Knowledge-Based Systems 251 (2022) 109254

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Relational tree ensembles and feature rankings

Matej Petković [a,b,e], Michelangelo Ceci [c,a,f,*], Gianvito Pio [c,f], Blaž Škrlj [a,b], Kristian Kersting [d], Sašo Džeroski [a,b]

[a] Jozef Stefan Institute, Jamova 39, Ljubljana, Slovenia
[b] Jozef Stefan International Postgraduate School, Jamova 39, Ljubljana, Slovenia
[c] Università degli Studi di Bari Aldo Moro, via E. Orabona 4, Bari, Italy
[d] TU Darmstadt, Hochschulstrasse 1, Darmstadt, Germany
[e] Faculty of Mathematics and Physics, Jadranska 21, Ljubljana, Slovenia
[f] Big Data Lab. National Interuniversity Consortium for Informatics, CINI, Via Ariosto 25, Rome, Italy

ARTICLE INFO

Article history:
Received 11 November 2021
Received in revised form 8 May 2022
Accepted 9 June 2022
Available online 18 June 2022

Keywords:
Relational learning
Tree ensembles
Feature ranking
Propositionalization

ABSTRACT

As the complexity of data increases, so does the importance of powerful representations, such as relational and logical representations, as well as the need for machine learning methods that can learn predictive models in such representations. A characteristic of these representations is that they give rise to a huge number of features to be considered, thus drastically increasing the difficulty of learning in terms of computational complexity and the curse of dimensionality. Despite this, methods for ranking features in this context, i.e., estimating their importance are practically non-existent.

Among the most well-known methods for feature ranking are those based on ensembles, and in particular tree ensembles. To develop methods for feature ranking in a relational context, we adopt the relational tree ensemble approach. We thus first develop methods for learning ensembles of relational trees, extending a wide spectrum of tree-based ensemble methods from the propositional to the relational context, resulting in methods for bagging and random forests of relational trees, as well as gradient boosted ensembles thereof. Complex relational features are considered in our ensembles: by using complex aggregates, we extend the standard collection of features that correspond to existential queries, such as 'Does this person have any children?', to more complex features that correspond to aggregation queries, such as 'What is the average age of this person's children?'. We also calculate feature importance scores and rankings from the different kinds of relational tree ensembles learned, with different kinds of relational features. The rankings provide insight into and explain the ensemble models, which would be otherwise difficult to understand.

We compare the methods for learning single trees and different tree ensembles, using only existential qualifiers and using the whole set of relational features, against 10 state-of-the-art methods on a collection of benchmark relational datasets, deriving also the corresponding feature rankings. Overall, the bagging ensembles perform the best, with gradient boosted ensembles following closely. The use of aggregates is beneficial and in some datasets drastically improves performance: In these cases, aggregate-based features clearly stand out in the feature rankings derived from the ensembles.

1. Introduction

Nowadays, data are increasingly complex, and the standard tabular representation, where each row of a table corresponds to an object and each column corresponds to a characteristic, does not suffice anymore. Indeed, in real-world scenarios, we can easily find objects of multiple types, possibly described by different sets of characteristics, that are interconnected to each other through multiple types of relationships. Finance, biology, epidemiology and geography are just a few of the application domains where such a scenario is very common. In biology, for instance, protein networks [1] encode complex data about proteins and their functions, together with different protein–protein interactions. Additional examples are reported in Section 5.1.

The application of standard machine learning methods to such data is not straightforward, due to the additional complexity introduced by the need to "navigate" the relationships. However, discarding the additional information about connected objects

* Corresponding author at: Università degli Studi di Bari Aldo Moro, via E. Orabona 4, Bari, Italy.

E-mail addresses: matej.petkovic@ijs.si (M. Petković), michelangelo.ceci@uniba.it (M. Ceci), gianvito.pio@uniba.it (G. Pio), blaz.skrlj@ijs.si (B. Škrlj), kersting@cs.tu-darmstadt.de (K. Kersting), saso.dzeroski@ijs.si (S. Džeroski).

https://doi.org/10.1016/j.knosys.2022.109254
0950-7051/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

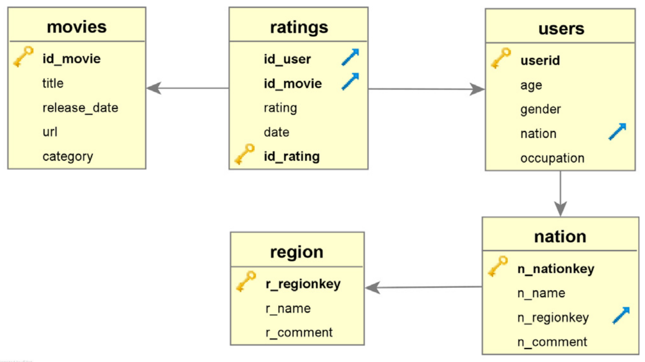


Fig. 1. The relational schema of the Movies dataset, consisting of 5 tables. Fields shown in bold represent primary keys (with a key symbol) and foreign keys (with an arrow). The task at hand is to predict the property Category (genre) in the target table Movies.

may lead to learning weak models. For example, if we are interested in learning a classification model for the genre of a movie, the characteristics of movies themselves, e.g., titles and release dates, might not be enough to discriminate among different genres. If we include additional data about the ratings (with their characteristics, e.g., date and value), as well as about users who provided such ratings (with their characteristics, e.g., age, gender and job), the model may have a better chance of accurately predicting the genre since, for example, comedies attract a different kind of viewers as compared to dramas.

In this scenario, data can naturally be represented through a relational database, where data are stored in multiple tables, interconnected via foreign keys that implicitly represent and define the relationships among objects and their characteristics. For example, the dataset *Movies* (see Section 5.1) can be represented through the relational schema depicted in Fig. 1, where the table storing movies (the target of the learning task) is linked to the table *Ratings* from which it is possible to specify which *Users* provided such ratings as well as their characteristics, spread over the table *Users* itself and other tables representing their geographical location (*Nations* and *Regions*).

A possible solution to analyze such data consists in the adoption of *propositionalization* approaches. These approaches are based on the transformation of relational data into a propositional (or attribute-value or feature-based) form, by constructing additional features that represent the relationships and the properties of data related to target objects. This approach decouples the construction of features from the phase of learning the predictive model [2], in order to apply standard classification methods to the transformed data [3]. Solutions based on propositionalization mainly resort to two different approaches, namely, database-oriented [4] or logic-oriented techniques [5]. The former work by materializing joins according to foreign keys and allowing for fast aggregations. The latter require data represented as Prolog facts and are able to consider complex background knowledge, enabling logic-oriented propositionalization to provide expressive first-order models by constructing first-order logic features, at the cost of possible lower efficiency.

In contrast to propositionalization approaches, *structural* approaches take into account the structure of original data and directly navigate the tables following the foreign keys. Therefore, the whole hypothesis space is directly explored during the learning process (a more comprehensive overview is given in Section 2).

Following this line of research, in this paper, we propose a novel method, called RE3PY,[1] to solve the classification task in the relational setting, focusing on the prediction of the value of a categorical attribute (of previously unseen examples) of a target table. RE3PY is based on the top-down induction of decision trees [6], properly extended to the relational setting. Split candidates are based on conditions on *paths* involving multiple tables (following the foreign keys) and aggregates of attributes. Following the example in Fig. 1, a path can e.g., convey information on: (1) the rating of a movie; (2) the author of the rating; (3) the age of the author, that can be aggregated through the average, minimum, maximum aggregation functions, back to the movie. In this case, we obtain new aggregated features associated to the movie, representing the average, the minimum and the maximum age, respectively, of the authors of the ratings of a movie. Similarly, we can generate new features representing the nationality of the majority of the authors of ratings associated to a given movie.

In RE3PY, by iterating over the possible combinations of relations and aggregates, we automatically construct multiple new features for the examples of the target table during the learning phase and. Once the most promising features have been selected, others are discarded, making the process much more space-efficient than approaches based on propositionalization [7]. Moreover, RE3PY implements different ensemble approaches such as Bagging, Random Forests and Gradient Boosting, that have proven to provide a significant contribution in terms of prediction accuracy [8–10].

An additional advantage of our method is the possibility to explain the returned predictions. This is in line with recent advances in machine learning where the interpretability of the models and the explainability of the predictions are gaining increasing attention, especially in application domains such as medicine [11–13]. Similarly to classical classification and regression trees, the models learned by RE3PY are inherently interpretable. However, the use of ensemble techniques compromises the explainability of the predictions. Our method addresses this aspect by providing information about the contribution provided by each feature in the learned model. This is achieved by extending ensemble-based feature ranking approaches [9,14] to the relational setting. Feature ranking is a task, tightly coupled with predictive modeling, which has the goal of determining to what extent a given descriptive attribute (feature) influences the prediction of a target class/attribute. In the standard tabular setting, approaches for feature ranking have been widely investigated [15], whereas to the best of our knowledge, ours is the first approach being able to perform feature ranking in the relational setting.

The remainder of the paper is organized as follows: Section 2 briefly describes work related to this paper. In Section 3 we provide the problem statement, while in Section 4 we describe in detail the proposed method. In Section 5, we describe the experimental setup, whereas in Section 6 we present and discuss the results of our experimental evaluation. Finally, Section 7 concludes the paper and outlines future research directions.

2. Related work

The method proposed in this paper has its roots in the fields of relational learning and heterogeneous network classification (by means of collective, active and transductive inference, predictive clustering trees and network embedding techniques). In the following, we briefly present existing work related to these research fields.

[1] The implementation is available at https://pypi.org/project/re3py/.

2.1. Relational classification

In the literature, we can find several historical methods, mainly based on Inductive Logic Programming (ILP), that can solve the classification task in the relational setting. Noteworthy approaches include FOIL [16], that learns function-free Horn clauses (a subset of first-order predicate calculus); PROGOL [17], that combines inverse entailment with general-to-specific search through a refinement graph that guarantees a solution with the maximum compression in the search space; and TILDE [18], that learns a predicate logic theory by means of a so-called logical decision tree, a first-order logic upgrade of the classical decision tree. Such historical approaches, although able to produce highly-interpretable and possibly accurate models, have turned out to be very inefficient in real-world scenarios. Ensemble approaches, which produce less interpretable and more accurate models, have also been introduced in the relational classification setting. Quinlan [19] introduced boosting in the relational learning system FOIL. Van Assche et al. [20] introduced relational random forests, based on TILDE, which can use aggregates on the tests. More recently, the method BoostSRL [21] has been proposed for the induction of gradient boosting ensembles of relational trees. However, in contrast to TILDE, it does not consider aggregates, which makes tree (ensemble) induction faster at the price of a lower quality of the models. Moreover, neither TILDE nor BoostSRL, which can be considered the closest approaches to the method proposed in this paper, offer the possibility to perform feature ranking. Additional differences, mainly related to the language bias, that originate from the strategy adopted to construct the splits in the internal nodes of the trees, are discussed in Section 4, where our method is presented.

A probabilistic approach is followed by 1BC [22] and Mr-SBC [23,24], that are based on the naïve Bayes classification method and use first-order classification rules for the computation of the posterior probability for each class. In addition, during the generation of classification rules, Mr-SBC exploits: *(i)* logical factorization to simplify the computation of the posterior probabilities [25]; *(ii)* both discrete and continuous attributes, that are discretized through a supervised approach; *(iii)* the database schema.

Two relevant extensions of Mr-SBC have been proposed [26], namely ST-MrSBC (Self-Training MrSBC), an ensemble-based approach that can capture autocorrelation phenomena by resorting to a self-training method, and MT-MrSBC (Multi-Type MrSBC) that also predicts possible missing values and finds dependencies among the models built for different relations.

The tool RelWEKA [27] extends the well-known WEKA toolkit to work in the relational setting. In particular, it exploits specific relational distance measures (e.g., the Relational Instance-Based Learning (RIBL) measure [28]), and kernels (e.g., the Minkowski RIBL set distance [29]) to implement relational variants of classical algorithms, such as k-NN and Support Vector Machines.

Recently, the approach LazyBum has been proposed [30], which, although considered a propositionalization-based approach for learning decision trees with aggregates, works in a dynamic fashion and builds aggregate features during tree construction. However, differently from our approach, it uses only one aggregate per feature, does not follow an ensemble learning strategy and does not support feature ranking. Further details on this last aspect will be discussed later.

2.2. Collective, active and transductive inference

More recent approaches model relational data as information networks. In this setting, tuples in a relational database correspond to nodes in a network, while foreign keys correspond to edges [7]. Most existing works consider the within-network (or transductive) setting, that is, they model a partially labeled network and estimate the label for unlabeled nodes. Moreover, initial works mainly focused on homogeneous networks, where all the nodes are of the same type, related by one type of relationship. Approaches in this area are mainly based on collective inference [31–34], active inference [35], semi-supervised and transductive inference [36–39]. One relevant example in the transductive classification setting is GNetMine [40], a graph-based regularization approach, that models the link structure in arbitrary information networks with an arbitrary number of types of objects and links. GNetMine analyzes each sub-network associated with each type of link and aims at preserving its consistency. Another relevant example is RankClass [41], that combines ranking and classification tasks, by assuming that highly-ranked objects within a class should play more important roles in classification or, vice-versa, that class labels can be relevant to build a good ranking. RankClass iteratively builds a graph-based ranking model and, on the basis of the current ranking, adjusts the graph structure, in order to strengthen the weights of links in the subnetwork corresponding to each specific class, and to weaken links in the rest of the network.

Recently, a collective classification approach [42] has been proposed which aims at classifying objects of the same type in a heterogeneous network, on the basis of the analysis of meta-paths [43]. Meta-paths are sequences of link types connecting two objects to be classified, and are used to effectively assign labels to groups of interconnected instances. Classification is performed in a probabilistic fashion, on the basis of feature values of the objects under analysis, on "relational features" associated with meta-paths, and on the labels associated with objects appearing in meta-paths. For further insights on this stream of research we refer the reader to the work by Dong et al. [44].

2.3. Predictive clustering for network data

Some recent works have also proposed the adoption of the predictive clustering framework in the context of network data. However, most of them are able to work only with homogeneous networks and, therefore, cannot analyze the possible heterogeneity of data and relationships that is typical of the relational setting. For example, a recent approach by Steinhaeuser et al. [45] combines a descriptive task (clustering) with a predictive task, and uses the network to represent data in a unified framework for identifying and characterizing patterns in climate data. The network is actually built a-posteriori according to the Pearson's correlation coefficient, measured between pairs of nodes on the time series collected for the same variable. A clustering algorithm then groups interconnected nodes, aiming to minimize the pairwise walking distance between nodes in the same cluster. Finally, the prediction model is learned locally for each cluster. This method tackles a regression task and, as clarified before, is not able to work with heterogeneous networks.

Another approach [46] learns predictive clustering trees to classify nodes in a network. Although the trees are learned with a classical top-down procedure, the adopted heuristics exploit an autocorrelation-based measure, thus explicitly taking autocorrelation into account. The same principle is adopted in network predictive clustering trees [47], which are used to predict gene functions. The difference is that, in this last work, a hierarchical multi-label classification scheme is considered. However, both methods are not able to work with heterogeneous networks and, therefore, cannot analyze real-world scenarios, that include different types of entities, related through multiple types of relationships.

The first attempt that really exploits a predictive clustering approach for classification tasks in heterogeneous networks (and, equivalently, in the relational setting) is HENPC [48]. HENPC is able to solve multi-type classification tasks on heterogeneous networks. Methodologically, HENPC extracts possibly overlapping and hierarchically-organized heterogeneous clusters and exploits them to classify unlabeled nodes according to labeled nodes falling in the same clusters. Unlike the approach proposed in the present paper, HENPC is not based on the induction of trees, but rather on a bottom-up approach for the identification of heterogeneous clusters, that generally leads to non-optimal computational efficiency.

2.4. Embedding techniques

An alternative strategy to solve classification tasks in heterogeneous networks is based on the adoption of embedding techniques. Specifically, heterogeneous network/graph embedding (see [49] for a review) aims to learn a representation in a numerical feature space for nodes and/or links, subsequently used by classical machine learning methods to address node classification, clustering and link prediction tasks. In this case, the challenge is to learn a feature space that is able to represent in a propositional way multiple types of nodes and links of the heterogeneous network, as well as their attributes, when present. The purpose is to preserve the similarity in terms of network structure and attribute values. A relevant example in this context is the system metapath2vec [50], that identifies a set of neighborhood nodes of each node, according to random walks that follow *pre-defined* meta-paths. metapath2vec exploits the identified neighbors through a heterogeneous skip-gram model to embed nodes into a numerical feature space. Although metapath2vec enables the application of multiple classification approaches, it exhibits two main limitations: *(i)* it considers the type of nodes and links, and the structure of the network, but it is not able to naturally model attribute values for nodes and links, which instead need to be explicitly represented as nodes (this transformation is better known as *reification* [51] in Semantic Web), in order to be taken into consideration by random walks; *(ii)* the meta-paths to consider in the analysis must be manually specified a-priori.

Starting from metapath2vec, other similar approaches have been proposed in the literature. Among them, we can mention HeteSpaceyWalk [52] that is based on heterogeneous spacey random walks to aggregate different meta-paths. It mainly exploits a second-order hyper-matrix to control the probability of transitioning between nodes of different types. Following the same line of research, the authors of the method JUST [53] proposed a flexible approach that, for each step of the random walk, decides between changing and keeping the current type of nodes, without imposing a specific meta-path to follow. Also in this case, attribute values are not naturally represented.

Finally, some recent research attempts have been devoted to the adoption of Graph Convolutional Networks (GCNs) to model relational data and solve node classification tasks. Together with the pivotal work by Schlichtkrull et al. [54], it is worth mentioning the method AHN2Vec [55], an embedding method that is able to represent nodes in a numerical feature space, starting from a heterogeneous attributed network. The method first embeds node attribute values using a graph convolutional neural network, taking into account possibly different attribute spaces of different types of nodes. Then, it exploits meta-path-based random walks to identify nodes in the neighborhood, and feeds a skipgram model with their embedding. Therefore, also in this case, it is necessary to manually specify the meta-paths to consider.

All the embedding techniques for heterogeneous networks are mainly based on neural network architectures, with the result that they do not naturally provide any explanation of their output. Note that also our approach can be considered an embedding technique, since it builds features starting form the network structure and attribute values (see the work by Lavrač et al. [7] for a discussion on the analogies between propositionalization and embedding). The advantage of our approach is that *(i)* it preserves the explainability of the output since the generated features are still interpretable and *(ii)* it does not rely heavily on random walks, that inevitably introduce non-determinism.

In conclusion, multiple attempts have been made to solve the classification task in the relational setting, but existing approaches are affected by one or more limitations. Namely, they: *(i)* are only able to work with a restricted number of object types (often one type), *(ii)* do not properly exploit the information coming from related objects (discarding relevant information), *(iii)* suffer from a computational viewpoint, *(iv)* do not exploit the potential of the ensemble learning setting, *(v)* do not appear highly accurate, *(vi)* do not naturally exploit the attributes associated to nodes, or *(vii)* or *(vi)* are not able to provide explanations for the predictions made. In this respect, the work presented in this paper can be considered the first attempt to achieve all of the above characteristics simultaneously.

3. Problem statement

Before presenting our method, we introduce some notation and definitions, which are necessary to formally define the problem we address.

Given:

- A training database D composed of a set of n relational tables $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$;
- A set of primary key constraints on tables in $\mathcal{T}$;
- A set of foreign key constraints on tables in $\mathcal{T}$;
- A target table $T_t \in \mathcal{T}$;
- A discrete target attribute y belonging to T_t, different from the primary key of T_t, with values in $\mathcal{Y}_t$.

Find: a classification model $\psi_t : T_t \rightarrow \mathcal{Y}_t$, which is able to classify all the unlabeled instances (i.e., tuples) of the target table T_t in a database D', which has the same schema as D.

Note that the learning setting we consider is not transductive, i.e., the applicability of the learned classification model is not limited to the unlabeled instances known at training time, but can be applied to any database D', with the same schema as D, for which the values of the target attribute y are unknown.

Example 1. Let us consider the database schema in Fig. 1, related to the `movies` database. The target table T_t is the table `Movies`, the target attribute y is the attribute `Category` and the foreign keys are represented by the relationships among the tables.

The learner should exploit all the information from D, including the non-target tables. To better clarify the way non-target tables are used, we introduce the definition of *task-relevant* objects and *reference* objects. Intuitively, a reference object *ro* corresponds to a tuple of the target table T_t, while a task-relevant object *tro* corresponds to a tuple of a non-target table, which is connected to *ro* by means of a foreign key path.

We formally introduce the concepts of *foreign key path* and *task-relevant objects* by means of the following definitions:

Definition 1. A foreign key path is defined as an ordered sequence of tables $p_i = \langle T_{i_1}, T_{i_2} \ldots, T_{i_s} \rangle$, , where: $\forall_{j=1,2,\ldots,s}\ T_{i_j} \in \mathcal{T}$ and $\forall_{j=1,2\ldots,s-1}\ T_{i_j}$ has a foreign key to the table $T_{i_{j+1}}$ (or vice-versa). Note that the same table may be traversed multiple times within the same path if it is involved in multiple foreign keys.

Example 2. With reference to the database depicted in Fig. 1, an example of a possible foreign key path is the sequence ⟨`Movies`, `Ratings`, `Users`, `Nations`, `Regions`⟩. This foreign key path is of length 5 (i.e., $s = 5$). Shorter paths representing sub-sequences are also valid, such as ⟨`Movies`, `Ratings`, `Users`⟩.

Definition 2. A task-relevant object *tro* $\in T_k$ ($T_k \in \mathcal{T}$) is *1-related* to a reference object *ro* $\in T_t$ according to a foreign key path $p_i = \langle T_{i_1}, T_{i_2} \rangle$ if and only if all the following conditions are satisfied:

- $T_{i_1} = T_t$;
- $T_{i_2} = T_k$;
- there exists a foreign key constraint from T_k to T_t (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. A task-relevant object *tro* $\in T_k$ ($T_k \in \mathcal{T}$) is *l-related* to a reference object *ro* $\in T_t$ according to a foreign key path $p_i = \langle T_{i_1}, T_{i_2}, \dots, T_{i_s} \rangle$ if and only if all the following conditions are satisfied:

- $s = l + 1$;
- $T_{i_1} = T_t$;
- $T_{i_s} = T_k$;
- there exists a task-relevant object *newTro* $\in T_{i_{s-1}}$ such that *newTro* is $(l-1)$*-related* to *ro* according to p_i;
- there exists a foreign key constraint *fk* from $T_{i_{s-1}}$ to T_s (or vice-versa) such that the foreign key of *newTro* assumes the same value of the primary key of *tro* (or vice-versa).

Definition 4. A task-relevant object *tro* $\in T_k$ is *related* to a reference object *ro* $\in T_t$ according to a foreign key path p_i if and only if there exists $l \geq 1$ such that *tro* is *l*-related to *ro* according to p_i. We indicate as $R(ro, l, p_i)$, the set of task-relevant objects *l*-related to the reference object *ro* according to p_i, and as $R(ro, p_i)$ the set of all the task-relevant objects related to the reference object *ro* according to p_i, for any $l \geq 1$.

In Fig. 2, we give a small example of a database instance, which refers to the schema of Fig. 1. From this figure, it is possible to identify the task-relevant objects for two foreign key paths, namely ⟨`Movies`, `Ratings`, `Users`⟩ (on the left) and ⟨`Movies`, `Ratings`, `Users`, `Ratings`, `Movies`⟩ (on the right). Following a path, it is possible to navigate the objects and their features involved when analyzing a specific reference object.

Example 3. Considering the specific movie *m1*, the objects and their features considered by following the foreign key path ⟨`Movies`, `Ratings`, `Users`⟩, expressed in the logical formalism, are:

```
movies(m1),
   movies_title(m1, fourRooms1995),
   movies_releaseDate(m1, 19950101),
   movies_url(m1, httpusimdbcomMtitle-exactFour20Rooms201995),
   movies_category(m1,thriller),
ratings(r1),
   ratings_userID(r1, u1),
   ratings_movieID(r1, m1),
   ratings_stars(r1, 3),
   ratings_date(r1, 19950105),
ratings(r2),
   ratings_userID(r2, u3),
   ratings_movieID(r2, m1),
   ratings_stars(r2, 2),
   ratings_date(r2, 19950106),
users(u1),
   users_age(u1, 22),
   users_nationID(u1, us),
   users_gender(u1, m),
   users_occupation(u1, writer),
users(u3),
   users_age(u1, 62),
   users_nationID(u1, es),
   users_gender(u2, m),
   users_occupation(u1, engineer). □
```

From this example, it is possible to see that, in the logical formalism, we have two types of predicates: *(i)* unary predicates (e.g., `movies`(·), and `ratings`(·)), used to identify a row (object) in a table (e.g., the predicate `movies`(·) is used to instantiate a row in the table *movies*); *(ii)* binary predicates (e.g., `movies_title`(·, ·) and `ratings_userID`(·, ·)), used to instantiate attribute values (e.g., `movies_title(m1, fourRooms1995)` is used to instantiate the attribute *title* of the table *movies*, for the row *m1*). Note that binary predicates, by means of variables, are also able to implicitly instantiate relationships (e.g. user *u1* gave three stars to the movie "fourRooms1995" in the rating *r1*).

Finally, since our method is strongly based on the concept of aggregates, as introduced in Section 1, in the following we formally define them.

Definition 5. Let $p_i = \langle T_{i_1}, T_{i_2}, \dots, T_{i_s} \rangle$ be a foreign key path of length *s*, *X* be a generic attribute of the table T_{i_s}, and $agg = [\mathrm{aggr}_1(), \mathrm{aggr}_2(), \dots, \mathrm{aggr}_{s-1}()]$ be a list of aggregate functions, all compatible with the type[2] of *X*, where $aggr_j$ applies to the *j*-related objects. The aggregate of *X* with respect to the reference object *ro* and the path p_i is defined as:

$$A(ro, p_i, agg, X) =$$
$$= \underset{tro_1 \in R(ro,1,p_i)}{\mathrm{aggr}_1} \Big(\underset{\substack{tro_2 \in R(ro,2,p_i), \\ tro_2 \frown tro_1}}{\mathrm{aggr}_2} \Big(\dots \underset{\substack{tro_{s-1} \in R(ro,s\text{-}1,p_i), \\ tro_{s-1} \frown tro_{s-2}}}{\mathrm{aggr}_{s\text{-}1}} (tro_{s-1}.X) \dots \Big)\Big) \tag{1}$$

where $tro_j \frown tro_{j-1}$ indicates that there is a foreign key between the two objects tro_j and tro_{j-1}.

Example 4. Considering the following foreign key path p_i = ⟨`movies`, `ratings`, `users`⟩, the feature *Age* of the table *users*, and the list of aggregates *agg* = [avg, avg], the aggregate of Age with respect to the reference object *ro* and the path p_i corresponds to:

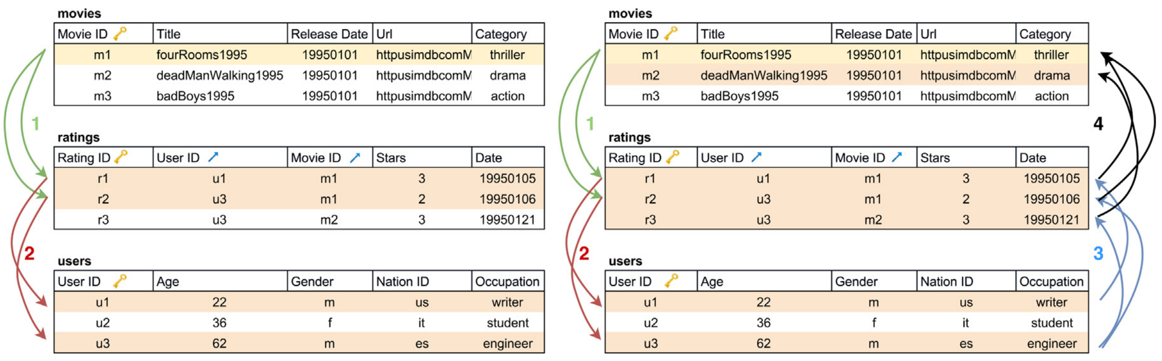
$$A(ro, p_i, agg, Age) =$$
$$= \mathrm{avg}_{tro_1 \in R(ro,1,p_i)} \Big(\underset{\substack{tro_2 \in R(ro,2,p_i), \\ tro_2 \frown tro_1}}{\mathrm{avg}} (tro_2.Age) \Big) \tag{2}$$

In this particular case, $R(ro, 1, p_i)$ is the set of ratings of a given movie *ro*, and $R(ro, 2, p_i)$ is the set of users who provided the ratings. The feature *A* represents the average age of users who have rated a given movie. The internal *avg* aggregation function just returns the age of the user who has made a given rating, since each user has exactly one age value.

Namely, aggregation functions may also be applied to one single value, in the cases of many-to-one or one-to-one relationships from the target table. Multiple values are actually aggregated into one value in the case of one-to-many relationships.

Using the above definitions, in the following section, we describe the proposed method RE3PY, which learns a classification function ϕ_t.

[2] *Average minimum*, *maximum*, *count*, *countDistinct* for numeric types; *mode*, *count* and *countDistinct* for categorical types.



movies

Movie ID	Title	Release Date	Url	Category
m1	fourRooms1995	19950101	httpusimdbcomM	thriller
m2	deadManWalking1995	19950101	httpusimdbcomM	drama
m3	badBoys1995	19950101	httpusimdbcomM	action

ratings

Rating ID	User ID	Movie ID	Stars	Date
r1	u1	m1	3	19950105
r2	u3	m1	2	19950106
r3	u3	m2	3	19950121

users

User ID	Age	Gender	Nation ID	Occupation
u1	22	m	us	writer
u2	36	f	it	student
u3	62	m	es	engineer

Fig. 2. An example of a database instance, together with two possible foreign key paths: ⟨`Movies`, `Ratings`, `Users`⟩ (left) and ⟨`Movies`, `Ratings`, `Users`, `Ratings`, `Movies`⟩ (right). In both cases, the reference object *m1* is emphasized in light orange, whereas the corresponding task-relevant objects that are found in two (left), respectively four steps (right), are emphasized in pink. Primary keys are indicated through a key symbol, while foreign keys are indicated through an arrow.

4. The proposed method RE3PY

In this section, we formally introduce the proposed method. We start with the language bias and the role of aggregates. We then describe the tree induction and finally discuss the proposed ensemble approaches.

4.1. Language bias and aggregates

The core characteristic of our method is the on-the-fly construction of new features, according to Eq. (1), that are considered in candidate splits by our algorithm for the top-down induction of decision trees (TDIDT) [6]. Such new features are constructed by navigating the foreign key paths, as shown in Example 3.

Algorithmically, given a foreign-key path and a reference object *ro*, we start navigating from the target table. Considering the database instance in Fig. 2 and the path ⟨`movies`, `ratings`, `users`⟩, the first step is to identify all 1-related task-relevant objects, e.g., all the ratings provided for a given movie. For example, for the movie *m1*, we identify the ratings *r1* and *r2*. Then, for each of the 1-related task-relevant objects, the walk continues towards the 2-related task relevant objects, namely the users *u1* (accessed from *r1*) and *u3* (accessed from *r2*). The procedure is shown in Fig. 3(c). The general schema of finding the task-relevant objects is shown in Fig. 3(a).

The features of the traversed objects are aggregated back into a single value through a chain of aggregation functions, as shown in Eq. (1). In other words, each $(l-1)$-related object is replaced by the aggregated value obtained by applying the function aggr_l to the feature under consideration of the corresponding group of *l*-related objects (see Fig. 3(b)). Following the example above, Fig. 3(d) shows the computation of the average age of a user that reviewed a given movie.

The possible foreign key paths depend on the dataset at hand, while the possible aggregation functions depend on the type of attributes, namely:

- average (avg), minimum (min), maximum (max), count and countDistinct, for numeric types;
- mode, count, and countDistinct, for categorical types.

Any combination of a foreign-key path, an attribute of the corresponding *s*-related objects (where *s* is the length of the path), and a list of aggregation functions, defines a candidate feature *f*, i.e., a mapping $ro \mapsto f(ro)$.

In Example 4, where we have the foreign key path p_i = ⟨`movies`, `ratings`, `users`⟩, the feature *Age* of the table *users*, and the list of aggregates agg = [avg, avg], the feature *f* corresponds to the average age of the users who rated a given movie. More formally:

$$f(ro) = A(ro, p_i, agg, Age) =$$

$$= \text{avg}_{tro_1 \in R(ro,1,p_i)} (\text{avg}_{\substack{tro_2 \in R(ro,2,p_i),\\ tro_2 \frown tro_1}} (tro_2.Age)) \quad (3)$$

Note that the *existence* of related task-relevant objects, e.g., the existence of some ratings associated to a given movie (which is the only kind of features considered by approaches like BoostSRL [21]), can be checked by using the *count* aggregation function and the test $f(ro) > 0$ in the internal node of the tree.

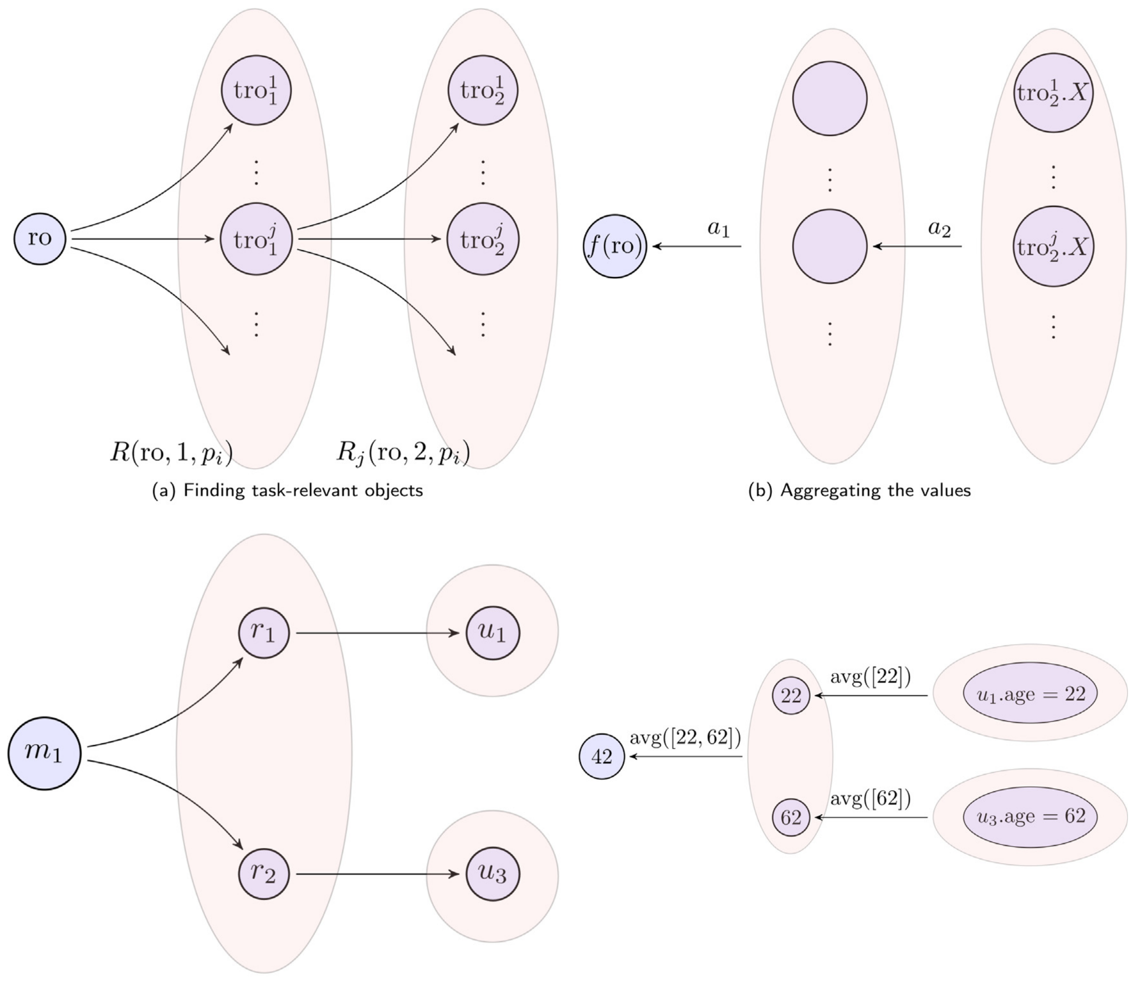
4.2. Single tree induction

Our classification method is based on the top-down induction of decision trees (TDIDT). In particular, both original features associated to the target table and those generated through aggregates, as shown in the previous section, are considered for the evaluation of possible splits in the internal nodes of the tree.

Tree induction starts with the call *Tree*$(E, \emptyset)$, where *E* is the set of all the examples of the target table and the current foreign key path is empty. The pseudocode of *Tree*$(\cdot, \cdot)$, which induces the decision tree, is given in Alg. 1.

Algorithm 1 *Tree*(examples E, current foreign-key path p)

```
1:  {$h^*$, $f^*$ and $S^*$ represent the current best quality, feature and test, i.e., splitting point}
2:  $(h^*, f^*, S^*) = (0, none, none)$
3:  {Loop over the possible features $f$ and splitting sets $S$ for that feature, that can be identified on the set of examples $E$ according to the foreign-key path $p$}
4:  for all valid pairs $(f, S)$, according to $E$ and $p$ do
5:      {If the heuristics $h$ computed according to the feature $f$ and the splitting set $S$ is better than the current best value ($h^*$), update $h^*, f^*$ and $S^*$}
6:      if $h(E, f, S) > h^*$ then
7:          $(h^*, f^*, S^*) = (h(E, f, S), f, S)$
8:  {If no valid test was found, return a leaf node that predicts the majority class}
9:  if $f^* = none$ then
10:     return $Leaf$(majority_class($E$))
11: else
12:     {Otherwise, the examples are split into positive ($E_+$) and negative ($E_-$) subsets, according to the identified best test $f^*(ro) \in S^*$, and the $Tree$ function is recursively called on both $E_+$ and $E_-$}
13:     $E_+, E_-$=$splitToPosNeg(E, ro \mapsto f^*(ro) \in S^*)$
14:     $tree_+ = Tree(E_+, path(f^*))$
15:     $tree_- = Tree(E_-, path(f^*))$
16:     {Finally, return an internal node, where the left and the right children are the results of the previous recursive calls to the $Tree$ function.}
17:     return $Internal(ro \mapsto f^*(ro) \in S^*, tree_+, tree_-)$
```



(a) Finding task-relevant objects (b) Aggregating the values

(c) Task-relevant objects for the movie m_1 (d) Average age of a user who reviewed m_1

Fig. 3. Two-step construction of a feature f in an internal node of a tree. Figure (a) depicts identification of all 1-related (i.e., $R(ro, 1, p_i)$) and 2-related (i.e., $R(ro, 2, p_i)$) task-relevant objects following p_i. For clarity, we show only a subset $R_j(\mathrm{ro}, 2, p_i)$ of 2-related objects — those that are identified from tro_1^j. Figure (b) depicts aggregation of the value of the feature X, back to the target object. For clarity, we only show the aggregation of the groups $R(\mathrm{ro}, 1, p_i)$ and $R_j(\mathrm{ro}, 2, p_i)$. Figures (c) and (d) provide a concrete example (based on the Movies dataset): the considered foreign-key path p_i contains three tables, and the feature f is the average age of the users who have reviewed a given movie. The value of this feature for the movie m_1 is 42.

First, for a given internal node of the tree, all valid tests t in the form "$f(\mathrm{ro}) \in S$" are generated, based on the possible features f and of their domain subsets S. If f is a numeric feature, then $S = (\vartheta, \infty)$ or $(-\infty, \vartheta]$, for an appropriate ϑ. This corresponds to the tests $f(ro) > \vartheta$ and $f(ro) \leq \vartheta$. Otherwise (i.e., when f is nominal), S is a subset of the domain of the attribute (i.e., of the set of its possible values). For example, considering the test "$t = count\ r.Date > 213$" at the root of the tree in Fig. 4, we have $f(ro) = count\ r.Date$ and $S = (213, \infty)$.

A test is not valid if any of the following stopping criteria apply: *(i)* the maximum depth of the tree is reached, *(ii)* the heuristic score of the test is too low, *(iii)* the set of examples falling in the positive or negative branch is too small. The best combination (f^*, S^*) is selected, according to a given heuristic h (Alg. 1, lines 4–7) that, in this work, corresponds to the GINI index [6]. If the set of candidate tests is not empty, the optimal test is of the form $f^*(ro) \in S^*$.

Starting from p, the algorithm can define a new path that starts from it, performing up to ℓ steps. The algorithm follows a depth-first search strategy that also exploits *backtracking*, that is, it can consider any (non-empty) prefix of an already defined path p and move forward, up to at most ℓ steps, from that point. Note that ℓ defines the maximum look-ahead (depth) from the current foreign-key path, and does not correspond to the maximum length of foreign-key paths.

If no valid test can be found, a leaf node with the majority class as the prediction is returned (Alg. 1, line 10). Otherwise, the set of examples E is split into positive (E_+) and negative (E_-) subsets, according to the identified best test $f^*(ro) \in S^*$ (Alg. 1, line 13). In this case, an internal node with the selected test is created, and the left and right children are created by recursively calling the algorithm $Tree(E_+, path(f^*))$ for the positive (left) branch and $Tree(E_-, path(f^*))$ for the negative (right) branch, where $path(f^*)$ is the foreign-key path of the selected test. An example of a relational tree built from the Movies dataset is shown in Fig. 4.

4.3. Relational ensembles in RE3PY

In the literature, one well-known approach to improve predictive performance of tree-based predictive models is that ensemble learning. Typical solutions are based, among the others, on Bagging [8], Random Forests [9] and Gradient Boosting [10].

The concept of a *bag* (a bootstrap replicate), used in Bagging and Random Forests, is different from that in the standard tabular setting. The main difference is that, in the relational setting, the objects are connected and random sampling would lead to losing relationships. In RE3PY, bootstrapping is performed only

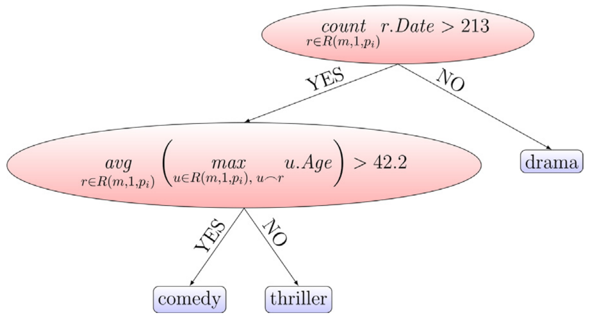

$\underset{r\in R(m,1,p_i)}{count}\ r.Date > 213$
YES
NO
$\underset{r\in R(m,1,p_i)}{avg}\left(\underset{u\in R(m,1,p_i),\ u\frown r}{max}\ u.Age\right) > 42.2$
drama
YES
NO
comedy
thriller

Fig. 4. An example tree for the *Movies* dataset. In the root node, the reviews of the input movie *m* are counted (for technical reasons, we still need a predicate from the table *Ratings*, so the attribute Date was chosen, but note that id_rating could have been chosen as well). If the number of ratings for movie *m* is not greater than 213, the genre of *m* is predicted to be *drama*. Otherwise, the average age of users *u* who rated the movie *m* is computed (note that each user has one single value for his age and the aggregation function, *max* in this case, does not affect the result). If the average age is greater than 42.2, the predicted genre is *comedy*, otherwise it is *thriller*.

on the target table, while task-relevant objects are implicitly sampled according to their relationships with the sampled reference objects.

In addition to sampling the instances, Random Forests also sample a subset of features for each learned tree. In the specific relational setting considered by our system, possible features are represented by original features associated to the target table, or additional features computed by following a given foreign-key path and computing aggregates. Therefore, in RE3PY, we extend the ensemble approach of Random Forests by sampling random subsets of possible features from such an extended set of features. It is noteworthy that this is a cheap operation from a computational viewpoint. Indeed, for the purpose of selecting a sample, we simply enumerate the possible features without actually computing and evaluating their discriminatory power through the heuristic h. In other words, possible tests are actually evaluated only on sampled features.

Finally, boosting-based approaches come with their own randomization mechanisms. Similarly to Random Forests, a subset of examples and features is considered for each tree. The difference is that sampling happens without replacement up to a predefined sample size. Random sampling is not the only way adopted by boosting-based approaches to improve the predictive power of a single tree. More precisely, trees are built in a sequential manner where the first model ϕ_t^1 is built using the target values from the training data, whereas each subsequent model ϕ_t^{j+1}, $j \geq 1$, is built – loosely speaking – on the target values that point in the direction of the gradient of the error of the previous model. To prevent overfitting, the length of the step in the direction of the gradient is determined by the shrinkage parameter $\eta \in (0, 1)$ [10]. The final model is then defined as:

$$\phi_t = \sum_j \phi_t^j \tag{4}$$

4.4. Feature ranking in the relational setting

With the increasing complexity of data under analysis, as well as of the application domains in which predictive models are adopted, the need to explain the output is becoming crucial, especially in specific sectors like medicine [11–13] and life sciences [56–58]. To explain black-box models, such as neural networks, typical approaches rely on permutation-based feature scores: the importance of a given feature is estimated by evaluating the relative increase of the error rate of the predictive model when the values of the feature are permuted.

This strategy cannot be directly applied in the relational setting, since the concept of permutation is not defined for a predicate. On the other hand, we recall that single trees are inherently interpretable, and can naturally be inspected to understand which aspects are influencing the predictions. This is not true for their ensemble counterparts that, although generally appear more accurate, lose this important characteristic. However, the tree induction mechanism allows us to define an alternative strategy to compute feature scores, that can be easily extended to ensembles thereof. In the following, we first define our approach to compute a feature score in the relational setting for single trees and then we extend it to ensembles.

Given a tree whose internal nodes contain tests based on predicates, as described in Section 4, the number of occurrences for a predicate may be considered the most straightforward way to evaluate the importance of that predicate. However, also the depth in the tree is important, since appearing close to the root means that the predicate has been selected earlier during the greedy search performed by the algorithm for TDIDT and, therefore, on the basis of a larger subset of examples (reference objects).

Therefore, an occurrence of a predicate in the test of an internal node should be weighted by the number of reference objects that reach the node. Such intuitive intuition would be the direct extension of the concept of *Symbolic* ranking [59] for the tabular data. However, it is not adequate in our case due to the look-ahead parameter $\ell \geq 1$, which makes a chosen predicate only a part of the feature constructed in the node. For example, a generic feature given in Eq. (1) consists of s parts: $s - 1$ hops between the tables, and the predicate on X. Although the condition expressed in a node $\mathcal{N}$ involves the set of features $f(\mathcal{N})$, only the last steps($\mathcal{N}$) of the condition are constructed in the node $\mathcal{N}$, where steps($\mathcal{N}$) ($\leq \ell$) represents the number of steps added in the node $\mathcal{N}$ of the tree but not present in its parent node.

Thus, the symbolic importance of the feature X for a tree τ can be formally defined as:

$$imp_S(X) = \sum_{\mathcal{N}\in\tau} \mathbb{1}[X \in f(\mathcal{N})]/\text{steps}(\mathcal{N}) \tag{5}$$

where $\mathbb{1}$ is the indicator function.

Symbolic ranking can appear coarse since the reward for a predicate that appears in a given node is always a discrete "jump" that does not depend on the actual quality of the split. This issue can be overcome by adopting the Genie3 score [14], that weights the contribution of each node, when computing the importance of a feature, on the basis of the quality of the split. Genie3 was originally designed for tabular data, but in the following we extend it to the relational setting. Formally, let $h^*(\mathcal{N})$ be the quality of the split performed at node $\mathcal{N}$ measured through the heuristic h. The importance of a predicate X measured through the Genie3 score extended to the relational setting is therefore computed as follows:

$$imp_{\text{Genie3}} = \sum_{\mathcal{N}\in\tau} \mathbb{1}[X \in f(\mathcal{N})]\ h^*(\mathcal{N})/\text{steps}(\mathcal{N}). \tag{6}$$

Note that, when data consist of only one table (containing the reference objects) both measures boil down to their original counterparts designed for tabular data.

The extension of such feature scores to the ensemble-based approaches is quite straightforward. In particular, since the trees of Random Forests and Bagging can be seen as independent realizations of the same random variable, we can simply compute the average of the feature score observed over the trees of the ensemble. On the other hand, the predictions of the trees in Gradient Boosting ensembles need to be summed according to Eq. (4). Coherently, the correct aggregation of per-tree scores feature rankings is also the sum. However, since $\sum_{i=1}^{n} x_i > \sum_{i=1}^{n} y_i \Leftrightarrow (\sum_{i=1}^{n} x_i)/n > (\sum_{i=1}^{n} y_i)/n$, for any numbers x_i and y_i, both averaging and summation lead to the same feature ranking.

Note that computing partial results, while building the set of trees, may reveal how the importance of a feature evolves with each additional tree. In Section 6.5, we will show how feature rankings computed through Bagging, Random Forests and Boosting converge with different running times.

4.5. Differences and advantages over TILDE, BoostSRL and LazyBum

As briefly introduced in Section 2, there have been several attempts in the literature to build classification models from relational data. Among existing approaches, TILDE, BoostSRL and LazyBum are the closest to RE3PY. Therefore, in the following we highlight the main differences and advantages of RE3PY, with respect to them.

First, RE3PY simultaneously exhibits the predictive performance of ensembles and the possibility to explain the predictions, thanks to its ability to produce a ranking of the features.

Second, both TILDE and BoostSRL can handle only nominal values, while RE3PY can also handle numeric attributes.[3]

Third, aggregates are not fully exploited by TILDE, BoostSRL and LazyBum. In particular, BoostSRL is only based on the existence of related task-relevant objects. This means that it is able to construct additional features that would correspond to a particular case of our *count* aggregation function (with $\vartheta = 0$). On the other hand, TILDE and LazyBum also use additional aggregates like RE3PY, but in a different way. In particular, RE3PY uses a sequence of aggregates to compute complex aggregates on the values of variables introduced at the last step of the foreign-key path, while the intermediate objects are only used to reach the last object. On the contrary, LazyBum uses only one aggregate per feature and can produce tests like *Is the average age of users that have rated a given movie, greater than 60?*. TILDE uses the intermediate steps for constraining the sets of task-relevant objects. For example, they can produce[4] tests like *Is the average age of users that have provided at least 5 ratings in total, and have rated a given movie, greater than 60?* but cannot produce tests like *Is the average rate provided by the user that have provided at least 5 ratings in total and have rated a given movie, equal to five?*. One consequence of this aspect is that RE3PY is able to aggregate every variable multiple times (see the example in Fig. 4), but TILDE and LazyBum are not. For the exact formulation of the possible tests in TILDE and LazyBum we refer reader to the works by Vens [60], and Schouterden et al. [30], respectively.

Fourth, more operationally speaking, RE3PY, available as a Python package,[5] is able to build models based on single trees, Random Forests, Bagging and Gradient Boosting ensembles. On the contrary, with BoostSRL, it is possible to build only single trees and Gradient Boosting ensembles.

4.6. Computational complexity

In this section, we estimate the computational complexity of RE3PY, starting from that of single trees and then extending it to ensembles thereof. In the following, we work under the standard assumption that the induced trees are balanced, i.e., their depth is $\mathcal{O}(n)$, where n is the number of reference objects (i.e., rows in the target table). Moreover, we introduce the notation for the following upper bounds. Let:

- F be denote the maximal number of features in the internal nodes;
- s be the maximum length of foreign-key paths;
- b be the branching factor, i.e., the maximum number of 1-related objects for a given object.

According to such assumptions, we can derive that the time complexity of growing a single tree is $\mathcal{O}(Fnb^{s}\log n)$. A thorough derivation of the formula is given in A.

In the case of ensembles, the time complexity can be estimated as follows: bagging boils down to growing T trees, so the time complexity corresponds to T times the time complexity of a single tree. However, T is a constant that does not asymptotically affect the complexity. Moreover, trees are grown independently and, as clarified by Breiman [8], *bagging is almost a dream procedure for parallel computing*. Thus a bagging ensemble can be grown in the same time as a single tree, provided we have enough computing units.

Similarly holds for Random Forests: sequential induction is T times more expensive, but the procedure can be easily parallelized. Moreover, the algorithm considers only $F' \leq F$ features at each internal node of a tree. Typical values for F' are $F' = \lceil\sqrt{F}\rceil$ or $F' = \lceil\log_2 F\rceil$ which can result in substantial speed-ups when the number of features F is large.

Finally, for Gradient Boosting, trees are not grown independently, since the target values are being updated according to the gradients of the predictive errors of the ensemble built so far. However, similar to Random Forests, they can focus on a subset of features and a subset of examples. Moreover, in contrast to the trees with Bagging and Random Forests, which are fully grown (motivated by the bias–variance decomposition of the error [8]), boosted trees are typically prepruned according to a maximum depth and/or a minimum amount of examples per leaf.

5. Experimental evaluation

In this section, we describe the experiments we performed to evaluate the performance of the proposed method. In particular, we will investigate the following research questions:

Q1 Does the considered relational setting provide advantages over its classical propositional counterpart that only focuses on the target table?

Q2 Does the exploitation of the proposed aggregates provide advantages over the use of only existentially-qualified features?

Q3 Do relational tree ensembles provide further advantages over single relational, trees? If yes, which ensemble strategy performs best?

Q4 How does the proposed method perform with respect to state-of-the-art competitors?

Q5 Does the proposed feature ranking method provide real clues on the importance of relational features?

Q6 How sensitive is the proposed method to its input parameters?

In order to answer research questions *Q2* and *Q1*, we compare the results obtained with the full version of RE3PY that uses aggregate relational features to those achieved with two different (restricted) settings, namely, considering only existential qualifiers (to answer research question *Q2*), and discarding all the tables other than the target tables (to answer research question

[3] Actually, the values may be also ordinal, since the trees are invariant to the monotonous transformation of the attributes. Thus, only the order is important and not the actual values.

[4] By combining test in a path from the root to the leaf of a decision tree.

[5] https://pypi.org/project/re3py/

Table 1
Quantitative characteristics of the considered datasets. *Predicates* represents the total number of predicates summed over all the tables in the dataset; *Target facts* is the number of reference objects (i.e., rows in the target table); *Descriptive facts* corresponds to the number of facts describing object attributes and relationships through predicates.

Dataset	Tables	Predicates	Descriptive facts	Target facts	Classes
BASKET	9	118	630038	95	2
IMDB	21	57	614662	8816	4
MOVIE	5	16	183469	1422	4
STACK	7	52	383040	5855	5
UWCSE	12	15	1961	115	5
YELP	9	51	3348181	24959	4
WEBKB	6	9	36131	500	3
CARC	6	12	64640	329	2
MUTA	6	13	32942	188	2

Q1). Moreover, to properly answer research question *Q3*, we evaluate the performance of our system with the three different ensemble approaches introduced in Section 4.3, namely bagging, boosting and random forests, and compare the results with those achieved when learning a single relational decision tree.

All the experiments evaluate model performance through 10-fold cross validation in terms of classification accuracy.

In the following subsections, we first describe the datasets and the competitor systems considered in our empirical comparison. Finally, we report the results, summarized through some statistical tests, and discuss them with respect to each research question.

5.1. Datasets

In the following, we briefly describe the datasets considered in our experiments. In Table 1, we report their quantitative characteristics, while further details and their relational schemata are reported in Appendix B.

- **BASKET**[6] stores the statistics of basketball players of NBA and ABA collected during 2004 and 2005.
- **IMDB**[7] is an extension of the MovieLens10M dataset that also includes data about pages from IMDb and reviews from Rotten Tomatoes.
- **MOVIE** is built from the MovieLens100k[8] dataset and contains the ratings assigned to movies by users, collected through the *movielens* recommender system.
- **STACK**[9] is an anonymized dump of data about the Stack Overflow website, that consist of users, comments, posts, votes, history and links.
- **UWCSE**[10] contains data about the Dept. of Computer Science and Engineering of the University of Washington, including faculty members, projects, publications, courses, and multiple relationships among them.
- **YELP**[11] contains reviews and comments about business activities on the website Yelp. It also includes businesses and their characteristics, users, check-in information, friendship relationships, tips and reviews.
- **WEBKB**[12] contains the textual content, links and anchors of webpages collected by the World Wide Knowledge Base project of the CMU group.
- **CARCINOGENESIS (CARC)**[13] contains data about molecules (e.g., their atoms and their atomic charge) which are used to predict their carcinogenicity.
- **MUTAGENESIS (MUTA),**[13] analogously to CARC, contains data about molecules, e.g., their atoms, bonds, etc., which can be used to predict their mutagenicity.

5.2. Competitor approaches

We compare the results of our approach with 10 state-of-the-art methods that are able to solve classification tasks in a relational setting. The name of the methods and brief descriptions thereof are given below.

- **RelIBk** [27] is the relational variant, available in RelWeka, of the well-known k-nearest neighbors algorithm. As distance measure, we adopted the Relational Instance-Based Learning (RIBL) measure [28].
- **RelSMO** [27] is the relational variant, available in RelWeka, of Platt's Sequential Minimal Optimization algorithm [61]. This algorithm is based on kernel Support Vector Machines and adopts the Minkowski RIBL set distance [29].
- **GNetMine** [40] is a graph-based regularization framework that works in transductive setting. It explicitly aims at preserving the consistency over each relation and each link when assigning a label to unlabeled instances.
- **MrSBC** [23][14] is a method that induces a set of first-order rules from the tables of the relational schema and exploits a naïve Bayes classification method to classify unlabeled instances.
- **ST-MrSBC** (Self-Training MrSBC) [26][15] is a method based on MrSBC, that can capture autocorrelation phenomena using a variant of the self-training method. It bases its predictions on the ensemble of models built over all the self-training iterations.
- **MT-MrSBC** (Multi-Type MrSBC) [26][15] is a method based on MrSBC that iteratively analyzes instances of multiple relations, in order to predict unknown labels in multiple target tables, in order to learn multiple classification functions simultaneously. It builds an ensemble of classifiers for each relation, and is able not only to exploit autocorrelation phenomena, but also dependencies among the models built for different target tables. Here we consider both of its variants, namely **LexicographicMT-MrSBC**, that analyzes the relations in a predefined, lexicographic order, and **RandomMT-MrSBC**, that analyzes the relations in a random order at each iteration.
- **HENPC** [48][16] is a method that exploits the predictive clustering framework on heterogeneous networks represented as relational databases. In particular, it extracts a hierarchy of heterogeneous clusters and exploits it for classification purposes in a transductive setting.
- **BoostSRL** (Boosting for statistical relational learning) [21] is a generalization of the gradient boosting method towards relational data. As introduced in Section 2, the tests in the trees built by BoostSRL are similar to those of our system when no aggregations are used (only existential qualifiers).
- **metapath2vec (m2v)** [50] is a state-of-the-art node embedding method for heterogeneous networks. In this case, we use a logistic regression classifier on the embeddings learned by metapath2vec to solve the node classification task.

6 http://www.cs.cmu.edu/~awm/10701/project/data.html
7 http://grouplens.org/datasets/hetrec-2011/
8 http://grouplens.org/datasets/movielens/
9 https://archive.org/details/stackexchange
10 http://alchemy.cs.washington.edu/data/uw-cse/
11 http://www.yelp.com/dataset_challenge
12 http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/
13 http://kt.ijs.si/janez_kranjc/ilp_datasets/
14 http://www.di.uniba.it/~ceci/micFiles/systems/MURENA.html
15 https://figshare.com/articles/Ensemble_MT-MrSBC_and_Ensemble_ST-MrSBC_systems/4334048/7
16 http://www.di.uniba.it/~gianvitopio/systems/henpc/index.html

5.3. Parameter settings

RE3PY has been evaluated using its full capability of considering the aggregates (henceforth denoted as **AGG-All**), as well using other (restricted) language settings, that consider existential aggregates only (henceforth denoted as **AGG-Exist**), or the target table only (henceforth denoted as **NO-Rel**). Moreover, we performed the experiments considering the ensemble strategies proposed in Section 4.3, namely Random Forests (RF), Boosting and Bagging, as well without any ensemble strategy (i.e., learning a single tree). We set the parameters of RE3PY as follows. We set $\ell = 2$ and the number of trees equal to 50 for the ensemble-based variants of RE3PY. For Bagging, no additional parameters are necessary, since the bias–variance decomposition suggests the induction of fully-grown trees [8]. As for Random Forests, we set the number of features to sample for each tree equal to the square root of the number of features [9]. For Gradient Boosting and single trees, we performed a grid search over all the possible combinations of parameter values via internal 3-fold cross-validation. In particular, for Gradient Boosting, we searched over $\eta(\mathit{shrinkage}) \in \{0.05, 0.2, 0.4, 0.6\}$, $\mathit{chosen\_examples} \in \{0.6, 0.8, 1.0\}$ (i.e., the proportion of the examples that are randomly chosen for tree induction), $\mathit{chosen\_features} \in \{0.2F, 0.4F, 0.6F, 0.8F, F, \sqrt{F}\}$ (where F is the number of all features), and $\mathit{depth} \in \{2, 4, 6, 8\}$. For single trees, we considered $\mathit{leaf\_size} \in \{1, 5, 10, 15, 20\}$ (minimum number of examples per leaf) and $\mathit{impurity} \in \{0, 0.01, 0.02, 0.05, 0.1, 0.2\}$ (minimal relative decrease of the impurity due to a split): The configuration $\mathit{leaf\_size} = 1$ and $\mathit{impurity} = 0$ leads to fully grown trees.

We adopted the same grid search approach for BoostSRL, but focusing only on *shrinkage* and *depth*, since the other parameters are not supported. Moreover, since BoostSRL cannot operate with numeric variables, we used equal-width discretization into 10 bins wherever needed.

For metapath2vec, in order to avoid the introduction of any possible bias due to the manual choice of specific metapaths, we included all the possible metapaths with a maximum length of 3, given that the considered datasets do not require longer metapaths. For fair comparison, since metapath2vec does not naturally consider attributed networks, we pre-processed data by performing attribute reification [51] in order to transform attribute values into nodes. This pre-processing is performed both on nominal and continuous (after equal-size discretization) attributes. We optimized the parameters of the logistic regressor used with metapath2vec through grid search. Specifically, we selected the best results by varying $C \in [0.1; 10]$ (inverse of regularization strength) with a step of 0.2; $\mathit{dual} \in \{\mathit{True}, \mathit{False}\}$ (that represents the adoption of the dual or primal formulation), $\mathit{penalty} \in \{L1, L2, \mathit{elasticnet}\}$ (the norm used in the penalization), and $\mathit{class\_weight} \in \{\mathit{None}, \mathit{Balanced}\}$ (that disables/enables the over-weighting of instances belonging to minority classes).

As regards the other competitors, we took their best published results obtained in previous experiments on the same datasets and with the same experimental setting (including the same splits for the 10-fold cross-validation) [26,48].

All the comparisons, between the different RE3PY language settings, between the different RE3PY ensemble strategies, as well as between RE3PY and competitor systems, have been performed by computing the average rank achieved over the considered datasets. Finally, for all the comparisons, we adopted the Friedman test at $\alpha = 0.05$ to evaluate if the observed differences were statistically significant.

Table 2
The accuracy of RE3PY on all the datasets, with different relational (language) settings and with different ensemble strategies (ST stands for Single Trees). The last column gives the average rank of each setting on the specific dataset across the four ensemble strategies. The best language setting is emphasized in bold.

	Setting	ST	RF	Boosting	Bagging	Avg rank
BASKET	AGG-All	0.968	0.958	0.979	0.979	**1.5**
	AGG-Exist	0.979	0.968	0.968	0.968	**1.5**
	NO-Rel	0.695	0.695	0.695	0.695	3
IMDB	AGG-All	0.587	0.612	0.632	0.628	**1**
	AGG-Exist	0.578	0.576	0.577	0.574	3
	NO-Rel	**0.586**	0.610	0.625	0.616	2
MOVIE	AGG-All	0.517	0.556	0.568	0.567	**1**
	AGG-Exist	0.510	0.510	0.510	0.510	2
	NO-Rel	0.508	0.495	0.507	0.493	3
STACK	AGG-All	0.963	0.950	0.951	0.951	**1.13**
	AGG-Exist	0.950	0.950	0.950	0.950	1.87
	NO-Rel	0.801	0.801	0.801	0.801	3
UWCSE	AGG-All	0.826	0.826	0.835	0.861	1
	AGG-Exist	0.200	0.209	0.191	0.200	3
	NO-Rel	0.252	0.252	0.243	0.252	2
YELP	AGG-All	0.670	0.779	0.879	0.855	**1**
	AGG-Exist	0.588	0.587	0.579	0.588	2.75
	NO-Rel	0.596	0.599	0.574	0.599	2.25
WEBKB	AGG-All	0.924	0.814	N/A	0.932	2.25
	AGG-Exist	0.966	0.968	N/A	0.970	**1.5**
	NO-Rel	0.456	0.388	0.488	0.430	2.5
CARC	AGG-All	0.523	0.547	0.562	0.562	**1.25**
	AGG-Exist	0.535	0.538	0.550	0.538	2.25
	NO-Rel	0.553	0.553	0.553	0.553	2.5
MUTA	AGG-All	0.910	0.920	0.872	0.904	**1.25**
	AGG-Exist	0.665	0.665	0.665	0.665	3
	NO-Rel	0.856	0.888	0.878	0.867	1.75

6. Results

In Table 2, we show the results (in terms of accuracy) obtained by RE3PY in the experiments considering all the possibilities along the following dimensions of analysis:

(i) the dataset;
(ii) the RE3PY relational language setting: all aggregates (AGG-All); existential aggregates only (AGG-Exist); target table only (NO-Rel), where related task-relevant objects are ignored; and
(iii) the RE3PY ensemble strategy.

Starting from these raw results, in the following subsections we discuss the possible answers to different aspects of the research questions introduced in Section 5.

6.1. Q1 — Contribution of the relational setting

In this section, we focus on evaluating the advantages of using relations in learning, that is, using related data stored in additional tables (i.e., task-relevant objects and their relationships to reference objects), as compared to using (focusing on) only the target table. This analysis allows us to evaluate the contribution provided by the relational setting. In Table 2, we evaluate this aspect by comparing AGG-All and AGG-Exist with NO-Rel. From the results, we can see that our full system that exploits all the aggregates (AGG-All) outperforms the variant NO-Rel. A closer look reveals that this is so in 32 out of 36 cases (i.e., in all the cases except for the combinations single trees and Random Forests on CARCINOGENESIS, single trees on IMDB, and Boosting on WEBKB that did not finish within two days).

Intuitively, the obtained results are reasonable. For example, the only descriptive facts of the Movies table of the MOVIE

Table 3
Pairwise statistical comparisons of the relational languages in RE3PY. Results of the Friedman test at $\alpha = 0.05$, corrected through the Benjamini–Hochberg procedure. Statistically significant differences are emphasized in bold.

AGG-All vs NO-Rel	Winner	p-value
Single trees	AGG-All	**0.0081**
RF	AGG-All	**0.0081**
Boosting	AGG-All	0.0955
Bagging	AGG-All	**0.0000**

AGG-Exist vs NO-Rel	Winner	p-value
Single trees	AGG-Exist	0.7600
RF	AGG-Exist	0.7600
Boosting	AGG-Exist	0.7600
Bagging	AGG-Exist	0.7600

AGG-All vs AGG-Exist	Winner	p-value
Single trees	AGG-All	0.3470
RF	AGG-All	0.1980
Boosting	AGG-All	**0.0006**
Bagging	AGG-All	**0.0000**

dataset are the release date, the url and the title. Clearly, they cannot reveal much about the category of a movie. On the other hand, knowing something about the users who provided some ratings on the movies (e.g., their gender or age) can help. The general superiority of the variant AGG-All over NO-Rel is also confirmed by a Friedman test, which reveals that the difference in performance is statistically significant (see Table 3 — AGG-All vs NO-Rel).

On the other hand, the RE3PY variant that uses only existential qualifiers (AGG-Exist) outperforms the NO-Rel variant in only 17 cases out of 36. Coherently, the Friedman test shows that AGG-Exist generally performs better than NO-Rel, but the difference does not appear to be statistically significant at all (see Table 3 — AGG-Exist vs NO-Rel).

Overall, these results confirm that the use of additional information conveyed by task-relevant objects is clearly beneficial, but that it may introduce noise if not handled properly. In particular, the use of only existential qualifiers appears to be not enough and can, on the contrary, negatively affect the results in several situations. This becomes clear after looking at the Avg Rank column in Table 2.

6.2. Q2 — Contribution of the aggregates

In this subsection, we focus on evaluating the advantages of using aggregates in the splits of the trees (the possible aggregates that we use are listed in Section 4.1). In Tables 2 and 3, we can evaluate this aspect by comparing AGG-All with AGG-Exist.

As mentioned in the previous subsection, using aggregates and, therefore, a richer set of possible splits, turns out to be beneficial. Indeed, AGG-All outperforms AGG-Exist in 30 out of 36 cases (i.e., all the combinations except single trees and Random Forests on BASKET and WEBKB, Bagging on WEBKB, and single trees on CARCINOGENESIS). These results are also confirmed by the Friedman test, that shows that the difference between the performance of the two relational languages is statistically significant (see Table 3 — AGG-All vs AGG-Exist).

A closer analysis of the results reveals that for the BASKET dataset many trees have the test on `coach_seas_team`, *count* as aggregation, and the threshold 0 in their root. This means that, in terms of the adopted heuristic, other (more complex) aggregates, possibly exploitable by AGG-All, did not provide a relevant contribution. A similar situation was also observed for WEBKB and CARCINOGENESIS.

In contrast, on the UWCSE dataset, we noticed a significant improvement. In this dataset, the goal is to determine what is the discipline a professor is teaching. By looking at the models, we noticed that one of the splits identified by AGG-All that filters out 18 professors in the `ai` field, is based on the number of persons who had already taught this course, and checking if this number is less than 5. In this case, the use of only existential quantifiers does not provide much information, since every course is taught by at least one professor, and such a pattern would not be discriminative.

6.3. Q3 — Effect of the ensemble strategy

In this subsection, we evaluate the possible contribution provided by the use of relational ensemble approaches, as compared to inducing a single decision tree.

The bias–variance decomposition of the predictive error for Bagging and Random Forests [8] shows that they are expected to exhibit better performances than a single tree, since their bias is the same and the variance is lower. A similar advantage has been (mostly empirically) shown for gradient boosting [62]. In the case of relational data, the derivation of the decomposition is the same. Therefore, the adoption of Bagging or Random Forests is generally advisable. As for Boosting, growing many trees might be computationally too expensive, especially since they cannot be grown in parallel (as demonstrated by the missing results for the WEBKB dataset in Table 2).

Looking at the results of our experiments reported in Table 2 in a column-wise fashion, we now discuss whether such observations also apply to our relational setting with the considered datasets and ensemble strategies.

Comparing **Bagging** with single trees, the only difference would be the bootstrapping of the training set. Therefore, the performance of Bagging should never be substantially worse than that of single trees, provided that the number of examples in the training set is not too small. Looking at the results, we can observe that Bagging wins (obtains a higher accuracy) in 13 cases out of 27, with 11 ties. In the 3 cases in which single trees perform better, the difference is always almost negligible (e.g., 1% for the STACK dataset). In the other direction, the improvement achieved by the use of Bagging may be substantial (e.g., 19% on the YELP dataset).

For **Random Forests**, which additionally sample the features, an extremely large number of split candidates may lead to some performance reduction if the sample size is too small and the majority of the possible splits is irrelevant. This effect theoretically vanishes with an infinite number of trees, but in practice the number of trees is finite. The experimental results show that, in comparison to single trees, the performance of Random Forests is similar to that of Bagging. In particular, we observe 10 wins, 10 ties and 7 loses. A relevant exception is the WEBKB dataset, where the number of possible splits is quite high, due to some nominal variables. As a result, considering $\sqrt{F}$ features per sample appears to be not enough, and a single tree has an advantage of 11% in terms of accuracy.

Looking at the results obtained with **Boosting**, we can observe 10 wins and 6 ties with respect to single trees. Also in this case, we can see substantial performance improvements for the winning cases (see, for example, the results on the YELP dataset), and tiny differences when Boosting loses.

A general comparison among the different ensemble approaches is shown in Table 4. Here, we report the rank of each ensemble approach on each dataset, as well as the average rank.

Although the absolute differences in accuracy are typically not large, Bagging and Boosting consistently outperform Random

Table 4
The ranks observed on each dataset and the average ranks of the three different ensemble strategies used in the relational language setting AGG-All.

Dataset/Ensemble strategy	RF	Bagging	Boosting
BASKET	3	1.5	1.5
IMDB	3	2	1
MOVIE	3	2	1
STACK	3	1.5	1.5
UWCSE	3	1	2
YELP	3	2	1
WEBKB	2	1	3
CARC	3	1.5	1.5
MUTA	1	2	3
Avg rank	2.67	1.61	1.72

Forests, which are ranked in the last position in most cases. The (average) ranks of the other two methods are pretty similar and, performance-wise, it is hard to prefer one approach over the other. However, note that every Boosting model is selected after 288 internal cross-validations used for determining the best parameter setting. Moreover, as already emphasized, trees in Boosting cannot be induced in parallel. In contrast, Bagging has no hyperparameters to tune, and it is easily parallelizable. Therefore, we can conclude that in our relational setting, although RE3PY offers all the available options, learning an ensemble based on Bagging should be generally preferred.

6.4. Q4 — Comparative performance evaluation of RE3PY

In this subsection, we compare the results obtained by RE3PY using all the aggregates (i.e., AGG-All) with Bagging, with those achieved by state-of-the-art methods in their best configurations (see [26,48]). The results are shown in Table 5.

The overall superiority of RE3PY over the competitors is clearly visible in terms of the average ranks computed over all the datasets. Indeed, looking at the last row of Table 5, it is possible to observe that RE3PY ranks, on average, 2.11. The second best method (i.e., metapath2vec) ranks, on average, 3.44.

HENPC outperforms RE3PY on WEBKB. This is probably due to the very small number of features in the dataset, which does not allow RE3PY to take full advantage of the ensemble learning approach. This is confirmed by the relatively small advantage of bagging over single trees in RE3PY. On the other hand, HENPC could not successfully complete the experiments on CARC, due to issues in managing foreign key loops. On this dataset, the only system that outperforms RE3PY is RelSMO, which uses RIBL to generate features and SVMs to solve the learning task. A possible explanation, for this specific case, is that RelSMO takes full advantage of the robustness of SVMs to the high number of (possibly redundant) features generated from all the tables in CARC. The other competitors suffer from the high number of (possibly redundant) relational features compounded with a relatively small number of target instances. In line with general properties of decision trees, RE3PY is also able to select and exploit the most relevant features, resulting in classification accuracy quite similar to that of RelSMO. Finally, for MUTAGENESIS, the impression is that probabilistic approaches and distance-based methods (see RelIBk) perform quite well. This is probably due to the relatively simple structure of MUTAGENESIS. In this case, however, RE3PY still achieves accuracy comparable to those of the best-performing methods.

According to the results, metapath2vec appears to be the best performing method after RE3PY. Although RE3PY provides the best results on three datasets (BASKET, IMDB and STACK) and metapath2vec provides the best results on three datasets (MOVIE, UWCSE and YELP), for the remaining datasets RE3PY outperforms metapath2vec by a large margin (see CARC and MUTA) or achieves comparable results (see WEBKB). This, as already mentioned, leads to RE3PY being better ranked than metapath2vec. A closer analysis of the results explains this behavior with an important limitation of metapath2vec. Specifically, due to its inability to deal with attributed networks, we had to use reification in order to guarantee a fair comparison. Although running metapath2vec without reification would have obviously led to much worse performance, reification introduces problems with numerical attributes, which have to be disctretized a-priori. On the other hand, the aggregations performed by RE3PY allow us to take full advantage of numerical attributes. These considerations are supported by the fact that RE3PY outperforms metapath2vec on all the attributed networks that contain continuous attributes (i.e., BASKET, IMDB, STACK, CARC and MUTA). In this direct comparison, we also stress the fact that RE3PY provides interpretable results (in the form of trees), which is not the case for metapath2vec.

Note that, except for BoostSRL and metapath2vec, the results shown for the competitors can be considered over-optimistic. Namely, they are the outcome of aposteriori selection (on the testing set) of the best parameter configuration. The latter is not chosen by an internal cross-validation (on the training set).

6.5. Q5 — Relational feature ranking

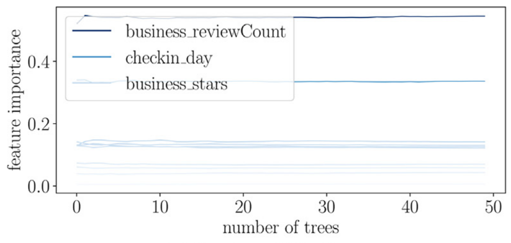
In this section, we show how relational feature ranking can provide clues about how an otherwise black-box relational model works. Indeed, as introduced in Section 4.4, RE3PY includes a feature ranking approach for the relational setting that allows us to identify the contribution of predicates in the construction of ensemble-based models.

As an example, we take a detailed look at the YELP dataset (Fig. 5), for which we show the rankings obtained for the two best-performing ensemble approaches, i.e., Bagging and Boosting, and the two relational languages AGG-All and AGG-Exist. We recall that, in this dataset, the goal is to learn a predictive model that discriminates between `Beauty and Spas`, `Health and Medical`, `Restaurants` and `Shopping` types of businesses.

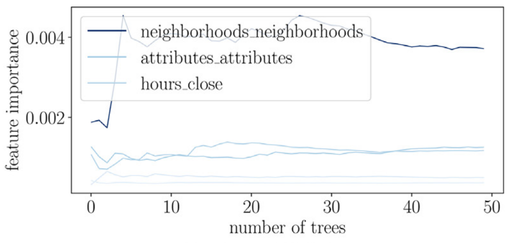
For the Bagging-based models, the difference between AGG-All and AGG-Exist is relevant. Indeed, the importance of the features when using aggregates (Fig. 5(a)) converges quickly as the number of trees increases, and the top-ranked predicates are clearly meaningful (see `business_reviewCount` — the number of reviews, or `business_stars` — the number of stars in a review). On the other hand, the importance of the features for the model learned without aggregates (Fig. 5(b)) does not converge quickly and the top-ranked attributes appear to be meaningless for the task at hand. Indeed, the `neighborhoods` cannot completely determine the business type, nor can the existence of `hours_close`, since every business is located somewhere and every business has a closing time. This difference is also confirmed by the absolute values of the feature scores: they are much lower if the aggregates are not used, meaning that the variance reduction in splits is typically low and that the features cannot effectively discriminate among the classes.

Similar conclusions can be drawn for the Boosting-based models (see Figs. 5(c) and 5(d)). Note that although both versions of Boosting rank `hours_close` among the top-3, AGG-All can exploit it more effectively thanks to the multiple possible aggregates that can be computed on it.

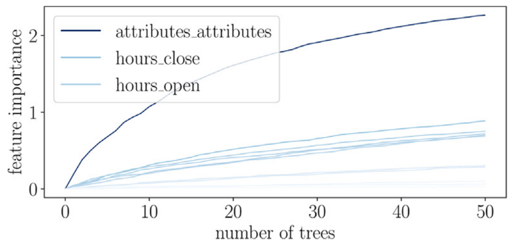
For the other datasets, we show the Genie3 feature importance scores computed from the Bagging-based models that use all the aggregates (AGG-All) in Fig. 6. All the attributes mentioned in the graphs are aggregated, if they do not belong to the target table. For instance, for MOVIE, the best feature is an aggregation over "ratings.movieID", such as count(), which in this case indicates the number of ratings for the given movie.



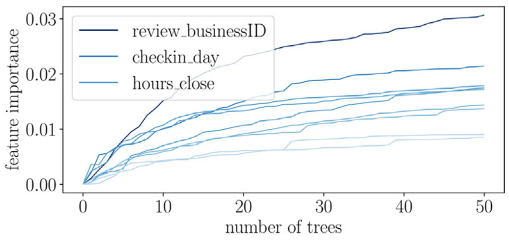
(a) Bagging - AGG-All



(b) Bagging - AGG-Exist

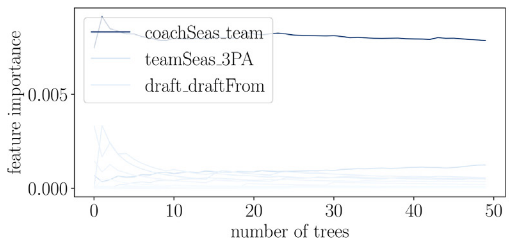


(c) Boosting - AGG-All

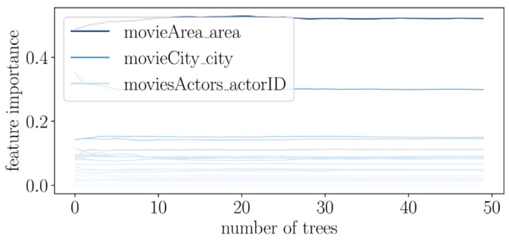


(d) Boosting - AGG-Exist

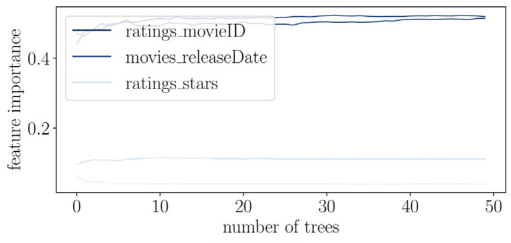
Fig. 5. Feature importance scores for the YELP dataset, for different ensemble configurations. The curve for a given feature shows how the measured value of the Genie3 feature importance score changes with the number of induced trees in the ensemble.



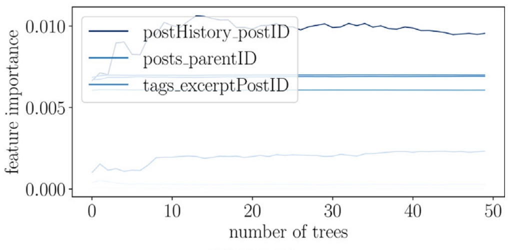
(a) BASKET



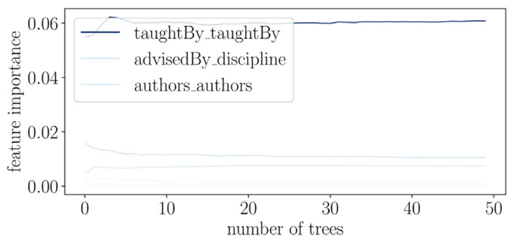
(b) IMDB



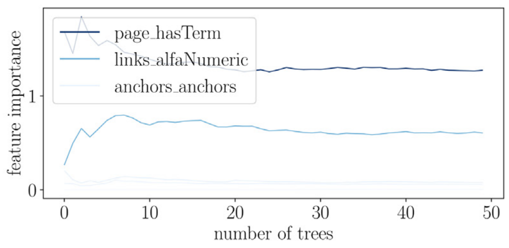
(c) MOVIE



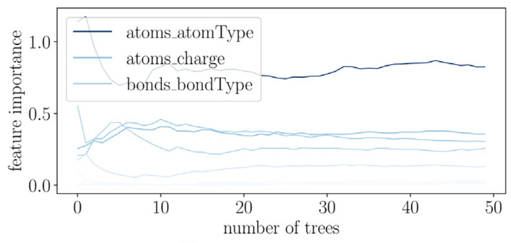
(d) STACK



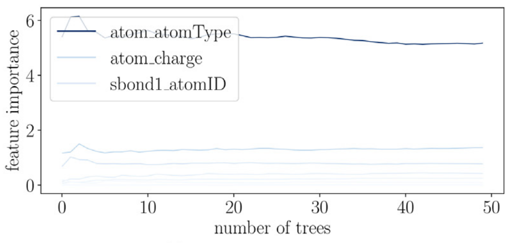
(e) UWCSE



(f) WEBKB



(g) MUTAGENESIS



(h) CARCINOGENESIS

Fig. 6. Feature importance scores returned by RE3PY (Bagging — AGG-All), for all the datasets (except for YELP, for which they are already shown in Fig. 5(a)).

Table 5
Best accuracies of the models learned by RE3PY and its competitors. The best result for each dataset is shown in bold. In the last row, we report the average rank of each approach across the 9 datasets.

Method Dataset	MrSBC	ST MrSBC	MT MrSBC Lex	MT MrSBC Rand	G Net Mine	HENPC	Rel SMO	Rel IBk	Boost SRL	m2v	RE3PY AGG-All
BASKET	0.294	0.264	0.716	0.716	0.747	0.664	0.959	0.959	0.706	0.842	**0.979**
IMDB	0.612	0.586	0.588	0.585	0.576	0.599	0.554	0.581	0.570	0.620	**0.628**
MOVIE	0.433	0.469	0.467	0.462	0.485	0.562	0.510	0.510	0.510	**0.616**	0.567
STACK	0.810	0.758	0.755	0.754	0.213	0.597	0.127	0.876	0.428	0.773	**0.951**
UWCSE	0.274	0.289	0.289	0.289	0.253	0.545	0.205	0.152	0.230	**0.983**	0.861
YELP	0.460	0.581	0.527	0.581	0.497	0.628	0.593	0.538	0.253	**0.931**	0.855
WEBKB	0.585	0.570	0.570	0.570	0.937	**0.978**	0.612	0.606	0.504	0.936	0.932
CARC	0.553	0.553	0.553	0.553	0.440	N/A	**0.586**	0.552	0.446	0.550	0.562
MUTA	0.899	0.920	0.920	0.920	0.663	0.665	0.883	**0.940**	0.336	0.718	0.904
Avg rank	6.72	6.39	6.22	6.94	7.72	5.44	6.11	5.44	9.33	3.44	**2.11**

6.6. Q6 — Parameter sensitivity

In this subsection, we try to answer the last research question, related to the sensitivity of the RE3PY performance to its parameters. Bagging is a special case of Random Forests, where the latter have only one additional parameter, i.e., the size of the subset of the features that is considered in internal splits. The latter is also one of the parameters of Boosting. In the following, we thus only present a detailed sensitivity analysis for Boosting.

As described in Section 5.3, the parameters of Boosting are: depth of the tree, (proportion of) sampled features, (proportion of) sampled examples, and shrinkage. To analyze the influence of a specific parameter, we estimate the distribution of accuracy values for the models learned when such a parameter is fixed to a chosen value and the other parameters vary, through the Python module *seaborn*.[17] The larger the differences among the distributions for different values of the parameter, the larger the influence of the parameter.

In Fig. 7, we show the result of the sensitivity analysis on the UWCSE dataset, that is representative of the sensitivity analysis result across all the datasets. The depth of the trees typically does not play a major role, provided that it is higher than 2 (see Fig. 7(a)). As for the number of sampled features (see Fig. 7(b)), it is clear that taking $\sqrt{F}$ generally does not suffice. Although the optimal value of this parameter depends on the specific dataset, it is clearly beneficial to diversify the trees and choose less then F features (i.e., a proportion less than 1.0). As for the proportion of the sampled examples (see Fig. 7(c)), typically, the more examples the better the results. However, the differences between the performances observed for the higher two values (0.8 and 1.0) are almost negligible. On the other hand, taking only 60% of the examples seems to affect the results negatively. This is interesting, since the expected proportion of examples sampled by Bagging (and Random Forests) is $1 - 1/e \approx 0.63$, but Bagging still achieves state-of-the-art results.

For the dataset at hand, the most influential parameter turns out to be the shrinkage (see Fig. 7(d)). Here, there is a clear preference towards lower values, which means that the dataset is quite hard to model and it is preferable to converge to the final model more slowly.

The results of the parameter sensitivity analysis for the other datasets are available at https://github.com/re3py/re3py.

7. Conclusions

In this paper we present RE3PY, a novel relational classification method. Although many relational classification methods exist in the literature, none of them simultaneously exhibits all characteristics of RE3PY. In particular, RE3PY performs top-down induction of decision trees, exploits multiple aggregates and builds ensembles with multiple strategies (Bagging, Random Forests, and Gradient Boosting). Moreover, while Boosting-based methods with only existential aggregates have already been proposed [21], to the best of our knowledge, RE3PY is the first approach that can exploit multiple kinds of aggregates together with Boosting. Finally, the ability to provide feature rankings in the relational context is an additional strength of RE3PY: it can provide clues about the returned predictions as well as about the importance of individual relational features.

The experiments performed on 9 datasets against 10 state-of-the-art methods, along multiple dimensions of analysis, show that the use of relational data can provide advantages and increase prediction accuracy, especially if properly exploited through multiple possible aggregates as done by RE3PY. Moreover, the multiple ensemble strategies implemented in RE3PY clearly provide advantages in terms of accuracy, as well as flexibility: Among these, Bagging is a clear winner.

Finally, a comparison of the results obtained by RE3PY with those achieved by competitors shows that RE3PY consistently outperforms all of them.

RE3PY is fully open source and publicly available, together with all considered datasets and obtained results. This guarantees full replicability of the experiments, as well as full reusability of the method in multiple contexts.

As future work, we plan to extend the RE3PY method to also solve regression tasks, as well as to handle more complex outputs and tasks (e.g., multi-target regression and classification, multi-label or hierarchical multi-label classification), exploiting different heuristics during the induction of the trees. In addition, we will extend RE3PY to work in the semi-supervised learning setting. Moreover, we will also extend it to work for unsupervised tasks, where learning in the relational setting has not been thoroughly investigated.

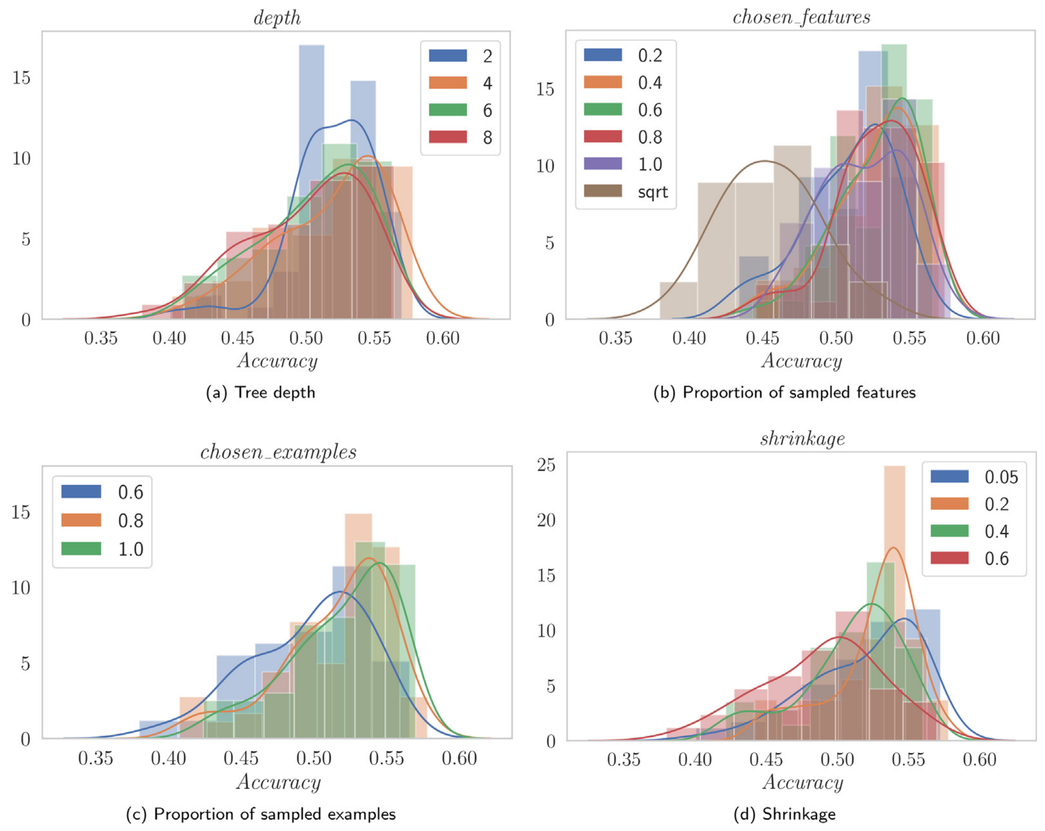
CRediT authorship contribution statement

Matej Petković: Methodology, Software, Formal analysis, Validation, Visualization, Writing – original draft. **Michelangelo Ceci:** Conceptualization, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition. **Gianvito Pio:** Methodology, Formal analysis, Visualization, Writing – original draft, Writing – review & editing. **Blaž Škrlj:** Software, Validation. **Kristian Kersting:** Conceptualization. **Sašo Džeroski:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

[17] https://seaborn.pydata.org/



(a) Tree depth (b) Proportion of sampled features (c) Proportion of sampled examples (d) Shrinkage

Fig. 7. Analysis of the parameter influence through histograms estimated from accuracy values, comparing accuracy distributions on the UWCSE dataset obtained by fixing a given parameter and varying the others.

Acknowledgments

This work was supported by: the Slovenian Research Agency (SRA) via the research program P2-0103, young researcher grants to MP and BŠ, and the grants N2-0128 and J2-2505; the European Commission via TAILOR (a project funded within the EU Horizon 2020 research and innovation programme under GA No 952215); the Italian Ministry of Universities and Research through the project "Big Data Analytics", AIM 1852414, activity 1, line 1 (GP). MP thanks Barbara Krašovec, Janez Srakar and Jan Jona Javoršek for their help with the computing infrastructure.

Appendix A. Upper bound for the number of candidate features and its impact on the time complexity

When deriving the computational complexity of the method in Section 4.6, we assumed an upper bound for the number of features in internal nodes equal to F. In this Appendix, we discuss this upper bound more thoroughly, and explain the factor $F \cdot b^s$ in the time complexity $\mathcal{O}(F \cdot n \cdot b^s \cdot \log n)$, where, as explained in Section 4.6, s is the length of the longest considered foreign-key paths, b is the branching factor and n is the number of reference objects (i.e., rows in the target table).

In the following, we assume that each table contains K foreign keys, and that all predicates in the tables are numeric (this leads to the worst-case scenario, because of the additional sorting operation). If the foreign key paths contain at most s tables, then the number of possible paths is $\mathcal{O}(K^s)$. If a is the number of used (numeric) aggregates, then every path can be aggregated in a^s ways. This means that $F = a^s K^s = (aK)^s$. Note that, potentially, this result can be infeasibly high. However, the parameter s is indirectly controlled by the user-defined look-ahead parameter ℓ, that defines the maximal number of additional steps made in an internal node of a tree. In our experiments, we show we can achieve state-of-the-art results with $\ell = 2$.

As for the evaluation of a feature $f(ro)$ for a given reference object ro, our implementation makes the identification of all *tros* at a given step in the path a constant operation (i.e., we store the pointers to all 1-related sets). Moreover, the aggregation of the 1-related set of each object is computed exactly once. Therefore, the total time needed to evaluate a single split, for a foreign-key path of at most s tables, is proportional to the sum of the size of the 1-related sets of objects involved in the path. Formally, given b the branching factor, the aggregation is done in $\mathcal{O}(\sum_{i=1}^{s} b^i) = \mathcal{O}(b^s)$ operations (assuming $b > 1$). This step needs to be performed for every reference objects, so the total time for evaluating a single split is $\mathcal{O}(n'b^s)$ where n' is the number of reference objects that arrive to a given node.

This result can be used to formalize the time complexity of the RE3PY. In particular, summing over different tree depths $d \in \{1, 2, \dots, \log_2 n\}$ (assuming balancing), the total tree-induction time corresponds to:

$$\mathcal{O}\left(\sum_{d=1}^{\log_2 n} \underbrace{2^d}_{\substack{\text{number of nodes}\\ \text{at depth } d}} \cdot \Bigg[\underbrace{\frac{n}{2^d} \cdot F \cdot b^s}_{\substack{\text{feature value computation}\\ \text{for all } n/2^d \text{ } ros \text{ in a node}}} + F \cdot \underbrace{\left(\frac{n}{2^d} \log_2 \frac{n}{2^d} + \frac{n}{2^d}\right)}_{\substack{\text{sorting the } ros\\ \text{with respect to a feature}\\ +\\ \text{evaluation of the splits}}} \Bigg]\right)$$

$$= \mathcal{O}\left(\sum_{d=1}^{\log_2 n} Fn[b^s + \log_2 n]\right)$$

$$\overset{(*)}{=} \mathcal{O}\left(\sum_{d=1}^{\log n} Fnb^s\right) = \mathcal{O}\left(Fnb^s \log_2 n\right) = \mathcal{O}\left((aK)^s nb^s \log_2 n\right)$$

The step $(*)$ is motivated by the observation that $s \approx \ell \log_2 n$. Indeed, the number of additional hops increases by at most ℓ

every time we increase the depth of the tree. Thus $b^s \approx b^{\ell \log_2 n}$. Since $b \geq 2$, it also follows that $b^{\ell \log_2 n} \geq 2^{\ell \log_2 n} = n^\ell$. Since $n^\ell \geq \log_2 n$, then $b^{\ell \log_2 n} > \log_2 n \implies b^s > \log_2 n$.

The equation $\mathcal{O}(Fnb^s \log_2 n) = \mathcal{O}((aK)^s nb^s \log_2 n)$ completes and confirms the complexity analysis reported in Section 4.6.

Appendix B. Details and schema of the datasets

In this Appendix, we report further details about the considered datasets and their relational schemata.

- **BASKET**[6] This dataset stores the statistics of basketball players of NBA and ABA collected during 2004 and 2005. It includes data regarding the regular season, playoff and all-star games, as well as data related to coaches, players and drafts. The target table is *teams* and the target attribute is *league*, which can have two possible values: *national* or *american* (see Fig. B.1).
- **IMDB.**[7] This dataset is an extension of the MovieLens10M dataset, published by GroupLens. In particular, it includes data from the Movielens dataset, data about the pages from the Internet Movie Database (IMDb) and reviews from Rotten Tomatoes. We kept only the users with both rating and tagging information. The target table is *movies*, and the

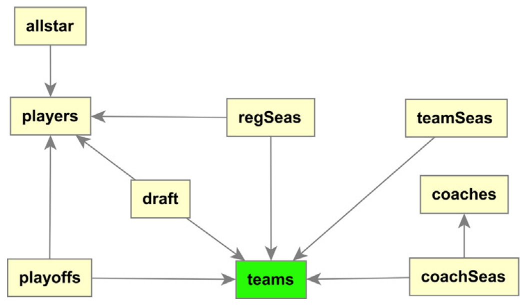


Fig. B.1. The schema of the dataset BASKET, showing the tables and the foreign keys. The target table is shown in green.

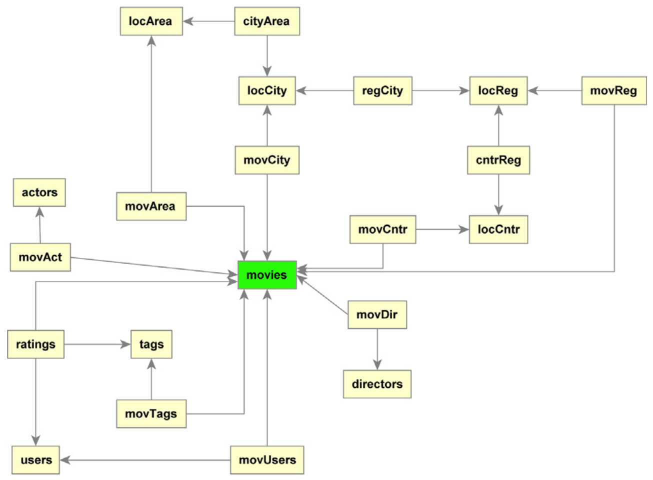


Fig. B.2. The schema of the dataset IMDB, showing the tables and the foreign keys. The target table is shown in green.

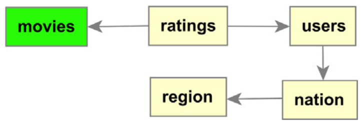


Fig. B.3. The schema of the dataset MOVIE, showing the tables and the foreign keys. The target table is shown in green.

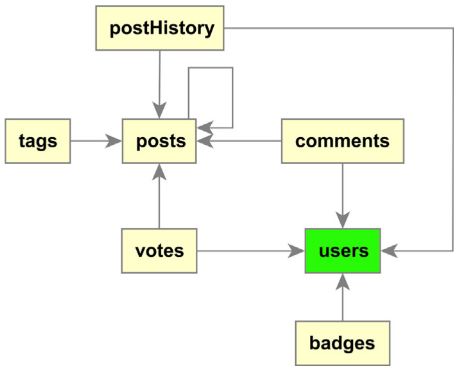


Fig. B.4. The schema of the dataset STACK, showing the tables and the foreign keys. The target table is shown in green.

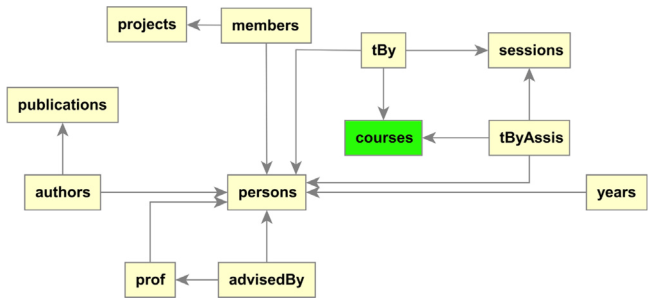


Fig. B.5. The schema of the dataset UWCSE, showing the tables and the foreign keys. The target table is shown in green.

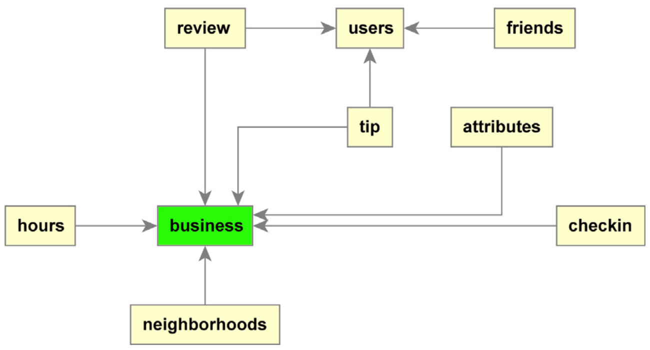


Fig. B.6. The schema of the dataset YELP, showing the tables and the foreign keys. The target table is shown in green.

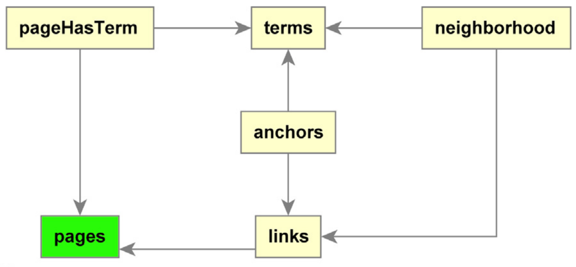


Fig. B.7. The schema of the dataset WEBKB, showing the tables and the foreign keys. The target table is shown in green.

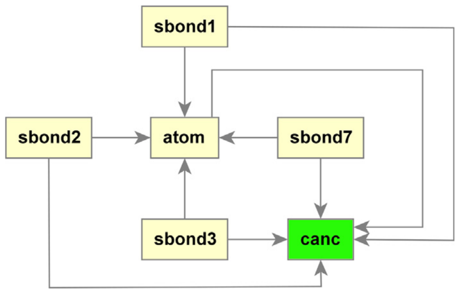


Fig. B.8. The schema of the dataset CARC, showing the tables and the foreign keys. The target table is shown in green.

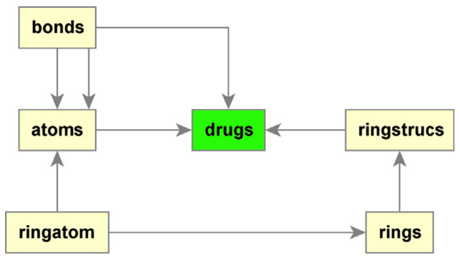


Fig. B.9. The schema of the dataset MUTA, showing the tables and the foreign keys. The target table is shown in green.

target attribute is *genre*, with possible values of *comedy*, *thriller*, *drama* or *action* (see Fig. B.2).

- **MOVIE.** This dataset is built from the MovieLens100k[8] dataset and contains the ratings assigned to movies by users, collected through the *movielens* recommender system. The target table is *movies*, and the target attribute is *category*, which has possible values of *comedy*, *thriller*, *drama* or *action* (see Fig. B.3).
- **STACK.**[9] This dataset is an anonymized dump of the Stack Exchange network. We considered the data about the Stack Overflow website, that consist of users, comments, posts, votes, history and links. The target table is *posts*, and the target attribute is *posttypeid*, the values of which can be *1(Questions)*, *2(Answers)*, *3(Wiki)*, *4(TagWikiExcerpt)* and *5(TagWiki)* (see Fig. B.4).
- **UWCSE.**[10] This dataset contains data about the Department of Computer Science and Engineering of the University of Washington, including faculty members, projects, publications and the courses they teach. Moreover, it also contains data describing the relationships among faculty members (professor, assistant professor, students' adviser, etc.). The target table is *courses* and the target attribute is *discipline*, with values *graphics*, *theory*, *ai*, *language* or *systems* (see Fig. B.5).
- **YELP.**[11] This dataset contains data related to Yelp, that is a website collecting reviews and comments about business activities. Data include businesses and their characteristics, users, check-in information, friendship relationships, tips and reviews. The target table is *business* and the target attribute is *category*, with possible values *Restaurants*, *Beauty & Spas*, *Health & Medical* and *Shopping* (see Fig. B.6).
- **WEBKB.**[12] This dataset contains the textual content, links and anchors of webpages collected by the World Wide Knowledge Base project of the CMU group. The target table is *pages* and the target attribute *category* can have the values *student*, *faculty* and *course* (see Fig. B.7).
- **CARCINOGENESIS (CARC).**[13] This dataset contains data about molecules (e.g., their atoms and their atomic charge) which are used to predict their carcinogenicity. The target table is *canc* and the target attribute is *class*, that can be *1*(cancerogenic) or *0*(not cancerogenic) (see Fig. B.8).
- **MUTAGENESIS (MUTA).**[13] Similarly to CARC, this dataset contains data about molecules, e.g., their atoms, bonds, etc., which can be used to predict their mutagenicity. The target table is *drugs* and the target attribute is *active*, with values *1* (mutagenic) or *0* (not mutagenic) (see Fig. B.9).

References

[1] B. Škrlj, J. Kralj, N. Lavrač, Targeted end-to-end knowledge graph decomposition, in: Proceedings of the International Conference on Inductive Logic Programming LNCS 11105, Springer, 2018, pp. 157–171.
[2] S. Kramer, N. Lavrač, P. Flach, Relational Data Mining, Springer-Verlag, Berlin Heidelberg Germany, 2001, pp. 262–291.
[3] M. Krogel, S. Rawles, F. Zelezny, P. Flach, N. Lavrac, S. Wrobel, Comparative evaluation of approaches to propositionalization, in: Proceedings of the International Conference on Inductive Logic Programming, in: LNCS, vol. 2835, Springer, 2003, pp. 197–214.
[4] J. Knobbe, M. Haas, A. Siebes, Propositionalisation and aggregates, in: Proceedings of the Conference on Principles and Practice of Knowledge Discovery in Databases, in: LNCS, vol. 2168, Springer, 2001, pp. 277–288.
[5] N. Lavrač, S. Džeroski, Inductive Logic Programming: Techniques and Applications, Ellis Horwood, 1994.
[6] L. Breiman, J. Friedman, R. Olshen, C.J. Stone, Classification and Regression Trees, Chapman & Hall/CRC, 1984.
[7] N. Lavrač, B. Škrlj, M. Robnik-Šikonja, Propositionalization and embeddings: two sides of the same coin, Mach. Learn. 109 (7) (2020) 1465–1507.
[8] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[9] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[10] J.H. Friedman, Greedy function approximation: A gradient boosting machine, Ann. Statist. 29 (5) (2001) 1189–1232.
[11] A. Holzinger, G. Langs, H. Denk, K. Zatloukal, H. Müller, Causability and explainability of artificial intelligence in medicine, WIREs Data Min. Knowl. Discov. 9 (4) (2019) e1312.
[12] M. Hoogendoorn, P. Szolovits, L.M. Moons, M.E. Numans, Utilizing uncoded consultation notes from electronic medical records for predictive modeling of colorectal cancer, Artif. Intell. Med. 69 (2016) 53–61.
[13] E. Tjoa, C. Guan, A survey on explainable artificial intelligence (XAI): Towards medical XAI, 2019.
[14] V.A. Huynh-Thu, A. Irrthum, L. Wehenkel, P. Geurts, Inferring regulatory networks from expression data using tree-based methods, PLoS One 5 (9) (2010) 1–10, http://dx.doi.org/10.1371/journal.pone.0012776.
[15] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182.
[16] J.R. Quinlan, R.M. Cameron-Jones, FOIL: A midterm report, in: Proceedings of the 6th European Conference on Machine Learning, in: ECML'93, Springer-Verlag, Berlin, Heidelberg, 1993, pp. 1–20.
[17] S. Muggleton, Inverse entailment and Progol, New Gener. Comput. 13 (3) (1995) 245–286.
[18] H. Blockeel, L.D. Raedt, Top-down induction of first-order logical decision trees, Artificial Intelligence 101 (1) (1998) 285–297.
[19] R. Quinlan, Relational Data Mining, Springer, 2001, pp. 292–304.
[20] A. Van Assche, C. Vens, H. Blockeel, S. Dzeroski, A random forest approach to relational learning, in: ICML 2004 Workshop on Statistical Relational Learning and Its Connections to Other Fields, 2004, pp. 110–116.
[21] S. Natarajan, K. Kersting, T. Khot, J. Shavlik, Boosted Statistical Relational Learners: From Benchmarks To Data-Driven Medicine, in: SpringerBriefs in Computer Science, Springer, 2014, pp. 1–74.
[22] P.A. Flach, N. Lachiche, Naive Bayesian classification of structured data, Mach. Learn. 57 (3) (2004) 233–269.
[23] M. Ceci, A. Appice, D. Malerba, Mr-SBC: A multi-relational Naïve Bayes classifier, in: Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings, in: LNCS, vol. 2838, Springer, 2003, pp. 95–106.
[24] M. Ceci, A. Appice, D. Malerba, Discovering emerging patterns in spatial databases: A multi-relational approach, in: Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007, Proceedings, in: LNCS, vol. 4702, Springer, 2007 pp. 390–397.

[25] M. Ceci, A. Appice, Spatial associative classification: propositional vs structural approach, J. Intell. Inf. Syst. 27 (3) (2006) 191–213.
[26] F. Serafino, G. Pio, M. Ceci, Ensemble learning for multi-type classification in heterogeneous networks, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2326–2339.
[27] A. Woznica, A. Kalousis, M. Hilario, Learning to combine distances for complex representations, in: Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, in: ACM International Conference Proceeding Series, vol. 227, ACM, 2007, pp. 1031–1038.
[28] M. Kirsten, S. Wrobel, T. Horváth, Relational data mining, Springer-Verlag, Berlin, Heidelberg, 2001, pp. 213–230.
[29] J.B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika 29 (1) (1964) 1–27.
[30] J. Schouterden, J. Davis, H. Blockeel, LazyBum: Decision tree learning using lazy propositionalization, in: Inductive Logic Programming — 29th International Conference, ILP, Proceedings, in: LNCS, vol. 11770, Springer, 2019, pp. 98–113.
[31] S.A. Macskassy, F. Provost, Classification in networked data: A toolkit and a univariate case study, J. Mach. Learn. Res. 8 (2007) 935–983.
[32] B. Gallagher, H. Tong, T. Eliassi-Rad, C. Faloutsos, Using ghost edges for classification in sparsely labeled networks, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 256–264.
[33] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (2008) 93–106.
[34] D. Jensen, J. Neville, B. Gallagher, Why collective inference improves relational classification, in: Proceedings of the 10th ACM International Conference on Knowledge Discovery and Databases, ACM, 2004, pp. 593–598.
[35] M. Bilgic, L. Getoor, Effective label acquisition for collective classification, in: ACM International Conference on Knowledge Discovery and Databases, ACM, 2008, pp. 43–51.
[36] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in: Proceedings of the 20th International Conference on Machine Learning, AAAI Press, 2003, pp. 912–919.
[37] D. Malerba, M. Ceci, A. Appice, A relational approach to probabilistic classification in a transductive setting, Eng. Appl. Artif. Intell. 22 (1) (2009) 109–116.
[38] H. Rahmani, H. Blockeel, A. Bender, Predicting the functions of proteins in protein-protein interaction networks from global information, J. Mach. Learn. Res. 8 (2010) 82–97.
[39] A. Appice, M. Ceci, D. Malerba, An iterative learning algorithm for within-network regression in the transductive setting, in: International Conference on Discovery Science 2009, Springer, 2009, pp. 36–50.
[40] M. Ji, Y. Sun, M. Danilevsky, J. Han, J. Gao, Graph regularized transductive classification on heterogeneous information networks, in: Machine Learning and Knowledge Discovery in Databases, European Conference, in: LNCS, vol. 6321, Springer, 2010, pp. 570–586.
[41] M. Ji, J. Han, M. Danilevsky, Ranking-based classification of heterogeneous information networks, in: Proceedings of the 17th International Conference on Knowledge Discovery and Databases, ACM, 2011, pp. 1298–1306.
[42] X. Kong, P.S. Yu, Y. Ding, D.J. Wild, Meta path-based collective classification in heterogeneous information networks, in: 21st ACM International Conference on Information and Knowledge Management, 2012, ACM, 2012, pp. 1567–1571.
[43] C. Yang, M. Liu, F. He, X. Zhang, J. Peng, J. Han, Similarity modeling on heterogeneous networks via automatic path discovery, in: Machine Learning and Knowledge Discovery in Databases — European Conference, Proceedings, Part II, in: LNCS, vol. 11052, Springer, 2018, pp. 37–54.
[44] Y. Dong, Z. Hu, K. Wang, Y. Sun, J. Tang, Heterogeneous network representation learning, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, 2020, pp. 4861–4867, ijcai.org.
[45] K. Steinhaeuser, N.V. Chawla, A.R. Ganguly, Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science, Stat. Anal. Data Min. 4 (5) (2011) 497–511.
[46] D. Stojanova, M. Ceci, A. Appice, S. Dzeroski, Network regression with predictive clustering trees, Data Min. Knowl. Discov. 25 (2) (2012) 378–413.
[47] D. Stojanova, M. Ceci, D. Malerba, S. Dzeroski, Using PPI network autocorrelation in hierarchical multi-label classification trees for gene function prediction, BMC Bioinformatics 14 (2013) 285.
[48] G. Pio, F. Serafino, D. Malerba, M. Ceci, Multi-type clustering and classification from heterogeneous networks, Inf. Sci. 425 (2018) 107–126.
[49] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, P.S. Yu, A survey on heterogeneous graph embedding: Methods, techniques, applications and sources, 2020, arXiv:2011.14867.
[50] Y. Dong, N.V. Chawla, A. Swami, Metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 135–144, http://dx.doi.org/10.1145/3097983.3098036.
[51] L. Badea, Reifying concepts in description logics, in: Proceedings of the 15th International Joint Conference on Artifical Intelligence — Volume 1, in: IJCAI'97, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1997, pp. 142–147.
[52] Y. He, Y. Song, J. Li, C. Ji, J. Peng, H. Peng, HeteSpaceyWalk: A heterogeneous spacey random walk for heterogeneous information network embedding, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, in: CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 639–648, http://dx.doi.org/10.1145/3357384.3358061.
[53] R. Hussein, D. Yang, P. Cudré-Mauroux, Are meta-paths necessary? Revisiting heterogeneous graph embeddings, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, in: CIKM '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 437–446, http://dx.doi.org/10.1145/3269206.3271777.
[54] M. Schlichtkrull, T.N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, M. Alam (Eds.), The Semantic Web, Springer International Publishing, Cham, 2018, pp. 593–607.
[55] Y. Wang, L.H. Zhou, Attributed heterogeneous network embedding based on graph convolutional neural network, in: 2021 International Conference on Communications, Information System and Computer Engineering, CISCE, 2021, pp. 653–659, http://dx.doi.org/10.1109/CISCE52179.2021.9446004.
[56] D. Grissa, M. Pétéra, M. Brandolini, A. Napoli, B. Comte, E. Pujos-Guillot, Feature selection methods for early predictive biomarker discovery using untargeted metabolomic data, Front. Mol. Biosci. 3 (2016) 30.
[57] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.
[58] M. Tsagris, V. Lagani, I. Tsamardinos, Feature selection for high-dimensional temporal data, BMC Bioinformatics 19 (1) (2018) 17.
[59] M. Petković, D. Kocev, S. Džeroski, Feature ranking for multi-target regression, Mach. Learn. 109 (2020) 1179–1204.
[60] C. Vens, Complex Aggregates in Relational Learning (Ph.D. thesis), Faculteit Ingenieurswetenschappen, Katholieke Univeristeit Leuven, 2007.
[61] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in Kernel Methods — Support Vector Learning, MIT Press, 1998, pp. 185–208.
[62] Y.L. Suen, P. Melville, R.J. Mooney, Combining bias and variance reduction techniques for regression trees, in: Proceedings of the 16th European Conference on Machine Learning, Springer, 2005, pp. 741–749.