Mining Model Trees: A Multi-relational Approach

Annalisa Appice, Michelangelo Ceci, and Donato Malerba

Dipartimento di Informatica, Università degli Studi
Via Orabona, 4 - 70126 Bari, Italy
{appice,ceci,malerba}@di.uniba.it

Abstract. In many data mining tools that support regression tasks, training data are stored in a single table containing both the target field (dependent variable) and the attributes (independent variables). Generally, only intra-tuple relationships between the attributes and the target field are found, while inter-tuple relationships are not considered and (inter-table) relationships between several tuples of distinct tables are not even explorable. Disregarding inter-table relationships can be a severe limitation in many real-word applications that involve the prediction of numerical values from data that are naturally organized in a relational model involving several tables (multi-relational model). In this paper, we present a new data mining algorithm, named Mr-SMOTI, which induces model trees from a multi-relational model. A model tree is a tree-structured prediction model whose leaves are associated with multiple linear regression models. The particular feature of Mr-SMOTI is that internal nodes of the induced model tree can be of two types: regression nodes, which add a variable to some multiple linear models according to a stepwise strategy, and split nodes, which perform tests on attributes or the join condition and eventually partition the training set. The induced model tree is a multi-relational pattern that can be represented by means of selection graphs, which can be translated into SQL, or equivalently into first order logic expressions.

1 Introduction

Prediction is arguably considered the main goal of data mining, with the greatest potential payoff [28]. The two principal prediction problems are *classification* and *regression*. Samples of past experience with known answers (labels) are examined and generalized in future cases. For classification labels are a finite number of unordered categories. For regression the answer is a number. Traditionally, in a regression problem sample data are described by a set of *m independent* variables X_i (both numerical and categorical) and a *dependent* variable Y, which normally takes values in $\hat{A}$. According to the data mining terminology, X_i's are the *attributes*, while Y is the *target* field.

Regression problems have been very well studied in statistics. In general, the model is assumed to be a linear combination of independent variables and the coefficients of the combination are determined by the method of the least squares [4]. Refinements and extensions to non-linear models are also well-known and applied in many real world applications. However, classical statistical methods have several limitations. First, (non-)linear regression models are often hard to understand. Second,

T. Horváth and A. Yamamoto (Eds.): ILP 2003, LNAI 2835, pp. 1-21, 2003.

all these statistical models are based on the assumption that all independent variables are equally relevant in the whole sample space. Third, the least square method does not allow prior domain knowledge to be used in the construction of the regression model. To solve some of these problems regression tree methods have been developed.

Regression trees [3] are supposed to be more comprehensible then classical regression models. They are built top-down by recursively partitioning the sample space. An attribute may be of varying importance for different regions of the sample space. A constant is associated to each leaf of a regression tree, so that the prediction performed by a regression tree is the same for all sample data falling in the same leaf.

A generalisation of regression trees is represented by model trees, which associate multiple linear models with each leaf. Hence, different values can be predicted for sample data falling in the same leaf. Some of the model tree induction systems are M5 [23], RETIS, [9], M5' [27], HTL [26], TSIR [17], and SMOTI [18]. The last two systems are characterised by a tree structure with two types of nodes: regression nodes, which perform only straight-line regression, and splitting nodes, which partition the feature space. The multiple linear model associated to each leaf is then the composition of the straight-line regressions reported along the path from the root to the leaf. In [18] some differences between TSIR and SMOTI have been reported, the most important of which is that the method implemented in SMOTI is the only one for which the composition of straight-line regressions found along a path from the root to a leaf can be correctly interpreted as a multiple linear model built stepwise.

All model-tree induction systems reported above work on data represented by m+1 attribute-value pairs. They input training data from a file, with the exception of SMOTI, which interfaces a relational database and requires the data set to be stored in a single table. Therefore, all methods implemented in the current model tree induction systems are based on the *single-table assumption*, according to which all training samples are stored in a single table (or "relation" in database terminology), and that there is one row (or "tuple") in this table for each object of interest [29]. In other words, training data can be described by a fixed set of attributes, each of which can only have a single, primitive value.

The single-table assumption underlying this representation paradigm only allows fairly simple objects to be analysed by means of current model tree systems. More complex structured objects require a representation in a relational database containing multiple tables [12]. (Multi-)relational data mining (MRDM) algorithms and systems are capable of directly dealing with multiple tables or relations as they are found in today's relational databases [8]. The data taken as input by MRDM systems consists of several tables and not just one table (multi-relational model). Moreover, the patterns output by these systems are relational, that is, involve multiple relations from a relational database. They are typically stated in a more expressive language than patterns described on a single data table. For instance, subsets of first-order logic are used to express relational patterns. Considering this strong link with logics, it is not surprising that many algorithms for MRDM originate from the field of inductive logic programming (ILP) [15]

In this paper, we present a multi-relational extension of SMOTI. The new system, named Mr-SMOTI, induces *relational model trees* from structural data possibly described by multiple records in multiple tables. As in the case of SMOTI, induced relational model trees can contain both regression and split nodes. Differently from

SMOTI, attributes involved in both types of nodes can belong to different tables of the relational database. The join of these tables is dynamically determined on the basis of the database schema and aims to involve attributes of different relations in order to build a predictive model for a target field.

In the next section we first review related works in order to clarify the innovative aspects of our approach. In Section 3 we briefly introduce the method implemented in SMOTI that operates under the single-table assumption. In Section 4 we draw on the multi-relational regression framework, based on an extended graphical language (*selection graph*), to mine relational model trees directly from relational databases, through SQL queries. In Section 5 we show how selection graphs can support the stepwise induction of multi-relational model trees from structural data. In Section 6 we present some experimental results. Finally, we draw some conclusions and sketch possible directions of further research.

2 Related Work

The problem of mining patterns (e.g. prediction models) over data that reside in multiple tables is generally solved by *moulding* a relational database into a single table format, such that traditional attribute-value algorithms are able to work on [13]. This approach corresponds to the concept of *propositionalization* in machine learning and has been applied to regression tasks as well. In [7], the DINUS [15] algorithm is applied to transform a Datalog representation of a dynamic system into a propositional form (i.e., attribute-value pairs), so that a classical model tree induction system based on the single-table assumption (e.g. RETIS) can be applied. One way of performing the propositionalization is to create a single relation by deriving attributes from other joined tables. However, this produces an extremely large table with lots of data being repeated which is difficult to handle. A different approach is the construction of a single central relation that summarises and/or aggregates information which can be found in other tables. Also this approach has some drawbacks, since information about how data were originally structured is lost. Therefore, a proper way of explicitly and efficiently dealing with multiple relations is necessary.

