regardless of the encoding of wavelets. Consider data with four time stamps (W4, see Figure 2 for the nomenclature of wavelet expansion's coefficients): the most informative variable to predict failure depends on the prediction horizon: it is the average profitability for P07, the average net turnover (FL) for P08, and becomes the variation of the traditional cash flow (TCF) over the *first* time stamps for the farthest prediction horizon (P09), probably indicating in this case a latency between the signs of failure and failure itself. When removing FL from data (|L), the most important ratio used becomes (logically) QRA — still considering its average value —, as turnover impacts both financial assets and debt (*i.e.*, QRA). When considering only blinkers (|B), the most important range becomes the full time series (DX0, DX1, DY0, DY1, CF0), with a more pronounced importance of LEGAL FORM when time series contain less time stamps (W4).

dts, adts and dnfs display the importance of being able to tune the model, to predict and interpret failure accurately Table 4 displays the average true risks of four algorithms building various classifiers. Without surprise, ADTs bring the best results overall, in part because they are able to capture both linearities and non-linearities in data (Freund and Mason (1999)), and also because the ADTs built are quite large compared to the trees built. Considering only DTs and DNFs on Full datasets only, we see that DTs outperform DNFs on W4P06-10, W4P07, W4P08 and W6P07-10, while they are outperformed on W4P09, W6P09 and W6P10. Even when a DT can easily be transformed in an equivalent DNF, by creating one rule for each possible path from the root to a leaf (Ravi Kumar and Ravi (2007)), one caveat of this procedure is that rules are forced to share common variables. For example, *all* rules share the variable at the root of the tree. While this constraint may be hard to understand from the interpretation standpoint, it may also be damaging from the accuracy standpoint, for data on which the best DNF are such that no variable belongs to many rules. Consider for example the DNF obtained on W6P10 in Table 1: in this DNF, no rule shares a common variable with others. To write it in the form of a DT, it would necessitate a DT far bigger than the DNF, which perhaps explains the fact that DT learning algorithms (J48, CART) perform clearly worse than RIPPER. Also, DNF belong to the classifiers that are the simplest to interpret: consider the DNF obtained on W4P07 in Table 1. Reading it from left to right, the upmost rule says that if the average return on investment ratio is very small, if the average quick ratio does not exceed 418.81, and if net earnings (GW) significantly *fall down* (recall that Haar's mother wavelet is piecewise *decreasing*, Figure 1) during the *last times tamps*, then the firm faces rapid failure. It is also interesting to interpret rules as the prediction horizon increases: for example, consider the DNF obtained on W6P10 in Table 1. Its bottommost rule says that if available funds (CF) increase during the first half of time stamps, but fiscal and social debts sufficiently increase during the *second* half of time stamps, while the average return on investment ratio is small, then the firm faces failure *years* after time series measurements. Finally, the rules built display timely patterns whose time support may vary a lot depending on the prediction horizon: the topmost rule for W6P10 involves variables that *all* span the window from 2003 to 2008 (Section A), while the second rule obtained on W6P09 involves variables whose window span 2003-04 (GW2, BX2), or 2005-06 (DX3). It is worthwhile remarking that, with six time stamps, the DNF for the farthest prediction horizon (P10) has wavelet coefficients which rely on the whole span (2003-08) for 80% of them, while the DNF for the shorter prediction horizon (P9) has wavelet coefficients which rely on the whole span for 50% of them only, perhaps indicating that far prediction horizons should rely on the most robust wavelet coefficients (because the time window of $\phi(t)$ and $\psi(t)$ are twice as large than those of $\psi(2t)$ and $\psi(2t-1)$, Figure 2).

adts allow to drill down in the relevance of ratios and blinkers Figure 6 summarizes the relevance diagrams over all datasets, for each of the financial ratios of Laitinen (1993) and our blinkers. Recall that only the predictors that appear at the root of an ADT are plotted. Since the algorithm prevents the two logits (eq. (3.3)) associated to some variable to be of the same sign (Freund and Mason (1999)), it comes that only two quadrants are used in the real plane: (+,-) and (-,+). The actual quadrant used is extremely important. For example, numerical variables that tend to be negatively correlated with failure risk should be located in quadrant (+,-): this is the case for the coefficient $\langle h, \phi \rangle$ of all financial predictor in (Laitinen (1993)), and one may check that ROI0, QRA0, SCA0, TCF0 are all in this quadrant in Figure 6. TFD0 does not appear in this Figure, seemingly because its discriminative power on risk default seems smaller than for the other ratios (remark that TFD never occurs on the topmost DT variables on the Full dataset, as seen in Table 3). Furthermore, ROI2, ROI0 and QRA0, the only predictors that appear thrice (see the triangles in Figure 6), are clearly selected as the best for the task at hand by the algorithm.

