CentER

Discussion Paper

No. 2003–30

DERIVATION OF MONOTONE DECISION MODELS FROM NON-MONOTONE DATA

By Hennie Daniels, Marina Velikova

March 2003

ISSN 0924-7815

TILBURG UNIVERSITY

Derivation of monotone decision models from non-monotone data

Hennie Daniels[1, 2] and Marina Velikova[1]

[1]Tilburg University, CentER for Economic Research, Tilburg
[2]Erasmus University Rotterdam, ERIM Institute of Advanced Management Studies, Rotterdam.

Abstract
The objective of data mining is the extraction of knowledge from databases. In practice, one often encounters difficulties with models that are constructed purely by search, without incorporation of knowledge about the domain of application. In economic decision making such as credit loan approval or risk analysis, one often requires models that are monotone with respect to the decision variables involved. If the model is obtained by a blind search through the data, it does mostly not have this property even if the underlying database is monotone. In this paper, we present methods to enforce monotonicity of decision models. We propose measures to express the degree of monotonicity of the data and an algorithm to make data sets monotone. In addition, it is shown that monotone decision trees derived from cleaned data perform better compared to trees derived from raw data.

Keywords: data mining, domain knowledge, monotonicity, monotone data sets, decision trees

JEL-code: Statistical Decision Theory, Operations Research [C440]; Computational Techniques [C630]

1. Introduction

Data mining has attracted a lot of interest in recent years due to the growing amount of data collected in business and the need to turn this data into useful knowledge. The objective of a data mining system is to derive valuable knowledge implicitly present in large databases. Although, in data mining literature, the main emphasis is put on the analysis and interpretation phase, there are more aspects such as data selection and data pre-processing, which determine the successful implementation of any data mining system. The right description of the domain as well as data cleaning, data integration and data transformation can significantly improve the efficiency of the data mining process.

Apart from limitations regarding data quality, there can also be problems in the application of the model if knowledge discovery is conducted by blind research. Frequently the models derived are incompatible with business regulations. These incompatibilities can often be resolved by integrating expert knowledge in the data-mining process.

Another problem that may occur is the lack of interpretability of the model. In general, human decision makers require that the model is easy to understand and do not accept black box models, for example neural networks or very complex decision tree's.

Therefore, there is a need for integration of the knowledge discovered by standard data mining algorithms with the knowledge based on intuition and experience of the domain experts.

In this paper, we explicitly describe the implementation of a special form of a prior knowledge that is typical in economic decision problems, namely the monotonicity of classification rules.

In recent years, several researchers became interested in the incorporation of monotonicity constraints in different data mining methods. In ([Dan, 99]) and ([Wang, 94]) classes of monotone neural networks are introduced. In the first paper the authors implement an algorithm for training neural networks that are monotone by construction. In ([Wang, 94]) the monotonicity of the neural network is guaranteed by enforcing constraints during the training process.

Also, in the application of decision trees, several methods have been developed to solve classification problems with monotonicity constraints. In ([Ben-David, 95]), a new splitting measure for constructing a decision tree was proposed including a non-monotonicity index and standard impurity measure such as entropy. In this way, monotonicity properties of the tree and classification error can be balanced. Potharst ([Pot, 99]) provides a study for building monotone decision trees using only monotone data sets. The author presents algorithms to generate monotone trees by adding so-called corner elements to nodes. Instead of enforcing monotonicity during tree construction, Potharst and Feelders ([Pot, 02]) describe an alternative in less satisfactory models. Most of the methods mentioned above require that the data set is monotone. This constraint limits the applicability of the methods because, often, real databases are non-monotone due to the noise in the data. Frequent occurring causes are errors at data entry, inconsistencies after merging data sets, discrepancies due to the change of data over time, etc.

The rest of this paper is organised as follows.

Firstly, in section 2 we formulate the monotonicity constraints for regression and classification problems. Then, we construct measures to check if the data set is monotone by defining indicators to measure the degree of monotonicity. By comparing the value of the indicators with benchmark data sets, one can verify the monotonicity of the data under study.

Furthermore, in section 4 we develop an algorithm for cleaning the data by removing noise, such that the resulting data set is monotone. This is done by relabeling the dependent variable. It is shown on artificially generated data sets that the algorithm is capable of almost completely restoring the original data up to noise levels of 15%. Finally, we show that decision trees generated from cleaned (monotone) data outperform trees generated from the original data set, in the sense that the out-of-sample classification error is smaller and also the number of leaves is less. All this is illustrated in a case study on bond rating in section 6.

2. Monotonicity and measures for monotonicity

In many economic classification and regression problems, it is known that the dependent variable has a distribution that is monotone with respect to the independent variables. Economic theory would state that people tend to buy less of a product if its price increases (ceteris paribus), so there would be a negative relationship between price and demand. The strength of this relationship and the precise functional form are, however, not always dictated by economic theory. Another well-known example is the dependence of labour wages as a function of age and education ([Muk, 94]). In loan acceptance, the decision rule should be monotone with respect to income for example, i.e., it would not be acceptable that a high-income applicant is rejected, whereas a low-income applicant with otherwise equal characteristics is accepted. Monotonicity is also imposed in so-called hedonic price models where the price of a consumer good depends on a bundle of characteristics for which a valuation exists ([Har, 78]).

