Machine Learning (2023) 112:4379–4408
https://doi.org/10.1007/s10994-022-06181-0

Feature ranking for semi-supervised learning

Matej Petković[1,2] · Sašo Džeroski[1,2] · Dragi Kocev[1,2]

Received: 11 March 2020 / Revised: 28 January 2022 / Accepted: 9 May 2022 /
Published online: 22 June 2022

Abstract
The data used for analysis are becoming increasingly complex along several directions: high dimensionality, number of examples and availability of labels for the examples. This poses a variety of challenges for the existing machine learning methods, related to analyzing datasets with a large number of examples that are described in a high-dimensional space, where not all examples have labels provided. For example, when investigating the toxicity of chemical compounds, there are many compounds available that can be described with information-rich high-dimensional representations, but not all of the compounds have information on their toxicity. To address these challenges, we propose methods for semi-supervised learning (SSL) of feature rankings. The feature rankings are learned in the context of classification and regression, as well as in the context of structured output prediction (multi-label classification, MLC, hierarchical multi-label classification, HMLC and multi-target regression, MTR) tasks. This is the first work that treats the task of feature ranking uniformly across various tasks of semi-supervised structured output prediction. To the best of our knowledge, it is also the first work on SSL of feature rankings for the tasks of HMLC and MTR. More specifically, we propose two approaches—based on predictive clustering tree ensembles and the Relief family of algorithms—and evaluate their performance across 38 benchmark datasets. The extensive evaluation reveals that rankings based on Random Forest ensembles perform the best for classification tasks (incl. MLC and HMLC tasks) and are the fastest for all tasks, while ensembles based on extremely randomized trees work best for the regression tasks. Semi-supervised feature rankings outperform their supervised counterparts across the majority of datasets for all of the different tasks, showing the benefit of using unlabeled in addition to labeled data.

Keywords Feature ranking · Semi-supervised learning · Tree ensembles · Relief · Structured output prediction · Multi-target prediction

Editors: Annalisa Appice, Grigorios Tsoumakas.

✉ Dragi Kocev
dragi.kocev@ijs.si

Extended author information available on the last page of the article

1 Introduction

In the era of massive and complex data, predictive modeling is undergoing some significant changes. Since data are becoming ever more high dimensional, i.e., the target attributes potentially depend on a large number of descriptive attributes, there is a need to provide better understanding of the importance or *relevance* of the descriptive attributes for predicting the target attributes. This is achieved through the task of feature ranking (Guyon and Elisseeff 2003; Jong et al. 2004; Nilsson et al. 2007; Petković et al. 2020): the output of a feature ranking algorithm is a list (also called a *feature ranking*) of the descriptive attributes x_i ordered by their importance (relevance) $importance(x_i)$ to the target attribute(s). The obtained feature ranking can then be used in two contexts: (1) to better understand the relevance of the descriptive variables for the target variable or (2) to perform feature selection and reduce the number of descriptive variables. The latter not only decreases the computational complexity of building a predictive model later on, but also makes the models (that use fewer features) easier to explain and understand. This is of high importance in a variety of application domains such as medicine (Holzinger et al. 2019; Hoogendoorn et al. 2016; Tjoa and Guan 2020), life sciences (Grissa et al. 2016; Saeys et al. 2007; Tsagris et al. 2018) and ecological modeling (Bhardwaj and Patra 2018; Galelli et al. 2014; Zhou et al. 2018).

For some domains, having a large number of features can be computationally demanding, while at the same time labelling even a moderate number of examples might be very expensive. For instance, when doing sentiment analysis of text, e.g., tweets (Kralj Novak et al. 2015), or determining properties of new chemical compounds (DiMasi et al. 2003), e.g., in QSAR (quantitative structure activity relationship) studies (which is one of the considered datasets in the experiments), one can only label a limited quantity of data, since labeling demands a lot of human effort and time (labeling tweets Kralj Novak et al. 2015), or is expensive (performing wet lab QSAR experiments). Since the cases where many examples remain unlabeled are not that rare, advances in predictive modeling have brought us to the point where we can make use of them. In this work, we focus on semi-supervised learning (SSL) techniques that handle data where some examples are labeled and some are not (as opposed to supervised learning (SL), where all examples are labeled).

The SSL approaches are all based on the assumption that the distribution of the target values is well-reflected in the structure of the data, i.e., on the clustering hypothesis, stated below:

Assumption 1 (*Clustering Hypothesis*) Clusters of examples (in the input space of descriptive attributes) well resemble the distribution of target values (in the output space).

The hypothesis is not precisely defined, but a way of quantifying its validity is proposed in Sect. 6.2. The rationale behind the hypothesis is as follows. If the clustering hypothesis is satisfied, then an SSL algorithm that can make use of unlabeled data, may outperform the classical SL algorithms that simply ignore unlabeled data. This holds for predictive modeling tasks (Levatić 2017; Zhu et al. 2009), and as we show in this work, for feature ranking tasks also.

In addition to the massiveness, the complexity of the data is also increasing. Predictive modeling is no longer limited to the standard classification and regression tasks, but also tackles their generalizations. For example, in classification, the target variable may take only one of the possible values, for each example in the data. On the other hand,

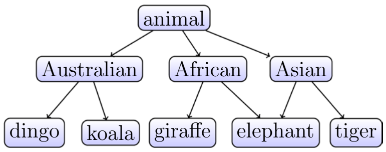


Fig. 1 An example hierarchy of animal related labels

problems such as automatic tagging (e.g., the `Emotions` dataset (see Sect. 6.2), where the task is to determine emotions that a given musical piece carries) allow for more than one label per example (e.g., a song can be *sad* and *dramatic* at the same time). A further generalization of this problem is hierarchical multi-label classification, where the possible labels are organized into a hierarchy, such as the one in Fig. 1 (Sect. 2), which shows animal-related labels. If a model labels an example as a *koala* (Sect. 6.2), it should also label it with the generalizations of this label, i.e., *Australian* and *animal*. Similarly, the task of regression can be generalized to multi-target regression, i.e., to predicting more than one numeric variable at the same time, e.g., predicting canopy density and tree height in forests (cf. the `Forestry` dataset in Sect. 6.2).

The main motivation for the generalized predictive modeling tasks is that considering all the target variables at the same time may exploit the potential interactions among them. These are typically ignored when one predicts each variable separately. Moreover, building a single model for all targets can dramatically lower the computational costs.

In many cases, the data is at the same time semi-supervised (has missing target values), high dimensional and has a structured target, as for example in gene function prediction: Labeling genes with their functions is expensive (semi-supervision), the genes can be described with a large number of variables (high dimensionality), and genes can have multiple functions that are organized into a hierarchy (structured target). Thus, designing feature ranking algorithms that (1) can use unlabeled data, and (2) can handle a variety of target types, including structured ones, is a relevant task that we address in this work. To the best of our knowledge, this is the first work that treats jointly the tasks of feature ranking and semi-supervised learning in the context of predicting different types of structured outputs.

We propose two general feature ranking approaches. In the first approach, a ranking is computed from an ensemble of predictive clustering trees (Kocev et al. 2013; Blockeel 1998), which can handle structured outputs and SSL (Levatić 2017), whereas the second approach is based on the distance-based Relief family of algorithms (Kira and Rendell 1992). In a preliminary study, we investigated the performance of the ensemble-based approach for the task of classification (Petković et al. 2019). In this work, we substantially extend our previous study in several directions:

1. Additional datasets for classification are considered.
2. Additional four tasks are considered (multi-label and hierarchical multi-label classification, single- and multi-target regression), and the ensemble-based feature ranking methods are evaluated in these cases.
3. The Relief family of algorithms is extended to SSL, and evaluated for all five tasks (in comparison to the ensemble-based feature ranking methods).
4. We compare our feature ranking methods to competing methods from the literature, where such methods exist.

The remainder of the paper is organized as follows. In Sect. 2, we give the formal definitions of the different predictive modeling tasks, and introduce the notation. Section 3 surveys the related work, whereas Secs. 4 and 5 define the ensemble-based and Relief-based feature importance scores, respectively. Section 6 fully describes the experimental setup. We present and discuss the results in Sect. 7, and conclude with Sect. 8.

The implementation of the methods, as well as the results are available at http://source.ijs.si/mpetkovic/ssl-ranking.

2 Preliminaries

Basic notation. The data $\mathscr{D}$ consist of examples $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is a vector of values of D descriptive variables (features), and $\boldsymbol{y}$ is a vector of values of T target variables. The domain $\mathcal{X}_i$ of the feature x_i is either numeric, i.e., $\mathcal{X}_i \subseteq \mathbb{R}$, or categorical, i.e., it is a finite set of categorical values, e.g., $\mathcal{X}_i = \{A, B, AB, 0\}$ if the feature x_i describes blood type. Both numeric and categorical types are considered primitive *unstructured* types. The domain $\mathcal{Y}$ of the target variable depends on the predictive modeling task at hand. In this paper, we consider five tasks, two having unstructured, and three having structured target data types.

Regression (STR). In this case, the target is a single numeric variable. Since we later also consider its generalization (multi-target regression), we refer to this task as single-target regression (STR).

Multi-target regression (MTR). Here, the target variable is a vector with T numeric variables as components, i.e., $\mathcal{Y} \subseteq \mathbb{R}^T$. Equivalently, we can define MTR as having T numeric targets, hence the name. In the special case of $T = 1$, MTR boils down to STR.

Classification. In this case, the target is categorical. Since the algorithms considered in this paper can handle any classification task, we do not distinguish between binary ($|\mathcal{Y}| = 2$) and multi-class classification ($|\mathcal{Y}| > 2$).

Multi-label classification (MLC). In MLC, the target domain is the power set $\mathcal{P}(\mathscr{L})$ of some set $\mathscr{L}$ of categorical values, whose elements are typically referred to as labels. Thus, the target values are sets. Typically, the target value $\boldsymbol{y}$ of the example $(\boldsymbol{x}, \boldsymbol{y})$ is referred to as a set of labels that are relevant for this example. The sets $\boldsymbol{y}$ can be of any cardinality, thus the labels are not mutually exclusive, as is the case with the task of (standard) classification (where $|\mathcal{Y}| = 1$).

Hierarchical multi-label classification (HMLC). This is a generalization of MLC where the domain is again a power set of some label set $\mathscr{L}$, which, additionally, is now partially-ordered via some ordering $\prec$. An example hierarchy (of animal-related labels), which results from such an ordering is shown in the corresponding Haase diagram in Fig. 1.

If $\ell_1 \prec \ell_2$, the label ℓ_1 is a predecessor of the label ℓ_2. If, additionally, there is no such label ℓ, such that $\ell_1 \prec \ell \prec \ell_2$, we say that ℓ_1 is a parent of ℓ_2. If a label does not have any parents, it is called a root. A hierarchy can be either tree-shaped, i.e., every label has at most one parent, or it can be a directed acyclic graph (DAG). Since the label `elephant` has two parents (`African` and `Asian`), the hierarchy in Fig. 1 is not a tree, but rather a DAG.

Regarding predictive modeling, the ordering results in a hierarchical constraint, i.e., if a label ℓ is predicted to be relevant for a given example, then, also its predecessors must be predicted to be relevant, e.g., if a given example is `koala`, it must also be `Australian` and `animal`.

In the cases of MLC and HMLC, each set of relevant labels $S \subseteq \mathscr{L}$ is conveniently represented by the 0/1 vector $\boldsymbol{s}$ of length $|\mathscr{L}|$, whose j-th component equals one if and only if $\ell_j \in S$. Thus, we will also use the notation $T = |\mathscr{L}|$.

