Financial Applications of Classification and Regression Trees

A Master Thesis Presented

by

Anton Andriyashin

(188779)

to

Prof. Dr. Wolfgang Härdle

CASE - Center of Applied Statistics and Economics

Humboldt University, Berlin

in partial fulfillment of the requirements

for the degree of

Master of Arts

Berlin, March 24, 2005

Declaration of Authorship

I hereby confirm that I have authored this Master thesis independently and without use of others than the indicated resources. All passages, which are literally or in general matter taken out of publications or other resources, are marked as such.

Anton Andriyashin

Berlin, March 24, 2005

Abstract

This study gives an outline of modern theory of classification and regression trees (CART) and shows the advantages of CART applications in finance. Practical issues regarding CART applications and core implementation are presented. The second part of the work is mainly concentrated on DAX30 market simulation results and shows how a CART-based business application can perform on stock market as well as what supplementary results can be got using CART as a forecasting system. In this realm comparison of technical and fundamental approaches is performed. Finally, information ageing effect in the context of learning sample construction is analyzed.

Keywords: CART, decision trees, financial applications, information ageing effect, simulation

Contents

1 Introduction to Classification and Regression Trees (CART)

Classification and Regression Trees is a relatively new method of discriminant analysis developped by a group of American scientists [1] during the last 25 years. As it is for discriminant analysis, the aim of CART is to classify a group of observations or a single observation into a subset of *known classes*. Comparing to classical parametric discriminant analysis CART offers a number of particular benefits like a high degree of results' interpretability, high precision and fast computation.

CART is a non-parametric tool of discriminant analysis which is designed to represent decision rules in a form of so called *binary trees*. Binary trees split learning sample data imposing univariate linear restrictions and represent resulting data clusters hierarchically starting from *root node* for the whole learning sample itself and ending with relatively homogenous small groups of observations. For each *terminal node* class tag or forecasted value is assigned, hence resulting tree structure can be interpreted as a decision rule.

Recall that traditional discriminant analysis operates only with discrete dependent variables i.e. classes can be enumerated and represented in this way. Imagine that instead of classes learning sample has some numerical data with continuous variables. Moreover, there can be a connection between a continuous variable and a subset of other continuous or disctere variables. In this situation classical discriminant analysis can not be applied while CART will be able to produce decision trees and conduct a classification. Trees with continuous dependent variable are called *regression trees* and in fact are estimators of non-parametric regression model describing possible reltionship between different variables of learning sample.

The following sections will provide an outlook of how decision trees are created, what challenges arise during practical applications and, of course, a number of examples will illustrate the power of CART in financial applications.

1.1 What is CART?

Classification and Regression Trees is a special method of creating decision rules to distinguish between clusters of observations and determine the class of new observations. A particular feature of CART is that decision rules are represented via binary trees, that is why it can be extremely easy to apply these rules in practice.

Consider the following real life example of how high risk patients (those who will not survive at least 30 days after heart attack is admitted) were identified at San Diego Medical Center, University of California on the basis of initial 24-hour data. A classification rule using at most three natural questions was produced (see Figure 1). Comparing with classical discriminant analysis the precision of CART classification was even higher while the rule itself does not require any additional calculations and can be easily used virtually by anybody.

Here observation class is obviously a binary variable: low risk (0) and high risk (1). Consider, however, different situation when we would be interested in the expected

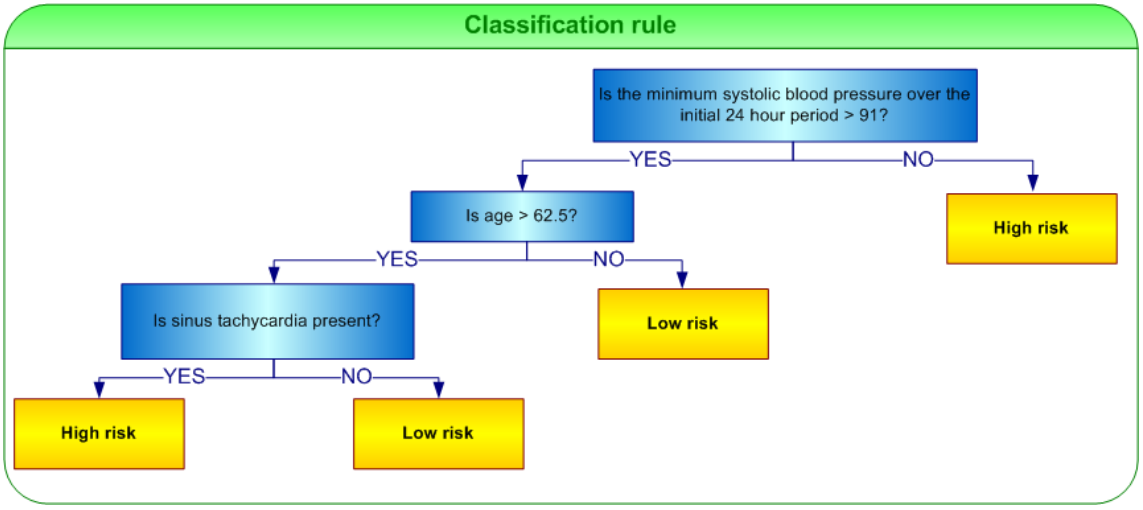

Figure 1: Decision tree example

amount of days the patient will be able to survive. Obviously the decision tree will change its structure but the approach remains the same and *terminal nodes* will contain data on mean expected number of days the patient will survive. Once we have got the decision tree, a new patient can be diagnosed and be assigned to one of two possible groups – everything on the basis of at most three simple questions. Obviously for this example there is an implied assumption that there is a relationship between a set of patient's characteristics and expected durance of his/her life. This relationship was estimated using CART applied to the learning sample where there is information about patients and how long they lived. The decision rule then was explicitely presented via the binary tree. Let X_i denote i-th factor from the factor space of learning sample $\mathbb{X}$ consisting of p variables,

$$\bigcup_{i=1}^{p} X_i = \mathbb{X}$$

Let Y be a dependent variable – binary or continuous. Then we assume that there exists a mapping

$$f : \mathbb{X} \to Y$$

which is estimated in the *step-function class*, that is why decision tree can be represented as it appears in the example. Obviously the estimator is not unique because learning sample data can be split in numerous ways, hence the main question is to find some "good" ways of splitting data so that future observations will be classified correctly.

1.2 CART advantages

When learning sample data are being split in order to find best *node questions*, an iterative computer procedure is initiated and all possible splits are processed. Hence one

may question the efficiency of such method. Nonetheless, CART appears to be a well tuned procedure with numerous benefits.

- CART is non-parametric
  This is very important even for analyzed dependencies of known nature where researchers are able to set up some *a priori* functional forms and becomes extremely important while performing explorative data mining of complex high dimensional structures. When no data strucuture hypotheses are available, non-parametric analysis becomes merely the single effective data mining tool. Moreover, when building such a model one should not make any additional assumptions concerning model errors distribution which becomes a substantial obstacle when sample errors distribution does not match the required one.

- CART does not require variables to be selected in advance
  This means that from a given subset of variables constituting a learning sample CART will automatically select the most significant ones in some sense. Hence even if learning sample holds some irrelevant information due to e.g. *measurement errors* or *misspecification*, the model will choose correct splits by itself and hence account for disturbances automatically.

- CART is very efficient in computational terms
  Although all possible data splits are analyzed, CART architecture is flexible to account all of them and do it quite quickly.

- Results are invariate with respect to monotone transformations of independent variables
  Hence it is relatively easy for a researcher to rescale properly the input variables so that the intrepretability of results could be even higher.

- CART can handle datasets with complex structures
  This becomes more important once no *a priori* information about dataset is available. On the other hand this peculiarity allows to consider a bigger variety of possible model specifications.

- CART is extremely robust to the effect of outliers
  Due to data-splitting nature of decision rules creation it is possible to distiguish between datasets with different characteristics and hence to neutralize outliers in separate nodes.

- CART can use any combination of continuous and categorical data
  That is why researcher is no more limited to a particular class of data and created models will be able to capture more real-life effects to make prediction accuracy higher.

Hence CART can be easily applied to a variety of fields especially in financial applications where nowadays there is a growing need for a set of robust and efficient classification methods.

1.3 How do decision trees work?

In order to create a decision tree two major questions should be answered. Firstly, a proper variable and a proper question value must be determined for each of the tree's nodes. For instance, in the example above there was a question if the patient's age is greater than 62.5 years. Hence *age variable* was somehow the most significant at that particular tree node and a threshold of 62.5 was also determined due to some particular criterion. Secondly, a right configuration i.e. *size* of decision tree must be found since it is possible to split all learning sample data until absolutely class homogenous groups of observations are left whereas such kind of structure will obviously lack predictive power.

Let X_i be the i-th variable of the learning sample $\mathbb{X}$, then an arbitrary decision tree node poses a question like

$$X_i \leq x \tag{1}$$

where x is some constant.

Since such kind of restrictions are univariate, data splitting is always *orthogonal*. Figure 2 is an example of how similar questions may work with simulated two-dimensional data. Here only two questions were sufficient to split the data reasonably.

But how the optimal values were determined? First let's analyze the question of *existance* of splits for an arbitrary data set. Is it always possible to filter the data in the way that each cluster has only class homogenous datasets? The answer is positive since every coordinate of a p-dimensional observation $X_i = (X_i^1, X_i^2, \ldots, X_i^p)$ in the space $\mathbb{X}$ can be bounded in the following way:

$$\begin{cases} a_i^1 < X_i^1 < b_i^1 \\ a_i^2 < X_i^2 < b_i^2 \\ \vdots \\ a_i^p < X_i^p < b_i^p \end{cases} \tag{2}$$

for some arbitrary constants a_i^j and b_i^j.

On the other hand it is quite obvious that these constants are *not unique*, since it is always possible to find p-dimensional vector $\epsilon \to 0$ so that

$$\begin{cases} a_i^1 - \epsilon^1 < X_i^1 < b_i^1 + \epsilon^1 \\ a_i^2 - \epsilon^2 < X_i^2 < b_i^2 + \epsilon^2 \\ \vdots \\ a_i^p - \epsilon^p < X_i^p < b_i^p + \epsilon^p \end{cases} \tag{3}$$

Hence for any learning sample splitting algorithm alowing to separate class homogenous clusters of observations always exists since in limiting case one can split the sample up to a set of single observations. But the solution is non-unique, that is why we will introduce effective ways of splitting the data.

Hence some criterion measuring data homogeneity or *impurity* should be introduced. Then different feasible data splits should be compared according to this measure and

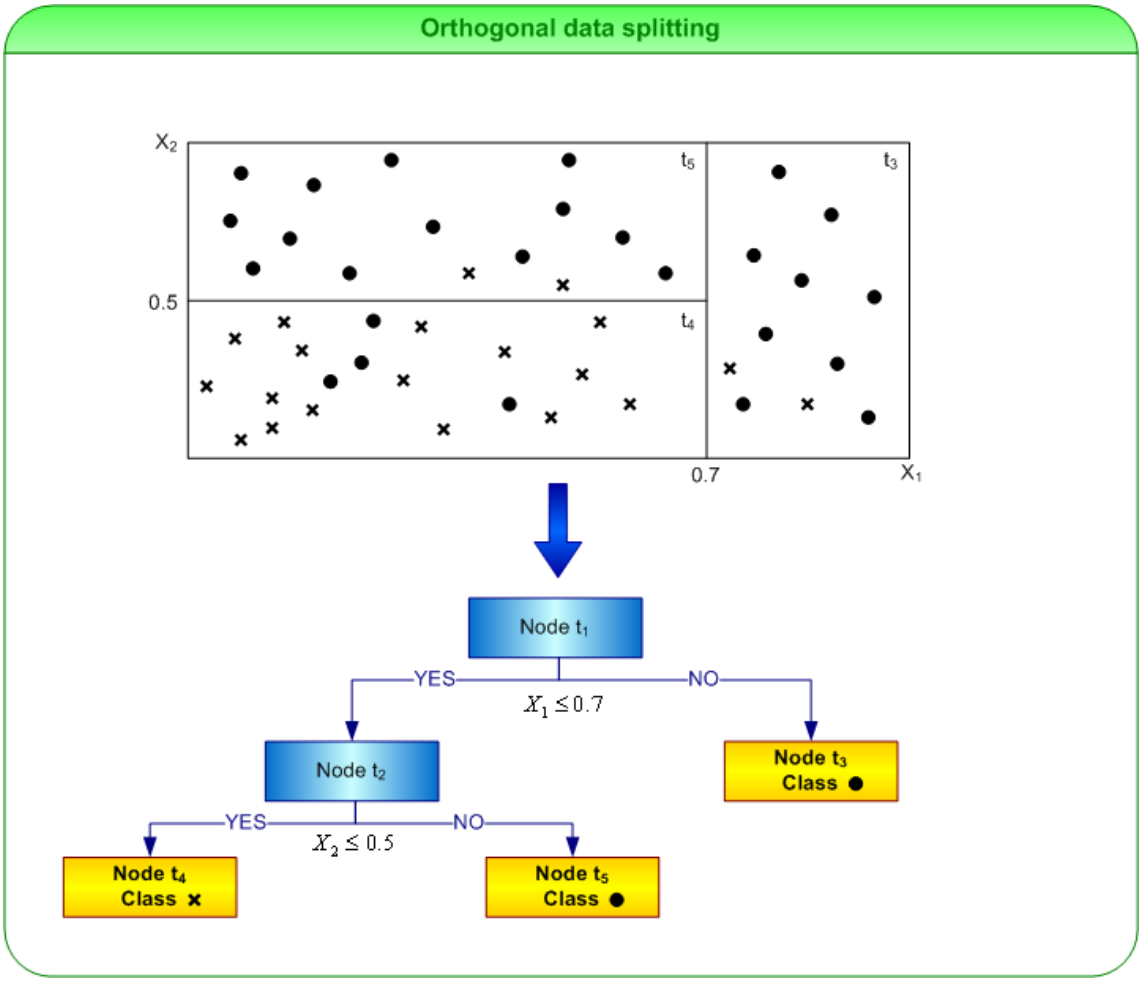

Figure 2: Simulated data splitting via orthogonal clusters

thus it will be possible to find an optimal split for a given tree node once such measure is determined and possesses some "good" properties.