The idea of mining relational regression models from multiple relations is not new. In particular, the learning problem of *Relational Regression* [5] has been formulated in the *normal* ILP framework. So far two approaches to solve Relational Regression problems have been proposed in ILP. The former uses a *separate-and-conquer* (or sequential covering) strategy to build a set of Prolog clauses. The latter uses a *divide-and-conquer* strategy to induce tree-based models and then translate these models into Prolog programs. A system that follows the first approach is FORS [10], while three systems that follow the second approach are SRT [14], S-CART [8], and TILDE-RT [2]. In contrast to DINUS/RETIS all these systems solve a Relational Regression problem in its original representation, and do not require transformation of the problem. Moreover, they can utilise relational non-determinate background knowledge. SRT generates a series of increasingly complex trees containing a literal (an atomic formulation or its negation) or a conjunction of literals in each node, and subsequently returns the best tree according to a criterion based on minimum

description length. A numerical constant value is assigned to each leaf. Analogously, S-CART and TILDE-RT follow the general procedure of top-down tree induction [3]. In particular, S-CART recursively builds a binary tree, selecting a possible conjunction of one or more literals in each node as provided by user-defined schemata [25] until a stopping criterion is fulfilled. The value predicted in a node is simply the mean of values of all examples covered by the node. The algorithm keeps track of the examples in each node and the conjunctions of literals in each path leading to the respective node. This information can be turned into a clausal theory (e.g. a set of first order regression rules).

All these approaches are mostly based on data stored as Prolog facts. Moreover, in real-world applications, where facts correspond to tuples stored on relational databases, some pre-processing is required in order to transform tuples into facts. However, much of the pre-processing, which is often expensive in terms of computation and storage, may be unnecessary since that part of the hypothesis space may never be explored. In addition, in applications where data can frequently change, pre-processing has to be frequently repeated. This means that little attention has been given to data stored in relational database and to how knowledge of a data model can help to guide the search process [3, 13]. A solution can be found by combining the achievements of the Knowledge Discovery in Database (KDD) field on the integration of data mining with database systems, with some results reported in the ILP field on how to correctly upgrade propositional data mining algorithms to multi-relational representations. In the next sections we show how to develop a new multi-relational data mining system by upgrading SMOTI to multi-relational representations and by tightly integrating the new system with a relational DBMS, namely Oracle^{R} 9i.

3 Stepwise Model Tree Induction

SMOTI (Stepwise Model Tree Induction) performs the top-down induction of models trees by considering not only a partitioning procedure, but also by some intermediate prediction functions [18][1]. This means that there are two types of nodes in the tree: regression nodes and splitting nodes (Fig. 1). The former compute straight-line

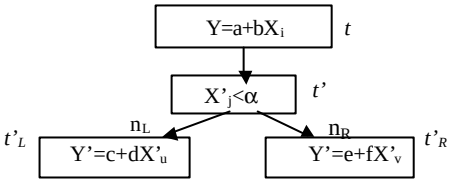


Fig. 1. A model tree with both a regression node (t) and a splitting node (t')

regressions, while the latter partition the sample space. They pass dowrn training data to their children in two different ways. For a splitting node *t*, only a subgroup of the *N(t)* training data in *t* is passed to each child, with no change on training cases. For a regression node *t*, all the data are passed down to its only child, but the values of both

[1] The work reported in [18] has been substantially extended and revisited. Details of the improved algorithm implemented in the data mining system KDB2000 are reported in [19].

the dependent and independent numeric variables not included in the multiple linear model associated to t are transformed in order to remove the linear effect of those variables already included. Thus, descendants of a regression node will operate on a modified training set. Indeed, according to the statistical theory of linear regression [4], the incremental construction of a multiple linear model is made by removing the linear effect of introduced variables each time a new independent variable is added to the model.

For instance, let us consider the problem of building a multiple regression model with two independent variables through a sequence of straight-line regressions:

$$\hat{Y} = a + bX_1 + cX_2.$$

We start regressing Y on X_1, so that the model $\hat{Y} = a_1 + b_1X_1$ is built. This fitted equation does not predict Y exactly. By adding the new variable X_2, the prediction might improve. Instead of starting from scratch and building a model with both X_1 and X_2, we can build a linear model for X_2 given X_1: $\hat{X}_2 = a_2 + b_2X_1$. Then we compute the residuals on X_2: $X'_2 = X_2 - (a_2 + b_2X_1)$ and on Y: $Y' = Y - (a_1 + b_1X_1)$. Finally, we regress Y' on X'_2 alone: $\hat{Y}' = a_3 + b_3X'_2$.

By substituting the equations of X'_2 and Y' in the last equation we have:

$$\widehat{Y-(a_1 + b_1X_1)} = a_3 + b_3(X_2 - (a_2 + b_2X_1)).$$

Since $\widehat{Y - (a_1 + b_1X_1)} = \hat{Y} - (a_1 + b_1X_1)$ we have:

$$\hat{Y} = (a_3 + a_1 - a_2b_3) + (b_1 - b_2b_3)X_1 + b_3X_2.$$

It can be proven that this last model coincides with the first model built, that is, $a = a_3 + a_1 - a_2b_3$, $b = b_1 - b_2b_3$ and $c = b_3$. Therefore, when the first regression line of Y on X_1 is built we pass down both the residuals of Y and *the residuals of the regression of* X_2 *on* X_1. This means that we remove the linear effect of the variables already included in the model (X_1) from both the response variable (Y) and those variables to be selected for the next regression step (X_2).

4 Regression Problem in a Multi-relational Framework

Traditional research for a regression task in KDD has focused mainly on propositional techniques involving the attribute-value paradigm. This implies that relationships between fields of one tuple can be found, but not relationships between several tuples of one or more tables. It seems that this is an important limitation, since a relational database consists of a set of tables and a set of associations between pairs of tables. Both tables and associations are known as relations. Each association describes how records in one table relate to records in another table. Most associations correspond to *foreign key relations.* These relations can be seen as having two directions. One goes from a table where the attribute is primary key to a table where the attribute is foreign key (*one-to-many*), and the other one is in the reverse way (*many-to-one*). An object in a relational database can consist of several records fragmented across several tables and connected by associations (Fig. 2). Although the data model can consist of multiple tables, there must be only a single kind of object that is central to the analysis (*target table*). The assumption is that each record in the target table will correspond to

a single object in the database. Any information pertaining to each object which is stored in other tables can be retrieved by following the associations in the data model. Once the target table has been selected, a particular numeric attribute of that table can be chosen for regression purposes (*target attribute*).

