Fundamenta Informaticae 129 (2014) 1–32
DOI 10.3233/FI-2013-0000
IOS Press

Multi-Relational Model Tree Induction Tightly-Coupled with a Relational Database

Annalisa Appice*, **Michelangelo Ceci, Donato Malerba**
Dipartimento di Informatica
Università degli Studi Aldo Moro di Bari, Italy
{*annalisa.appice, michelangelo.ceci, donato.malerba*}*@uniba.it*

Abstract. Multi-Relational Data Mining (MRDM) refers to the process of discovering implicit, previously unknown and potentially useful information from data scattered in multiple tables of a relational database. Following the mainstream of MRDM research, we tackle the regression where the goal is to examine samples of past experience with known continuous answers (response) and generalize future cases through an inductive process. Mr-SMOTI, the solution we propose, resorts to the structural approach in order to recursively partition data stored into a tightly-coupled database and build a multi-relational model tree which captures the linear dependence between the response variable and one or more explanatory variables. The model tree is top-down induced by choosing, at each step, either to partition the training space or to introduce a regression variable in the linear models with the leaves. The tight-coupling with the database makes the knowledge on data structures (foreign keys) available free of charge to guide the search in the multi-relational pattern space. Experiments on artificial and real databases demonstrate that in general Mr-SMOTI outperforms both SMOTI and M5' which are two propositional model tree induction systems, and TILDE-RT which is a state-of-art structural model tree induction system.

Keywords: Data mining, Mining methods and algorithms, Regression, Relational Model Trees, Relational DBMS coupling, Lookahead in Model Tree Induction

1. Introduction

In the recent years, we are assisting to a growing attention for data mining applications where data are inherently relational, that is, they are scattered through multiple tables from a relational database.

*Address for correspondence: Dipartimento di Informatica, Università degli Studi Aldo Moro di Bari, Italy

Some examples of applications are found in mining the web, which can be seen as a massive graph of linked pages [19], in social network analysis, which focuses on modeling the relationships between social (individuals, groups, organizations) and computational (computers and web sites) actors [38], in computational biology, which studies interactions between components (genes and proteins) of the living cells [33], and in mining geo-referenced objects, which are characterized by implicit spatial relationships defined by geometry and positioning [28].

Multi-relational data mining (MRDM) is a multi-disciplinary field which provides sound methods for the discovery of *relational* patterns that involve multiple relations and are stated in a more expressive language (e.g., predicate calculus and SQL) than patterns defined on a single data table.

Many data mining tasks (e.g., classification, clustering, association analysis) have been adapted to a multi-relational setting. In particular, several studies have been reported for the regression task, which is the main focus of this paper. In the regression task, some labeled observations (training cases) are given, which are instances of an unknown continuous function f, and the problem is to build a general model that approximates the function f and can be subsequently used to make predictions on new unlabeled observations.

Handling multi-relational data adds difficulties to the regression task. First of all, data available in distinct database relations describe different objects (or entities) involved in the phenomenon under study. These objects play different roles, and it is necessary to distinguish between *reference* (or target) *objects*, which are the main subject of analysis, and *task-relevant objects*, which are related to the former and contribute to account for the variation. The response (dependent) variable of a regression model is a property of the reference objects, while the explanatory (independent) variables of the model are associated to both reference and task-relevant objects.

Second, for each reference object there might be many associated task-relevant objects (non-determinacy) [22]. This "many-to-one" relationship is logically modeled by means of foreign key database constraints and often introduce additional parts of a structure like adjacent nodes in a graph. Generally, the introduction of non-determinate task-relevant objects in a multi-relational regression model does not immediately reduce the error, thus causing a serious problem for greedy hill-climbing search [6]. On the other hand, complete search is impractical due to the extremely large search spaces.

Third, explanatory variables of task-relevant objects can be linked in several alternative ways to the response variable. Therefore, it is necessary to establish how the value of the response variable should be estimated due to the multiple predictions that are possible for one reference object.

Fourth, efficiency and scalability are two critical issues for multi-relational methods [5]. Efficiency concerns the search space, which is often impressively large, the search strategy, which should be guided by heuristics, orderings and explicit user preferences, and the evaluation of candidate models, which might involve one or more joins between relations. Scalability refers to the ability to maintain performance as the size of the database being mined increases. Processing data in main memory is only possible when enough memory is available. Moreover, in data-intensive processes it is important to exploit powerful mechanisms for accessing, filtering and indexing data, such as those available in database management systems (DBMS).

Approaches to multi-relational mining of regression models can be classified into *propositional* and *structural*. The propositional approach requires the transformation of the original multi-relational data into a single-relational (propositional or attribute-value) representation by constructing features which capture relational properties of data. This kind of transformation, named *propositionalization*, decouples feature construction from model construction [24]. In this way, the resulting representation can be the

input to a wide range of robust and well-known conventional regression methods which operate on a single table. This approach is adopted by Džeroski *et al.* [12], who apply the model tree induction system RETIS [16] to a propositional representation of multi-relational data in order to construct a regression model.

The propositional approach presents some drawbacks. The full equivalence between the original and the transformed (propositionalized) training sets is possible only in special cases. However, even when possible, the output table size is unacceptable in practice [34] and some form of feature selection is required. Therefore, the transformed problem is different from the original one for pragmatic reasons. Moreover, propositionalization anticipates learning, so that it is not possible to detect the need for a representation change during the learning process in order to generate accurate models. To mitigate this problem, a method that creates structural features in a class-sensitive manner has been proposed [23]. Finally, translating back the outputs of the propositional learner into a relational form is possible only in particular cases [26] (Section 5.7, pp. 114-122), when the non-determinacy issue does not occur and propositionalization is not based on aggregation.

The structural approach takes into account the original data structure, so that the whole hypothesis space is directly explored by the learning method. FORS [17] is the first multi-relational regression method based on the structural approach. It is based on a separate-and-conquer search strategy to induce a set of first order logic clauses from training cases and user-defined background knowledge. The regression function associated to each clause is built on the best subset of variables among all possible combinations of at most ξ continuous explanatory variables in the clause (ξ is a user defined parameter). The expensive search performed by FORS hampers the scalability. FORS allows regression functions to include explanatory variables of non-determinate task-relevant objects, however it naively solves the problem of the multiple predictions by just choosing one of the options [17].

Alternative solutions implement the computationally more efficient divide-and-conquer search strategy. In particular, SRT [21], its successor S-CART [22] and TILDE-RT [4] recursively build binary trees by selecting at each step a conjunction of literals which instantiate some user-defined schemata. These schemata constrain variable bindings and types and can be used to overcome limitations of the myopic search especially when task-relevant objects are non-determinate. Their definition is not straightforward and requires a good domain knowledge to find the best trade-off between specificity and generality. The regression model at the leaves of the tree induced by SRT and TILDE-RT is a constant function (the mean). Hence both systems induce a *regression tree*. Optionally, S-CART can build a more complex *model tree* with linear regression functions at the leaves. These functions, however, may include only the explanatory variables of the reference objects [25]. The separation of the partitioning stage from the selection of a regression model at the leaves makes the evaluation of the best conjunction at a node incoherent with respect to the model tree returned by S-CART. As pointed out in [29], this incoherence may hamper the construction of the correct model tree.

ReMauve [41] is a multi-relational model tree learner which deals with non-determinate task-relevant objects by means of user-defined aggregate functions (e.g., avg, min, max, count) to compute on non-determinate variables. However, besides the information loss, aggregates suffer from problems of statistical significance when computed on very few values or on skewed values [8]. For efficiency reason, the linear regression function used for predictions at each leaf includes only variables (aggregates or not) involved in the tests found along the path from the root to the leaf. On the other hand, the evaluation of the binary test on an explanatory variable at an internal node is simply based on an estimation of the error made when two straight-line regressions built on such explanatory variable are associated with the

two children. This is computationally efficient, but it introduces incoherence between partitioning and prediction stage during the tree construction.

Finally, TRENDI [10] is a hybrid method which induces a relational regression tree *à la* TILDE-RT and makes predictions on the basis of the k closest training examples at a leaf. Distance functions defined for relational representations are used to select the neighboring examples. Once again, the partitioning and prediction stages are independent. Moreover, kernel regressors do not capture the structure of the domain as linear models do. Relational regression tree induction is recently investigated in data stream setting where the adopted approach is that of computing on-line summaries on the recently flowed linked data, use these summaries for the tree induction and assign constant function to the leaves [15].

To overcome limitations of existing methods with respect to the four issues reported above, a different algorithm is proposed in this paper. Following the methodology reported in [40] for upgrading propositional learners towards multi-relational representations, we identify the propositional model tree learner SMOTI [29] as the best matching for the learning task. SMOTI integrates the partitioning and the prediction stages. Specifically, trees induced by SMOTI include two types of internal nodes: regression nodes and splitting nodes. The former are associated with straight-line regressions, while the latter are associated with splitting tests. Continuous variables in the regression nodes contribute to the stepwise construction of the multiple linear models associated with the leaves. The selection measure for tests is coherent with respect to the linear regression models associated with the leaves. A peculiarity of this tree structure is its capability of distinguishing variables with a *global* effect from those with a *local* effect in the regression models. Variables of the regression nodes selected at higher levels in the tree have a global effect, since they affect several regression models associated with the leaves. Experimental results [29] showed that model trees induced by SMOTI are generally simple and can be easily interpreted.

Mr-SMOTI (*M*ulti-*R*elational *S*tepwise *Mo*del *T*ree *I*nduction), the relational upgrade of SMOTI proposed in this paper, provides a solution to these problems: a) mining relational regression models without propositionalizing the training set; b) dealing with non-determinate task-relevant objects without aggregate functions; c) allowing all (determinate and non-determinate) continuous variables to appear in regression models at the leaves; d) choosing tests at internal nodes by an evaluation function which is coherent with the models at the leaves; e) distinguishing variables with global and local effect; f) supporting scalability though a tight integration with the DBMS; g) taming complexity through materialization of intermediate results.

The paper is based on our past works [1], [2] which are now extended in the following directions:

1. Motivations and contributions of this work with respect to the state of art of research in relational data mining and regression analysis are clearly stated in the Introduction. The relational regression problem is formally defined. The basic terminology is opportunely introduced. Relational model trees with splitting and regression nodes are formalized.

2. The related works (including more recent advances in relational regression) are systematically analyzed and discussed. Limits of existing solutions are discussed. Proposed solutions are reported when presenting Mr-SMOTI.

3. The technical presentation of Mr-SMOTI is completely revised. The algorithm is extended with respect to the ancestor system presented in [1] [2]. In particular, splitting test evaluation is now provided with a kind of look-ahead; model tree induction is significantly sped-up by materializing intermediate results.

4. The empirical evaluation of Mr-SMOTI is completely novel. We consider both artificially generated relational databases and several real databases, while in [1] [2] we performed only experiments with two real databases. We evaluate both accuracy and complexity of induced model trees as well as learning times. We compare Mr-SMOTI with several propositional and relational competitors. Wilcoxon statistical tests are now used for the pairwise comparison between each pair of systems.

The paper is organized as follows. In the next Section, basic concepts of stepwise model tree induction are explained, while in Section 3, the multi-relation regression problem is formulated. Mr-SMOTI is presented in Section 4 and the database integration is discussed in Section 5. In Section 6, we report the complexity analysis of Mr-SMOTI. Lastly, experimental results are reported in Section 7 and some conclusions are drawn.

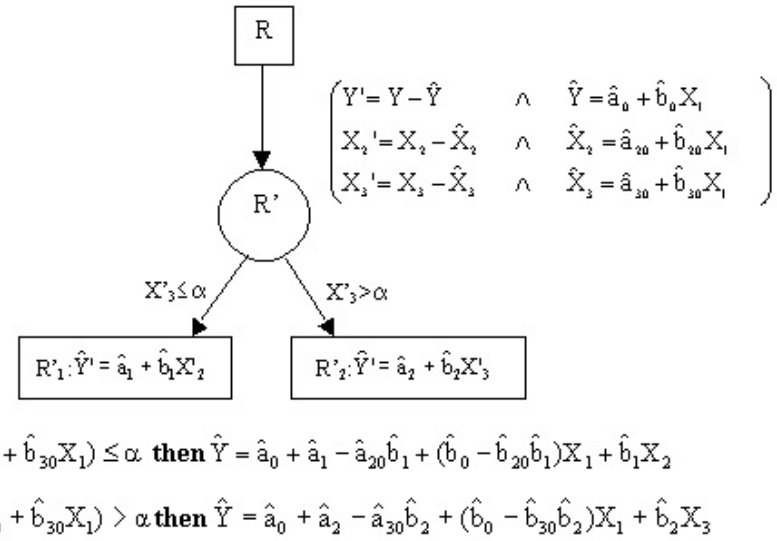
2. Stepwise Model Tree Induction

This section is devoted to presenting the stepwise construction of a model tree with regression and splitting nodes. In this Section, the training set is assumed to be represented as a single table in order to simplify the introduction of some basic concepts. Issues related to the multi-relational representation of data and regression patterns as well as the presentation of our algorithm that properly addresses these issues are postponed to the next two Sections.

