Model-agnostic interpretation by visualization of feature perturbations

Wilson E. Marcílio-Jr[a], Danilo M. Eler[a] and Fabrício Breve[b]

[a]Faculty of Sciences and Technology, São Paulo State University (UNESP), Presidente Prudente, SP 19060-900, Brazil
[b]Institute of Geosciences and Exact Sciences, São Paulo State University (UNESP), Rio Claro, SP 13506-900, Brazil

ARTICLE INFO

Keywords:
machine learning interpretation; particle swarm optimization; visualization

ABSTRACT

Interpretation of machine learning models has become one of the most important topics of research due to the necessity of maintaining control and avoid bias in these algorithms. Since many machine learning algorithms are published every day, there is a need for novel model-agnostic interpretation approaches that could be used to interpret a great variety of algorithms. One particularly useful way to interpret machine learning models is to feed different input data to understand the changes in the prediction. Using such an approach, practitioners can define relations among patterns of data and a model's decision. In this work, we propose a model-agnostic interpretation approach that uses visualization of feature perturbations induced by the particle swarm optimization algorithm. We validate our approach both qualitatively and quantitatively on publicly available datasets, showing the capability to enhance the interpretation of different classifiers while yielding very stable results if compared with the state of the art algorithms.

1. Introduction

Machine learning (ML) algorithms have been achieving unprecedented capability in various tasks, such as in Natural Language Processing (Peters et al., 2018; Devlin et al., 2018), Computer Vision (Szegedy et al., 2015; Krizhevsky et al., 2012; He et al., 2016; Chollet, 2017), and others. Nevertheless, the employment of ML models on applications where the consequences of mistakes could be catastrophic for organizations (Luo, 2016) has required the necessity to include model interpretation capabilities in the development of ML solutions. Interpretability increases the performance of human-model teams (Bansal et al., 2019) and improves the ability of practitioners to debug machine learning models (Kulesza et al., 2015), leading to better hyper-parameter tuning.

To this end, Explainable Artificial Intelligence (XAI) efforts focus on the interpretability side-by-side of model performance. In this case, the literature presents various approaches to decrease the lack of interpretation faced by complex models. For example, the surrogate models mimic machine learning models to provide global or local explanations (Doshi-Velez and Kim, 2017, 2018) while visualization techniques usually focus on specific models to devise graphical representations and enhance understanding about models decisions and functionality (Krause et al., 2016; Pezzotti et al., 2018; Marcilio-Jr et al., 2020) – such visualization techniques can emphasize nuances and provide insights about the execution of these black-box models to indicate if there is a bias towards a specific class. Another class of XAI methods that has been receiving a lot of attention is the model-agnostic explanation methods, which create surrogate models to return the contribution of each feature to a model's prediction (Lundberg and Lee, 2017; Ribeiro et al., 2016). Using these strategies, specialists could look at the model's prediction and analyze which features contribute the most for it, and further assess if these contributions make sense with the problem domain. For example, a doctor would look to the features taken into consideration after a model predicts a tumor as benign or not, then, such prediction could be assessed as reliable or not even if the model's prediction was correct. The characteristic of gaining insights even if a model does not have high performance is one of the most important and useful aspects of XAI methods. Although there is a lot of research focusing on interpretability aspects of machine learning models, only a few provide model-agnostic interpretation based on visual exploration (Zhang et al., 2019; Hinterreiter et al., 2020). These existing approaches still lack flexibility aspects concerning whether it could be applied to interpret classification or regression tasks (Hinterreiter et al., 2020) or assign feature important that could not correspond to the true importance given by machine learning models (Zhang et al., 2019).

In this work, we propose a novel model-agnostic interpretation approach based on the visualization of feature perturbations generated by the Particle Swarm Optimization (PSO) algorithm (Olsson, 2010). Our approach works by using the PSO algorithm to minimize a function that induces the most change in a model's prediction. Then, with carefully chosen visual encodings, the decisions of a model can be inspected. We use a prototype tool with coordinated views to visualize the similarity between classes using a hybrid visualization of a radial layout and node-link diagram, besides using detailed and summary visualizations based on dot plot to communicate the importance of features to a model's prediction. Our methodology is validated through several case studies by showing its applicability to understand machine learning models. Then, we numerically evaluate our approach according to feature importance, showing that our method yields very stable results across different datasets and surpass a few well-established methods in the sense of assigning correct importance to features. To the best of our knowledge, this is the first research study exploring the PSO algorithm to

*E-mails: wilson.marcilio@unesp.br, danilo.eler@unesp.br, fabricio.breve@unesp.br
ORCID(s):

generate perturbation of feature spaces to interpret machine learning models. Moreover, this is the first research study visualizing the execution of the PSO algorithm. Summarily, the contributions of this work are:

- Support machine learning models and feature spaces interpretation using PSO algorithm;
- Novel visual metaphors to visualize PSO execution and to interpret machine learning models.

This work is organized as follows: Section 2 presents the related works on model explainability; in Section 3 we delineate our methodology; in Section 4, we explain our visualization design; in Section 5 we show use cases to demonstrate how our method can be used to interpret a model's prediction; Section 6 provides a numerical evaluation of our technique regarding its ability to truly reflect the feature importance; discussions are provided in Section 7; we conclude our work in Section 8.