Semi-supervised learning (SSL). The unknown target values will be denoted by question marks (?). If the target value of the example is known, we say that the example is labeled, otherwise the example is unlabeled. This terminology applies to all types of targets and is not to be confused with the class-values being called labels in the tasks of MLC and HMLC.

3 Related work

In general, feature ranking methods are divided into three groups (Stańczyk and Jain 2015). *Filter* methods do not need any underlying predictive model to compute the ranking. *Embedded* methods compute the ranking directly from some predictive model. *Wrapper* methods are more appropriate for feature selection than feature ranking, since they directly estimate the quality of a candidate feature subset by the predictive power of the corresponding model, and this guides the selection process.

Filters are typically the fastest, but can be myopic, i.e., can neglect possible feature interactions, whereas the embedded methods are a bit slower, but can additionally serve as an explanation of the predictions of the underlying model. The importance of the task of feature ranking is reflected in numerous methods solving this task in the context of classification and STR (Guyon and Elisseeff 2003; Stańczyk and Jain 2015). However, the territory of feature ranking for SSL is mainly uncharted, especially when it comes to feature ranking in the context of structured output prediction.

An overview of SSL feature ranking methods for classification and STR is given in Sheikhpour et al. (2017). However, the vast majority of the methods described there are either supervised or unsupervised (ignoring the labels completely). Two exceptions are the SSL Laplacian score (Doquire and Verleysen 2013), applicable to STR problems (we refer to it as STR-Laplace), and the SEFR method (Bellal et al. 2012), applicable to classification problems.[1]

The SEFR method is a filter and stems from graph theory. It first converts a dataset into a graph, encoded as a weighted incidence matrix, whose weights correspond to the distances between the examples in the data. The distances are measured in the descriptive space, but more weight is put on the labeled examples. One of the drawbacks of the original method is that it can only handle numeric features. Our modification that overcomes this is described in Sect. 6.6.

For structured output prediction in SSL, we could only find competing feature ranking methods in the case of MLC. Model agnostic feature ranking (Gharroudi et al. 2016) is based on the random forest feature ranking score (Breiman 2001), computed from an ensemble of base learners. Next, a whole group of methods are based on—roughly speaking—constructing a (constrained) linear model for predicting labels and extracting feature importances from the weights in the model. The most appropriate model is found by minimizing an objective function that depends on the method. For example, this has been

[1] In our previous work (Petković et al. 2019), we show that our ensemble based method outperforms SEFR on the task of semi-supervised classification.

done in Chang et al. (2014b), where a separate model for each label is built. A similar approached has been followed in Wang et al. (2017), where similarities between examples are included into the objective function, and $l_{2,1}$ regularization is used, which would assure that the weights in the model are sparse. For computing similarities, a graph Laplacian was used, as in Chang et al. (2014a), where, additionally, a constant term was added to the linear model and Frobenius regularization was used. Similarly, the SFSS approach (Ma et al. 2012) uses a graph Laplacian, a constant term and $\ell_{2,1}$ normalization. In Alalga et al. (2016), a graph Laplacian is used directly (without an underlying predictive model). The drawback of all these graph-Laplacian-based models is that their time complexity is cubic in terms of the number of instances (due to, e.g., matrix inverse computation), which might be prohibitive for their use in practice.

All these approaches can be used also for feature selection (when a number of top-ranked features are selected). However, some approaches can be used *only* for feature selection, e.g., Li et al. (2010), where features are recursively eliminated with the help of MLC-kNN classifier.

Note that none of the authors of the above methods provide a publicly available implementation of their methods. Therefore, we have implemented the SFSS (Ma et al. 2012), MLC-Laplace (Alalga et al. 2016) and STR-Laplace (Doquire and Verleysen 2013) methods and used them in comparison with the feature ranking scores that we propose here. Our ensemble-based scores belong to the group of embedded methods, and crucially depend on ensembles of SSL predictive clustering trees (PCTs) (Levatić 2017). We thus describe below SSL PCTs and ensembles thereof.

3.1 Predictive clustering trees

PCTs are a generalization of standard decision trees. They can handle various structured output prediction tasks and have been recently adapted to SSL (Levatić 2017). This work considers the SSL of PCTs for classification, STR, MTR (Levatić et al. 2018), MLC, and HMLC.

Algorithm 1 BestTest(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: for each test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $E$
4:     $h = |E| impu(E) - \sum_{E_i \in \mathcal{P}} |E_i| impu(E_i)$
5:     if $h > h^*$ then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