Let $\boldsymbol{X}$ be the feature space spanned by m explanatory variables X_1, $X_2 \ldots X_m$ and let Y denote the (continuous) response variable. The training set S is a set of couples $(\boldsymbol{x}, y) \in \boldsymbol{X} \times Y$ such that $y = f(\boldsymbol{x}) + \epsilon$, f is an unknown continuous function and ϵ is the error. Model tree learners approximate f by recursively partitioning the feature space $\boldsymbol{X}$ into a finite number of subspaces, and by associating a linear function with each subspace.

A common characteristic of almost all Top Down Induction Model Tree (TDIMT) systems is that the (multiple) regression model associated with a leaf is built on the basis of those training cases falling in the corresponding partition of the feature space. Models in the leaves have only a local validity and do not consider the global effects that some variables might have in the underlying model function. In model trees, global effects can be represented by variables that are introduced in the multiple models at higher levels of the tree. This corresponds to a tree structure with splitting nodes to partition training data and regression nodes to perform straight-line regressions.

SMOTI is a TDIMT system which constructs multiple linear models by intermixing partitioning steps with regression steps. The incremental construction of a multiple linear regression model (see Figure 1) is made by removing the linear effect of the introduced variables each time a new explanatory variable is added to the model ($\hat{Y} = \hat{a} + \hat{b}X$) and passing down both the residuals of Y and the residuals of explanatory variables to be selected for the next regression step. SMOTI exhibits the following characteristics: (1) model trees have splitting and regression nodes; leaves are always regression nodes, (2) a (multiple) linear model is associated with each leaf. It involves all continuous variables in the regression nodes along the path from the root the leaf, (3) variables involved in regression nodes at top levels of the tree capture global effects, while those involved in regression nodes close to the leaves capture local effects, (4) the heuristic evaluation function is coherent with the linear model at the leaves, (5) a subset of continuous variables may be involved in multiple linear models associated with the leaves, thus



$$\textbf{if } X_3 - (\hat{a}_{30} + \hat{b}_{30}X_1) \leq \alpha \textbf{ then } \hat{Y} = \hat{a}_0 + \hat{a}_1 - \hat{a}_{20}\hat{b}_1 + (\hat{b}_0 - \hat{b}_{20}\hat{b}_1)X_1 + \hat{b}_1 X_2$$

$$\textbf{if } X_3 - (\hat{a}_{30} + \hat{b}_{30}X_1) > \alpha \textbf{ then } \hat{Y} = \hat{a}_0 + \hat{a}_2 - \hat{a}_{30}\hat{b}_2 + (\hat{b}_0 - \hat{b}_{30}\hat{b}_2)X_1 + \hat{b}_2 X_3$$

Figure 1. A model tree top-down induced by intermixing partitioning steps with regression steps. The feature space R is described by three continuous explanatory variables X_1, X_2 and X_3. The contribution of X_1 is "reliably" estimated on the whole R and the initial regression model is approximated by the straight-line regression $Y = \widehat{a}_0 + \widehat{b}_0 X_1$. Features are replaced by residuals (R'). The effects of the variables X_2 and X_3 are added locally to the subregions R'_1 ($X'_3 \leq \alpha$) and R'_2 ($X'_3 > \alpha$), respectively.

solving problems due to collinearity[9] that lead to poorly estimate regression coefficients in presence of explanatory variables linearly related to each other.

Despite its ideal characteristics, SMOTI does not deal with data spread out in multiple tables of a relational database. Moreover, training data from a single table is stored in the main-memory, which results in high performance for computationally intensive processes, but only when *enough memory* is available to store all necessary data.

3. Problem definition

The multi-relational regression problem is formalized as follows.
Given

- a training database D which is composed by $h+1$ relational tables, $D = \{T_1, T_2, \ldots, T_{h+1}\}$;
- a set PK of primary key constrains on tables in D;
- a set FK of foreign key constrains on tables in D;
- a set of reference objects S which are stored in the target table $T \in D$;
- h sets of task-relevant objects $R_1, R_2, \ldots, R_h$, where each set R_i is stored in the non-target table T_i with $T_i \in D - \{T\}$;
- the response attribute Y that is a continuous attribute of T, different from the primary key, with a domain in $\mathbb{R}$.

Find a regression model to predict the response attribute Y by taking into account the arrangement of the task-relevant objects related to according to FK.

The learner receives full information from D with the response attribute values and uses FK to navigate the schema of D starting from the target table T in order to retrieve the set of task-relevant objects which are related to the reference ones. In the followings, the relation between reference objects and task-relevant objects is formally defined on the basis of FK.

Definition 3.1. A task-relevant object $tro \in R_i$ is 1-*related* to a reference object $ro \in S$ if and only if there exists a foreign key constraint $fk \in FK$ from R_i to S (or vice-versa) such that the foreign key of tro assumes the same value of the primary key of ro (or vice-versa).

Definition 3.2. A task-relevant object $tro \in R_i$ is k-*related* to a reference object $ro \in S$ if and only if there exists a task-relevant object $newTro \in R_j$ such that $newTro$ is $(k-1)$-*related* to ro and if there exists a foreign key constraint fk from R_j to R_i (or vice-versa) such that the foreign key of $newTro$ assumes the same value of the primary key of tro (or vice-versa).

Definition 3.3. A task-relevant object $tro \in R_i$ is *related* to a reference object $ro \in S$ if and only if there exists some $k \geq 1$ such that tro is k-related to ro.

The last definition permits to formally define the notion of unit of analysis as it is intended in multi-relational data mining.

Definition 3.4. The unit of analysis $D[ro]$ of a reference object ro is defined as follows:

$$D[ro] = D[ro|R(ro)] \cup \bigcup_{tro_i \in R(ro)} D[tro_i|R(ro)], \tag{1}$$

where

- $R(ro)$ is the set of task-relevant objects related to ro;
- $D[ro|R(ro)]$ contains properties of ro and relations between ro and some $tro_i \in R(ro)$; and
- $D[tro_i|R(ro)]$ contains properties of tro_i and relations between tro_i and some $tro_j \in R(ro)$.

Properties and relations can be represented in the logical formalism as *property predicates* and *structural predicates*, respectively. The former are binary predicates which define the value taken by an attribute of an object. The latter are binary predicates which relate task-relevant objects as well as reference objects with task-relevant objects.

Example 3.5. Let Mutagenesis [39] be the database which describes the structure of 188 molecules in terms of atoms and bonds. Molecule table is the target table, while Atom and Bond tables play the role of non-target tables. The constant $m1$ denotes a molecule, while the constants $a1$, $a2$ and $a3$ identify three atoms and the constants $b1$ and $b2$ identify two bonds (see Figure 2 for a database representation of interactions among $m1$, $a1$, $a2$, $a3$, $b1$ and $b2$).

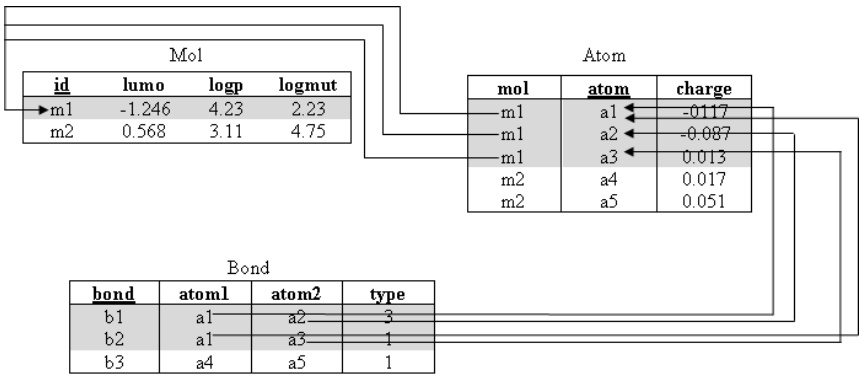


Figure 2. Database representation of some examples in Mutagenesis. In grey background examples considered in Example 3.5.

$D[m1] = mol\_lumo(m1, -1.246), mol\_logp(m1, 4.23), \mathbf{mol\_logmut(m1, 2.23)},$
$mol\_atom(m1, a1), mol\_atom(m1, a2), mol\_atom(m1, a3),$
$atom\_charge(a1, -0.117), atom\_charge(a2, -0.087),$
$atom\_charge(a3, 0.013), atom\_bond1(a1, b1), atom\_bond2(a2, b1),$
$atom\_bond1(a1, b2), atom\_bond2(a3, b2), bond\_type(b1, 3), bond\_type(b2, 1).$

where $mol\_lumo(\cdot,\cdot)$, $mol\_logp(\cdot,\cdot)$ and $atom\_charge(\cdot,\cdot)$ are property predicates; $mol\_atom(\cdot,\cdot)$, $atom\_bond1(\cdot,\cdot)$ and $atom\_bond2(\cdot,\cdot)$ are structural predicates; mol_logmut is the property predicate which denotes the response.

In this work, units of analysis are mined in order to learn regression models in form of *relational* model trees. As in model trees induced in classical data mining, the basic idea of relational model trees is that of recursively partitioning units of analysis and associating (local) multiple linear functions to the leaves. The difference resides in the relational nature of training data which leads to model trees in form of relational patterns. In the followings, we formally define the semantics of relational model trees. Peculiarity of relational model trees we mine in this paper is that, as in SMOTI, they have two kinds of internal nodes: splitting nodes and regression nodes.

Definition 3.6. A relational model tree is a binary tree $\tau = (N, B)$ where:

1. N is the set of nodes and B is the set of edges.
2. Each internal node $t \in N$ is associated with a set of reference objects $S(t)$ that is a subset of S. The root is associated with the entire set S.
3. Each leaf node is associated with a multiple linear regression function to predict Y: regression variables are attributes of the units of analysis.
4. Each edge $\langle t_i, t_j \rangle \in B$ can be either :
   - a regression edge, that adds a variable to the model or
   - a splitting edge, that is a Boolean test which permits to identify the set $S(t_j) \subseteq S(t_i)$.

A regression edge $\langle t_i, t_j \rangle \in B$ outcomes from an internal node t_i of the tree. This kind of internal node, called *regression node*, has a unique child t_j, that is, $\nexists t_k \in N, t_k \neq t_j$ for which $\langle t_i, t_k \rangle \in B$.

A splitting edge $\langle t_i, t_j \rangle \in B$ also outcomes from an internal node t_i of the tree. But, this kind of internal node, called *splitting node*, has two children. The splitting node t_i partitions the set $S(t_i)$ according to the Boolean tests on the outcoming edges $\langle t_i, t_j \rangle$ and $\langle t_i, t_k \rangle$. Both outcoming edges are splitting edges which satisfy two conditions, that is,

1. $S(t_j) \bigcap S(t_k) = \emptyset$ (mutual exclusivity), and
2. $S(t_j) \bigcup S(t_k) = S(t_i)$.

In the case of propositional learners, Boolean tests are attribute-value conditions and mutual exclusivity is straightforwardly guaranteed by means of opposing conditions (e.g., $X_i \leq \alpha$ and $X_i > \alpha$ for continuous variables). In multi-relational data mining, Boolean tests are expressed as First Order Logic conjunctive formulae with existentially quantified variables and opposing conditions cannot be simply obtained by negation. Let $Q(t_i)$ be the conjunction of Boolean tests labeling the edges from the root to the node t_i in N. $Q(t_i)$ is an intensional description of $S(t_i)$, that is,

$S(t_i) = \{ro \in S \mid D[ro] \models Q(t_i) \quad i.e., \quad D[ro]$ logically entails $Q(t_i)\}$.

The edge connecting the split node t_i to its left child t_j is labeled as f, where f is first order logic conjunctive formula with existentially quantified variables such that $Q(t_i) \wedge f$ satisfies linkedness property [14]). As reported in [4], to guarantee mutually exclusiveness the edge connecting the split node t_i to its right child t_k must be labeled as $Q(t_i) \wedge \neg(Q(t_i) \wedge f)$.