The mathematical formulation of monotonicity is straightforward. We assume that y is the dependent variable and takes values in $\boldsymbol{Y}$ and the vector of independent variables is x and takes values in $\boldsymbol{X}$. In the applications discussed here, $\boldsymbol{Y}$ is a one-dimensional space of prices or classes and $\boldsymbol{X}$ is a k-dimensional space of characteristics of products or customers. Furthermore, a data set D = (y^n, x^n) of n points in $\boldsymbol{Y}*\boldsymbol{X}$ is defined, which can be considered as a random sample of the joint distribution of (y, x). In a regression problem, the goal is to estimate the average dependence of y given x, $E(y \mid x)$. $E(y \mid x)$ depends monotonically on x, if

$$x^1 \geq x^2 \Rightarrow E(y \mid x^1) \geq E(y \mid x^2) \tag{1}$$

where $x^1 \geq x^2$ is a partial ordering on $\boldsymbol{X}$ defined by $x_i^1 \geq x_i^2$ for i = 1, 2, …, k.

In cases of a classification problem, a classification rule $\ell(x)$ assigns a class (label) to each vector x in $\boldsymbol{X}$ and monotonicity of is given by:

$$x^1 \geq x^2 \Rightarrow \ell(x^1) \geq \ell(x^2) \tag{2}$$

It can be shown in many cases that monotone models perform better than non-monotone models if monotonicity is present in the problem. This is mainly due to the fact that monotone models suppress over fitting. Some data mining algorithms can be applied to cases where the data set is partially monotone ([Dan, 99]), ([Pot, 02]) whereas other are restricted to the cases where the data set is totally monotone (see Definition 2) ([Mak, 99]), ([Pot, 02]).

There are quite a number of contributions in the literature that discuss monotonicity and measures for monotonicity of models derived from data. In ([Dan, 99]), a monotonicity index to measure the degree of monotonicity of a neural network with respect to each input variable is defined. The value of this index is between zero, indicating a non-monotone relationship, and 1, indicating a monotone relationship. To test whether a given decision tree is monotone or not, Potharst ([Pot, 99]) describes a procedure using the maximal and minimal elements of the leaf nodes of the decision trees. The degree of the non-monotonicity of the tree is computed as percentage of non-monotone leaf nodes respective to the total number of leaves. In ([Ben, 95]) another measure for the degree of non-monotonicity of a decision tree is proposed, which gives equal weight to each pair of non-monotone leaf nodes. A modification of this measure is given in ([Pot, 02]).

All these measures express the degree of monotonicity after a model has been derived from data. However, in practice, one would like to check whether or not a given data set is monotone before a decision model is constructed in order to verify the assumptions of theory. One obvious question, therefore, is how to measure the degree of monotonicity of a data set. A straightforward method is to compute the fraction of monotone pairs with respect to the total number of pairs in a data set. Another measures are the number of monotone points or the number of label changes in a data set. Apart from measures we also need benchmark data sets to compare with the indicators computed from the data. There are various ways to generate benchmark data sets. In the next section, we define three classes of benchmark data sets and derive measures to express the degree of monotonicity of these data sets.

3. Benchmark data sets

Suppose B_{π}^{N} denote an ensemble of samples of N points drawn from a probability distribution $\pi(x, \ell)$. Here π is defined in $X^*\{1,2\ldots L\}$, where X is a subset of the k-dimensional space $\Re^k$ and $\{1,2\ldots L\}$ is the set of labels. A sample D drawn from B_{π}^{N} is a data set of N points and we use the notation $D = (x^n, \ell_{x^n})_{n=1}^{N}$ throughout the paper. Given $D \in B_{\pi}^{N}$, we define three classes of benchmark data sets, which are subsets of $B_{\pi_0}^{N}$, the set of samples of N points drawn from probability distribution $\pi_0(x, \ell)$. For any of these classes of benchmark data sets, we assume that the explanatory variables and the labels are independent i.e.

$$\pi_0(x, \ell) = \pi_1(x) \,.\, \pi_2(\ell).$$

Furthermore, we assume that π_1 has a probability distribution with density ρ, defined on a subset X and π_2 is represented by $\sigma_1, \sigma_2 \ldots \sigma_L$ with $\sum_{i=1}^{L} \sigma_i = 1$.

The benchmark data sets are defined as follows. Benchmark-1 denotes the collection of all data sets generated with the same structure of independent variables like that of D and labels drawn from the distribution of the labels in D. As a second class of benchmark data sets we define Benchmark-2, which represents all data sets with the same number of explanatory variables like D, points drawn from uniform distribution between 0 and 1 and labels drawn from the distribution of the labels in D. Finally, Benchmark-3 is the ensemble of all randomly generated data sets with the same number of explanatory variables and labels like D, but now the points and labels are drawn from uniform distribution.