Algorithm 2 PCT(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = \mathrm{BestTest}(E)$
2: if $t^* = none$ then
3:     return $Leaf(prototype(E))$
4: else
5:     for each $E_i \in \mathcal{P}^*$ do
6:         $tree_i = \mathrm{PCT}(E_i)$
7:     return $Node(t^*, \bigcup_i \{tree_i\})$
```

The pseudo-code of the algorithm for learning PCTs is given in Algorithms 1 and 2. For each of the predictive modeling tasks, one has to specify the impurity function *impu* that is used in the best test search (Alg. 1), and the prototype function *prototype* that creates the predictions in the leaf nodes (Alg. 2). After the two functions are specified, a PCT is induced in the standard top-down-tree-induction manner.

Starting with the whole dataset $\mathscr{D}_{\mathrm{TRAIN}}$, we find the test (Alg. 2, line 1) that greedily splits the data so that the heuristic score of the test, i.e., the decrease of the impurity *impu* of the data after applying the test, is maximized. For a given test, the corresponding decrease is computed in line 4 of Alg. 1.

If no useful test is found, the algorithm creates a leaf node and computes the prediction as the prototype of the examples in the node (Alg. 2, line 3). Otherwise, an internal node $\mathscr{N}$ with the chosen test is constructed, and the PCT-induction algorithm is recursively called on the subsets in the partition of the data, defined by the test. The resulting trees become child nodes of the node $\mathscr{N}$ (Alg 2, line 7).

The impurity functions for a given subset $E \subseteq \mathscr{D}_{\mathrm{TRAIN}}$ in the considered tasks are defined as weighted averages of the feature impurities $impu(E, x_i)$, and target impurities $impu(E, y_j)$.

For nominal variables z, the impurity is defined in terms of the Gini Index $Gini(E, z) = 1 - \sum_v p_E^2(v)$, where the sum goes over the possible values v of the variable z, and $p_E(v)$ is the relative frequency of the value v in the subset E. In order not to favor any variable a priori, the impurity is defined as the normalized Gini value, i.e., $impu(E, z) = Gini(E, z)/Gini(\mathscr{D}_{\mathrm{TRAIN}}, z)$. This applies to nominal features and the target in classification.

For numeric variables z, the impurity is defined in terms of their variance $Var(E, z)$, i.e., $impu(E, z) = Var(E, z)/Var(\mathscr{D}_{\mathrm{TRAIN}}, z)$. This applies to numeric features and targets in other predictive modeling tasks, since the sets in MLC and HMLC are also represented by 0/1 vectors, and the labels are treated as real-value targets. However, note that computing the Gini-index of a binary variable is equivalent to computing the variance of this variable if the two values are mapped to 0 and 1. When computing the single-variable impurities, missing values are ignored.

In a fully-supervised scenario, the impurity of data is measured only on the target side. However, (the majority of) target values may be missing in the semi-supervised case. Therefore, for SSL, also the features are taken into account when calculating the impurity, which is defined as

$$impu(E) = w \cdot \frac{1}{T} \sum_{j=1}^{T} \alpha_j impu(E, y_j) + (1 - w) \cdot \frac{1}{D} \sum_{i=1}^{D} \beta_i impu(E, x_i), \tag{1}$$

where the level of supervision is controlled by the user-defined parameter $w \in [0, 1]$. Setting it to 1 means fully-supervised tree-induction (and consequently ignoring unlabeled data). The other extreme, i.e., $w = 0$, corresponds to fully-unsupervised tree-induction (also known as clustering). The dimensional weights α_j and β_i are typically all set to 1, except for HMLC where $\alpha_i = 1$ for the roots of the hierarchy, and $\alpha_i = \alpha \cdot mean$(parent weights) otherwise, where $\alpha \in (0, 1)$ is a user-defined parameter. A MLC problem is considered a HMLC problem where all labels are roots.

The prototype function returns the majority class in the classification case, and the per-component mean $[\bar{y}_1, \ldots, \bar{y}_T]$ of target vectors otherwise. In all cases, the prototypes (predictions) are computed from the training examples in a given node/leaf. In the cases of MLC and HMLC, the values $\bar{y}_j$ can be additionally thresholded to obtain the actual subsets, i.e., $\hat{y} = \{\ell_j \mid \bar{y}_j \geq \vartheta, 1 \leq j \leq T\}$, where taking $\vartheta = 0.5$ corresponds to computing the majority values of each label.

3.2 Ensemble methods

To obtain a better predictive model, more than one tree can be grown, for a given dataset, which results in an ensemble of trees. The predictions of an ensemble are the averaged predictions of the trees (or, in general, arbitrary base models) in the ensemble. However, a necessary condition for an ensemble to outperform its base models is, that the base models are diverse (Hansen and Salamon 1990). Some randomization must thus be introduced into tree-induction, and three ways to do so have been used (Levatić 2017).

Bagging. Instead of growing one tree using $\mathcal{D}_{\text{TRAIN}}$, a bootstrap replicate of $\mathcal{D}_{\text{TRAIN}}$ is independently created for each tree, and used for tree induction.

Random Forests (RFs). In addition to the bootstrap replication mechanism of Bagging, RFs use an additional mechanism of randomization. For each internal node of a given tree, RFs consider only a random subset (of size $D' < D$) of all features when searching for the best test, e.g., $D' = ceil(\sqrt{D})$.

Extremely Randomized PCTs (ETs). As in Random Forests, a subset of features can be considered in every internal node (this is not a necessity), but additionally, only one test per feature is randomly chosen and evaluated. In contrast to Random Forests (and Bagging), ETs did not originally use bootstrapping (Geurts et al. 2006). However, Petković et al. (2019) show that it is beneficial to use bootstrapping with ETs when the features are (mostly) binary: Otherwise, each feature offers only one possible split and choosing one split (out of one) at random does not have the desired effect.

4 Ensemble-based feature ranking

Once an ensemble (for a given predictive modeling task) has been built, we come to the main focus of this work: computing a feature ranking out of it. This can be done by using any of three feature ranking scores (Petković et al. 2020): Symbolic, Genie3 or Random Forest score. Note that the basic versions of the Genie3 and Random Forest scores for classification and regression had been proposed earlier in Huynh-Thu et al. (2010) and Breiman (2001), respectively. The exact (generalized) definitions of the scores are given below:

$$importance_{\mathrm{SYMB}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} |E(\mathcal{N})| / |\mathcal{D}_{\mathrm{TRAIN}}|, \tag{2}$$

$$importance_{\mathrm{GENIE3}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} h^*(\mathcal{N}), \tag{3}$$

$$importance_{\mathrm{RF}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \frac{e(\mathrm{OOB}_{\mathcal{T}}^i) - e(\mathrm{OOB}_{\mathcal{T}})}{e(\mathrm{OOB}_{\mathcal{T}})}. \tag{4}$$

The three proposed importance scores can be all computed from a single PCT. To stabilize the scores, they are rather computed from an ensemble. Since the trees in an ensemble are grown independently, the variance of each score $importance(x_i)$ decreases linearly with the number of trees.

Here, $\mathcal{E}$ is an ensemble of trees $\mathcal{T}$, $\mathcal{T}(x_i)$ is the set of the internal nodes $\mathcal{N}$ of a tree $\mathcal{T}$ where the feature x_i appears in the test, $E(\mathcal{N}) \subseteq \mathcal{D}_{\mathrm{TRAIN}}$ is the set of examples that reach the node $\mathcal{N}$, and h^* is the heuristic value of the chosen test. $e(\mathrm{OOB}_{\mathcal{T}})$ is the value of the error measure e, obtained when using $\mathcal{T}$ as a predictive model for the set $\mathrm{OOB}_{\mathcal{T}}$ of out-of-bag examples for a tree $\mathcal{T}$, i.e., examples that were not chosen into the bootstrap replicate, and thus not seen during the induction of $\mathcal{T}$. Similarly, $e(\mathrm{OOB}_{\mathcal{T}}{}^i)$ is the value of the error measure e obtained when using $\mathcal{T}$ on $\mathrm{OOB}_{\mathcal{T}}{}^i$, i.e., on $\mathrm{OOB}_{\mathcal{T}}$ with randomly permuted values of the feature x_i. For each feature, a new random permutation is generated, using the uniform distribution.

The Symbolic and Genie3 rankings take into account node statistics: the Symbolic score's award is proportional to the number of examples that reach the node, while Genie3 also takes into account the heuristic value of the test for the node, which is proportional to $|E(\mathcal{N})|$ (see Alg. 1, line 4).

The Random Forest score, on the other hand, measures to what extent noising, i.e., permuting, the feature values decreases the predictive performance of the tree. In Eq. (4), it is assumed that e is a loss function, i.e., lower e values are better, as is the case, for example, in regression where (relative root) mean squared errors are used. Otherwise, e.g., for classification tasks and the F_1 measure, the importance of a feature is defined as $-importance_{\mathrm{RF}}$ from Eq. (4). Originally, it was designed to explain the predictions of the RFs ensemble (Breiman 2001) (hence the name), but it can be used with any predictive model. However, it is especially appropriate for trees, because their predictions can be computed fast, provided the trees are balanced.

4.1 Ensemble-based ranking for SSL in structured output prediction

The PCT ensemble-based feature ranking methods for different structured output prediction (SOP) tasks were introduced by Petković et al. (2020), Petković et al. (2020) and evaluated for different SL SOP tasks. In the SL case, PCTs use a heuristic based on impurity reduction on the target space, as defined by Eq. (1), in the special case when $w = 1$. As for SSL, the general case of Eq. (1) applies.

Having implemented SSL PCTs and ensembles of SSL PCTs, the SSL feature ranking methods then follow a similar design as for the supervised counterparts. The feature ranking methods have been evaluated in the case of SSL classification. However, they have not as yet been evaluated for STR and SOP tasks, which is the topic of this paper.

4.2 Does the ensemble method matter?

From Eqs. (2)–(4), it is evident that all three feature ranking scores can in theory be computed from a single tree, and averaging them over the trees in the ensemble only gives a more stable estimate of $\mathbb{E}[importance(x_i)]$. However, one might expect that bagging, RFs and ETs on average yield the same order of features (or even the same importance values) since the latter two are more randomized versions of the bagging method. Here, we sketch a proof that this is not (necessarily) the case.

One of the cases when the expected orders of features are equal is a dataset where each of the two binary features x_1 and x_2 completely determine the target y, e.g., $y = x_1$ and $y = 1 - x_2$, and the third feature x_3 is effectively random noise. It is clear that the expected values of the importances are in all cases $importance(x_1) = importance(x_2) > importance(x_3)$.

One of the cases where bagging gives rankings different from those of RFs, is a dataset where knowing the values of any of the feature pairs (x_1, x_2) and (x_3, x_i), for $4 \le i \le D$ again completely reconstructs the target value y, and $h(x_1) > h(x_i) > \max\{h(x_2), h(x_3)\}$, for $i \ge 4$. Additionally, we assume that $D = 100$, so many pairs (x_3, x_i) determine the target values.

When using bagging, the feature x_1 would be chosen in the root node of the trees, since $h(x_1) = \max_i h(x_i)$. In the remaining two internal nodes of the trees x_2 would be chosen. Therefore, x_1 and x_2 would be the most important features. On the other hand, when using RFs with $D' = 1$, in the majority of the cases, one of the features x_i, $i \ge 4$, would be chosen in the root node. Then, sooner or later, x_3 would be chosen. Unlike in the bagging-based ranking, x_3 is now more important than x_1.

4.3 Time complexity

In predictive clustering, the attributes in the data belong to three (not mutually exclusive) categories: (1) descriptive attributes are those that can appear in the tests of internal nodes of a tree, (2) target attributes are those for which predictions in the leaf nodes of a tree are made, and (3) clustering attributes are those that are used in computing the heuristic when evaluating the candidate tests. Let their numbers be D, T and C, respectively, and let M be the number of examples in $\mathscr{D}_{\mathrm{TRAIN}}$. Let D' be the number of descriptive attributes that are randomly chosen by random forests at each node. Note that in the SSL scenario (if $w \notin \{0, 1\}$), we have the relation $C = D + T$. Assuming that the trees are balanced, we can deduce that growing a single semi-supervised tree takes $\mathcal{O}(MD' \log M(\log M + C))$ (Levatić 2017).

After growing a tree, ranking scores are updated in $\mathcal{O}(M)$ time (since $\mathcal{O}(M)$ is the number of internal nodes in the tree) for the Symbolic and Genie3 score, whereas updating the Random Forest scores takes $\mathcal{O}(DM \log M)$. Thus, computing the feature ranking scores does not change the $\mathcal{O}$-complexity of growing a tree, and we can compute all the rankings from a single ensemble. Growing an ensemble $\mathcal{E}$ and computing the rankings takes $\mathcal{O}(|\mathcal{E}|MD' \log M(\log M + C))$.

5 Relief-based feature ranking

The Relief family of feature ranking algorithms does not use any predictive model. Its members can handle various predictive modeling tasks, including classification (Kira and Rendell 1992), regression (Kononenko and Robnik-Šikonja 2003), MTR (Petković et al. 2020), MLC (Petković et al. 2018; Reyes et al. 2015), and HMLC (Petković et al. 2020). The main intuition behind Relief is the following: the feature x_i is relevant if the differences in the target space between two neighboring examples are notable if and only if the differences in the feature values of x_i between these two examples are notable.

5.1 Supervised relief

More precisely, if $\boldsymbol{r} = (\boldsymbol{x}^1, \boldsymbol{y}^1) \in \mathscr{D}_{\text{TRAIN}}$ is randomly chosen, and $\boldsymbol{n} = (\boldsymbol{x}^2, \boldsymbol{y}^2)$ is one of its nearest k neighbors, then the computed importances $importance_{\text{Relief}}(x_i)$ of the Relief algorithms equal the estimated value of

$$P_1 - P_2 = P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \mid \boldsymbol{y}^1 \neq \boldsymbol{y}^2) - P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \mid \boldsymbol{y}^1 = \boldsymbol{y}^2), \tag{5}$$

where the probabilities are modeled by the distances between $\boldsymbol{r}$ and $\boldsymbol{n}$ in the appropriate subspaces. For the descriptive space $\mathcal{X}$ spanned by the domains $\mathcal{X}_i$ of the features x_i, we have

$$d_{\mathcal{X}}(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{1}{D}\sum_{i=1}^{D} d_i(\boldsymbol{x}^1, \boldsymbol{x}^2);\ \ d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) = \begin{cases} \mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2] & : \mathcal{X}_i \nsubseteq \mathbb{R} \\ \frac{|x_i^1 - x_i^2|}{\max_i \boldsymbol{x}_i - \min_i \boldsymbol{x}_i} & : \mathcal{X}_i \subseteq \mathbb{R} \end{cases} \tag{6}$$

where $\mathbf{1}$ denotes the indicator function. The definition of the target space distance $d_{\mathcal{Y}}$ depends on the target domain. In the cases of classification and regression, the categorical and numeric part of the definition d_i in Eq. (6) apply, respectively. Similarly, in multi-target regression, $d_{\mathcal{Y}}$ is the analogue of $d_{\mathcal{X}}$ above.

In the cases of MLC and HMLC, we have more than one option for the target distance definition (Petković et al. 2018), but in order to be as consistent as possible with the STR and MTR cases, we use the Hamming distance between label sets. Recalling that sets $S \subseteq \mathscr{L}$ are presented as 0/1 vectors $\boldsymbol{s}$ (Sect. 2), the Hamming distance $d_{\mathcal{Y}}$ is defined as

$$d_{\mathcal{Y}}(S^1, S^2) = \gamma \sum_{i=1}^{|\mathscr{L}|} \alpha_i \mathbf{1}[\boldsymbol{s}_i^1 \neq \boldsymbol{s}_i^2] \tag{7}$$

where the weights α_i are based on the hierarchy and are defined as in Eq. (1), and γ is the normalization factor that assures that $d_{\mathcal{Y}}$ maps to [0, 1]: $\gamma = \frac{1}{|\mathscr{L}|}$ in the MLC case, while its value depends on the data in the HMLC case (Petković et al. 2020).

To estimate the conditional probabilities $P_{1,2}$ from Eq. (5), they are first expressed in the unconditional form, e.g., $P_1 = P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \wedge \boldsymbol{y}^1 \neq \boldsymbol{y}^2)/P(\boldsymbol{y}^1 \neq \boldsymbol{y}^2)$. Then, the numerator is modeled as the product $d_i d_{\mathcal{Y}}$, whereas the nominator is modeled as $d_{\mathcal{Y}}$. The probability P_2 is estimated analogously.

5.2 Semi-supervised relief

In the SSL version of the above tasks, we have to resort to the predictive clustering paradigm, using descriptive and clustering attributes instead of descriptive and target ones. More precisely, the descriptive distance is defined as above. As for the clustering distance, it equals $d_{\mathcal{Y}}$ when the target value of both $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ are known, and equals $d_{\mathcal{X}}$ otherwise. The contribution of each pair to the estimate of probabilities is weighted according to their distance to the labeled data. The exact description of the algorithm is given in Alg. 3.

Algorithm 3 SSL-Relief($\mathscr{D}_{\text{TRAIN}}$, m, k, $[w_0, w_1]$)