Thus, a multiple regression problem in a multi-relational framework can be defined as follows. Given a schema of a relational database D, a target table T_0, a target attribute Y within the target table T_0, the goal is to mine a multi-relational multiple regression model to predict the estimated target attribute Y. Mined models not only involve attribute-value descriptions, but also structural information denoted by the associations in D.

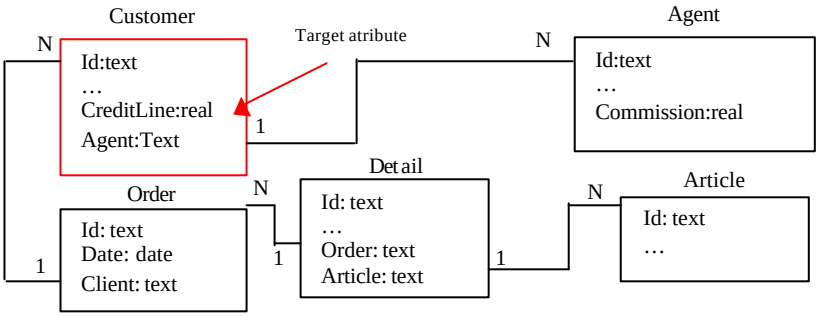


Fig. 2. The data model of an example database used in relational regression

Relational regression models induced stepwise as in SMOTI can be expressed in the graphical language of *selection graphs*. The classical definition of a selection graph is reported in [11, 12, 16]. Nevertheless, we present an extension of this definition in order to make the selection graphs more appropriate to our task.

Definition of *selection graph*

A selection graph G is a directed graph (N, A), such that:

– each node in N is a 4-tuple (T, C, R, s), named *selection node*, where:
  - $T = (X_1, X_2, \ldots X_n)$ is a table in the relational schema D.
  - C is a set of *conditions* on attributes in T of type $T.X'_i \; OP \; c$, where X'_i is one of the attributes X_i in T after the removal of the effects of some variables already introduced in the relational regression model through regression nodes. OP is one of the usual comparison operators ($<$, $\geq$, in, not in …) and c is a constant value.
  - R is a set of tuples $R=\{(RX_j, a_j, b_j) \mid j=1,\ldots,l\}$ where RX_j is a regression term already introduced in the multiple linear model, l is the number of such terms, $a_j = (a_{j1}, a_{j2}, \ldots, a_{jn})$ and $b_j = (b_{j1}, b_{j2}, \ldots, b_{jn})$ are the regression coefficients computed to remove the effect of each term RX_j from all numerical attributes in T:
    $$X'_i = X_i - \sum_{j=1,\ldots,l} (a_{ji} + b_{ji} \times RX_j) \quad \forall \; i = 1,\ldots,n \text{ and } X_i \text{ is numerical}$$
  - s is a flag with possible values *open* or *closed*.

– A, a set of tuples (p, q, fk, e), where:
  - p and q are selection nodes.

- *fk* is a foreign key association between *p.T* and *q.T* in the relational schema *D* (one-to-many or many-to-one).
- *e* is a flag with possible values *present* or *absent*.

Selection graphs contain at least a node n_0 that corresponds to the target table T_0. They can be graphically represented by a directed labelled graph (Fig. 3.a). The value of *s* is expressed by the absence or presence of a cross in the node, representing the value *open* and *close*, respectively. Similarly the value for *e* is indicated by the presence (*absent* value) or absence (*present* value) of a cross on the corresponding arrow representing the labelled arc. The direction of the arrow (left-to-right and right-to-left) corresponds to the multiplicity of the association *fk* (one-to-many and many-to-one, respectively). Every arc between the nodes *p* and *q* imposes some constraints on how one or more records in the table *q.T* are related to each record in table *p.T*, according to the list of conditions in *q.C*. The association between *p.T* and *q.T* induces some grouping (Fig. 3.b) in the records in *q.T*, and thus selects some records in *p.T*. In particular, a *present* arc selects those records that belong to the *join* between the tables and match the list of conditions. On the other hand, an *absent* arc corresponds to the negation of the joining condition and the representations of the complementary sets of objects. Intuitively, the tuples in the target table T_0 that are explained by a selection graph *G* are those for which tuples exist or not in linked tables that satisfy the conditions defined for those tables.

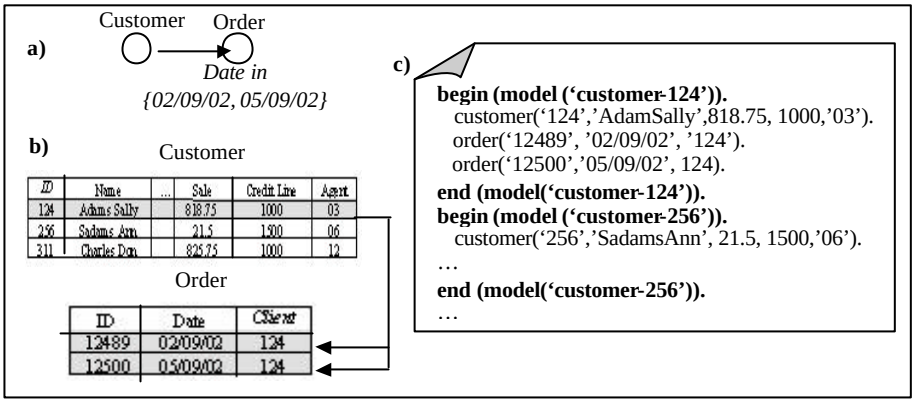
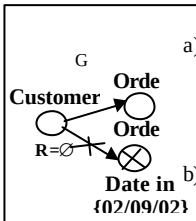


Fig. 3. (a) Example of selection graph; (b) corresponding grouping and (c) logic representation of objects selected from an instance of the example database

Selection graphs are more intuitive than expressions in SQL or Prolog, because they reflect the structure of the relational data model, and refinements of existing graphs may be defined in terms of addition or updating of arcs and/or nodes. The given definition of selection graph cannot allow to represent recursive relationships. Therefore a selection graph can be straightforwardly translated into either SQL or into first order logic expressions (Fig. 4). In this case a subgraph pointed by an absent arc is translated into a negated inner sub-query.



a)
```
SELECT n0.ID, n0.Name, n0.Adress,…, n0.Sale, n0.CreditLine, n0Agent,
       n1.ID, n1.Date, n1.Client
FROM Customer n0, Order n1
WHERE n0.ID=n1.Client and n0.ID not in
      (SELECT n2.Client FROM Order n2 WHERE n2.Date in {02/09/02}));
```

b)
```
← customer(N0_ID, …,N0_Agent),
   order(N1_ID,N1_Date, N0_ID),
  ¬ (order(N2_ID,N2_Date, N0_ID), N2_Date=02/09/02).
```

Fig. 4. (a) SQL and (b) first order logic translation of a selection graph G

5 Multi-relational Stepwise Model Tree Induction

Mr-SMOTI induces model trees whose nodes (regression, split or leaf) involve multi-relational patterns that can be represented with *selection graphs*, that is *each node of the tree corresponds to a selection graph*. Essentially Mr-SMOTI, like the propositional version SMOTI, builds a tree-structured multi-relational regression model by adding split and/or regression nodes through a process of successive refinements of the current selection graph until a stopping criterion is fulfilled and a leaf node is introduced. Thus, the model associated to each leaf is computed by combining all straight-line regressions in the regression refinements along the path from the root to the leaf.

