Ensemble- and distance-based feature ranking for unsupervised learning

Matej Petković1,2 | Dragi Kocev1,2 | Blaž Škrlj1,2 | Sašo Džeroski1,2

1Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
2Department of Information and Communication Technologies, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

Correspondence
Matej Petković, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia. Email: matej.petkovic@ijs.si

Funding information
Javna Agencija za Raziskovalno Dejavnost RS, Grant/Award Number: P2-0103; Javna Agencija za Raziskovalno Dejavnost RS, Grant/Award Number: young researcher grant; Javna Agencija za Raziskovalno Dejavnost RS, Grant/Award Number: J7-9400; Javna Agencija za Raziskovalno Dejavnost RS, Grant/Award Number: J7-1815; Javna Agencija za Raziskovalno Dejavnost RS, Grant/Award Number: J2-9230; Javna Agencija za Raziskovalno Dejavnost RS, Grant/Award Number: N2-0128; European Commission, Grant/Award Number: 825619; European Commission, Grant/Award Number: 952215

Abstract
In this study, we propose two novel (groups of) methods for unsupervised feature ranking and selection. The first group includes feature ranking scores (Genie3 score, RandomForest score) that are computed from ensembles of predictive clustering trees. The second method is UR-ELIEF, the unsupervised extension of the Relief family of feature ranking algorithms. Using 26 benchmark data sets and 5 baselines, we show that both the Genie3 score (computed from the ensemble of extra trees) and the UR-ELIEF method outperform the existing methods and that Genie3 performs best overall, in terms of predictive power of the top-ranked features. Additionally, we analyze the influence of the hyper-parameters of the proposed methods on their performance and show that for the Genie3 score the highest quality is achieved by the most efficient parameter configuration. Finally, we propose a way of discovering the location of the features in the ranking, which are the most relevant in reality.

KEYWORDS
extra trees, feature ranking, relief, tree ensembles, unsupervised learning

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

© 2021 The Authors. International Journal of Intelligent Systems Published by Wiley Periodicals LLC
1 INTRODUCTION

High-dimensional data are becoming part of everyday life, making the learning of models much harder, slower, or even infeasible, unless some dimensionality reduction is performed. The problem (high dimensionality) may be solved by feature ranking approaches. In the supervised scenario, where the examples in the data are described as \((x, y)\) pair, with \(x\) a tuple of values of descriptive attributes (features) \(x_i\) and \(y\) a target value, the goal of feature ranking is to assign a real-valued score to each feature, which defines how relevant the feature is for predicting the target \(y\). By assessing the relevance correctly, one can (i) discard the irrelevant features and make learning predictive models faster/feasible, or (ii) understand the data better, which helps when a machine learning expert collaborates with a domain expert.

However, sometimes the examples in the data are hard to label since labeling either demands a lot of manual work (e.g., labeling tweets\(^1\)), or is expensive for some other reason (e.g., determining the activity of chemical compounds\(^2\)).

On the spectrum of major machine learning paradigms, semi-supervised learning is situated between supervised learning, which uses only labeled examples (with known values of the targets) to learn predictive models, and unsupervised learning, where only examples without labels are used, typically to group examples into clusters of similar examples. Similarity between examples is determined with respect to a given distance measure, typically Euclidean. Just as we need feature ranking in the context of supervised and semi-supervised learning, we also need it in unsupervised learning.

Since we do not have targets to predict in unsupervised learning, feature importance in this setting cannot mean how relevant the features are for predicting the targets (and building a predictive model for that purpose). Rather, feature importance in this context measures how relevant a given feature is for constructing a clustering from the given data. Other formulations and motivations of relevance in ranking in an unsupervised context have been considered, such as the one recently proposed by Doquet and Sebag\(^3\): We do not know how relevant is a given feature (for predicting a target), but we can assess how redundant it is in the context of the other features (a feature is redundant if we can predict it accurately from other features). However, we believe the relevance to constructing a clustering captures more appropriately the spirit of unsupervised feature ranking and it is this notion that is embedded within the approaches for unsupervised feature ranking that we put forward in this paper. The intuition of Doquet and Sebag\(^3\) corresponds more to feature selection, the task of finding an optimal set of features, where redundancy is to be avoided. Feature selection can be viewed as a special case of feature ranking/relevance estimation, where every feature is assigned a score of either 1 (the feature is kept) or 0 (the feature is discarded). We can obtain a feature selection by first obtaining a ranking (in terms of relevance scores) and then keeping only the features with the scores exceeding some user-defined threshold.\(^4\)

In this study, we propose two groups of novel methods for unsupervised feature ranking. Their implementation, as well as the results, are available at https://github.com/Petkomat/unsupervised_ranking. The first group consists of two ensemble-based feature ranking scores that are computed from ensembles of predictive clustering trees (PCTs) used for hierarchical clustering.\(^5\) At each internal node of a tree, the algorithm finds a feature along which the data are most efficiently divided into subclusters. In the leaves, prototypes of the clusters are computed. These prototypes can be seen as a prediction for feature values for a new-coming example, thus the features that are (more often) used in the splits, are (more) relevant for predicting the values of all the features.

The second group consists of only one member named URELIEF, that is, the unsupervised extension of the RELIEF family of algorithms.\(^6,7\) It does not use any underlying model, but rather
estimates the feature relevances via the influence of the features to the distances between the neighboring examples in the data. In other words, if the distance along a given feature strongly contributes to the overall distance between examples, the feature is deemed important. In distance-based clustering, the feature would be important for the clustering of the examples.

The rest of the paper is organized as follows. Section 2 gives a brief description of related work, in particular the baseline methods that we compare to. In Section 3, a detailed description of the proposed ensemble-based scores is given, whereas in Section 4 this is done for URELIEF. Section 5 describes the experimental setup, including the evaluation procedure. Section 6 discusses the experimental results, and Section 7 concludes the paper.

2 RELATED WORK

While the literature on supervised feature selection has been extensively reviewed and studied, the literature on the topic of unsupervised feature ranking has not been extensively and systematically surveyed. The closest overview is the recent work by Solorio-Fernández et al. that overviews methods for unsupervised feature selection. It divides the methods based on the strategy they use for selecting the features, into three groups: filter, wrapper, and hybrid. The filter methods evaluate the features based on intrinsic properties of the data, the wrapper methods evaluate the features based on the results of a specific clustering algorithm, and the hybrid methods exploit the advantages of both filter and wrapper methods with a special focus on the balance between the computational cost and the quality of the selected features. Here, we briefly summarize the most prominent and relevant work on the topic of unsupervised feature ranking.