```
1: $\mathbf{imp}$ = zero list of length $D$
2: $\boldsymbol{P}_{\text{diffAttr, diffCluster}}$, $\boldsymbol{P}_{\text{diffAttr}}$ = zero lists of length $D$
3: $P_{\text{diffCluster}} = 0.0$
4: $\boldsymbol{w}$ = $\textit{computeInstanceInfluence}(\mathscr{D}_{\text{TRAIN}}, w_0, w_1)$
5: $s = 0$                # sum of weights of the pairs, used in normalization
6: for iteration $= 1, 2, \ldots, m$ do
7:     $\boldsymbol{r}$ = random example from $\mathscr{D}$
8:     $\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_k$ = $k$ nearest neighbors of $\boldsymbol{r}$
9:     for $\ell = 1, 2, \ldots, k$ do
10:        $w = \boldsymbol{w}[\boldsymbol{r}] \cdot \boldsymbol{w}[\boldsymbol{n}_\ell]$
11:        $s \mathrel{+}= w$
12:        if $\boldsymbol{r}$ and $\boldsymbol{n}_\ell$ are labeled then
13:            $d_{\text{cluster}} = d_{\mathcal{Y}}(\boldsymbol{r}, \boldsymbol{n}_\ell)$
14:        else
15:            $d_{\text{cluster}} = d_{\mathcal{X}}(\boldsymbol{r}, \boldsymbol{n}_\ell)$
16:        $P_{\text{diffCluster}} \mathrel{+}= w\, d_{\text{cluster}}(\boldsymbol{r}, \boldsymbol{n}_\ell)$
17:        for $i = 1, 2, \ldots, D$ do
18:            $\boldsymbol{P}_{\text{diffAttr}}[i] \mathrel{+}= w\, d_i(\boldsymbol{r}, \boldsymbol{n}_\ell)$
19:            $\boldsymbol{P}_{\text{diffAttr, diffCluster}}[i] \mathrel{+}= w\, d_i(\boldsymbol{r}, \boldsymbol{n}_\ell)\, d_{\text{cluster}}(\boldsymbol{r}, \boldsymbol{n}_\ell)$
20: for $i = 1, 2, \ldots, D$ do
21:     $\mathbf{imp}[i] = \frac{\boldsymbol{P}_{\text{diffAttr, diffCluster}}[i]}{P_{\text{diffCluster}}} - \frac{\boldsymbol{P}_{\text{diffAttr}}[i] - \boldsymbol{P}_{\text{diffAttr, diffCluster}}[i]}{s - P_{\text{diffCluster}}}$
22: return imp
```

SSL-Relief takes as input the standard parameters ($\mathscr{D}_{\text{TRAIN}}$, the number of iterations m, and the number of Relief neighbors k), as well as the interval $[w_0, w_1] \subseteq [0, 1]$, which the influence levels of $\boldsymbol{r}$-$\boldsymbol{n}$ pairs are computed from (line 4): First, for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathscr{D}_{\text{TRAIN}}$, we find the distance $d_{\boldsymbol{x}}$ to its nearest labeled neighbor. If $d = 0$, i.e., the value $\boldsymbol{y}$ is known, the influence w of this example is set to 1. Otherwise, the influence of the example is defined by a linear function $d \mapsto w(d)$ that goes through the points $(\max_{(\boldsymbol{x},?)} d_{\boldsymbol{x}}, w_0)$ and $(\min_{(\boldsymbol{x},?)} d_{\boldsymbol{x}}, w_1)$. Thus, the standard regression version of Relief is obtained when no target values are missing.

5.3 Time complexity

The actual implementation of SSL-Relief does not follow the Alg. 3 word for word. It first computes all nearest neighbors to speed up the computation. This takes $\mathcal{O}(mMD)$ steps, since the majority of the steps in this stage is needed for computing the distances in the descriptive space. We use the brute-force method, because it is, for the data used in the experiments (see Sect. 6.2), still more efficient than, for example, k-D trees. Since the number of iterations is typically set to be a proportion of M (in our case $m = M$), the number of steps is $\mathcal{O}(M^2D)$. When computing the instances' influence (line 4) only the nearest neighbor of every instance is needed, so this can be done after the K-nearest neighbors are computed, within a negligible number of steps.

In the second stage, the probability estimates are computed and the worst-case time complexity is achieved when all examples are labeled, since this is the case when we have to additionally compute $d_{\mathcal{Y}}$ (otherwise, we use the stored distances $d_{\mathcal{X}}$). The number of steps needed for a single computation of $d_{\mathcal{Y}}$ depends on the domain: $\mathcal{O}(1)$ suffices for classification and STR, whereas $\mathcal{O}(T)$ steps are required in the MTR, MLC and HMLC cases.

The estimate updates themselves take $\mathcal{O}(D)$ steps per neighbor, thus, the worst case time complexity is $\mathcal{O}(M^2D + kM(T + D)) = \mathcal{O}(M^2D + kMC)$ where $C = D + T$ is (again) the number of clustering attributes.

6 Experimental setup

In this section, we experimentally evaluate the proposed feature ranking methods. We first pose a set of experimental questions below. We then describe in detail how the experimental evaluation is carried out to answer them.

6.1 Experimental questions

The evaluation is based on the following experimental questions:

1. For a given ensemble-based feature ranking score, which ensemble method is the most appropriate?
2. Are there any qualitative differences between the semi-supervised and supervised feature rankings?
3. Can the use of unlabeled data improve feature ranking?
4. Which feature ranking algorithm performs best?

6.2 Datasets

All datasets used in the evaluation are well-known benchmark problems that come from different domains. For classification, we use the datasets from Petković et al. (2019) and include five new datasets. The latter are listed as the last five in Table 1 (below the splitting line).

MLC can be seen as a special case of HMLC with a trivial hierarchy of depth 1. We thus show the basic characteristics of the considered MLC and HMLC problems in a single table (Table 2), separating the MLC and HMLC datasets by a line. Similarly, the regression problems (for STR and MTR) are shown in Table 3.

The given characteristics of the data differ from tasks to task, but the last column of every table (CH) always gives an estimate of how well the clustering hypothesis (Assumption 1) holds. For all predictive modeling tasks, this estimate is based on k-means clustering (Arthur and Vassilvitskii 2007) or, more precisely, on the agreement between the distribution of the target values in these clusters. The number of clusters was set to the number of classes in the case of classification, and to 8 otherwise, i.e., the default scikit-learn's (Pedregosa et al. 2011) parameters are kept. The k-means algorithm is run five times and the highest agreement across the five runs is reported.

CH computation. In the case of classification, the measure at hand is the Adjusted Random Index (Hubert and Arabie 1985) (ARI). It computes the agreement between the classes that examples are assigned via clustering, and the actual class values. The optimal value of ARI is 1, whereas the value 0 corresponds to the case when clustering is independent of class distribution.

In the other cases, we compute the variance of each target variable. These are the original target variables in the STR and MTR case. For MLC and HMLC case, the label sets are represented as 0/1 vectors and we compute the variance of their components. Let $\mathcal{C}$ be

Table 1 Basic properties of the classification datasets: number of examples $|\mathcal{D}|$, number of features D, number of classes (the y-domain size $|\mathcal{Y}|$), the proportion of examples in the majority class (MC), and the CH value

Dataset	$\|\mathcal{D}\|$	D	$\|\mathcal{Y}\|$	MC	CH
Arrhythmia (Lichman, 2013)	452	279	16	0.54	0.02
Bank (Lichman, 2013; Moro et al., 2011)	4521	16	2	0.88	0.00
Chess (Lichman, 2013)	3196	36	2	0.52	0.22
Dis (Gijsbers, 2017)	3772	28	2	0.98	0.00
Gasdrift (Lichman, 2013)	13,910	128	6	0.22	0.02
Pageblocks (Lichman, 2013)	5473	10	5	0.90	0.03
Phishing (Lichman, 2013)	11,055	30	2	0.56	0.00
Tic-tac-toe (Lichman, 2013)	958	9	2	0.65	0.70
Aapc (Džeroski et al., 1997)	335	84	3	0.47	0.34
Coil2000 (Van Der Putten and Van Someren, 2004)	9822	85	2	0.94	0.00
Digits (Xu et al., 1992)	1797	64	10	0.10	0.00
Pgp (Levatić et al., 2013)	932	183	2	0.52	0.00
Thyroid (Lichman, 2013)	3772	27	2	0.94	0.01

the set of the obtained clusters, i.e., $c \subseteq \mathcal{D}$, for each cluster $c \in \mathcal{C}$. Then, for every target variable y_j, we compute $v_j = \sum_c p(c)Var(c, y_j)/Var(\mathcal{D}, y)$, i.e., the relative decrease of the variance after the clustering is applied, where $p(c) = |c|/|\mathcal{D}|$. It can be proved (using the standard formula for the estimation of sample variance and some algebraic manipulation) that $v_j \leq 1$. Trivially, $v_j \geq 0$. We average the contributions v_j over the target variables to obtain the score v. In the case of HMLC, we use weighted average where the weights are proportional to the hierarchical weights α_i, defined in Sect. 3.1. Finally, the tables report the values of $CH = 1 - v \in [0, 1]$, to make the value 1 optimal.

Table 2 Basic properties of the MLC (above the line) and HMLC (below the line) datasets: number of examples $|\mathcal{D}|$, number of features D, number of labels $|\mathcal{L}|$, label cardinality (average number of labels per example) ℓ_c, the depth of the hierarchy, and the CH value

Dataset	$\|\mathcal{D}\|$	D	$\|\mathcal{L}\|$	ℓ_c	Depth	Shape	CH
Bibtex (Katakis et al., 2008)	7395	1836	159	2.4	1	tree	0.02
Birds (Briggs et al., 2013)	645	260	19	1.0	1	tree	0.05
Emotions (Trochidis et al., 2008)	593	72	6	1.9	1	tree	0.04
Genbase (Diplaris et al., 2005)	662	1185	27	1.3	1	tree	0.26
Medical (Pestian et al., 2007)	978	1449	45	1.3	1	tree	0.04
Scene (Boutell et al., 2004)	2407	294	6	1.1	1	tree	0.21
Clef07a-is (Dimitrovski et al., 2008)	11,006	80	96	3.0	3.0	tree	0.05
Ecogen (Chen et al., 2004)	1893	138	56	15.5	3.0	tree	0.03
Enron-corr (Klimt and Yang, 2004)	1648	1001	67	5.3	3.0	tree	0.03
Expr-yeast-FUN (Clare, 2003)	3788	552	594	8.9	4.0	tree	0.00
Gasch1-yeast-FUN (Clare, 2003)	3773	173	594	8.9	4.0	tree	0.01
Pheno-yeast-FUN (Clare, 2003)	1592	69	594	9.1	4.0	tree	0.00

Table 3 Basic properties of the STR and MTR datasets: number of examples $|\mathscr{D}|$, number of features D, number of targets T, and the CH value

Dataset	$\|\mathscr{D}\|$	D	T	CH
CHEMBL2850 (Gijsbers, 2017)	1211	1024	1	0.09
CHEMBL2973 (Gijsbers, 2017)	1521	1024	1	0.18
Mortgage sps5 (sps5)	1049	15	1	0.57
Pol sps5 (sps5)	5000	26	1	0.12
QSAR (Gijsbers, 2017)	2145	1024	1	0.20
Treasury sps5 (sps5)	1049	15	1	0.54
Atp1d (Spyromitros-Xioufis et al., 2016)	337	411	6	0.49
CollembolaV2 (Kampichler et al., 2000)	393	47	3	0.02
Edm1 (Karalič and Bratko, 1997)	154	16	2	0.23
Forestry-LIDAR-IRS (Stojanova, 2009)	2730	28	2	0.19
Oes10 (Spyromitros-Xioufis et al., 2016)	403	298	16	0.63
Scm20d (Spyromitros-Xioufis et al., 2016)	8966	61	16	0.16
Soil-quality (Demšar et al., 2006)	1944	142	3	0.07

6.3 Parameter instantiation

We parametrize the used methods as follows. The number of trees in the ensembles was set to 100 (Kocev et al. 2013). The number of features that are considered in each internal node was set to $\sqrt{D}$ for RFs and D for ETs (Geurts et al. 2006). The optimal value of the level of supervision parameter w for computing the ensembles of SSL PCTs was selected by internal 4-fold cross-validation on the training data set that contains both labelled and unlabelled data. For selecting the optimal value, only the labelled data and the evaluation metrics from Sect. 6.5 were used. The considered values for w were $w \in \{0, 0.1, 0.2, \dots, 0.9, 1\}$.