It is also important to point out that for any such criterion the number of feasible splits is always finite. Moreover, it is always possible to split data if question values x are taken as variable values from the learning sample. If there are two one-dimensional observations v_1 and v_2 then obviously a filter in form of $X \leq v_1$ will cluster each observation. Note, however, that a similar filter $X \leq \frac{v_1+v_2}{2}$ will result in same split. Hence it is always sufficient to make a grid based on observation values in the learning sample in order to find the best x given some criterion. For reasons of *robustness* a symmetric grid is usually considered i.e. means between different observations are used to construct filters.

Depending on the type of a decision tree or more precisely – on the type of dependent variable, different splitting approaches based on so called impurity functions are available. We will start with classification trees and then proceed with regression ones.

Suppose next that each split is determined optimally and learning sample is clustered up to absolutely class homogenous groups – a tree representing such a structure is called a *maximum tree* since it is impossible to make additional data split without violating a condition that some base node (or so called *parent node*) has observations of different classes. The next important question arising during tree construction is where to *stop splitting* i.e. how to determine the optimal decision tree configuration. There are different available approaches which share the logic for both types of the trees: classification and regression. We will then try to analyze advantages and disadvantages of these methods especially in the realm of financial applications.

1.4 Impurity measures for classification trees

Let's introduce some basic learning sample data characteristics we will operate in future. Suppose there are N observations in the learning sample and N_j is the overall number of observations belonging to class j, $j = \overline{1, J}$. Then define *class probabilities* as following:

$$\{\pi(j)\}_{j=1}^{j=J} = \left\{\frac{N_j}{N}\right\}_{j=1}^{j=J} \tag{4}$$

i.e. a proportion of observations belonging to particular class relative to overall number of observations.

Let $N(t)$ be the number of observations in node t and $N_j(t)$ – the number of observations belonging to j-th class in the same node t. Then a joint probability of the event that an observation of j-th class falls into node t is:

$$p(j, t) = \pi(j)\frac{N_j(t)}{N_j} \tag{5}$$

Hence $p(t) = \sum_{j=1}^{J} p(j, t)$ and *condtional probability* of an observation to belong to node t given that its class is j is computed as following:

$$p(j\,|\,t) = \frac{p(j, t)}{p(t)} = \frac{N_j(t)}{N(t)} \tag{6}$$

i.e. proprtion of class j in node t. It is obvious that $\sum_{j=1}^{J} p(j\,|\,t) = 1$.

Let's now introduce a new measure of the tree which shows the degree of class homogeneity in a given node and call this characteristic an *impurity measure* $i(t)$ which will be able to represent a class homogeneity indicator for a given tree node and hence will help to find optimal question value x as well as proper variable number for a node equestion. But first we will define an *impurity function* $\varphi(t)$ which is determined on subsets $\{p_1, \ldots, p_J\}$ for $\forall J$ and $p_j \geq 0$, $j = \overline{1, J}$, $\sum_{j=1}^{J} p_j = 1$ so that:

1. φ has a unique maximum at point $\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right)$;

2. φ has a unique minimum at points $(1, 0, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, …, $(0, 0, 0, \ldots, 1)$;

3. φ is symmetric function of $p_1, \ldots, p_J$

Each function satisfying these conditions can be called an impurity function. Given function φ, define *impurity measure* $i(t)$ for a node t as:

$$i(t) = \varphi\left(p(1|t), p(2|t), \ldots, p(J|t)\right) \tag{7}$$

It is important to point out that from given definitions it follows that it is possible to define *multiple* impurity measures for the same node t.

Denote an arbitrary data split by s, then for a given tode t which we will call a *parent node* two *child nodes* arise: t_L and t_R representing correspondingly observations subsets meeting and not meeting filter s so that a fraction p_L of data from t falls at left child node and $p_R = 1 - p_L$ is the share of data in t_R.

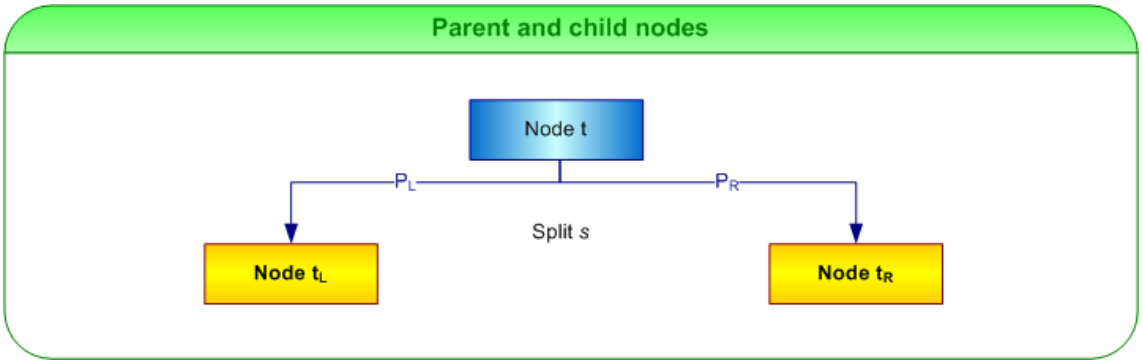

Figure 3: Parent and child nodes hierarchy

Hence a *quality measure* of how well split s filters the data according to class heterogeneity (which can be defined arbitrary) is

$$\Delta i(s, t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R) \tag{8}$$

Obviously, the higher is the value of $\Delta i(s, t)$ – the better split we have since it was possible to reduce data impurity more significantly. Since $t_L \cup t_R = t$, the value $\Delta i(s, t)$ represents a change of data impurity in t solely due to split s.

To find the optimal question value x and the proper variable for that question i.e. to find optimal s it is natural to maximize $\Delta i(s, t)$ by different s at each node t. Hence such kind of procedure will enable one to build a decision tree of any configuration up to a maximum tree.

While searching for optimal value s^*, the value of $i(t)$ in fact remains constant, hence it is equivalent that

$$s^* = \underset{s}{\operatorname{argmax}}\, \Delta i(s,t) = \underset{s}{\operatorname{argmax}}\, \{-p_L i(t_L) - p_R i(t_R)\} = \\ = \underset{s}{\operatorname{argmin}}\, \{p_L i(t_L) + p_R i(t_R)\} \tag{9}$$

where t_L and t_R are implicit functions of s.

If resulting nodes are not class homogenous, the same procedure can be looped until a decision tree becomes of required configuration. Classes are then assigned to terminal nodes using the following rule:

$$\text{If } p(j|\,t) = \max_i p(i|\,t), \text{ then } j^*(t) = j \tag{10}$$

If maximum is not unique, then class $j^*(t)$ is assigned arbitrary from the pool of arguments $\{i\}$ for which $p(i|\,t)$ takes its maximum value.

By this momement function $i(t)$ was not defined, hence the proposed algorithm is in fact quite versatile. But before proceeding with specification of $i(t)$, it is worth pointing out the following. Maximizing the increment of impurity function means that only two levels of a decision tree are taken into account whereas other parts of the tree can not influence optimal split. That is why the procedure can be characterized only as *locally optimal*.

Is it possible to build a *globally optimal* algorithm of data splitting? Imagine that locally optimal maximum tree has n terminal nodes. Then to build a globally optimal decision tree one should check every possible tree structure with the same number of terminal nodes which will rocket the amount of necessary computations. Moreover, n is just an estimate of how many terminal nodes a globally optimal tree should have because in fact it can be different. Hence the procedure should be looped so that all possible combinations are taken into account – this, of course, makes the amount of computations overwhelming.

Thus in practice a locally optimal variant is used, but one should keep in mind that locally optimal tree is not necessarily globally optimal and, that is probably more important, *globally optimal tree is not necessarily locally optimal*.

In financial sphere where computations are sometimes required to be done virtually online or at least to be conducted very quickly, the speed matter becomes crucial, that is why it is reasonable to apply locally optimal procedures. Once an enhanced precision is required e.g. in credit scoring or portfolio optimizations and computation speed recedes into the background, then if sufficient computing power is available it will be possible to construct globally optimal decision rules. Nonetheless, to the best knowledge of the author nowadays commercial versions of software capable of producing globally optimal structures are not present on the market.

Let's get back to the question of impurity function specification. Perhaps the most natural way to define data impurity is to use the *variance* measure. Assign 1 to all observations at node t belonging to class j and 0 to others. Then a sample variance estimate for node t observations is:

$$p(j\,|\,t)\,(1-p(j\,|\,t)) \tag{11}$$

Summing over all J classes we get a so called *Gini index*:

$$\sum_{j=1}^{J} p(j\,|\,t)\,(1-p(j\,|\,t)) = 1-\sum_{j=1}^{J} p^2(j\,|\,t) \tag{12}$$

Thus Gini index can be considered as a function $\varphi(p_1,\ldots,p_J)$ which in its turn is a second degree polynom with non-negative coefficients. For each convex function it holds that for $\forall\alpha \geq 0$:

$$\begin{aligned}\varphi\left(\alpha p_1+(1-\alpha)p'_1,\ \alpha p_2+(1-\alpha)p'_2,\ \ldots,\ \alpha p_J+(1-\alpha)p'_J\right) > \\ > \alpha\varphi(p_1,\ldots,p_J)+(1-\alpha)\varphi(p'_1,\ldots,p'_J)\end{aligned}$$

Since

$$\begin{aligned}\varphi(\alpha p_1+(1-\alpha)p'_1,\ldots,\alpha p_J+(1-\alpha)p'_J) = \left[1-\alpha\sum_{j=1}^{J}p_j^2\right]+ \\ +\left[1-(1-\alpha)\sum_{j=1}^{J}{p'_j}^2\right] = 2-\alpha\sum_{j=1}^{J}p_j^2-(1-\alpha)\sum_{j=1}^{J}{p'_j}^2\end{aligned}$$

and

$$\begin{aligned}\alpha\varphi(p_1,\ldots,p_J)+(1-\alpha)\varphi(p'_1,\ldots,p'_J) = \left[\alpha-\alpha\sum_{j=1}^{J}p_j^2\right]+\left[(1-\alpha)-(1-\alpha)\sum_{j=1}^{J}{p'_j}^2\right] = \\ = 1-\alpha\sum_{j=1}^{J}p_j^2-(1-\alpha)\sum_{j=1}^{J}{p'_j}^2\end{aligned}$$

we conclude

$$2-\alpha\sum_{j=1}^{J}p_j^2-(1-\alpha)\sum_{j=1}^{J}{p'_j}^2 > 1-\alpha\sum_{j=1}^{J}p_j^2-(1-\alpha)\sum_{j=1}^{J}{p'_j}^2$$

that is why function φ is convex.

This property of Gini index is quite important. Since $\varphi(p_1,\ldots,p_J)$ is a convex function and $p_L+p_R=1$, we get:

$$\begin{aligned}i(t_L)p_L+i(t_R)p_R = \varphi\left(p(1\,|\,t_L),\ldots,p(J\,|\,t_L)\right)p_L+\varphi\left(p(1\,|\,t_R),\ldots,p(J\,|\,t_R)\right)p_R \leq \\ \leq \varphi\left(p_Lp(1\,|\,t_L)+p_Rp(1\,|\,t_R),\ldots,p_Lp(J\,|\,t_L)+p_Rp(J\,|\,t_R)\right)\end{aligned}$$

where inequality becomes an equality in case $p(j\,|\,t_L)=p(j\,|\,t_R)$, $j=\overline{1,J}$.

