Estimating the Importance of Relational Features by using Gradient Boosting*

Matej Petković[1,2][0000−0002−0495−9046], Michelangelo Ceci[1,3][0000−0002−6690−7583], Kristian Kersting[4][0000−0002−2873−9152], and Sašo Džeroski[1,2][0000−0003−2363−712X]

[1] Jozef Stefan Institute, Jamova 39, Ljubljana, Slovenia
[2] Jozef Stefan Postgraduate School, Jamova 39, Ljubljana, Slovenia
{matej.petkovic,saso.dzeroski}@ijs.si
[3] Università degli Studi di Bari Aldo Moro, via E. Orabona 4, Bari, Italy
michelangelo.ceci@uniba.it
[4] CS Department, TU Darmstadt, Hochschulstrasse 1, Darmstadt, Germany
kersting@cs.tu-darmstadt.de

Abstract. With data becoming more and more complex, the standard tabular data format often does not suffice to represent datasets. Richer representations, such as relational ones, are needed. However, a relational representation opens a much larger space of possible descriptors (features) of the examples that are to be classified. Consequently, it is important to assess which features are relevant (and to what extent) for predicting the target. In this work, we propose a novel relational feature ranking method that is based on our novel version of gradient-boosted relational trees and extends the Genie3 score towards relational data. By running the algorithm on six well-known benchmark problems, we show that it yields meaningful feature rankings, provided that the underlying classifier can learn the target concept successfully.

Keywords: feature ranking · relational trees · gradient boosting

1 Introduction

One of the most frequently addressed tasks of machine learning is predictive modeling, where the goal is to build a model by using a set of known examples, which are given as pairs of target values and vectors of feature values. The model should generalize well to previously unseen combinations of feature values and accurately predict the corresponding target value. In this paper, we limit ourselves to classification, i.e., to the case when the target can take one of finitely many nominal values. For example, when modeling the genre of a movie, the possible values might be `thriller`, `drama`, `comedy` and `action`.

In the simplest and most common case, the data comes as a single table where rows correspond to examples and columns to features, including the target.

* This is financially supported by the Slovenian Research Agency (grants P2-0103, N2-0128, and a young researcher grant to MP).

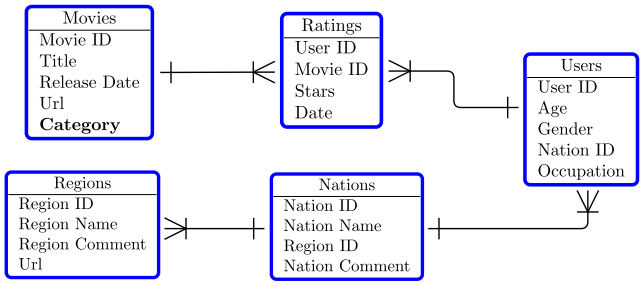

Fig. 1: The `movies` dataset consists of 5 tables and 4 relations among them. Single and multiple ends on the connections among the tables represent one-to- (or -to-one) and many-to- (or -to-many) relations, respectively. The task at hand is to predict the property Category (genre) of a movie (shown in bold).

However, when predicting the genre of a movie, it might be beneficial to not only know the properties of the movie (i.e., the feature values), but also to have access to the properties of ratings of the movie (e.g., number of stars and date) on some website. As a consequence, the data is now represented by two tables: one describing the movies and the other one describing the ratings. A link between them might then be the relation $\mathit{belongsTo}(\text{rating}, \text{movie})$ that tells which movie a given rating rates. The actual `movies` dataset used in our experiments, whose structure is shown in Fig. 1, actually contains five additional tables and four relations among them.

In the era of abundance of data, such complex datasets are more and more frequent in various domains, e.g., sports statistics, businesses, epidemiology [14], among others. In biology, for instance, protein networks [18] encode complex data about proteins and their functions, together with different protein-protein interactions. Concrete examples are given in Sec. 5. In all these cases, relational data are very much of interest because they offer a powerful representation language where a broad spectrum of predictive modeling tasks can be applied. A notable example is link prediction where the goal is to predict whether two examples are in a given relation or not [4], say, two researchers are co-authors of a paper [16]. It is also possible to predict links of multiple types [3]. Another prominent task is classification. For example, movies can be categorized into different genres. Again, predicting multiple targets is possible [12,14].