The amount of supervision in SSL-Relief is adaptive, which allows for a coarser set of values, and we consider $w_{1,2} \in \{0, 0.25, 0.5, 0.75, 1\}$ (where $w_1 \leq w_2$). The considered numbers k of Relief neighbors were $k \in \{15, 20, 30\}$, and the best hyper-parameter setting option (the values of w_1, w_2, and k) was again chosen via internal 4-fold cross-validation. Since more is better when the number of iterations m in Relief is concerned, this parameter was set to $m = |\mathscr{D}|$.

The possible numbers of labeled examples L in the training datasets were $L \in \{50, 100, 200, 350, 500\}$ (Levatić 2017).

6.4 Evaluation pipeline

For the tasks of MLC and HMLC, the data come with predefined training and test parts ($\mathscr{D}_{\text{TRAIN}}$ and $\mathscr{D}_{\text{TEST}}$). This is not the case for the tasks of classification, STR and MTR, where 10-fold cross validation is performed. To obtain the training-test pairs in cross-validation, we follow the procedure used by Petković et al. (2019), as shown in Fig. 2.

Each dataset $\mathscr{D}$ is randomly split into $x = 10$ folds which results in the test sets $\mathscr{D}_{\text{TEST}i}$, $0 \leq i < x$. In contrast to cross-validation in the SL scenario, where $\mathscr{D}_{\text{TRAIN}i} = \cup_{j \neq i} \mathscr{D}_{\text{TEST}j}$, we first define the copy $\mathscr{D}_{\text{TEST}_i^L}$ of $\mathscr{D}_{\text{TEST}i}$ in which we keep the target values for $\lfloor L/(x-1) \rfloor + r_i$ randomly selected examples (orange parts of columns in Fig. 2) and remove the others (white parts). Here, $\lfloor \cdot \rfloor$ is the floor function, r is the reminder of L when

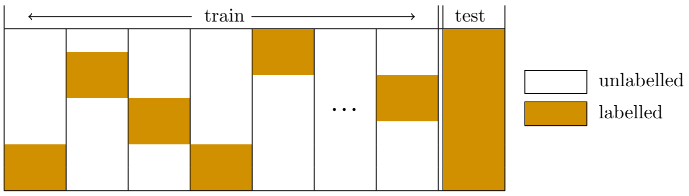


Fig. 2 Training and test set creation in SSL cross-validation: in the test fold, all examples keep their labels, whereas the folds that form the training set, together contain (approximately) L labeled examples

divided by $x - 1$, and $r_i = 1$ if $i < r$ and 0 otherwise. This assures that every training set $\mathscr{D}_{\text{TRAIN}_i}^{L} = \cup_{j \neq i} \mathscr{D}_{\text{TEST}_i}^{L}$ contains a number of labeled examples as close as possible to L.

For the MLC and HMLC data, we can choose L labeled instances from the training set and delete the target values for the others. This is done for different numbers L of labeled examples, and we make sure that the implication $L_1 \leq L_2 \Rightarrow$ *labeled examples of* $\mathscr{D}_{\text{TRAIN}_i}^{L_1}$ *are a subset of the labeled examples in* $\mathscr{D}_{\text{TRAIN}_i}^{L_2}$ holds.

The ranking evaluation proceeds as follows. First, SSL-ranking is computed from $\mathscr{D}_{\text{TRAIN}_i}^{L}$ and its SL counterpart is computed on the $\mathscr{D}_{\text{TRAIN}_i}^{L}$ with the unlabeled examples removed. Afterward, both rankings are evaluated on $\mathscr{D}_{\text{TEST}_i}^{L}$ (in the cases of MLC and HMLC, $\mathscr{D}_{\text{TRAIN}}{}^{L}$ and $\mathscr{D}_{\text{TEST}}{}^{L}$ are used).

This is done by using the kNN algorithm with $k \in \{20, 40\}$ where a weighted version of the standard squared Euclidean distance is used. For two input vectors $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, the distance d between them is defined as $d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sum_{i=1}^{D} w_i d_i^2(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2)$, where d_i is defined as in Eq. (6). The dimensional weights w_i are defined as $w_i = \max\{importance(x_i), 0\}$, since Random Forest and Relief ranking can award a feature a negative score. In the degenerated case where the resulting w_i values all equal 0 we define $w_i = 1$, for all features x_i. The first step is necessary to ignore the features that are of lower importance than a randomly generated one would be. The second step is necessary to ensure d is well-defined. We chose more than one value of k to show the qualitative differences between the supervised and semi-supervised feature rankings.

The evaluation through kNN was chosen because of three main reasons. First it can be used for all the considered predictive modeling tasks. Second, this is a distance based method, hence, it can easily make use of the information contained in the feature importances in the learning phase. Third, kNN is simple: its only parameter is the number of neighbors. In the prediction stage, the neighbors' contributions to the predicted value are equally weighted, so we do not introduce additional parameters that would influence the performance.

6.5 Evaluation measures

To asses the predictive performance of a kNN model, the following evaluation measures are used: F_1 for classification (macro-averaged for multi-class problems), Root Relative Squared Error (RRMSE) for STR and MTR, and area under the average precision-recall curve for MLC and HMLC ($\text{AU}\overline{\text{PRC}}$). Their definitions are given in Table 4. In the cross-validation setting, we average the scores over the folds (taking test set sizes into account).

For each ranking and dataset, we construct a curve that consists of points $(L, performance_L)$, where L ranges across the versions of the dataset with different amounts

Table 4 Evaluation measures, for different predictive modeling tasks

Task	Measure	Definition
Classification	F_1	$2/(1/p + 1/r)$
MLC, HMLC	$\mathrm{AU}\overline{\mathrm{PRC}}$	Area under average precision-recall curve, where precision and recall are micro-averaged across labels (Vens et al. 2008)
STR, MTR	RRMSE	$\frac{1}{T}\sum_{j=1}^{T}\sqrt{\frac{1}{\lvert\mathscr{D}_{\mathrm{TEST}}\rvert}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathscr{D}_{\mathrm{TEST}}}\frac{(\hat{\boldsymbol{y}}_j-\boldsymbol{y}_j)^2}{Var(\mathscr{D}_{\mathrm{TEST}},\boldsymbol{y}_j)}}$

The F_1 measure and $\mathrm{AU}\overline{\mathrm{PRC}}$ are defined in terms of precision $p = tp/(tp + fp)$ and recall $r = tp/(tp + fn)$, where the numbers tp, fp and fn denote the number of true positive, false positive and false negative examples, respectively

of labeled data. The comparison of two methods is then based either (1) on these curves directly (see Figs. 3 and 4), or (2) on the area under the computed curves.

6.6 The considered methods

The methods, that our proposed methods are compared to, depend on the predictive modeling task considered:

- Classification: we have shown (Petković et al. 2019) (by comparing them to competitors) that ensemble-based SSL ranking algorithms have state-of-the-art performance (better than that of the SEFR method). Thus, only their versions and the Relief's SSL and SL versions are compared here.
- STR and MTR: as mentioned before (Sect. 3), the existing SSL ranking state-of-the-art competitor for STR is the STR-Laplace method. Thus, we compare STR-Laplace (for STR only), and the SL/SSL versions of both ensemble-based rankings and Relief-based rankings.
- MLC and HMLC: for MLC, an existing competitors are SFSS and MLC-Laplace, which we consider here. To the best of our knowledge, there are no existing methods that can perform feature ranking in the SL/SSL HMLC scenarios. We also consider both versions of the ensemble-based and the Relief-based rankings.

Despite our best efforts, we could not obtain any existing implementation of the Laplace method for STR and MLC, so we provide ours together with the rest of the code. Also note that the ensemble-based and Relief-based methods work out of the box, i.e., no data preprocessing is necessary, whereas by design, Laplace can handle only numeric features. To overcome this issue, we extend the method by the following procedure: (1) transform the nominal features using 1-hot encoding, (2) compute the Laplace scores s_i, (3) for the originally nominal features x_i, define their score s_i as the sum of the scores of the corresponding 1-hot encoded features, and, finally, (4) define the importance scores $importance_{\mathrm{Laplace}}(x_i) = S + s - s_i$ (where S and s denote the maximum and the minimum of the scores, respectively). The last step is necessary since less is better, for the originally computed Laplace scores. The transformation $s_i \mapsto S + s - s_i$ maps S to s and vice-versa, thus, the scale remains intact. The other problem of the method are constant features (they cause 0/0 values), present, for example, in QSAR data: these had to be manually removed.

Table 5 Average ranks of the SSL and SL ensemble methods, for a fixed ensemble-based importance score and predictive modeling task

Task	Score	SSL ensemble			SL ensemble		
		RFs	ETs	Bagging	RFs	ETs	Bagging
Classification	Genie3	2.00	2.15	**1.85**	**1.69**	2.46	1.85
	Random Forest	**1.92**	2.15	1.92	2.00	2.08	**1.92**
	Symbolic	**1.77**	2.23	2.00	**1.77**	2.23	2.00
MLC	Genie3	**1.50**	2.67	1.83	2.17	2.17	**1.67**
	Random Forest	**2.00**	2.00	2.00	**1.67**	2.00	2.33
	Symbolic	**1.33**	2.33	2.33	2.00	2.50	**1.50**
HMLC	Genie3	**1.67**	2.00	2.33	**1.67**	1.83	2.50
	Random Forest	2.17	**1.83**	2.00	1.83	**1.67**	2.50
	Symbolic	2.17	2.00	**1.83**	**1.67**	2.00	2.33
STR	Genie3	**1.67**	2.00	2.33	2.17	2.00	**1.83**
	Random Forest	2.00	**1.83**	2.17	2.67	**1.67**	1.67
	Symbolic	2.17	**1.50**	2.33	2.33	**1.83**	1.83
MTR	Genie3	2.14	**1.86**	2.00	2.43	**1.71**	1.86
	Random Forest	2.29	2.14	**1.57**	2.43	**1.71**	1.86
	Symbolic	**2.00**	2.00	2.00	2.29	**1.71**	2.00

The best ranks are shown in bold, unless all three methods perform equally well. In the case of ties, the most efficient method's rank is shown in bold (see Table 6)

7 Results

Unless stated otherwise, the rankings are compared in terms of the areas under the performance curves (see Sect. 6.5) of the *k*NN classifier using the corresponding importance scores as weights. When an SSL-ranking is compared to a SL-ranking, and the difference Δ between the two performances is computed, $\Delta > 0$ always corresponds to the SSL-ranking performing better.

7.1 The optimal ensemble method for ensemble-based ranking

We first determine the most appropriate ensemble method, for each of the three ensemble scores, and their two versions (SSL and SL). The results in Table 5 give the average ranks of the ensemble methods in each setting, in terms of the areas under the performance curves (as described above).

We observe that for both regression tasks (STR and MTR), RFs ensembles almost never perform best (with the exception of Genie3 SSL-rankings), whereas for the (other three) classification tasks, they quite consistently outperform the other two ensemble methods. The differences among the average ranks are typically not considerable (except for the most of the MLC rankings, and supervised MTR rankings) which is probably due to the fact that the split selection mechanisms of the considered ensemble methods are still quite similar, and the trees are fully-grown, so sooner or later, a relevant feature appears in a tree node. In the case of ties, we choose the more efficient method (see