Recall that

$$\frac{p(j,t_L)}{p(t)} = \frac{p(t_L)}{p(t)} \cdot \frac{p(j,t_L)}{p(t_L)} = p_L \cdot p(j|\, t_L)$$

and since

$$p(j|\, t) = \frac{p(j,t_L) + p(j,t_R)}{p(t)} = p_L p(j|\, t_L) + p_R p(j|\, t_R)$$

we can conclude that

$$i(t_L)p_L + i(t_R)p_R \leq i(t) \tag{13}$$

Hence each variant of data split leads to $\Delta i(s,t) > 0$ unless $p(j|\, t_R) = p(j|\, t_L) = p(j|\, t)$ i.e. when even the best available univariate filter can not decrease class heterogeneity.

Given the way how Gini index is computed it becomes obvious that this impurity measure can be quite effective. First, it is relatively cheap in terms of computation speed and second, as it was mentioned before, Gini index is relatively robust to the effect of outliers – a few outliers can not drastically change the values of $p(j|\, t)$, $j = \overline{1,J}$, hence s^* is not affected.

But of course impurity measure can be defined in a number of different ways, for practical applications a so called *twoing rule* should also be considered.

Its idea is completely different. Instead of looking for maximization of impurity measure change at a particular node, twoing rule tries to balance constructed tree in a special way as if learning sample had only two classes. The reason for such an algorithm is that a decision rule based on twoing criterion is able to distinguish observations between general factors on top levels of the tree and take into account specific data characteristics at lower levels.

If $S = \{1, \ldots, J\}$ is a set of learning sample classes, let's divide it into two subsets

$$S_1 = \{j_1, \ldots, j_n\}\,, \text{ and } S_2 = S \backslash S_1$$

so that all observations belonging to S_1 get dummy class 1, and the rest – dummy class 2.

The next step is to calculate $\Delta i(s,t)$ for different s *as if there were only two dummy classes*. Since actually $\Delta i(s,t)$ depends on S_1, the value $\Delta i(s,t,S_1)$ is maximized. That is why we get a *two-step procedure*: first, find $s^*(S_1)$ maximizing $\Delta i(s,t,S_1)$ and second, find a *superclass* S_1^* maximizing $\Delta i(s^*(S_1),t,S_1)$.

In other words the idea of twoing criterion is two find such a combination of superclasses at each node as if impurity increment was maximized for the data only with two classes $S = \{1,2\}$.

This method provides one big advantage: it finds so called *strategic nodes* i.e. nodes filtering observations in the way that they are different to the maximum feasible extent.

Although applying twoing rule may seem to be desirable especially for data with a big number of classes, another challenge can arise, namely computational speed. Let's assume that learning sample has J classes, then a set S can be split into S_1 and S_2 by 2^{J-1} ways. For 11 classes data in learning sample it will create more than 1000 combinations. Fortunately there is a result helping to reduce drastically the amount of necessary computations.

It can be proven [1] that in a classification task with two classes and impurity measure $p(1|t)p(2|t)$ for an arbitrary split s a superclass $S_1(s)$ is determined as following:

$$S_1(s) = \{j : p(j|t_L) \geq p(j|t_R)\},$$

$$\max_{S_1} \Delta i(s,t,S_1) = \frac{p_L p_R}{4}\left[\sum_{j=1}^{J} |p(j|t_L) - p(j|t_R)|\right]^2 \tag{14}$$

Hence twoing rule can be easily applied in practice as well as Gini index, although the first criterion works a bit slower.

1.5 Gini index and twoing rule in practice

In this section we will conclude the overview of two most popular impurity measures for classification trees by looking at practical issues of using these two rules.

Consider[1] some learning dataset wih 400 observations characterizing automobiles: their make, type, color, technical parameters, age etc. The aim is to build a decision tree splitting different cars by their makes basing on other feasible relevant parameters.

Look at this classification tree constructed using Gini index.

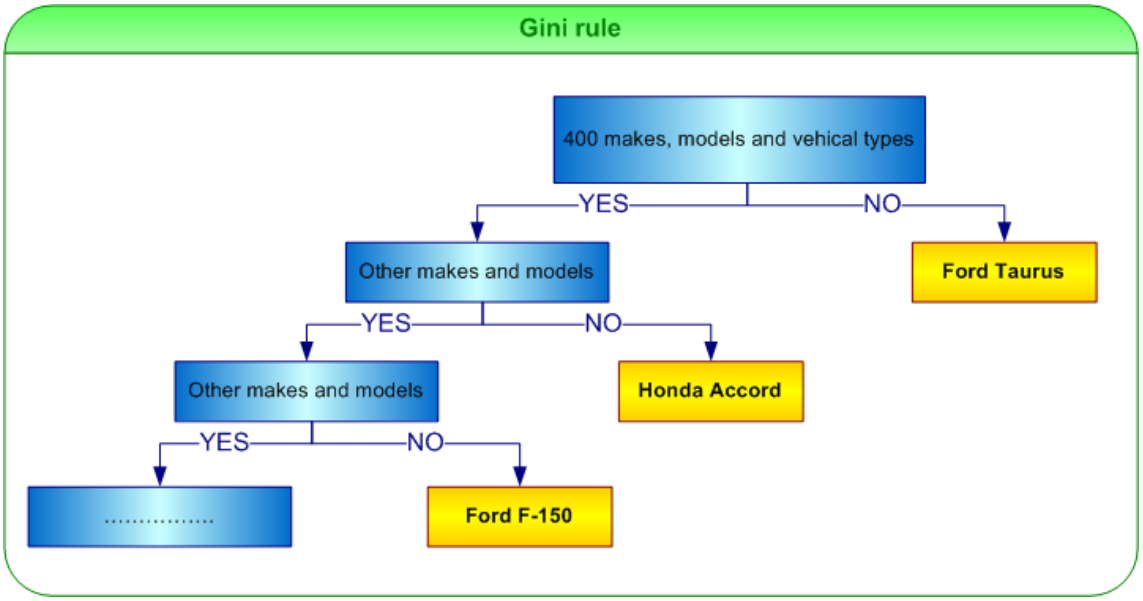

Figure 4: Classification tree constructed by Gini index

A particular feature here is that at each node observations belonging to one make are filtered i.e. observations with most striking characteristics are separated. As a result a decision tree is able to pick out autmobile makes quite easily.

[1]Example is taken from *Critical Features of High Performance Decision Trees* Salford Systems advertisement leaflet

The next tree (Figure 5) for the same data is somewhat different.

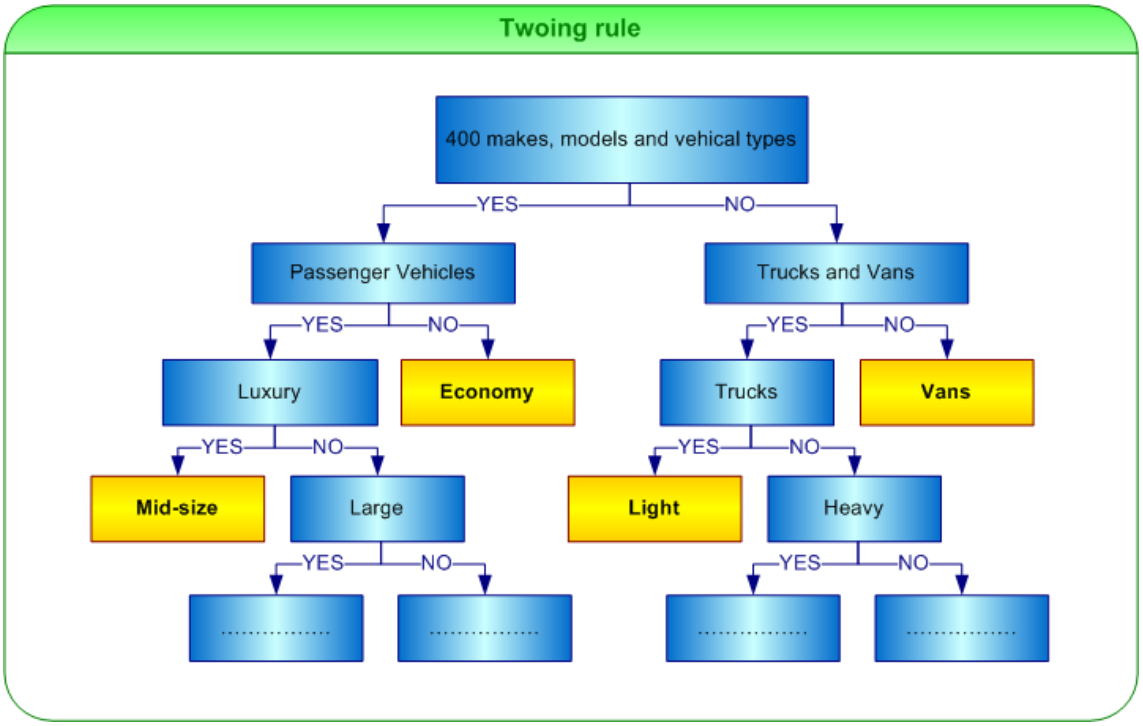

Figure 5: Classification tree constructed by twoing index

Instead of specifying particular car makes at each node, application of twoing rule results in demonstration of strategic nodes i.e. questions which distinguish between different car classes to the maximum extent. This feature can be vital when high-dimensional datasets with a big number of classes are processed. A typical example is speech recognition problem – every word can be coded with a new class, then if twoing rule is applied, classification tree probably will split different words by the number of syllables: with one and more than one syllables. At the next step other similar words' characteristics will be probably taken into account.

For financial applications some researchers claim that Gini index is more preferable than twoing rule. It is impossible, of course, to derive such an absolute dominance theoretically, but some simulations which we carried out for DAX data showed that such dominance *can exist* at least for particular datasets.

1.6 Optimal size of a decision tree: overview of available methods

By this moment we were interested in determining the best split s^* at a particular node. The next and perhaps more important question is how to determine the optimal tree size i.e. when to *stop splitting*. Why is this important?

Consider the application of some impurity measure recursively to some dataset. In limiting case it is possible to stop splitting once the node has only observations of the same class i.e. when there is no advantage of splitting data anymore. If each terminal node has only class homogenous dataset, then such kind of tree accounts for all data variations: every point of the learning sample can be flawlessly classified using this *maximum tree*. But can be such approach fruitful in financial practice?

In classification procedures terms *underspecification* and *overspecification* are frequently used. The direct use of maximum tree is the obvious case of overspecification. This implies that learning sample is characterized *absolutely* by a decision tree i.e. without any errors. However in most cases application of maximum tree to real data results in severe errors. The reason is that maximum classification or regression tree accounts for *any*, even small and insignificant data variations which can be caused by random shocks or measurement errors. That is why when an unclassified observation is processed using maximum tree, with a high probability it can follow to a terminal node describing such kind of disturbance. Hence the recommendation of a rule will be obviously biased. That is why a maximum tree usually tries to explain purely random effects using factor space of the learning sample. But such an explanation is usually only spurious, moreover – nobody could guarantee that future random distirbances are accounted in the same way. There is a small probabiltity that when classifying a new observation it follows to "fundamental" part of the tree which in some cases can be of course a subset of terminal nodes but there is no way to regularize this event.

On the other hand a too small or simple decision rule is not a panacea. In this case significant relationships probably could not be revealed since only a few iterations were used to split the dataset. Hence decision rules become too rough and possibly do not account some fundamental data relationships.

Thus some special criterion is required to stop data splitting. Since tree building is mostly dependent on $\Delta i(s,t)$, the easiest way is to involve this variable into a rule. When data splitting becomes useless in terms of $\Delta i(s,t)$? The answer is quite obvious: in case $\Delta i(s,t) = 0$. But $\Delta i(s,t) = 0$ usually means the limiting case or approaching the maximum tree, hence the primary criterion could be of the following form – stop data splitting if

$$\Delta i(s,t) < \bar{\beta} \tag{15}$$

where $\bar{\beta}$ is some threshold value. So if for a subset of data all possible splits were tried (recall that there is always a finite number of splits) and $\max\limits_{s} \Delta i(s,t) < \bar{\beta}$ then no splitting is conducted. It is worth pointing out that setting $\bar{\beta} = 0$ is equivalent to building the maximum tree.

The value of $\bar{\beta}$ is unknown and could be determined from data simulations, but unfortunately this is not the only drawback of the method. Empirical simulations conducted by many researches show that impurity increment is frequently non-monotone, that is why even for small $\bar{\beta}$ decision tree may be underparametrized. Setting even smaller values for $\bar{\beta}$ will probably remedy the situation but at cost of tree overparametrization. Setting high values for $\bar{\beta}$ significantly increases the risk of underparametrization

explicitely and since there can be a high value of impurity increment preceded by a value smaller than $\bar{\beta}$, the situation can become even worse.

Another possible way to determine the adequate shape of a decision tree is to set up a restriction on the minimum number of observations $\bar{N}$ at each terminal node. If at terminal node t the number of observations is higher

$$N(t) > \bar{N} \tag{16}$$

then this node is also being split as data are still not supposed to be clustered well enough. Although the dynamic of $N(t)$ is *always monotone* due to the way question values are selected, the problem of estimation of $\bar{N}$ is still remaining. On the other hand, using historical simulations for financial data or artificial samples it is sometimes possible to estimate a more or less robust level of $\bar{N}$ that for some cases results in better productivity and even overall efficiency comparing with more advanced and demanding methods. Difficulties with estimation of $\bar{N}$ suggest the usage of some criterion which would not share described drawbacks and particularly would not require any *a priori* information like $\bar{N}$ or $\bar{\beta}$.