The main focus of the present paper, however, is not predictive modeling, but rather feature ranking. In the simplest case when the data is given as a single table, the task of feature ranking is to estimate the importance of each descriptive feature when predicting the target value. The features are then ranked with respect to their importance. A possible motivation for performing feature ranking is dimensionality reduction. We simply discard the less relevant features from the dataset, before proceeding to predictive modeling. This results in lower time and space complexity of learning the models, which may also be more accurate since a lower number of features reduces overfitting. Moreover, feature ranking

is sometimes the only way of explaining black box models such as ensembles of decision trees [1,5] or neural networks, which may be of crucial importance in domains such as medicine.

Feature ranking on relational data, however, is not a trivial task. In fact, with two tables or more (and relations between them) at hand, the notion of a feature is a more general concept. It can refer to a relation between tables or to a descriptive attribute, which is part of the description of the objects in a given table. Furthermore, the existing relations and descriptive attributes can be combined into new ones. For example, the schema in Fig. 1 implicitly defines a feature whose value for a given movie is the name of the nation which contributes the highest number of ratings for the movie. Consequently, the feature space can be extremely large, and discovering only the most relevant features for a given predictive modeling task might have a high value. A feature ranking algorithm can estimate the importance of the features explicitly given (such as release date in the `movies` dataset), or it can use some heuristic to construct new features from the ones existing in the descriptive relations.

In this paper, we propose a feature ranking algorithm that focuses on the second approach, i.e., heuristically introduces new features. We use gradient boosting of relational decision trees that can handle complex data and features. The heuristic for building the trees starts the search for relevant features at those which are explicitly given, and gradually proceeds towards more and more complex ones if needed. Once the ensemble is built, feature ranking is seamlessly computed out of it by using the adaptation of the Genie3 score [9].

The remainder of the paper is organized as follows. Sec. 2 explains the necessary background and notation, Sec. 3 reviews related work, Sec. 4 introduces our extension of gradient boosting to relational data and our novel feature ranking algorithm, Sec. 5 gives the experimental setup, while Sec. 6 the results are discussed. Finally, Sec. 7 concludes the work.

2 Background on Relational Predictive Modeling

Let $\mathcal{X}$ be a generic domain that identifies an object type and x be an example (instance) of such type. Every object $x \in \mathcal{X}$ is represented in terms of its ID (object identifier) and a list of attribute values. A relation r of arity $t \geq 1$ is defined as a subset of a Cartesian product of the domains $\mathcal{X}_i$ of relation's arguments X_i, $1 \leq i \leq t$. The fact that $x_1 \in \mathcal{X}_1, \ldots, x_t \in \mathcal{X}_t$ are in a relation r, is denoted either as $(x_1, \ldots, x_t) \in r$ or $r(x_1, \ldots, x_t)$.

If $t = 1$ the relation describes a property of a single object, e.g., *isMale*(user). If $t = 2$, the relation is said to be binary. A binary relation is one-to-many if each $x_1 \in \mathcal{X}_1$ is in a relation with *at least* one $x_2 \in \mathcal{X}_2$, and each x_2 is in a relation with *exactly* one x_1. The other three cases (one-to-one, many-to-one and many-to-many) are defined analogously.

The goal of the relational classification task is thus to learn a model that uses descriptive relations to predict the target relation $r(X_0, Y)$ where the first component corresponds to the example to be classified and the last component

corresponds to the target values. For the link prediction task, the target relation is of form $r(X_0^1, X_0^2, Y)$ where both X_0^1 and X_0^2 correspond to examples and the last component corresponds to the target values. Thus, a relational representation of the data elegantly unifies different classification tasks into the same framework. However, in this work, we focus on the $r(X_0, Y)$ target relations only.