To begin with, the SCIKIT-FEATURE repository offers unified implementations of some unsupervised feature ranking algorithms. One of them is the Laplace score. It is based on the graph theory and the creation of the Laplace matrix of a graph, whose vertices are the examples and whose edges correspond to the level of similarity among the examples.

Next, SPEC (spectral clustering) is a unification of many existing feature ranking scores. These include the RELIEF and the Laplace scores. The actual score that the authors experiment with is thus a result of particular instantiations, such as choosing an appropriate similarity measure, for example, radial basis kernels.

Furthermore, the similarity matrix (calculated by using a radial-basis kernel) is also one of the building blocks of the MCFS algorithm. However, in addition to calculating this matrix, the algorithm proceeds to solving a generalized eigenvalue problem. It ensures the sparsity of the scores by including an L1-regularization term in the objective function.

Next, another variation of the above approaches is NDFS. Here, L2,1-regularization is applied rather than L1-regularization. The best features are those which are the most closely related to the labels of the clusters constructed within the algorithm.

The last method that we discuss here is AGNOS-S. This is the best performing variation among the unsupervised ranking methods proposed by Doquet and Sebag in their award-winning paper at the ECML 2019 conference. This is an auto-encoder-based feature ranking method, where slack variables are included in the auto-encoder (and the loss function) to ensure that more important features get higher weights. The method also estimates the intrinsic dimension of the data, determining the width of the middle layer of the auto-encoder, which the method optimizes.

We use the above approaches as baselines when evaluating the quality of the methods we introduce in this paper. In the remainder of the section, we discuss some practical issues that one might face when using these methods. First, none of the baselines can handle nominal (categorical)
features. This can be only partially solved by 1-hot encoding, since this changes the initial set of features. Second, there are some time-complexity-related issues. The algorithms Laplace, SPEC, MCFS, and NDFS have all quadratic complexity in the number of examples (this holds for the proposed URelief algorithm as well). Since most of the considered data sets (see Table 1 in Section 5.1) have a lower number of examples, this is not as problematic as having quadratic time (or space) complexity in terms of the number of features, as is the case for the NDFS algorithm. The third issue is more specific and concerns only the Agnos-S method. As mentioned above, the method relies on the quality of the estimation of the intrinsic dimension of a data set. However, it has been shown\(^\text{17}\) that the method might underestimate the dimension when the data is sparse, that is, the number of examples is low as compared to the number of features.

| Data set     | Examples | Features | Domain         |
|--------------|----------|----------|----------------|
| ALLAML       | 72       | 7129     | biology        |
| arcene       | 200      | 10,000   | mass spectrometry |
| BASEHOCK     | 1993     | 4862     | text data      |
| Carcinom     | 174      | 9182     | biology        |
| CLL-SUB-111  | 111      | 11,340   | biology        |
| COIL20       | 1440     | 1024     | face image     |
| colon        | 62       | 2000     | biology        |
| gisette      | 7000     | 5000     | digit recognition |
| GLI-85       | 85       | 22,283   | biology        |
| GLIOMA       | 50       | 4434     | biology        |
| Isolet       | 1560     | 617      | spoken letter recognition |
| leukemia     | 72       | 7070     | biology        |
| lung         | 203      | 3312     | biology        |
| lymphoma     | 96       | 4026     | biology        |
| madelon      | 2600     | 500      | artificial     |
| nci9         | 60       | 9712     | biology        |
| ORL          | 400      | 1024     | face image     |
| PCMAC        | 1943     | 3289     | text data      |
| pixraw10P    | 100      | 10,000   | face image     |
| Prostate-GE  | 102      | 5966     | biology        |
| RELATHE      | 1427     | 4322     | text data      |
| SMK-CAN-187  | 187      | 19,993   | biology        |
| TOX-171      | 171      | 5748     | biology        |
| USPS         | 9298     | 256      | hand written image |
| warpPIE10P   | 210      | 2420     | face image     |
| Yale         | 165      | 1024     | face image     |
The first two proposed feature ranking scores are embedded in ensembles of PCTs. In this section, we first describe the PCTs, proceed to ensembles thereof, and conclude with feature ranking scores.

### 3.1 Single PCTs

PCTs are a fully modular generalization of standard decision trees\(^{18}\) where three groups of attributes need to be defined.

The attributes that can be used in the tests (splits) in the internal nodes of the trees (line 2 in Algorithm 2) are descriptive attributes (or features). The attributes that are used to assess the quality of split candidates, that is, are part of the formula of impurity function \(\text{impu}\), are clustering attributes. The attributes that are predicted in the leaves, that is, returned by the prototype function, are target attributes.

When PCTs are used in predictive modeling, clustering and target attributes mostly coincide (semi-supervised learning is one of the exceptions\(^{19}\)), and there is typically only one such attribute (multi-target tasks are an exception\(^{20}\)). For example, in classification, we are predicting the value of a single nominal attribute \(y\), which is also used in impurity calculations. The remaining attributes are descriptive.

However, we use PCTs for (hierarchical) clustering. Here, the groups of descriptive, clustering, and target attributes coincide, since any of the attributes can be used in a test, all of them are taken into account when computing the impurity of clusters, and all of them are “predicted” when the prototypes of the clusters are computed in the leaves.

The exact pseudocode for inducing a PCT is given in Algorithm 1, which takes as its input a subset of examples \(E \subseteq D_{\text{TRAIN}}\), and returns a PCT.