Thus a procedure called *cross-validation* is usually employed.

1.7 Cross-validation as a method of optimal decision tree pruning

Usually cross-validation implies a procedure which uses available data in the way so that the bigger part of them is employed as a *training set* and the rest – as a *test set*. Then the process is looped so that different parts of the data become learning and training set, so that at the end each datapoint was employed both as a member of test and learning sets. The aim of this procedure is to extract maximum information from the learning sample especially in the situations of data scarceness.

The procedure is implemented in the following way. First, learning sample is *randomly* divided into V parts – after that $(V-1)$ refer to training set and one part – to test set. Using training set a decision tree is constructed while the rest of the data (test set) is used to verify the tree quality since its actual class/response value is known from learning sample. At the next step the pool of data which was used as test set becomes a part of learning set whereas another $\frac{1}{V}$-th part of the data becomes a test set. The loop stops when all data points were employed in such a way i.e. maximum information from the data was extracted.

The aim of cross-validation is to compare the *quality of the tree* in different configurations i.e. trees of different size. Define $L \backslash L_v$, $\forall v = \overline{1, V}$ as the training set and L_v as the test set where L is the learning sample itself. For a given classification rule d_v basing on learning set $L \backslash L_v$ it is then possible to estimate its quality in the following way:

$$E^1\left(d^{(v)}\right) = \frac{1}{N_v} \sum_{(l_n, j_n) \in L_v} I\left(d^{(v)}(l_n) \neq j_n\right) \tag{17}$$

where l_n is a test set observation with class j_n and $E^1\left(d^{(v)}\right)$ is a one-iteration estimate.

Since none of the observations from L_v was engaged during construction of decision rule $d^{(v)}$, it is then possible to define the *cross-validation measure of tree quality* as

$$E^{CV}(d) = \frac{1}{V}\sum_{v=1}^{V} E^1\left(d^{(v)}\right) \tag{18}$$

The next important point is how to choose V. Although V is not an internal calibration parameter like $\bar{\beta}$, it is still quite important because this parameter is the key factor in speed-precision trade-off. It is worth noting that cross-validation can be *extremely slow* for big N and V, hence an adequate balance is required. Usually the value of V can be specified given the precise task formulation where time constraint becomes extremely important. Imagine that an online classification system is required e.g. for classifying different high-frequency stock exchange operations, then if one classification takes say several minutes and time constraint is only one second – cross-validation is obviously the wrong algorithm to apply. Nonetheless, usually time constraints are not so extremely tough, that is why the choice of V is mainly dependent on feasible computing power.

Unfortunately for small values of V cross-validation estimates can be *unstable* since each iteration a cluster of data is selected *randomly* and the number of iterations itself is relatively small, thus the overall estimation result is somewhat random. Empirical simulations showed that e.g. for DAX30 stock selection classification problem (see next part of the study) where the individual stock average yield was about 25%, the deviation caused by this kind of randomness was about 10% which is, of course, absolutely inadequate.

Since $N\left(1-\frac{1}{V}\right) \to N$ for big $V \leq N$ the only way out can be to increase V. In the limiting case where $V = N$ randomness obviously disappears but only at cost of overwhelmingly increased amount of computations. For practical financial applications with high-dimensional datasets significant increase of V can be not feasible unless supercomputers are employed.

Nowadays cross-validation with $V = 10$ is an industry standard and for many applications different researches claim that result robustness is at acceptable level, hence we can conclude that cross-validation can be recommended as the primary method for decision tree optimizations.

But employing cross-validation method itself for all possible tree configurations is also not feasible due to computational constraints. Here the question arises – is it possible to check not all subtrees of the maximum tree but only special *key subtrees*? With results introduced in [1] it appeared to be possible.

1.8 Cost-complexity function and cross-validation

The idea of the method is to introduce some new measure that would be able to take into account *tree complexity* i.e. its size which can be estimated by the *number of terminal nodes*: then maximum tree will get a penalty for its big size, on the other hand it will be able to make perfect in-sample predictions. Small trees will, of course, get much lower

penalty for their size but their predicting abilities are naturally limited. Optimization procedure based on such trade-off criterion could determine the best decision tree size.

Define *internal misclassification error* of an arbitrary observation at node t as $e(t) = 1 - \max_j p(j|t)$, define also $E(t) = e(t)p(t)$. Then *internal misclassification tree error* is $E(T) = \sum_{t \in \tilde{T}} E(t)$ where $\tilde{T}$ is a set of terminal nodes.

This estimates are called *internal* because they are based solely on the learning sample on the contrary to cross-validation which artificially introduces both learning and test sets. It may seem that using $E(T)$ as tree quality measure is sufficient but unfortunately it is not so. Consider the case of maximum tree where $E(T_{MAX}) = 0$ i.e. the tree is of best configuration. As it was discussed above, maximum tree can represent optimal decision rules only in quite rare cases.

For any subtree $T \leq T_{MAX}$ define the number of terminal nodes $\left|\tilde{T}\right|$ as a measure of its complexity. Then the following cost-complexity function could be used to optimize decision tree size:

$$E_\alpha(T) = E(T) + \alpha \left|\tilde{T}\right| \tag{19}$$

where $\alpha \geq 0$ is a complexity parameter and $\alpha \left|\tilde{T}\right|$ is cost component: the more complex is the tree (the higher is the number of terminal nodes) – the lower is $E(T)$ but at the same time the higher is the penalty $\alpha \left|\tilde{T}\right|$ and vice versa.

Alhough α can have infinite number of values, the number of subtrees of T_{MAX} resulting in minimization of $E_\alpha(T)$ is *finite*. Hence pruning of T_{MAX} leads to creation of subtrees sequence $T_1, T_2, T_3, \ldots$ with a decreasing number of terminal nodes. Since the sequence is finite, if $T(\alpha)$ is an optimal subtree for some arbitrary α, then it will remain optimal until complexity parameter is not changed to some α' when $T(\alpha')$ becomes a new optimal subtree until complexity parameter value is α'' and so on.

The main question is if the optimal subtree $T \leq T_{MAX}$ for a given α minimizing $E_\alpha(T)$ always *exists* and if it is *unique*? Moreover, for the reasons of computational efficiency which are usually crucial in financial applications we are interested if the sequence of optimal subtrees for different values of α is *nested* i.e. $T_1 \succ T_2 \succ \ldots \succ \{t_0\}$ where t_0 is the root node (learning sample itself)? In this case the number of subtrees to check is obviously reduced drastically.

In [1] it is shown that for $\forall \alpha \geq 0$ there exists an optimal tree $T(\alpha)$ in the sense that

1. $E_\alpha\left(T(\alpha)\right) = \min_{T \leq T_{MAX}} E_\alpha(T) = \min_{T \leq T_{MAX}} \left[E(T) + \alpha \left|\tilde{T}\right|\right]$
2. if $E_\alpha(T) = E_\alpha\left(T(\alpha)\right)$ then $T(\alpha) \leq T$.

This result is then not only proof of existance, but also a proof of uniqueness: consider another optimal subtree T' so that T and T' both minimize E_α and are *not nested*, then $T(\alpha)$ does not exist in accordance with second condition.

The idea of introducing cost-complexity function at this stage is to check only a subset of different subtrees of T_{MAX}: optimal subtrees for different values of α. The starting point is to define the first optimal subtree in the sequence so that $E(T_1) = E(T_{MAX})$ and the size of T_1 is minimum among other subtrees with the same cost level. To get T_1 out of T_{MAX} for each terminal node of T_{MAX} it is necessary to verify the condition $E(t) = E(t_L) + E(t_R)$ and if it is fulfilled – node t is pruned. The process is looped until no extra pruning is available – the resulting tree $T(0)$ becomes T_1.

Define node t as *ancestor* of t' and t' as *descendant* of t if there is a connected path down the tree leading from t to t'.

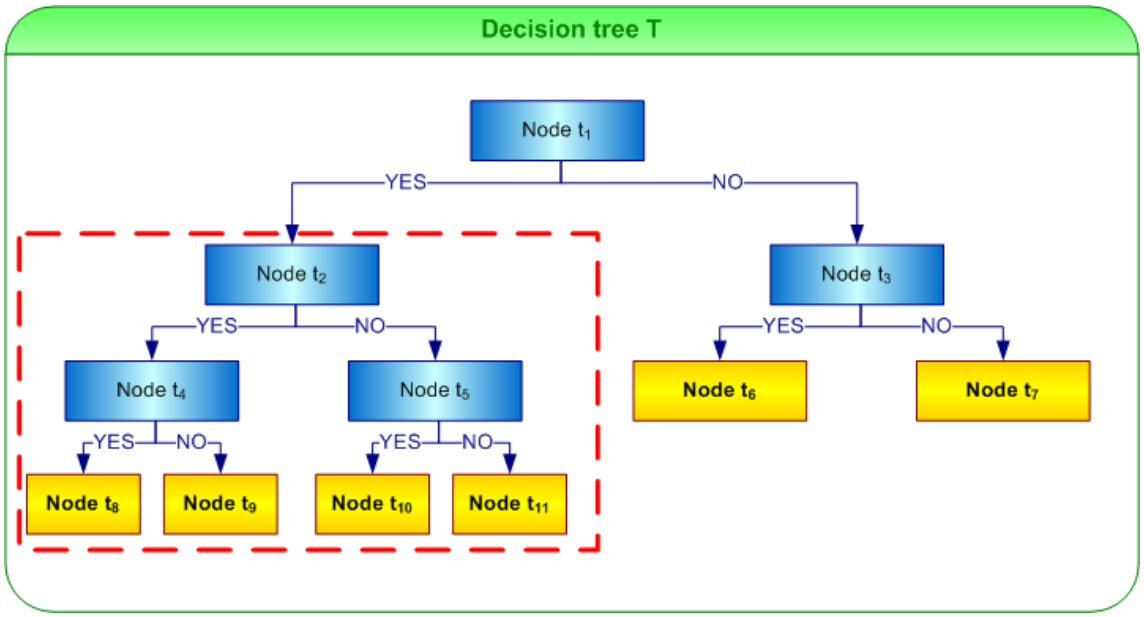

Figure 6: Decision tree hierarchy

In this example nodes t_4, t_5, t_8, t_9, t_{10} and t_{11} are descendants of t_2 while nodes t_6 and t_7 are not descendants of t_2 although they are positioned lower since it is not possible to connect them with a path from t_2 to these nodes without engaging t_1. Analogously nodes t_4, t_2 and t_1 are ancestors of t_9 and t_3 is not ancestor of t_9.

Define the *branch* T_t of the tree T as a subtree based on node t and all its descendants. For the example above marked area represents the branch T_{t_2}. This brach can be considered as a separate tree.

Pruning a branch T_t from a tree T means deleting all descendant nodes of t. Denote transformed tree as $T - T_t$. For our example pruning the branch T_{t_2} will result in a new tree on Figure 8.

Now for any branch T_t define an *internal misclassification estimate* as:

$$E(T_t) = \sum_{t' \in \tilde{T}_t} E(t') \tag{20}$$

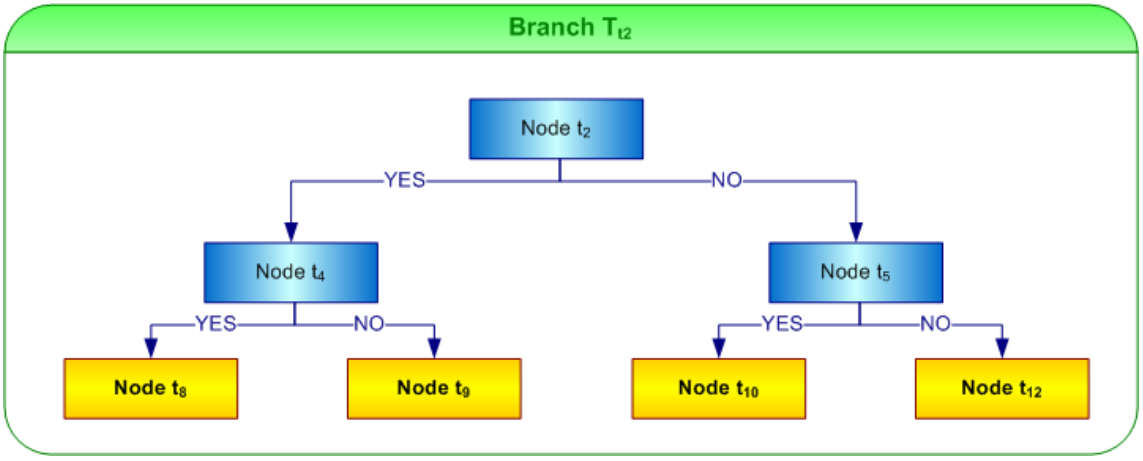

Figure 7: The branch T_{t_2} of original tree T

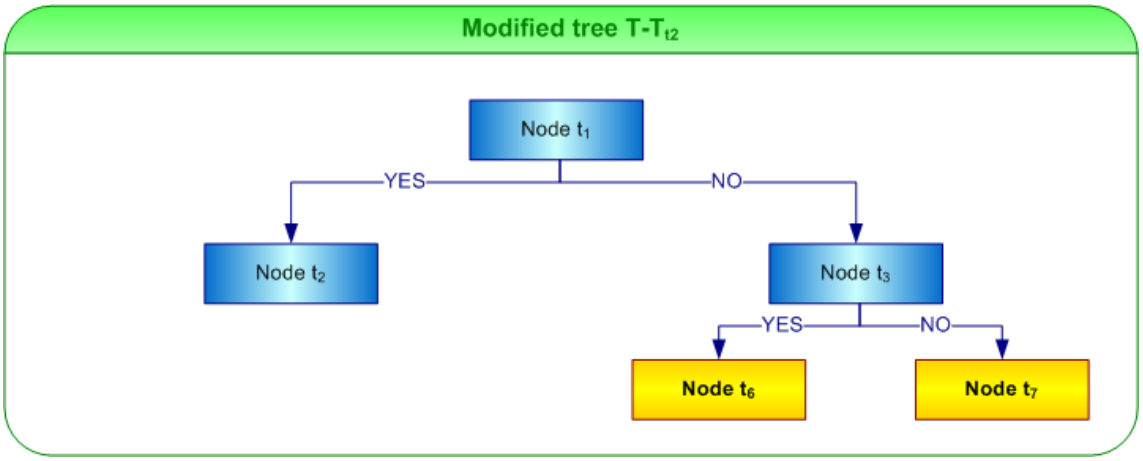

Figure 8: $T - T_{t_2}$ – pruned tree T

where $\tilde{T}_t$ is the set of terminal nodes of T_t. Hence for arbitrary node t of T_1 is true that