A dataset is typically stored as a group of tables and relations between them. For example, the `movies` dataset in Fig. 1 contains five tables that describe five different types of objects: movies, ratings, users, nations, and regions. Every movie is given as a 5-tuple, consisting of movie ID, title, release date, url, and category. Similarly, ratings are given as 4-tuples of user ID (the author of the rating), movie ID (the movie that rating refers to), stars and date.

Since both the fist component of movies and the second component of ratings are of the same type (movie ID), the dataset also implicitly defines the one-to-many relation via which we can find, for example, all ratings of a given movie. Similar links exist between users and ratings (connected via user ID columns), between users and nations, and between nations and regions. Should there be a relation among users, e.g., *isARelative*, we would need an additional table, e.g., relatives, that would contain two columns (both of the type user ID) and would explicitly list all the pairs of relatives.

Prior to applying our algorithm (and also the majority of those discussed in the related work), the data at hand should be converted into a pure relational representation, coherent with the formal description above, where all facts are given as relation elements $r(x_1, \ldots, x_t)$, e.g., $gender(\text{Ana}, \text{female})$.

3 Related Work

Since our feature ranking is embedded into a classifier [7], it is closely related to relational ensemble techniques. Gradient boosting of relational trees is proposed in [11]. It takes relations (given as sets of tuples) as the input and builds gradient-boosted regression trees. As usual, multi-class problems demand 1-hot encoding of the target variable, which converts the original dataset into a series of 1-versus-all classification problems, and an ensemble is built for each of them separately. In turn, a tree induction in an ensemble bases on the TILDE learner [17] and its predecessor FOIL [13], which results in two possible limitations of the method.

First, it allows only for the nominal descriptive features, thus the numeric ones (e.g., age of a user) should first be discretized into bins which results in the loss of ordering of values. The second possible limitation are candidate tests in the internal nodes of the trees. Without loss of generality, we assume that the variable X_0 that corresponds to the example ID whose target values is to be predicted, always appears as the first component of a relation r. In this case, the candidate splits are of two types. First, a split can be a conjunction of predicates

$$r_1(X_0, x_1^2, \ldots) \wedge \cdots \wedge r_j(X_0, x_j^2, \ldots) \wedge r_{j+1}(x_{j+1}^1, x_{j+1}^2, \ldots) \wedge \cdots \wedge r_\ell(x_\ell^1, x_\ell^2, \ldots)$$

where $\ell \geq 1$, and x_i^k is the value of the variable at position k for relation r_i. Second, some of the variables X_i^k may not be grounded yet (i.e., their value is

not determined). In that case, a split is of the form

$$\exists\, X_{i_1}^{k_1} \ldots \exists\, X_{i_n}^{k_n} :\ r_1(X_0, x_1^2, \ldots) \wedge \cdots \wedge r_\ell(x_\ell^1, x_\ell^2, \ldots) \tag{1}$$

where n is the number of non-grounded variables. When the actual example ID x_0 reaches a split, X_0 takes its value, the split is evaluated, and the example follows the YES or NO branch accordingly. That means that having splits like *Is the average age of users that rated a movie, larger than 60?* is not possible. That was overcome by introducing aggregates into TILDE [17]. There, also constrained aggregation is possible, e.g., *Is the average age of users that have contributed at least 5 ratings in total, and have rated the given movie, larger than 60?*. For the exact formulation of the possible split tests, we refer reader to the [17] where the extension of the method to relational random forests is described.

Regarding relational feature ranking methods, there is only the FARS method [8], which belongs to the group of filters [7], i.e., no predictive model is needed for computing the ranking. As such, it cannot be used for explaining the decisions of classifiers. It is suitable for classification datasets and is based on propagation of the class values from the table that contains the target attribute to the other tables. The method supports neither estimation of numeric attributes nor the estimation of implicitly defined features.

4 Our Method

We first describe the induction of our relational trees (including the proposed test split generation) and boosting ensembles. Afterward, the proposed feature ranking approach is introduced.

4.1 Relational Gradient Boosting with Aggregates

Relational trees are built with the standard top-down induction procedure [2] whose main part is finding the optimal test (according to a heuristic score, e.g., Gini index [2]) that splits the data into two subgroups. The point at which relational tree induction differs from the standard one is how the candidate splits are created. Indeed, one option is using TILDE with aggregates. However, also from the feature ranking perspective, we find the following feature-value back-propagation splits more appropriate.