#### Algorithm 1 PCT(\(E\))

1: \((t^*, h^*, P^*) = \text{BestTest}(E)\)

2: if \(h^* = \text{none}\) then

3: \(\text{return Leaf (prototype}(E)\)\)

4: else

5: for each \(E_i \in P^*\) do

6: \(t_{E_i} = \text{PCT}(E_i)\)

7: \(\text{return Node}(t^*, \bigcup_{i}[t_{E_i}])\)

#### Algorithm 2 BestTest(\(E\))

1: \((t^*, h^*, P^*) = (\text{none}, 0, \emptyset)\)

2: for each test \(t\) do

3: \(P = \text{partition induced by } t \text{ on } E\)

4: \(h = |E| \text{impu}(E) - \sum_{E_i \in P}|E_i| \text{impu}(E_i)\)

5: if \(h > h^*\) then

6: \((t^*, h^*, P^*) = (t, h, P)\)

7: return \((t^*, h^*, P^*)\)

In clustering (unsupervised learning), the impurity function for a given subset \(E \subseteq D_{\text{TRAIN}}\) is defined as the average of \(\text{impu}(E) = \frac{1}{n} \sum_{i=1}^{n} \text{impu}(E, x_i)\) of the clustering-attribute impurities \(\text{impu}(E, x_i)\), which are defined as follows.

For nominal variables \(x\), the impurity is defined in terms of the Gini Index \(\text{Gini}(E, x) = 1 - \sum_v p_E^2(v)\), where the sum goes over the possible values \(v\) of the variable \(x\), and \(p_E(x)\) is the relative frequency of the value \(v\) in the subset \(E\). To avoid any bias toward variables with high overall impurity, the impurity is defined as the normalized Gini value, that is, \(\text{impu}(E, x) = \text{Gini}(E, x) / \text{Gini}(D_{\text{TRAIN}}, x)\).
For numeric variables \( x \), the impurity is similarly defined as the normalized variance of \( x \), that is,

\[
\text{impurity}(E, x) = \frac{\text{Var}(E, x)}{\text{Var}(\mathcal{D}_{\text{TRAIN}}, x)}.
\]

The prototype function returns a vector \([\hat{x}_1, ..., \hat{x}_n]\) where \(\hat{x}_i\) is a mean of the attribute \(x_i\) if the attribute is numeric, and the mode of the attribute, if the attribute is nominal. The means and modes are computed from the training examples in a given leaf.

### 3.2 Ensembles of PCTs

Single trees are known for their instability, that is, undesirably high variance.\(^{21}\) To overcome that, ensemble methods such as bagging,\(^{22}\) random forests,\(^{21}\) and ensembles of extremely randomized trees (extra trees)\(^{23}\) have been proposed. All of these methods have been adapted to learn ensembles of PCTs.\(^{20,24}\)

A tree ensemble is a set of trees that are grown on a possibly manipulated training set and/or a randomized tree induction algorithm. More precisely, in bagging, every tree in an ensemble is grown on an independent bootstrap replicate of the training set \(\mathcal{D}_{\text{TRAIN}}\). Random forests additionally consider only a random subset of descriptive attributes, when finding the best test (line 2 of Algorithm 2). The subset is drawn independently in every internal node.

Extra trees, as the name suggests, go even further regarding randomization and, as the random forests, first choose a subset of descriptive attributes, and then evaluate only one test per attribute. For example, if \(x_1\) is one of the chosen numeric features, random forests evaluate the tests \(x_1 \leq \theta\), for all possible values of \(\theta\), whereas extra trees additionally randomly choose only one \(\theta_i\), for each chosen feature \(x_i\).

Originally, extra trees do not use bootstrapping. However, we have included it in learning extra PCTs since (i) our preliminary experiments show that it is beneficial to use it (especially when there are many binary features in the data), and (ii) one of the feature scores described in the next section requires bootstrapping.

### 3.3 Feature ranking scores

We propose to use two feature ranking scores that can be computed from any of the ensembles described in the previous section.

The first one is Genie3,\(^{25}\) originally defined for predictive modeling tasks, such as classification or regression. Within the framework of predictive clustering, we can extend the score to unsupervised modeling as

\[
\text{importance}_{\text{Genie3}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{T \in \mathcal{E}} \sum_{N \in T(x_i)} h^*(N),
\]

where \(\mathcal{E}\) is an ensemble of trees \(T\), and \(T(x_i)\) is the set of internal nodes of the tree \(T\) where \(x_i\) is part of the test. Finally, \(h^*(N)\) is the quality of the test in the node \(N\), as assessed by the heuristic in Algorithm 2, line 4.

Note that \(h^*(N)\) is proportional to the number of examples that reach the node \(N\) (since those nodes have bigger influence), and the impurity reduction value. One could thus also use a simpler version of the Genie3 score (namely Symbolic score\(^{26}\)) that ignores the impurity reduction factor, and weights every appearance of the feature \(x_i\) by the number of examples that reach that node.
The second feature ranking score that we propose is a generalization of the RandomForest score. Instead of evaluating a feature directly via the quality of the tests, this approach compares how much introducing some noise in the feature degrades the performance of a model. More precisely, for every tree $T$ in the ensemble, we first define the set of out-of-bag examples $\text{OOB}_T$, that is, the examples that were not chosen into the bootstrap replicate of the training set. Next, we define the set $\text{OOB}_T^i$ that is obtained from the set $\text{OOB}_T$ by permuting the values of the feature $x_i$. Let $e$ be some error measure (e.g., the average of the per-feature variances in a cluster). Then, the RandomForest (RF) score is defined as

$$
\text{importance}_{\text{RF}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{T \in \mathcal{E}} \frac{e(\text{OOB}_T^i) - e(\text{OOB}_T)}{e(\text{OOB}_T)},
$$

that is, as the relative increase of the error measure $e$. The larger the increase, the more important the feature. The score has been originally defined for random forests (hence the name), but can be used with practically any ensemble approach based on bagging.

### 3.4 Time complexity

Since the sets of descriptive, clustering, and target attributes coincide with the number of all attributes, we denote their sizes by $n$. The other dimension of the data set, that is, the number of examples, is denoted by $m$.

In the analysis, we assume that the trees are approximately balanced, that is, their depth is $O(\log m)$. The largest amount of computation is performed when estimating the quality of the tests. In a given node, evaluating split candidates that a single attribute yields, requires $O(m' \log m' + m'n)$ steps (sorting the $m'$ examples that reach the node according to their attribute values) and efficiently evaluating either one (extra trees) or all thresholded tests (bagging, random forests).