One may also remark that nearby points have approximately identical logit coordinates, which may indicate an equivalent discriminative power for the risk default prediction. This observation is useful to group financial predictors of Laitinen (1993) and blinkers on sets of predictors that bear approximately identical behaviors. For example, DX0 and ROI3 may be grouped, and also CF3 and QRA1.

Finally, the relevance diagram displays the fact that the most important intervals for blinkers with ADTs correspond to the coordinates of $\phi(t), \psi(t), \psi(2t-1)$ (Figure 2), that is, the informative variations of time series for blinkers cover the most recent time stamps. This is not the case for financial ratios (Laitinen (1993)), for which virtually all wavelet expansion coefficients are useful for failure prediction. This goes hand in hand with the fact that these ratios are crafted to capture business failure information — which is thus spread over all wavelet expansion coefficients —, while company tax return, which contains blinkers, is eventually more efficient at

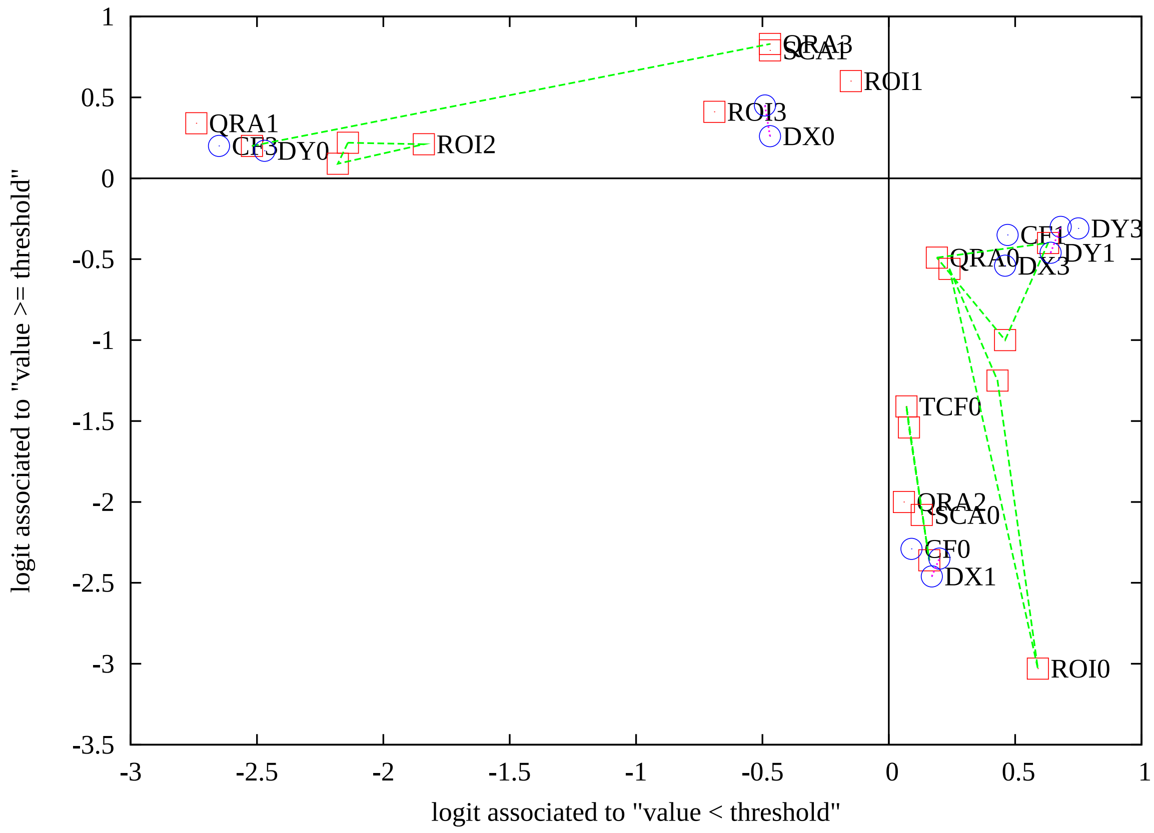

Figure 6: Relevance diagram (Section 3) showing only among the top decision nodes those that are the financial predictors of Laitinen (1993) (three letters) and our blinkers (two letters). Lines or triangles group sets of tests on the same variable (but on different thresholds, see text for details).

providing the latest informations for the events it can contribute to predict, events to which business failure is just an example.