Consider Fig. 2, which depicts our candidate test generation. Each test is generated in two stages. First (as shown in Fig. 2a), we start from an example ID x_0, and follow any relation r_1 where x_0 can appear in. The group g_1 of all tuples $\boldsymbol{x}_1^i \in r_1$, which x_0 is part of, is thus found. Then, for each of the tuples $\boldsymbol{x}_1^i$, we recursively repeat the search from $\boldsymbol{x}_1^i$, thus finding the group of examples g_2^i by following any relation r_2 that shares at least one input domain $\mathcal{X}$ with relation r_1. The search is finished after at most ℓ steps which is a user-defined parameter. In Fig. 2a, we have $\ell = 2$, but mostly, already $\ell = 1$ suffices. For example, following the schema of the `movie` data set in Fig. 1, we might start from $x_0 =$ `titanic`,

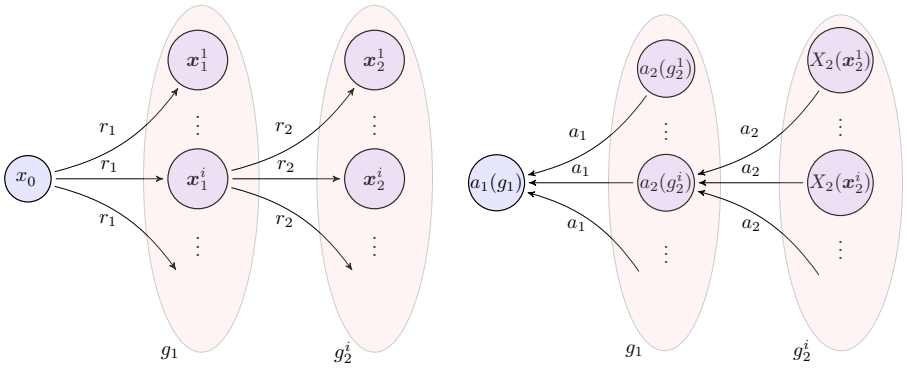

Fig. 2: Candidate test generation: Finding related tuples by following descriptive relations (a), and back-propagation of the feature values by aggregation (b).

and find the group g_1 of all pairs $\boldsymbol{x}_1^i = (\texttt{titanic}, \texttt{ratingID}^i)$. For each such pair, a (singleton) group g_2^i of all pairs $(\texttt{ratingID}^i, \texttt{stars})$ is found.

After the search is finished, the back-propagation of feature values by aggregation starts, by choosing one of the variables that were introduced in the last step of the search. Let this be X_ℓ. Its type defines possible starting aggregates. We can always use *count* or *countUnique*. Additionally, we can use *max*, *min*, *mean*, and *sum* if X_2 is numeric, and *mode* if X_2 is nominal. The data type returned by the first aggregate in turn defines the possible options for the next one in the chain. In general, we proceed form right to left, as illustrated in Fig. 2b.

By using aggregate a_2, we aggregate every group g_2^i over X_2. Following the example above, the variable at hand would be the number of stars. Then, the tuple $\boldsymbol{x}_1^i \in g_1$ is effectively replaced by the aggregated value $a_2(g_2^i)$. After this is done for all tuples $\boldsymbol{x}_1^i$, the group g_1 is similarly aggregated into the final value $a_1(g_1)$. If we set a_2 to *mean* and a_1 to *max*, in the above example, we compute the maximal rating of a movie (the mean of a single value is the value itself).

Finally, the procedure for generating candidate splits proceeds to finding the optimal threshold ϑ the aggregated values are compared against, and chooses the best test among the candidates. This covers also the existentially quantified split tests in Eq. (1) if $\vartheta = 0$ and the aggregates are set to *count*. This motivates the idea to allow the algorithm to continue the search for a good split in the YES-child at any of the steps from its parent node. Therefore, the evaluation of the tests becomes more time-consuming deeper in the tree, so gradient boosting [5] might be more efficient than, e.g., bagging [1] where bias-variance decomposition of the error reveals that trees should be grown to a full depth.