A data point is denoted by $x = (x_1, x_2 \ldots x_k, \ell_x)$, where $x_1, x_2 \ldots x_k$ are the values of the independent variables and ℓ_x is the label . We define:

$$\begin{aligned} \text{Down}(x) &= \{y \in X \mid y < x\}, \\ \text{Up}(x) &= \{y \in X \mid y > x\}, \\ \text{Incomp}(x) &= \text{complement of } \text{Up}(x) \cup \text{Down}(x) \end{aligned}$$

and

$$V_1 = \int_{y<x} \pi_1(y)\, dy$$

$$V_2 = \int_{y>x} \pi_1(y)\, dy$$

$$V_3 = 1 - V_1 - V_2$$

In other words, V_1, V_2 and V_3 are the probabilities that a randomly chosen point belongs to the sets Down(x), Up(x) and Incomp(x), respectively.

Firstly, we derive a measure for the expectation value of the fraction of monotone pairs in a given data set considering randomly generated data. Using this benchmark, we can compare it with the degree of monotonicity in the data set under study computed as the proportion of the number of monotone pairs from the total number of pairs. In the next step, we also derive a formula for computing the expected number of points that are monotone in a given data set.

Lemma 1

For a randomly generated data set with k- independent variables and L labels, the expectation value of the fraction of monotone pairs in the dataset, denoted by f_M, is:

$$f_M = 1 - \frac{1}{L}\left(E\{V_1\} \sum_{\ell=1}^{L} \sum_{i=\ell+1}^{L} \sigma_i + E\{V_2\} \sum_{\ell=1}^{L} \sum_{i=1}^{\ell-1} \sigma_i \right). \qquad (3)$$

Proof: Since $f_M + f_{Nm} = 1$, where f_{Nm} denotes the expectation value of the fraction of non-monotone pairs in a dataset, we will show that

$$f_{Nm} = \frac{1}{L}\left(E\{V_1\}\sum_{\ell=1}^{L}\sum_{i=\ell+1}^{L}\sigma_i + E\{V_2\}\sum_{\ell=1}^{L}\sum_{i=1}^{\ell-1}\sigma_i \right).$$

Let $D_0 \in B_{\pi_0}^N$ and $x \in D_0$. The points which are involved in non-monotone relationships with respect to x are all points $y \in Down(x)$ with label(y) > ℓ_x and all points $y \in Up(x)$ with label(y) < ℓ_x.

The expected value of the fraction of non-monotone pairs with respect to x, Nm_x, is:

$$Nm_x = V_1 \sum_{i=\ell_x+1}^{L}\sigma_i + V_2 \sum_{i=1}^{\ell_x-1}\sigma_i .$$

Since $\pi(x, \ell) = \pi_1(x) \cdot \pi_2(\ell)$, we may write $E_{x,\ell} = E_x E_\ell$ and therefore:

$$E_{x,\ell}\{Nm\} = E_x\{V_1\}E_\ell\left\{\sum_{i=\ell_x+1}^{L}\sigma_i\right\} + E_x\{V_2\}E_\ell\left\{\sum_{i=1}^{\ell_x-1}\sigma_i\right\} . \qquad (4)$$

Furthermore,

$$E\left\{\sum_{i=\ell_x+1}^{L}\sigma_i\right\} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{i=\ell+1}^{L}\sigma_i \qquad \text{and} \qquad E\left\{\sum_{i=1}^{\ell_x-1}\sigma_i\right\} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{i=1}^{\ell-1}\sigma_i .$$

Hence, the expectation value of the fraction of non-monotone pairs with respect to one point is:

$$E\{Nm\} = \frac{1}{L}\left(E\{V_1\}\sum_{\ell=1}^{L}\sum_{i=\ell+1}^{L}\sigma_i + E\{V_2\}\sum_{\ell=1}^{L}\sum_{i=1}^{\ell-1}\sigma_i \right). \qquad (5)$$

To compute the fraction of non-monotone pairs in the dataset, f_{Nm}, the result in (5) firstly is multiplied by the total number of points, N, and then by N-1 corresponding to the possible links for each point. Since, in this way, we will double the number of non-monotone pairs the expectation value of the number of non-monotone points is $\frac{N(N-1)}{2}E\{Nm\}$. Consequently, the fraction is:

$$f_{Nm} = E\{Nm\}.$$

In the simple case, where the independent variables are drawn from a **uniform distribution** on $[0,1]^k$ and the class labels have equal probability, i.e.:

$$\sigma_i = \frac{1}{L} \quad \text{for } i = 1,2\ldots L,$$

we get

$$f_{Nm} = 2^{-k}\,\frac{L-1}{L}. \qquad (6)$$

We now derive a formula for computing the expected number of points that are monotone in a arbitrary data set from the ensemble $B_{\pi_0}^N$. Let us firstly define the term "monotone point".

Definition 1.
We call z a monotone point if z participates only in monotone relationships in the sense defined in (2).