5.1 The Algorithm

Mr-SMOTI is basically a *divide-and-conquer* algorithm that starts with a root selection graph G containing only the target node n_0. This graph corresponds to the entire set of objects of interest in the relational database D (the target table T_0). At each step the system chooses the optimal refinement (split or regression) according to a heuristic function. In particular, a split refinement corresponds to either the updating of an existing node by adding a new selection condition or the introduction of a new node and a new arc in the current selection graph. On the other hand, a regression refinement corresponds to the updating of regression terms in existing nodes. The optimal refinement (and its complement in the case of a split), are used to create the regression functions associated to the root of the left (/right) branch. This procedure is recursively applied to each branch until a stopping criterion is fulfilled.

Mr-SMOTI (D: database, G: selection_graph)
begin
G_S, G_R, R: selection_graph;
T_left, T_right: model_tree;
    G_R:= *optimal_regression_refinement* (G, D);
    if *stopping_criteria* (G_R, D) then return *leaf* (G_R);
    G_S:= *optimal_split_refinement* (G, D);
    R:= *best_refinement* (G_R, G_S);

```
    if(R=G_R)          //the optimal refinement is a regression node
        T_left := Mr-SMOTI (D,R);
                T_right := ∅;
    else               // the optimal refinement is a split node
        T_left := Mr-SMOTI (D,R);
        T_right := Mr-SMOTI (D, comp (R));
    return model_tree(R, T_left, T_right).
end
```

The functions *optimal_split_refinement* and *optimal_regression_refinement* take the current selection graph G corresponding to the current node t and consider every possible split and regression refinement. The choice of which refinements are candidates is determined by the current selection graph G, the structure of data model in D, and notably by the multiplicity of associations within this data model.

The validity of either a splitting refinement (G_S) together with its complement ($comp(G_S)$), or a regression refinement (G_R) is based on two distinct evaluation measures, $s(G_S, comp(G_S))$ and $r(G_R)$, respectively. Both $s(G_S, comp(G_S))$ and $r(G_R)$, are mean square errors (MSE) [2], therefore they can be actually compared to choose between three different possibilities:

- growing the model tree by adding the node t_{G_R} corresponding to the regression refinement G_R;
- growing the model tree by adding the nodes t_{GS} and $t_{Comp(GS)}$ corresponding to the splitting refinement G_S and its complement $comp(G_S)$[3];
- stopping the tree's growth at the current node t.

Let T be the multi-relational model tree currently built stepwise, G the selection graph associated to the node t in T and t_{G_S} ($t_{comp(G_S)}$) the left (right) child of t, associated to a split refinement G_S (the complementary split refinement $comp(G_S)$) of the selection graph G, $s(G_S, comp(G_S))$ is defined as:

$$s(G_S, comp(G_S)) = \frac{N(t_{G_S})}{N(t_{G_S}) + N(t_{comp(G_S)})} R(G_S) + \frac{N(t_{comp(G_S)})}{N(t_{G_S}) + N(t_{comp(G_S)})} R(comp(G_S)),$$

where $N(t_{G_S})$ ($N(t_{comp(G_S)})$) is the number of training tuples covered by the refinement G_S ($comp(G_S)$), and $R(G_S)$ ($R(comp(G_S))$) is the resubstitution error of the left (right) child, computed as follows:

$$R(G_S) = \sqrt{\frac{1}{N(t_{G_S})} \sum_{j=1}^{N(t_{G_S})} (y_j - \hat{y}_j)^2} \qquad R(comp(G_S)) = \sqrt{\frac{1}{N(t_{comp(G_S)})} \sum_{j=1}^{N(t_{comp(G_S)})} (y_j - \hat{y}_j)^2}.$$

Therefore the evaluation measure $s(G_S, comp(G_S))$ is coherently defined on the basis of the partially defined multiple linear regression models $\hat{Y}$ built by combining

[2] Mr-SMOTI minimises the square error with respect to the partially constructed regression model .

[3] Mr-SMOTI requires that the subsets of target objects belonging to patterns deriving from the same parent by applying some kind of refinement must be complementary. Because of this the split refinements are introduced together with their complementary refinement.

the best straight-line regression associated to $t_{G_S}(t_{comp(G_S)})$, with all regressions introduced along the path from the root to $t_{G_S}(t_{comp(G_S)})$.

The evaluation of a regression step $Y=a+bX_i$ at regression refinement G_R, cannot be naïvely based on the resubstitution error $R(G_R)$:

$$R(G_R) = \sqrt{\frac{1}{N(t_{G_R})} \sum_{j=1}^{N(t_{G_R})} (y_j - \hat{y}_j)^2},$$

where t_{G_R} is the node representing the regression refinement G_R and $N(t_{G_R})$ is the number of training tuples covered by the refinement G_R. The predicted value $\hat{y}_j$ is computed by combining all regression lines introduced in T along the path from the root to t_{G_R}. This would result in values of $r(G_R)$ less than or equal to values of $s(G_S,comp(G_S))$ for splitting refinement involving X_i [18]. Indeed, the splitting test "looks-ahead" to the best multiple linear regressions after the current split is performed, while the regression step does not perform such a look-ahead. A fairer comparison would be to grown the model tree at a further level in order to base the computation of $r(G_R)$ on the best split refinement G_S, after the current regression refinement is performed. Therefore, $r(G_R)$ is defined as follows:

$$r(G_R) = min\{R(G_R), s(G_S, comp(G_S))\}.$$

Having defined both $s(Gs, comp(Gs))$ and $r(G_R)$, the criterion for selecting the best refinement is fully characterised as well. At each step of the induction process, Mr-SMOTI chooses the apparently most promising refinement, according to a greedy strategy.