Example 3.7. Let us consider two units of analysis:

$D[m1] = mol(m1), mol\_logmut(m1, 2.23), mol\_atom(m1, a1),$
$atom\_bond1(a1, b1), bond\_type(b1, 2),$
$atom\_bond2(a1, b2), bond\_type(b2, 3).$

$D[m2] = mol(m2), mol\_logmut(m2, 4.75), mol\_atom(m2, a2),$
$atom\_bond1(a2, b3), bond\_type(b3, 3).$

The tree in Figure 3 correctly places $D[m1]$ in t_4 and $D[m2]$ in t_5 based on:

$Q(t_4) = \exists U\ mol(U)\ \wedge \exists V\ mol\_atom(U, V)\ \wedge \exists Z\ atom\_bond1(V, Z)$
$\wedge bond\_type(Z, 2)$

$Q(t_5) = \exists U\ mol(U)\ \wedge \exists V\ mol\_atom(U, V)\ \wedge \exists Z\ atom\_bond1(V, Z)$
$\wedge \neg(mol(U)\ \wedge \exists V'\ mol\_atom(U, V')$
$\wedge \exists Z'\ atom\_bond1(V', Z') \wedge bond\_type(Z', 2))$

4. Multi-Relational Stepwise Model Tree Induction

Mr-SMOTI is the multi-relational upgrade of the propositional learner SMOTI. It is designed to *approximately* achieve the propositional system as a special case. The upgrade preserves all characteristics of SMOTI but, at the same time, deals with data and patterns in a multi-relational formalism. The construction of the tree starts with a root node t_0 which is associated with the entire set S of reference objects and recursively proceeds by choosing from three possibilities, that is, (1) growing the tree by adding a splitting node t, (2) growing the tree by adding a regression node t, and (3) stopping the tree's growth at

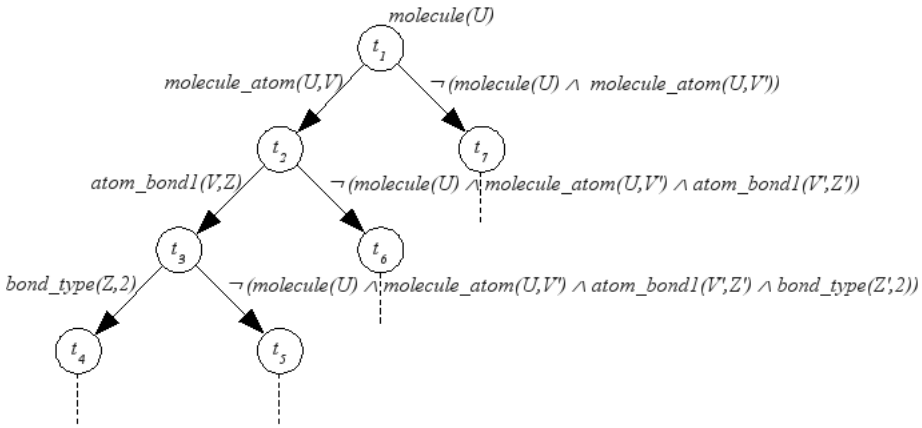


Figure 3. Opposing split conditions in a relational model tree.

the node t. The validity of either a splitting test or a regression step is based on two evaluation measures, $\sigma(t)$ and $\rho(t)$, respectively, which are both Root Mean Squared errors, and then the tests can be actually compared to choose between a splitting node or a regression node at each step.

4.1. Generating the Set of Candidate Tests for a Splitting Node

One of the main aspects in which Mr-SMOTI differs from its propositional counterpart is in the computation of the candidate opposing Boolean tests for a splitting node t. Let $Q(t)$ be the conjunction of Boolean tests labeling the edges from the root to t and $S(t)$ the set of reference objects falling in t. The candidate tests at the node t are searched in the lattice of refinements of $Q(t)$ spanned by a generality order between first order logic conjunctive formulae. Given two first order logic conjunctive formulae Q_1 and Q_2, $Q_1 \succeq Q_2$ denotes that Q_1 is more general than Q_2, or equally, that Q_2 is more specific than Q_1. By adopting the θ-subsumption onto first order logic conjunction formulae [35], a generality order over the candidate space is defined. Formally:

Definition 4.1. Q_1 is more general than Q_2 under θ-subsumption, denoted as $Q_1 \succeq_\theta Q_2$, if and only if Q_1 θ-subsumes Q_2, that is, a substitution θ exists such that $set(Q_1)\theta \subseteq set(Q_2)$ where $set(Q)$ denotes the set of literals of a first order logic conjunctive formula Q.

Example 4.2. The first order logic conjunctive formula $Q : \exists U\ mol(U)$ θ-subsumes the first order logic conjunctive formulae:
$Q_1 : \exists U\ mol(U) \wedge \exists V\ mol\_atom(U, V)$
$Q_2 : \exists U\ mol(U) \wedge \exists V\ (mol\_lumo(U, V) \wedge V \leq 4.7)$
$Q_3 : \exists U\ mol(U) \wedge \exists V\ mol\_atom(U, V) \wedge \exists Z\ (atom\_charge(V, Z) \wedge Z \leq 0.01)$
with θ an empty substitution.

θ-subsumption is a quasi-ordering, since it satisfies the reflexivity and transitivity property but not the anti-symmetric property. Hence, the candidate space is a quasi-ordered set, which can be operationally explored by means of the downward refinement operator γ which computes a set of refinements of Q with respect to $\succeq_\theta$.

Definition 4.3. Let $\mathbb{F}$ be a first order logic language and $\succeq_\theta$ be the θ-subsumption relation defined in Definition 4.1, a downward refinement operator for ($\mathbb{F}$, $\succeq_\theta$) is a function γ such that $\gamma(Q) \subseteq \{Q' | Q \succeq_\theta Q'\}$, for every $Q \in \mathbb{F}$.

In Mr-SMOTI, the downward refinement operator γ always consists in adding a conjunctive formula f (candidate test) to the first order logic formula $Q(t)$ and preserving the property of linkedness in the resulting first order logic conjunctive formula, that is, f shares one variable with one of the literal of $Q(t)$. f and $\neg(Q(t) \wedge f)$ are candidate opposing Boolean tests to be evaluated (labels on the left and right edges outcoming a splitting node).

The search in Mr-SMOTI takes into account a lookahead, that allows the algorithm to anticipate decisions to be taken at lower levels of the tree. Consequently, the conjunctive formula f is in one of the following forms:

1. $\exists Z\ table\_property(V, Z) \wedge cond(Z)$ where $table\_property(\cdot,\cdot)$ is a property predicate and V is a variable of $Q(t)$ (linkedness);
2. $\bigwedge_{k=1}^{l} \exists Z_k\ table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k})$, where Z_k corresponds to Z_{i_k} (or to Z_{j_k}) while Z_{j_k} (or Z_{i_k}) is a variable of $Q(t) \wedge \bigwedge_{h=1}^{k-1} \exists Z_h\ table_{i_h}\_table_{j_h}(Z_{i_h}, Z_{j_h})$. This way linkedness of $Q(t) \wedge f$ is guaranteed;
3. $\bigwedge_{k=1}^{l} \exists Z_k\ table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k}) \wedge (\exists Z\ table\_property(Z_l, Z) \wedge cond(Z))$, where Z_k corresponds to Z_{i_k} (or to Z_{j_k}) while Z_{j_k} (or Z_{i_k}) is a variable of $Q(t) \wedge \bigwedge_{h=1}^{k-1} \exists Z_h\ table_{i_h}\_table_{j_h}(Z_{i_h}, Z_{j_h})$.

In this formulation, $cond(Z)$ is in the form $Z \leq v$ or $Z \in \{v_1, v_2, \ldots, v_p\}$ depending on the fact that Z refers to either a continuous or a discrete attribute. l ranges between 1 and $MaxLookahead$ (user-defined parameter).

Example 4.4. Let us consider Mutagenesis and the first order logic formula:
$Q : \exists U\ mol(U)$.

As refinements of Q, with $MaxLookahead = 2$, the following formulae can be evaluated among others:

$Q_1 : \exists U\ mol(U) \wedge \exists V\ mol\_atom(U, V) \wedge \exists Z\ (atom\_charge(V, Z) \wedge Z \leq 0.01)$
$Q_2 : \exists U\ mol(U) \wedge \exists V\ mol\_atom(U, V) \wedge \exists Z\ atom\_bond(V, Z)$
$Q_3 : \exists U\ mol(U) \wedge \exists V\ mol\_atom(U, V) \wedge \exists Z\ atom\_bond(V, Z) \wedge$
$\wedge \exists W\ (bond\_type(Z, W) \wedge W \in \{1, 2\}$

This lookahead is computationally expensive, but it may lead to significant improvements in the accuracy of the resulting tree. In the algorithm originally proposed in [1], no lookahead is considered and the candidate space is explored at only one level: f is either a single structural predicate or a property predicate with the corresponding condition ($cond(\cdot)$).

In Mr-SMOTI, the condition $cond(Z)$ on a continuous property attribute Z, that is in the form $Z \leq v$, chooses v among the cut points found by an equal-frequency discretization of the values of Z. This discretization is performed on-line over the set, denoted as $\mathbb{D}_Z(t)$, of distinct values of Z which are observable in the units of analysis of the reference objects falling in $S(t)$. The number of retrieved cut points is set to the square root of the cardinality of $\mathbb{D}_Z(t)$. The motivation of this discretization step is twofold. On the one hand, there is an efficiency concern. In fact, by using the discrete cut points as splitting values, a lower number of candidate splits is generated and evaluated. Thus the candidate evaluation phase is sped up. On the other hand, as proved in [7], a discretization step may result in an higher accuracy rate by avoiding to overfit the training data.

Differently, the condition $cond(Z)$ on a discrete property attribute Z, that is in the form $Z \in \{v_1, \ldots, v_p\}$, chooses $\{v_1, \ldots, v_p\} \subset \mathbb{D}_Z(t)$. As in [30], Mr-SMOTI relies on a non-optimal greedy strategy to determine the split values. It starts with an empty set $L_Z = \oslash$ and a full set $R_Z = \mathbb{D}_Z(t)$ and moves one discrete value from R_Z to L_Z, such that the move results in a better split. The best split is determined according to the evaluation measure σ.

4.2. Generating the Set of Candidate Regression Steps for a Regression Node

A regression node t computes a straight-line regression, that is in the form

$$\widehat{Y} = \widehat{a} + \widehat{b}X. \tag{2}$$

The variable X refers to a continuous attribute (property) which is associated to either the reference object or the task-relevant objects introduced by means of structural predicates in $Q(t)$. In addition, X has not been introduced in any regression node along the path from the root to t.

Example 4.5. Let t be a node with $Q(t) = \exists U\ mol(U) \wedge \exists V\ mol\_atom(U,V) \wedge \exists Z\ (atom\_charge(V,Z) \wedge Z \leq 0.01)$. As candidate regression variables at the node t, Mr-SMOTI considers X_1, X_2, and X_3 which represent the attributes lumo and logp of the reference object molecule and the attribute charge of the task-relevant object atom. The literals which may be added to $Q(t)$ are "$\exists X_1\ mol\_lumo(X, X_1)$", "$\exists X_2\ mol\_logp(X, X_2)$","$\exists X_3\ atom\_charge(Y, X_3)$".

The intercept a and the slope b [9] are estimated as follows:

$$\widehat{b} = \frac{\sum_i \left((x_i - \bar{X})(y_i - \bar{Y})\right)}{\sum_i \left(x_i - \bar{X}\right)} \text{ and } \widehat{a} = \bar{Y} - b\bar{X} \tag{3}$$

where $\bar{X}$ ($\bar{Y}$) is the mean computed on x_i (y_i) values for the units of analysis of reference objects in $S(t)$. When X is a non-determinate variable which denotes a property of task-relevant objects which are many-to-one associated to the reference object, X may assume multiple values in correspondence of the same reference object. Multiple values are spanned on separate attribute-value observations, one for each task-relevant object, while the response value is kept as it appears in the corresponding reference object. This attribute-value data sample is used to compute the intercept $\widehat{a}$ and the slope $\widehat{b}$.

For each regression step, Mr-SMOTI introduces a new regression variable in the model (see Figure 4). Then the linear effect of the introduced variable is removed from the data. In other words, when

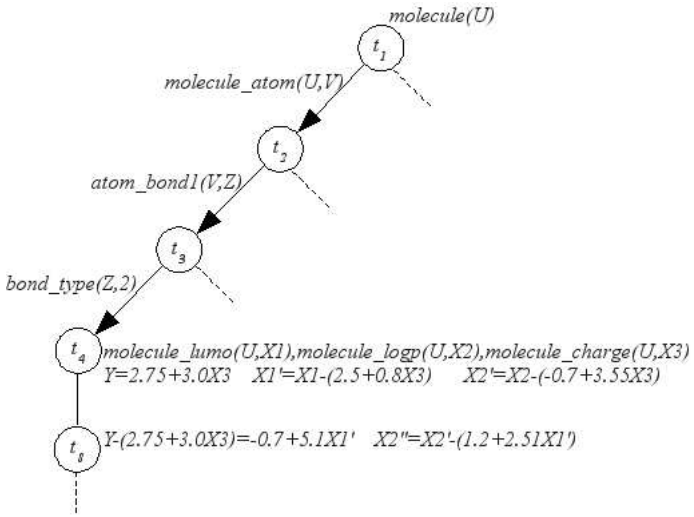


Figure 4. Regression nodes (t_4 and t_8) in a relational model tree. $Y' = Y - (2.75 + 3.0X3)$ denotes the residual of the response attribute passed down from t_4.

the linear regression function of Y on X is built, the residuals of Y and the residuals of each $X_i \neq X$ to be selected for the next regression step are computed and used for subsequent steps. X_i refers to a continuous attribute associated to either the reference object or the task-relevant objects introduced with structural predicates in $Q(t)$ and which have not been already introduced in the model by a regression node along the path from the root to t. The residuals of Y are computed as :

$$Y' = Y - (\widehat{a} + \widehat{b}X). \tag{4}$$

Similarly, residuals of each $X_i \neq X$ are computed as follows:

$$X_i' = X_i - (\widehat{a}_i + \widehat{b}_i X) \qquad X_i \neq X. \tag{5}$$

Splitting and regression nodes following the regression node t will now involve the residuals X_i' in place of X_i.