Hardest data and dt rank variables, with outsiders to ratios and blinkers Filtering datasets to keep ratios (|L) or blinkers (|B) is an *ad hoc* reduction of the number of observation variables. A tantalizing question it whether subsets of variables, of perhaps less intuitive relevance, may be combined to get performances comparable — or superior — to whichever combination of ratios or blinkers. Table 5 provides us with a compact representation of the easiest stages of this comparison, that is, leaving one or two time series in data (instead of the three in blinkers, or the five in ratios).

The results obtained are a clear advocacy to carefully select or devise variables out of raw data. Taking as reference our set of initial variables, for which the average error on the Full dataset is 7.92% (Table 4), we see from Table 5 that *nine* combinations of two predictors (only) beat the Full dataset. The nine combinations of variables that beat the full dataset are: (BX, ROI), (BX, QRA), (CAF, CF), (CAF, DX), (CAF, TCF), (CF, DY), (DY, FL), (FL, ROI), (QRA, TCF). Remark that each of them contains at least one blinker (four out of nine) or one financial ratio (five out of nine). Remark also that it is a combination of two *blinkers* (CF, DY) which brings

	BX	CAF	CF	DV	DW	DX	DY	FL	GU	GW	ROI	QRA	TCF	SCA	TFD
BX	*10.45*	7.92	9.27	9.18	10.00	8.91	9.19	9.55	9.10	8.29	7.30	7.12	9.55	8.65	8.56
CAF		*10.81*	7.66	8.74	9.73	7.66	8.29	8.11	9.28	9.82	8.47	8.20	7.48	8.11	9.82
CF			*9.18*	10.18	9.82	**9.01**	**6.40**	8.83	9.55	8.92	9.10	8.74	8.92	9.28	10.18
DV				*15.22*	8.74	9.19	9.55	10.00	11.53	9.10	9.82	9.82	8.92	8.74	9.01
DW					*26.76*	9.19	11.98	8.29	13.24	10.27	9.37	8.65	10.54	9.37	9.10
DX						*10.54*	**9.46**	8.02	9.10	8.92	9.64	8.83	8.56	9.19	9.91
DY							*12.88*	7.84	9.55	9.55	8.38	7.93	10.45	10.54	9.46
FL								*8.28*	9.46	9.10	7.66	9.19	8.65	9.10	8.56
GU									*13.15*	8.74	9.82	9.82	10.45	9.46	9.37
GW										*10.09*	9.10	8.11	9.46	8.83	10.63
ROI											*9.81*	**8.20**	**7.93**	**9.28**	**6.58**
QRA												*9.82*	**7.39**	**9.10**	**8.20**
TCF													*10.99*	**9.64**	**10.54**
SCA														*10.09*	**10.00**
TFD															*8.10*

Table 5: Average results of J48 on W4P06-10 when filtering further time series to keep only one (diagonal, *italicized*) or two (off-diagonal) time series, for each possible combinations of time series. **Bold faces** are used to pinpoint combinations of blinkers, or combinations of ratios.

the smallest error of all (6.40% only, a decrease of more than 1.5% over the Full dataset).

5. Conclusion

(Automated) business failure prediction faces a number of challenges, chief among which (i) obtaining accurate models and (ii) obtaining models easily interpretable — viz. by business professionals also freshmen to classifiers. The current directions of research in such automated predictions, that place high expectations on so-called hybrid systems (Ravi Kumar and Ravi (2007)), face pitfalls in these two objectives simultaneously: bigger models are indeed less easy to interpret, while being more prone to over-fitting. In the context of the last financial crisis, these pitfalls obviously become of the utmost importance.

Recent paper in the business failure prediction field have started to use boosting algorithms like ADABOOST (Sun *et al.* (2011)), yet they do not seem to be willing to ride their luck in exploiting boosting to its fullest potential — that of an extremely powerful classification theory which brings complex classifiers while avoiding over-fitting in most cases. It is important to insist on this fundamental result which states that boosting is a methodology to combine predictors slightly different from random, resulting in a big(er) predictor arbitrarily accurate (Hastie *et al.* (2002)). In other words, boosting tells us how to flock basic building classification blocks so as to be sure to meet objective (i) above, *even when* the basic building blocks are only a little more accurate than the unbiased coin. Since usual boosting algorithms like J48, CART, ADTREE bring models that are easily interpretable (compared *e.g.* to neural networks), objective (ii) above can also be tackled with boosting.