Table 6 Average ranks of the ensemble methods, in terms of induction times

Task	RFs	ETs	Bagging
Classification	**1.00**	2.23	2.77
MLC	**1.00**	2.67	2.33
HMLC	**1.00**	2.50	2.50
STR	**1.17**	2.33	2.50
MTR	**1.29**	1.71	3.00

Table 7 Proportions of datasets where the SSL feature rankings capture more global properties of the data, as compared to supervised rankings

Task	Classification	MLC	HMLC	STR	MTR
$P[\Delta > 0]$	0.73	0.83	0.96	1.00	0.93

The differences δ_{20} and δ_{40} of the areas under the performance curves of 20NN and 40NN models are computed in such a way that $\delta > 0$ means that the SSL ranking performs better. Therefore, if $\Delta = \delta_{40} - \delta_{20} > 0$, then the SSL-ranking is more global, and $\Delta < 0$ means that the SSL ranking is more local

Table 6): RFs are always the most efficient, whereas the second place is determined by the number of possible splits per feature. For lower values (e.g., when most of the features are binary, as is the case for MLC and HMLC data), bagging is faster than ETs.

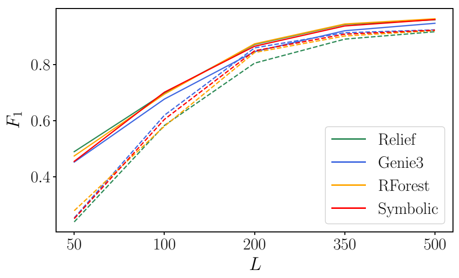
To make the graphs in the next section more readable, we plot, for every score, only the curve that corresponds to the most suitable ensemble method for this score.

7.2 Qualitative difference between SSL and SL rankings

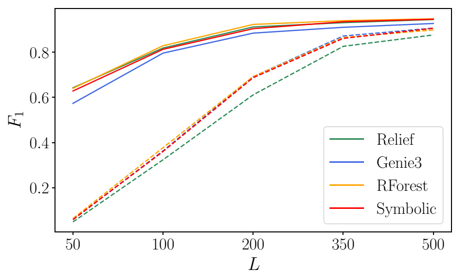
We first discuss the qualitative difference between the SSL-rankings and their supervised counterparts. In the process of obtaining a feature ranking, the SSL-version of the ranking algorithm sees more examples than its supervised version, and it turns out that this is well-reflected in the results. Figs. 3 and 4 show the results for five datasets (one dataset for each task) in terms of the performance of the rankings, as assessed by kNN models, for $k \in \{20, 40\}$. The use of these two values of k shows that SSL-rankings (solid lines) tend to capture a more global picture of the data (i.e., work better with larger values of k), whereas the supervised ones (dashed lines) reflect a more local picture (i.e., work better with smaller values of k).

This phenomenon is most visible in the performances on the two regression datasets. In the case of the `treasury` dataset, SSL-rankings perform worse than supervised ones on the local scale for smaller numbers L of labeled examples (Fig. 4a), and are equal or better for $L \geq 200$. However, on the global scale (Fig. 4b), the SSL-rankings are clear winners. A similar situation is observed for the other datasets in Figs. 3 and 4 , and also in general.

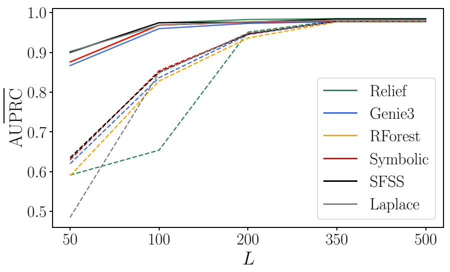
Table 7 reveals that for the vast majority of the rankings (and datasets), the SSL rankings are more global. This proportion is the highest for STR data (it even equals 100%), and is understandably the lowest for classification, where the datasets have the smallest number of examples on average.



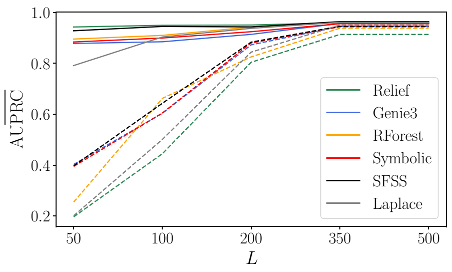
(a) classification: digits, 20NN



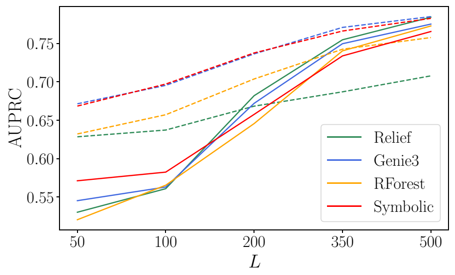
(b) classification: digits, 40NN



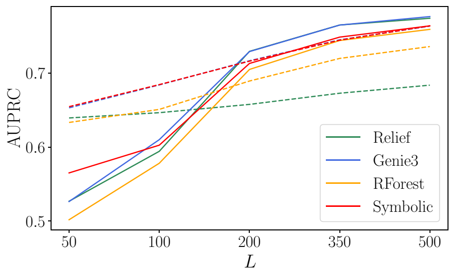
(c) MLC: genbase, 20NN



(d) MLC: genbase, 40NN



(e) HMLC: ecogen, 20NN



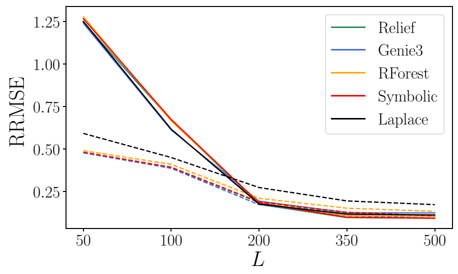
(f) HMLC: ecogen, 40NN

Fig. 3 Comparison of the SL and SSL feature rankings, for different classification tasks. The curves for an SSL ranking and the corresponding SL ranking are shown as a solid and a dashed line of the same color. The graphs in the left column use 20NN models in the evaluation, whereas those in the right column use 40NN models

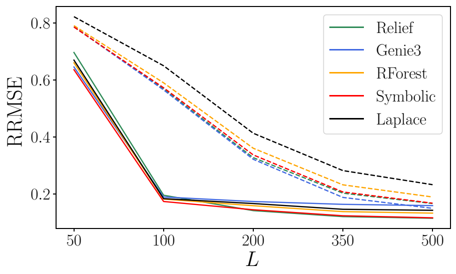
7.3 Can unlabeled data improve feature rankings?

To answer this question, we compare the SSL feature rankings to their supervised counterparts. In the previous section, we explained why sometimes the answer is not straightforward and depends on whether we consider a global or a local scale. Given that the question is whether the ranking can be improved by using unlabeled data, and given the qualitative differences between the SSL- and SL-rankings from the previous section, we fix the number of neighbors to $k = 40$.

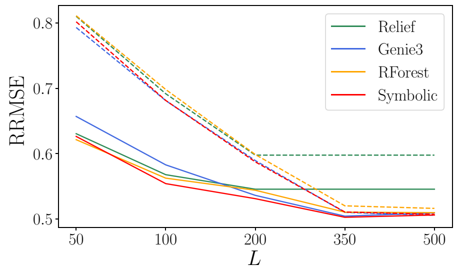
We start with the classification results given in Table 8. From the mainly positive numbers in the table, one can conclude that SSL-rankings successfully recognize the structure in the data, and outperform their supervised analogs, even in most of the cases



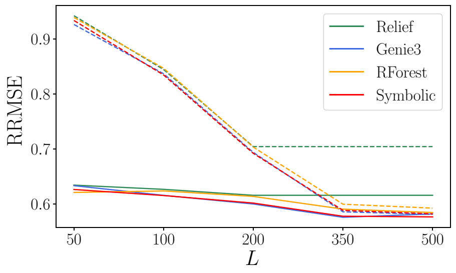
(a) STR: treasury, 20NN



(b) STR: treasury, 40NN



(c) MTR: oes10, 20NN



(d) MTR: oes10, 40NN

Fig. 4 Comparison of the SL and SSL feature rankings, for single- and multi-target regression tasks. The curves for an SSL ranking and the corresponding SL ranking are shown as a solid and a dashed line of the same color. The graphs in the left column use 20NN models in the evaluation, whereas those in the right column use 40NN models

Table 8 Classification tasks: the differences Δ in performance between SSL- and SL-rankings, as measured by areas under the curves of F_1-values of the 40NN models with ranking-induced feature weights in the distance

Classification dataset	Genie3	RForest	Symbolic	Relief	CH
Arrhythmia	0.039	0.008	0.022	0.006	0.02
Bank	0.064	0.067	0.061	0.050	0.00
Chess	− 0.084	0.021	− 0.081	0.022	0.22
Dis	0.066	0.046	0.050	0.123	0.00
Gasdrift	0.041	0.038	0.053	0.109	0.02
Pageblocks	0.272	0.250	0.250	0.243	0.03
Phishing	− 0.125	− 0.128	− 0.132	− 0.115	0.00
Tic-tac-toe	0.148	0.225	0.152	0.141	0.70
Aapc	0.115	0.041	0.067	0.110	0.34
Coil2000	0.019	0.029	0.022	0.020	0.00
Digits	0.170	0.204	0.198	0.245	0.00
Pgp	0.043	0.113	0.096	0.139	0.00
Thyroid	0.288	0.268	0.285	0.265	0.01

where the CH values are low, e.g., for the `digits` dataset in Fig. 3a, or, most notably, for `pageblocs` and `phishing`.

Continuing with the results for MLC (the upper part of Table 9), we first see that CH values are rather low, since, in contrast to the ARI values from classification, correction for chance is not incorporated into these CH values. An exception to this are the `genbase`

Table 9 The differences Δ of areas under the curves of AU$\overline{\text{PRC}}$-values of the 40NN models whose distance weights are based on SSL- and SL-rankings for MLC and HMLC (below the line) datasets

MLC/HMLC Dataset	Genie3	RForest	Symbolic	Relief	MLC Laplace	SFSS	CH
Bibtex	– 0.115	– 0.078	– 0.100	– 0.019	– 0.121	– 0.100	0.02
Birds	0.039	0.052	0.021	– 0.013	– 0.014	– 0.029	0.05
Emotions	0.012	0.028	0.011	0.041	0.033	0.044	0.04
Genbase	0.091	0.121	0.094	0.131	0.105	0.189	0.26
Medical	– 0.067	0.008	– 0.058	0.042	– 0.017	0.014	0.04
Scene	0.045	0.048	0.063	0.082	0.076	0.119	0.21
Clef07a-is	– 0.097	– 0.066	– 0.102	– 0.041			0.05
Ecogen	– 0.007	– 0.003	– 0.018	0.051			0.03
Enron-corr	– 0.068	– 0.062	– 0.023	– 0.064			0.03
Expr-yeast-fun	– 0.090	– 0.103	– 0.086	– 0.071			0.00
Gasch1-yeast-FUN	– 0.080	– 0.087	– 0.084	– 0.096			0.01
Pheno-yeast-FUN	– 0.032	– 0.031	– 0.036	– 0.029			0.00

(see Fig. 3c, d) and the `scene` dataset. For both datasets, the SSL-versions of the rankings outperform their SL-analogs. This also holds for the `birds` and `emotions` datasets, for all rankings, and additionally for the `medical` dataset in the case of Relief.

The bottom part of Table 9 gives the results for HMLC datasets. One can notice that Assumption 1 is never satisfied (low CH values), and that SSL-scores mostly could not overcome this, with the exception of Relief rankings on the `ecogen` dataset. However, inspecting the corresponding curves in detail (Fig. 3f), reveals that the negative differences between the performance of SSL-rankings and SL-rankings are mostly due to the bad start of SSL-rankings: for $L \geq 200$, the SSL-versions prevail.

We finish this section with the regression results. The upper part of Table 10 shows that when CH is well-satisfied, i.e., for the datasets `mortgage` and `treasury` (see Fig. 4b), the SSL-rankings outperform the SL-rankings. Moreover, this also holds for the `pol` data (except for the Relief rankings). Inspecting the datasets where negative values are present (most notably the `qsar` dataset) reveals the same phenomenon as in the HMLC case: for extremely low values of L, e.g., $L = 50$, the SSL-rankings do not perform well, possibly because knowing the labels of 50 out of approximately 2000 examples simply does not suffice. With more and more labels known, the performance of SSL-rankings drastically improves, while the performance of SL-rankings improves only slightly. Finally, for $L \geq 200$ or $L \geq 350$, all SSL-rankings again outperform the SL-ones.

Similar findings hold for the MTR data and the results in the bottom part of Table 10. The SSL-rankings perform well from the very beginning on the three datasets where CH holds the most, i.e., `oes10` (see Fig. 4d), `atp1d`, and `edm1`, but can only catch up with the SL-rankings (and possibly outperform them) for larger values of L in the other cases.

7.4 Which SSL-ranking performs best?

To answer this question, we compare the predictive performances of the corresponding 40NN models and report their ranks in Table 11. The results reveal that, for the majority of the tasks, ensemble-based rankings perform best. However, in some cases, the winners

Table 10 The differences Δ of areas under the curves of RRMSE-values of the 40NN models with distance weights based on SSL- and SL-rankings for STR and MTR (below the line) datasets

STR/MTR dataset	Genie3	RForest	Symbolic	Relief	STR-Laplace	CH
CHEMBL2850	– 0.047	– 0.063	0.014	– 0.092	– 0.010	0.09
CHEMBL2973	– 0.143	– 0.103	– 0.114	– 0.168	– 0.109	0.18
Mortgage	0.074	0.092	0.097	0.078	0.120	0.57
Pol	0.027	0.249	0.127	– 0.049	0.278	0.12
QSAR	– 0.347	– 0.446	– 0.442	– 0.523	– 0.262	0.20
Treasury	0.118	0.172	0.165	0.155	0.215	0.54
Atp1d	0.048	0.024	0.048	0.093		0.49
CollembolaV2	– 0.048	– 0.014	– 0.010	– 0.002		0.02
Edm1	0.002	0.018	0.004	0.006		0.23
Forestry-LIDAR-IRS	– 0.115	– 0.070	– 0.101	– 0.114		0.19
Oes10	0.083	0.084	0.083	0.122		0.63
Scm20d	– 2.357	– 2.317	– 2.295	– 2.281		0.16
Soil-quality	– 0.044	– 0.085	– 0.080	– 0.111		0.07

Table 11 The average ranks of different SSL-ranking algorithms in terms of the performance of the corresponding 40NN models

Task	Genie3	Random Forest	Symbolic	Relief	Laplace	SFSS
Classification	2.62	2.46	2.62	**2.31**		
MLC	4.17	3.00	3.67	3.33	4.33	**2.50**
HMLC	**2.00**	2.83	2.50	2.67		
STR	3.00	3.33	**1.83**	4.17	2.67	
MTR	2.57	2.57	**1.71**	3.14		

The best result (the lowest rank) in every row is shown in bold

The column Laplace refers to STR-Laplace and MLC-Laplace, as appropriate