Taking into account that there are $n'$ chosen features in each node (for bagging, $n' = n$), and the assumed depth of the tree, we can derive the total time complexity of growing a tree: $O((m \log m + mn)n' \log m) = O((\log m + n)n'm \log m)$. Thus, bagging is quadratic in the number of features which is a problem if trees are not grown in parallel. The latter is also the reason why we did use an unsupervised adaptation of gradient-boosted trees. One can lose the inner $\log m$ term by presorting the data with respect to all the features in advance.

Since the trees are grown independently, the algorithm is easy to parallelize, and the time complexity of growing any of the ensembles is the same.

After that, the cost of computing the Genie3 ranking is negligible, since the ensemble stores the quality of the nodes and we can traverse the nodes of a single tree in time $O(m)$. Computing the RandomForest ranking is a bit more time-consuming. We have to permute the values of a feature, and send the examples through each tree, for every feature. This is done in $O(nm \log m)$ time. This is still faster than inducing a tree.

### 4 URelief-based feature ranking

In contrast to ensemble-based feature ranking scores, the RELIEF family of feature ranking algorithms does not use any predictive model. Its members can handle various predictive modeling tasks, including classification, regression, and others.
In this study, we extend the RELIEF family to unsupervised ranking, by again taking the predictive-clustering point of view. We follow the main intuition behind RELIEF,7 which for two examples that are close to each other (in the clustering space) states: The feature $x_i$ is relevant if the differences in the target space between the two examples are notable if and only if the differences in the feature values of $x_i$ between these two examples are notable.

As in PCT induction above, we again define clustering and target space to consist of all $n$ attributes. We estimate the relevance of the descriptive attributes, which again comprise all $n$ attributes.

Being close or having a notable difference is operationalized in the algorithm via the distances in the appropriate spaces. For the clustering (target) space $\mathcal{X}$ spanned by the domains $\mathcal{X}_i$ of the features $x_i$, we have

$$d_{\mathcal{X}}(x^1, x^2) = \frac{1}{n} \sum_{i=1}^{n} d_i(x^1, x^2); \quad d_i(x^1, x^2) = \left\{ \begin{array}{ll} 1[x_i^1 \neq x_i^2] & : \mathcal{X}_i \subseteq \mathbb{R} \\ \frac{|x_i^1 - x_i^2|}{\max x_i - \min x_i} & : \mathcal{X}_i \subseteq \mathbb{R} \end{array} \right. \quad (3)$$

where $1$ denotes the indicator function. These distances are then used in the estimation of the expression

$$P\left(x_i^1 \neq x_i^2 | x^1 \neq x^2 \right) - P\left(x_i^1 \neq x_i^2 | x^1 = x^2 \right), \quad (4)$$

which is the output of the URELIEF algorithm. Above, $x^{1,2} \in \mathcal{X}$ are two examples, and $x_i^{1,2}$ values of the feature $x_i$ for these two examples. Clearly, if the features are numeric, the conditional probabilities above may be trivial or not be well defined. As explained below, they serve merely as a motivation for the algorithm.

The unsupervised version of the algorithm, URELIEF is shown in Algorithm 3. Within the code, the probability $P_{\text{diffAttr}, i} = P(x_i^1 \neq x_i^2)$ is modeled as the distance $d_i$ (line 9), whereas the probability $P_{\text{diffClus}} = P(x^1 \neq x^2)$ is modeled as the distance $d_{\mathcal{X}}$ (line 7). The conditional probabilities are estimated via the Bayes formula (line 12). The product of two events, for example, $P(x_i^1 \neq x_i^2 \land x^1 \neq x^2)$ is modeled as the product of the two probabilities: This is used together with the Bayes formula in line 12.

**Algorithm 3** URelief($D_{\text{TRAIN}}$, $I$, $K$)

1: $w$, $P_{\text{diffAttr}}$, $P_{\text{diffClus}} = 0$ lists of length $n$
2: $P_{\text{diffClus}} = 0.0$
3: for iteration =1, 2, ..., $I$ do
4: $r =$ random example from $D_{\text{TRAIN}}$
5: $n_1$, $n_2$, ..., $n_K = K$ nearest neighbors of $r$ with respect to $d_{\mathcal{X}}$
6: for $k = 1, 2, ..., K$ do
7: $P_{\text{diffClus}} + = d_{\mathcal{X}}(r, n_k)/(IK)$
8: for $i = 1, 2, ..., n$ do
9: $P_{\text{diffAttr}}[i] + = d_i(r, n_k)/(IK)$
10: $P_{\text{diffAttr}, \text{diffClus}}[i] + = d_i(r, n_k) d_{\mathcal{X}}(r, n_k)/(IK)$
11: for $i = 1, 2, ..., n$ do
12: $w_i = \frac{P_{\text{diffAttr}, \text{diffClus}}[i]}{P_{\text{diffClus}}} - \frac{P_{\text{diffAttr}}[i] - P_{\text{diffAttr}, \text{diffClus}}[i]}{1 - P_{\text{diffClus}}}$
13: return $w$
The algorithm is iterative. On every iteration, an example $r$ is chosen from the training set (line 4), and its nearest neighbors $n_k$ are computed in the clustering space. After that, the estimates for the needed probabilities are updated according to the distances between $r$ and $n_k$. The updates are normalized, so that the final estimate of every probability falls into the interval $[0, 1]$. Finally, in line 12, the weights (feature importance scores) are computed.

The time complexity of URELIEF can be estimated as follows. To make full use of the efficient vectorized implementation of scipy’s $c$dist for computing the distances among the examples, we first compute all nearest neighbors. This takes $O(mn)$ steps. Since the number of iterations is typically set to be a proportion of $m$, the number of steps is $O(m^2n)$. When updating the probability estimates, only the $K$ nearest neighbors are considered, so the total number of steps is also $O(m^2n)$, since $K$ is upper-bounded by some constant.

Thus, regarding the time complexity, URELIEF is more appropriate for high-dimensional data than ensemble-based feature rankings, since it is linear in the number of features. However, the results show that this comes at a performance price.

## 5 EXPERIMENTAL SETUP

In this section, we describe the experimental setup of the empirical evaluation of the proposed methods. We state the experimental questions and competing methods, describe the evaluation procedure, give a brief description of the data sets, and finish with the parameter instantiation of the different ranking methods.