The function *stopping_criteria* determines whether the current optimal refinement must be transformed into a leaf according to the minimal number of *target objects* (*minObject*) covered by the selection graph which is associated to the current node and the minimal threshold for the *coefficient of determination* (*minR*) of the prediction function built stepwise [4]. This coefficient is a scale-free one-number summary of the strength of the relationship between independent variables in the actual multiple linear model and the dependent variable.

The regression model built by Mr-SMOTI can be viewed as a set of SQL queries associated with each leaf in the tree. These queries predict an estimate of the target attribute according to the multiple model built stepwise. The prediction is averaged by means of a grouping on the target objects. The complementary nature of different branches of a model tree ensures that a given target object cannot be assigned a conflicting model.

5.2 The Refinements

Split refinements are an extension of the refinement operations proposed in [11] to perform a split test in a multi-relational decision tree. Whenever a split is introduced in a model tree, Mr-SMOTI is in fact refining the selection graph associated to the current node, by adding either a condition or an open node linked by a present arc.

Given a selection graph G, the *add condition* refinement returns the refined selection graph G_s by simply adding a split condition to an open node $n_i \in G.N$ without changing the structure of G. The split condition can be a test on either a continuous or a discrete attribute of the table associated to the node n_i. The first is in the form $X_i £ a$. The value of a is one of the cut points found by an equal frequency discretization of the ordered distinct values of X_i. A discrete test is in the form X_i *in* U_i, with U_i a subset of the range of X_i. A greedy strategy as suggested by [20] is used to identify U_i. Initially $U_i = \emptyset$ is considered, the possible refinement is obtained by moving one discrete value from the range of X_i to U_i, such that the move results in a better split. The evaluation measure $s(G_S, comp(G_S))$ is computed, therefore a better split decreases $s(G_s, comp(G_S))$. The process is iterated until there is no improvement in the splits.

The *add linked node* refinement instantiates an association of the data model D by means of a *present arc*, together with its corresponding table, represented as an *open node*, and adds these to the selection graph G. Knowledge of the nature and multiplicity is used to guide and optimise this search. Since the investigated associations are *foreign key associations*, the proposed refinements can have two directions: *backward* or *forward*. The former correspond to *many-to-one* associations, while the latter describe *one-to-many* associations in the data model. This means that a backward refinement of the selection graph G does not partition the set of target objects covered by G but extends their descriptions (training data) by considering tuples joined in the table which are represented by the new added node.

Each split refinement of type *add condition* or *add linked node* is introduced together with its complementary refinement. Let G be the selection graph associated to the current node t and G_S a split refinement of G associated to the left sub-tree of t. The first order logic expression translating G_S is:

$$\neg\, Q_G, conj\,(Q_{G_S}),$$

where Q_G is the translation of G and *conj* is the condition corresponding to the split refinement. The complementary refinement ($comp(G_S)$) associated with the right sub-tree could not be the expression $\neg\, Q_G, \emptyset conj$. Indeed, the selection graphs (queries) of the left and right sub-tree must be complementary: for each object into the current node (Q_G succeeds) exactly one of both queries should succeed. Consider in Figure 5 the refinement G_S of the selection graph G, obtained by adding a condition on the table *Order* that is not a target table. In this case the complementary of adding a literal (Date in {02/09/02}) is not equivalent to adding its negation (¬(Date in {02/09/02})), while at the same time switching the branches of T. This is an important difference compared with the propositional case, where a test and its simple negation generate a partition of the training data. The complementary set associated to the complementary refinement of G_S must contain the target objects in G (and the linked information in connected nodes) that are associated with *none* of the tuples in *Order* satisfying the refinement condition.

In [11], Knobbe *et al.* propose a complementary refinement named *add negative condition* that should solve the problem of mutual exclusion between *an add condition refinement and its complement*. If the node that is being refined does not represent the target table, $comp(G_S)$ is built from G by introducing an absent arc from the parent of n_i to the clone of the entire sub-graph of G that is rooted in n_i. The introduced sub-graph has a root (a clone of the node to be refined) that is a closed

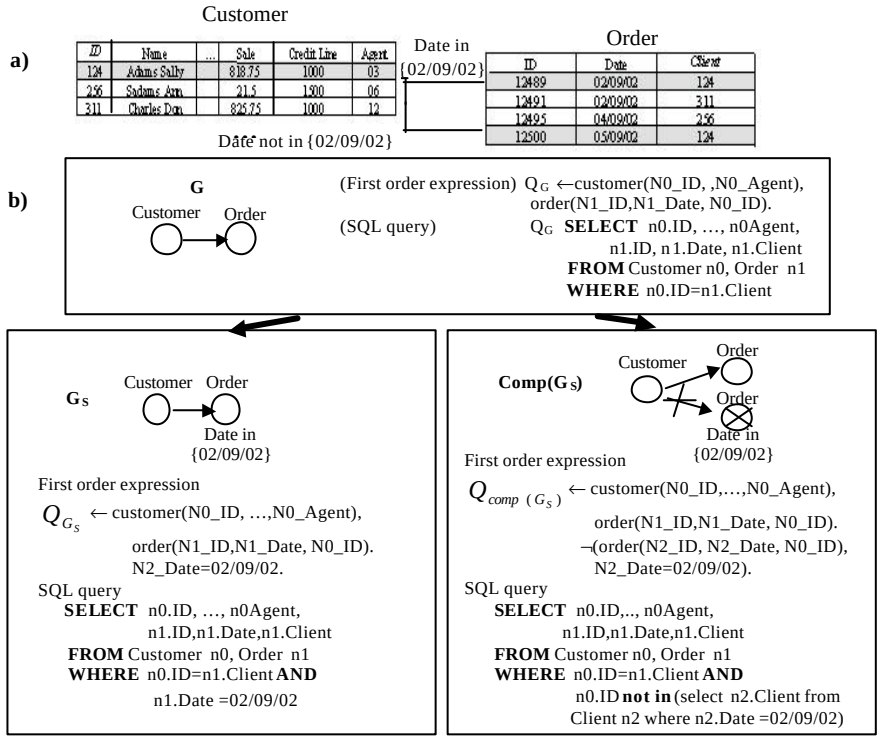


Fig. 5. Explanation of (a) the partitioning of training objects according to (b) a split refinement G_S and its complement *comp*(G_S)

node updated with the refinement condition that is not negated. In this way the complementary operation builds a selection graph that negates an entire inner sub-query and not simply a condition. As was observed in [16], this approach fails to build complementary refinements when the node to be refined is *not directly* connected to the target node.