2. Related Works

The adoption of machine learning-based approaches in healthcare (Balagopalan et al., 2018; Esteva et al., 2017; Krause et al., 2016), finance (Modarres et al., 2018), governance (Meijer and Wessels, 2019), and other areas must account for explainability factors. For example, doctors would like to understand the decisions that a model made after predicting a tumor as benign or not. Besides trying to define interpretability of machine learning with scientific rigor (Doshi-Velez and Kim, 2017, 2018), as well as identifying the human factors in model interpretability, such as practices, challenges, and the needs (Hong et al., 2020), the literature is focused on proposing novel techniques to interpret machine learning models.

These techniques are often categorized into two classes: global and local (Doshi-Velez and Kim, 2018). Global methods present a summary of the input features' contributions to the model as a whole (Lundberg et al., 2020), which could deceive one understanding of the structures of the internal model's decisions. Such methods usually try to understand a model structure and functionality by applying combinations of input-output to build a mental map of a model's decision, such as building surrogate models to measure the importance of features by adding perturbation to them (Craven and Shavlik, 1995). On the other hand, local interpretability methods explain each data observation separately. LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017), for example, propose model-agnostic explanations by explaining models' predictions through the contribution of features. In this case, this set of feature contributions help at explaining the model's prediction. Note that global interpretation for LIME and SHAP is achieved by averaging the local contributions. Other methods are based on applying various model-agnostic approaches that require repeatedly executing the model for each explanation (Baehrens et al., 2010; Strumbelj and Kononenko, 2013; Datta et al., 2016) – note that LIME and SHAP also repeatedly execute the model.

Finally, the visualization community has been putting a lot of effort into using graphical representations to help researchers to interpret neural networks (Smilkov et al., 2017; Kahng et al., 2018), to understand deep learning models' training processes (Rauber et al., 2017; Pezzotti et al., 2018; Marcilio-Jr et al., 2020), such as visualizing neurons' activations using heatmap representations (Clavien et al., 2019).

In this work, we present a model-agnostic explanation approach in which perturbation weights are found through Particle Swarm Optimization. Our method searches for the minimum weight that induces the greater change in the model performance, as a way to find the importance of a feature. Besides, we present a visualization approach to interpreting the results generated by our algorithm.

3. Methodology

In this work, we aim to interpret a given model's prediction. We want to understand which decisions a model took to classify data samples correctly or incorrectly by inspecting the importance of the features.

To devise an interpretation for a model's prediction, let us first discuss how this could be done using the confusion of a trained classifier (when a classifier assigns an incorrect class to a data sample). To explore the confusions of a classifier, we create perturbations on the test set (data samples used to evaluate a model's performance). These perturbations correspond to real numbers greater than zero multiplied by the feature values of the test set. Here, if a classifier maintains its performance after perturbing a specific feature, the classifier is stable to such a feature since perturbations do not influence the prediction. Thus, the feature perturbations can measure the stability of a model.

Fixing a class of interest (c) and a feature ($\mathcal{F}_i$) from the test set, we measure the stability of the model upon feature $\mathcal{F}_i$ based on two variables: a perturbation weight (w) inducing changes on all of the values of $\mathcal{F}_i$ and the performance score of the model after predicting labels for the test set with perturbation w applied to $\mathcal{F}_i$. In other words, we multiply a weight w to all values of feature $\mathcal{F}_i$ in the test set and use the model (trained with original training set) to measure the classifier performance after perturbation, as illustrated in Figure 1. Notice that, with the perturbation on $\mathcal{F}_3$, the performance decreased from 0.98 (with $w = 1$) to 0.68 (with $w = 1.3$).

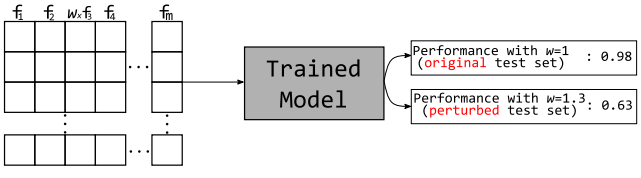

Figure 1: Perturbing a feature with weight. After multiplying $\mathcal{F}_3$ by w, the performance of the trained model is measured on the perturbed dataset.

Based on such an approach, changing the weight of important features (the ones that influence the classification)

produces instability to the model by decreasing its performance. Thus, the higher is the decrease the more important is the feature associated with it. Notice that, since we multiply the perturbation weights by the features, these weights must have two characteristics: to induce a lot of decrease in the performance and to be close to one (i.e., when $w = 1$ there is no perturbation). Figure 2 illustrates three possible scenarios for comparing perturbation weights and their respective model's performance: in the first case (**a**), two different weights induce the same accuracy, thus, we pick W_1 which is the closest to one; in the second case (**b**), we pick the second weight (W_2) since it induces the greater decrease; finally, we pick the first weight (W_1) in the third case (**c**) since it induces the greater decrease.

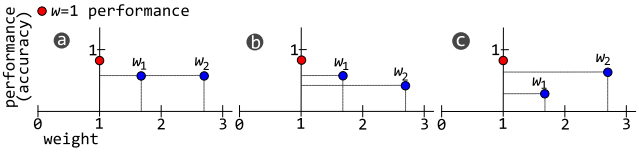

Figure 2: Analysis of perturbation weights according to their similarity to a model's performance. **a**: Both weights induce the same decrease in performance, w_1 is picked since it is closer to 1. **b**: w_2 is picked since induces greater decrease. **c**: w_1 is picked since it induces greater decrease.

Notice that we try to minimize the performance and the perturbation at the same time. In other words, we try to induce more errors in the model using perturbation weights as close to one as possible. From these two objectives, minimizing performance is more important. For instance, to minimize the performance using the accuracy score as the performance measure, we can maximize Equation 1. Notice that h is just a jargon of the PSO technique (the algorithm we used to maximize the function, explained in the following paragraphs), which is usually the name of the function that has to be optimized.

$$h = |1 - Accuracy(y, y')| \tag{1}$$

where $accuracy(y, y')$ measures the accuracy of a model using the predicted labels y' and the true labels y. Since y' corresponds to the labels returned by prediction on a perturbed test set and recalling that perfect accuracy (when y equals y') assumes value 1, maximizing Equation 1 essentially means that we are looking for the weight w that induces the higher decrease in accuracy value.

Equation 1 depends only on y', which depends on the w, thus, we have to define how labels y' are predicted by the model after perturbation using w. First, the perturbation weights have to be separately defined for each pair of class and feature since the importance of features can be different from one class to another. For example, while a feature that describes the number of goals scored in a football game can be important for a forward player getting the prize of woman-of-the-match, it could have no importance for a goalkeeper getting the same prize. So that, to generate y', the model (f) receives the matrix that joins disturbed and non-disturbed data samples, where the disturbing data samples corresponding to the instances of the same class as the analyzed (c). Equation 2 illustrates the idea, where f denotes a classification model, D^c denotes the sub-matrix whose data samples have class equals to c and $D^{!c}$ denotes the sub-matrix whose data samples are not from class c – notice that D corresponds to the test set. Finally, $w\times^i$ means the multiplication of all feature values of $\mathcal{F}_i$ by w.

$$y' = f([(w \times^i D^c)D^{!c}]) \tag{2}$$

Now, the question is how to find these weights. In this work, we use the Particle Swarm Optimization (PSO) (Olsson, 2010) algorithm, which is a bio-inspired optimization algorithm that uses particles on the search space to optimize a given function. The PSO considers a set of particles used to cooperate to search for the solution to an optimization problem. Each particle has its behavior and its group behavior (defined by a neighborhood). At the beginning of the algorithm, all of the particles receive a random position in the search space, then, they are evaluated in each iteration according to an optimization function to verify if they are close or not to the solution of the problem. Notice that each update of a particle is influenced by the position of the best particle in its neighborhood and its past positions. During iterations, the best particles will lead the others throughout the search space.

Our implementation of the PSO algorithm (see Algorithm 1) tries to find the best weights that would maximize Equation 2 while considering the importance of these weights according to the scheme of Figure 2. The first lines of the algorithm (lines 1-3) correspond to the initialization step of the particles (X and P) and the velocity of movement (V) of these particles. Notice that X and P are initialized as matrices with ones since they represent the weights for each feature and we choose to initialize the search with no perturbation (i.e., with all weights equals one). These matrices have the dimensions of the number of particles (p_m) by the number of features (m). For a fixed number of iterations, we set as one all of the weights except for the one corresponding to the feature in the analysis (lines 6 and 7) – notice that X corresponds to the best particles' positions until the current iteration and P corresponds to the positions of the particles in the current iteration. Setting to one all of the weights will induce change only in the feature of interest.

The main part of the algorithm is used to find the best position for the particle p (P_p) and the position of the best neighbor of p (P_g). For each particle p (line 8), we use the function h (see Equation 1) to verify if the current position of the particle P_p is better than the previous position (X_p) or if $P_{p,feature}$ is closer than $X_{p,feature}$ to one when the induced performance by these two weights are equal – remember that from Figure 2 we choose the weight closest to one if the performance is the same. Having set the best particle (p) in line 10, we initialize the best particle in the group (g) as being p as well. Lines 12 – 14 do essentially the same thing as

Algorithm 1 Finding minimum feature perturbation with PSO.

1: **procedure PSO**(iterations, feature, f, p_n, m, D, y, V_{min}, V_{max}, k, χ)
2: $\quad X \leftarrow [1]_{p_n,m}$
3: $\quad P \leftarrow X$
4: $\quad V \leftarrow [V_{min} \leq random \leq V_{max}]_{p_n,m}$
5: $\quad$ **while** it < iterations **do**
6: $\quad\quad X[:, j \neq \text{feature}] \leftarrow 1.0$
7: $\quad\quad P[:, j \neq \text{feature}] \leftarrow 1.0$
8: $\quad\quad$ **for** each p in p_n **do**
9: $\quad\quad\quad$ **if** $h(X_p, D, y, f) > h(P_p, D, y, f)$ **or** $\hookrightarrow\ h(X_p, D, y, f) \neq 0.0$ **and** $\hookrightarrow\ h(X_p, D, y, f) = h(P_p, D, y, f)$ **and** $\hookrightarrow\ \| 1 - X_{p,\text{feature}} \| < \| 1 - P_{p,\text{feature}} \|$ **then**
10: $\quad\quad\quad\quad P_p \leftarrow X_p$
11: $\quad\quad\quad g \leftarrow p$
12: $\quad\quad\quad$ **for** each j in neighborhood(p) **do**
13: $\quad\quad\quad\quad$ **if** $h(P_j, D, y, f) > h(P_g, D, y, f)$ **or** $\hookrightarrow\ h(P_j, D, y, f) \neq 0.0$ **and** $\hookrightarrow\ h(P_j, D, y, f) = h(P_g, D, y, f)$ **and** $\hookrightarrow\ \| 1 - P_{j,\text{feature}} \| < \| 1 - P_{g,\text{feature}} \|$ **then**
14: $\quad\quad\quad\quad\quad g \leftarrow j$
15: $\quad\quad\quad V_p \leftarrow \chi \times [V_p + \phi_1 \times (P_p - X_p) + \phi_2 \times (P_g - X_p)]$
16: $\quad\quad\quad X_p \leftarrow X_p + V_p$
$\quad$ **return** P

discussed for lines 8-10, however, looking only to the neighborhood of p. In the end, p contains the index of the best particle and g contains the index of the best particle in the group.

The final part of the iteration of the algorithm consists of the update of X_p (the matrix row storing the best position of the particle p) based on the positions of the best particle (p) and the best in the group (g). Line 15 uses such information to compute the new velocity vector based on the previous velocity (V_p) and the position of the particle (X_p). According to the PSO algorithm, ϕ_1 and ϕ_2 are random vectors and the multiplication $\phi_1 \times (P_p - X_p)$ corresponds to the individual behavior of the particle while $\phi_2 \times (P_g - X_p)$ corresponds to the social behavior of the particle. In other words, these two equations mean to move particle p based on its current best value and based on the state of its neighborhood. Finally, the parameter χ is the constriction coefficient and it helps the algorithm in the convergence, we set $\chi = 0.729$ since it is a well-established value in the literature.

Although the algorithm has a lot of parameters, the literature shows that $V_{min} = -1$ and $V_{max} = 1$ works fine, so we fixed these parameters. The neighborhood was defined as $k = 1$ following a ring pattern, i.e., for each particle p, its set of neighbor corresponds to the indices i, $p - k \leq i \leq p + k$, $i \neq p$. Finally, at each iteration, we normalize set X values to be inside the range $[0, 10]$.

After execution, $|p_m|$ weights are available for the feature in analysis in the Xk, $feature$ column vector to compute the feature's importance. Thus, the mean of the absolute difference between the weights and one $\sigma_w = \frac{1}{p_m} \sum_{j=1}^{p_m} |w_j - 1|$ and the mean of the scores when predicting using each weight minus the score with no perturbation $\sigma_s = \frac{1}{p_m} \sum_{j=1}^{p_m} |S(w_j) - S(1)|$ is taken. Then, the importance for each feature is computed as follows:

Table 1
Feature importance (I) defined by the mean weight (σ_w), and the performance weight (σ_s).

Feature	σ_w	σ_s	I
petal width (cm)	0.37515	0.43333	4.29485
petal length (cm)	0.42242	0.43333	4.29000
sepal length (cm)	0.03375	0.00000	0.00000
sepal width (cm)	0.31219	0.00000	0.00000

$$I_i = \sigma_{s_i} \times (X_{max} - [(\frac{\sigma_{w_i}}{max(W_{\sigma_{s_i}})})/X_{max}]) \quad (3)$$

where $W_{\sigma_{s_i}}$ is the set of weights with respective score same as σ_{s_i}. Essentially, with Equation 3 we derive a number that depends on the mean weight σ_{w_i} and its contribution to the model's performance. In this case, with σ_{s_i} multiplied by X_{max} we can order the importance based on accuracy, then, σ_{s_i} multiplied by the normalized weight (among those with same mean score σ_{s_i}) results in ordering for features with the same σ_{s_i}. After computing the importance of each feature, they can be ordered in a decreasing way. Table 1 exemplifies such an ordering for the *Iris* dataset, notice that the features are arranged as if they were ordered based on σ_w and σ_s, consecutively. Notice that, the higher is I more important is the feature.

4. Visualization Design

Using the methodology described in the previous section, we defined several design requirements (DR) to help with the interpretation of classification models using the Particle Swarm Optimization (PSO) algorithm. The visualization is based on the visual inspection of the PSO algorithm execution with carefully chosen visual variables to help with the interpretation of classification results. Using the information generated by the PSO algorithm and other requirements to interpret a model's decision, we want to be able to visualize:

- **DR1**: the weights that most influence on the classifier decision;
- **DR2**: how much is the influence of a feature;
- **DR3**: how is the confusion of the classifier based on different perturbation weights;
- **DR4**: the distribution of values for the features;

- **DR5**: the PSO execution;
- **DR6**: the similarity among the classes in the dataset.

To accomplish these design requirements (DR), we follow a strategy to create the visualization tool centered on the PSO execution, where the interpretation of the classifier's decisions consists of inspecting the feature perturbations generated by PSO's particles. In the following sections, we start explaining the detailed and the summary visualization for a class of interest, then, we show how to measure the similarity between classes in terms of classifier confusion.

4.1. Detailed View

In the PSO algorithm, the particles assume different values (weights) during iterations. These weights correspond to the perturbations we want to find to interpret a model's prediction. When we multiply a weight w_i^j to a feature $\mathcal{F}_k$, we can assess how much the perturbation w_i^j induces change on the classification performance – notice that i represents the particle index and j represents the current iteration of the PSO algorithm. With such an idea in mind, we can use graphical variables to encode the relationship between the perturbation (w_i^j) and the feature. Notice that the perturbation consists of multiplying w_i^j by $\mathcal{F}_k$ ($w_i^j \times \mathcal{F}_k$), in other words, multiplying w_i^j by the column k of the test set.

To visualize the perturbations and the result on the performance measure after a model's prediction, we encode the weights as circles in a horizontal axis while a color scale represents the change in the performance of the model, as illustrated in Figure 3. Such an encoding shows consistency with the PSO execution since we try to move the weights from the initial position (close to one) to positions where there is a lot of perturbation to the model. Each circle corresponds to a weight assumed by a particle during PSO execution – notice that the y-axis does not have any meaning.

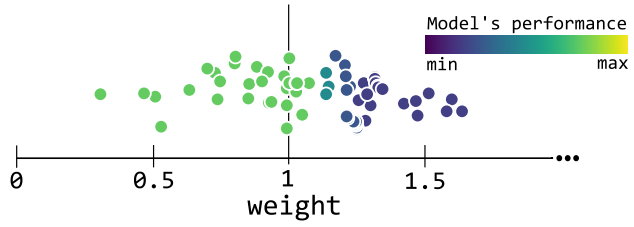

Figure 3: Encoding particle swarm optimization execution. Each circle corresponds to a value (weight) assumed by a particle during PSO's execution while the colors encode the model performance when a feature is multiplied by those weights. Here, the color-coding indicates that the classifier is sensitive when the feature values increase.

Using the visual encoding of Figure 3 we can visualize the weights that most influence the classifier decision (DR1) by comparing the position (distance to one) and the induced model's performance. The circles and their color also help to understand how is the influence (DR2) and the PSO execution (DR5) throughout iterations based on the weights assumed by the particles. However, other additional information could help to interpret the classifier decisions and learning processes. Firstly, the distribution of values (DR4) for the feature $\mathcal{F}_k$ help to contrast the classifier performance with the pattern seeing in data samples for a particular class. Lastly, for the non-binary classification problem, it is interesting to know how different weights influence the classifier's confusion with other classes (DR3). Figure 4 shows information for two features. Apart from distribution plots showing the feature values (**c**), we encode the confusion of the classifier after perturbation using proportions of the confused classes (**b**) (indicated by color hue) – notice that higher bars show a greater number of data samples classified with that specific class. The circles in (**a**) encode the information already discussed and exemplified with the scheme of Figure 3. Finally, the horizontal red segment (**d**) shows the best weight (among all of the weights generated by the particle) for the feature concerning the definition of our optimization problem, that is, the minimum weight that induces the higher loss of the performance.

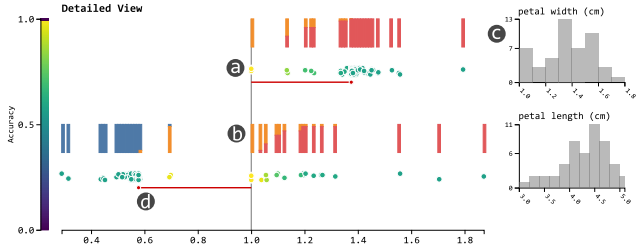

Figure 4: Detailed View for classifier interpretation using PSO. Using the PSO execution encoding (**a**) as explained in Figure 3 we visualized how the weights affect on model's prediction. The stacked bar-charts encode the proportion of classification inside a weight range ($[a, a+\epsilon]$) to assist on the classifier's confusion analysis. Distribution plots (**c**) of the feature values for the class of interest also help with the interpretation of the results. Finally, the horizontal red segment (**d**) also assists in the identification of the best weight for the feature concerning the definition of our optimization function.

One important thing to mention is that the proportion displaying the confusion is not computed using one single weight but a range of weights inside a window. More specifically, we divide the weight space in windows of length ϵ and use all of the weights inside each window to calculate the proportion. Since each perturbation weight generates m classification results – where m is the number of data samples in the test set –, we compute the mean of classification for each class to derive the proportion of the bars in the stacked bar chart.

Lastly, the red circle and the horizontal red segment encode the best result retrieved by the PSO algorithm according to the function defined for our problem, that is, the minimum weight that induced the higher loss of performance.

4.2. Summary View

The visualization design presented in the previous section suffers from visual scalability issues when analyzing a

large dataset (in terms of dimensionality). So, we designed a summary version of the information presented in the detailed view where users can choose which features to explore throughout the interaction. More specifically, in the *Summary View*, we show the relationship between perturbation weights and the model's performance.

To summarize the information contained in the *Detailed View*, we use a variable γ to create windows on the weight axis, as illustrated in Figure 5. Then, each feature weight inside a window induces a performance score (ps) to the model being analyzed. Since the mean of performance score inside a window still lies inside the score range (from 0 to 1 for accuracy), windows are color-coded based on the same color-scale used for the *Detailed View*. Notice that for some weights there is no influence on the model's prediction (see the performance for weights lower than one in Figure 5) while there is an upper bound for how much change on the performance a feature perturbation can induce (see the performance for weights greater than approximately 1.25).

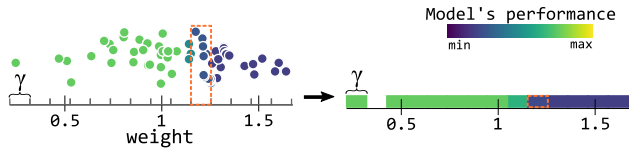

Figure 5: Aggregation of particle weights. Using a range ($[a, a+\gamma]$), the mean of the weights inside the range is encoded by the color-scale. The area highlighted in orange shows how different weights are aggregated in a single block.

Figure 6 shows the results of summarizing information using the discussed strategy. Besides the performance of the classifiers inside the windows defined using γ, the best weight (as the one for the *Detailed View*) is shown through a vertical segment in red. Lastly, the boxes on the right of each axis show which features are being currently inspected on the *Detailed View*. Users can toggle the boxes to show (when the boxes are filled with gray) or hide (when the boxes are filled with white) feature information.

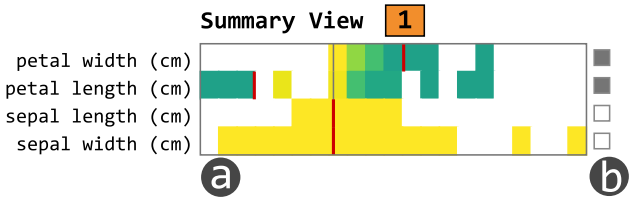

Figure 6: Summary View showing all information about PSO execution in an aggregated way for class 1 (orange). The color of the blocks (**a**) encodes classifier performance (e.g. accuracy) and the red segments show the best weight for each feature. The grey vertical segment shows the weight with no perturbation ($w = 1$). Finally, the boxes on the right of each feature show which features are being inspected at the moment (grey - selected, while - unselected).

4.3. Similarity View

Besides assessing how the perturbations affect a model's performance based on a local perspective, a global view, where we understand how the classification of class a is confused with class b, can be helpful to guide users on the detailed inspection. Notice that confusion is not reflexive, that is, the confusion from a to b is not the same as the confusion from b to a.

Let us define the similarity between two classes a and b (from a to b) as a measure of confusion of the classifier between these classes. Firstly, we must give higher weight to the data points confused with b when the weight induced to such confusion is closer to one. This means we give more interaction to the classes in which little perturbation induces more confusion. So that, the similarity from class a to class b for feature i is defined as it follows:

$$I_i^{a\to b} = \frac{\sum_{winW} (W_{max} - w) \times |\text{argmax}(f(w \times D^{test}_{:,i})) = b|}{\sum_{w \in W} (W_{max} - w) \times |D^{test}|} \tag{4}$$

Equation 4 ranges from 0 to 1, where 1 means that all of the test data samples were classified as b. Notice that $|\text{argmax}(f(w \times D^{test}_{:,i})) = b|$ denotes the number of data points classified as b after multiplying w to the column corresponding to feature i. W_{max} corresponds to the greatest value that a particle can assume in the PSO algorithm (here, we set $W_{max} = 10$). Using that equation, we can define the similarity from a to b as the mean of similarities of each feature, as shown in Equation 5.

$$I^{a\to b} = \frac{1}{m} \sum_{i=1}^{m} I_i^{a\to b}. \tag{5}$$

To visualize the similarity information among all classes, we use a node-link layout together with an encoding inspired on the UpSet (Lex et al., 2014) visualization. As illustrated in Figure 7, every pair of combination is encoded by a cross on the lines of the classes. Then, to communicate the interaction a to b for every pair of (a, b), we use the outer radius of an arc positioned on the crossing between a and b. Notice in Figure 7, the encoding for interaction between two classes follows the direction of the source class to the target, as demonstrated for $I^{1\to 2}$ and $I^{2\to 1}$. For instance, the figure shows the example of a Similarity View for the *Iris* dataset. Notice that, as it is very understanding about this well-known dataset, two classes interact the most and are responsible for errors in the classification.

5. Use cases

In this section, we use the feature perturbations generated by PSO to interpret the model's behavior upon the classification of several datasets. All of the experiments were performed with a computer with the following configuration:

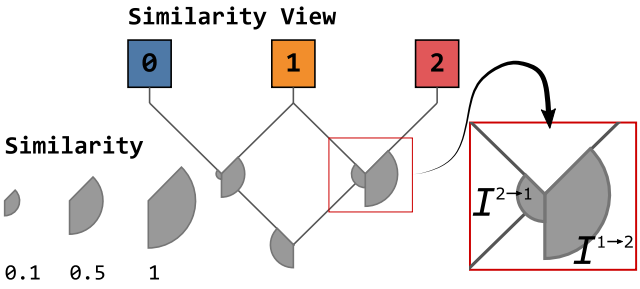

Figure 7: Similarity View encodes the relationship between each pair of classes. The similarity from class a to b ($I^{a\to b}$) is visualized by the outer radius of a radial layout segment. Notice that such an encoding follows the direction of the interaction, as indicated by the red square.

Intel (R) Core(TM) i7-8700 CPU @ 3.20 GHz, 32GB RAM, Windows 10 64 bits. Since the focus is on the interpretation of the results rather than the performance of the classifier, all of the hyper-parameters and details are described in the **Supplementary File**.

5.1. Vertebral Column

In this first use case, we interpret the XGBoost Classifier (Chen and Guestrin, 2016) applied to the *Vertebral Column* (Dua and Graff, 2017) dataset, composed by 310 instances described by six biomechanical features derived from the shape and orientation of the pelvis and lumbar spine: `pelvic incidence`, `pelvic tilt`, `lumbar lordosis angle`, `sacral slope`, `pelvic radius`, and `grade of spondylolisthesis`. The dataset is divided into three classes: class 0 for patients with Hernia, class 1 for patients with Spondylolisthesis – a disturbance of the spine in which a bone (vertebra) slides forward over the bone below it, and class 2 for normal patients. Figure 8 shows the interaction among these three classes. From the Similarity View, we can understand that there is a lot of confusion between classes 0 and 2 induced by the perturbation weights generated by the PSO execution, which indicates that it is a difficult task to determine whether a data sample belongs to either of these two classes. On the other hand, class 1 seems to have very distinctive features from the other two classes, where the confusion is mostly present in the backward form ($I^{0\to1}$ and $I^{2\to1}$).

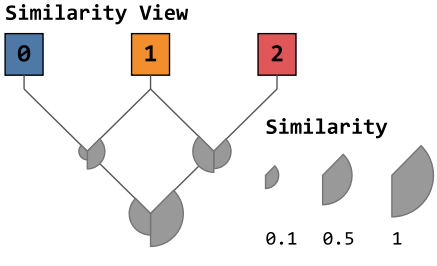

Figure 8: Similarity View shows there is a lot of interaction between classes 0 (blue) and 2 (red). However, class 1 (orange) seems to have very distinctive characteristics.

To inspect how the classes 0 and 2 present such complexity to the classifier, Figure 9 shows the summary result of all features in the dataset and detail of only the most distinctive features for these two classes. The first thing to notice is that the classifier confuses both of the classes with class 1 when `degree_spondylolisthesis` increases – as we show later, the degree of spondylolisthesis is the most determinant feature for patients with Spondylolisthesis. Further that, we can identify that those normal patients (class 2) present lower values for pelvic tilt and higher values for sacral slope and pelvic radius, that is, see how the confusion from class 0 to class 2 increases when the feature `pelvic tilt` is multiplied by a number below one or when the feature `sacral slopse` is multiplied by a number higher than one in Figure 9**a**. On the other hand, the values for these features are found to be the opposite in patients with Hernia (Labelle et al., 2005; Roussouly and Pinheiro-Franco, 2011a,b). We can see that by analyzing the changes induced by the perturbations generated with the PSO algorithm, we could understand a lot of characteristics of the dataset together with the classifier decisions. Here, while the classifier could learn the parameters to be more confident when the features are pushed to the limit (higher or lower values), it seems that the classifier needs improvement to correctly classify data samples that share class boundaries – when feature values are similar.

The class 1, as shown in Figure 10, presents different results only when the perturbations are applied to the degree_spondylolisthesis feature. Notice that how the results are consistent with the Similarity Plot, in which there are very low similarity going from class 1 to the others.

Now, we must verify if the results of the classifier explained by the PSO perturbation weights make sense. That is, to verify if the patients with Spondylolisthesis, the degree of spondylolisthesis are known to be greater. Interestingly, the degree of spondylolisthesis is the most important factor to determine if a patient has Spondylolisthesis or not (Labelle et al., 2005). This makes sense if, with a higher degree, a vertebra bone presents more deviation from a bone below it – which constitutes the Spondylolisthesis. Here, we see that the classifier used `degree_spondylolisthesis` to induce a separation between class 1 and the other two classes 0 and 2.