$$E(t) > E(T_t) \tag{21}$$

Consider now *cost-complexity misclassification estimate* for branches or single nodes. Define a single node estimator as

$$E\left(\{t\}\right) = E(t) + \alpha \tag{22}$$

where $\{t\}$ is a subtree consisting of signle node t and branch estimate as

$$E_\alpha(T_t) = E(T_t) + \alpha \left|\tilde{T}_t\right| \tag{23}$$

When $E_\alpha(T_t) < E_\alpha\left(\{t\}\right)$ the branch T_t is preferred to single node $\{t\}$ according to

cost-complexity misclassification estimation. But for some critical α both values will become equal – this critical value of α can be determined from the following inequality:

$$E_\alpha(T_t) < E_\alpha(\{t\}) \tag{24}$$

which is equivalent to

$$\alpha < \frac{E(t) - E(T_t)}{\left|\tilde{T}_t\right| - 1} \tag{25}$$

where $\alpha > 0$ since $E(t) > E(T_t)$

To get the next member of optimal subtrees sequence i.e. T_2 out of T_1 a special node called *weak link* is determined. For this purpose a function $g_1(t)$, $t \in T_1$ is defined as

$$g_1(t) = \begin{cases} \frac{E(t)-E(T_t)}{\left|\tilde{T}_t\right|-1}, & t \notin \tilde{T}_1 \\ +\infty, & t \in \tilde{T}_1 \end{cases} \tag{26}$$

Then node $\bar{t}_1$ is a weak link in T_1 if

$$g_1(\bar{t}_1) = \min_{t \in T_1} g_1(t) \tag{27}$$

and new value for α_2 is defined as

$$\alpha_2 = g_1(\bar{t}_1) \tag{28}$$

New tree $T_2 \prec T_1$ in the sequence is obviuosly defined by pruning the branch $T_{\bar{t}_1}$ i.e.

$$T_2 = T_1 - T_{\bar{t}_1} \tag{29}$$

The process is looped until root node $\{t_0\}$ – the final member of sequnce – is reached.

When there are multiple weak links detected, for instance $g_k(\bar{t}_k) = g_k(\bar{t}'_k)$, then both branches are pruned i.e. $T_{k+1} = T_k - T_{\bar{t}_k} - T_{\bar{t}'_k}$

In this way it is possible to get the sequence of optimal subtrees $T_{MAX} \succ T_1 \succ T_2 \succ T_3 \succ \ldots \succ \{t_0\}$ for which it is possible to prove that the sequence $\{\alpha_k\}$ is increasing i.e. $\alpha_k < \alpha_{k+1}$, $k \geq 1$ and $\alpha_1 = 0$. For $k \geq 1$: $\alpha_k \leq \alpha < \alpha_{k+1}$ and $T(\alpha) = T(\alpha_k) = T_k$.

Practically this tells us how to implement the search algorithm. First, maximum tree T_{MAX} is taken, then T_1 is found after what weak link $\bar{t}_1$ is detected and branch $T_{\bar{t}_1}$ is pruned off, α_2 is calculated and the process is looped.

When the algorithm is applied to T_1, the number of pruned nodes is usually quite significant. For instance, consider the following typical empirical evidence:

Tree	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
$\left\|\tilde{T}_k\right\|$	71	63	58	40	34	19	10	9	7	6	5	2	1

Table 1: Typical pruning speed

When the trees become smaller, the difference in the number of terminal nodes also gets smaller.

Finally, it is worth mentioning that the sequence of optimally pruned subtrees is a subset of trees which might be constructed using direct method of internal misclassification estimator minimization given a fixed number of terminal nodes. Consider an example of tree $T(\alpha)$ with 7 terminal nodes, then there is no other subtree T with 7 terminal nodes having lower $E(T)$. Otherwise

$$E_\alpha(T) = E(T) + 7\alpha < E_\alpha\left(T(\alpha)\right) = \min_{T \leq T_{MAX}} E_\alpha(T)$$

which is impossible by definition.

Applying the method of V-fold cross-validation to the sequence $T_{MAX} \succ T_1 \succ T_2 \succ T_3 \succ \ldots \succ \{t_0\}$, an *optimal tree* is determined.

On the other hand it is frequently pointed out that choice of tree with minimum value of $E^{CV}(T)$ is not always adequate since $E^{CV}(T)$ is not too robust i.e. there is a whole range of values $E^{CV}(T)$ satisfying $E^{CV}(T) < E^{CV}_{MIN}(T) + \varepsilon$ for small $\varepsilon > 0$. Moreover, when $V < N$ a simpe change of random generator seed will definitely result in changed values of $\left|\tilde{T}_k\right|$ minimizing $\hat{E}(T_K)$. Hence a so called *one standard error* empirical rule is applied which states that if T_{k_0} is the tree minimizing $E^{CV}(T_{k_0})$ from the sequence $T_{MAX} \succ T_1 \succ T_2 \succ T_3 \succ \ldots \succ \{t_0\}$, then a value k_1 and a correspondent tree T_{k_1} are selected so that

$$\underset{k_1}{\operatorname{argmax}}\, \hat{E}(T_{k_1}) \leq \hat{E}(T_{k_0}) + \sigma\left(\hat{E}(T_{k_0})\right) \tag{30}$$

where $\sigma(\cdot)$ denotes sample estimate of standard error and $\hat{E}(\cdot)$ – the relevant sample estimators.

The dotted line on Figure 9 shows the area where the values of $\hat{E}(T_k)$ only slightly differ from $\min_{|\tilde{T}_k|} \hat{E}(T_k)$. The left edge which is roughly equivalent to 12 terminal nodes shows the application of one standard error rule.

The use of one standard error rule allows not only to achieve more robust results but also to get trees of lower complexity given the error comparable with $\min_{|\tilde{T}_k|} \hat{E}(T_k)$.

1.9 Regression trees: what is the difference?

By this moment we mainly concentrated on decision trees as a structure and covered some aspects of classification trees creation. Although *regression trees* share similar logic, there are some important peculiarities which should be covered not forgetting, of course, the technical aspects of regression trees building.

Recall that the only difference between classification and regression tree is the type of dependent variable. When it is discrete, a decision tree is called classification, regression tree is a decision tree with a *continuous* dependent variable.

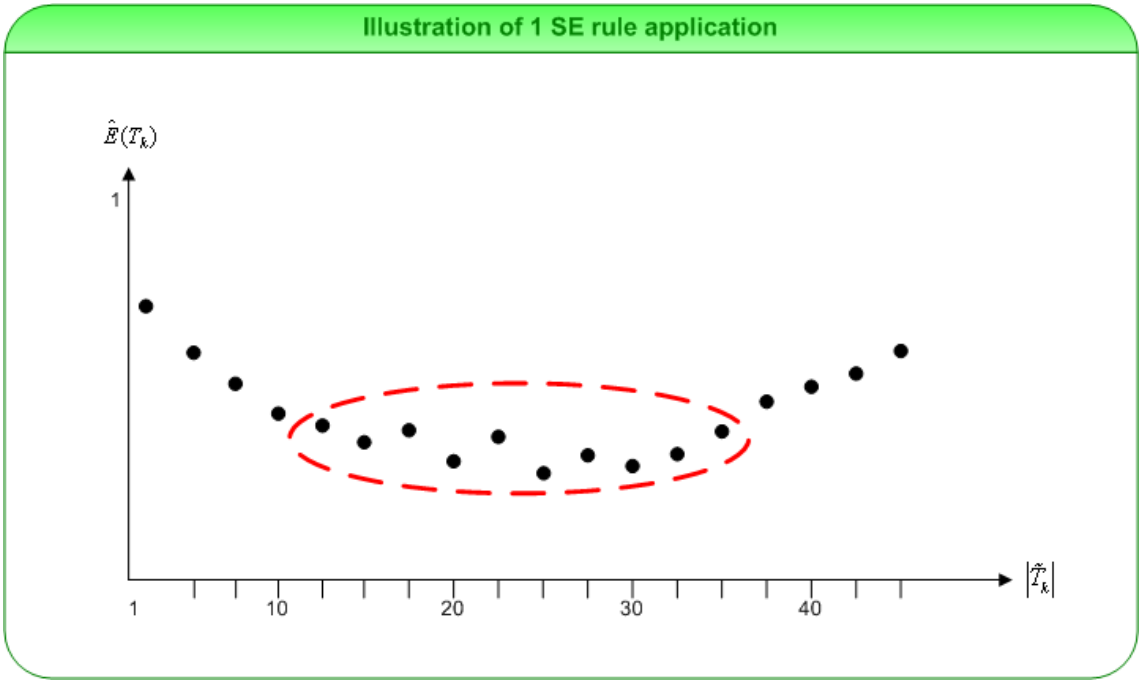

Figure 9: The example of relationship between $\hat{E}(T_k)$ and number of terminal nodes

Main stages of classification trees creation remain the same for regression ones. First and foremost, there should be some criterion worked out so that learning sample data can be efficiently split. At the second stage, the maximum regression tree should be efficiently pruned as well.

Gini index and twoing rule discussed in previous sections assume that the number of classes is finite and hence introduce some measures based mainly on $p(j|t)$ for arbitrary class j and node t. But since in case of continuous dependent variable there are no more classes, this approach cannot be used anymore unless groups of continuous values are effectively substituted with artificial classes.

Since there are no classes anymore – how can be the maximum regression tree determined? Analogously with discrete case, absolute homogeneity can be then described only after some adequate impurity measure for regression trees is introduced.

Recall the idea of *Gini index*, then it becomes quite natural to use the *variance* as impurity indicator. Since for each node data variance can be easily computed, then splitting criterion for an arbitrary node t can be written as

$$s^* = \underset{s}{\operatorname{argmax}} \left(p_L \operatorname{var} \left(t_L(s) \right) + p_R \operatorname{var} \left(t_R(s) \right) \right) \tag{31}$$

where t_L and t_R are emerging child nodes which are, of course, directly dependent on the choice of s^*.

Hence maximum regression tree can be easily defined as a structure where each node has only the same predicted values. It is important to point out that since continuous

data have much higher chances to take different values comparing with discrete ones, the size of maximum regression tree is usually very big.

When maximum regression tree is properly defined, it is then of no problem to get an optimally-size tree. Like with classification trees, maximum regression tree is usually supposed to be upwardly pruned with the help of cost-complexity function and cross-validation. That is why the majority of results presented above is applied to regression trees as well.

2 CART in practice

2.1 Regression trees vs classification trees in financial applications

It may seem plausible to use regression or classification trees in accordance with type of dependent variable. However, empirical results does not necessarily support this point of view.

In many occasions classification trees can be preferred to regression ones. It usually happens because of not sufficiently high degree of regression trees robustness in terms of classification rule structure. As it was mentioned before, because of possible extremely high complexiy of maximum regression tree the derived rules tend to be *overparametrized* i.e. even optimally derived size of the regression tree is too big.

The simplest example to consider is to look at two linearly separable datasets which are though can not be obviously separated by orthogonal hyperplanes. Then even if true decision rule is just a linear inequality, regression tree will reproduce it only at cost of high complexity. In case when learning sample data are relatively noisy, this will add some more obstacles as we will see below.