are not clear, e.g., in the case of classification. Still, Symbolic ranking quite clearly outperforms the others on both regression tasks, STR and MTR. The best performing algorithm in MLC tasks is SFSS, followed by the Random Forest and Symbolic feature importance scores.

To complement this analysis, we also compute the average ranks of the algorithms for their induction times.

As explained in Sect. 7.1, for the ensemble-based rankings, RFs are always preferable in terms of speed. They can still be outperformed by Relief if the number of features is higher and the number of examples is moderate, which follows directly from the $\mathcal{O}$-values in Sects. 4.3 and 5.3.

The ensemble and Relief methods are implemented in the Clus system (Java), whereas our implementation of the STR and MLC Laplace scores is, as mentioned before, Python-based (scikit-learn and numpy). Thus, even though Laplace and Relief have the same core operations (finding nearest neighbors), using the highly-optimized scikit-learn's methods (such as *k*NN) puts Laplace in the first place, whereas Relief is (second but) last, for STR problems. Similarly, for the MLC feature ranking implementations, our optimized

Table 12 The average ranks of different SSL-ranking algorithms in terms of their induction times

Task	L	RFs	ETs	Bagging	Relief	Laplace	SFSS
Classification	50	**1.15**	2.46	3.15	3.23		
	500	**1.31**	2.69	3.54	2.46		
MLC	50	2.83	4.67	5.17	**1.83**	3.50	3.00
	500	2.8	4.67	5.00	**1.67**	3.67	3.17
HMLC	50	**1.17**	2.83	3.17	2.83		
	500	**1.33**	3.00	3.50	2.17		
STR	50	2.17	3.50	3.83	4.50	**1.00**	
	500	2.67	3.17	4.67	3.50	**1.00**	
MTR	50	**1.86**	2.29	3.71	2.14		
	500	**1.71**	2.43	3.71	2.14		

Since the time complexity of ensemble-based rankings (almost) equals the induction time of the ensembles, we report the latter. For each task, we show the ranks for both extreme values of L. The column Laplace refers to both STR-Laplace and MLC-Laplace, as appropriate

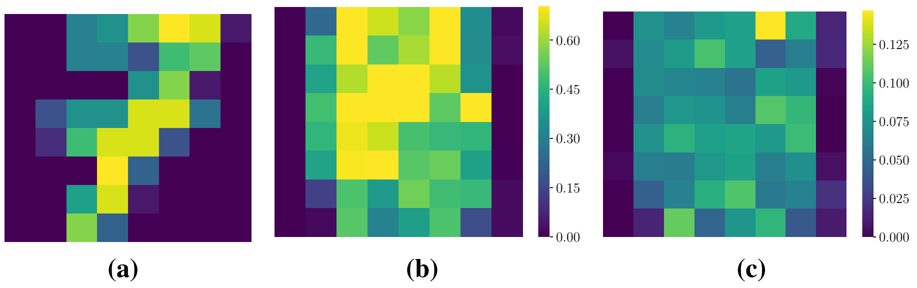


Fig. 5 A detailed analysis of an SSL Random Forest feature ranking computed from a random forest ensemble, on the digits dataset, where each feature corresponds to a pixel in an 8×8 grid: **a** an example (digit 7) from the data set; **b** average feature importance scores, and **c** their standard deviations

and vectorized Python implementations of Laplace and SFSS are faster than, e.g., ETs (Table 12).

For the dataset `digits` (see Fig. 3a, b), we additionally analyze the meaningfulness of the rankings. Namely, the feature values and feature importance scores can there be compactly shown in a 8×8 grid, representing the image of a digit. Figure 5 depicts an example from this dataset (Fig. 5a), which shows how digits are centered in the images and go from the very top to the very bottom of the images. Conversely, the first and the last column of an image are mostly blank, which is reflected well in the feature ranking in Fig. 5b. There, the average SSL Random Forest feature ranking (averaged over folds in CV) is shown and we can see that the most relevant features are in the center of the image, whereas the features in the first and the last column are irrelevant. The stability of feature ranking is measured as the standard deviation of the scores and is shown in Fig. 5c. The results for other feature ranking scores are similar.

8 Conclusions

In this work, we focus on **feature ranking in the context of semi-supervised learning**. Feature rankings are learned in the context of (single-target) classification and regression as well as in the context of structured output prediction (multi-label classification, hierarchical multi-label classification and multi-target regression). This is the first work that treats the task of feature ranking for semi-supervised structured output prediction in an uniform and comprehensive manner - it treats all the different prediction tasks in an unified way.

We propose, develop and evaluate **two approaches to SSL feature ranking for SOP**, based on tree ensembles and the Relief family of algorithms, respectively. The tree ensemble-based rankings can be learned using three ensemble learning methods (Bagging, Random Forests, Extra Trees) coupled with three scoring functions (Genie3, Symbolic and random forest scoring). The Relief-based rankings use the regression variant of the Relief algorithm for extension towards the SOP tasks. This is the first extension of the Relief algorithm towards semi-supervised learning.

An extensive **experimental evaluation of the proposed methods is carried out on 38 benchmark datasets** for the five machine learning tasks: 13 for classification, 6 for multi-label classification, 6 for hierarchical multi-label classification, 6 for regression and 7 for multi-target regression. Whenever available, we compare the performance of our newly proposed methods to the performance of existing state-of-the-art methods. Furthermore, we compare the performance of the semi-supervised feature ranking methods with the performance of their supervised counterparts.

The results from the extensive evaluation are best summarized through the answers of the research questions we have posed ourselves at the start:

1. *For a given ensemble-based feature ranking score, which ensemble method is the most appropriate?*

   Generally, Random Forests perform the best for the classification tasks (classification, muilti-label classification and hierarchical multi-label classification), while Extra-PCTs perform the best for the regression tasks (regression, multi-target regression). Furthermore, across all tasks, Random Forests are the most efficient ensemble method in terms of induction times.
2. *Are there any qualitative differences between the semi-supervised and supervised feature rankings?*

   The semi-supervised rankings tend to capture a more global picture of the data (i.e., provide better feature weights for *k*NN prediction with larger *k*-neighbourhoods), whereas the supervised ones reflect a more local view.
3. *Can the use of unlabeled data improve feature ranking?*

   When the clustering hypothesis (Assumption 1) holds—and even in some other cases—supervised feature rankings outperform their supervised counterparts. This is true for all the considered prediction tasks. However, note that the clustering hypothesis does not hold for most of the MTR and HMLC datasets.
4. *Which feature ranking algorithm performs best?*

   Different SSL feature ranking methods perform the best for the different tasks: Symbolic ranking is the best for regression and multi-target regression, SFSS and Random forest ranking for multi-label classification (with Random forest ranking scaling much better with data size), Genie3 for hierarchical multi-label classification, and Relief for classification.

We conclude the paper with several directions for further work. Due to the broad scope of the present paper, where we consider five different tasks of predictive modelling and the corresponding tasks of feature ranking, we only consider a fairly limited number of datasets for each task. We plan to extend the evaluation and comparison of different SSL feature ranking methods to additional datasets for each task, which would also enable tests of the statistical significance of the differences in performance. Moreover, we will look for datasets that are even larger in terms of number of examples, descriptive features and targets (labels) to investigate the scalability of the proposed methods. We would also like to apply the developed approaches to practically relevant tasks of SSL and/or structured output prediction (such as sentiment analysis and drug design/repurposing).

On the side of further development of SSL feature ranking methods, one direction would be to generalize the Laplace method to be able to handle all of the tasks considered here (e.g., HMLC and MTR). Another direction would consider the use of the developed feature ranking methods in the context of feature selection, where not only the relevance of individual features, but also their correlations and relative redundancy are considered. This issue is also related to investigating the stability of the feature rankings. Finally, recent work (Petković et al. 2020) has considered ensemble-based feature ranking for relational classification: extensions in the direction of SSL and structured output prediction would be worth exploring.

Acknowledgements The computational experiments presented here were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative, and we thank the administrators Barbara Krašovec and Janez Srakar for their assistance.

Author contributions MP, SD and DK designed the study, the methods and the experiments. MP implemented the methods, performed the experiments, and drafted the paper. MP and DK analyzed the results from the experiments. MP, SD and DK revised the paper.

Funding We acknowledge the financial support of the Slovenian Research Agency via the Grants P2-0103, J7-9400, J7-1815, J2-9230, and N2-0128 and a young researcher grant to MP. We also acknowledge the financial support of the European Commission through the project TAILOR - Foundations of Trustworthy AI - Integrating Reasoning, Learning and Optimization (Grant No. 952215).

Data availability The complete results are available at http://source.ijs.si/mpetkovic/ssl-ranking/. All of the used datasets are referenced and available at the cited resources.

Code availability The code is available for download at http://source.ijs.si/mpetkovic/ssl-ranking/.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Alalga, A., Benabdeslem, K., & Taleb, N. (2016). Soft-constrained Laplacian score for semi-supervised multi-label feature selection. *Knowledge and Information Systems, 47*(1), 75–98.

Arthur, D., & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms*, SODA '07 (pp. 1027–1035), USA. Society for Industrial and Applied Mathematics.

Bellal, F., Elghazel, H., & Aussem, A. (2012). A semi-supervised feature ranking method with ensemble learning. *Pattern Recognition Letters, 33*(10), 1426–1433.

Bhardwaj, K., & Patra, S. (2018). An unsupervised technique for optimal feature selection in attribute profiles for spectral-spatial classification of hyperspectral images. *ISPRS Journal of Photogrammetry and Remote Sensing, 138*, 139–150.

Bilken University. (2020). Function approximation repository. Accessible at http://funapp.cs.bilkent.edu.tr/DataSets/.

Blockeel, H. (1998). *Top-down induction of first order logical decision trees*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium.