Using aggregates is necessary since the preliminary experiments (not part of this work) show that this increases the expressiveness of the splits which reflects in substantially improved predictive power of the models.

4.2 Feature Ranking

Now, we are ready to introduce our novel relational feature ranking. Let (R, A, ϑ) be a triplet denoting a test in a node $\mathcal{N}$ in a tree $\mathcal{T}$, where R and A are lists of relations and aggregates used in the test. The size s of a test is defined as $s = |R|$ $(= |A|)$. Let $E(\mathcal{N})$ be the set of examples that reach the node $\mathcal{N}$, and $h(\mathcal{N})$ the heuristic value of the split in the node. We write $r \in \mathcal{N}$ if $r \in R$. Then, a natural extension of the Genie3 score [9] for the already existing relations r is

$$importance(r) = \sum_{\mathcal{T}} \sum_{\mathcal{N} \in \mathcal{T}} \frac{\mathbf{1}[r \in \mathcal{N}]}{s(\mathcal{N})} h(\mathcal{N}) |E(\mathcal{N})|, \tag{2}$$

where $\mathbf{1}$ is the indicator function. The definitions says that all relations that appear in a given node are rewarded equally and proportionally to the heuristic value. The term $|E(\mathcal{N})|$ assures that relations that appear higher in a tree (and influence more examples) receive bigger award.

Please note that Eq. (2) is naturally extended to (parts of) lists R of relations by summing up the importances of their atomic parts. Note also that the relations and combinations thereof that do not appear in the ensemble, have the importance score with the value 0.

5 Experimental Setting

In order to investigate the performance of our relational feature ranking methods, we computed feature rankings for six well-known datasets. In addition to the `movie` dataset (shown in Fig. 1), these were: `basket` [10] (basketball players, coaches, teams, etc.), `IMDB` [6] (movies, actors, directors, etc.), `Stack` [15] (user posts, users and comments), and `Yelp` [19] (different businesses, their reviews, users etc.). Additional statistics for the data are given in Tab. 1.

In our experiments, the quality of the underlying predictive model will be used as a proxy for the quality of the ranking, thus 10-fold cross-validation (CV) is performed. For each training set, internal 3-fold CV was performed to tune the boosting parameters via grid-search. The parameters (and their possible

Table 1: Data characteristics: number of tables, number of relations in the final representation, sum of relation sizes (number of descriptive facts), number of target facts, number of classes, and the proportion of the majority class.

dataset	tables	relations	descriptive facts	target facts	classes	majority class
basket	9	118	630038	95	2	0.70
IMDB	21	57	614662	8816	4	0.58
movie	5	16	183469	1422	4	0.51
Stack	7	52	383040	5855	5	0.36
UWCSE	12	15	1961	115	5	0.25
Yelp	9	51	3348181	24959	4	0.57

values) that were optimized are shrinkage ($\{0.05, 0.2, 0.4, 0.6\}$), proportion of chosen examples ($\{0.6, 0.8, 1.0\}$), proportion of the evaluated tests in a node ($\{0.2, 0.4, \ldots, 1.0, \text{sqrt}\}$), and maximal depth of trees ($\{2, 4, 6, 8\}$). Ensemble size was set to 50 and the size of splits was $\ell \leq 2$.

6 Results and Discussion

The experimental results are summarized in Fig. 3. It shows the feature importances, averaged over the 10-folds of CV, and the three most relevant features, for each dataset. We observe two qualitatively different results: for the datasets `basket`, `Stack` and `UWCSE` (and to some extend `Yelp`), it is evident that feature importance score have mostly converged to their final values, meaning that the Gini heuristic in the splits goes down to 0.0 and the trees with higher indices do not influence the ranking or predictions of the model. This is confirmed by the accuracy of the corresponding models: 0.98 (`basket`), 0.95 (`Stack`), 0.83 (`basket`) and 0.88 (`Yelp`). Since accuracy on the training sets are even higher, this means that only a few training examples are being miss-classified and the target values at 50-th iteration are mostly close to 0.0.