The residual computation may not be limited to the phase of introducing a regression node in the tree. A splitting node, that introduces a structural predicate, potentially introduces new continuous variables available as properties for the analysis. The effect of the regression variables already introduced in the model has to be removed from these variables. Residuals of these new variables are computed to be used in place of original variables.

4.3. Evaluating Splitting Conditions and Regression Steps

The validity of a splitting condition or a regression step at a node t is based on two heuristic measures, $\sigma(t)$ and $\rho(t)$, respectively. Both measures are founded on the evaluation measures defined in [29] for the propositional SMOTI. They enable Mr-SMOTI to choose, at each step of tree induction, the apparently most promising node according to a greedy strategy.

A candidate split t is evaluated according to the multiple linear models at the leaves. Let t_L (t_R) be the left (right) child of t, $\sigma(t)$ is defined as:

$$\sigma(t) = \frac{\#S(t_L)}{\#S(t)} R(t_L) + \frac{\#S(t_R)}{\#S(t))} R(t_R), \tag{6}$$

where $\#S(t)$ is the number of reference objects reaching t, $\#S(t_L)$ ($\#S(t_R)$) is the number of reference objects passed down to the left (right) child, and $R(t_L)$ ($R(t_R)$) is the resubstitution error of the left (right) child, that is:

$$R(t_L) = \sqrt{\frac{\sum_{ro_i \in S(t_L)} (y_i - avg(\widehat{\mathbf{y}}_\mathbf{i}))^2}{\#S(t_L)}} \left(\quad R(t_R) = \sqrt{\frac{\sum_{ro_i \in S(t_R)} (y_i - avg(\widehat{\mathbf{y}}_\mathbf{i}))^2}{\#S(t_R)}} \right). \tag{7}$$

To estimate $\widehat{Y}$, it is sufficient to consider the best straight-line regression associated to t_L (t_R). This regression potentially defines a multiple linear model due to the residuals computed at the regression nodes along the path from the root to t. These regression nodes already introduced the effect of the regression variables in the model. In this case, the main difference with the propositional SMOTI is that the estimate $\widehat{Y}$ may involve non-determinate variables and output one or *more* response values ($\widehat{\mathbf{y}}_\mathbf{i}$) for the same reference object ro_i. The solution is to aggregate the response values predicted for the same reference object and output only the "average" of these responses.

A regression step $\widehat{Y} = \widehat{a} + \widehat{b}X$ at a node t is not naïvely evaluated on the basis of the resubstitution error $R(t)$. As pointed out in [29] for the propositional case, this may result in values of $R(t)$ less or equal to values of $\sigma(t)$. Indeed, the splitting test evaluation "looks-ahead" to the best multiple linear regressions after the split at t is performed, while the regression step does not. A fairer comparison would be growing the tree at a further level and basing the computation of $\rho(t)$ on the best multiple linear regressions after the regression step at t is performed. Let t' be the child of the regression node t, and let us suppose that it performs a splitting condition. The best splitting condition in t' can be chosen on the basis of $\sigma(t')$ as indicated above. Then $\rho(t)$ can be defined as follows:

$$\rho(t) = \min\{R(t); \sigma(t')\} \tag{8}$$

In this way the criterion for selecting the optimal node is fully characterized by taking into account the multi-relational nature of the task.

4.4. Stopping Criteria

In Mr-SMOTI, three different stopping criteria are implemented. The first requires that the number of reference objects in each node be greater than a minimum threshold value. The second stops the induction process at the current node t and transforms it into a leaf node when :

- all continuous attributes (properties) associated either to the reference object or to the task-relevant objects which are introduced by means of structural predicates in $Q(t)$ have been considered in regression nodes along the path from the root to t, and

- it not possible to consider any splitting condition with a structural predicate that makes available new continuous attributes for further analysis.

The third stops the induction process when the determination coefficient at t, denoted as $R^2(t)$, is greater than a minimum threshold value. The determination coefficient is a scale-free one-number summary of the strength of the relationship between the regression variables in the actual multiple linear model and the response variable. $R^2(t)$ is computed as:

$$R^2(t) = \left(\sum_{ro_i \in S(t)} (avg(\widehat{\mathbf{y}}_\mathbf{i}) - \bar{Y}(t))^2 \right) / \left(\sum_{ro_i \in S(t)} (y_i - \bar{Y}(t))^2 \right), \quad (9)$$

where $\bar{Y}(t)$ is the mean of response values of reference objects falling in $S(t)$.

If at least one stopping criterion is satisfied, t is a leaf node. The multiple regression model at t is freely obtained by combining all univariate regression lines associated with regression nodes along the path from the root to t. The presence of non-determinate variables requires that multiple response values predicted for the same reference object are aggregated by the average.

5. Relational DBMS Integration

Scalability and efficiency are crucial issues when mining huge amount of data stored into a relational database. Motivations of database integration, translation of model trees in SQL queries and tricks to speed-up model tree induction when being tightly-coupled with a database are discussed below.

5.1. Motivations

Most data mining systems still process data in main memory resulting in high performance when enough memory is available to store all necessary data. However, in data-intensive processes it is important to exploit powerful mechanisms for accessing, filtering and indexing data, such as those available in relational database management systems (RDBMSs). For instance, organizing data according to data indexes can speed-up join SQL operations. These considerations motivate a tight coupling between data mining systems and RDBMSs in order to (1) guarantee the applicability of data mining algorithms to large datasets; (2) exploit knowledge of a data model available, free of charge, in the database schema; (3) specify which data stored in a database have to be mined; (4) avoid useless preprocessing leading to redundant data storage which may be unnecessary when part of the space of the hypothesis may be never explored; (5) support a uniform and direct access to data and patterns (expressed as SQL queries) both stored in the database.

Some examples of integration between data mining systems and DBMSs are presented in [36] for association rules, in [32] for clustering and in [37] for decision trees. In [31], the system MiningMart is proposed to approach the knowledge discovery in database by building upon database facilities and integrating data mining algorithms into database environment. In all these examples the need of data mining primitives [13] which exploit DBMS extension facilities, e.g., packages, cartridges or extenders, is advocated. Two examples of tight integration of MRDM systems with a RDBMS are MRDTL [27] and SubgroupMiner [20]. These two systems solve the tasks of classification (with decision trees) and subgroup discovery, respectively.

Following the inspiration of MRDTL, our implementation of Mr-SMOTI assumes data stored in an Oracle DBMS (11g) and exploits the structured query language (SQL) to gather all information needed to construct the model tree from multi-relational data. The regression model itself is expressed in terms of SQL query (one for each node).

5.2. Translating a Model Tree into a Hierarchy of SQL Queries

Mr-SMOTI uses SQL queries to represent splitting and regression nodes of the induced model tree.

Let t be node of the model tree, $S(t)$ be the set of reference objects which fall in t and $Q(t)$ be the conjunction of Boolean tests along the path from the root to t which intensionally describes $S(t)$. $S(t)$ can be retrieved in the database by answering an SQL query formulated as follows.

The SELECT clause projects the primary key attribute of the target table. The FROM clause joins the target table which contains the reference objects and the tables which contain the task-relevant objects which are related to the reference objects according to structural predicates which appear in $Q(t)$. The WHERE clause specifies both the Boolean expressions which are represented as property predicates in $Q(t)$ and possibly the NOT EXISTS conditions on sub-queries. These nested sub-queries are used to express the negation of a splitting condition (right child of a splitting node). An example of SQL query associated to a splitting node is reported below.

Example 5.1. Let us consider the splitting node t_3 and its descendants t_4 (left child) and t_5 (right child) which are reported in Figure 3 (and commented in Example 3.7). By considering that:

$Q(t_3) = \exists U\ mol(U)\ \wedge \exists V\ mol\_atom(U,V)\ \wedge \exists Z\ atom\_bond1(V,Z)$
$S(t_3)$ is retrieved from the database by running the SQL query:

```
SELECT distinct T.MOLID
FROM (MOLECULE T inner join ATOM T1 on T.MOLID=T1.MOLID)
            inner join BOND T2 on T1.ATOMID=T2.ATOMID1
```

Similarly,

$Q(t_4) = \exists U\ mol(U)\ \wedge \exists V\ mol\_atom(U,V)\ \wedge \exists Z\ atom\_bond1(V,Z)$
$\wedge bond\_type(Z,2).$

is transformed into the SQL query:

```
SELECT distinct T.MOLID
FROM (MOLECULE T inner join ATOM T1 on T.MOLID=T1.MOLID)
            inner join BOND T2 on T1.ATOMID=T2.ATOMID1
WHERE T2.type in ('2')
```

in order to retrieve the reference objects which fall in $S(t_4)$.
Differently,

$Q(t_5) = \exists U\ mol(U)\ \wedge \exists V\ mol\_atom(U,V)\ \wedge \exists Z\ atom\_bond1(V,Z)$
$\wedge \neg(mol(U)\ \wedge \exists V'\ mol\_atom(U,V')$
$\wedge \exists Z'\ atom\_bond1(V',Z') \wedge bond\_type(Z',2)).$

is transformed into the SQL query:

```
SELECT distinct T.MOLID
FROM (MOLECULE T inner join ATOM T1 on T.MOLID=T1.MOLID)
            inner join BOND T2 on T1.ATOMID=T2.ATOMID1
```

```
WHERE NOT EXIST(
            SELECT distinct T.MOLID
            FROM (MOLECULE T_1
               inner join ATOM T1_1 on T_1.MOLID=T1_1.MOLID )
               inner join BOND T2_1 on T1_1.ATOMID=T2_1.ATOMID1
               WHERE T2_1.type in ('2') AND T.MOLID=T_1.MOLID
               )
```

in order to retrieve the reference objects which fall in $S(t_5)$.

The regression model at a regression node t is that constructed by combining stepwise univariate regression lines along the path from the root to t. This model can be formulated as an SQL query.

The SELECT clause projects the primary key attribute of the target table and the average computed on the linear combination of the univariate regressions. Both the FROM clause and the WHERE clause are that defined above to identify $S(t)$. The GROUP BY clause groups non-determinate task-relevant objects which are related to same reference object.

Example 5.2. Let us consider the nodes t_4 and t_8 which are reported in Figure 4. t_4 is a regression node which represents the univariate regression line $LOGMUT = 2.75 + 3.0 \times CHARGE$ then the SQL query which represents this model at t_4 is the following:

```
SELECT T.MOLID, avg(2.75 + 3.0*T1.CHARGE) as EstimatedLOGMUT
FROM (MOLECULE T inner join ATOM T1 on T.MOLID=T1.MOLID)
            inner join BOND T2 on T1.ATOMID=T2.ATOMID1
WHERE T2.type in ('2') GROUP BY T.MOLID.
```

Similarly, t_8 is a regression node which represents the univariate regression line:

$$\underbrace{LOGMUT - (2.75 + 3.0 \times CHARGE)}_{LOGMUT'} = -0.7 + 5.1 \times \underbrace{(LUMO - (2.5 + 0.8CHARGE))}_{LUMO'}$$

that is, $LOGMUT = 2.75 + 3.0CHARGE - 0.7 + 5.1(LUMO - (2.5 + 0.8CHARGE))$. The SQL query which represents this model at t_8 is:

```
SELECT T.MOLID, avg(2.75 + 3.0*T1.CHARGE +
         -0.7 + 5.1*(T.LUMO-(2.5+0.8 *T1.CHARGE))) as EstLOGMUT
FROM (MOLECULE T inner join ATOM T1 on T.MOLID=T1.MOLID)
               inner join BOND T2 on T1.ATOMID=T2.ATOMID1
WHERE T2.type in ('2') GROUP BY T.MOLID.
```

5.3. Speeding-up Model Tree Induction in Databases

Mr-SMOTI uses SQL operations to compute slope and intercept of univariate linear regression functions, retrieve candidate thresholds for splitting tests, and obtain counts and resubstitution errors to evaluate splitting tests and regression steps at each level of the tree construction.

The implementation of Mr-SMOTI so far described suffers from efficiency problems that make its application to complex real-world domains infeasible in practice. As Mr-SMOTI gets further down in the model tree construction, the SQL query at the corresponding node grows in complexity. The efficiency

problem is investigated in [3] for relational decision trees: as more and more nodes are added to the tree, the longer it takes to execute the corresponding SQL query. In Mr-SMOTI some results of computations at higher levels of the model tree are stored to be reused when constructing lower levels of the tree. In particular, units of analysis defined by the SQL queries are stored in (materialized) tables. This way, join operations to retrieve multiple times these units of analysis from the database are avoided.