Definition 2.
A data set is monotone if all points are monotone.

Let $D_0 \in B^N_{\pi_0}$ and $x \in D_0$. Then, x is monotone if and only if:

$$\text{Max}\{\text{labels in Down}(x)\} \le \ell_x \le \text{Min}\{\text{labels in Up}(x)\}$$

We define three probabilities:

p_1 = Prob(all labels in Down(x) $\le \ell_x$)
p_2 = Prob(all labels in Up(x) $\ge \ell_x$)
p_3 = Prob(x is monotone with respect to all points in Incomp(x))

Note that $p_3 = 1$ because the relationship between incomparable points is always monotone.
Suppose card(Down(x)) = n_1, card(Up(x)) = n_2. Then p_1 and p_2 can be computed as:

$$p_1 = \left(\sum_{i=1}^{\ell} \sigma_i \right)^{n_1} \quad \text{and} \quad p_2 = \left(\sum_{i=\ell}^{L} \sigma_i \right)^{n_2},$$

because the labels are independent of one another.
Now, $q(x,\ell_x)$ is defined as the probability that a point x is monotone given its coordinates and label, ℓ_x. It can be computed by using the formula of total probability ([Tri, 82], p.34):

$$q(x,\ell_x) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1-n_1} p_1\, p_2\, p_3\, s(n_1, n_2, n_3) \tag{7}$$

where $s(n_1, n_2, n_3)$ is the probability that card(Down(x)) = n_1, card(Up(x)) = n_2 and card(Incomp(x)) = n_3. This is a multinomial problem and $s(n_1, n_2, n_3)$ can be computed using the formula for generalized Bernoulli trials ([Tri, 82], p.46):

$$s(n_1, n_2, n_3) = \frac{(N-1)!}{n_1! n_2! n_3!}\, V_1^{n_1}\, V_2^{n_2}\, V_3^{n_3}.$$

Substituting $s(n_1, n_2, n_3)$, p_1, p_2 and p_3 in (7) with their equivalents, we get:

$$q(x,\ell_x) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1-n_1} \frac{(N-1)!}{n_1! n_2! n_3!}\, V_1^{n_1} \left(\sum_{i=1}^{\ell_x} \sigma_i \right)^{n_1} V_2^{n_2} \left(\sum_{i=\ell_x}^{L} \sigma_i \right)^{n_2} V_3^{n_3}$$

$$= \left[V_1 \left(\sum_{i=1}^{\ell_x} \sigma_i \right) + V_2 \left(\sum_{i=\ell_x}^{L} \sigma_i \right) + V_3 \right]^{N-1}. \tag{9}$$

by the multinomial theorem ([Tri, 82]).

The expected number of monotone points for a data set in $B^N_{\pi_0}$ is

$$E_l\, E_x\, q(x,\ell_x) = \frac{1}{L} \sum_{\ell=1}^{L} \sigma_\ell \int_X q(x,\ell).\pi_1(x)\, dx. \tag{10}$$

As the analytical computation of this expression is difficult, it can be approximated by $\frac{1}{N} \sum_{x \in D'} q(x, \ell_x)$, where D′ is randomly generated data set. This method is used in the bond

rating case study, presented in Section 6, to compute the number of monotone points in Benchmark-2.

Both measures the expectation value of monotone pairs and the expectation value of monotone points, derived in this section, express the expectation value of the degree of monotonicity in a data set and can be used as benchmarks to compare with the indicators of the data set under study. Apart from them, one can consider as an alternative benchmark the number of label changes necessary to convert a non-monotone into monotone data set. For this purpose, we develop an algorithm for relabeling, introduced in the next section.

4. Algorithm for relabeling

The objective of the algorithm is to transform a given non-monotone data set into monotone one by changing the value of the dependent variable. This process is called *relabeling*. The idea is to reduce the number of non-monotone pairs by relabeling one data point in each step. In order to do this, a data point is chosen for which the increase in correctly labelled points is maximal (this is not necessarily the point which is involved in the maximal number of non-monotone pairs). The process is continued until the data set is monotone (see definition 2).
The correctness of the algorithm follows from lemma 2 and lemma 3. In lemma 2, it can be seen that it is always possible to reduce the number of non-monotone pairs by changing the label of only one point as long as the data set is non-monotone. In lemma 3, it is shown that there is a canonical choice for the new label for which a maximal reduction can be obtained. There may be more than one label for which this can be achieved but these are all smaller or all larger than the current label of the point.
Before describing the algorithm in details, let us first introduce some notations. The initial data set is denoted by $D = (x^n, \ell_{x^n})_{n=1}^{N}$, where x^n is a vector of independent variables and ℓ_{x^n} is a label (dependent variable) with range 1,2…L. For each data set D, Q(D) denotes the set of all non-monotone points.
For each data point $x \in Q(D)$, we define $A_i(x) \subset Down(x)$ and $B_i(x) \subset Up(x)$ by:

$A_i(x) = \{y < x \mid label\ (y) = i\}$,
$B_i(x) = \{y > x \mid label\ (y) = i\}$ for i =1,2…L.
a_i and b_i denote the number of points in $A_i(x)$ and $B_i(x)$, respectively
$N_{\ell_{x^n}}$ denotes the total number of points correctly labelled with respect to x for the current label of x, ℓ_{x^n}, i.e.

$$N_{\ell_{x^n}} = a_1 + a_2 + ... + a_{\ell_{x^n}} + b_{\ell_{x^n}} + ... + b_{L-1} + b_L .$$

We assume that all data points in the data set D are unique i.e. no points are represented twice. For each data point $x \in Q(D)$ we compute the label ℓ' for which there is a maximal increase in the number of correctly labelled points with respect to x, if the label of x is changed into ℓ'. The maximal increase is denoted by I_{max}. In case there is more than one label with one and the same maximal increase in correctly labelled points, we choose the closest label to the current label of x. In the next step, we select a point $x \in Q(D)$ for which I_{max} is the largest and change its label. This process is repeated until the data set is monotone.

Algorithm

Step 1 – Initialisation:

Compute $Q(D)$ on the basis of D

Step 2 – Main program

Step 2.1 As long as $Q(D) \neq \varnothing$

For each data point $x \in Q(D)$ compute

2.1.1 $I_{max} = \max \{ N_{\ell'} - N_{\ell_x} \mid 1 \leq \ell' < L \}$

2.1.2 Λ - set of indices ℓ' for which $N_{\ell'} - N_{\ell_x}$ is maximal

2.1.3 Form a triple (x, I_{max}, λ) where $\lambda \in \Lambda$ is the closest label to ℓ_x, (in Lemma 3 it is shown that λ is unique).

Step 2.2 From all triples, choose the one where I_{max} is maximal and change the label into ℓ'.

Step 2.3 Update $Q(D)$ on the basis of the modified data set D.

In general, the points correctly labelled with respect to x are all points incomparable to x as well as the points in $A_1 \cup A_2 \cup \ldots \cup A_{\ell_x}$ and $B_{\ell_x} \cup B_{\ell_x+1} \cup \ldots \cup B_L$. Since the number of the points incomparable to x is constant and it they do not contribute to I_{max}, we may completely ignore them.

Lemma 2

Let D^k denote the data set D after k-iterations. If $Q(D^k) \neq \varnothing$ there is at least one point $x \in Q(D^k)$ that can be relabelled such that the number of non-monotone pairs is reduced.

Proof: Since $Q(D^k)$ is a non-empty partially ordered set there is a maximal point x with label ℓ_x. Because x participates in at least one non-monotone pair there is another point $y \in Q(D^k)$ with label ℓ_y so that $x > y$ and $\ell_x < \ell_y$. We define:

s – the number of points that are smaller than x and have label $\ell < \ell_x$

s_1– the number of points that are smaller than x and have label ℓ, $\ell_x < \ell \leq \ell_y$,

t –the number of points larger than x. All these points have label $\ell \geq \ell_y$, because x is maximal.

If we relabel x with ℓ_y the increase in the number of correctly labelled points with respect to x, I^x_{max}, will be:

$$I^x_{max} = N_{\ell y} - N_{\ell x} = s + s_1 + t - s - t = s_1$$

But $s_1 \geq 1$ as there is at least one point that is smaller than x and has label ℓ_y, that is y. Therefore, by relabeling x with ℓ_y, the number of correctly labelled points is increased by at least 1.

Lemma 3

Suppose that the maximal increase I^x_{max} in correctly labelled points with respect tox can be obtained by at least two labels r and s, $r < s$. Then

$$r < s < \ell_x \text{ or } \ell_x < r < s$$

where ℓ_x is the label of x.

Proof: In order to prove this we assume that $r < \ell_x < s$ and we will show that this leads to a contradiction.

First, we choose labels p and q closest to l_x such that $r \leq p < \ell_x < q \leq s$ and the maximal increase for p and q is I^x_{max}. Then,

$$N^x_p = N^x_q > N^x_i \quad \text{for all } i \in (p+1, q-1) \tag{10}$$

For the current label of x, ℓ_x, the number of correctly labelled points is:

$$N^x_{\ell_x} = \sum_{j=1}^{\ell_x} a^x_j + \sum_{j=\ell_x}^{L} b^x_j$$

and if x is labelled with q it is:

$$N^x_q = \sum_{j=1}^{q} a^x_j + \sum_{j=q}^{L} b^x_j$$

Therefore, the maximal increase for x is

$$I^x_{max} = N^x_q - N^x_{\ell_x} = \sum_{j=\ell_x+1}^{q} a^x_j - \sum_{j=\ell_x}^{q-1} b^x_j$$

We now define the set $S = \bigcup B_i(x)$ and show that S is non-empty.

According to (10) $N^x_p > N^x_{p+1}$ i.e.

$$\sum_{j=1}^{p} a^x_j + b^x_p + \sum_{j=p+1}^{L} b^x_j > \sum_{j=1}^{p} a^x_j + a^x_{p+1} + \sum_{j=p+1}^{L} b^x_j \qquad => \qquad b^x_p > a^x_{p+1} \qquad (11)$$

From (11) it follows that $b_p \geq 1$ and therefore, B_p and respectively S are non-empty sets.
Moreover, as we consider the case where $p<\ell_x<q$ it is impossible to have p+1=q. Otherwise ℓ_x should be equal either to p or to q and then $I^x_{max} = 0$ contradicting the fact that I^x_{max} is maximal.
W now choose a maximal point $y \in S$. For its current label ℓ_y ($p \leq \ell_y \leq \ell_{x-1}$), the number of correctly labelled points with respect to y is:

$$N^y_{\ell_y} = \sum_{j=1}^{\ell_y} a^y_j + \sum_{j=\ell_y}^{\ell_x-1} b^y_j + \sum_{j=\ell_x}^{L} b^y_j$$

and if y is labelled with q, it is:

$$N^y_q = \sum_{j=1}^{q} a^y_j + \sum_{j=q}^{L} b^y_j$$

Therefore, the increase in correctly labelled points with respect toy is

$$I^y = N^y_q - N^y_{\ell_y} = \sum_{j=\ell_y+1}^{q} a^y_j - \sum_{j=\ell_y}^{\ell_x-1} b^y_j - \sum_{j=\ell_x}^{q-1} b^y_j \qquad (12)$$

We will now show that $I^y > I^x_{max}$.
Since $\ell_y \leq \ell_x$-1, the first term in (12) can be rewritten as

$$\sum_{j=\ell_y+1}^{q} a^y_j = \sum_{j=\ell_y+1}^{\ell_x} a^y_j + \sum_{j=\ell_x+1}^{q} a^y_j \qquad (13)$$

Considering both terms in (13), we have

$$\sum_{j=\ell_y+1}^{\ell_x} a^y_j \geq a^y_{\ell_x} \geq 1, \text{ because } x \in A_{\ell x}(y) \qquad (13.1)$$

and

$$\sum_{j=\ell_x+1}^{q} a^y_j \geq \sum_{j=\ell_x+1}^{q} a^x_j \text{ , because } A_{\ell x+1}(x) \cup \ldots \cup A_q(x) \subset A_{\ell x+1}(y) \cup \ldots \cup A_q(y) \text{ (since x<y)} \qquad (13.2)$$

Moreover, since $B_{\ell x}(y) \cup \ldots \cup B_{q-1}(y) \subset B_{\ell x}(x) \cup \ldots \cup B_{q-1}(x)$ for the second term in (12) we have

$$\sum_{j=\ell_x}^{q-1} b_j^y \le \sum_{j=\ell_x}^{q-1} b_j^x \tag{14}$$

Finally, y is a maximal point in S i.e. there is no data point that is larger than y and has label j, $p \le j \le \ell_{x-1}$, which implies that

$$\sum_{j=p}^{\ell_x-1} b_j^y = 0$$

and therefore,

$$\sum_{j=\ell_y}^{\ell_x-1} b_j^y = 0 \tag{15}$$

According to (13.1), (13.2), (14) and (15)

$$I^y = N_q^y - N_{\ell_y}^y = \sum_{j=\ell_x+1}^{q} a_j^x - \sum_{j=\ell_x}^{q-1} b_j^x + 1 = I_{max}^x + 1$$

Hence, $I^y > I^x_{max}$, contradicting the fact that I^x_{max} is maximal.

The correctness of the algorithm follows easily from lemma 2 and lemma 3. In each step, the number of points participating in non-monotone pairs is reduced by at least one as shown in lemma 2. Since the algorithm can only terminate when Q(D)=0, the resulting data set is monotone. By lemma 3, it follows that there is only one canonical choice for the new labels.

5. Simulation results

In order to examine the performance of the algorithm and the models based on the original and cleaned data sets, an experimental study is conducted using artificially generated data sets.

Firstly, to check to what extent the algorithm can remove noise added to a monotone data set, we conducted the following experiment. We firstly generated a data set with random points uniformly distributed between 0 and 1 and computed the label of each point by applying a monotone function on the independent variables. Subsequently, the continuous dependent variable (label) was discretized into finite number of classes. In the next step, we converted the monotone data set into a non-monotone data set by adding random noise to the discrete labels. At this point, the algorithm was applied and the percentage of correctly restored labels was computed by comparing the labels. This experiment was repeated 10 times with different numbers of points, independent variables, and labels as well as different percentages of noise. Depending on the number of the explanatory variables, several monotone functions were used to construct the initial label such as $x_1 * sin\frac{\pi}{2} x_2$ based on two variables x_1 and x_2. The final results are summarized in Table 1 below:

# points in a data set	# independent variables	# labels	Noise	Restoration (%)
100	2	3	15 %	99 %
100	2	3	15 %	98 %
100	2	4	11 %	96 %
100	3	4	15 %	94 %
100	5	3	15 %	88 %
200	2	3	15 %	97 %
200	3	4	16 %	92 %
200	3	5	16 %	92 %
200	5	4	15 %	89 %
200	7	5	15 %	88 %

Table 1: The results received after implementation of the algorithm

The results show that the algorithm restores the original data set (7 of 10 times the restoration is above 90%) to a large extent. In the rest of the cases, the restoration is to a lesser degree due to the increase of the number of independent variables and labels.
In the next step, to determine the model performance of the original non-monotone data set and the transformed monotone data set, both were applied in a tree-based algorithm presented in ([Pot, 02]) that is in many respects similar to the CART program described in ([Bre, 84]). The program only makes binary splits and uses the Gini-index as the splitting criterion. Furthermore, cost-complexity pruning is applied to generate a nested sequence of trees from which the best one is selected on the basis of test set performance. During tree construction, the algorithm records the minimum and maximum element for each node. These are used to check whether a tree is monotone.

On the basis of this algorithm, we repeated the following experiment 50 times with the first data set given in Table 1, using both the original and transformed data sets. Each data set was randomly partitioned (within classes) into a training set of 50 observations and test set of 50 observations. The training set was used to construct a sequence of trees using cost-complexity pruning. From this sequence, the best tree was selected on the basis of error rate on the test set computed as a proportion of misclassified observations (in case of a tie, the smallest tree was chosen). Finally, it was checked whether the tree was monotone and if not, the upper bound for the degree of non-monotonicity was computed by giving a pair t_1, t_2 of non-monotone leaf nodes weight $2^* p(t_1) * p(t_2)$, where $p(t_i)$ denotes the proportion of cases in leaf i.

The results show that the model yielded by the monotone data set performs better than the one yielded by the non-monotone data set, considering the average error on the trees – the average error rate for the monotone data set on both monotone and non-monotone trees is less than half that for non-monotone data set, likewise on both trees. Moreover, the average degree of non-monotonicity for monotone data set is very low in comparison with the result for the non-monotone data set. The results are summarized in Table 2:

	Monotone data set	Non-monotone data set
# monotone trees	45	41
# non-monotone trees	5	9
Average error rate on monotone trees	0.147	0.283
Average number of leaf nodes on monotone trees	5.6	3.6
Average error rate on non-monotone trees	0.156	0.293
Average number of leaf nodes on non-monotone trees	8.4	12.1
Average degree of non-monotonicity	0.003	0.062

Table 2: Comparison of the results received from monotone and non-monotone data sets

6. Case study on Bond rating

As explained in ([Dan, 99]), bond ratings are subjective opinions on the ability to service interest and debt by economic entities such as industrial and financial companies, or municipals and public utilities. Bond ratings are published by two major bond rating agencies, Moody's and Standard & Poor's, in the form of a letter code, ranging from AAA–for excellent financial strength–to D for entities in default. Bond ratings are based on extensive financial analysis by the bond rating agencies. The exact determinants of a bond rating, however, are unknown, since the interpretation of financial information relies heavily on professional judgement.

Publications of bond rating agencies offer some insight into the relevant factors that determine bond ratings. Bond rating analysis recognises the following areas of attention:

- Profitability
- Liquidity
- Asset protection
- Indenture provisions
- Quality of management

Bond rating models use independent variables, often calculated as ratios, which are predominantly derived from public financial statements. However, not all of the above-mentioned areas can be covered by financial statement figures. Aspects like quality of management, market positions and asset protection can only be captured to a limited extent.

From the Standard & Poor's Bond Guide (April 1994), 256 companies were selected. The bond ratings of these companies range from AAA to D. The ratings are not homogeneously distributed. The largest classes are A, BBB and B. Only a few companies have ratings lower than CCC. Therefore, we decided to remove all ratings below CCC. As in other studies, the + and − signs were omitted (for example, AA+, AA and AA− are all considered as AA). The bond ratings were quantified by assigning numbers from 1 for AAA to 7 for CCC. From the S&P Bond Guide, several financial figures were obtained. From Datastream additional financial figures and ratios relating to leverage, coverage, liquidity, profitability and size were downloaded. These figures have been restated to five-year averages and trend indicators, resulting in 45 explanatory variables. For each variable, the linear correlation with the quantified bond rating was calculated. For the purposes of the current case study, only five variables with the highest correlation were chosen. They are presented in Table 3.

Symbol	Definition
D/C	Debt to capital ratio
CF/D	5 years average cash flow to debt ratio
CF	5 years average cash flows (in 100 millions)
Cov	3 years average interest coverage ratio
Vol/Cov	3 years volatility of interest coverage

Table 3. Definition of the model variables

In the dataset thus constructed, there are 32640 pairs of observations of which 111 are non-monotone. To verify the monotonicity of the data set, we compare the measures for the degree of monotonicity derived from the given data with those derived from three benchmark data sets (Table 4). Benchmark-1 is constructed by using the same structure of the explanatory variables of the bond rating data set and by drawing labels from the distribution of the given labels. In Benchmark-2, the points are drawn from uniform distribution between 0 and 1 whereas the labels are again drawn from the distribution of the labels in the bond rating data set. As Benchmark-3 we use data sets with uniformly distributed points between 0 and 1 and 7 labels drawn from uniform distribution. To compute the number of monotone points in Benchmark-2 and the fraction of monotone pairs in Benchmark-3, we use the formulae derived in Section 3.

Indicators	Bond rating	Benchmark-1	Benchmark-2	Benchmark-3
Number of points	256	256	256	256
Comparable pairs	9 685	9 685	2 192	2 192
Fraction of monotone pairs	99.7 %	88.0 %	97.6 %	97.3 %
Monotone points	168	5	42	39

Table 4. Comparison of the measures for the degree of monotonicity between the bond rating data set and benchmarks

The results strongly imply the existence of monotone relationships in the bond rating data set and the algorithm for relabeling could be a useful tool for making the data set monotone.

The implementation of the algorithm on the bond rating dataset led to the labels of 28 points being changed. In order to determine the classification performance of trees generated on the basis of raw and cleaned data, we applied the tree construction algorithm described in the previous section. The results are presented in Table 5.

	Monotone dataset	Non-monotone dataset
# monotone trees	9	9
# non-monotone trees	6	6
Average error rate of monotone trees	0,474	0,510
Average number of leaf nodes of monotone trees	5,78	7,44
Average degree of non-monotonicity of non-monotone trees	0,024	0,020

Table 5: Comparison of the results received from monotone and non-monotone bond rating datasets

In the next step, we held a two-sample t-test of the null hypothesis that average error rate on monotone trees is one and the same for the monotone and non-monotone data set against one-sided alternative that the former is less than the latter. The test yielded a p-value 0.0145, which leads to rejection of the null hypothesis and respectively to the conclusion that the average error on monotone trees for the monotone data set is significantly less than that for non-monotone data set. This result, along with the result that the monotone data set yielded on average smaller monotone trees than non-monotone data set, shows that the model derived from cleaned data performs better than the model derived from the original data.

7. Conclusion

In the present paper, we have shown that the incorporation of prior knowledge can significantly improve the effectiveness of a data mining process. We explicitly consider a very common form of domain knowledge, which is present in many economic problems, that is the monotone relationship between dependent variable (label) and explanatory variables. Usually, the data sets used for solving monotone classification problems are non-monotone due to the noise in the data, which can result in unreliable output and incompatibility of the model with policy rules and business regulations. Therefore, in this paper, we introduce an algorithm for relabeling the dependent variable in a non-monotone data set and thus transform it into monotone one. Using the algorithm in a practical case study of predicting house prices, we show that the transformed (monotone) data set yields a model that has better performance and yields more reliable outcomes than that of the non-monotone data set.

References

[Ben, 95]: Ben-David, A., "Monotonicity Maintenance in Information-Theoretic Machine Learning Algorithms", *Machine Learning*, 19**,** pp. 29-43, (1995).

[Bre, 84]: Breiman, L., Friedman, J. H., Olshen, R. A. and Stone C. T., "Classification and Regression Trees", Wadsworth, (1984).

[Dan, 99]: Daniels, H. A. M. and Kamp, B., "Application of MLP networks to bond rating and house pricing", *Neural Computation and Applications*, 8, pp. 226-234, (1999).

[Fee, 00]: Feelders, A., Daniels, H. A. M. and Holsheimer, M, "Methodological and practical aspects of data mining", *Information & Management*, 37, pp. 271-281, (2000).

[Har, 78]: Harrison, O. and Rubinfeld, D., "Hedonic prices and the demand for clean air", *Journal of Environmental Economics and Management*, 53, pp. 81-102, (1978).

[Tri, 82]: Trivedi, K.S., "Probability & Statistics with Reliability, Queuing and Computer Science Applications", Prentice-Hall, (1982)

[Mak, 99]: Makino, K., Suda, T., Ono, H. and Ibaraki, T., "Data Analysis by Positive Decision Trees", *IEICE Transactions on Information and Systems*, Volume E82-D, No.1, pp.76-88, (1999).

[Muk, 94]: Mukarjee, H. and Stern, S., "Feasible Nonparametric Estimation of Multiargument Monotone Functions", *Journal of the American Statistical Association*, 89, No.425, pp. 77-80, (1994).

[Nun 91]: Nunez, M., "The Use of Background Knowledge in Decision Tree Induction", *Machine Learning*, 6, pp. 231-250, (1991).

[Pot, 99]: Potharst, R., "Classification using decision trees and neural nets", Erasmus Universiteit Rotterdam, *SIKS Dissertation Series* No. 99-2, (1999).

[Pot, 02]: Potharst, R. and Feelders, A.,"Classification trees for problems with monotonicity constraints", *SIGKDD Explorations Newsletter*, Volume 4, Issue 1, (2002).

[Wan, 94]: Wang, S., "A neural network method of density estimation for univariate unimodal data", *Neural Computation & Applications*, 2, pp. 160- 167, (1994).