The example in Figure 6 proves that the proposed mechanism could build a refinement G_S and a complementary refinement *comp*(G_S) that are not mutually exclusive. To overcome this problem the complementary refinement *comp*(G_S) should be obtained by adding an absent arc from the target node n_0 to the clone of the sub-graph containing the *entire join path* from n_0 to the node to be refined. The introduced sub-graph has a root (a clone of n_0) that is a *closed* node and is updated with the refinement condition that is not negated. A new absent arc is also introduced between the target node and its closed clone. This arc is an instance of the implicit relationship between the primary key of the target table and the own itself (Figure 7).

Similarly, when we consider the complementary refinement for an *add linked node* refinement we make the same considerations as when a negated condition is going to be added. This means that when the closed node to be added is not directly connected to the target node in *G*, a procedure similar to that described when an add condition refinement is complemented must be followed.

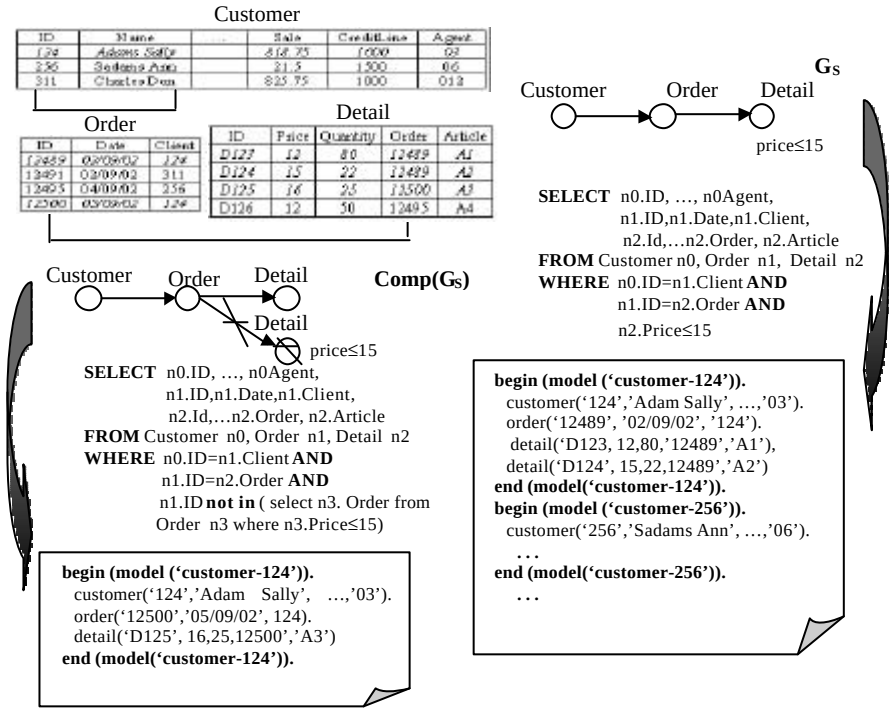
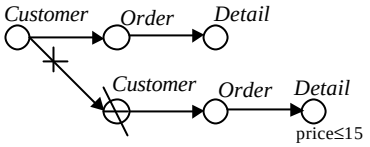


Fig. 6. Example of (a) refinement (G_S) by adding a condition on a node not directly connected to the target node and (b) the corresponding complementary refinement, proposed by *Knobbe et al.*, that does not satisfy the mutual exclusion




```
SELECT n0.ID, …, n0Agent,
       n1.ID,n1.Date,n1.Client,
       n2.Id,…n2.Order, n3.Article
FROM Customer n0, Order n1, , Detail n2
WHERE n0.ID=n1.Client AND
      n1.ID=n2.Order AND
      n0.ID not in ( select n0. ID from
        Customer n3, Order n4, , Detail n5
        where n3.ID=n4.Client and
        n4.ID=n5.Order and n5.Price≤15)
```

Fig. 7. Example of correct complementary refinement when adding a condition on a node not directly connected to the target node

Finally, a *regression refinement* G_R (Figure 8) corresponds to performing a regression step ($Y' = a_Y + b_Y \cdot n_i.T.X_j'$) on the residual of a continuous attribute ($n_i.T.X_j'$) not yet introduced in the model currently built. Regression coefficients (a_Y and b_Y) are estimated according to the values of Y' and $n_i.T.X_j'$ of each single tuple selected by the current selection graph G. The regression attribute must belong to a table represented by a node in the parent graph G. For each node, the list of regressions R is updated by adding the regression term ($n_i.T.X_j'$) introduced in the model and the coefficients a and b computed to update the residuals of all continuous attributes in

the node. According to the evaluation function $r(G_R)$, a regression refinement includes a look-ahead capability. The complementary refinement of a regression step is empty.

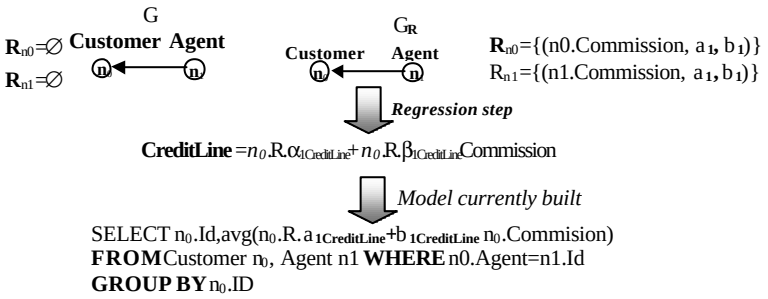


Fig. 8. Example of a regression refinement G_R that performs a regression step on the attribute *Agent.Commission*. The nodes n_0 and n_1 are updated with the vectors of coefficients a and b in order to remove the effect of the regression attribute from the continuous attributes in $n_0.T$ and $n_1.T$, respectively

6 Experimental Evaluation

Mr-SMOTI has been applied to the biological problems of predicting both the mutagenic activity of molecules [21] and the biodegradability of chemical compounds in water [6]. A *mutagenesis* dataset consists of 230 molecules divided into two subsets: 188 molecules for which linear regression yields good results and 42 molecules that are regression-unfriendly. In our experiments we used the atom and bond structure of regression-friendly molecules by adding boolean indicators *Ind1* and *Ind2* as one setting (B_1) and adding *Lumo* and *Logp* properties to get a second setting (B_2). Similarly *biodegradability* dataset consists of 328 chemical molecules structurally described in terms of atom and bond. In all the experimental results reported below the thresholds for stopping criteria are fixed as follows: the minimum number of target objects falling in each internal node must be greater than the square root of the number of target objects in the entire training set and the determination coefficient in each internal node must be below 0.80.