What can be the typical classification task of a financial application? It can be a direct classification like CRM (customer relationship management), insurance class diagnostics etc. or the task may be, say, to construct a profitable portfolio of stocks basing on some extra evidence like technical and fundamental data. In the last case an investor is obviously interested in the future stock prices i.e. some forecasts are to be made. Since stock price is a continuous variable, the obvious way is to use regression tree to represent a future stock price estimate. But for the reasons mentioned above the characteristics of such an estimate can be quite far from the optimal one. That's why one of the ways to overcome the problem is to somehow substitute a regression tree with a classification one. The effect will probably be even more bigger if the number of classes of a modified dataset is relatively moderate.

This can be achieved by introducing *artificial classes* i.e. classes based on dependent variable values. For a stock price example three classes can be introduced so that finally an investor gets a decision "long-short-neutral" type rule based on current information set. For this one period relative price increments can be used so that if R_t denotes stock price at t, then the most important value is $\Delta R = \frac{R_{t+1}-R_t}{R_t}$ since new classes set S can be difened as

$$S = I(\Delta R > \bar{R}) - I(\Delta R < \bar{R}) \tag{32}$$

where $\bar{R}$ is an arbitrary non-negative constant, $I(\cdot)$ – indicator function

In other words $S = 1$ implies expected price growth and *long* position, $S = 0$ – non-siginificant future one-period price fluctuations and *neutral* position and $S = -1$ – *short* position. Constant $\bar{R}$ can be set according to investor's risk aversion prefences, historical simulations and/or other *a priori* information.

Empirical simulations with DAX data showed that such approach can dramatically

improve model performance, hence classification trees can effectively substitute regression structures even in case of continuous dependent variable.

2.2 CART challanges in business applications

Recall that by model assumptions decision trees are constructed using only *univariate orthogonal splits*. Although in some cases data seem to be linearly separable, there are lots of situations when it is not true.

There can be several reasons for that, for instance such kind of effect may stand for nonlinear fundamental dependency or learning sample data were measured with errors and thus a lot of random disturbancies could occur. Anyway, in such a situation it can be quite difficult to construct an efficient decision rule.

Let's examine the case of non-linearity first. How does a decision tree normally try to represent that kind of structure?

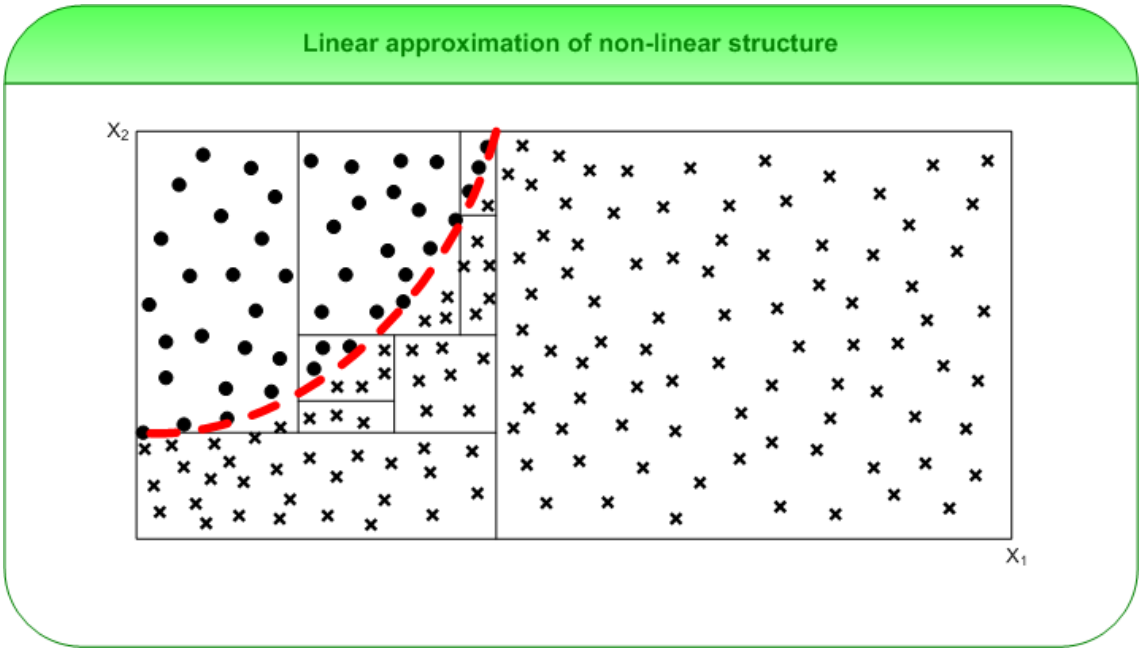

Figure 10: Non-linear separable data

Hence we can conclude that decision trees can quite effectively capture even non-linear dependencies by means of step approximations. In real applications the majority of situations follows this or similar case, but as we can see a decision tree can still be applied there.

The question is if it may be possible to substitute linear filters with *non-linear* and univariate with *multivariate*. Although some properties of resulting decision rules may become better, the cost is too high leaving alone technical issues of such kind of algorithm. The problem is that there is infinite number of different variables combinations and even if one is restricted to a certain class of functions (e.g. linear) it will still be

extremely hard to determine the "correct filter" at a particular node i.e. a filter with a proper dimension. Brute force approach can probably help only for low-dimensional data, so the sitution is similar with derivation of *globally optimal* trees. Supercomputers can, of course, solve the problem.

A more serious issue is the existance of measurement errors and/or highly noised data. Consider the following example on Figure 11.

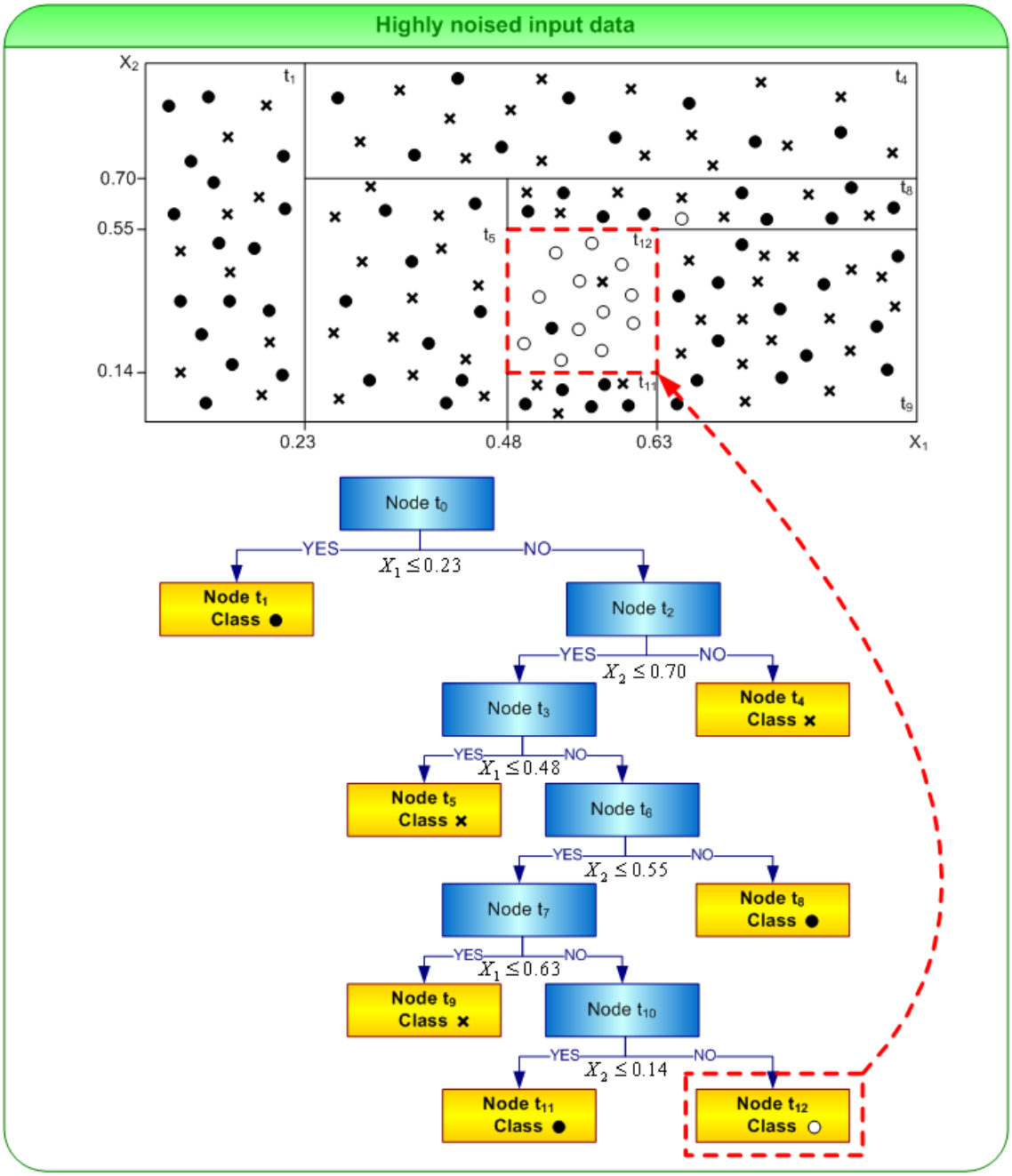

Figure 11: Data with significant random disturbances and a fundamental relationship

This situation is quite typical for financial applications. The majority of data is highly noised although there exists a cluster which is quite homogenous and surely represents some fundamental dependency. This cluster is t_{12} and appears only in the bottom of the tree. Apparently an investor would like to get an explanation for this data portion but is it really feasible using standard methods described above?

Such kind of data pattern will surely result in underparametrization of the final rule i.e. only small number of nodes from tree on Figure 11 will be present. Using 1-SE rule in conjunction with cost-complexity function and cross-validation will typically produce a tree constituted by nodes t_0, t_1, t_2, t_3 and t_4 and hence information in node t_{12} will be lost! That is why traditional approaches does not necessary appear to be versatile and there is an obvious need to develop alternative approaches to surmount obstacles like this.

An alternative method of decision tree pruning was developped by the author. Although its detailed description and overview of theoretical properties is out of the scope of this work, the next section provides an outlook how efficient can alternative approaches be when traditional solutions provide only inadequate results.

2.3 Recursive DAX30 stocks portfolio creation: an example

This example continues the implications started in 2.1 and shows how successful can classification trees be applied even in case of noisy data with non-linear dependencies.

The problem framework is the following. An investor is operating on DAX30 market and possesses a database with technical and fundamental data referring to the DAX30 companies, he (she) also has a time series of historical stock close prices. Investor's task is to create a recursive portfolio out of some or all available DAX30 stocks so that to maximize the wealth and reduce portfolio's possible deviation. At each period an investor can buy/sell/hold a particular stock while each transaction leads to supplementary costs at the rate of 10 b.p. (basis points).

Moreover, investor supposes that there is a certain relationship between technical and fundamental parameters and future stock price, but he(she) has no idea about the form (class) of such mapping (function).

More formally, the task is the following. Let $\bar{M}$ be a fixed amount of cash available to an investor at each time period. This money can be invested into $N(t)$ different DAX30 stocks at time period t. Let R_{it} be the yield of a particular i-th stock due to the one-period change of stock's price. Denote the weight of that i-th stock in portfolio investor should construct as w_{it}. A long/short/neutral position π_{it} against each stock can be taken that is why *control variable is discrete* and can take only one of three possible values out of the positions set $\mathbb{S}$. In order to reduce portfolio risk investor *closes active positions* at the end of each period, hence transaction costs arise:

$$0.001 \cdot \left[I\left\{R_{it} > 0\right\} \cdot I\left\{\pi_{it} = \pi_{i,t-1} \neq 0\right\} - I\left\{R_{it} < 0\right\} \cdot I\left\{\pi_{it} = \pi_{i,t-1} \neq 0\right\} \right] P_{it} Q_{it} \quad (33)$$

where P_{it} – the price and Q_{it} – the quantity of i-th stock in the portfolio in t-th period of time.

Moreover, to reduce the risk of returns investor is supposed to make equally-weighted portfolio i.e.

$$w_{it} = \frac{1}{N(t)} \pi^*_{it}(\cdot) \tag{34}$$

Investor assumes that there is a relationship between R_{it} and an information set $F_{i,t-1}$ of technical and fundamental historical data of relevant companies determined by mapping $f_{i,t}$ but the *precise form of this mapping is unknown* to investor. Nonetheless, there is a set of empirical rules $\mathbb{A}$ concerning some patterns of this relationship and history of stock prices movements available.

Given these assumptions, investor's problem can be stated as:

$$\begin{cases} \sum\limits_{t=\tau}^{T} \sum\limits_{i=1}^{N(t)} w_{it} \left[E\left(R_{it} \middle| F_{i,t-1} \right) \left(1 - I\left\{ R_{it} > 0 \right\} \cdot I\left\{ \pi_{it} = \pi_{i,t-1} \neq 0 \right\} + \right. \right. \\ \left. \left. + I\left\{ R_{it} < 0 \right\} \cdot I\left\{ \pi_{it} = \pi_{i,t-1} \neq 0 \right\} \right) \right] \rightarrow \max\limits_{\pi_{it}} \\ \\ f_{it} : F_{i,(t-1)} \rightarrow R_{it} \\ \pi_{it} \in \mathbb{S} = \{-1; 0; 1\} \\ R_{it} \in \mathbb{A} \\ \sum\limits_{i=1}^{N(t)} \left[1 + 0.001 \cdot I\left\{ \pi_{it} = \pi_{i,t-1} \neq 0 \right\} \right] \cdot \pi_{it} P_{it} Q_{it} = \bar{M} \\ w_{it} = \frac{1}{N(t)} \pi^*_{it}\left(F_{i,t-1} \right) \\ F_{i,T} \supset F_{i,T-1} \supset \ldots \supset F_{i,\tau} \ \forall i = \overline{1, N(t)} \end{cases} \tag{35}$$

where subset $\mathbb{S}$ is described by (32).

As a result of *a priori* investor's expectations $\mathbb{A}$ and other factors an implied estimation of mapping $f_{it} : F_{i,(t-1)} \rightarrow R_{it}$ is carried out. On its basis function $\pi^*_{it}\left(F_{i,t-1} \right) \in \mathbb{S}$ is estimated in the *class of binary decision rules* represented by classification trees.

Hence as a result investor gets a decision rule in class "long/short/neutral" dependent on historical technical and fundamental data of DAX30 companies.

Using actual data it was possible to simulate such kind of strategy. Unfortunately direct application of classical approach didn't result in even positive profit, that's why model 35 was used instead.

The optimal trajectories π^*_{it} were tested for the period of November 27, 2000 – June 7, 2004. After proper calibration was made, investment activity was imitated (Figure 12) basing on calculated optimal strategies. Simulations were implemented in Matlab R13 and C++ by means of created program complex.

Active strategy implies that solutions of (35) were used and capital dynamic was recorded. On the contrary, *passive strategy* stands for holding equally-weighted DAX30 portfolio so that model performance evaluation could be carried out. It is important to point out that equally-weighted DAX30 portfolio and DAX30 index itself have *different yields* because DAX30 is computed in a more complicated way. As another comparison ground, wealth curves for *risk-free* bonds yielding 4% and 9% annually are presented. From the picture it is clearly seen that active strategy outperforms others and is in fact very efficient.

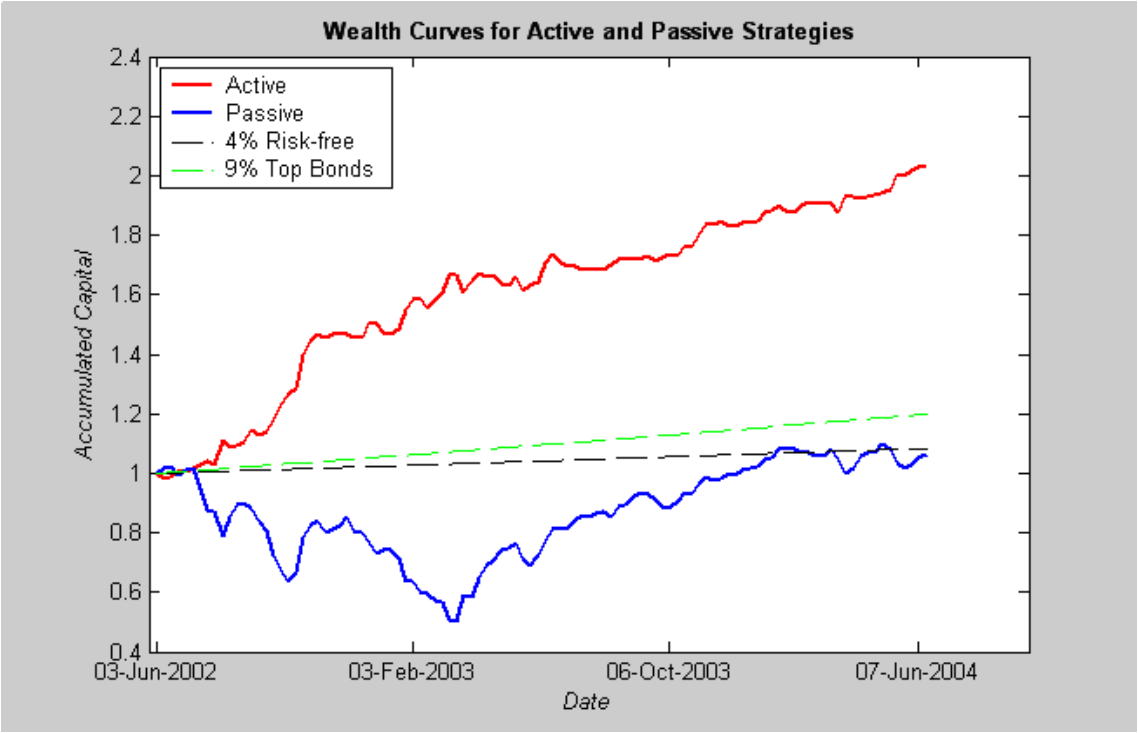

Figure 12: Simulation results

The next two figures show the weekly return distributions for both active and passive strategies. The active strategy distribution clearly implies the positive skewness of the yield.

Average portfolio yield is **51.2673%** while Sharp index is **19.7637** since the yield standard deviation is only **2.5940%**. All the values are expressed in annual terms. The similar study of *CityGroup* [16] which aim was to classify stocks of US technological companies produced on average 19.62% with a standard deviation of 11.96% (Sharp index is 1.23 and risk-free rate is 4.92%). Another study by *JPMorgan* [15] aimed at classification of some US stocks (no more information is available) produced similar results: average yield is 14.6% with a standard deviation of 9.5% (Sharp index is 1.54). Both studies employed traditional approach of classification trees.

We can not, of course, directly compare these results since there are different stock markets involved etc., but on the other hand the timeline is the same and hence we can conclude that at least for the last two years proposed active strategy clearly outperforms investment strategies of *JPMorgan* and *CityGroup* applied to US stocks in *absolute terms*.

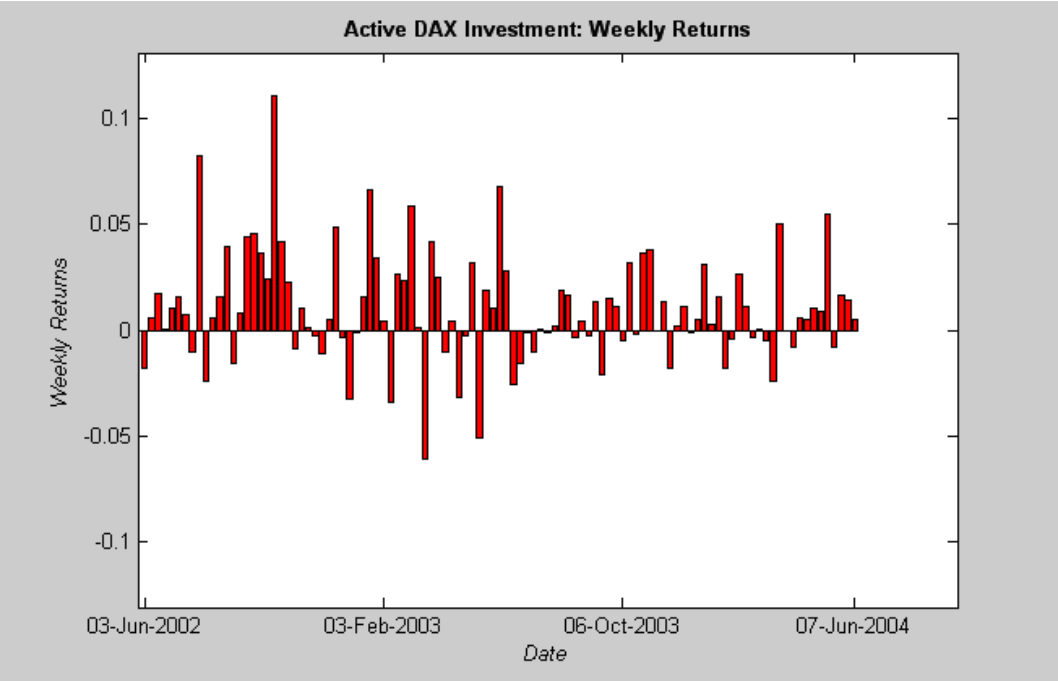

Figure 13: Active strategy weekly returns

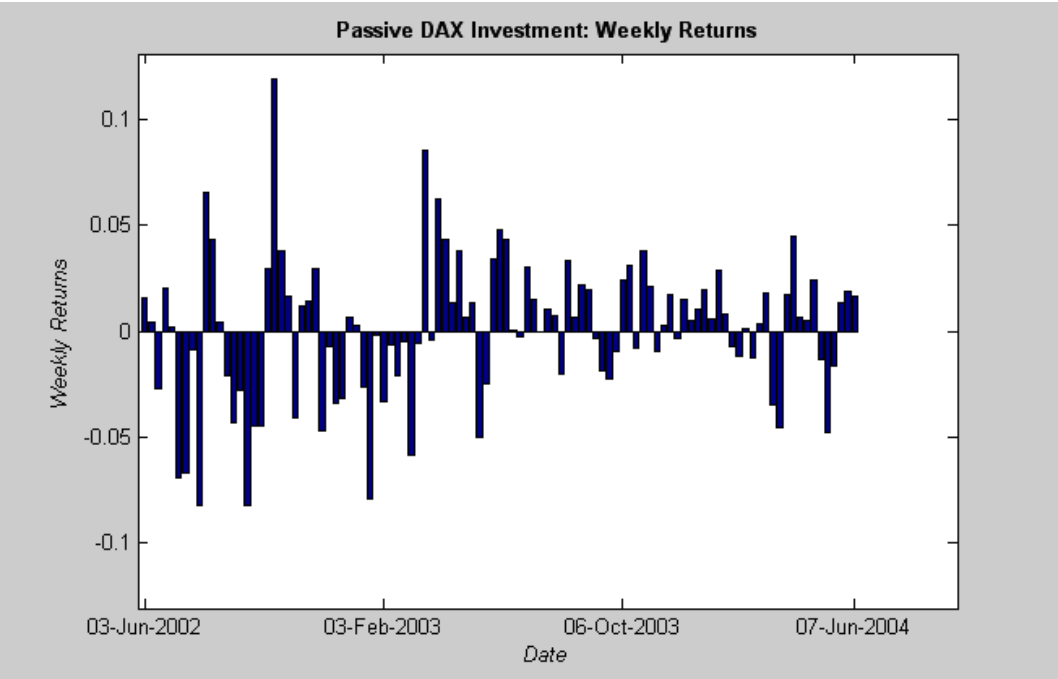

Figure 14: Passive strategy weekly returns

The next issue to analyze is how modified CART algorithms affected the results. For this purpose during the implementation both classical and a set of modified methods were allowed to use and special calibration criterion derived the best algorithm for the moment. The results are the following: in 66.6% of situations modified algorithms were acknowledged to be better comparing with conjuction of cost-complexity function and cross-validation. Moreover, a forced application of classical approach resulted in much lower precision in almost 95% of cases, hence one can conclude that modified CART core plays a prominent role in the showed financial results.

2.4 Technical vs fundamental analysis: example continued

Although performance itself may be vital for financial engineering, there are numerous situations when classification trees provide another valuable information. We will continue to work with the same dataset and try to find out which variables appeared to be most signficant in constructed trees and how this result can affect the impression of technical and fundamental analysis relevance.

The next table clues up about available variables used in the analysis.

Variable name	Variable type	Regulariry estimate
Close price	Fund./Tech.	1 day
Momentum	Technical	1 day
Stohastic	Technical	1 day
MA	Technical	1 day
MACD	Technical	1 day
MA standard error	Technical	1 day
ROC	Technical	1 day
TRIX	Technical	1 day
BV	Fundamental	1 month
CF	Fundamental	1 month
Dividends paid	Fundamental	1 month
EBITDA	Fundamental	1 month
EPS	Fundamental	1 month
Number of stocks outstanding	Fund./Tech.	3–6 months
Sales	Fundamental	1 month

Table 2: List of available variables

Some of these variables were transformated e.g. to adjust different periods of regularity or to match different company scales.

Variable name	Variable type	Regulariry estimate
$\frac{P_{t-1}-P_{t-2}}{P_{t-2}}$	Fund./Tech.	1 day
$\frac{Sales_{it}}{P_{it}}$	Fundamental	1 day
$\frac{CF_{it}}{P_{it}}$	Fundamental	1 day
$\frac{EPS_{it}}{P_{it}}$	Fundamental	1 day
$\Delta_{12}\frac{EPS_{it}}{P_{it}}$	Fundamental	1 day
ROE_{it}	Technical	1 day
$Momentum_{it}$	Technical	1 day
$Stohastic_{it}$	Technical	1 day
$\frac{MA_{it}}{P_{it}}$	Technical	1 day
$MACD_{it}$	Technical	1 day
$\sigma(\frac{MA_{it}}{P_{it}})$	Technical	1 day
ROC_{it}	Technical	1 day
$TRIX_{it}$	Technical	1 day

Table 3: List of transformed variables used in simulation

where ROE_{it} is return on equity estimated using market prices so that

$$ROE_{it} = \frac{\frac{P_{it}-P_{i,t-1}}{P_{i,t-1}}Q_{it}}{P_{it}Q_{it}} = \frac{1}{P_{i,t-1}} - \frac{1}{P_{it}}$$

Other variables appeared to be insignifant during the calibration procedure determined by $\mathbb{A}$.