6. Complexity analysis

The computational complexity of adding a splitting node t to the tree depends on the complexity of generating each candidate test for t multiplied by the complexity of the best regression step selection in the children nodes t_L and t_R. On the contrary, the computational complexity of adding a regression node t depends on the complexity of a regression step selection in t multiplied by the complexity of the best splitting test in its child.

A splitting candidate test can be a conjunctive formula in the form (see Section 4.1):

1. $\exists Z\ table\_property(V, Z) \wedge cond(Z)$; or
2. $\bigwedge_{k=1}^{l} \exists Z_k\ table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k})$; or
3. $\bigwedge_{k=1}^{l} \exists Z_k\ table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k}) \wedge (\exists Z\ table\_property(Z_l, Z) \wedge cond(Z))$ where $1 \leq l \leq MaxLookahead$.

Let $\mathcal{P}$ be the maximum number candidate conjunctive formulas in the form $\bigwedge_{k=1}^{l} \exists Z_k\ table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k})$ by considering only those which guarantee linkedness of the query at the node t; $\mathcal{N}$ be the maximum number of tuples in tables involved in $\bigwedge_{k=1}^{l} \exists Z_k\ table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k})$; $\mathcal{M}_{new}$ be the total number of attributes (excluding the keys) which define the schema of the tables now added to that already considered before t; $\mathcal{M}_{old}$ be the total number of attributes (excluding the keys) which define the schema of tables already considered before t; $\mathcal{M}$ be equal to $\mathcal{M}_{old} + \mathcal{M}_{new}$; h be the number of regression steps already performed before t.

Concerning a candidate split of type (1), the cost depends on Z which is one of the continuous or discrete attributes in tables already considered before t. In the continuous case, a threshold α has to be selected for Z and the number of candidate thresholds is $\sqrt{\mathcal{N}}$. Thresholds are determined with equal-width discretization after sorting distinct values of Z. The determination of all possible continuous thresholds has a complexity O($\mathcal{M}_{old}\mathcal{N}\log\mathcal{N}$) when an optimal algorithm is used to sort the values. For each threshold, the best straight-line regression at both children has to be computed; this step has a complexity of O($\mathcal{M}_{old}\mathcal{N}$) in the worst case, hence, the number of expected thresholds is $\sqrt{\mathcal{N}}\mathcal{M}_{old}$. Hence, the splitting test has a complexity O($\mathcal{M}_{old}\mathcal{N}\log\mathcal{N} + \mathcal{M}_{old}\sqrt{\mathcal{N}}(\mathcal{M}_{old}\mathcal{N})$), that is O($\mathcal{M}_{old}^2\mathcal{N}^{\frac{3}{2}}$)

under the reasonable assumption that $\log \mathcal{N} \leq \sqrt{\mathcal{N}}$. Similarly, in the discrete case, the determination of all possible thresholds has the worst time complexity O($\mathcal{M}_{old}k^2$), where k is the maximum number of distinct values of a discrete attribute. Hence, the selection of the best discrete condition has a complexity O($\mathcal{M}_{old}^2\mathcal{N}^2$), under the reasonable assumption that $k^2 \leq \mathcal{N}$. Therefore, finding the best splitting candidate test of type (1) has a complexity O($\mathcal{M}_{old}^2\mathcal{N}^2$).

Concerning a candidate split of type (2), the cost of the computation of a conjunctive formula $\bigwedge_{k=1}^{l} \exists Z_k \; table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k})$ is O($l\mathcal{N}$) when an optimal algorithm to join tables is used[1]. For each of the $\mathcal{M}_{new}$ new variables added with $\bigwedge_{k=1}^{l} \exists Z_k \; table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k})$, h straight-line regressions (one for each regression step already done) are required to remove the effect of the variables already included in the model. The cost of removing residuals is O($h\mathcal{M}_{new}\mathcal{N}$). Finally, the best straight-line regression at both children is determined, which has a complexity of O($\mathcal{M}\mathcal{N}$) in the worst case. Therefore, finding the best splitting candidate test of type (2) has a complexity O($\mathcal{P} \cdot (l\mathcal{N} + h\mathcal{M}_{new}\mathcal{N} + M\mathcal{N})$) that is O($\mathcal{P}h\mathcal{M}\mathcal{N}$) under the assumption that $l \leq MaxLookahead \leq \mathcal{M}$ and $\mathcal{M}_{new} \leq \mathcal{M}$.

Concerning a candidate split of type (3), the condition $(\exists Z \; table\_property(Z_l, Z) \wedge cond(Z))$ has to be evaluated only for each attribute of the table Z_l. Let m_l be the number of (non key) attributes in Z_l, the cost of finding the best splitting candidate test on Z_l has a complexity O($m_l\mathcal{M}\mathcal{N}^2$). By considering that the cost of the computation of a conjunctive formula $\bigwedge_{k=1}^{l} \exists Z_k \; table_{i_k}\_table_{j_k}(Z_{i_k}, Z_{j_k})$ is O($l\mathcal{N}$) and the cost of removing residuals is O($h\mathcal{M}_{new}\mathcal{N}$), the cost of finding the best splitting candidate test of type (3) is O($\mathcal{P} \cdot (l\mathcal{N} + h\mathcal{M}_{new}\mathcal{N} + m_l\mathcal{M}\mathcal{N}^2)$). Under the assumption that $l \leq \mathcal{M}$, $h \leq \mathcal{M}$, $m_l \leq \mathcal{M}$ and $\mathcal{M}_{new} \leq \mathcal{M}$, this cost complexity is O($\mathcal{P}\mathcal{M}^2\mathcal{N}^2$).

Therefore, finding the best splitting candidate test has a complexity O($\mathcal{P}\mathcal{M}^2\mathcal{N}^2$).

The selection of the best regression step requires the computation, for each of the $\mathcal{M}_{old}$ variables, of $\mathcal{M}_{old}$ straight-line regressions (one for the regression node plus $\mathcal{M}_{old} - 1$ to remove the effect of the regressed variable) and the updating of the dataset. This takes time O($\mathcal{M}_{old}^2\mathcal{N}$). Moreover, for each straight-line regression, a splitting test is required, which has a worst case complexity of O($\mathcal{P}\mathcal{M}^2\mathcal{N}^2$). Therefore, the selection of the best regression step has a complexity of O($\mathcal{M}_{old}^2\mathcal{N} + \mathcal{P}\mathcal{M}_{old}\mathcal{M}^2\mathcal{N}^2$)), that is, O($\mathcal{P}\mathcal{M}_{old}\mathcal{M}^2\mathcal{N}^2$).

The above results lead to an O($\mathcal{P}\mathcal{M}_{old}\mathcal{M}^2\mathcal{N}^2$) worst case complexity for the selection of any node (splitting or regression).

7. Experimental results

Mr-SMOTI is a module of the MRDM system MURENA (`http://www.di.uniba.it/%7Ececi/micFiles/systems/The%20MURENA%20project.html`) and is evaluated both on artificial data and on real-world biological and spatial data in order to seek answers to the following questions.

1. Does the lookahead in determining splitting tests improve accuracy of model trees induced by Mr-SMOTI?

[1]By appropriately defining indexes, we assume that the DBMS uses a merge join algorithm.

2. Has the model tree induced by Mr-SMOTI better performances (accuracy and tree size) than the propositional model tree learners (SMOTI or M5')?

3. How does Mr-SMOTI compare to other multi-relational systems (as TILDE-RT or ReMAUVE)?

4. How effective is Mr-SMOTI in distinguishing between global effect and local effect of (possibly non-determinate) variables which are included in the constructed relational regression models?

5. Does the table materialization of intermediate SQL queries actually speed up the model tree induction?

In the next sub-Section, we present the experimental setting. Then, we illustrate results obtained with artificial data in order to answer questions 1-4 and results obtained with real data in order to answer questions 2-5.

7.1. Experimental Setting

Each dataset is analyzed by means of a k-fold cross-validation (k=10). For each dataset, the target table is divided into k blocks of nearly equal size and a subset of tuples related to the tuples of the target table block is extracted by following the foreign key constraints. This way, k databases are created. For each trial, Mr-SMOTI and competitors are trained on k-1 databases and tested on the hold-out testing database. The accuracy of trees is evaluated on the basis of the average Root Mean Squared error (AvgRMSE), that is:

$$AvgRMSE = \frac{1}{k}\sum_{i=1}^{k}\sqrt{\frac{1}{\#S_{D_i}}\sum_{ro_j \in S_{D_i}}(y_j - \widehat{y}_j)^2} \tag{10}$$

where $D = \{D_1, \ldots, D_k\}$ is the cross-validation partition, S_{D_i} is the set of reference objects stored into the target table of D_i, $\#S_{D_i}$ is the cardinality of S_{D_i} and $\widehat{y}_j$ is the value predicted for $ro_j \in S_{D_i}$. The complexity of the trees is evaluated on the basis of the average number of leaves. The pairwise comparison of distinct methods is performed by the non-parametric Wilcoxon rank test. The level of significance for the statistical tests is set at 0.05.

The experimental setting used to run TILDE-RT is that suggested by the authors [4]. This experimental setting is defined by means of user-defined schemata of candidate splitting tests which require some ILP expertise to be formulated. In particular, we use schemata where the maximum lookahead equals two. The empirical comparison with other MRDM systems is not possible since the systems are not publicly available. For ReMAUVE we consider results reported in [41]. To run the propositional learners, two transformations are considered. The former (P1) creates a single table by computing join operations for the foreign key paths rooted in the target table and selecting attributes from these tables. This transformation may create multiple tuples for the same reference object. In P1, the average of the multiple response values for the same reference object is considered as final prediction. The latter transformation (P2) differs from the previous one because it does not generate multiple tuples for the same reference object. In P2, the description of a reference object is enriched with the aggregates computed for each attribute of the task-relevant objects which are related to the reference object by a database foreign key path. As aggregation operators, we consider the average for continuous values and the mode for discrete values. This way, a single response value is directly predicted for each reference object.

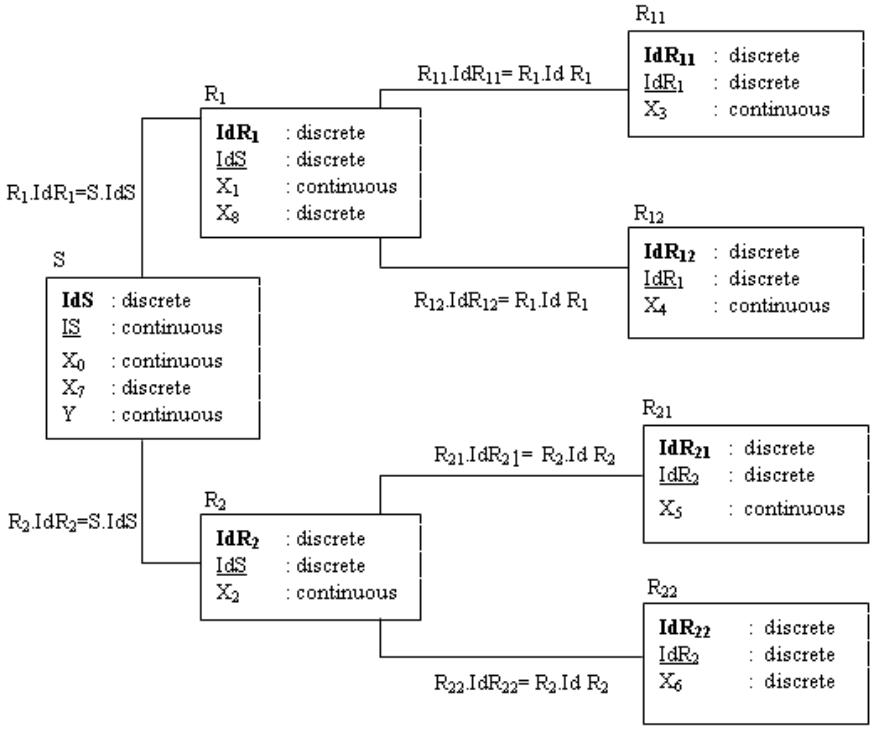


Figure 5. The database schema of artificial data. The response variable is Y. Primary keys are reported in bold style, while foreign keys are reported in underline style.

7.2. Artificial Data

Mr-SMOTI is tested on artificial data randomly generated for relational model trees with both splitting and regression nodes. Trees are built for data stored in a relational database according to the schema reported in Figure 5. The data schema includes nine explanatory variables (six continuous and three discrete) where the discrete variables assume values in the set $\{A, B, C, D\}$. Artificial trees are recursively grown on the maximum depth of the tree. The probability of adding a splitting (or regression) node to the tree is 0.5. Ten model trees are generated for each depth value, for a total of thirty trees. Twenty reference objects (tuples of the target table S in Figure 5) are randomly generated for each leaf. The relational arrangement of data is also taken into account and, for each tuple of S, twenty foreign key related tuples are generated and stored in the tables R_1 and R_2, respectively. Similarly, for each tuple of R_1 (R_2), twenty foreign key related tuples are generated and stored in the tables R_{11} and R_{12} (R_{21} and R_{22}), respectively. Data points are generated according to the constraints associated with both splitting and regression nodes. In the case of a splitting node, the distribution of cases between left and right children should take into account the number of leaves in each subtree and satisfy the semantics of opposing conditions in a relational model tree. In the case of a regression node, the constraints are the (partial) multiple linear model associated with the node as well as the linear models defined for the residuals of the attributes not included in the model. The noise is introduced by adding a normally distributed error $\sim N(0, 0.001)$ to the linear models at the leaves.

In the experiments, the minimum number of reference objects falling in an internal node is greater than the square root of the number of reference objects in the training set, and the determination coefficient in an internal node is below 0.99. To evaluate the effectiveness of the lookahead in the splitting test determination, we run Mr-SMOTI with no lookahead as it is originally presented in [1] (configuration L0), and the new version of Mr-SMOTI with $MaxLookahead = 1$ (configuration L1) and $MaxLookahead = 2$ (configuration L2). In Table 1 a comparison between the three configurations of Mr-SMOTI is reported. This table collects the number of times that a configuration outperforms the others. Table 2 collects results of Wilcoxon tests on the accuracy of trees mined by Mr-SMOTI, SMOTI, M5' and TILDE-RT. For each pair of compared systems, we report the number of tests for which it results that Mr-SMOTI works better than the other system. The number of tests statistically significant at the level of significance p is in boldface.

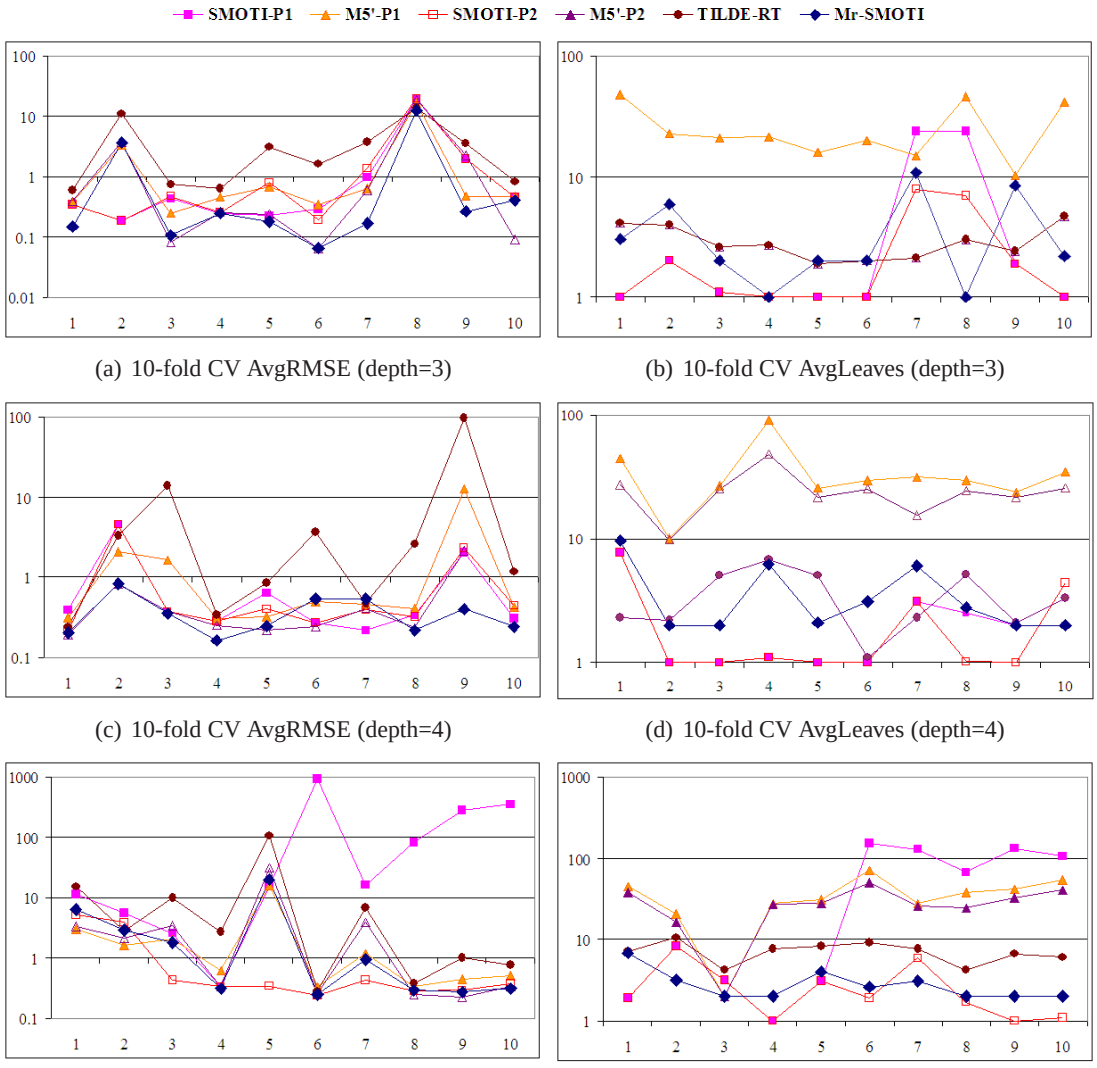
Table 1. Mr-SMOTI with L0 vs L1 vs L2: Number of times that the average RMSE (number of leaves) of the Mr-SMOTI configuration is minimum.

TREE DEPTH	Mr-SMOTI CONFIGURATION					
	RMSE			LEAVES		
	L0	L1	L2	L0	L1	L2
3	2	2	6	0	0	10
4	4	2	4	0	0	10
5	3	1	6	0	0	10
Total	9	5	16	0	0	30

Table 2. Mr-SMOTI vs SMOTI, M5' and TILDE-RT: Wilcoxon tests on the accuracy of systems. 10 datasets are generated for different tree depths. For each depth, the number of statistically significant tests ($p \leq 0.05$) are in boldface. "$+$" ("$-$") means that Mr-SMOTI performs better (worse) than SMOTI, M5' or TILDE-RT.

TREE DEPTH		P1-SMOTI		P1-M5		P2-SMOTI		P2-M5		TILDE-RT	
		+	-	+	-	+	-	+	-	+	-
L0	3	8(**4**)	2(**2**)	9(**6**)	1(**1**)	8(**5**)	3(**2**)	7(**4**)	3(**1**)	9(**8**)	1(**1**)
	4	9(**5**)	1(**1**)	9(**7**)	1(**0**)	9(**5**)	1(**1**)	6(**0**)	4(**2**)	9(**8**)	1(**1**)
	5	10(**5**)	0(**0**)	7(**6**)	3(**1**)	4(**2**)	6(**3**)	6(**3**)	4(**3**)	9(**8**)	1(**1**)
L1	3	9(**3**)	1(**1**)	9(**6**)	1(**0**)	7(**4**)	3(**1**)	7(**4**)	3(**1**)	10(**9**)	0(**0**)
	4	9(**6**)	1(**1**)	9(**7**)	1(**0**)	8(**6**)	2(**0**)	6(**2**)	4(**0**)	9(**8**)	1(**0**)
	5	10(**5**)	0(**0**)	7(**5**)	3(**2**)	6(**3**)	4(**2**)	6(**3**)	4(**2**)	9(**8**)	1(**0**)
L2	3	9(**3**)	1(**1**)	9(**7**)	1(**0**)	9(**4**)	1(**1**)	9(**5**)	1(**0**)	10(**9**)	0(**0**)
	4	9(**6**)	1(**1**)	10(**5**)	0(**0**)	10(**5**)	0(**0**)	7(**3**)	3(**0**)	10(**8**)	0(**0**)
	5	9(**8**)	1(**0**)	8(**4**)	2(**1**)	7(**3**)	3(**2**)	7(**2**)	3(**1**)	9(**8**)	1(**0**)

Several conclusions are drawn from these experimental results: first, lookahead capability improves the performances of Mr-SMOTI both in terms of RMSE and in terms of number of leaves. Second, the comparison with the relational data mining system TILDE-RT is clearly in favor of Mr-SMOTI on data generated from model trees where both local and global effects are captured. Third, Mr-SMOTI performs generally better than SMOTI and M5' independently from the data transformation, that is the structural approach improves the propositional one, although not all tests are statistically significant at the level of significance p. This last conclusion is confirmed by punctually comparing performances of



(a) 10-fold CV AvgRMSE (depth=3)

(b) 10-fold CV AvgLeaves (depth=3)

(c) 10-fold CV AvgRMSE (depth=4)

(d) 10-fold CV AvgLeaves (depth=4)

(e) 10-fold CV AvgRMSE (depth=5)

(f) 10-fold CV AvgLeaves (depth=5)

Figure 6. The average RMSE and number of leaves (Y axis) in log scale for 10 model trees (X axis) generated for different depths. The comparison regards SMOTI (squares), M5' (triangles), TILDE-RT (circles) and Mr-SMOTI (diamonds). To run SMOTI and M5', data is stored in a single table (by P1 and P2). Mr-SMOTI is run in L2 configuration.

model trees mined by the different systems. Both the average RMSE and the average number of leaves are reported in Figure 6. Results of Mr-SMOTI refer to model trees mined in the L2 configuration. Here, the number of leaves is an indicator of the complexity of the induced regression models. In this case, results show that the model induced by Mr-SMOTI is much simpler than the model induced by M5'. The relative simplicity of Mr-SMOTI regression models makes them easier to be interpreted, that is, the tree structure can be easily navigated to distinguish among global and local effects of explanatory variables.

```
The theoretical model tree (depth=3)
     |- CONT_SPLIT: X5
     ||- REG_STEP: Y = -6.08 + 9.61 X4 (case: S⋈R2 ⋈R21 and X5 ≤6.04)
     |||- LEAF: Y = 0.0 - 0.26 X0
     ||- REG_STEP: Y = -18.44 + -0.008 X0 (case: not (S⋈R2 ⋈R21 and X5 ≤6.04))
     |||- LEAF: Y = 0.0 + 0.26 X1

SMOTI-P1*
     |- CONT_SPLIT: X0
     ||- LEAF: Y = -6.14 + 9.62 X4 (case: X0 ≤ -15.37)
     ||- REG_STEP: Y = -21.39 + 0.13 X1 (case: X0 >-15.37)
     |||- LEAF: Y = -0.00000000047 + -0.06 X0
*A deeper analysis of this dataset has revealed that the split tests X5 ≤ 6.04 and X0 ≤ -15.37
equally partition the single table representation of the training data obtained according to P1.

SMOTI-P2
     |- CONT_SPLIT: X0
     ||- LEAF: Y = -6.14 + 9.62 X4 (case: X0 ≤-15.37)
     |- REG_STEP: Y = -21.59 + 0.14X1 (case: X0 >-15.37)
     |- LEAF: Y = -0.0000000014 + -0.06 X0

 Mr-SMOTI (L2)**
     |- LOOK_CONT_SPLIT: S⋈R2 ⋈R21 on X5
     ||- REG_STEP: Y = 136.13 + 5.11 X0 (case: S⋈R2 ⋈R21 and X5 ≤-8.92)
     |||- LEAF: Y = 9.85E-6 + 5.19 X5
     ||- LOOK_CONT_SPLIT: S⋈R2 ⋈R21 on X5 (case: not(S⋈R2 ⋈R21 and X5 ≤-8.92))
     |||- LEAF: Y = 4.98 + 1.15 X0 (case: S⋈R2 ⋈R21 and X5 ≤-8.06)
     |||- LEAF: Y = -18.40 - 0.008 X0(case: not (S⋈R2 ⋈R21 and X5 ≤-8.06))
** The partitioning performed by the root splitting test on X5 is close to the one performed
by the corresponding root test in the artificial tree. The only change is in moving three
reference objects from the left child to the right one.
```

Figure 7. From top to bottom: A theoretical model tree of depth 3 used in the experiments; the model trees induced by SMOTI from one of the cross-validated training sets which is transformed into an single table format according to both P1 and P2; the corresponding relational model tree built by Mr-SMOTI in a L2 configuration for the same data.

In contrast, model trees mined by M5' cannot capture global and local effects and they cannot be easily interpreted by humans because of the complexity of the models (there is an increase of one order of magnitude in the number of leaves from Mr-SMOTI to M5'-P1). The comparison between Mr-SMOTI and SMOTI requires further comments. Both systems capture global and local effects. Nevertheless, the structural approach takes advantage of the tight-coupling of Mr-SMOTI with the DBMS and avoids the consideration of useless features obtained by the data transformation. Indeed, the side effect of useless features may lead to models that overfit training data, but fail to predict new data. The comparison on the average RMSE almost always confirms the advantages of the structural approach independently from the adopted data transformation (P1 or P2). A deeper analysis of the cases where SMOTI outperforms Mr-SMOTI reveals that the higher performance of SMOTI may depend on the fact that Mr-SMOTI is provided with a lookahead capability only in exploring the candidate splitting tests, while no lookahead is performed to explore the candidate regression steps. This is confirmed by the trees reported in Figure 7. The single table representation of the multi-relational data schema makes SMOTI able to choose a candidate regression variable among all the continuous variables (X_0, X_1, X_2, X_4 and X_5) of the "entire" database schema. Differently, Mr-SMOTI chooses the regression variables only among the continuous

variables of the "portion" of the database schema, that is, the tables involved in splitting tests along the path from the root to the current node ($S \bowtie R_2 \bowtie R_{21}$ in Figure 5). Providing Mr-SMOTI with a lookahead to explore the candidate regression steps will overcome this limitation; in any case, the inherent complexity of broadening the search beam to the entire schema discourages this choice.