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition, 37*(9), 1757–1771.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Briggs, F., Huang, Y., Raich, R., Eftaxias, K., Lei, Z., Cukierski, W., Frey Hadley, S., Hadley, A., Betts, M., Fern, X. Z., Irvine, J., Neal, L., Thomas, A., Fodor, G., Tsoumakas, G., Ng Hong, W., Nguyen, T. N. T., Huttunen, H., Ruusuvuori, P., ... Milakov, M. (2013). The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *IEEE international workshop on machine learning for signal processing, MLSP, 2013* (pp. 1–8).

Chang, X., Nie, F., Yang, Y., & Huang, H. (2014a). A convex formulation for semi-supervised multi-label feature selection. In *Proceedings of the twenty-eighth AAAI conference on artificial intelligence*, AAAI'14 (pp. 1171–1177). AAAI Press.

Chang, X., Shen, H., Wang, S., Liu, J., & Li, X. (2014b). Semi-supervised feature analysis for multimedia annotation by mining label correlation. In V. S. Tseng, B. T. Ho, Z.-H. Zhou, A. L. P. Chen, & H. Kao (Eds.), *Advances in knowledge discovery and data mining. Lecture notes in computer science* (pp. 74–85). Berlin: Springer.

Chen, B.-J., Chang, M.-W., & Lin, C.-J. (2004). Load forecasting using support vector machines: A study on EUNITE competition 2001. *IEEE Transactions on Power Systems, 19*(4), 1821–1830.

Clare, A. (2003). *Machine learning and data mining for yeast functional genomics*. PhD thesis, University of Wales Aberystwyth, Aberystwyth, Wales, UK.

Demšar, D., Džeroski, S., Larsen, T., Struyf, J., Axelsen, J., Bruus, M., & Krogh, P. H. (2006). Using multi-objective classification to model communities of soil microarthropods. *Ecological Modelling, 191*, 131–143.

DiMasi, J. A., Hansen, R. W., & Grabowski, H. G. (2003). The price of innovation: New estimates of drug development costs. *Journal of Health Economics, 22*(2), 151–185.

Dimitrovski, I., Kocev, D., Loskovska, S., & Džeroski, S. (2008). Hierchical annotation of medical images. In *Proceedings of the 11th international multiconference: Information Society IS 2008* (pp. 174–181). IJS, Ljubljana.

Diplaris, S., Tsoumakas, G., Mitkas, P., & Vlahavas, I. (2005). Protein classification with multiple algorithms. In *10th Panhellenic conference on informatics (PCI 2005)* (pp. 448–456).

Doquire, G., & Verleysen, M. (2013). A graph Laplacian based approach to semi-supervised feature selection for regression problems. *Neurocomputing, 121*, 5–13.

Džeroski, S., Potamias, G., Moustakis, V., & Charissis, G. (1997). Automated revision of expert rules for treating acute abdominal pain in children. In *Proceedings of the 6th conference on artificial intelligence in medicine in Europe*, AIME '97 (pp. 98–109). Berlin: Springer.

Galelli, S., Humphrey, G. B., Maier, H. R., Castelletti, A., Dandy, G. C., & Gibbs, M. S. (2014). An evaluation framework for input variable selection algorithms for environmental data-driven models. *Environmental Modelling & Software, 62*, 33–51.

Geurts, P., Erns, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning, 36*(1), 3–42.

Gharroudi, O., Elghazel, H., & Aussem, A. (2016). A semi-supervised ensemble approach for multi-label learning. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)* (pp. 1197–1204).

Gijsbers, P. (2017). Dis data. Retrieved from OpenML repository https://www.openml.org/d/40713.

Grissa, D., Pétéra, M., Brandolini, M., Napoli, A., Comte, B., & Pujos-Guillot, E. (2016). Feature selection methods for early predictive biomarker discovery using untargeted metabolomic data. *Frontiers in Molecular Biosciences, 3*, 30.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*, 993–1001.

Holzinger, A., Langs, G., Denk, H., Zatloukal, K., & Müller, H. (2019). Causability and explainability of artificial intelligence in medicine. *WIREs Data Mining and Knowledge Discovery, 9*(4), e1312.

Hoogendoorn, M., Szolovits, P., Moons, L. M., & Numans, M. E. (2016). Utilizing uncoded consultation notes from electronic medical records for predictive modeling of colorectal cancer. *Artificial Intelligence in Medicine, 69*, 53–61.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification, 2*(1), 193–218.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., & Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE, 5*(9), 1–10.

Jong, K., Mary, J., Cornuéjols, A., Marchiori, E., & Sebag, M. (2004). Ensemble feature ranking. In *PKDD-LNCS, 2302* (pp. 267–278).

Kampichler, C., Džeroski, S., & Wieland, R. (2000). Application of machine learning techniques to the analysis of soil ecological data bases: Relationships between habitat features and collembolan community characteristics. *Soil Biology and Biochemistry, 32*(2), 197–209.

Karalič, A., & Bratko, I. (1997). First order regression. *Machine Learning, 26*(2–3), 147–176.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 discovery challenge*.

Kira, K. & Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the tenth national conference on artificial intelligence, AAAI'92* (pp. 129–134). AAAI Press.

Klimt, B. & Yang, Y. (2004). The enron corpus: A new dataset for email classification research. In *ECML '04: Proceedings of the 18th European conference on machine learning—LNCS 3201* (pp. 217–226). Berlin: Springer.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition, 46*(3), 817–833.

Kononenko, I., & Robnik-Šikonja, M. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning Journal, 55*, 23–69.

Kralj Novak, P., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of emojis. *PLoS ONE, 10*, e0144296.

Levatić, J. (2017). *Semi-supervised learning for structured output prediction*. PhD thesis, Jožef Stefan Postgraduate School, Ljubljana, Slovenia.

Levatić, J., Cúrak, J., Kralj, M., Šmuc, T., Osmak, M., & Supek, F. (2013). Accurate models for p-gp drug recognition induced from a cancer cell line cytotoxicity screen. *Journal of Medicinal Chemistry, 56*(14), 5691–5708.

Levatić, J., Kocev, D., Ceci, M., & Džeroski, S. (2018). Semi-supervised trees for multi-target regression. *Information Sciences, 450*(C), 109–127.

Li, G.-Z., You, M., Ge, L., Yang, J., & Yang, M. (2010). Feature selection for semi-supervised multi-label learning with application to gene function analysis. In *Proceedings of the first ACM international conference on bioinformatics and computational biology* (pp. 354–357).

Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Ma, Z., Nie, F., Yang, Y., Uijlings, J., Sebe, N., & Hauptmann, A. (2012). Discriminating joint feature analysis for multimedia data understanding. *IEEE Transactions on Multimedia, 14*, 1662–1672.

Moro, S., Cortez, P., & Laureano, R. (2011). Using data mining for bank direct marketing: An application of the crisp-dm methodology. In *Proceedings of the European simulation and modelling conference*.

Nilsson, R., Peña, J. M., Björkegren, J., & Tegnér, J. (2007). Consistent feature selection for pattern recognition in polynomial time. *Journal of Machine Learning Research, 8*, 589–612.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Pestian, J. P., Brew, C., Matykiewicz, P., Hovermale, D. J., Johnson, N., Bretonnel Cohen, K., & Duch, W. (2007). A shared task involving multi-label classification of clinical free text. In *Proceedings* *of the workshop on BioNLP 2007: Biological, translational, and clinical language processing (BioNLP '07)* (pp. 97–104).

Petković, M., Ceci, M., Kersting, K., & Džeroski, S. (2020). Estimating the importance of relational features by using gradient boosting. In D. Helic, G. Leitner, M. Stettinger, A. Felfernig, & Z. Ras (Eds.), *International symposium on methodologies for intelligent systems* (pp. 362–371). Springer.

Petković, M., Džeroski, S., & Kocev, D. (2019). Ensemble-based feature ranking for semi-supervised classification. In P. Kralj Novak, T. Šmuc, & S. Džeroski (Eds.), *Discovery science* (pp. 290–305). Springer.

Petković, M., Džeroski, S., & Kocev, D. (2020). Feature ranking for hierarchical multi-label classification with tree ensemble methods. *Acta Polytechnica Hungarica, 17*(10), 129–148.

Petković, M., Kocev, D., & Džeroski, S. (2018). Feature ranking with relief for multi-label classification: Does distance matter? In L. Soldatova, J. Vanschoren, G. Papadopoulos, & M. Ceci (Eds.), *Discovery science* (pp. 51–65). Springer.

Petković, M., Kocev, D., & Džeroski, S. (2020). Feature ranking for multi-target regression. *Machine Learning, 109*(11), 2141–2159.

Reyes, O., Morell, C., & Ventura, S. (2015). Scalable extensions of the reliefF algorithm for weighting and selecting features on the multi-label learning context. *Neurocomputing, 161*, 168–182.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics, 23*(19), 2507–2517.

Sheikhpour, R., Sarram, M., Gharaghani, S., & Chahooki, M. (2017). A survey on semi-supervised feature selection methods. *Pattern Recognition, 64*(C), 141–158.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2016). Multi-target regression via input space expansion: Treating targets as inputs. *Machine Learning, 104*(1), 55–98.

Stańczyk, U., & Jain, L. C. (Eds.). (2015). *Feature selection for data and pattern recognition*. Studies in computational intelligence. Berlin: Springer.

Stojanova, D. (2009). Estimating forest properties from remotely sensed data by using machine learning. M.Sc. Thesis. Jožef Stefan International Postgraduate School.

Tjoa, E., & Guan, C. (2020). A survey on explainable artificial intelligence (XAI): Towards medical XAI. *IEEE Transactions on Neural Networks and Learning Systems, 4*, 5. https://doi.org/10.1109/tnnls.2020.3027314

Trochidis, K., Tsoumakas, G., Kalliris, G., & Vlahavas, I. (2008). Multilabel classification of music into emotions. In *2008 International conference on music information retrieval (ISMIR 2008)* (pp. 325–330).

Tsagris, M., Lagani, V., & Tsamardinos, I. (2018). Feature selection for high-dimensional temporal data. *BMC Bioinformatics, 19*(1), 17.

Van Der Putten, P., & Van Someren, M. (2004). A bias-variance analysis of a real world learning problem: The coil challenge 2000. *Machine Learning, 57*(1–2), 177–195.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning, 73*(2), 185–214.

Wang, X.-D., Chen, R.-C., Qun Hong, C., Qiang Zeng, Z., & Li Zhou, Z. (2017). Semi-supervised multi-label feature selection via label correlation analysis with l1-norm graph embedding. *Image and Vision Computing, 63*, 10–23.

Xu, L., Krzyzak, A., & Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics, 22*(3), 418–435.

Zhou, Y., Zhang, R., Wang, S., & Wang, F. (2018). Feature selection method based on high-resolution remote sensing images and the effect of sensitive features on classification accuracy. *Sensors, 18*(7), 2013.

Zhu, X., Goldberg, A. B., Brachman, R., & Dietterich, T. (2009). *Introduction to semi-supervised learning*. San Rafael: Morgan and Claypool Publishers.

Authors and Affiliations

Matej Petković[1,2] · Sašo Džeroski[1,2] · Dragi Kocev[1,2]

Matej Petković
matej.petkovic@ijs.si

Sašo Džeroski
saso.dzeroski@ijs.si

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

[2] Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia