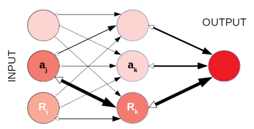
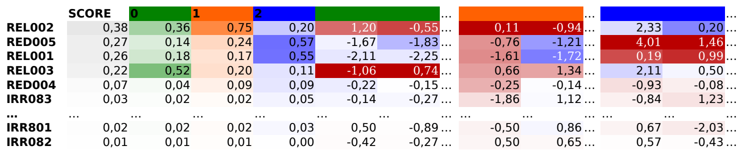
Graphical Abstract

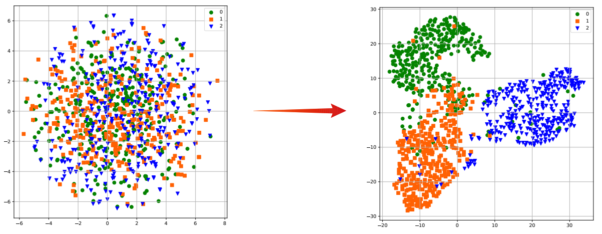
Relevance aggregation for neural networks interpretability and knowledge discovery on tabular data

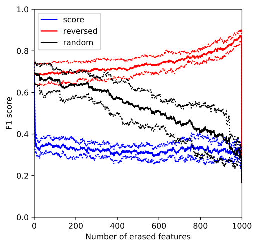
Bruno Iochins Grisci[1,2], Mathias J. Krause[2], Marcio Dorn[1,*]





	SCORE	0	1	2			...			...			...
REL002	0,38	0,36	0,75	0,20	1,20	-0,55	...	0,11	-0,94	...	2,33	0,20	...
RED005	0,27	0,14	0,24	0,57	-1,67	-1,83	...	-0,76	-1,21	...	4,01	1,46	...
REL001	0,26	0,18	0,17	0,55	-2,11	-2,25	...	-1,61	-1,72	...	0,19	0,99	...
REL003	0,22	0,52	0,20	0,11	-1,06	0,74	...	0,66	1,34	...	2,11	0,50	...
RED004	0,07	0,04	0,09	0,09	-0,22	-0,15	...	-0,25	-0,14	...	-0,93	-0,08	...
IRR083	0,03	0,02	0,02	0,05	-0,14	-0,27	...	-1,86	1,12	...	-0,84	1,23	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...
IRR801	0,02	0,02	0,02	0,03	0,50	-0,89	...	-0,50	0,86	...	0,67	-2,03	...
IRR082	0,01	0,01	0,01	0,00	-0,42	-0,27	...	0,50	0,65	...	0,57	-0,43	...





Highlights

Relevance aggregation for neural networks interpretability and knowledge discovery on tabular data

Bruno Iochins Grisci[1,2], Mathias J. Krause[2], Marcio Dorn[1,*]

- Relevance aggregation generates scores for each input feature from several samples.
- It correctly identified which features are important for the network's predictions.
- The set of relevant input features can be different from class to class.
- New methods allow the visualization of the patterns learned by the model.
- Relevance aggregation can help to identify incorrect rules or machine bias.

Relevance aggregation for neural networks interpretability and knowledge discovery on tabular data

Bruno Iochins Grisci[1,2], Mathias J. Krause[2], Marcio Dorn[1,*]

[1] *Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre - RS, 91501-970, Brazil*

[2] *Lattice Boltzmann Research Group, Institute for Applied and Numerical Mathematics, Karlsruhe Institute of Technology, Karlsruhe, D-76049, Germany*

bigrisci@inf.ufrgs.br, mathias.krause@kit.edu, mdorn@inf.ufrgs.br

[*] *Corresponding author*

Abstract

The lack of interpretability of neural networks is partially why they are not adopted in a wider variety of applications. Many works focus on explaining their predictions, but few take tabular data into consideration, which led to a small adoption even though this data is of high academic and business interest. We present relevance aggregation, an algorithm that combines the relevance computed from several samples as learned by a neural network and generates scores for each input feature. We also present two methods for visualizing the learned patterns, leading to a better model comprehension. The method was tested in synthetic and real-world datasets (breast cancer gene expression, online shopping behavior, and national high school exam) for classification and regression tasks. It correctly identified which features are the most important for the network's predictions. The selected features can be distinct for each class. The rank of features scores also matches their contribution to the model's performance. The results selected relevant features from the data, paving the way for knowledge discovery. The top-ranked features were consistently able to improve the performance of another independent classifier. For poorly trained neural networks, relevance aggregation helped identify incorrect rules or machine bias.

Keywords: Neural networks, Relevance propagation, Tabular data, Interpretable machine learning, Knowledge discovery, Feature selection

1. Introduction

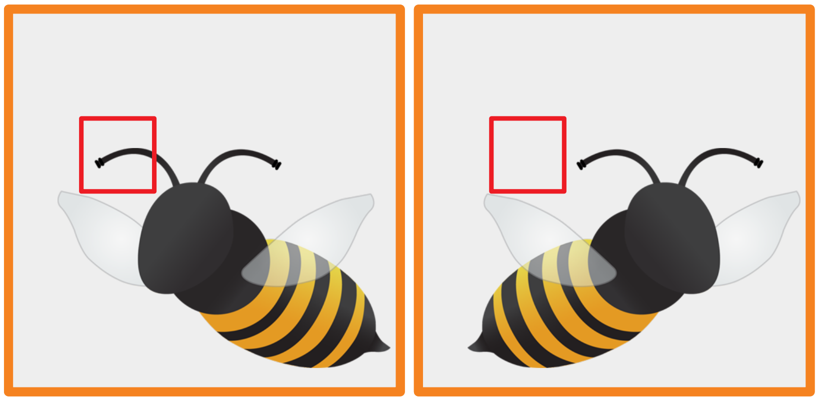
Even though neural networks have achieved state-of-the-art results in many challenging tasks and their adoption has become widespread in the past years, they are still widely regarded as "black-box" models. Their learned behaviors are hard to explain or predict due to their intrinsically complex structures. When mapping an input to output, the features learned are only implicitly described by a large number of internal model parameters [20]. The lack of a better understanding of the decisions being made can lead to underperformance, distrust, or *machine bias* [24; 29; 36]. Interpretable machine learning is the collection of algorithms and techniques that allow humans to understand the cause of a decision made by a machine learning model, including neural networks [29] and other predictors, like Support Vector Machines [12].

A great portion of recent works on the interpretability of neural networks mainly focus on image data, such as the research on neural networks "circuits" [34]. Other methods like deconvolution [48] or VisualBackProp [10] were designed with convolutional neural networks in mind. Algorithms that use surrogate models such as Local Interpretable Model-Agnostic Explanations (LIME) [37] require the definition of a neighborhood, which is not well defined for tabular data [29]. Some works are also based on the attention mechanism [49]. This work will focus on algorithms that explain single predictions and can be applied to types of neural networks that better suit tabular data. Examples are Layer-wise Relevance Propagation (LRP) [6], DeepLIFT [41], and sensitivity analysis [42]. Such algorithms will be discussed in more detail in Section 2.

We define tabular data as any data (binary, continuous, ordinal, or categorical) that is unstructured, often represented in a table format (samples as rows and dimensions as columns) [29]. By "unstructured" we mean the lack of direct spatial, sequential, or temporal relationship between the different dimensions of the data points. In other words, the vector of dimensions (the columns in the table) can be permuted in any way without changing the meaning or information about the samples, given that all samples have the same permutation.

This characteristic is made evident when comparing tabular to image data, in which the dimensions (pixels) have a spatial structure. Altering the order of the pixels of an image alters the overall meaning of the image. Moreover, the information conveyed by an individual pixel in one sample can differ from the pixel information in the same position in another sample [30]. This difference does not occur in tabular data, for which the same dimensions (columns) will always retain the same meaning across samples. Fig. 1 illustrates this concept. The fixed representation of dimensions in tabular data is going to be exploited by the method described in Section 3.

The terminology can also differ between tabular and image data. For the latter, the terms "inputs" (when the data is being given to a machine learning model) or "dimensions" often refer to the raw values of the pixels, while "features" usually is used to describe higher-order information and structures, for instance, edge detectors. For tabular data, the difference between these terms is blurrier. In this work, "dimensions," "inputs," and "features" refer to the same objects: the columns of the tables.



Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*

Figure 1: **Comparison between image and tabular data.** In the left image, the "meaning" of the pixels inside the red square can be understood as the bee antennae, but in the right image, the same pixels are only representing the picture background. Compare this to samples from the Iris dataset [18], in which the columns will always represent the same feature across all samples, for instance, the sepal length (in red).

Despite the focus on image and text data in deep learning and explainable AI research [29], much of the available data in science and business is tabular. The knowledge domains range from microarray genes expression from cancer patients [16], single-nucleotide polymorphism in forensic genetics [4], patient therapy data [47], hemogram exam data from COVID-19 patients [5; 19; 46], e-commerce [38], to astrophysics [27].

The lack of interpretable results is one of the reasons why artificial neural networks are not being more seriously applied to such data, disregarding their predictive power. Lamy et al. [24], for instance, point out how deep learning is not unanimously well regarded in medicine and advises against its use in breast cancer tabular data. This lack of explainability led to simpler statistical or linear models often being chosen despite their lower predictive power or inability to deal with the underlying complexity and nonlinearity [32]. There are, thus, still gaps in how to “open black-boxes” neural networks while retaining their advantages, such as solving nonlinear tasks and autonomous feature learning. Section 3 describes a way of achieving this.

The advent of explainable machine learning also unlocks another possible way to perform feature selection. Feature selection algorithms can help machine learning models achieve better generalization, prediction performance, and scalability for datasets with a large dimensionality and the presence of irrelevant, redundant, or noisy input features [2]. Usually, the selection happens as a preprocessing step. Still, with interpretable algorithms, a model can be trained without selection and afterward be inspected, so we know the input features it learned. The features considered unhelpful or contradictory can be removed, and the model retrained on the “improved” subset of input features [30]. Besides the uses for model validation and improvement, such algorithms can lead to knowledge discovery on scientific problems with limited human intuition or domain knowledge [32]. Examples of these applications will be discussed in Section 4.

In this work, we tackle neural network interpretability when trained on tabular data, describing how already existing algorithms can be adapted for this goal. Moreover, several strategies for visualizing and validating the interpretation outcome are applied to synthetic and real-world datasets.

2. Related work

This section gives a more in-depth explanation of Layer-wise Relevance Propagation (LRP), that will be used by the method proposed in Section 3. It also discusses applications of interpretability algorithms.

2.1. Layer-wise Relevance Propagation

LRP is an algorithm for neural network interpretation capable of identifying the specific features in the input responsible for the network’s output for each sample [6]. For instance, for an image classifier, it can indicate which pixels were considered when deciding to which class the image belongs [31] (Fig. 2). It found many applications in analyzing the inner-workings and quality of image and text classifiers [3]. It can be used to extract knowledge from trained machine learning models and datasets, as well as to discover their biases [30; 32].

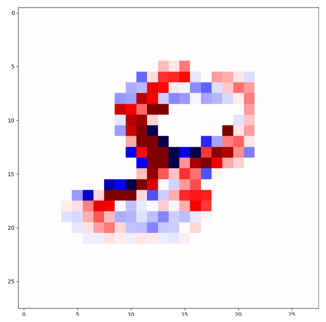


Figure 2: **Example of a relevance heatmap created with the LRP algorithm.** The original image is a handwritten digit "8" from the MNIST dataset [14]. A neural network was trained to classify the digits in ten different classes. This heatmap shows which pixels in the sample image were deemed relevant by the network for this prediction: darker red pixels have more relevance, darker blue pixels have more "counter-relevance", and white pixels are irrelevant. For this particular sample it is possible to see that the red pixels follow the shape of a regular "8", while blue pixels appear in the bottom where the trace is not so clear.

LRP works with two passes through a trained neural network (Fig. 3):

1. **Feedforward**: goes from the input layer to the output layer. The activation of the neurons in the kth layer is $a_k = \phi(W_{jk}a_j + b_k)$, in which $\phi(.)$ is the activation function, W_{jk} are the weights from layer j to layer k, a_j are the activations from layer j, and b_k are the biases from layer k. The index of individual neurons in the layers was omitted. This corresponds to the black-headed arrows in Fig. 3 and is the standard feedforward pass of a neural network.
2. **Backward:** sends the output value back through the network structure as a relevance message that is distributed among the neurons in the previous layers and stops when the input layer is reached [6]. This corresponds to the white-headed arrows in Fig. 3. The backpropagation procedure must follow a conservation property, in which the relevance message received by a neuron must be redistributed to the previous

layer in an equal amount [6]:

$$\sum_j R_j = \sum_k R_k \tag{1}$$

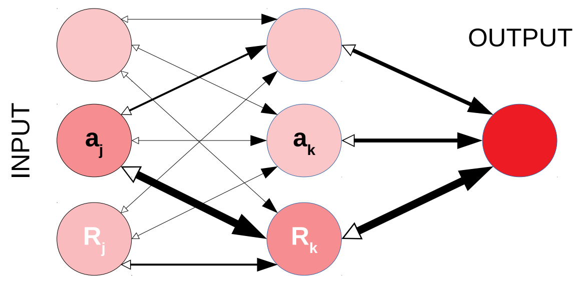


Figure 3: **A schematic diagram of the two passes required for computing the relevance.** The black-headed arrows show the feedforward pass, while the white-headed arrows show the backward pass. The color intensity at the neurons represents the amount of relevance they received from the next layer. The output neuron has the darkest color because the output value is the original relevance. a_i is the activation of a neuron in the ith layer, and R_i is the relevance of a neuron in the ith layer.

The propagation can also be studied as a succession of Taylor expansions within the Deep Taylor Decomposition framework [30; 31]. The computation of the backward pass can use several rules that take into account the input domain and layer type. A description of the most common rules can be found in Montavon et al. [30]. We describe here the two LRP rules used in this work: LRP-$\alpha\beta$ [6] in Eq. 2 and w^2-rule [31] in Eq. 3.

$$R_j = \sum_k \left(\alpha \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} - \beta \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-} \right) R_k \tag{2}$$

$$R_j = \sum_k \frac{w_{jk}^2}{\sum_j w_{jk}^2} R_k \tag{3}$$

For both equations, k and j are the kth and jth 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. LRP-$\alpha\beta$ is a rule that satisfies local conservation properties, and that achieved good empirical results [32]. The constants must obey the properties $\alpha - \beta = 1$ and $\beta \geq 0$ [6]. This rule can be interpreted as a relevance signal αR_k that is redistributed to previous layers in proportion to its excitatory effect on a_k, and a "counter-relevance" $-\beta R_k$ that is redistributed to previous layers in proportion to its inhibitory effect on a_k [32]. The w^2-rule is a special rule for the input layer of a neural network when any real-valued input is admissible. It redistributes relevance proportionally to the square magnitude of the weights [31].

The relevance at the input neurons is the relevance of the corresponding feature regarding the neural network output, or how much each input contributes towards the current prediction. The closer the relevance of an input is to zero, the less it contributes, meaning it is less relevant. A positive relevance means that the feature is collaborating towards the network prediction, while a negative relevance or "counter-relevance" is "against" the same prediction. Fig. 2 has an example of this concept.

Previous research has pointed to several advantages of the LRP algorithm when compared with other techniques. LRP can provide an explanatory input pattern that indicates evidence for and against the network prediction, for any network with monotonous activation, even for non-continuous functions [39]. This allows LRP to work on a wide variety of neural network architectures, layer types, activation functions, and training algorithms. It also showed better quantitative and qualitative results than sensitivity maps and the deconvolution algorithm [32; 39].

LRP has some drawbacks when applied to tabular data, however. Usually, for this kind of data, a neural network would have fully connected layers instead of convolutional or pooling layers. It was shown empirically that the capability of LRP producing good explanations for networks with many fully connected layers is jeopardized because of loss of selectivity. This is due to the LRP redistributing relevance to too many lower-layer neurons (as opposed to in convolutional layers) [32].

Another challenge, based on experimental results, is the application of LRP when the input features are not in the same space, for instance, a mixture of numerical and categorical features with binary encoding. This scenario can produce less explainable results even for normalized features. One way to mitigate this problem is to stratify the numerical features [47].

2.2. Applications

As already mentioned, researchers applied explainable machine learning and LRP specifically in the inspection and comparison of deep learning models and data. LRP has also been used for knowledge discovery and decision interpretation in tasks such as MRI-Based Alzheimer's disease classification [9], classification of audio signals [7], and the prediction of morphological and molecular tumor profiles [8]. In common, all these works deal with spatial or temporal data, while our method intends to bring LRP to the analysis of tabular data.

We highlight that Böhle et al. [9] proposed a strategy to analyze the LRP results on specific brain areas in 3D images by summing the relevance of their voxels that is similar in some aspects to the method presented in the next section. This was possible because magnetic resonance imaging follows the same format and allows a relative alignment across subjects, which is not necessarily true for other types of image data. Even in Böhle et al. [9], individual differences make impossible a perfect match between the reference location and individual patients. The possibility of perfect feature alignment between all tabular data samples will be exploited by the proposed method in the next section. LRP has also been used as a way to explain therapy predictions of metastatic breast cancer [47] from a mixture of tabular (demographic, tumor, and metastasis information) and sequential data (time-stamped clinical events). This work used an LSTM (Long Short-Term Memory) with an embedding layer and a feedforward network for the classification and LRP for feature selection based on how frequently the features received the largest amount of relevance. A similar method to LRP, DeepLIFT [41], was used by Fiosina et al. [17] in the task of augmentation of small RNA expression profile. The work compared samples based on the scores from DeepLIFT to understand which sRNAs are important for a particular prediction using tabular data.

3. Proposed method

We propose what we call "relevance aggregation", to serve as a generic interpretatability method for feedforward multilayer neural networks and tabular data. Relevance aggregation aims to use algorithms capable of interpreting the individual decisions of neural networks, such as LRP, to retrieve the input features that were deemed by the model as the most relevant for its predictions when performing classification or regression. Clustering interpretation can also be a possibility due to recent works on "neuralization-propagation" [23].

A naïve version of the method is shown in the pseudo-code of Algorithm 1. A practical implementation can make use of matrix and vector operations to be more efficient. The main idea is to train a neural network on the desired data (line 3), and then compute the relevance of each input at each sample (line 6). The algorithm uses the absolute values to give the same importance to relevance (positive values) and counter-relevance (negative values) (line 7). The values are scaled so that all samples have the same weight when aggregating (line 7). This is particularly important for regression tasks, in which the difference in the target value would bias the relevance towards samples with higher targets.

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
```

The aggregation is done at two levels. The first is by class, returning the average of the rescaled relevance of each input only for the samples belonging to the same class (line 12). The choice of classes is arbitrary. In the regular case for classification, one would use the real classes of the dataset, but it is possible to split the samples to suit the specific application better. For regression, the default behavior would be to consider that all samples belong to the same class. However, another possibility is to split the samples according to the ranges of the target value. This intermediate step to compute aggregation scores for different classes allows the differentiation of scores between groups of samples, acknowledging that the neural network may use distinct sets of features to identify each group.

The second level of aggregation is accomplished by averaging the relevance scores of each class (line 16), returning a global relevance score that can be sorted (line 18). The final score is the average of the class scores so that each class contributes in equal amounts to the result. It is common to deal with unbalanced datasets, and in this scenario, the relevance of samples from the larger class would overshadow the others. Examples of such datasets are discussed in Sections 4.2 and 4.3.

3.1. Handling categorical data

It is often the case in tabular data to have categorical features that must be encoded with numerical values before they can be used as input for a neural network. One-hot is a common example of encoding in which a binary vector of length equal to the number of possible categories represents a feature. The index of the corresponding category value receives a value of 1, and all others receive 0.

With this encoding as an example, it can be handled by relevance aggregation in two ways. The first is to consider the original feature is now b binary features and take their scores individually. This scheme is useful to compare the relevance of each possible value of the original feature. The second way of computing the score is based on how to obtain the pixel relevance from a color image: summing the relevance of the pixel at each color channel [39]. Analogously, we can get a single score value for a categorical feature by summing all the b scores from the encoding. However, in the context of relevance aggregation, it was chosen to average the scores with the arithmetic mean instead, to keep the values between zero and one, and to prevent artifact introduction when distinct categorical features have different encoding lengths.

3.2. Hyperparameters of aggregation

Most of the options of hyperparameters for relevance aggregation are decisions about the neural network and the relevance algorithm. In regard of Algorithm 1, the main choices are the types of average and aggregation used in lines 12 and 16. In this work, we opted for the geometric mean in Eq. 4, as it would not change the range of values and because it is less affected by outliers than the arithmetic mean.

$$f(x_1, ..., x_L) = (\Pi_{l=1}^{L} |x_l|)^{1/L} \tag{4}$$

The chosen rules for computing the relevance were the LRP-$\alpha_2\beta_1$ for hidden layers, and the w^2-rule for the input layer, for the reasons presented in Section 2.1. The LRP-$\alpha_2\beta_1$ helps the identification of features that contradict the prediction by allowing the flow of negative relevance [32]. The relevance heatmaps, sparsity, and relevance orders are stable for small β values [9]. Nevertheless, other propagation rules or similar methods for explaining, such as DeepLIFT, can easily be incorporated into the relevance aggregation algorithm depending on specific needs.

For LRP, it is possible to select which output of the neural network will be inspected. In this work, we always use the relevance of the predicted class, but it may be useful to choose a specific output for different applications. For classification, the relevance should be computed from the outputs before applying the softmax function, while the linear output is used for regression.

Regarding the neural network, it is up to the user to select the appropriate model, topology, and training algorithms. This work focuses only on feedforward multi fully connected layer neural networks due to the nature of tabular data. In the case of using LRP for computing the relevance, to produce good explanations, it is recommended to apply dropout, to use ReLU as the activation function, and to force the biases to be zero or negative during training time to help the relevance distribution [32].

3.3. Motivation

This method should work for a wide variety of tabular data, including mixtures of them (for instance datasets with real-valued and categorical features), and both (multiclass) classification and regression tasks. In Section 4, we show how the method performs on diverse datasets.

One question that may arise is why aggregating the relevance is needed in the first place, instead of applying the regular interpretation algorithms

Table 1: **The difference between seeing the "raw" relevance from LRP and the relevance scores from relevance aggregation.** These values came from the experiment with a breast cancer gene expression dataset described in Section 4.2. Each cell shows the relevance of a specific feature (columns) and sample (rows), as obtained with LRP. The last row is the final relevance score obtained (considering the full dataset). The six samples were randomly picked, one from each of six different classes. The full dataset has 151 samples and $54,676$ features, making the individual inspection of the relevance infeasible. With the relevance score it is possible to sort the features according to their global contribution to the neural network prediction. In this example, the feature "240701_at" is the most relevant, with a score of 0.416.

	240701_at	**223259_at**	**1552801_at**	**1568691_at**	**207226_at**	**217051_s_at**	...
s105	0.065	0.062	0.049	0.009	0.003	0.003	...
s101	0.036	0.049	0.016	0.005	0.002	0.008	...
s162	0.043	0.034	0.037	0.030	0.017	0.003	...
s200	0.018	0.014	0.006	0.008	0.002	0.004	...
s183	0.014	0.015	0.008	0.007	0.003	0.001	...
s174	0.024	0.027	0.043	0.026	0.013	-0.001	...
...	...	...	...	...	...	...	...
score	**0.416**	**0.390**	**0.278**	**0.158**	**0.076**	**0.030**	...

and analyzing the individual samples. The first reason is the heterogeneity present in the data that makes interpretations at the sample level less informative or clear. Meanwhile, the aggregation reveals the contribution of the features globally or group-wise. This means that with aggregation, it is possible to discover which features the network is relying on for the classification of specific classes or even stratified groups in regression. Our experiments in Section 4 show that for multiclass classification, the set of the most relevant features is different for each class, which can provide new insights about the inner-works of the network and the data itself.

Aggregation also solves the problem of choosing *which* sample to inspect among potentially thousands of candidates by offering a global view of the relevance. While for image and text data, it may be feasible to inspect the results of individual samples visually; for tabular data, it would mean to check several vectors of numerical values one by one (illustrated in Table 1) [29], what is tiresome and prone to error.

The third reason for using aggregation is related to two already discussed topics: the loss of selectivity in LRP when applied to fully connected layers and the fixed order of tabular data features. From the conservation property (Eq. 1), the total amount of relevance cannot change from layer to layer, so fully connected layers (that do not share weights as convolutional layers do) with many input neurons will sparsify the relevance in the previous layer. By combining the relevance of multiple samples in a single score, the "noise" in each input has less influence, and a clearer pattern of what is relevant emerges. What makes this pattern possible are the fixed positions for each feature in the data, allowing the direct aggregation of relevance over multiple samples by matching their features. As previously discussed, for image or text data aligning the actual features is not trivial, while for tabular data, we have this for granted (Fig. 1).

Finally, by exploiting algorithms such as LRP, there is no need to retrain models or create local surrogates to approximate the behavior of the black-box predictor as in LIME [37]. LRP also takes into consideration feature interaction, something that is ignored by many explainable machine learning algorithms [29].

3.4. Visualization

As implied in the last section, computing the relevance scores for each feature is only useful with practical ways to understand and interpret the results. For this reason, we also propose two methods for visualizing the output from relevance aggregation.

1. **Table heatmap**: inspired by the relevance heatmaps in Bach et al. [6], the table heatmap is a simple way to display the relevance of tabular data. Each row represents a feature ordered by their global or class scores (first columns). Each column represents a sample from the dataset. The cells contain the original data value and are colored according to the scaled relevance (blue for negative, red for positive, white for zero, and the intensity increasing with the magnitude of the value). Examples can be seen in Figs. 5, 10, 12a, and 14a.
2. **Weighted t-SNE:** t-SNE is a popular method for visualizing high-dimensional data in 2D [28]. In the early steps, the algorithm must compute the distance between the data points, usually relying on the Euclidean distance. We propose that using the weighted Euclidean distance in Eq. 5, with the relevance scores as weights, one should get a 2D visualization closer to the data separation learned by the neural network. The scores range from 0 (less relevant) to 1 (more relevant), so when scaling each dimension by its score, the dimensions with higher relevance will account for a greater portion of the distance between the

points, and thus will have more influence in their position in the final visualization. Examples can be seen in Figs. 6, 11, 12c, and 14c.

$$d(p, q) = \sum_{i=1}^{n} \sqrt{(w_i(q_i - p_i))^2} \tag{5}$$

4. Experiments and results

We now present a series of experiments with synthetic and real-world data to test and validate relevance aggregation. To present it as a general method, we selected datasets with distinct sizes, data types, and tasks (binary or multi-class classification and regression). All code was written in Python 3.7 with the libraries SciPy [45], Pandas, NumPy, and Scikit-learn [35].

All neural networks were trained with Keras using the Adam optimizer, the (class weighted) categorical cross-entropy (for classification) or mean squared error (MSE) (for regression) as loss function, and ReLU as the activation function. The output layer uses softmax for classification and linear function for regression. The training of all networks was under the restrictions listed in Section 3.2, including the use of (10%) dropout layers. When needed, the inputs were normalized with the standard score (separated in training and testing sets). The number of trainable parameters ranged from $1,241$ to $5,498,156$. We note that the trained networks are not a result of relevance aggregation, but a necessary previous step.

For all experiments the relevance and scores were computed as noted in Section 3.2, using the LRP Toolbox[1] [25]. The results were computed using stratified 10-fold cross-validation, except for the breast cancer data that used 3-fold due to the available number of samples. When comparing multiple algorithms, the same data partitions were used to allow a fair comparison. As done in Yang et al. [47], to make the experiments more challenging and realistic, the relevance and scores reported are always regarding the test samples. This also avoids any resulting bias from the training process.

The experiments ran on a 64-bit Ubuntu 18, Intel Xeon E5-2650V4 30 MB, 2 CPUs, 2.2Ghz, 48 cores/threads, 128G, 8TB, Titan Xp Pascal machine. Despite not being the focus of the work, we note that dealing with tabular data is often cheaper than other types of data. The processing time

[1]https://github.com/sebastian-lapuschkin/lrp_toolbox

of the whole experiment pipeline ranged from minutes to a few hours for the larger datasets.

4.1. Synthetic data

Two major questions should be answered to validate relevance aggregation: (i) are the input features with larger scores contributing the most to the neural network's output? And (ii) do these features convey relevant information about the original data? We start our experiments with synthetic data as it makes answering the second question relatively straightforward.

The first two datasets were adapted from the exclusive-OR (XOR) problem as a function of two out of n inputs presented by Tan et al. [44]. In this problem, the XOR function is computed from two fixed binary features (known to the user, but not to the machine learning models) in the input data, and the remaining $n - 2$ ($n = 50$) features are random binary values with no impact on the output. For this dataset, 500 samples were randomly created, keeping the classes balanced. This task is presented in its classification (two classes: 0 or 1) and regression (target value equal to the XOR) form. Thus, the algorithm's goal is to correctly compute the XOR function from the two truly relevant features.

Despite appearing to be a simple task, the nonlinearity and presence of many irrelevant features make the problem harder, as can be seen by the results in the corresponding columns of Table 2. Nevertheless, the neural networks were capable of reaching satisfactory predictions. This does not necessarily mean that the networks learned the expected behavior, however. They can be overfitting using all available inputs, or learning misguiding correlations that may appear between the random inputs over the limited sampling, as happened to the decision tree in Fig. 4.

For a decision tree, it is trivial to check the input features being considered, but the process is trickier for neural networks. However, we can inspect them using relevance aggregation as described in Section 3, and for the same data that originated the tree in Fig. 4, we obtain from the neural network the relevance scores in Fig. 5a (visualized with the table heatmap detailed in Section 3.4). This heatmap shows that only two input features received large scores (0.919 and 0.897), and the same is true if the classes are treated separately. These two high scoring features are indeed the two truly relevant features chosen for computing the XOR function when the dataset was created. It is now possible to know that the neural network made a better prediction and correctly learned the relevant rules from the original data.

Table 2: **Classification and regression performance of different algorithms for all datasets.** This table presents the average F1 score (for classification) and MSE (for regression) and standard deviation from stratified 10-fold cross-validation (3-fold for the breast cancer dataset) for the trained neural networks (NN), decision trees (DT), and support vector machines (SVM). It also lists the results for SVM trained only with the top ranked features from relevance aggregation (RelAgg), DT, mMRM, and Kruskal-Wallis Test (KW). Retraining SVM using the top ranked features from relevance aggregation always improved its predictions. Number of top ranked features are 2 (XOR), 5 (synthetic, e-commerce, ENEM), and 10 (breast cancer). The best values are in bold.

	Classification (F1 score)				Regression (MSE)		
	XOR	3-classes	Breast cancer	E-commerce	XOR	Synthetic	ENEM
NN	0.982 ± .021	0.691 ± .048	0.900 ± .029	0.759 ± .013	0.018 ± .019	3.295 ± .384	8392.814 ± 1186.210
DT	0.499 ± .069	0.829 ± .023	0.599 ± .039	**0.793 ± .015**	0.266 ± .018	5.423 ± .904	6630.980 ± 268.285
SVM	0.629 ± .000	0.696 ± .000	0.905 ± .000	0.669 ± .000	0.202 ± .000	4.030 ± .000	7563.633 ± .000
SVM (RelAgg)	**1.000 ± .000**	**0.922 ± .000**	**0.956 ± .019**	0.777 ± .001	**0.010 ± .000**	**1.612 ± .327**	6354.633 ± 200.827
SVM (DT)	0.653 ± .174	0.916 ± .004	0.843 ± .086	0.763 ± .020	0.325 ± .014	2.429 ± .000	**6128.120 ± 79.470**
SVM (mRMR)	0.515 ± .005	**0.922 ± .001**	0.514 ± .023	0.312 ± .126	-	-	-
SVM (KW)	0.513 ± .006	**0.922 ± .000**	0.812 ± .078	0.780 ± .000	-	-	-

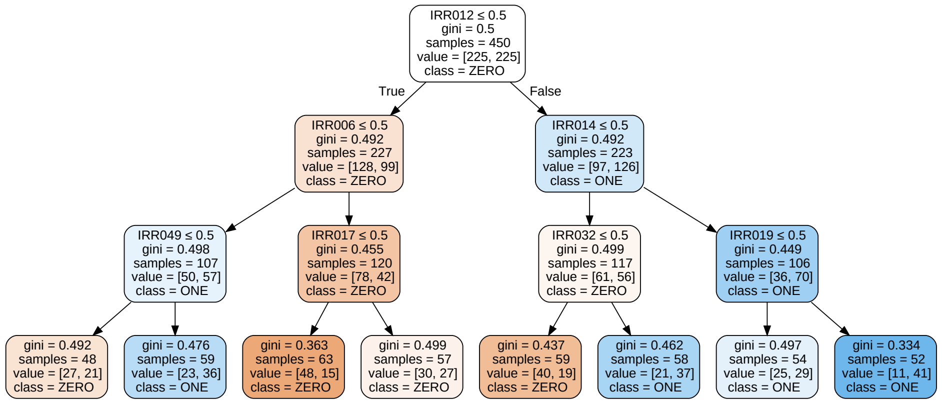
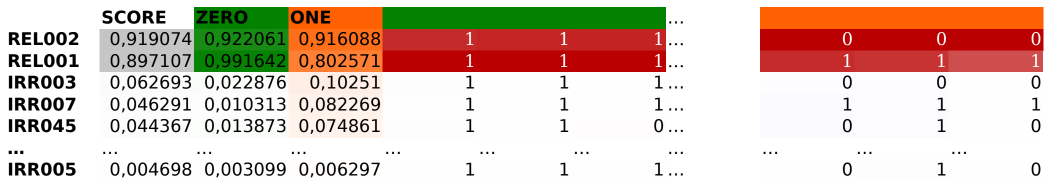


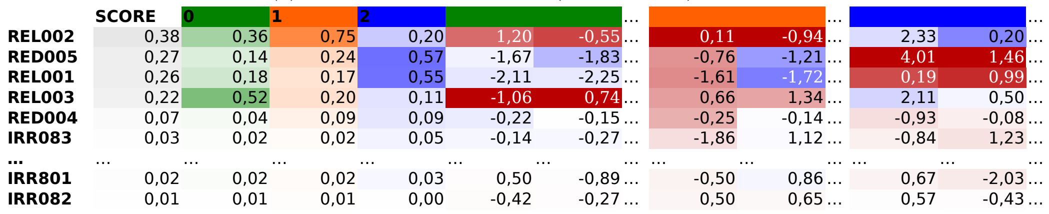
Figure 4: **Decision tree generated for the XOR problem (classification).** It failed to learn the truly relevant inputs from the data (same partition from Fig. 5a) and instead relies on seven irrelevant inputs. The trees were trained with Scikit-learn, with a minimum of 0.1 samples per leaf, balanced class weights, and Gini index.

This idea can be summarized with the "selection accuracy" metric. In the case of relevance aggregation, the selection accuracy can be measured as the ratio of truly relevant input features that received the highest relevance scores. For the example in Fig. 5a, the selection accuracy of relevance aggregation is 1.0 because the two features used to compute the XOR function are placed in the first and second positions in the ranking of the scores. For other ranked based feature selection algorithms, this metric can be computed in the same way. For decision trees, we define the "selection accuracy" as



	SCORE	ZERO	ONE	🟩	🟩	🟩	...	🟧	🟧	🟧
REL002	0,919074	0,922061	0,916088	1	1	1	...	0	0	0
REL001	0,897107	0,991642	0,802571	1	1	1	...	1	1	1
IRR003	0,062693	0,022876	0,10251	1	1	1	...	0	0	0
IRR007	0,046291	0,010313	0,082269	1	1	1	...	1	1	1
IRR045	0,044367	0,013873	0,074861	1	1	0	...	0	1	0
...	...	...	...	...	...	...	...	...	...	...
IRR005	0,004698	0,003099	0,006297	1	1	1	...	0	1	0

(a) Table heatmap for XOR (classification) dataset.



	SCORE	0	1	2	🟩	🟩	...	🟧	🟧	...	🟦	🟦	...
REL002	0,38	0,36	0,75	0,20	1,20	-0,55	...	0,11	-0,94	...	2,33	0,20	...
RED005	0,27	0,14	0,24	0,57	-1,67	-1,83	...	-0,76	-1,21	...	4,01	1,46	...
REL001	0,26	0,18	0,17	0,55	-2,11	-2,25	...	-1,61	-1,72	...	0,19	0,99	...
REL003	0,22	0,52	0,20	0,11	-1,06	0,74	...	0,66	1,34	...	2,11	0,50	...
RED004	0,07	0,04	0,09	0,09	-0,22	-0,15	...	-0,25	-0,14	...	-0,93	-0,08	...
IRR083	0,03	0,02	0,02	0,05	-0,14	-0,27	...	-1,86	1,12	...	-0,84	1,23	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...
IRR801	0,02	0,02	0,02	0,03	0,50	-0,89	...	-0,50	0,86	...	0,67	-2,03	...
IRR082	0,01	0,01	0,01	0,00	-0,42	-0,27	...	0,50	0,65	...	0,57	-0,43	...

(b) Table heatmap for synthetic 3-classes dataset.

Figure 5: **Excerpt of the table heatmaps for the XOR (classification) and 3-classes synthetic data.** Figure (a) shows the partial table heatmap (Section 3.4) for a neural network trained on the XOR (classification) dataset. The rows are ordered by the score value, and each represents one input feature (the complete table would contain 50 rows). The first column has the input features labels, columns "SCORE", "ZERO", and "ONE" have the relevance scores for the whole dataset, only class "zero" (when the XOR of the relevant inputs is 0), and only class "one" (when the XOR of the relevant inputs is 1), respectively. The higher the score, the darker the background color. The last six columns present three samples (the complete table would contain 500 samples) from each class ("zero" marked in green and "one" marked in orange), and its cells contain the original feature value. Darker red backgrounds represent larger positive relevance values, darker blue represents larger negative relevance values, and white represents relevance values close to zero ("irrelevant"). Figure (b) shows the partial table heatmap for a neural network trained on the synthetic 3-classes dataset, following the same representation of Figure (a). From this table, it is clear that the neural network deemed different input features relevant for each class, based on the score values in columns "0", "1", and "2".

the ratio of truly relevant features present in the nodes of the tree.

We compare the selection accuracy of neural networks (based on their relevance aggregation scores) over stratified 10-fold cross-validation against three other algorithms: (i) decision trees, (ii) minimum Redundancy Maximum Relevance Feature Selection (mRMR)[2] [15], and the Kruskal–Wallis H test (SciPy implementation). The results in the first two columns of Table 3 show that the neural networks were always able to learn the XOR function from the correct input features, for both the classification and regression

[2]http://home.penglab.com/proj/mRMR/

Table 3: **Selection accuracy for the synthetic datasets.** This table presents the average "selection accuracy" and standard deviation for four feature selection algorithms on four synthetic datasets. We define the "selection accuracy" as the ratio of the r truly relevant features ranked in the r first positions on the selection of each algorithm. The exception is the decision tree, which does not rank the selection. In this case, the selection accuracy is defined as the ratio of the r truly relevant features present in the nodes of the final tree. Values were not computed for mRMR and Kruskal-Wallis test on regression tasks, as they compute the difference between classes. The best results are in bold.

	XOR (classification)	**XOR (regression)**	**Synthetic 3-classes**	**Synthetic regression**
RelAgg	**1.00 ± .00**	**1.00 ± .00**	**1.00 ± .00**	**0.97 ± .07**
Decision Tree	0.35 ± .39	0.00 ± .00	0.78 ± .06	0.75 ± .00
mRMR	0.00 ± .00	-	0.80 ± .00	-
Kruskal-Wallis	0.05 ± .15	-	0.94 ± .09	-

variants. The other algorithms struggled in this task.

Besides the XOR problem, we also created a synthetic 3-classes dataset with 1,000 samples using the algorithm from [22; 35][3]. This dataset has 1,000 input features, from which only five are informative for the class separation (known to the user, but not to the machine learning models), and the remaining are irrelevant. Similarly, we also made a synthetic regression dataset using [13; 35][4], with 1,000 samples and four informative input features among the total of one hundred.

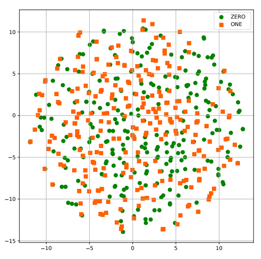
The predictive performance of the trained neural networks for these datasets is shown in Table 2, and Fig. 5b is an example of table heatmap for a neural network trained on the synthetic 3-classes data. As with the XOR data, using relevance aggregation, the truly relevant features were consistently placed in the top positions, receiving the largest relevance scores (Table 3).

Besides the table heatmaps present in Fig. 5, in Section 3.4 a second visualization method called "weighted t-SNE" was described, and Fig. 6 shows its usefulness. A clearer data distribution emerges by weighting the distances of the data points with the corresponding relevance scores of each dimension. It is possible to see the XOR problem's well-known pattern, the classes from the synthetic dataset disentangle, and the samples from the regression dataset are placed in a gradient of the target values.

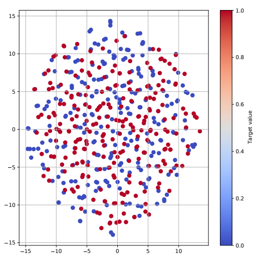
However, these are not "perfect" representations of the real relationship between the points, as evidenced by the XOR visualization. If they were

[3]scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

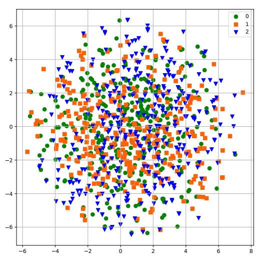
[4]scikit-learn.org/stable/modules/generated/sklearn.datasets.make_sparse_uncorrelated.html



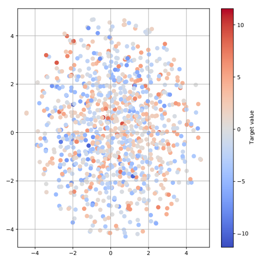
(a) XOR (class.)



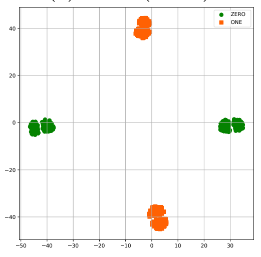
(c) XOR (reg.)



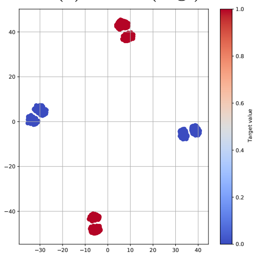
(e) 3-classes



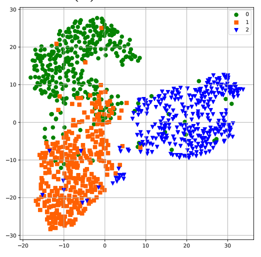
(g) Regression



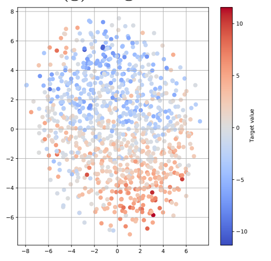
(b) XOR (class.) wgt.



(d) XOR (reg.) wgt.



(f) 3-classes wgt.



(h) Regression wgt.

Figure 6: **Visualization of the synthetic datasets with t-SNE and weighted t-SNE.** The first row (a, c, e, g) shows the regular t-SNE visualization for the four synthetic datasets. Each point represents one sample. For classification tasks the color corresponds to the classes, for regression tasks the color is a gradient from the lowest target value (dark blue) to the largest target value (dark red). Because most of the input features are random, there is no clear separation in the plots. The second row (b, d, f, h) shows the same samples visualized with weighted t-SNE (Section 3.4) after the training of a neural network. In this case, the distance between the points was scaled following the relevance score of each dimension according to the network. As can be seen, clear clusters or gradients emerge, but some noise remains.

indeed perfect, all samples would collapse to only four points in the chart. This noise is due to the “leaking” relevance that the neural network assigns to irrelevant input features, as shown by the third row downwards in Fig. 5a. We hypothesize that these distributions are more likely to represent how the neural network “perceives” the data after training than the visualization with regular t-SNE, in which the points are all entangled in a single cluster.

So far, we demonstrated that relevance aggregation is capable of retrieving the truly relevant input features from a dataset. This does not automatically prove the neural network relies more on these inputs than the others, which was the first question at the beginning of this subsection. Moreover, Fig. 5b indicates that the neural network is using distinct subsets of input features for making predictions about each class (the score rankings are different).

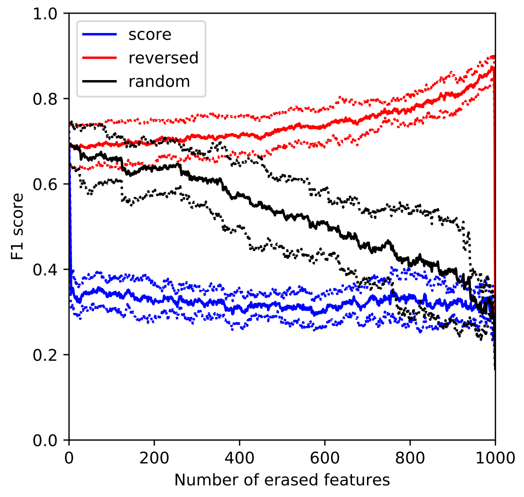
To check that the relevance scores obtained from relevance aggregation actually match the practical importance of the input features for the trained neural networks, we use a similar approach as the perturbation analysis presented in Samek et al. [39]. In the original paper, perturbation analysis was used to validate and compare the relevance heatmaps created with LRP and other interpretability algorithms on convolutional neural networks trained on image data. We adapted it to the models and data studied in this work.

The goal is to evaluate the impact of systematically "erasing" input features on the predictive power of trained neural networks, measured on the test data by the F1 score or the MSE. "Erasing" an input feature in this context means to manually set its value to zero on all samples being fed to the network. The value of zero was chosen because, after the initial data normalization, all input features have a mean equal to zero. If the scores from relevance aggregation reflect the importance of the features to the prediction, we expect that erasing input features from the largest to the smallest score will produce a drastic and continuous drop on the network classification or regression capacity. Analogously, erasing the input features in the reverse order should not affect the predictive power as much.

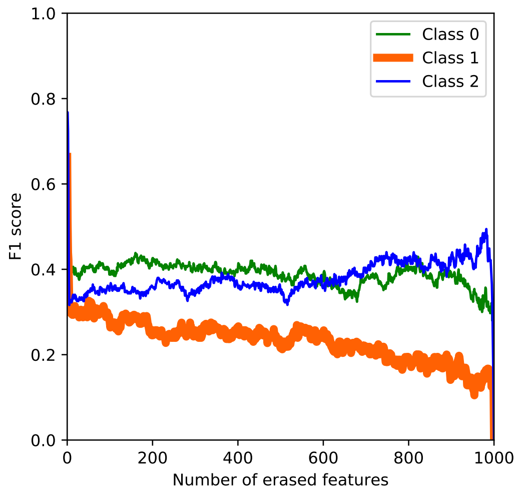
Taking the synthetic 3-classes dataset as an example, we can see this exact behavior taking place in Fig. 7a. Erasing the input features with larger scores leads to abrupt drops in the F1 score (blue line shows the average for ten networks). By contrast, erasing the input features with small scores can improve the model performance (red line), while erasing in a random order stays in the middle ground, gradually degrading the predictions (black line). This is evidence that most of the predictive capacity of the networks comes from the input features with high scores, and very little from the features with low scores.

Figs. 7b, 7c, and 7d show the same behavior but at a class level. In these figures, each line is the average F1 score of a specific class, and each plot shows the impact of erasing input features according to the class-specific scores (as seen in Fig. 5b). The classes are more affected by erasing input features according to their own scores. A larger decrease in the F1 score illustrates this, thus collaborating to the claim that relevance aggregation is identifying the features used at a class level and that the networks are using distinct input features to predict each class.

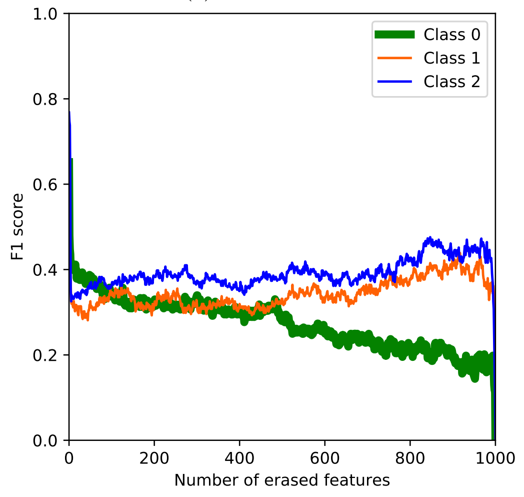
Finally, Adebayo et al. [1] show that some interpretability methods are independent of the model and the data generating process, thus failing in tasks that are sensitive to them, such as explaining the relationship between



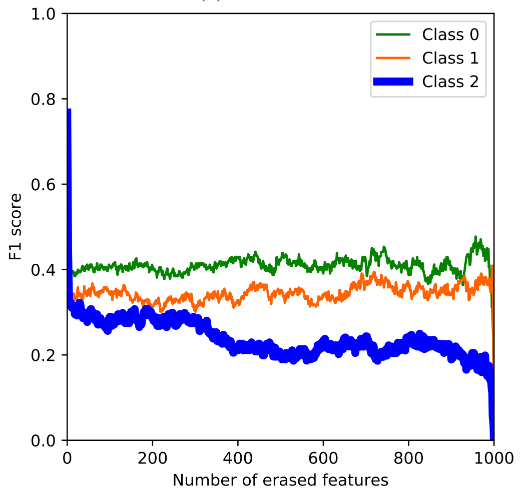
(a) Global score



(c) Class “1”



(b) Class “0”



(d) Class “2”

Figure 7: **Perturbation analysis for neural networks trained on the synthetic 3-classes dataset.** Figure (a) shows the average (10-fold cross-validation) impact of erasing input features (turning them to zero) on the F1 score. The solid blue line represents erasing the features from the largest relevance score to the smallest. The solid red line represents erasing the features in the reverse order. The solid black line represents erasing in random order. The dashed lines are the standard deviation. Figures (b), (c), and (d) show the average F1 score of each class independently. In each figure, the input features are erased following the score ranking (largest to smallest) of the correspondent class (thicker line). The standard deviation was omitted for clarity.

inputs and outputs as learned by the model. In their results, the input values dominate the results of methods like ϵ-LRP. To show that relevance aggregation is sensitive to changes in the model or the data, we adapted two experiments performed by Adebayo et al. [1] and used the synthetic 3-classes dataset and three repetitions.

For the parameter randomization test, we compare the rank of features obtained with networks trained on the data and networks with random weights and biases. If relevance aggregation is sensitive to model parameters, changing them should also change the ranking of features. If the result does not change, the explanation may be only sensitive to the data or the network topology. Using the Spearman Rank correlation to compare the ranks between networks with trained and random parameters, we obtained an average coefficient of 0.0067 ($p < 0.05$), so the results are not correlated, and relevance aggregation is sensitive to the parameters of the model. For the data randomization test, we compare the feature relevance from networks trained on the original data and networks trained on a version of the dataset with randomly permuted labels. If an explanation is not sensitive to this change, it could not explain the relationship between samples and labels. Using the Spearman Rank correlation to compare the results from the two groups, they are not correlated with -0.0114 ($p < 0.05$), thus relevance aggregation is sensitive to the relationship between samples and labels.

4.2. Breast cancer gene expression

We now move to the application of relevance aggregation to neural networks trained on real-world data. The first dataset comes from the Curated Microarray Database (CuMiDa) [16][5], a repository containing cancer microarray datasets assembled for machine learning research. This specific dataset has the expression levels of over $50,000$ genes from tissues with one of five types of breast cancer (HER, basal, cell line, luminal A, and luminal B), plus samples from healthy ("normal" in the dataset) tissue. The selection of genes from microarray data is needed because of the presence of irrelevant, redundant, and noisy expressions, and as a way for early tumor detection, cancer discovery, cancer diagnosis, and prognosis [2; 11].

Table 2 lists the average test F1 score of different classifiers trained on this data. The perturbation analysis presented in Fig. 8 shows that erasing

[5]http://sbcb.inf.ufrgs.br/cumida

the input features with the highest relevance scores causes a greater drop in the average quality of predictions than erasing randomly picked input features or the ones with the lowest scores, as was the case with the synthetic data (Fig. 7a). However, it was necessary to erase a greater portion of the input features before the drop becomes noticeable, what suggests that the neural networks are relying on thousand of input features to make predictions instead of identifying a small subset of informative inputs.

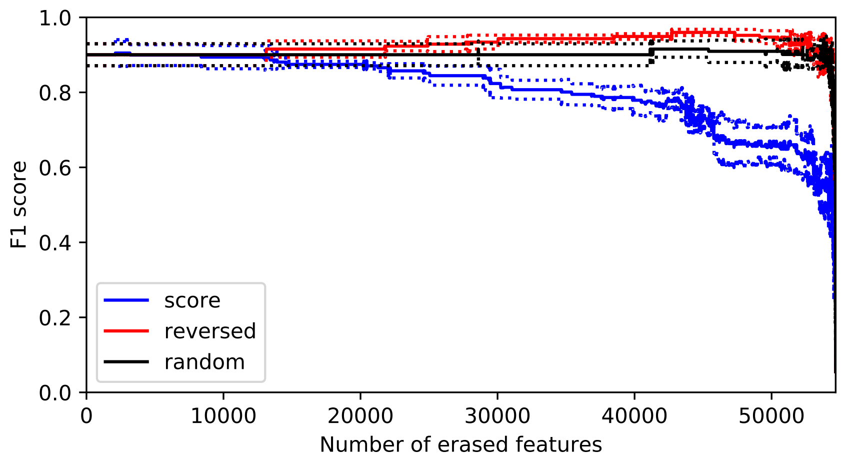
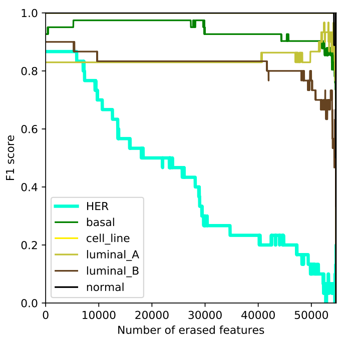


Figure 8: **Perturbation analysis for neural networks trained on the breast cancer dataset.** Figure details as in Fig. 7a.

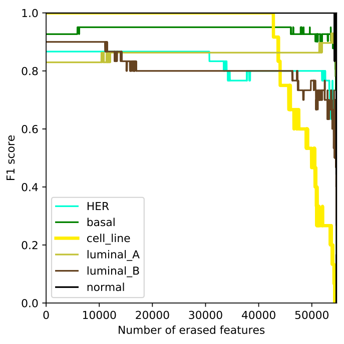
Nevertheless, as done in Fig. 7, it is possible to perform the same analysis dividing the data to see if the neural networks learned different rules for each class. As reported in Fig. 9, this appears to be the case, with the average F1 score of an individual class dropping much sooner when its input features with higher relevance scores are erased. In contrast, the performance of other classes remains stable for longer. The difference in the relevance assigned to input features for each class is further demonstrated in Fig. 10, which shows for a single neural network how the score values can change between classes.

It is also possible to note the effect of the class-specific relevance scores in the weighted t-SNE plots in Fig. 11, for the same network in Fig. 10. For instance, using the scores from the classes luminal A or luminal B leads to a clearer separation of their samples in the visualization (Figs. 11f and 11g). The cell line class is also better clustered with weighted t-SNE, while in the original, it is divided into two (Fig. 11a).

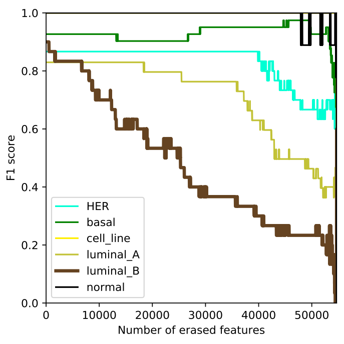
Unfortunately, with real-world data, it is not trivial to compute the se-



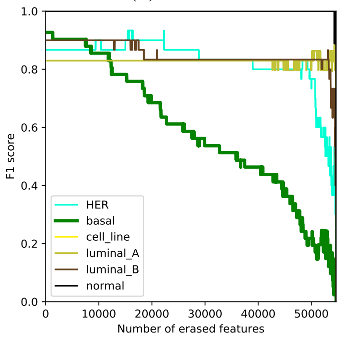
(a) HER



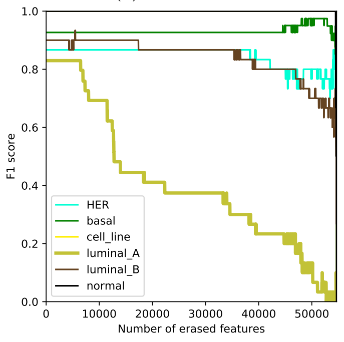
(c) Cell line



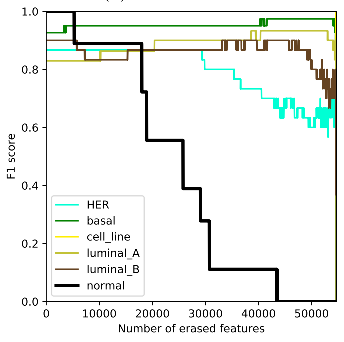
(e) Luminal B



(b) Basal



(d) Luminal A



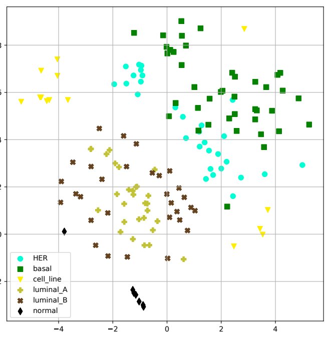
(f) Normal

Figure 9: **Perturbation analysis for neural networks trained on the breast cancer dataset, divided by class.** Each figure corresponds to the average F1 score for all classes when erasing input features following the relevance scores of a specific class (thicker line). The standard deviation was omitted for clarity. Figure details as in Fig. 7.

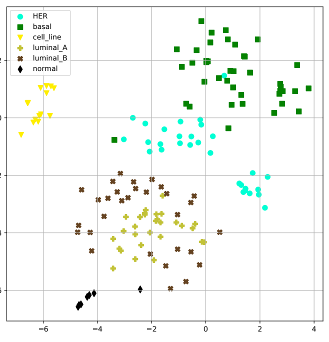
	SCORE	**HER**	**Basal**	**Cell line**	**Luminal A**	**Luminal B**	**Normal**
240701_at	0,416	0,636	0,585	0,501	0,297	0,477	0,196
223259_at	0,390	0,567	0,765	0,412	0,241	0,447	0,184
205635_at	0,389	0,512	0,327	0,371	0,360	0,280	0,556
206560_s_at	0,385	0,287	0,679	0,421	0,266	0,317	0,475
219415_at	0,383	0,256	0,773	0,380	0,270	0,291	0,534
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**
217051_s_at	0,030	0,055	0,162	0,033	0,057	0,065	0,001

Figure 10: **Excerpt of the table heatmaps for the breast cancer data.** Only the global relevance scores and relevance scores of the six classes for the top and bottom input features are shown. Figure details as in Fig. 5.

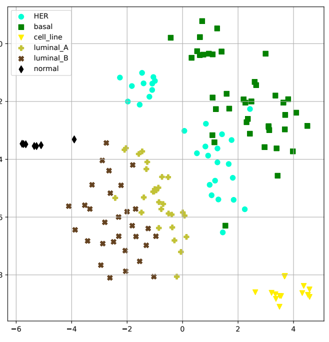
lection accuracy, as was in Section 4.1. Nevertheless, we can estimate the quality of the feature ranking by using only the input features with the highest scores to train another classifier and checking if its performance improved. It is assumed that gene selection performed with a classifier is specific to that



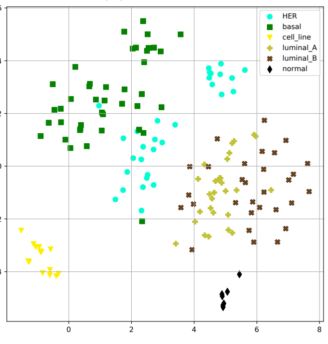
(a) Original



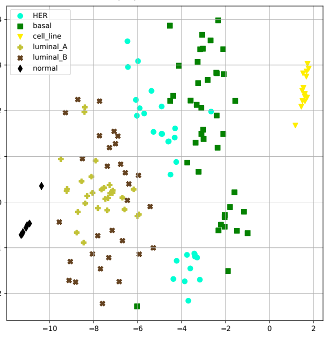
(d) Basal



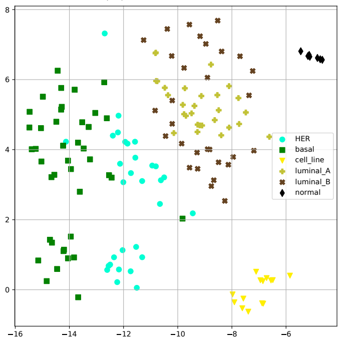
(g) Luminal B



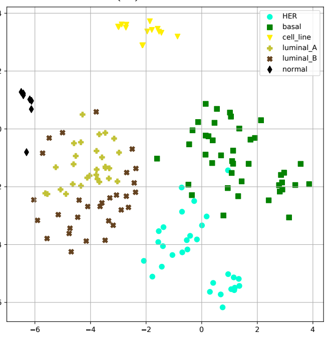
(b) Score



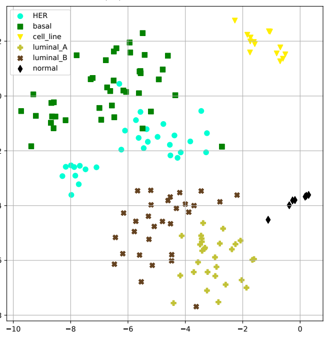
(e) Cell line



(h) Normal



(c) HER



(f) Luminal A

Figure 11: **Visualization of the breast cancer data with t-SNE.** Figure (a) shows the data using regular t-SNE, and Figure (b) using weighted t-SNE with the global relevance scores. Figures (c) to (h) show the data using weighted t-SNE with the relevance scores of each class. Points with different colors and symbols represent each class. Figure details as in Fig. 6.

model (in this case, the neural network), so there is no assurance that the selected genes will perform well in other predictors [2]. The chosen algorithm for the comparison was the support vector machine (SVM), as it is considered the state-of-the-art in gene expression classification [2; 21; 43].

Table 2 shows the average F1 score of an SVM trained on this dataset with all input features, and with only the top class ranked features from relevance aggregation and other feature selection algorithms. As can be seen, for this dataset, the use of genes selected with relevance aggregation improved the average performance of the SVM, even more than mRMR and the Kruskal-Wallis H test, that had already been used for this specific task [15; 21]. Furthermore, for all datasets, the SVM performance was improved by first selecting features with relevance aggregation (Table 2), which suggests that the ranking of features can be generalized. It has been argued that SVMs are insensitive to large numbers of irrelevant genes, and that feature selection could degrade their accuracy [43]. These results, however, indicate that neural networks can select features that improve SVMs predictions, which was also suggested by Grisci et al. [21].

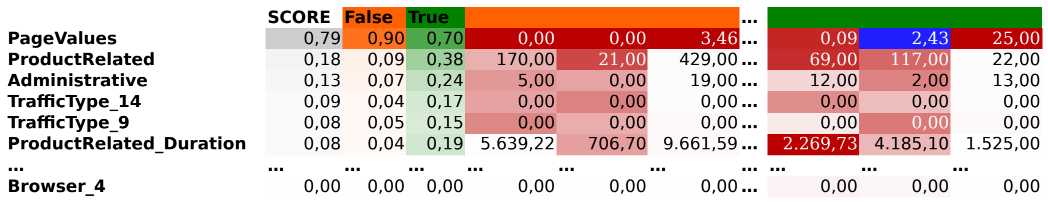
4.3. E-commerce

The second real-world data being experimented on regards the purchasing intention of an e-commerce website visitors using session and user information[6]. This is a binary classification task with numerical and categorical input features. Each of the 12,330 samples corresponds to a different user over one year, in which 84.5% did not complete a shopping order, and the rest completed it [38].

The average classification performance of different algorithms trained on this data is shown in Table 2. As happened in all experiments, using the top five features with the highest relevance score ranking according to relevance aggregation increased the F1 score of another classifier (SVM). The results of inspecting a neural network using relevance aggregation are shown in Fig. 12. The input feature with the highest relevance score in Fig. 12a is the "Page Value", a metric computed by "Google Analytics" that represents the average value of the web page visited by a user before a shopping transaction.

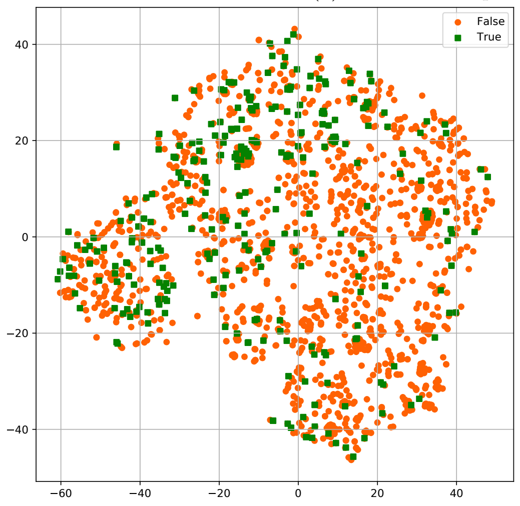
According to Sakar et al. [38], that used filter algorithms such as mRMR and mutual information to evaluate the discriminating information about the intent of the visitor carried by each input feature, "Page Value" is indeed the most informative. However, the second input feature with the highest relevance score in Fig. 12a, "Product Related", although also related to the

[6]archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset

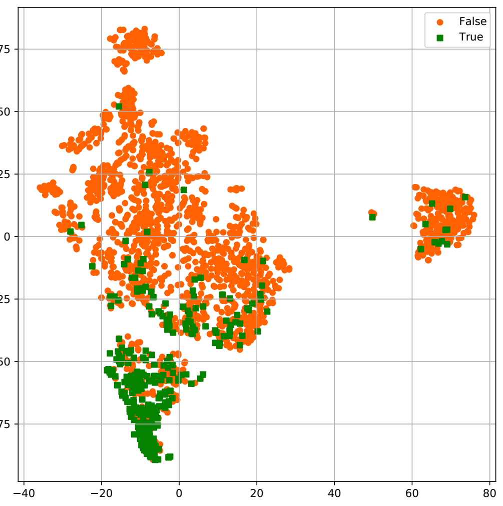


	SCORE	False	True				...			
PageValues	0,79	0,90	0,70	0,00	0,00	3,46	...	0,09	2,43	25,00
ProductRelated	0,18	0,09	0,38	170,00	21,00	429,00	...	69,00	117,00	22,00
Administrative	0,13	0,07	0,24	5,00	0,00	19,00	...	12,00	2,00	13,00
TrafficType_14	0,09	0,04	0,17	0,00	0,00	0,00	...	0,00	0,00	0,00
TrafficType_9	0,08	0,05	0,15	0,00	0,00	0,00	...	0,00	0,00	0,00
ProductRelated_Duration	0,08	0,04	0,19	5.639,22	706,70	9.661,59	...	2.269,73	4.185,10	1.525,00
...	...	...	...	...	...	...	...	...	...	...
Browser_4	0,00	0,00	0,00	0,00	0,00	0,00	...	0,00	0,00	0,00

(a) Table heatmap for e-commerce dataset.



(b) Original.



(c) Weighted.

Figure 12: **Visualization of the results with the e-commerce dataset.** Figure (a) shows an excerpt from a table heatmap of one of the neural networks—the columns "False" and "True" show how the score changed for the two classes. Figure (b) and (c) show the t-SNE of the data before and after weighting the distances using the relevance scores. Figures details as in Fig. 5 and 6.

class variable, was ranked in the lower positions by the algorithms in Sakar et al. [38]. This was due to the high correlation of this feature with the already selected "Page Value". Distinctly from these algorithms, the neural network in Fig. 12 is not necessarily penalizing the redundancy between features, even though it is made clear from the relevance scores that "Page Values" is the major contributor to the network prediction.

4.4. Brazilian National High School Exam

The final experiment was on data from the Brazilian "*Exame Nacional do Ensino Médio*" (National High School Exam) or ENEM from 2016. ENEM is an annual nationwide exam organized by the Ministry of Education consisting of multiple-choice questions about languages, math, natural sciences, and humanities, plus an essay writing. Among other uses, the results from ENEM are used in the application for most public and some private universities in Brazil [26].

This specific dataset[7] comes from the Brazilian contest website "Codenation"[8]. The task at hand is to learn regression by predicting the math exam scores of over 13 thousand candidates using hundreds of personal, educational, socioeconomic, and demographic input features. This data can be real-valued, ordinary, or categorical.

As with the previous experiments, the predictive results from the trained neural networks are in Table 2, and the use of the ten input features with the highest relevance score was able to decrease the MSE of an SVM trained on the same data. Fig. 13 shows that erasing the input features by the decreasing order of their relevance score also causes an increase of the neural networks' MSE. Fig. 14a is an example of a table heatmap for a neural network trained on this data, in which it is possible to see the difference in scores between samples with target value between zero and 700, and 700 and 952. Fig. 14c shows a weighted t-SNE with the same samples gradient behavior seen in Fig. 6h.

We highlight the two top-ranked input features from Fig. 14a. The first, with a relevance score of 0.97, was the candidates' score in the natural sciences exam. This comes as no surprise since the exam grades in natural sciences and mathematics are correlated in the dataset (0.584). In another application, this feature would probably be removed from the training. However, we left it as both a "sanity check," and a reminder that by using relevance aggregation, one can identify such features that may require attention.

The second highest relevance score was attributed to the input feature encoding the gender of the candidates, that in the dataset was either male or female. What this suggests is that the neural network learned to use the gender of a candidate to predict their math exam grade. This can be seen as a case of *machine bias*, a recent concern in both academia and industry, where trained machine learning or statistical models reproduce controversial societal asymmetries, often unknowingly to their designers [36]. Machine bias often comes in the form of gender or racial bias, and can be aggravate by the use of black-box models, as they hide the undesirable assumptions

[7]https://www.kaggle.com/davispeixoto/codenation-enem2

[8]https://www.codenation.dev/

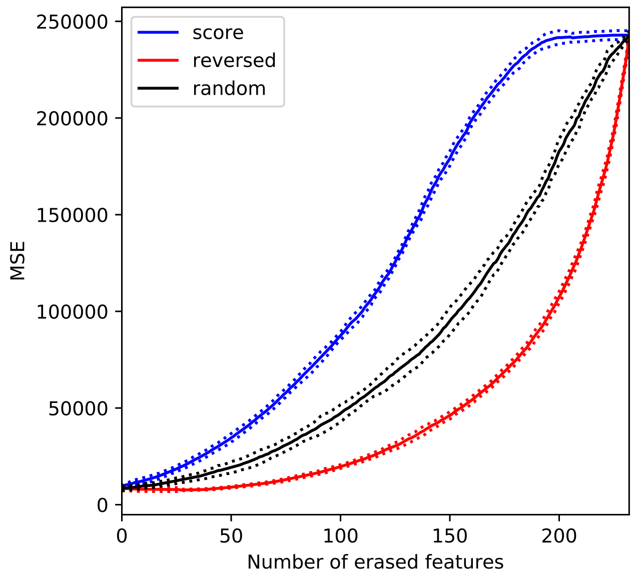


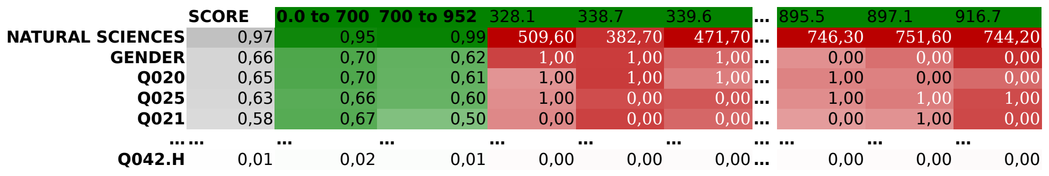
Figure 13: **Perturbation analysis for neural networks trained on the ENEM dataset.** Figure details as in Fig. 7a. Note that because the y-axis is the MSE, an increase in the values shows the deterioration of the predictions.

being made [29; 36].

In the specific case of ENEM, a study of a similar exam system indicates that in standardized test scores males tend to outperform females, even though females significantly outperform males in high school grade point average, showing that the gender gap can be impacted by the assessment method used by universities [40]. Another study shows that ENEM results can impact the female students' choice probabilities of college majors [26].

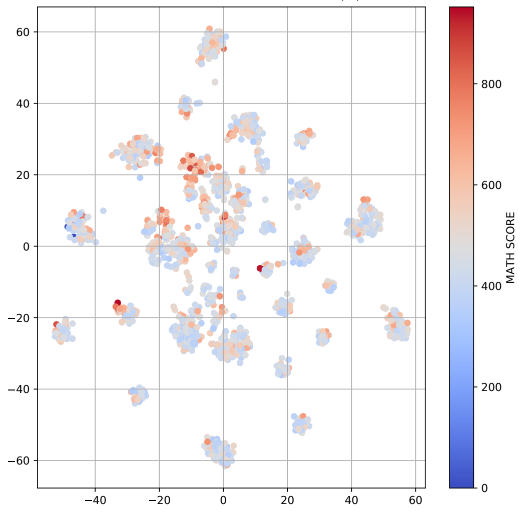
The gender, racial, and social inequalities among the scores of students taking ENEM has also been discussed in the Brazilian media[9]. According to the reports, that used the exam from 2016 as well, social and cultural factors are associated to the gender gap in the mathematics and natural sciences results. Data from PISA, the world largest international exam, organized by the Organisation for Economic Co-operation and Development (OECD), also reveals that parents and teachers show different levels of expectation for male and female students, and that girls report being less confident in their mathematical abilities. In countries where this inequality received attention,

[9]https://infograficos.estadao.com.br/educacao/enem/desigualdades-de-genero-e-raca/

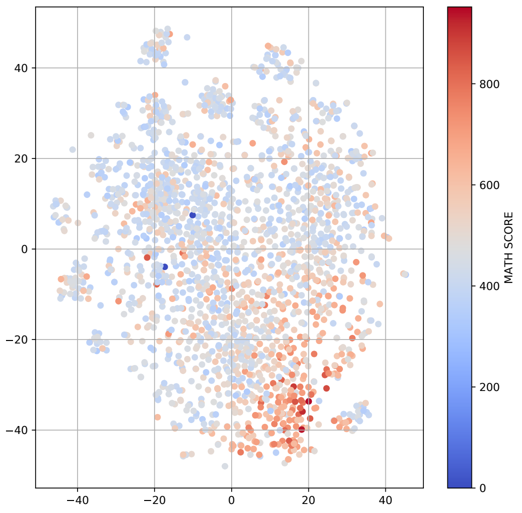


	SCORE	0.0 to 700	700 to 952	328.1	338.7	339.6	...	895.5	897.1	916.7
NATURAL SCIENCES	0,97	0,95	0,99	509,60	382,70	471,70	...	746,30	751,60	744,20
GENDER	0,66	0,70	0,62	1,00	1,00	1,00	...	0,00	0,00	0,00
Q020	0,65	0,70	0,61	1,00	1,00	1,00	...	1,00	0,00	0,00
Q025	0,63	0,66	0,60	1,00	0,00	0,00	...	1,00	1,00	1,00
Q021	0,58	0,67	0,50	0,00	0,00	0,00	...	0,00	1,00	0,00
...	...	...	...	...	...	...	...	...	...	...
Q042.H	0,01	0,02	0,01	0,00	0,00	0,00	...	0,00	0,00	0,00

(a) Table heatmap for ENEM dataset.



(b) Original.



(c) Weighted.

Figure 14: **Visualization of the results with the ENEM dataset.** Figure (a) shows an excerpt from a table heatmap of one of the neural networks. The columns "0.0 to 700" and "700 to 952" show how the score changed when isolating the largest target values. Figure (b) and (c) show the t-SNE of the data before and after weighting the distances using the relevance scores. Figures details as in Fig. 5 and 6.

the gap in mathematics ceased to exist [33].

Usually it is not desirable to reproduce such historical biases in future predictions, or at least this component of the decision process should be known to the model creators and users. The biased predictions can have the unintended consequence of reinforcing the asymmetry in expectations for different groups. The use of interpretability algorithms can help to shine light on these issues.

5. Conclusion

Neural networks have become a standard tool in several machine intelligence applications, but their lack of interpretability is still a bottleneck for their wide adoption. Several works focus on explaining the predictions of neural networks, but few take into consideration the use of tabular data. To contribute to this topic, we presented relevance aggregation, an algorithm that builds upon previous interpretability methods by aggregating the relevance computed from several samples. Our method was tested in a variety of synthetic and real-world datasets and worked with different types of tabular data for both classification and regression tasks. The results show that relevance aggregation can correctly identify which input features are the most important for the network's predictions and that this set of features can be distinct for each class.

We also presented two ways to visualize the results from relevance aggregation. We suggest that weighted t-SNE can better represent the patterns learned by a neural network, and thus lead to a better comprehension of the models.

The outcome of relevance aggregation can also help to identify relevant features from the original data, serving as a way to perform feature selection or knowledge discovery. The top-ranked features were consistently able to improve the performance of another classifier (SVM). However, the selection quality for knowledge discovery strongly depends on the classifier's quality, so the relevance scores should be seen as an indication of "true" relevance rather than a decisive conclusion about the nature of the original data. The relevance scores should always be taken into account together with the model performance metrics [9]. Nevertheless, even in the case of poorly trained neural networks, relevance aggregation can help identify incorrect or irrelevant rules or machine bias.

Data and code availability

The necessary source code and datasets used in the experiments, and the trained neural networks models and results can be accessed in GitHub: https://github.com/sbcblab/RelAgg.git. The microarray data used in this work is available in the CuMiDa database: http://sbcb.inf.ufrgs.br/cumida.

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], Conselho Nacional de Desenvolvimento Científico e Tecnológico (`CNPq`)

[*311611/2018-4*], Alexander von Humboldt-Stiftung (`AvH`) [*BRA 1190826 HFST* `CAPES-P`] - Germany, and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (`CAPES`) [`DAAD/CAPES PROBRAL` 88881.198766/2018-01] - Brazil. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. We gratefully acknowledge the support of the NVIDIA Corporation (hardware grant program).

References

[1] Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B., 2018. Sanity checks for saliency maps. Advances in neural information processing systems 31, 9505–9515.

[2] Ang, J.C., Mirzal, A., Haron, H., Hamed, H.N.A., 2015. Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. IEEE/ACM Transactions on Computational Biology and Bioinformatics 13, 971–989.

[3] Arras, L., Horn, F., Montavon, G., Müller, K.R., Samek, W., 2017. "what is relevant in a text document?": An interpretable machine learning approach. Plos One 12, e0181142.

[4] Avila, E., Felkl, A.B., Graebin, P., Nunes, C.P., Alho, C.S., 2019. Forensic characterization of brazilian regional populations through massive parallel sequencing of 124 snps included in hid ion ampliseq identity panel. Forensic Science International: Genetics 40, 74–84.

[5] Avila, E., Kahmann, A., Alho, C., Dorn, M., 2020. Hemogram data as a tool for decision-making in covid-19 management: applications to resource scarcity scenarios. PeerJ 8, e9482.

[6] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W., 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. Plos One 10, e0130140.

[7] Becker, S., Ackermann, M., Lapuschkin, S., Müller, K.R., Samek, W., 2018. Interpreting and explaining deep neural networks for classification of audio signals. arXiv preprint arXiv:1807.03418 .

[8] Binder, A., Bockmayr, M., Hägele, M., Wienert, S., Heim, D., Hellweg, K., Stenzinger, A., Parlow, L., Budczies, J., Goeppert, B., et al., 2018. Towards computational fluorescence microscopy: machine learning-based integrated prediction of morphological and molecular tumor profiles. arXiv preprint arXiv:1805.11178 .

[9] Böhle, M., Eitel, F., Weygandt, M., Ritter, K., 2019. Layer-wise relevance propagation for explaining deep neural network decisions in mri-based alzheimer's disease classification. Frontiers in Aging Neuroscience 11, 194.

[10] Bojarski, M., Choromanska, A., Choromanski, K., Firner, B., Ackel, L.J., Muller, U., Yeres, P., Zieba, K., 2018. Visualbackprop: Efficient visualization of cnns for autonomous driving, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Brisbane. pp. 1–8.

[11] Bolón-Canedo, V., Sánchez-Marono, N., Alonso-Betanzos, A., Benítez, J.M., Herrera, F., 2014. A review of microarray datasets and applied feature selection methods. Information Sciences 282, 111–135.

[12] Carrizosa, E., Nogales-Gómez, A., Morales, D.R., 2016. Strongly agree or strongly disagree?: Rating features in support vector machines. Information Sciences 329, 256–273.

[13] Celeux, G., El Anbari, M., Marin, J.M., Robert, C.P., et al., 2012. Regularization in regression: comparing bayesian and frequentist methods in a poorly informative situation. Bayesian Analysis 7, 477–502.

[14] Deng, L., 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine 29, 141–142.

[15] Ding, C., Peng, H., 2005. Minimum redundancy feature selection from microarray gene expression data. Journal of Bioinformatics and Computational Biology 3, 185–205.

[16] Feltes, B.C., Chandelier, E.B., Grisci, B.I., Dorn, M., 2019. Cumida: An extensively curated microarray database for benchmarking and testing of machine learning approaches in cancer research. Journal of Computational Biology 26, 376–386.

[17] Fiosina, J., Fiosins, M., Bonn, S., 2020. Explainable deep learning for augmentation of small rna expression profiles. Journal of Computational Biology 27, 234–247.

[18] Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. Annals of Eugenics 7, 179–188.

[19] Formica, V., Minieri, M., Bernardini, S., Ciotti, M., D'Agostini, C., Roselli, M., Andreoni, M., Morelli, C., Parisi, G., Federici, M., et al., 2020. Complete blood count might help to identify subjects with high probability of testing positive to sars-cov-2. Clinical Medicine 20, e114–9.

[20] Garcia, R., Telea, A.C., da Silva, B.C., Tørresen, J., Comba, J.L.D., 2018. A task-and-technique centered survey on visual analytics for deep learning model engineering. Computers & Graphics 77, 30–49.

[21] Grisci, B.I., Feltes, B.C., Dorn, M., 2019. Neuroevolution as a tool for microarray gene expression pattern identification in cancer research. Journal of Biomedical Informatics 89, 122–133.

[22] Guyon, I., 2003. Design of experiments of the nips 2003 variable selection benchmark, in: NIPS 2003 Workshop on Feature Extraction and Feature Selection, Whistler. pp. 1–30.

[23] Kauffmann, J., Esders, M., Montavon, G., Samek, W., Müller, K.R., 2019. From clustering to cluster explanations via neural networks. arXiv preprint arXiv:1906.07633 .

[24] Lamy, J.B., Sekar, B., Guezennec, G., Bouaud, J., Séroussi, B., 2019. Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach. Artificial Intelligence in Medicine 94, 42–53.

[25] Lapuschkin, S., Binder, A., Montavon, G., Müller, K.R., Samek, W., 2016. The lrp toolbox for artificial neural networks. Journal of Machine Learning Research 17, 1–5.

[26] Lemos, M.d.J., 2019. The Effect of gender on college major choice: evidence from Brazil. Master's thesis. Fundação Getulio Vargas. Brazil.

[27] Lyon, R.J., Stappers, B., Cooper, S., Brooke, J., Knowles, J., 2016. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. Monthly Notices of the Royal Astronomical Society 459, 1104–1123.

[28] Maaten, L.v.d., Hinton, G., 2008. Visualizing data using t-sne. Journal of Machine Learning Research 9, 2579–2605.

[29] Molnar, C., 2019. Interpretable Machine Learning. `https://christophm.github.io/interpretable-ml-book/`.

[30] Montavon, G., Binder, A., Lapuschkin, S., Samek, W., Müller, K.R., 2019. Layer-wise relevance propagation: an overview, in: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Springer, pp. 193–209.

[31] Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R., 2017. Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition 65, 211–222.

[32] Montavon, G., Samek, W., Müller, K.R., 2018. Methods for interpreting and understanding deep neural networks. Digital Signal Processing 73, 1–15.

[33] OECD, 2015. The ABC of Gender Equality in Education. doi:`https://doi.org/https://doi.org/10.1787/9789264229945-en`.

[34] Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., Carter, S., 2020. Zoom in: An introduction to circuits. Distill doi:`10.23915/distill.00024.001`.

[35] Pedregosa, F., Varoquaux, G., Duchesnay, E., et al., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[36] Prates, M.O., Avelar, P.H., Lamb, L.C., 2020. Assessing gender bias in machine translation: a case study with google translate. Neural Computing and Applications 32, 6363–6381.

[37] Ribeiro, M.T., Singh, S., Guestrin, C., 2016. "why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144.

[38] Sakar, C.O., Polat, S.O., Katircioglu, M., Kastro, Y., 2019. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks. Neural Computing and Applications 31, 6893–6908.

[39] Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R., 2017. Evaluating the visualization of what a deep neural network has learned. IEEE Transactions on Neural Networks and Learning Systems 28, 2660–2673.

[40] Saygin, P.O., 2020. Gender bias in standardized tests: evidence from a centralized college admissions system. Empirical Economics 59, 1037–1065.

[41] Shrikumar, A., Greenside, P., Kundaje, A., 2017. Learning important features through propagating activation differences, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org. pp. 3145–3153.

[42] Simonyan, K., Vedaldi, A., Zisserman, A., 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 .

[43] Statnikov, A., Wang, L., Aliferis, C.F., 2008. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. BMC Bioinformatics 9, 319.

[44] Tan, M., Hartley, M., Bister, M., Deklerck, R., 2009. Automated feature selection in neuroevolution. Evolutionary Intelligence 1, 271–292.

[45] Virtanen, P., Gommers, R., van Mulbregt, P., Contributors, et al., 2019. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. arXiv e-prints `arXiv:1907.10121`.

[46] Yan, L., Zhang, H.T., Goncalves, J., Xiao, Y., Wang, M., Guo, Y., Sun, C., Tang, X., Jing, L., Zhang, M., et al., 2020. An interpretable mortality prediction model for covid-19 patients. Nature Machine Intelligence 2, 283–288.

[47] Yang, Y., Tresp, V., Wunderle, M., Fasching, P.A., 2018. Explaining therapy predictions with layer-wise relevance propagation in neural networks, in: 2018 IEEE International Conference on Healthcare Informatics (ICHI), IEEE. pp. 152–162.

[48] Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, Springer, Zurich. pp. 818–833.

[49] Zhou, Q., Liu, X., Wang, Q., 2021. Interpretable duplicate question detection models based on attention mechanism. Information Sciences 543, 259–272.