The evaluation is based on the following experimental questions:

1. Which of the proposed ensemble scores is better?
2. Are the proposed feature ranking scores state-of-the-art (SOTA)?
3. What is the influence of the parameters of the proposed methods on the quality of the produced rankings?
4. How to locate the relevant features in a ranking?

To answer the first question, we compare the performance of the two ensemble-based ranking scores with Friedman’s (statistical) test. The evaluation procedure is discussed in Section 5.2.

To answer the second question, URELIEF, the better of the two ensemble-based scores, and five current SOTA methods are compared via Friedman’s test, together with Nemenyi’s post hoc test. The considered competitors are AGNOS-S, the Laplace score, NDFS, MCFS, and SPEC, all briefly described in Section 2.

To answer the third question, we compare the performances of the proposed scores obtained with different parameter configurations. We vary the ensemble method and the subset size for ensemble-based scores. We vary the numbers of neighbors and iterations for URELIEF.

To answer the fourth question, we construct a curve that measures the performance of the sets (of different sizes) of top-ranked features. A detailed description of the approach taken is given in Section 6.4.
5.1 | Data sets

We first planned to use all the 29 available data sets from the scikit-feature repository. To assure that the considered data sets are independent, as the statistical tests assume, we had to exclude three of them: lung_small (but lung is present), orlraws10P (but ORL is present) and warpAR10P (but warpPIE10P is present). Those included—together with the number of features and examples, are listed in Table 1 and available for download at https://github.com/jundongl/scikit-feature. Note that they typically have a low number of examples and a high number of features.

5.2 | Evaluation procedure

In the supervised learning setting, one of the popular procedures for evaluating feature ranking algorithms is the use of the weighted $k$-nearest neighbors ($k$NN) algorithm\(^\text{32}\) in which the distance among the examples employed weights features by their importance and is defined as

$$
    d(x^1, x^2) = \left( \sum_{i=1}^{n} \text{importance}(x_i) (x_i^1 - x_i^2)^p \right)^{1/p}
$$

for some $p > 0$. For example, $p = 2$ results in the weighted Euclidean distance, and the better the performance, the better are the feature-importance estimates.\(^\text{33}\) In the unsupervised scenario, choosing the correct evaluation procedure is trickier. For example, the above procedure cannot be extended to the unsupervised scenario, since we would predict the same attributes that are used for the distance computation. As a consequence, if one, for example, measures the performance of the model in terms of (root) mean squared error, it can be proven that the performance of the nearest-neighbors model is optimized when the weights $\text{importance}(x_i)$ are all equal.

Another popular (yet sometimes not appropriate procedure) for evaluating unsupervised feature rankings is to (i) use a supervised data set, (ii) compute a ranking ignoring the target variable, (iii) take some top-ranked features and learn a model that predicts the target variable. This approach only makes sense when the so-called clustering hypothesis holds,\(^\text{34}\) that is, when clusters of data examples (as computed in the descriptive space) well resemble the distribution of target values. Since the considered data sets also contain the target variable, they allow us to check this hypothesis by computing the adjusted rand index (ARI)\(^\text{35}\) between the actual class labels and the labels obtained from the $k$-means clustering, where $k$ was set to the number of classes in a given data set.

The ARI can take the values from the interval $[-1, 1]$, and is adjusted so that ARI = 0 if the two labelings are independent, that is, the clusters do not tell us anything about the target. The distribution of the ARI scores over the considered data sets is given in Figure 1 and shows that there are many data sets with ARI close to 0. Therefore, the supervised approach to evaluating unsupervised feature rankings is also inappropriate.

Therefore, we adopt another widely-used approach\(^\text{3}\) where some fixed number of features is chosen, and a predictive model is built using only this many top-ranked features. We build the models that use top 1, 2, 4, 8, 16, etc. features to analyze a given feature ranking in-depth (see Section 6.4). Therefore, we choose 16 features when comparing the qualities of feature rankings, since 16 is the closest to the 20 features that Doquet and Sebag\(^\text{3}\) use. When choosing a
model, we again follow Doquet and Sebag,3 and use a nearest-neighbors regressor. However, since some of the data sets contain a rather low number of examples (e.g., 50), we set the number of neighbors to 1.

Note that the evaluation procedure of Doquet and Sebag3 is not entirely correct, since it uses the same data for learning the ranking and evaluating it, and therefore measures only how well the ranking overfits the data. We fix this by using 10-fold cross-validation: A feature ranking is learned from a training fold, and the 1NN regressor with data from the training fold is tested on the testing fold. As the evaluation measure, we use root mean squared error: All the features in the data are numeric, since some competing methods cannot handle nominal features.

5.3 Parameter instantiation

Previous experiments with the semi-supervised version of RELIEF36 showed that the optimal number of neighbors is around 30, so we set $K = 30$ in URELIEF. Regarding the number of iterations $I$, higher values mean more accurate estimates, so we use $I = m$, $m$ being the number of examples in the data. Since the number of examples is not very high, this value of $I$ is not prohibitively large. We experiment with the other parameter configurations in Section 6.3, considering different numbers of neighbors and iterations.

For calculating the ensemble-based rankings, we set the size of the subset of chosen features to $n' = \log_2 n$ (rounded up), as suggested by Breiman.21 Moreover, this number is sufficiently low to make tree induction time-efficient. We use the extra tree ensembles (they evaluate only $n'$ tests per internal node) and induce the trees in parallel, using the Slovenian national supercomputing network. The number of trees was set to 100 and they were fully grown. We also experiment with other parameter configurations from Section 6.3, considering different subset sizes and ensemble methods.

For the baselines implemented in the SCIKIT-FEATURE repository, we use the default settings, following Doquet and Sebag.3 If a method demands the number of clusters at input (MCFS, NDFS), we set it to the number of classes in the given data set. For AGNOS-S, we use the shape of the network and the training parameters as reported by Doquet and Sebag.3
RESULTS

Below we discuss the results of the evaluation in light of our experimental questions.

6.1 Which ensemble score is better?

To answer this question, we directly compare the cross-validated RMSE performance of the 1NN regressors that use the top-ranked features, as determined by the Genie3 and Random-Forest scores calculated from extra tree ensembles. Their performance is reported in Table 2, together with the corresponding average rank: for a given data set, the better score gets rank 1, and the other gets rank 2.

Results suggest that Genie3 feature rankings are, on average, better than RandomForest ones. The differences in performance between the two methods are also statistically significant (at the significance level of 0.05): The \( p \) value from Friedman’s test is 0.0476. This is why we will use the Genie3 feature importance score in the main comparison of the methods presented below.

6.2 Are the proposed scores SOTA?

To answer this question, we compare the performance of Genie3 rankings, URELIEF rankings, and the baseline rankings (or, rather, the performance of the corresponding 1NN regressors), on different data sets. The corresponding RMSE values are reported in Table 3.

Comparing the methods we propose (Genie3, URELIEF) to their competitors in terms of average ranks, cf. Table 2, the best-performing feature-ranking method is Genie3. URELIEF also performs well, as it has the second-best average rank. To investigate whether the differences in performance among the algorithms are statistically significant, we apply Friedman’s statistical test (since we compare more than two methods). The null hypothesis is rejected at the significance level of 0.05, since the \( p \) value equals \( 9.4 \times 10^{-8} \). Thus, we can proceed to Nemenyi’s post hoc test that reveals where the differences occur. The results of these tests are presented with the average rank diagram in Figure 2.

In addition to the average ranks of the algorithms (the lower, the better), the critical distance (CD) is reported. If the average ranks of two algorithms are at least CD apart, the difference in the performance of the algorithms is considered statistically significant. The groups of algorithms, for which this does not hold, are connected by red lines.

The interpretation of the results is thus as follows. The top-performing group of algorithms, among which there are no statistically significant differences, consists of Genie3 and URELIEF. Moreover, Genie3 is statistically significantly better than all the remaining methods (i.e., those outside the top group). The other two groups are (i) URELIEF, Laplace, AGNOS-S, NDFS, and MCFS, and (ii) Laplace, AGNOS-S, NDFS, MCFS, and SPEC. Thus, the Genie3 method is the new SOTA in unsupervised feature ranking, whereas URELIEF is closer in performance to the previous SOTA methods.

6.3 Parameter setting influence on ranking performance

In the main line of experiments, we set the parameters of our feature ranking methods to the most time-efficient values, where the time complexity critically depends on them. For
ensemble-based scores, we chose extra tree ensembles and $\log_2 n$ as the number of considered features in internal nodes. As for URELIBF, we followed the previous findings since the (maximal) number of iterations is not a problem.

In this section, we investigate how the quality of the produced rankings depends on the parameters of the ranking methods. We start with URELIBF and then proceed to ensemble-based scores.
6.3.1 | URELIEF

The heat-plot in Figure 3 shows the average ranks of the feature rankings computed by using all the pairs of different values for the number of iterations $I \in \{0.1m, 0.25m, 0.5m, 0.75m, 1.0m\}$ and the number of neighbors $K \in \{5, 10, 15, 20, 30, 40\}$. We can see that more neighbors and more

| Data set       | Genie3 | URELIEF | Laplace | SPEC  | MCFS  | NDFS  | AGNOS-S |
|----------------|--------|---------|---------|-------|-------|-------|---------|
| ALLAML         | 1.15   | 1.11    | 1.16    | 1.15  | 1.16  | 1.15  | 1.22    |
| arcene         | 40.90  | 51.30   | 42.40   | 76.20 | 42.40 | 42.40 | 80.60   |
| BASEHOCK       | 0.17   | 0.18    | 0.18    | 0.18  | 0.18  | 0.18  | 0.16    |
| Carcinom       | 0.33   | 0.33    | 0.36    | 0.36  | 0.36  | 0.36  | 0.36    |
| CLL-SUB-111    | 1726.00| 1823.00 | 1735.00 | 2350.00| 2429.00| 1744.00| 2374.00 |
| COIL20         | 0.12   | 0.11    | 0.16    | 0.23  | 0.28  | 0.09  | 0.18    |
| colon          | 1.43   | 1.49    | 1.50    | 1.50  | 1.50  | 1.50  | 1.49    |
| gisette        | 269    | 278.00  | 291.00  | 315.00| 291.00| 291.00| 304.00  |
| GLI-85         | 1516.00| 1759.00 | 1521.00 | 1783.00| 1558.00| 1655.00| 1712.00 |
| GLIOMA         | 0.23   | 0.23    | 0.24    | 0.27  | 0.23  | 0.24  | 0.27    |
| Isolet         | 0.37   | 0.48    | 0.37    | 0.45  | 0.42  | 0.38  | 0.36    |
| leukemia       | 1.84   | 1.83    | 1.91    | 1.90  | 1.91  | 1.91  | 1.94    |
| lung           | 0.29   | 0.29    | 0.30    | 0.33  | 0.29  | 0.31  | 0.31    |
| lymphoma       | 1.73   | 1.81    | 2.01    | 2.01  | 2.01  | 2.01  | 1.85    |
| madelon        | 31.79  | 31.83   | 33.62   | 33.62 | 33.62 | 33.62 | 34.61   |
| nci9           | 1.62   | 1.64    | 1.60    | 1.66  | 1.60  | 1.60  | 1.76    |
| ORL            | 24.98  | 27.60   | 34.19   | 34.19 | 34.19 | 34.19 | 23.19   |
| PCMAC          | 0.18   | 0.19    | 0.16    | 0.18  | 0.18  | 0.19  | 0.16    |
| pixraw10P      | 6.65   | 7.93    | 9.30    | 8.48  | 9.30  | 9.30  | 6.48    |
| Prostate-GE    | 0.22   | 0.23    | 0.22    | 0.30  | 0.23  | 0.22  | 0.24    |
| RELATHE        | 0.22   | 0.22    | 0.23    | 0.24  | 0.23  | 0.23  | 0.32    |
| SMK-CAN-187    | 0.34   | 0.35    | 0.35    | 0.34  | 0.35  | 0.35  | 0.38    |
| TOX-171        | 206.00 | 212.00  | 207.00  | 221.00| 237.00| 219.00| 209.00  |
| USPS           | 0.33   | 0.35    | 0.33    | 0.38  | 0.40  | 0.29  | 0.34    |
| warpPIE10P     | 16.97  | 16.45   | 30.28   | 30.28 | 30.28 | 30.28 | 15.30   |
| Yale           | 46.77  | 51.29   | 56.76   | 56.76 | 56.76 | 56.76 | 42.73   |
| Avevare rank   | 1.90   | 3.42    | 4.04    | 5.25  | 4.60  | 4.46  | 4.33    |

Note: The best performance per data set is given in bold typeface. Additionally, the average rank for each of the feature ranking methods is given at the bottom.
iterations in general lead to better performance (lower ranks). Both observations are somewhat expected. More iterations mean more stable estimates of the probabilities present in Equation (4). The explanation for why the algorithm prefers a higher number of neighbors is that the number of features can be really high, thus some noise is definitely present. Therefore, having a higher number of neighbors averages out the noise at least to some extent. Note that we did not lose much by choosing $I = m$ and $K = 30$, since this is the third-best option.