The two most prominent members of the second class of results are `IMDB` and `movie` where the feature importance scores are still noticeably growing. On the other hand, the order of the features is mostly fixed, so trees are similar to each other, but unable to fully solve the predictive problem. Indeed, the accuracy of the corresponding two models is 0.63 (`IMDB`) and 0.57 (`movie`) which is closer to the default accuracy than in the previous cases (see Tab. 1).

The next observation is that for all six datasets, a group of 1–3 most important features is established. The difference between this group and the other features is most notable for the `UWCSE` dataset. Here, the goal is to predict, which discipline a given course belongs to, and the most important relation is $\mathit{taughtBy}(\text{course}, \text{person}, \text{session})$. This is not surprising as it is also the link to the $\mathit{advisedByDiscipline}(\text{person}, \text{person}, \text{discipline})$, ranked second. Since professors typically work only in one discipline, the discipline a professor advises someone in, is likely equal to a discipline that the professor teaches.

A similar difference between the top feature and the others is visible also in the cases of the `Yelp` dataset (as well as for `basketball`). In the case of `Yelp`, the goal is to predict the category of a business (e.g., restaurant, Health&Medical etc.), and counting in how many tuples of the *attributes* relation a business appears, is a good indication of the target value. For example, restaurants tend to have a lot of attributes such as classy, hipster, romantic, dessert etc. whereas Health&Medical places are sometimes described only by `ByAppointmentOnly`.

In the cases where the models are not that accurate, feature ranking is a way of seeing whether the model overfits and if some relations should be excluded from the descriptive space. This happens in the case of the `IMDB` dataset where the goal is to predict the genre of a movie but the most relevant feature is the area where the movie was taken.

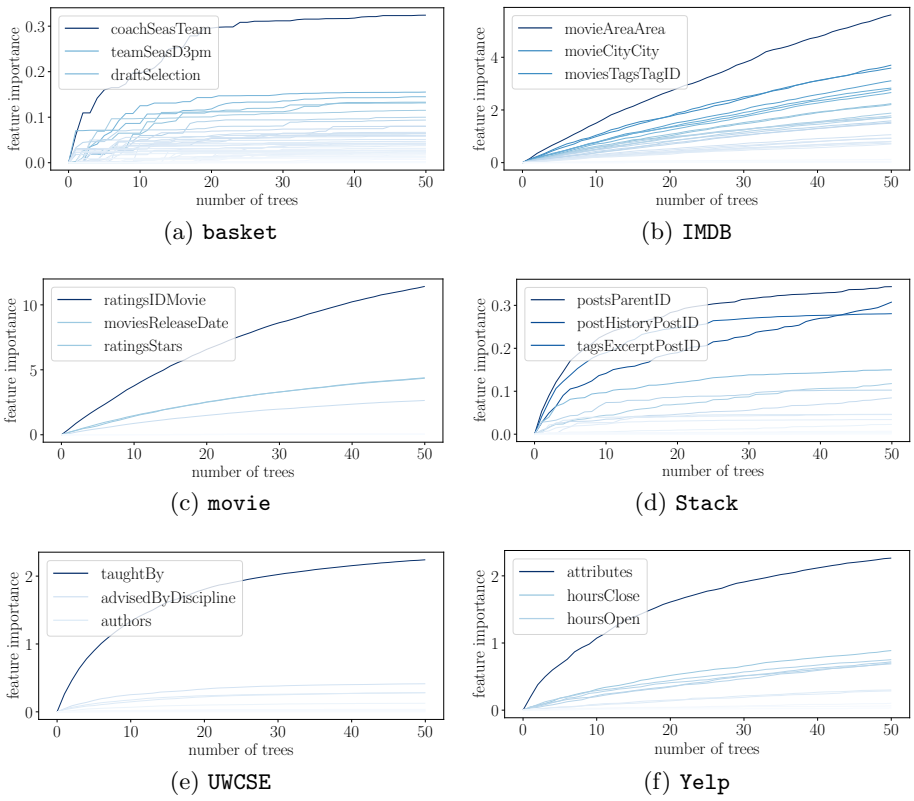

Fig. 3: Development of feature importance values with the number of trees, for different datasets. Every line corresponds to a feature that is present in an ensemble. It's color corresponds to the final feature importance at 50 trees. Additionally, the three most relevant features are listed.

7 Conclusions and Future Work

We have proposed an adaptation of the Genie3 feature ranking to relational data, using our adaptation of gradient boosting as the underlying ensemble, and evaluated its appropriateness empirically. The main motivation for choosing boosting was that the trees learned in boosting can be quite shallow as compared to those learned in bagging. However, parameter-tuning for boosted relational trees takes a considerable amount of time, so we plan to extend the proposed approach to other ensemble methods, such as bagging and random forests (and parallelize them). Also, the feature-ranking-motivated definition of the possible split tests will be evaluated in a predictive modeling scenario. Finally, we plan to compare the different relational ensembles against each other and against other learners.

Acknowledgements

We thank Martin Breskvar for his help with drawing Fig. 1 and the insightful discussions.

References

1. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (1996)
2. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
3. Davis, D., Lichtenwalter, R., Chawla, N.V.: Multi-relational link prediction in heterogeneous information networks. In: 2011 International Conference on Advances in Social Networks Analysis and Mining. pp. 281–288 (2011)
4. Dong, Y., Tang, J., Wu, S., Tian, J., Chawla, N.V., Rao, J., Cao, H.: Link prediction and recommendation across heterogeneous social networks. In: 2012 IEEE 12th International Conference on Data Mining. pp. 181–190 (2012)
5. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics **29**(5), 1189–1232 (2001)
6. GroupLens Research: Imdb dataset, data retrieved from `https://grouplens.org/datasets/hetrec-2011/`
7. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3**, 1157–1182 (2003)
8. He, J., Liu, H., Hu, B., Du, X., Wang, P.: Selecting effective features and relations for efficient multi-relational classification. Computational Intelligence **26**, 258–281 (2010)
9. Huynh-Thu, V.A., Irrthum, A., Wehenkel, L., Geurts, P.: Inferring regulatory networks from expression data using tree-based methods. PLOS ONE **5**(9), 1–10 (09 2010). https://doi.org/10.1371/journal.pone.0012776
10. Moore, A.W.: Basket dataset, data retrieved from `http://www.cs.cmu.edu/~awm/10701/project/data.html`
11. Natarajan, S., Kersting, K., Khot, T., Shavlik, J.: Boosted statistical relational learners: From benchmarks to data-driven medicine, pp. 1–74. Springer (2014)
12. Pio, G., Serafino, F., Malerba, D., Ceci, M.: Multi-type clustering and classification from heterogeneous networks. Information Sciences **425**, 107–126 (2018)
13. Quinlan, J.R.: Boosting first-order learning. In: Arikawa, S., Sharma, A.K. (eds.) Algorithmic Learning Theory. pp. 143–155. Springer Berlin Heidelberg (1996)
14. Serafino, F., Pio, G., Ceci, M.: Ensemble learning for multi-type classification in heterogeneous networks. IEEE Transactions on Knowledge and Data Engineering **30**(12), 2326–2339 (2018)
15. Stack Exchage: Stack dataset, data retrieved from `https://archive.org/details/stackexchange`
16. Sun, Y., Barber, R., Gupta, M., Aggarwal, C.C., Han, J.: Co-author relationship prediction in heterogeneous bibliographic networks. In: 2011 International Conference on Advances in Social Networks Analysis and Mining. pp. 121–128 (2011)
17. Vens, C.: Complex aggregates in Relational Learning. Ph.D. thesis, Faculteit Ingenieurswetenschappen, Katholieke Univeristeit Leuven (2007)
18. Škrlj, B., Kralj, J., Lavrač, N.: Targeted end-to-end knowledge graph decomposition. In: Riguzzi, F., Bellodi, E., Zese, R. (eds.) Inductive Logic Programming. Springer (2018)
19. Yelp: Yelp dataset, data retrieved from `www.yelp.com/dataset_challenge`