Supplementary material: Assessing Feature Scorer Results on High-Dimensional Datasets with t-SNE

Bruno Iochins Grisci[a], Mario Inostroza-Ponta[b] and Márcio Dorn[a,c,d,*]

[a]Institute of Informatics, Federal University of Rio Grande do Sul, Av. Bento Gonçalves 9500, Porto Alegre, 91501-970, RS, Brazil
[b]Computer Engineering Department, University of Santiago de Chile, Av. Libertador Bernardo O'Higgins, Santiago, 9170022, RM, Chile
[c]Center of Biotechnology, Federal University of Rio Grande do Sul, Av. Bento Gonçalves, Porto Alegre, 91501-970, RS, Brazil
[d]National Institute of Science and Technology - Forensic Science, Porto Alegre, RS, Brazil

ARTICLE INFO

Keywords:
dimensionality reduction
data visualization
feature scoring
machine learning interpretability

ABSTRACT

While vast literature on high-dimensional data visualization is available, there are not many works regarding the visualization of feature scorers and their results. Feature scorers are algorithms that assign numerical importance to each feature of multi-dimensional datasets. These importance scores can be used in several applications, such as feature selection, knowledge discovery, and machine learning interpretability. There are several feature scorers to choose from, and often no single metric or ground truth is available to guarantee the quality of their results. In this scenario, visualization can become valuable to support the decision of which method to choose and how good its results are. For this goal, this work presents "weighted t-SNE." It modifies the relationship between data points in the embedded 2D space to reflect the importance of each dimension of the original datasets as assessed by a feature scorer. This research discusses how to implement weighted t-SNE, proposes the silhouette coefficient as a numerical evaluation of the results, and shows several examples of its use in practice. Synthetic and real-world tabular datasets are used in the experiments together with nine feature scorers, ranging from Mutual Information to neural networks. Each feature scorer produces unique visualizations, and weighted t-SNE can be used to compare and choose the one that better suits a given dataset and task. Weighted t-SNE can also visually show the importance of features learned by machine learning models and help us see how they are organizing the data, increasing their interpretability.

1. Feature scorers

In the sections below we give further detail on how each of the feature scorers used in the experiments presented in the main text compute feature importance. A reproduction of the summary of the feature scorers from the main text is reproduced in Table 1. The definitions and equations listed below come mainly from the work of Barbieri et al. [1], Grisci et al. [2], Montavon et al. [3, 4], Molnar [5].

1.1. Kruskal Wallis H test

The one-way analysis of variance (ANOVA) is a statistical technique that evaluates variation within and across groups (i.e., within a random variable and between different variables, respectively) to determine whether a set of random variables originates from the same distribution. In this study, we employ the Kruskal-Wallis H test, a non-parametric alternative to ANOVA. Unlike ANOVA, this test does not require the assumption of normally distributed data. Its mathematical definition is outlined in Equation 1.

$$H = \left(\frac{12}{n(n+1)} \sum_{j=1}^{k} \frac{R_i^2}{n_i} \right) - 3(n+1) \tag{1}$$

*Corresponding author

bigrisci@inf.ufrgs.br (B.I. Grisci); mario.inostroza@usach.cl (M. Inostroza-Ponta); mdorn@inf.ufrgs.br (M. Dorn)
https://brunogrisci.github.io/ (B.I. Grisci); https://informatica.usach.cl/academico/mario-inostroza-ponta/ (M. Inostroza-Ponta); https://sbcb.inf.ufrgs.br/ (M. Dorn)
ORCID(s): 0000-0003-4083-5881 (B.I. Grisci); 0000-0003-1295-8972 (M. Inostroza-Ponta); 0000-0001-8534-3480 (M. Dorn)

Table 1
Properties of the feature scorers used in the experiments. The type refers to how the scorer works and the correlation to how interactions between features are treated.

Scorer	Type	Correlation	Reference
Kruskal-Wallis	Filter	Univariate	[6]
Mutual Information	Filter	Univariate	[7]
mRMR	Filter	Multivariate	[8]
ReliefF	Filter	Multivariate	[9]
Lasso	Embedded	Multivariate	[10]
Decision Tree	Embedded	Multivariate	[11]
Random Forest	Ensemble	Multivariate	[12]
Linear SVM	Embedded	Multivariate	[13]
Neural Network	Embedded	Multivariate	[2]

Where k is the number of compared groups, n is the total number of samples, n_i is the number of samples in the i-th group, and R_i^2 is the sum of the values in the i-th group.

In feature scoring, the goal is to give a numerical importance value to features that most effectively distinguish the target class. To achieve this, we compute the Kruskal-Wallis H test score for each feature by grouping its values according to the target class. Features with higher test scores are then selected [14]. Since this method evaluates each attribute independently in relation to the target class, it is considered univariate and does not account for potential relationships or dependencies between features.

1.2. Mutual Information

A core concept in information theory is entropy, which quantifies the uncertainty associated with a random variable in predicting the likelihood of an event [7]. The entropy $H(x)$ of a random variable is mathematically expressed in Equation 2.

$$H(x) = -\sum_{i=1}^{n} P(x_i)\log_2(P(x_i)) \tag{2}$$

Where n is the total number of instances, x is a random variable, x_i is the i-th value in the variable x, and $P(x_i)$ is the probability of x_i occurring.

Building on the idea of entropy, Mutual Information (MI) measures the amount of information one random variable provides about another [7]. Its formal definition is provided in Equation 3.

$$I(x;y) = \sum_{i=1}^{n}\sum_{j=1}^{n} P(x_i, y_j)\cdot \log\left(\frac{P(x_i, y_j)}{P(x_i)\cdot p(y_j)}\right) \tag{3}$$

Where n is the total number of instances, x and y are random variables, x_i and y_i are the i-th value in the variable x and y, respectively, $P(x_i)$ and $P(y_i)$ are the probabilities of x_i and y_i occurring, respectively, and $P(x_i, y_i)$ is the joint probability of x_i and y_i occurring.

Using mutual information, we can assess a feature's relevance by determining how much information it shares with the target class. This allows us to rank features based on their computed relevance. Like the Kruskal-Wallis filter, this method evaluates features individually and does not account for dependencies between them, making it a univariate approach.

1.3. mRMR

In certain scenarios, we aim to evaluate the contribution of a new variable when added to an existing set of variables. This can be achieved by assessing both the relevance and non-redundancy of the new variable. The Minimum Redundancy Maximum Relevance (mRMR) method, introduced by Peng et al. [8], is an iterative feature scoring algorithm. It prioritizes features that maximize mutual information with the target class while minimizing redundancy with respect to the features already included in the selected subset. At each iteration, the algorithm seeks to identify the feature x_j that optimizes the criterion defined in Equation 4.

$$\max_{x_j \in X-S} \left[I\left(x_j; c\right) - R(x_j, S) \right] \tag{4}$$

Where X is the set of all attributes, S is the set of already selected features, I is the mutual information function, and R is the redundancy function.

1.4. ReliefF

ReliefF is a widely used algorithm for feature scoring or selection, renowned for its intuitive and efficient approach. Its key strength lies in its core concept: for each instance in the dataset, it identifies the k-nearest neighbors from each class. It then evaluates how much each attribute varies between these instances, making it a lightweight yet effective method that accounts for feature correlations [15].

The algorithm begins by initializing a weight vector that represents the relevance of each feature. It then selects an instance and searches for its k-nearest neighbors within each class. Instances belonging to the same class as the target instance are referred to as hits, while those from different classes are called misses. The underlying logic is as follows: if the value of a given attribute A differs between the target instance and a miss, A is considered a useful feature for distinguishing between the classes, and its weight is increased. Conversely, if the value of attribute A differs between the target instance and a hit, A is less effective at separating classes (since they belong to the same class), and its weight is decreased. To ensure balance, misses are weighted by the probability $P(c)$ of their class occurring in the dataset, allowing misses and hits to contribute equally. The algorithm ultimately returns a weight vector that reflects the relevance of each feature.

1.5. Lasso

The Least Absolute Shrinkage and Selection Operator, commonly referred to as Lasso, is a linear regression technique that incorporates $l1$-regularization to enhance prediction accuracy and mitigate overfitting. Its mathematical formulation involves solving the optimization problem presented in Equation 5.

$$\min_{\beta_0, \beta} \left\{ \frac{1}{N} \left\| y - \beta_0 - X\beta \right\|_2^2 + \lambda \|\beta\|_1 \right\} \tag{5}$$

Where y is the outcome, X is the independent variables, N is the number of instances in the data, β is the unknown parameters to be calculated, and λ is the regularization term. The importance of each feature is simply the coefficient of that feature in the final regressor.

1.6. Decision Tree

This model represents a decision tree, where each node, structured in a top-down fashion, corresponds to a decision path determined by specific conditions (typically based on the values of a sample's attributes during prediction). Each leaf node, in turn, represents a final label or outcome associated with that particular path. As described by Loh [16], tree generation algorithms construct the tree recursively from the top down. At each node, splits (decisions) are created by evaluating attributes using an impurity function (such as the Gini index), which assigns scores to attributes deemed discriminant. This process continues until the tree is fully formed, with each branch representing a sequence of decisions leading to a final classification or outcome. Feature importance is calculated as the (normalized) total reduction in the impurity criterion achieved by that feature, in what is commonly referred to as the Gini importance.

1.7. Random Forest

Random Forest is an ensemble learning algorithm that harnesses the principle of the "wisdom of the crowd" to reduce the variance in outcomes generated by individual classifiers. The concept of Random Forest was first introduced by Breiman [12]. It consists of a collection of decision trees, each trained on different subsets of instances created through bootstrapping (random sampling with replacement). For a given instance, predictions are made by every tree in the forest, and the final predicted label is determined through a majority voting mechanism. This approach involves aggregating the feature weights computed by each tree in the Random Forest into a single averaged feature weight vector, enabling the ranking of features based on their importance. The feature importance is computed using the Gini importance as for the decision trees.

1.8. Support Vector Machine

Support Vector Machines (SVM) are a robust and efficient class of algorithms used for both regression and classification tasks. The method works by constructing an n-dimensional hyper-plane or a set of hyper-planes that optimally separate instances by class, maximizing the margin—the distance between the hyper-planes and the nearest instances from each class [13].

To handle cases where the data is not perfectly linearly separable, SVM employs the soft-margin approach, utilizing the hinge loss function. Additionally, it incorporates Thikonov regularization, also known as $l2$-regularization, to improve generalization. The process of computing the classifier is equivalent to solving the optimization problem defined in Equation 6.

$$\left[\frac{1}{n}\sum_{i=1}^{N}\max\left(0, 1 - y_i\left(\vec{w}\cdot\vec{x}_i - b\right)\right)\right] + \lambda\|\vec{w}\|^2 \tag{6}$$

Where $\vec{w}$ is the normal vector to the hyper-plane, $\frac{b}{\vec{w}}$ is the plane offset to from origin along $\vec{w}$, n is the number of instances in the data, x_i is i-th instance in the data, and y_i is every x_i class, which can be either 1 or -1, each representing one class. For the task of feature scoring, and using a linear kernel for the SVM, the importance of each feature is the coefficient of that feature in the final model.

1.9. Neural Networks

An Artificial Neural Network (ANN) is a computational model inspired by the structure and function of the biological brain. It consists of interconnected units called neurons, organized in layers, that work together to process and learn from data. Each neuron receives input, performs a simple computation, and passes the result to the next layer. Through training, the network adjusts its connections (weights) to recognize patterns, make predictions, or classify data. Retrieving feature importance from ANNs is not a trivial task, given their "black-box" nature. However, there are many methods that try to solve this problem, such as the Layer-Wise Relevance Propagation (LRP) [17] and its adaptation for tabular data, the relevance aggregation [2].

LRP works computing a backward pass that sends the output of the nework back through its structure, using several possible rules that take into account the input domain and layer type. The two LRP rules used in this work were LRP-$\alpha\beta$ [17] in Equation 7 and w^2-rule [18] in Equation 8.

$$R_j = \sum_k \left(\alpha\frac{a_j w_{jk}^{+}}{\sum_j a_j w_{jk}^{+}} - \beta\frac{a_j w_{jk}^{-}}{\sum_j a_j w_{jk}^{-}}\right) R_k \tag{7}$$

$$R_j = \sum_k \frac{w_{jk}^2}{\sum_j w_{jk}^2} R_k \tag{8}$$

For both equations, k and j are the k-th and j-th layers, a is the output of a neuron, w^+ and w^- are positive and negative weights, α and β are constants, and R is the relevance signal. The constants follow the property that $\alpha - \beta = 1$ and $\beta \geq 0$ [17]. The w^2-rule is a special case for the input layer when any real-valued input is admissible.

For tabular data, it is adequate to use relevance aggregation. The core concept involves training a neural network on the target dataset (line 3) and then calculating the relevance of each input for every sample (line 6). The algorithm employs absolute values to ensure equal consideration of both relevance (positive values) and counter-relevance (negative values) (line 7). The computed values are rescaled to ensure that all samples contribute equally during

Algorithm 1: Relevance aggregation

```
    Data: $D_{n \times m}$: data, $c$: classes, $network$: neural network
    Result: Ordered relevance scores
 1  begin
 2      $R, S, score \leftarrow [\,]$ ;
 3      train $network$ on $D_{n \times m}$;
 4      for $sample_{1 \times m}$ in $D_{n \times m}$ do
 5          $out \leftarrow$ predict($network$, $sample_{1 \times m}$);
 6          $rel_{1 \times m} \leftarrow$ compute_relevance($network$, $sample_{1 \times m}$, $out$);
 7          $rel_{1 \times m} \leftarrow$ abs($rel_{1 \times m}$) / max(abs($rel_{1 \times m}$));
 8          $R_{sample} \leftarrow rel_{1 \times m}$;
 9      end
10      for $feat_{n \times 1}$ in $R_{n \times m}$ do
11          for $class$ in $c$ do
12              $S_{feat,class} \leftarrow$ average($feat_{n \in class}$);
13          end
14      end
15      for $feat_{1 \times c}$ in $S_{m \times c}$ do
16          $score_{feat} \leftarrow$ average($feat_{1 \times c}$);
17      end
18      return sort($score_{m \times 1}$);
19  end
```

aggregation (line 7). This step is especially crucial for regression tasks, where differences in target values could otherwise skew the relevance toward samples with higher target magnitudes.

Aggregation is performed at two levels. The first level operates by class, computing the average of the rescaled relevance for each input, but only for samples that belong to the same class (line 12). For regression tasks, the default approach is to treat all samples as part of a single class. This intermediate step of calculating aggregation scores for different classes enables the distinction of relevance scores between sample groups, recognizing that the neural network may rely on different sets of features to identify each group. The second level of aggregation involves averaging the relevance scores across all classes (line 16), producing a global relevance score that can be sorted (line 18). The final score is derived by averaging the class scores, ensuring that each class contributes equally to the overall result. The algorithm 1 is reproduced from Grisci et al. [2].

CRediT authorship contribution statement

Bruno Iochins Grisci: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review and Editing, Visualization. **Mario Inostroza-Ponta:** Methodology, Validation, Writing - Review and Editing. **Márcio Dorn:** Methodology, Validation, Resources, Writing - Review and Editing, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

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

Acknowledgment

This work was supported by grants from the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) [19/2551-0001906-8; 24/2551-0000725-3], Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) [314082/2021-2; 440279/2022-4; 408154/2022-5], and the Emerging Leaders in the Americas

Program Scholarship with the support of the Government of Canada. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] M. C. Barbieri, B. I. Grisci, M. Dorn, Analysis and comparison of feature selection methods towards performance and stability, Expert Systems with Applications (2024) 123667.
[2] B. I. Grisci, M. J. Krause, M. Dorn, Relevance aggregation for neural networks interpretability and knowledge discovery on tabular data, Information Sciences 559 (2021) 111–129.
[3] G. Montavon, W. Samek, K.-R. Müller, Methods for interpreting and understanding deep neural networks, Digital Signal Processing 73 (2018) 1–15.
[4] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, K.-R. Müller, Layer-wise relevance propagation: an overview, in: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, Springer, Switzerland, 2019, pp. 193–209.
[5] C. Molnar, Interpretable Machine Learning, 2 ed., 2022. URL: https://christophm.github.io/interpretable-ml-book.
[6] W. H. Kruskal, W. A. Wallis, Use of ranks in one-criterion variance analysis, Journal of the American statistical Association 47 (1952) 583–621.
[7] J. R. Vergara, P. A. Estévez, A review of feature selection methods based on mutual information, Neural Computing and Applications 24 (2014) 175–186.
[8] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 1226–1238.
[9] I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the myopia of inductive learning algorithms with relieff, Applied Intelligence 7 (1997) 39–55.
[10] V. Fonti, E. Belitser, Feature selection using lasso, VU Amsterdam Research Paper in Business Analytics 30 (2017) 1–25.
[11] C. J. Stone, Classification and Regression Trees, 1st ed., Taylor & Francis Group, LLC, Boca Raton, FL, 1984.
[12] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.
[13] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (1995) 273–297.
[14] B. I. Grisci, B. C. Feltes, M. Dorn, Neuroevolution as a tool for microarray gene expression pattern identification in cancer research, Journal of Biomedical Informatics 89 (2019) 122–133.
[15] M. Robnik-Šikonja, I. Kononenko, Theoretical and empirical analysis of relieff and rrelieff, Machine learning 53 (2003) 23–69.
[16] W.-Y. Loh, Classification and regression trees, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1 (2011) 14–23.
[17] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, Plos One 10 (2015) 1–46.
[18] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, K.-R. Müller, Explaining nonlinear classification decisions with deep taylor decomposition, Pattern Recognition 65 (2017) 211–222.