Our paper may be viewed as a first step in the direction to use boosting — or more generally information geometric methods to which famed boosting algorithms belong (Nock and Nielsen (2009)) — to its fullest potential in the business failure prediction field.

But the boosting recipe to combine the ingredients faces the famed no-free lunch theorems of Wolpert and Macready (1995) like any other classification algorithm. In the context of business failure prediction, these theorems simply tell us that no fixed classification algorithm may be assumed to systematically beat another classification algorithm. The important word here is *fixed*, and, to dampen the effects of no-free lunch, it may be useful not to work on the boosting recipe itself, but rather on the "relevance" of its most basic ingredients: observation variables. We have displayed the importance of wavelets to capture timely patterns of failure, thus putting in the observations the trends of variations of time series. Our results show that these patterns are crucial: for example, they are much more important than the time at which failure occurs (the YEAR variable appears in seldom classifiers). We may wonder whether we can go much further from basic applications of wavelet expansions, and devise powerful algorithms that automatically provide boosting, as well as its basic classification blocks, with the most accurate variables for the failure prediction task. This shall be the subject of future works.

Acknowledgments

Work supported in part by ANR projects ANR-07-BLAN-0328-01 and ANR-07-MDCO-008-03. The authors would like to thank Erkki Laitinen for having shared with us papers and ideas on failure prediction.

References

Amari, S.-I. and Nagaoka, H. (2000). *Methods of Information Geometry*. Oxford University Press.

Balcaen, S. and Ooghe, H. (2006). 35 years of studies on business failure: an overview of the classic statistical methodologies and their related problems. *The British Accounting Review*, **38** (1), 63 – 93.

Breiman, L., Freidman, J. H., Olshen, R. A. and Stone, C. J. (1984). *Classification and regression trees*. Wadsworth.

Chui, C.-K. (1992). *An Introduction to Wavelets*. Academic Press.

Cohen, W.-W. (1995). Fast effective rule induction. In *Proc. of the 12 th International Conference on Machine Learning*, pp. 115–123.

Cover, T.-M. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, **13**, 21–27.

Crutzen, N. and Van Caillie, D. (2008). The business failure process: An integrative model of the literature. *Review of Business and Economics*, **53**, 288–316.

Freund, Y. and Mason, L. (1999). The alternating decision tree learning algorithm. In *Proc. of the 16 th International Conference on Machine Learning*, pp. 124–133.

— and Schapire, R. E. (1997). A Decision-Theoretic generalization of on-line learning and an application to Boosting. *Journal of Computer and System Sciences*, **55**, 119–139.

Friedman, J., Hastie, T. and Tibshirani, R. (2000). Additive Logistic Regression : a Statistical View of Boosting. *Annals of Statistics*, **28**, 337–374.

Gepp, A., Kumar, K. and Bhattacharya, S. (2010). Business failure prediction using decision trees. *Journal of Forecasting*, **29** (6), 536–555.

Hastie, T., Tibshirani, R. and Friedman, J. (2002). *The Elements of Statistical Learning*. Springer Series in Statistics.

Huang, S.-M., Tsai, C.-F., Yen, D.-C. and Cheng, Y.-L. (2008). A hybrid financial analysis model for business failure prediction. *Expert Systems with Application*, **35**, 1034–1040.

KEARNS, M. and MANSOUR, Y. (1999). On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, **58**, 109–128.

LAITINEN, E. (1993). Financial predictors for different phases of the failure process. *Omega*, **21** (2), 215 – 228.

LI, H. and SUN, J. (2011). Predicting business failure using support vector machines with straightforward wrapper: A re-sampling study. *Expert Systems with Applications (in press)*.

NOCK, R. and NIELSEN, F. (2008). On the efficient minimization of classification-calibrated surrogates. In *Advances in Neural Information Processing Systems*21*, pp. 1202–1208.

— and — (2009). Bregman divergences and surrogates for learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**, 2048–2059.

RAVI KUMAR, P. and RAVI, V. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques — a review. *European Journal of Operational Research*, **180**, 1–28.

RAVISANKAR, P., RAVI, V. and BOSE, I. (2010). Failure prediction of dotcom companies using neural network-genetic programming hybrids. *Information Sciences*, **180**, 1257–1267.

SUN, J., JIA, M.-Y. and LI, H. (2011). AdaBoost ensembles for financial distress prediction: an empirical comparison with data from Chinese listed companies. *Expert Systems with Applications*, **38**, 9305–9312.

WITTEN, I.-H. and FRANK, E. (2005). *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

WOLPERT, D.-H. and MACREADY, W.-G. (1995). *No free lunch theorems for search.* Tech. Rep. SFI-TR-95-02-010, Santa Fe Institute.

A. Appendix: Data and parameterizations

Our dataset comprises a thousand French firms. Each raw observation contains four subsets of variables of interest for the task at hand:

- Laitinen (1993)'s ratios: those ratios are meant to contain informations most relevant for failure prediction;
- *blinkers*: variables chosen from company tax return, presented by Crutzen and Van Caillie (2008) as containing information about the state of a firm a unable to meet its financial obligations either on the long or the short term. These variables should display the liquidity and solvency state of a firm;
- in order to capture the *solvency symptoms* presented in Crutzen and Van Caillie (2008), we also put other variables from the company tax return, which we think relevant to the various symptoms;
- we also put other variables, not meant to be related directly to failure: an information from the business sector, the birthdate and the legal form of the society.

Observations comprise fifteen time series and three numeric/symbolic variables that are not time series. We list hereafter the variables of each observation, starting by the variables that are not times series (and thus, that do not undergo wavelet expansion). For the variables in the company tax return, we sometimes indicate the section(s) to which the variable belongs.

Variables that are not time series

(1) NAF: section of the French nomenclature for activities, a nominal mono-valued variable whose domain corresponds to twenty-one letters in the Latin alphabet. For example, letter 'A' refers to Agriculture/Hunting/Forestry, while 'P' refers to Teaching activities;

(2) YEAR: birthdate year of the society, an integer;

(3) LEGAL FORM: legal form of the society, a nominal mono-valued variable whose domain contains twelve values, among *e.g. Limited company*, *Limited liability company*, etc. ;

Variables that are time series in company tax return

(4) BX: accounts receivable (in debts, current assets);

(5) CAF1: operational cash flow;

(6) CF: cash flow (in current assets);

(7) DV: loans (in debts);

(8) DW: advances on contracts in progress (in debts);

(9) DX: suppliers' debts (in debts);

(10) DY: fiscal and social debt (in debts);

(11) FL: net sales (in revenue);

(12) GU: total of financial expenses;

(13) GW: profit before tax;

Variables that are time series in financial ratios Laitinen (1993)

(14) ROI: return on investment ratio (divides profit plus interest expenses over total assets);

(15) QRA: quick ratio (divides financial assets by current debt);

(16) TCF: traditional cash flow ratio (divides traditional cash flow by net sales);

(17) SCA: shareholder capital to total assets ratio;

(18) TFD: traditional cash flow to total debt ratio;

We have chosen CF, DX, DY as *blinkers* for failure prediction, because they are *per se* correlated to failure: CF displays the available funds of the firm. It is a liquidity indicator. Concerning DX, financial difficulties should make this item increase, even before DY increases, because firms naturally prefer to honor debts regarding their suppliers prior to honoring fiscal or social debts. Finally, a firm which encounters financial difficulties should see DY increase.

This is for the description of the raw observations. To generate classes, we create seven datasets, depending on (i) whether we use four or six points for wavelet expansion, and (ii) the time stamp relative to time series on which we wish to predict failure (the *prediction horizon*). Summarizing, each dataset's name fit to the pattern "wXpY":

- $X \in \{4, 6\}$ gives the number of time stamps used to compute wavelet expansion. Remark that since 4 is a power of 2 (and time stamps are regularly spaced), time series over 4 points directly yield their wavelet expansion. When we use 6 time stamps, we compute a regular intrapolation of time series on 4 points prior to wavelet expansion. We obviously lose information, but still we can consider that the 4 time stamps contain an information more accurate than a raw time series already containing 4 time stamps. 4 points time series start in 2003 and end in 2006, 6 points time series end in 2008;
- Y gives the year(s) on which failure is detected. When Y is a range, *e.g.* 06-10, it means that all failures in the range are detected (in the former example, from 2006 to 2010) — hence, the dataset becomes intuitively harder because the yearly patterns of failure may vary from year to year.

The seven datasets are: w4p06-10, w4p07, w4p08, w4p09, w6p07-10, w6p09, w6p10.