6.3.2 | Ensemble-based scores

Here, we investigate the influence of the ensemble type (extra trees, random forests, bagging), and the feature subset size. Note that it is enough to consider only extra trees and random forests, since we consider the subset sizes of $n' \in \{\log_2 n, \sqrt{n}, n\}$, and the bagging approach is a special case of random forests when $n' = n$. The heat plots in Figure 4 show the performance of the rankings for both considered scores: Genie3 (Figure 4A) and Random Forest (Figure 4B).

The performance of both scores mostly increases when the subset size $n'$ decreases. A possible explanation for that is that such models overfit less to the data given that the number of features is high. It seems that this parameter is more important that the ensemble method, especially for the RandomForest score. Again, we did not lose much (performance-wise) when choosing the parameters for the main line of experiments to optimize for time complexity, avoiding the potentially quadratic number of operations in the number of features $n$.

6.4 | Where in a ranking are the relevant features?

Choosing a number of features to include in a model comes in handy when one wants to evaluate different feature rankings over different domains and many data sets. However, if a domain expert is interested in a particular data set, a more global view on the ranking might be desirable to understand the problem better. To achieve that, one can build predictive models that use the top 1, 2, 4, ..., $2^j$, ..., $2^{\lfloor \log_2 n \rfloor}$, and $n$ features and show the performance of the obtained models as a curve. Using the geometric (rather than the linear 1, 2, 3, ...) sequence of numbers of top-ranked features makes the curve construction feasible (building, e.g., more than 20,000 predictive models for the GLI-85 data set for the linear sequence might be too
time-consuming), while still showing enough details at the beginning of the ranking, which is its most interesting part.

Here, we chose two data sets, for which the quality of the rankings is completely different. Note that the results for the rest of the curves are available at https://github.com/Petkomat/unsupervised_ranking. Both figures in Figure 5 contain an additional baseline named Random. This is the average performance over 100 rankings where the orders of the features have been generated uniformly at random. Figure 5A shows the results for the lungs data set, where with the increasing number of features, the RMSE of the models mostly decreases. This means that, the rankings are mostly correct since relevant features have the highest importance. After URELIEF and Laplace both discover the most relevant feature, Genie3 quickly becomes the one (after four features) that ranks the features best.

The situation in Figure 5B is different. These are the rankings for the PCMAC data set. We can see that all the algorithms successfully discover a few most relevant features, but then, mostly useless and noisy features are placed in the ranking after them. This is evident from the increasing RMSE. Only after 32 or 64 features, the next relevant ones are positioned and RMSE decreases again.

**FIGURE 3** Average ranks (over the considered data sets) of the considered parameter configurations of URELIEF feature ranking. Lower ranks correspond to better performance [Color figure can be viewed at wileyonlinelibrary.com]

**FIGURE 4** Average ranks (across all data sets) of the considered parameter configurations (three different subset sizes and two ensemble methods), when the feature ranking score is fixed to one of two options (Genie3, RandomForest). Lower ranks correspond to better performance (A) Genie3 and (B) RandomForest [Color figure can be viewed at wileyonlinelibrary.com]
When a domain expert sees those two figures, the expert can locate the position of the relevant features, which will help him understand the problem better. Of course, when we use all the features (the last point of every curve), RMSE does not depend on the ranking and all the curves end at the same point.

**CONCLUSIONS**

In this study, we propose two novel approaches for unsupervised feature ranking. The first approach uses ensemble-based scores (Genie3 and RandomForest), computed from ensembles of PCTs. The second approach is an unsupervised version of the RELIEF algorithm (URELIEF).

After carefully choosing and discussing the evaluation procedure, we conduct an extensive empirical evaluation. We determine how the parameters of both approaches (ensemble-based scores and URELIEF) influence the quality of the rankings they produce. We show that for the ensemble-based scores, where parameters critically influence the time efficiency, the most efficient rankings fortunately also have the highest quality.

The comparative evaluation shows that the Genie3 score works better than the RandomForest score. We then compare Genie3 and URELIEF with five baselines showing that, on average, both proposed methods outperform the baselines. Since the difference in performance between Genie3 and the baselines is statistically significant, we recommend, all in all, to use the Genie3 score. This score can be efficiently computed from a parallelized ensemble of extremely randomized PCTs, where the feature subset size is set to $n' = \log_2 n$, which typically

**FIGURE 5** Error curves on two data sets (A) lung and (B) PCMAC for the feature rankings produced by the competing unsupervised ranking methods. Lower placement of a curve corresponds to better performance. The dotted line on each of the figures shows the average performance of 100 random feature rankings [Color figure can be viewed at wileyonlinelibrary.com]
results in the time complexity of $O(mn \log m \log n)$, where $m$ and $n$ are the numbers of examples and features.

**ACKNOWLEDGMENTS**

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. This study was supported by the Slovenian Research Agency via the grant P2-0103 and a young researcher grant to M. P. and B. Š. S. D. and D. K. also acknowledge the support by the Slovenian Research Agency (via grants J7-9400, J7-1815, J2-9230, and N2-0128), and the European Commission (projects AI4EU [grant number 825619] and TAILOR [grant number 952215]).