Since decision trees provide unprecedented level of rule intuition, it is worth examining the trees from different angle. What if this kind of algorithm is used as a supplementary means of preliminary data analysis so that an investor is interested in the most significant variables explaining furure stock prices while has no fixed specification for some reason or failed to apply classical regression analysis due to, say, inadequate residuals distribution? Then decision trees as a non-parametric tool could give an insight into the fundamental relationships.

Nowadays there are different approaches evaluating market efficiency and hence there are different groups of methods trying to reveal fundamental relationships like ones described above. Usually one distinguishes between technical and fundamental analysis – each of the methods has specific assumptions and tools and at the same time both of them are widely spread in modern financial applications. That is why it is of particular interest to analyze the most significant variables appeared in decision trees from this particular point of view.

Since root node filter is the most significant one, its variable was used as a proxy for the most important factor. Since the simulation performed was dynamic i.e. each period decision rules were reevaluated for each stock, it was possible to collect a large dataset with a distribution of root node variables. Refer to both tables above to get characteristics of different variables.

The results are quite promising.

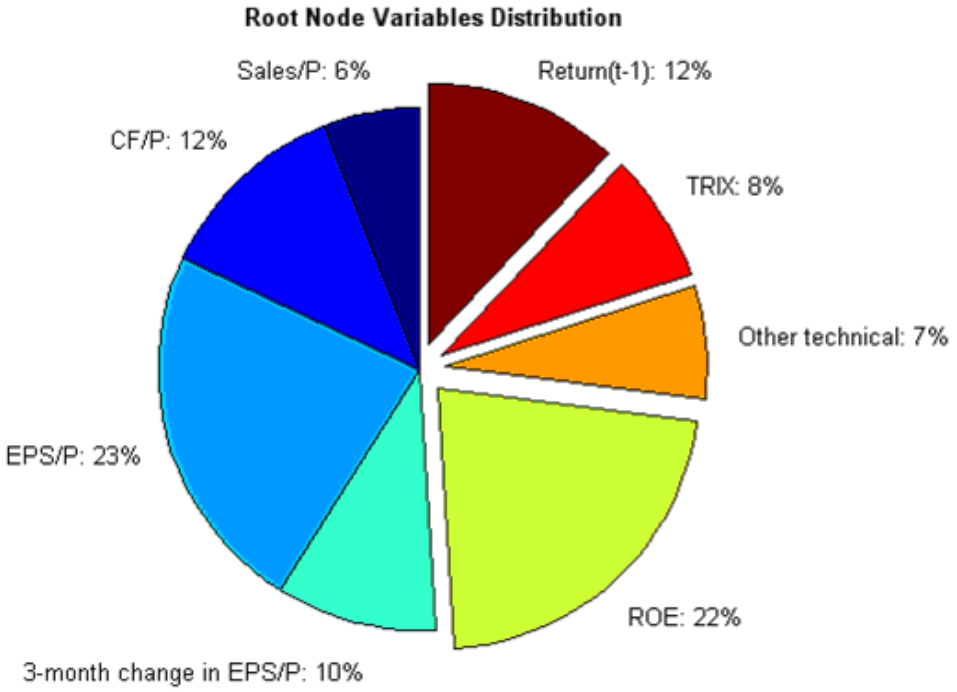

Figure 15: Root node filter variables distribution

The distribution of technical vs fundamental variables is about 50 by 50 so there is no dominance of one methodology over another. On the other hand, feasible investor strategy during simulation implied also the *combination of two types of variables*. As empirical results show, in 47% of cases the use of both types of variables simultaneously was acknowledged to be superior while in 33% only fundamental variables together with last period yield resulted in the best performance and finally in 20% of cases the use of single variable – last period yield – led to the highest results.

Although these results are valid only for a specific choice of $\mathbb{A}$ and, what is more important, for a specific dataset, the conclusion we make here is that the separate application of either technical or fundamental variable subsets does not appear to be a rational strategy.

2.5 Does the effect of information ageing influence financial performance?

In this section we will analyze another important issue – how different layers of information may affect the empirical properties of investor's strategy and what implications can be made out of performed simulations.

By definition information ageing is the effect when observations measured at different time periods have different impact on model forecasting power i.e. more recent observations play a more important role in the sample whereas the use old data could even deteriorate model predictions.

For dynamic imitation systems three major ways to create a learning sample can be specified. First of all, a learning sample can be *static* i.e. its size remains constant and *new observations do not have any influence* on it, for instance, that can be a set constituted by first 100 observations and even when new observations become available it does not change its structure. This is, of course, the most naive way to construct a learning sample but in some cases it may be adequate.

Another approach suggests to add each new observation to a learning sample so that most up-to-date information is accounted and a sample becomes *dynamically expanding* i.e. its size is constantly growing unless no new observations are available. Obviously this creation method implies that all observations have significant forecasting power and is usually applied when data dimensions are low.

When the analyzed time period is quite long, it could be a good idea to restrict possible negative influence of old observations on producing forecasts i.e. to avoid *information ageing effect*. In such a case, after a learning sample has at least a specified number of observations each new observation is being added while the oldest one is neglected so that a *dynamically stable* sample is created since its size is constant.

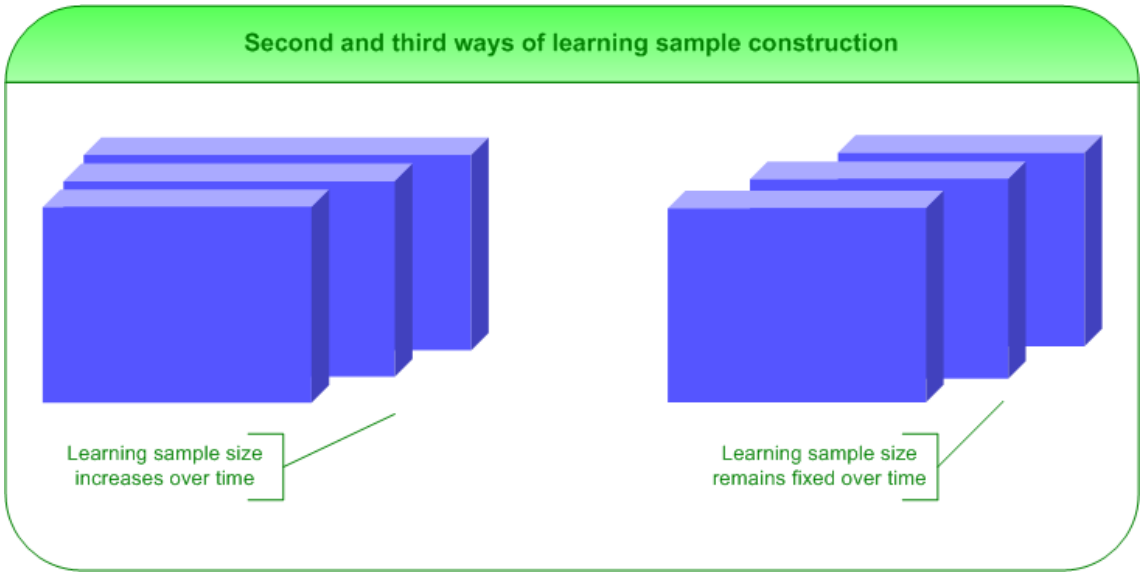

Figure 16: Learning sample and information ageing effect

It is important to point out that to the best author's knowledge the majority of financial studies uses only the first and/or the second approaches. For instance, in [10] it is stated that increased learning sample size led to better financial results since actual information was accounted.

The simulation for DAX30 stocks allowed all three types of learning sample creation, the optimal one was chosen at calibration stage. Here is the evidence of how frequently different approaches were used: in 66.6% cases *dynamically stable* learning sample was acknowledged to be best and in 33.3% – *dynamically expanding* one. Notice that static version didn't appear in the distribution at all.

Ex-post simulation analysis also showed that more than in 95% cases such kind of calibration decision was right, hence we can state that one possible reason for more than average simulated financial results is probably the proper accounting of information ageing effect.

3 Conclusion

Classification trees appear to be quite a powerful and versatile tool in modern finance applications. As empirical results show, simulated financial performance was of above average level when author-modified core of CART was applied to DAX30 data.

We can conclude that CART is not a mere effective non-parametric classification/regression tool but also a powerful means of explorative data analysis because of its exceptional interpretation capabilities. Such properties allowed us not only to demonstrate its power for pure financial applications like recursive portfolio creation but also to shed some light on the problem of technical and fundamental analysis application.

Although a lot of promising results were got during the study, there are at least two major directions towards the future research that are of particular interest. Firstly, a two-step hybrid CART-Logit model could be built so that first CART analyzes the dataset and extracts the most valuable information and then Logit model is built to account possible data peculiarities.

The second direction is to enhance the core of CART, for instance to try to construct an algorithm alowing to build more efficient trees comparing to one-level-optimal ones.

And, of course, it could be quite useful to test the built system applying it to different financial datasets.

References

[1] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, Charles J. Stone. *Classification and regression trees*. The Wadsworth Statistics/Probability Series, 1984, 358 p.

[2] Neill Brennan. *A Method for Selecting Stocks within Sectors*, electronic version, Schroder Salomon Smith Barney Equity Research: Europe, Quantitative Strategy, 20 April 2001, 24 p.

[3] *CERN Classification Standard. A single classification structure for all CERN purchases and inventory*. CERN - European Organization for Nuclear Research, Administrative Information Services,
http://ais.web.cern.ch/ais/projs/ccs/Project_Spec.html

[4] *Center-TRACON Automation System*. NASA Ames Research Center,
http://ctas.arc.nasa.gov/project_description/sw_overview.html

[5] *Current Earth Environments as Analogues for Extraterrestrial Environments*. General Meeting of the NASA Astrobiology Institute,
http://nai.arc.nasa.gov/library/downloads/annual_abstracts/BOOK-2of6.pdf

[6] *Escaping the Dangers of 'Risk Drag'*. Bernstein Investment Research and Management, http://www.bernstein.com/Public/story.aspx?cid=2602

[7] Eugene F. Fama, Kenneth R. French. *Multifactor Explanations of Asset Pricing Anomalies*. Journal of Finance, Volume 51, Issue 1 (Mar., 1996), 55-84.

[8] Eugene F. Fama, Kenneth R. French. *The Cross-Section of Expected Stock Returns*. Journal of Finance, Volume 47, Issue 2 (Jun., 1992), 427-465.

[9] Juergen Franke, Wolfgang Haerdle, Christian Hafner. *Introduction to the Statistics of Financial Markets*, electronic version.

[10] Graham Harman, Priya Paramenswaran, Martine Witt. *Shares, bonds or cash? Asset allocation in the new economy using CART*, electronic version, Salomon Smith Barney Equity Research: Australia, Quantitative Analysis, September 15, 2000, 28 p.

[11] Roger Hertog, Mark R. Gordon. *Can Active Management Beat Index Funds?*. Bernstein Investment Research and Management,
http://www.bernstein.com/Public/story.aspx?cid=1222&nid=185

[12] Inna Kolyshkina, Richard Brookes. *Data mining approaches to modelling insurance risk*, electronic version, PricewaterhouseCoopers, 22 October 2002, 20 p.

[13] Tjen-Sien Lim, Wei-Yin Loh, Yu-Shan Shih *A Comparison of the Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms*, electronic version, 2000, 27 p.

[14] John P. Mahedi. *Balance-Sheet Accruals: A Signal of Future Stock Performance in Overlooked Data*. Bernstein Investment Research and Management, http://www.bernstein.com/CmsObjectPC/pdfs/R&S_0408_BalanceSheetAccruals_JPM.pdf

[15] Lakshmi Seshadri. *JPMorgan US Quantitative Factor Model: August 2003 Stock Lists*, electronic version, JPMorgan Quantitative Equity and Derivatives Strategy, New York, August 6, 2003, 7 p.

[16] Eric H. Sorensen, Chee K. Ooi, Keith L. Miller *The Decision Tree Approach to Stock Selection*, electronic version, Salomon Smith Barney Equity Research: United States, Global Quantitative Research, November 11, 1999, 28 p.

[17] Dan Steinberg, N. Scott Cardell. *The hybrid CART-Logit Model in Classification and Data Mining*, http://www.salford-systems.com

[18] Dan Steinberg, N. Scott Cardell. *Improving data mining with new hybrid methods*. Presented at *DCI Database and Client Server World*, Boston, MA, 1998.