7.3. Real-World Data

Mr-SMOTI is tested on five real relational databases (see Table 3). Mutagenesis [39] and Biodegradability [11] are molecular databases used as a benchmark for several ILP systems. Stockport and NWE are collections of geo-referenced census data provided by the United Kingdom (UK) 1991 census and UK 1998 census, respectively. Stockport census data concerns the migration and unemployment phenomena and data is geo-referenced at the level of enumeration districts (EDs), while NWE census data includes the mortality rate and deprivation indexes geo-referenced at the ward level. Both Stockport and NWE datasets are provided in the European project SPIN!(`http://www.ais.fraunhofer.de/KD/SPIN/project.html`). Munich (`http://www.di.uniba.it/%7Ececi/micFiles/munichdb.tar.gz`) describes flats geo-referenced within the Munich subquarters.

Mr-SMOTI is run in the L2 configuration. Stopping criteria are set as in the experiments on artificial data, excepted for determination coefficient which is set at 0.9. Experimental results are reported in Table 4, where Mr-SMOTI is compared with SMOTI, M5' and TILDE-RT. Results of Wilcoxon tests are reported in Table 5.

The comparison with SMOTI is plainly in favor of Mr-SMOTI. This result confirms the advantages of the structural approach over its propositional counterpart. In addition, the low RMSE standard deviation proves that Mr-SMOTI is able to mine model trees whose predictive capability is not significantly affected by the specific cross-validation of training data. Finally, Mr-SMOTI builds model trees that are simpler than the corresponding trees built by SMOTI independently from the adopted data transformation: the highest accuracy with the simplest model. In contrast, the comparison with M5' does not exhibit a clear superiority of Mr-SMOTI, although results are still good. Most cases where M5' outperforms Mr-SMOTI occur when multi-relational data is transformed into a propositional table according to P2. In any case, when M5' outperforms Mr-SMOTI, it also outperforms TILDE-RT, that is, the other MRDM system considered in this study. Once again, the complexity of the M5' model always discourages us from interpreting it. The comparison with TILDE-RT confirms that model trees improve the accuracy of regression trees also in the multi-relational framework. The tree structure with splitting and regression nodes permits to capture both global and local effects of some explanatory attributes possibly belonging to different tables. The only database where TILDE-RT outperforms Mr-SMOTI is Mutagenesis. This happens in those database settings (B0 and B1) where the numeric attributes of molecules (i.e., lumo and logP) are not available for learning. In this case, candidate regression variables available for learning (e.g., charge of atoms) might be uninformative in predicting the response variable. Hence regression tree learners might be better than model tree learners since constants are better than linear models with uninformative regression variables (the linear model may suffer from overfitting problems). Finally, we consider accuracy and complexity of relational model trees induced by ReMAUVE on a 10-fold CV[2] of Mutagenesis database for both B1 and B2 settings. For B1, ReMAUVE reaches 1.98 average RMSE,

[2]ReMAUVE is not publicly available. We compare MR-SMOTI with ReMAUVE results reported in [41] on a differently generated 10-fold CV of Mutagenesis database. It is noteworthy that in [41] ReMAUVE is compared with an old version of Mr-SMOTI (L0) is reported. Here, we consider a new version of Mr-SMOTI run with L2 configuration.

Table 3. Real-world databases involved in the empirical evaluation of Mr-SMOTI

Code	Dataset	Database schema	Tuples	Response	Experimental Settings
D1	Mutagenesis	Molecule(**Id**, Ind1, Ind2, Lumo, LogP, Mutagenicity)	188	Mutagenicity	B0:Ind1, Ind2, Lumo and LogP are ignored
		Bond(Mol, At1, At2,Type)	5242		B1: Lumo and LogP are ignored
		Atom(**Id**, Mol,Type,Charge,Element)	4892		B2: entire database
D2	Biodegradability	Molecule(**Id**, Ind1, Ind2, Ind3, UpBound, LowBound, Average, LogP, MWeight, Biodeg)	328	Biodeg	B0: Ind1, Ind2, Ind3, UpBound, LowBound, Average, LogP, MWeight are ignored
		Bond(Mol, At1,At2,Type)	6615		B1: Ind1, Ind2, Ind3,
		Atom(**Id**, IdMol, Type)	6567		UpBound, LowBound, Average are ignored
		P2Count(Mol,Value)	1492		B2 : LogP, Mweight are ignored
		S2Count(Mol,Value)	9840		B3 : entire database
D3	Stockport Migrants	Ed(**Id**, Migrants, Unemployed, Area)	578		B0: only Ed and EdCloseToEd are consid-
		ShoppingZone(**Id**,Area)	54		ered in the mining step
D4	Stockport Unemployed	HousingZone(**Id**,Area)	9	Unemployed	B1: entire database
		EmploymentZone(**Id**,Area)	30		
		EdCloseToEd(Ed1, Ed2)	3146		
		EmpOverlapEd(Emp, Ed)	53		
		EmpOverlapEmp(Emp1, Emp2)	0		
		EmpOverlapShop(Emp, Sho)	1		
		EmpOverlapHous(Emp, Hou)	1		
		ShopOverlapEd(Sho, Ed)	134		
		ShopOverlapShop(Sho1, Sho2)	0		
		ShopOverlapHous(Sho, Hou)	0		
		HouseOverlapEd(Hs, Ed)	16		
		HouseOverlapHouse(Hs1, Hs2)	0		
D5	NWE	Wards(**Id**, Doe, Carstairs, Jarman, Townsend, MortalityRate)	212	MortalityRate	B0: entire database
		GreenAreas(Id, Type)	11		
		GreenOverlapWard(Gr, Wd)	11		
		Rail(Id)	1045		
		RailCrossWard(Rail,Wd)	1045		
		Road(Id, Type)	2763		
		RoadCrossWard(Rd,Wd, Type)	2763		
		Urban(Id,Type)	374		
		UrbanOverlapWard(Ub,Wd,)	374		
		Water(Id, Type)	1040		
		WaterCrossWard(Wt,Wd)	1040		
D6	Munich	Flats(**Id**, Extension, Year, MontlyRent)	2179	MonthlyRent	B0: entire database
		Quarter(**Id**, Area, District, Zone)	446		
		FlatInsideQuarter(Flat, Quart)	2179		
		QuarterCloseToQuarter(Q1, Q2)	1893		
		TransportStopInsideQuarter(**Id**, Quart)	111		

Table 4. Mr-SMOTI vs SMOTI, M5' and TILDE-RT: average (Avg) and standard deviation (StD) of the Root Mean Squared Error (RMSE) and number of leaves (#L) of the models mined on the 10-fold CV of databases. For SMOTI-P1, not all values are available, since the system returns error of memory. Best RMSE results are given in bold.

Databases			Mr-SMOTI		SMOTI		M5		SMOTI		M5		TILDE-RT	
			L2		P1		P1		P2		P2			
			RMSE	#L	RMSE	#L	RMSE	#L	RMSE	#L	RMSE	#L	RMSE	#L
D1	B0	Avg	1.7	11.9	2.4	124.9	1.7	660.9	2.6	22.5	**1.5**	68.2	1.51	9
		Std	0.2	1.3	1.0	6.2	0.3	14.8	2.7	2.8	0.3	1.5	0.2	1.1
	B1	Avg	1.3	9.6	1.6	108.8	1.2	697.9	1.4	21.3	**1.1**	68.5	1.2	11.7
		StD	0.1	1.1	0.5	3.6	0.3	12.7	0.4	1.76	0.28	2.71	0.30	3.23
	B2	Avg	**0.9**	3.5	2.2	34.5	1.0	140.5	1.0	18.3	1.0	71.3	1.1	14.9
		StD	0.2	0.5	2.7	11.7	0.4	5.5	0.3	4.1	0.2	1.6	0.4	4.6
D2	B0	Avg	1.4	14.2			1.4	156.6	1.6	33.8	1.4	73.5	**1.3**	7
		StD	0.2	0.6			0.1	8.1	0.3	4.2	0.1	2.2	0.2	0
	B1	Avg	**1.2**	13.1			1.7	295.6	2.1	33.5	1.3	113.9	**1.2**	7.8
		StD	0.1	1.4			0.2	18.7	1.2	3.2	0.1	2.7	0.1	1.6
	B2	Avg	0.5	2.6			**0.1**	61.3	**0.1**	13.5	0.2	39.1	0.6	4.6
		StD	0.1	0.5			0.05	19.3	0.04	1.2	0.1	2.5	0.2	1.3
	B3	Avg	0.4	3.5			**0.1**	54.4	0.05	13.8	0.2	39.6	0.6	4.6
		StD	0.2	0.5			0.04	4.1	0.03	1.0	0.1	2.8	0.2	1.3
D3	B0	Avg	**16.5**	19.6	18.3	143.5	22.5	730	23.4	50.1	18.7	224.2	**16.5**	6
		StD	1.3	0.8	2.9	8.9	12.4	113.7	7.7	8.7	4.2	4.1	1.9	2.3
	B1	Avg	**16.4**	18	**16.4**	145.5	39.9	708.8	24.6	45.4	18.4	222.5	**16.4**	5.9
		StD	1.6	6.5	2.4	24.6	50.9	121.0	8.4	17.2	3.8	3.9	1.9	2.3
D4	B0	Avg	**10.3**	18.2	12.5	135.1	11.4	754.6	20.8	37.4	11.8	223.9	10.5	7.6
		StD	1.6	1.2	3.1	33.0	3.4	53.3	11.5	7.0	2.7	3.3	1.9	0.5
	B1	Avg	**10.32**	17	15.53	142	11.19	735.9	17.90	48.3	12.96	224.3	10.48	7.4
		StD	1.652	0	10.17	9.46	2.95	66.73	1.45	6.79	2.496	3.68	1.927	0.51
D5		Avg	0.003	10.6	0.003	49.5	0.003	200	0.003	25.8	**0.002**	82.5	**0.002**	1.8
		StD	0.0007	0.8	0.02	8.1	0.0004	7.5	0.001	2.2	0.0003	2.6	0.00	0.8
D6		Avg	4.8	36.7	5.9	234.2	5.2	2236.2	5.5	93	**4.6**	812.1	4.8	8
		Std	0.2	1.41	0.6	6.6	0.5	60.8	0.73	4.3	0.2	6.1	0.2	3.2

Table 5. Mr-SMOTI vs SMOTI, M5' and TILDE-RT (TILDE-RT vs SMOTI and M5'): Wilcoxon test on the accuracy of systems. The symbol "+" ("-") means that Mr-SMOTI (TILDE-RT) performs better (worse) than SMOTI, M5' or TILDE-RT (SMOTI or M5'). "++" ("–") denotes the statistically significant values ($p \leq 0.05$).

Databases		Mr-SMOTi vs					TILDE-RT vs			
		SMOTI	M5	SMOTI	M5	TILDE-RT	SMOTI	M5	SMOTI	M5
		P1		P2			P1		P2	
D1	B0	++	+	+	-	-	++	+	+	+
	B1	+	-	-	–	-	++	+	+	-
	B2	+	+	+	+	+	+	+	+	-
D2	B0		+	+	+	+	-	+	+	+
	B1		++	++	+	+	-	+	+	+
	B2		–	–	–	+	-	–	–	–
	B3		–	–	–	++	-	–	–	–
D3	B0	++	++	++	++	+	++	++	++	++
	B1	+	++	++	+	+	+	++	++	+
D4	B0	++	++	++	++	+	++	++	++	++
	B1	++	++	+	++	+	+	++	+	++
D5		++	+	+	-	+	++	++	++	–
D6		++	++	++	–	++	++	++	++	–

with trees that contain, in average, 7 leaves. Differently, in Mr-SMOTI, we have 1.25 average RMSE, with trees that contain, in average, 9.6 leaves. For B2, ReMAUVE reaches 1.45 average RMSE, with trees that contain, in average, 3 leaves. For the same dataset, in Mr-SMOTI, we have 0.95 average RMSE, with trees that contain, in average, 3.5 leaves. By keeping in mind that this comparison has no statistical validity, Mr-SMOTI shows better accuracy than ReMAUVE at the price of slightly more complex models. No result is available for ReMAUVE on other databases considered in this study. In conclusion, Mr-SMOTI almost always improves the accuracy of SMOTI, M5' and TILDE-RT on spatial databases, while the advantages of Mr-SMOTI are not so evident on biological databases. In addition to the considerations stated above, this also depends on the fact that, although regression problems have been defined for both Mutagenesis and Biodegradability databases, graph mining methods (e.g., [18]) are known to perform much better on this type of data; however, the investigation of graph mining methods is beyond the scope of this paper. Conversely, Mr-SMOTI faces the problems raising from the spatial data [28], that is, i) the explanatory attributes that influence the response attribute may not come from a single spatial layer and ii) some explanatory attributes may have a spatially global effect on the response attribute, while other attributes have only a spatially local effect. In Example 7.1, the Munich database is analyzed to show the interpretability of the global effects in the mined tree.