**ORCID**

Matej Petković https://orcid.org/0000-0002-0495-9046
Dragi Kocev https://orcid.org/0000-0003-0687-0878
Blaž Škrlj https://orcid.org/0000-0002-9916-8756
Sašo Džeroski https://orcid.org/0000-0003-2363-712X

**REFERENCES**

1. Kralj Novak P, Smailović J, Sluban B, Mozetič I. Sentiment of emojis. *PLOS One*. 2015;10:1-22. https://doi.org/10.1371/journal.pone.0144296
2. DiMasi JA, Hansen RW, Grabowski HG. The price of innovation: new estimates of drug development costs. *J Health Econ.* 2003;22:151-185.
3. Doquet G, Sebag MAGnostic feature selection. In: Machine Learning and Knowledge Discovery in Databases European Conference, ECML PKDD 2019, Würzburg, September 16–20, 2019, LNCS 11906. 2020:343-358.
4. Seijo-Pardo B, Bolón-Canedo V, Alonso-Betanzos A. Alonso-Betanzos, On developing an automatic threshold applied to feature selection ensembles. *Inform Fusion*. 2019;45:227-245.
5. Blockeel H, De Raedt L, Ramon J. Top-down induction of clustering trees. In: ICML ’98 Proceedings of the Fifteenth International Conference on Machine Learning; 1998:55-63.
6. Kira K. The feature selection problem: Traditional methods and a new algorithm. In: Rosenbloom P, Szolovits P, eds. *Proceedings of the Tenth National Conference on Artificial Intelligence*. Menlo Park, California: The AAAI Press; 1992:129-134.
7. Kononenko I, Robnik-Šikonja M. Theoretical and empirical analysis of ReliefF and RReliefF. *Mach Learn*. 2003;55:23-69.
8. Guyon I, Elisseeff A. An introduction to variable and feature selection. *J Mach Learn Res.* 2003;3:1157-1182.
9. Saeys Y, Inza I, Larrañaga P. A review of feature selection techniques in bioinformatics. *Bioinformatics*. 2007;21:1084-1093.
10. Stańczyk U, Jain LC. Feature selection for data and pattern recognition. In: Kacprzyk J, ed. *Studies in Computational Intelligence*. Vol. XVIII. Berlin Heidelberg: Springer-Verlag; 2015:355.
11. Solorio-Fernández S, Carrasco-Ochoa J, Martinez-Trinidad J. A review of unsupervised feature selection methods. *Artif Intell Rev.* 2020; 53: 907-948.
12. Li J, Cheng K, Wang S, Morstatter F, Trevino RP, Tang J, Liu H. Feature selection: a data perspective. *ACM Comput Surv.* 2018. https://doi.org/10.1145/3136625
13. He X, Cai D, Niyogi P. Laplacian score for feature selection. In: Lu BL, Zhang L, Kwok J, eds. *Proceedings of the 18th International Conference on Neural Information Processing Systems*. Berlin Heidelberg: Springer-Verlag; 2005:507-514.
14. Zhao Z, Liu H. Spectral feature selection for supervised and unsupervised learning. In: Ghalamani Z, ed. *Proceedings of the 24th International Conference on Machine Learning*. New York, NY: Association for Computing Machinery; 2007:1151-1157.
15. Cai D, Zhang C, He X. Unsupervised feature selection for multi-cluster data. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery; 2010:333-342.

16. Li Z, Yang Y, Liu J, Zhou X, Lu H. Unsupervised feature selection using nonnegative spectral analysis. In: Hoffmann J, Selman B, eds. Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. Palo Alto, California: The AAAI Press; 2012: 1026-1032.

17. Facco E, d’Errico M, Rodriguez A, Laio A. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. Sci Rep. 2017;7:1-8. https://doi.org/10.1038/s41598-017-11873-y

18. Breiman L, Friedman J, Olshen R, Stone CJ. Classification and Regression Trees. Boca Ranton, Florida: Chapman and Hall/CRC; 1984.

19. Levatić J. Semi-supervised learning for structured output prediction [PhD thesis]. Ljubljana: Jožef Stefan Postgraduate School; 2017.

20. Kocev D, Vens C, Struyf J, Džeroski S. Tree ensembles for predicting structured outputs. Pattern Recogn. 2013;46:817-833.

21. Breiman L. Random forests. Mach Learn. 2001;45:5-32.

22. Breiman L. Bagging predictors. Mach Learn. 1996;24:123-140.

23. Geurts P, Erns D, Wehenkel L. Extremely randomized trees. Mach Learn. 2006;36:3-42.

24. Kocev D, Ceci M, Stepišnik T. Ensembles of extremely randomized predictive clustering trees for predicting structured outputs. Mach Learn. 2020;109:2213-2241.

25. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P. Inferring regulatory networks from expression data using tree-based methods. PLOS One. 2010;5:1-10.

26. Petković M, Kocev D, Džeroski S. Feature ranking for multi-target regression. Mach Learn. 2020;109:1179-1204.

27. Friedman JH. Greedy function approximation: a gradient boosting machine. Ann Stat. 2001;29:1189-1232.

28. Petković M, Kocev D, Džeroski S. Feature ranking with relief for multilabel classification: does distance matter? In: Soldatova L, Vanschoren J, Papadopoulos G, Ceci M, eds. Discovery Science. Cham: Springer International Publishing; 2018:51-65.

29. Reyes O, Morell C, Ventura S. Scalable extensions of the ReliefF algorithm for weighting and selecting features on the multi-label learning context. Neurocomputing. 2015;161:168-182.

30. Virtanen P, Gommers R, Oliphant TE, et al. Contributors, SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nat Methods. 2020;17:261-272.

31. Demšar J. Statistical comparisons of classifiers over multiple data sets. J. Mach Learn Res. 2006;7:1-30.

32. Wetschereck D. A study of distance based algorithms [PhD thesis]. Corvallis, OR: Oregon State University; 1994.

33. Cunningham P, Delany SJ. k-nearest neighbour classifiers. Tech. Rep. UCD-CSI-2007-4, University College Dublin; 2007.

34. Zhu X, Goldberg AB, Brachman R, Dietterich T. Introduction to Semi-Supervised Learning. San Rafael, California: Morgan and Claypool Publishers; 2009.

35. Hubert L, Arabie P. Comparing partitions. J Classif. 1985;2:193-218.

36. Petković M, Džeroski S, Kocev D. Feature ranking for semi-supervised learning: 2020. https://arxiv.org/abs/2008.03937

How to cite this article: Petković M, Kocev D, Škrlj B, Džeroski S. Ensemble- and distance-based feature ranking for unsupervised learning. Int J Intell Syst. 2021;36:3068–3086. https://doi.org/10.1002/int.22390