Each dataset is analysed by means of a 10-*fold* cross-validation. Figure 9 shows the test set performance of *Mr-SMOTI* and *TILDE-RT* in both domains, as measured by the *Pearson correlation coefficient.* The Pearson correlation coefficient (*PCC*), which is computed as follows:

$$PCC = \frac{\sum_{j=1..N}(y_j - \bar{y}_j)(\hat{y}_j - \bar{\hat{y}}_j)}{\sqrt{\sum_{j=1..N}(y_j - \bar{y}_j)^2} \times \sqrt{\sum_{j=1..N}(\hat{y}_j - \bar{\hat{y}}_j)^2}},$$

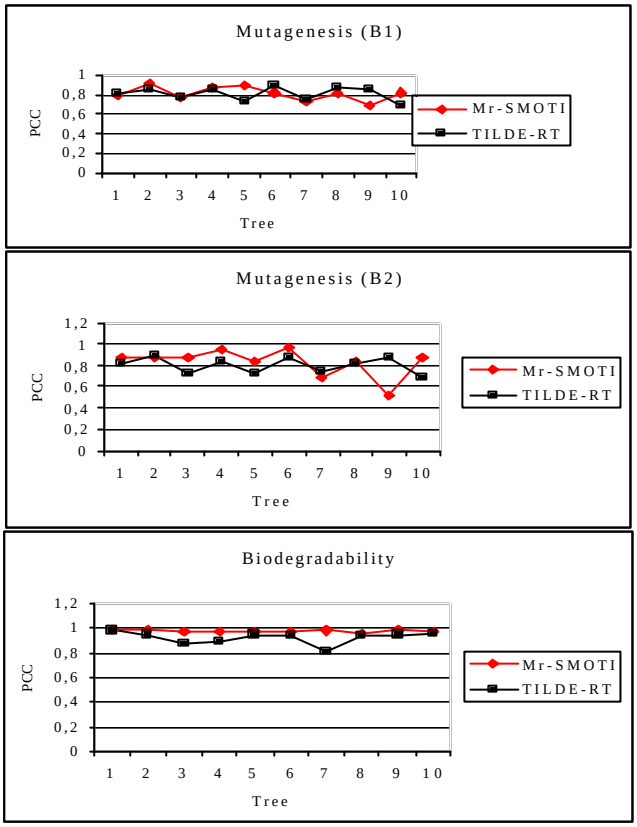


Fig. 9. Pearson correlation coefficient (Y axis) for multi-relational prediction models induced from the 10-fold cross validated datasets (X axis) of Mutagenesis (B1, B2) and Biodegradability datasets. The comparison concerns two systems: TILDE-RT (black squares) vs. Mr-SMOTI (purple diamonds)

is a measure of how much the value of a target attribute (y_j) in test objects correlates with the value ($\hat{y}_j$) predicted by the induced model.

Since the Pearson correlation coefficient does not measure the quantity error of a prediction, we include several other measures as proposed by Quinlan [24]. We have evaluated the predictive accuracy on the basis of the *average mean square error* (*Avg.MSE*), which is computed as follows:

$$Avg.MSE = \frac{1}{k}\sum_{V_i \in V} MSE(V_i) = \frac{1}{k}\sum_{V_i \in V}\sqrt{\frac{1}{N(V_i)}\sum_{j \in V_i}(y_{ij} - \hat{y}_{ij}(V - V_i))^2},$$

where $V=\{V_1, .., V_k\}$ is a k-cross-validation partition of the training data V (i.e., 10), $N(V_i)$ is the number of target objects in V_i, and $\hat{y}_j(V - V_i)$ is the value predicted for the j-th target object in V_i by the prediction model built from V-V_i .

The predictive accuracy is also estimated according to the *average error* (*AE*) averaged on a 10 fold cross-validation:

$$AvgAE = \frac{1}{k}\sum_{V_i \in V} AE(V_i) = \frac{1}{k}\sum_{V_i \in V}\sqrt{\frac{1}{N(V_i)}\sum_{j \in V_i} | y_{ij} - \hat{y}_{ij}(V - V_i) |}.$$

For pairwise comparison with *TILDE-RT* the non-parametric Wilcoxon two-sample paired signed rank test is used [22], since the number of folds (or "independent" trials) is relatively low and does not justify the application of parametric tests, such as the t-test. To perform the test, we assume that the experimental results of the two compared methods are independent pairs of sample data $\{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$. We then rank the absolute value of the differences $u_i - v_i$. The Wilcoxon test statistics W^+ and W^- are the sum of the ranks from the positive and negative differences, respectively. We test the null hypothesis H_0: "no difference in distributions" against the two-sided alternative H_a: "there is a difference in distributions". More formally, the hypotheses are: H_0: "$\mu_u = \mu_v$" against H_a: "$\mu_u \neq \mu_v$". Intuitively, when $W^+ >> W^-$ and viceversa, H_0 is rejected. Whether W^+ should be considered "much greater than" W^- depends on the significance level α. The basic assumption of the statistical test is that the two populations have the same continuous distribution (and no ties occur). Since, in our experiments, u_i and v_i are MSE, $W^+ >> W^-$ implies that the second method (V) is better than the first one (U).

The results of the Wilcoxon signed rank test on the accuracy of the induced multi-relational prediction model are reported in Table 1. The Wilcoxon test statistics W^+ (W^-) is the sum of the ranks from the positive (negative) differences between TILDE-RT and Mr-SMOTI. Therefore, the smaller W^+ (W^-), the better for Mr-SMOTI (TILDE-RT). Differences are considered statistically significant when the p-value is less than or equal to $\alpha/2$.

Table 1. Results of the Wilcoxon signed rank test on the accuracy of the induced models. The best value is in boldface, while the statistically significant values ($p \leq \alpha/2, \alpha=0.05$) are in italics

Dataset		*Accuracy*	*Mr-SMOTI*	*TILDE-RT*	*W+*	*W-*	*P*
Mutagenesis	*B1*	*Avg.MSE*	**1.165**	1.197	23	32	0.69
		Avg.AE	**0.887**	0.986	12	43	0.13
	B2	*Avg.MSE*	**1.118**	1.193	15	40	0.23
		Avg.AE	**0.845**	0.985	11	44	0.10
Biodegradability		*Avg.MSE*	***0.337***	0.588	0	55	*0.0019*
		Avg.AE	***0.186***	0.363	0	55	*0.0019*

Table 2. Number of leaves comparison for the 188 regression friendly elements of Mutagenesis (B1 and B2 setting) and the 328 elements of Biodegradability

System	*Mutagenesis – B1*	*Mutagenesis – B2*	*Biodegradability*
Mr-SMOTI	14.4	**9.2**	**2**
TILDE-RT	**11.7**	14.9	4.7

Experimental results on tree size are reported in Table 2. Results show that in the case of both mutagenesis dataset (B2 setting) and biodegradability dataset Mr-SMOTI builds simpler models (in number of leaves) without loosing accuracy.

7 Conclusions

This paper presents a novel approach to mining relational model trees. The proposed algorithm can work effectively when training data are stored in multiple tables of a relational DBMS. Information on the database schema is used to reduce the search space of patterns. Induced relational models are represented by selection graphs whose definition has been extended in order to describe model trees with either split nodes or regression nodes. The proposed algorithm has been implemented as a module of the system MURENA that is tightly coupled to the Oracle Database. As future work, we plan to extend the comparison of Mr-SMOTI to other multi-relational data mining systems on a larger set of benchmark datasets. In particular, we plan to apply Mr-SMOTI in the spatial task of supporting quantitative interpretation of maps and in the analysis of geo-referenced census data [1]. Moreover, we intend to use SQL primitives and parallel database servers to speed up the stepwise construction of multi-relational model trees from data stored in a large database.

Acknowledgments

This work has been supported by the annual Scientific Research Project "Metodi di apprendimento automatico e di data mining per sistemi di conoscenza basati sulla semantica" Year 2003, funded by the University of Bari. The authors thank Hendrik Blockeel for providing mutagenesis and biodegradability datasets.

References

[1] Appice A., Ceci M., Lanza A., Lisi F.A., and Malerba D.: *Discovery of Spatial Association Rules in Georeferenced Census Data: A Relational Mining Approach*, Intelligent Data Analysis, numero speciale su "Mining Official Data" (in press).

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

[3] Breiman L., Friedman J., Olshen R., and Stone J.: *Classification and regression tree*, Wadsworth & Brooks, 1984.

[4] Draper N.R., and Smith H.: *Applied regression analysis*, John Wiley & Sons, 1982.

[5] Dzeroski S.: *Numerical Constraints and Learnability in Inductive Logic Programming*. Ph.D thesis, University of Ljubljana, Slovenia, 1995.

[6] Dzeroski S., Blockeel H., Kramer S., Kompare B., Pfahringer B., and Van Laer W.: Experiments in predicting biodegradability. *Proceedings of the Ninth International Workshop on Inductive Logic Programming* (S. Dzeroski and P. Flach, eds.), LNAI, vol. 1634, Springer, pp. 80-91, 1999.

[7] Dzeroski S., Todoroski L., and Urbancic T: Handling real numbers in inductive logic programming: A step towards better behavioural clones. In *Machine Learning: ECML-95*, Eds. Lavrac N., and Wrobel S., Springer , Berlin Heidelberg New York, 1995.

[8] Dzeroski S., and Lavrac N. (Eds). *Relational Data Mining*. Springer-Verlag, 2001.

[9] Karalic A.: Linear regression in regression tree leaves. In *Proc. of ISSEK'92 (International School for Synthesis of Expert Knowledge)*, Bled, Slovenia, 1992.

[10] Karalic A.: First Order regression. Ph.D thesis, University of Ljubljana, Slovenia, 1995.
[11] Knobbe, J., Siebes, A., and Van der Wallen, D.M.G.: Multi-relational decision tree induction. In *Proc. 3rd European Conf. on Principles and Practice of Knowledge Discovery in Databases*, PKDD '99, 1999.
[12] Knobbe J., Blockeel H., Siebes, A., and Van der Wallen D.M.G.: Multi-relational Data Mining. In *Proc. of Benelearn'99*, 1999.
[13] Knobbe A.J., Haas M., and Siebes A: Propositionalisation and aggregates. In *Proc. 5th European Conf. on Principles of Data Mining and Knowledge Discovery*, Springer-Verlag, 2001.
[14] Kramer S.: Structural regression trees. In *Proc. 13th National Conf. on Artificial Intelligence*, 1996.
[15] Lavrac N., and Dzeroski S.: *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, Chichester, UK, 1994.
[16] Leiva H.A.: MRDTL: *A multi-relational decision tree learning algorithm*. Master thesis, University of Iowa, USA, 2002.
[17] Lubinsky D.: Tree Structured Interpretable Regression. In *Learning from Data*, Fisher D., and Lenz H.J. (Eds.), Lecture Notes in Statistics, 112, Springer, 1994.
[18] Malerba D., Appice A., Ceci M., and Monopoli M.: Trading-off versus global effects or regression nodes in model trees. In *Foundations of Intelligent Systems, 13th International Symposium, ISMIS 2002*, Hacid H.S., Ras Z.W. , Zighed D.A., and Kodratoff Y. (Eds.), Lecture Notes in Artificial Intelligence, 2366, Springer, Germany, 2002.
[19] Malerba D., Esposito F., Ceci M., and Appice A.: Top-down induction of model trees with regression and splitting nodes. LACAM Technical Report, 2003.
[20] Mehta M., Agrawal R., and Rissanen J.: SLIQ: A fast scalable classifier for data mining. In *Proceedings of the Fifth International Conference on Extending Database Technology*, 1996.
[21] Muggleton S., Srinivasan A., King R., and Sternberg M.: Biochemical knowledge discovery using Inductive Logic Programming. In *Proceedings of the first Conference on Discovery Science*, Motoda H. (ed), Springer-Verlag, Berlin, 1998.
[22] Orkin. M., and Drogin. R.: *Vital Statistics*. McGraw Hill. New York, 1990.
[23] Quinlan J. R.: Learning with continuous classes, in *Proceedings AI'92*, Adams & Sterling (Eds.), World Scientific, 1992.
[24] Quinlan J.R.: A case study in Machine Leaning, in *Proceedings ACSC-16*, Sixteenth Australian Computer Science Conferences, 1993.
[25] Silverstein, G., and Pazzani, M.J.: Relational cliches: Constraining constructive induction during relational learning. In *Proc. 8th Int. Workshop on Machine Learning*, 1991.
[26] Torgo L.: Functional Models for Regression Tree Leaves. In *Proceedings of the 14th International Conference (ICML'97)*, D. Fisher (Ed.), Nashville, Tennessee, 1997.
[27] Wang Y., and Witten I.H.: Inducing Model Trees for Continuous Classes. In *Poster Papers of the 9th European Conf. on Machine Learning (ECML'97)*, M. van Someren and G. Widmer (Eds.), Prague, Czech Republic, 1997.
[28] Weiss, S.M., and Indurkhya, N.: *Predictive Data Mining. A Practical Guide.* Morgan Kaufmann, San Francisco:CA, 1998.
[29] Wrobel, S.: Inductive logic programming for knowledge discovery in databases. In Dzeroski S., and Lavrac N. (Eds). *Relational Data Mining*. Springer-Verlag, pp. 74-101, 2001.