Example 7.1. For all 10-CV Munich databases, Mr-SMOTI builds a tree with a regression node in the root. The straight-line regression at the root is almost invariant for all trees and expresses the linear dependence between the degree of monthly rent per square meter of flats (response variable) and the year of construction (explanatory variable). This represents a global effect, which cannot be grasped by the leaves of the tree induced by TILDE-RT on the same data. Interestingly, the child of the root is always a splitting test on the residual of the flat extension once the effect of the year of construction has been removed. As for the root, the threshold selected for this split is almost the same for all trees. By exploring the left child of this splitting node, a new splitting test is performed. The data structure is

explored and a discrete test on the district name is performed by navigating the "inside" relation between flats and subquarters. The threshold of the split is almost the same for all trees, that is, the Munich area is spatially partitioned in almost the same way.

Although detecting global and local effect of variables is desirable, this has a computational cost. Mr-SMOTI is sped up by materializing SQL queries at intermediate nodes and using materialized results to grow the subsequently tree. Figure 8 plots the computation time of Mr-SMOTI (L2 configuration) for real databases. Results confirm the advantages of materialization except for Biodegradability (B3 and B4), where model trees include only variables from the target table and no join is associated to the intermediate nodes.

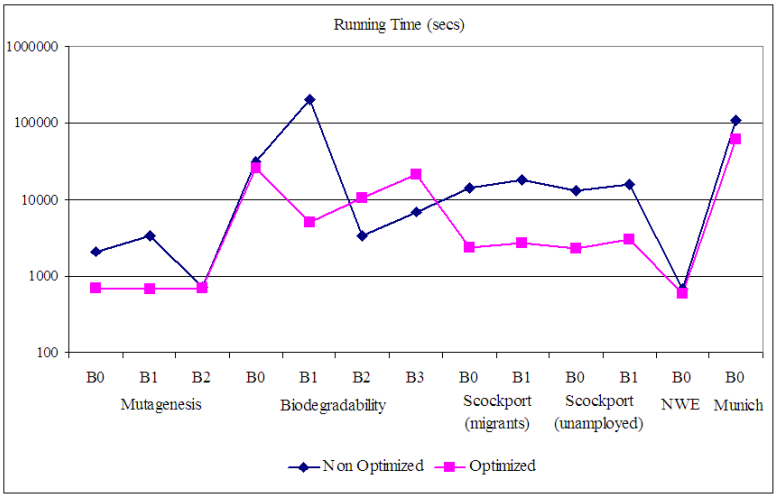


Figure 8. Mr-SMOTI (L2) running time on real databases. Experiments are performed on Intel Pentium 4 - 2.00 GHz CPU RAM 532Kb running Windows Professional 2000. Results show that materializing SQL queries at intermediate nodes speeds-up the stepwise construction of model trees with regression and splitting nodes (Optimized outperforms Non Optimized version). Running times are plotted according to a logarithmic scale.

8. Conclusions

We propose a model tree learner system, Mr-SMOTI, that deals with data stored in several tables of a tightly-coupled relational database. Relational model trees with both splitting nodes and regression nodes are built. The splitting nodes are associated with splitting tests involving one or more tables of the database schema, while the regression nodes are associated with straight-line regressions. The tree is built stepwise by navigating the database schema and by distinguishing variables having a global effect from others having only a local effect. Variables of the regression nodes selected at higher levels in the tree have a global effect, which is shared by all multiple models associated with the leaves. Nodes are represented as SQL queries stored in XML format. SQL queries at leaves can be subsequently used for predicting (unknown) response variables. The comparison between Mr-SMOTI and the propositional systems, SMOTI and M5', as well as the ILP system TILDE-RT, is performed on both artificial and real databases. The comparison demonstrates that Mr-SMOTI is competitive with respect to these

competitors. Advantages of solving the multi-relational regression problem in its original representation and detecting global and local effect of variables are not at the expense of predictive accuracy or model complexity.

Acknowledgments

This work has been carried out in fulfillment of the research objectives of the PRIN 2009 Project "Learning Techniques in Relational Domains and Their Applications" funded by the Italian Ministry for Universities and Research (MIUR). Authors wish to thank Lynn Rudd for help in correcting the paper.

References

[1] Appice, A., Ceci, M., Malerba, D.: Mining Model Trees: A Multi-relational Approach, in: *Proc. of the 13th International Conference on Inductive Logic Programming, ILP 2003*, vol. 2835 of *LNAI*, Springer-Verlag, 2003, 4–21.

[2] Appice, A., Ceci, M., Malerba, D.: MR-SMOTI: A Data Mining System for Regression Tasks Tightly-Coupled with a Relational Database, *KDID* (J.-F. Boulicaut, S. Dzeroski, Eds.), Rudjer Boskovic Institute, Zagreb, Croatia, 2003, ISBN 953-6690-34-9.

[3] Atramentov, A., Leiva, H., Honovar, V.: A Multi-relational Decision Tree Learning Algorithm, in: *Proceedings of the 13th International Conference on Inductive Logic Programming, ILP 2003*, vol. 2835 of *LNAI*, Springer-Verlag, 2003, 38–56.

[4] Blockeel, H.: *Top-down induction of first order logical decision trees*, Ph.D. Thesis, Department of Computer Science, Katholieke Universiteit, Leuven, Belgium, 1998.

[5] Blockeel, H., Sebag, M.: Scalability and efficiency in multi-relational data mining, *SIGKDD Explorations Newsletters*, **5**(1), 2003, 17–30.

[6] Botta, M., Giordana, A., Saitta, L., Sebag, M.: Relational learning: Hard problems and phase transition, in: *Congress of the Italian Association for Artificial Intelligence, AIIA 1999*, vol. 1792 of *LNAI*, Springer-Verlag, 2000, 178–189.

[7] Catlett, J.: On changing continuous attributes into ordered discrete attributes, *Proceedings of the Fifth European Working Session on Learning*, 1991.

[8] Chaudhuri, S., Das, G., Narasayya, V., Datar, M., Motwani, R.: Overcoming Limitations of Sampling for Aggregation Queries, *Proc. of the 17th International Conference on Data Engineering, ICDE 2001*, IEEE Computer Society, 2001.

[9] Draper, N. R., Smith, H.: *Applied regression analysis*, Wiley, 1982.

[10] Driessens, K., Džeroski, S.: Combining model-based and instance-based learning for first order regression, *Proceedings of the 22th International Conference on Machine Learning, ICML 2005*, ACM, 2005.

[11] Džeroski, S., Blockeel, H., Kramer, S., Kompare, B., Pfahringer, B., Van Laer, W.: Experiments in predicting biodegradability, in: *International Workshop on Inductive Logic Programming ILP 1999*, vol. 1634 of *LNAI*, Springer-Verlag, 1999, 80–91.

[12] Džeroski, S., Todoroski, L., Urbancic, T.: Handling real numbers in inductive logic programming: A step towards better behavioural clones, in: *European Conference of Machine Learning, ECML 1995*, vol. 912 of *LNAI*, Springer-Verlag, 1995, 283–286.

[13] Freitas, A. A., Lavington, S. H.: Using SQL primitives and parallel DB servers to speed up knowledge discovery in large relational databases, *Proceedings of the 13th European Meeting on Cybernetics and Systems Research, Cybernetics and Systems 1996*, 1996.

[14] Helft, N.: *Progress in Machine Learning*, chapter Inductive generalization: a logical framework, Sigma Press, 1987, 149–157.

[15] Ikonomovska, E., Džeroski, S.: Regression on evolving multi-relational data streams, *Proceedings of the 2011 Joint EDBT/ICDT Ph.D. Workshop*, PhD '11, ACM, New York, NY, USA, 2011, ISBN 978-1-4503-0696-6.

[16] Karalic, A.: Linear regression in regression tree leaves, *Proceedings of International School for Synthesis of Expert Knowledge, ISSEK 1992*, 1992.

[17] Karalic, A., Bratko, I.: First Order Regression, *Machine Learning*, **26**, 1997, 147–176.

[18] Karwath, A., De Raedt, L.: Predictive Graph Mining, *Proc. of the 2nd International Workshop on Mining Graphs, Trees and Sequences*, 2004.

[19] Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A. S.: The Web as a Graph: Measurements, Models and Methods, *LNCS*, **1627**, 1999, 1–17.

[20] Klosgen, W., May, M.: Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database, in: *Proc. of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2002*, vol. 2431 of *LNAI*, Springer-Verlag, 2002, 275–286.

[21] Kramer, S.: Structural regression trees, *Proc. of the National Conference on Artificial Intelligence*, 1996.

[22] Kramer, S.: *Relational Learning vs. Propositionalization: Investigations in Inductive Logic Programming and Propositional Machine Learning*, Ph.D. Thesis, Vienna University of Technology, Vienna, Austria, 1999.

[23] Kramer, S.: Demand-Driven Construction of Structural Features in ILP, in: *Proc. of the 11th International Conference on Inductive Logic Programming, ILP 2001*, vol. 2157 of *LNAI*, Springer-Verlag, 2001, 132–141.

[24] Kramer, S., Lavrač, N., Flach, P.: *Relational Data Mining*, chapter Propositionalization Approaches to Relational Data Mining, LNAI, Springer-Verlag, 2001, 262–291.

[25] Kramer, S., Widmer, G.: *Relational Data Mining*, chapter Inducing Classification and Regression Trees in First Order Logic, LNAI, Springer-Verlag, 2001, 140–156.

[26] Lavrač, N., Džeroski, S.: *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, 1994.

[27] Leiva, H. A.: *MRDTL: A multi-relational decision tree learning algorithm*, Master Thesis, University of Iowa, 2002.

[28] Malerba, D.: A relational perspective on spatial data mining, *IJDMMM*, **1**(1), 2008, 103–118.

[29] Malerba, D., Esposito, F., Ceci, M., Appice, A.: Top Down Induction of Model Trees with Regression and Splitting Nodes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(5), 2004, 612–625.

[30] Mehta, M., Agrawal, R., Rissanen, J.: SLIQ: A fast scalable classifier for data mining, *Proc. of the 5th International Conference on Extending Database Technology*, 1996.

[31] Morik, K., Scholz, M.: The MiningMart Approach to Knowledge Discovery in Databases, *Handbook of Intelligent IT*, 2003.

[32] Ordonez, C., Cereghini, P.: SQLEM: Fast Clustering in SQL using the EM Algorithm, *Proc. of the ACM SIGMOD 2000*, 29, 2000.

[33] Page, D., Craven, M.: Biological applications of multi-relational data mining, *SIGKDD Explor. Newsl*, **5**(1), 2003, 69–79, ISSN 1931-0145.

[34] Raedt, L. D.: Attribute-Value Learning Versus Inductive Logic Programming: The Missing Links (Extended Abstract), *Proc. of the 8th International Workshop on Inductive Logic Programming, ILP 1998*, 1446, Springer-Verlag, 1998, ISBN 3-540-64738-4.

[35] Raedt, L. D., Idestam-Almquist, P., Sablon, G.: Theta-Subsumption for Structural Matching, *Proc. of the 9th European Conference on Machine Learning, ECML 1997*, 1224, Springer-Verlag, 1997, ISBN 3-540-62858-4.

[36] Sarawagi, S., Thomas, S., Agrawal, R.: Integrating Mining with Relational Database Systems: Alternatives andImplications, *Proc. of ACM SIGMOD 1998*, 1998.

[37] Sattler, K., Dunemann, O.: SQL Database Primitives for Decision Tree Classifiers, *Proc. of the 10th ACM International Conference on Information and Knowledge Management, ACM CIKM 2001*, 2001.

[38] Scott, J. P.: *Social Network Analysis: A Handbook*, Sage Publications, 2005.

[39] Srinivasan, A., Muggleton, S., King, R. D., Sternberg, M. J. E.: Mutagenesis: ILP experiments in a non-determinate biological domain, *Proc. of the 4th Inductive Logic Programming Workshop, ILP 1994*, GMD-Studien, 1994.

[40] Van Laer, W., De Raedt, L.: *Relational Data Mining*, chapter How to Upgrade Propositional Learners to First Order logic: A Case Study, LNAI, Springer-Verlag, 2001, 235–261.

[41] Vens, C., Ramon, J., Blockeel, H.: ReMauve: A Relational Model Tree Learner, *Proc. of the 16th International Conference on Inductive Logic Programming, ILP 2006*, 4455, Springer-Verlag, 2006, ISBN 978-3-540-73846-6.