5.2. Heart disease

In this second case study, we use our approach to inspect and understand the features used by Random Forest Classifier (Breiman, 2001) to differentiate between patients with healthy and unhealthy hearts. The dataset contains two classes: healthy hearts (0) and unhealthy hearts (1). Each data sample is described by the following features: `age`, `sex` (1 - male; 0 - female), `chest pain type`, `resting blood pressure`, `serum cholesterol` (in mg/ml), `fasting blood sugar`, (> 120 mg/ml), `resting electrocardiographic results` (values 0, 1, 2), `maximum heart rate achieved`, `exercise induced angina`, `oldpeak` = ST depression induced by exercise relative to rest, the `slope of the peak exercise` ST segment, `number of major vessels` (0-3), and `thal` (3 - normal, 6 - fixed defect, 7 - reversable defect). Since there are only two classes, the Summary View does not add much information to the analysis,

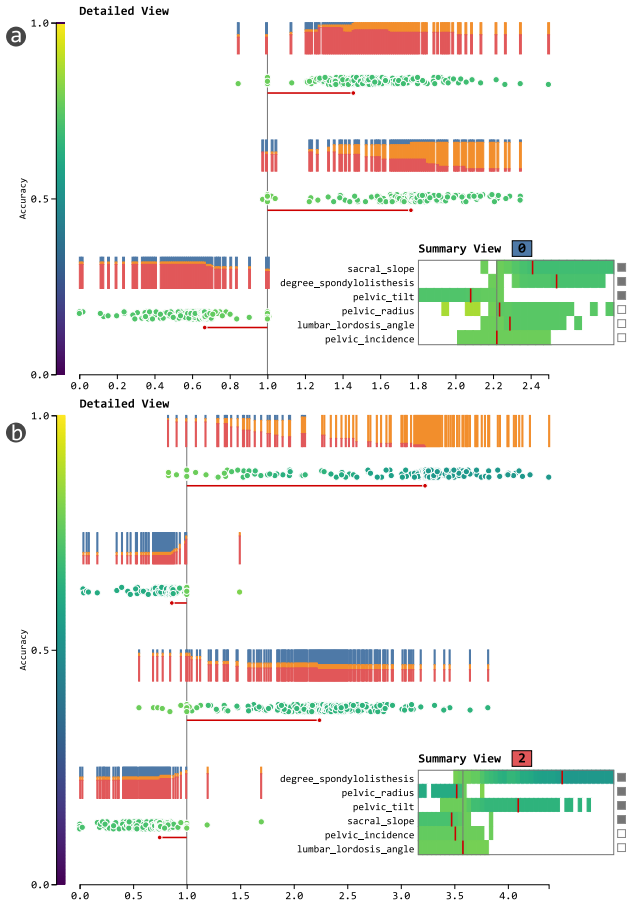

Figure 9: Detailed View for both patients with Hernia (**a**) and normal patients (**b**) shows that except for `degree of spondylolisthesis`, these two classes are highly confused between each other by the classifier.

thus, we omitted it for this case study.

Figure 11 shows the summary and detailed (for some features) of the importance given by the classifier to classify data samples as healthy hearts. The most important feature, that according to the classification model constitutes a healthy heart, is `max. heart rate`. In this case, healthy hearts are seen by the classification model as the ones with moderate rate beat – when the heartbeat gets higher, the model starts to confuse with unhealthy hearts. Then, the `chest pain`, defined by increasing values related to the severity (from 0 to 3), constitutes the second most important feature. As learned by the algorithm, healthy hearts present lower levels of severe pain (see the distribution plot of `chest pain`). The third most influential feature, `thal`, corresponds to an inherited blood disorder (Thalassemia) that causes one's body to have less hemoglobin than normal. Interestingly, heart problems (such as congestive heart failure and abnormal heart rhythms) can be associated with Thalassemia (GJameson JL, 2019; Tha, 2019). Finally, the model also gave importance to the `number of major vessels` feature. This feature indicates how many arteries are visible after a special dye (fluoroscopy) is injected into the blood vessels of the heart. Unlike the previously discussed feature (`thal`), there is a consistency to the algorithm as defining healthy hearts the one with a higher number of visible vessels.

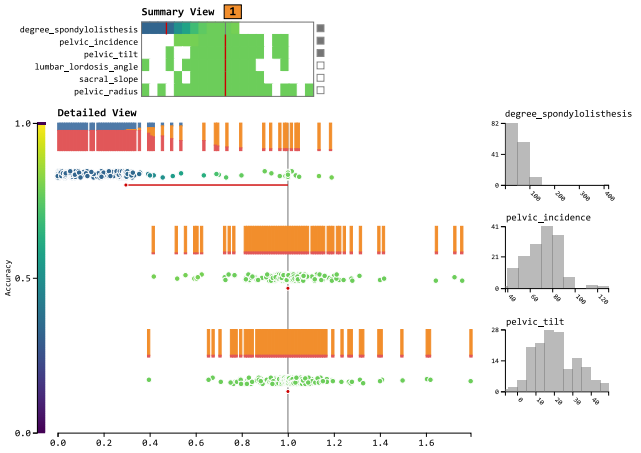

Figure 10: For patients with Spondylolisthesis, the classifier assigned the feature `degree of spondylolisthesis` as the most important feature, meaning that it is responsible for correct classification of such a class.

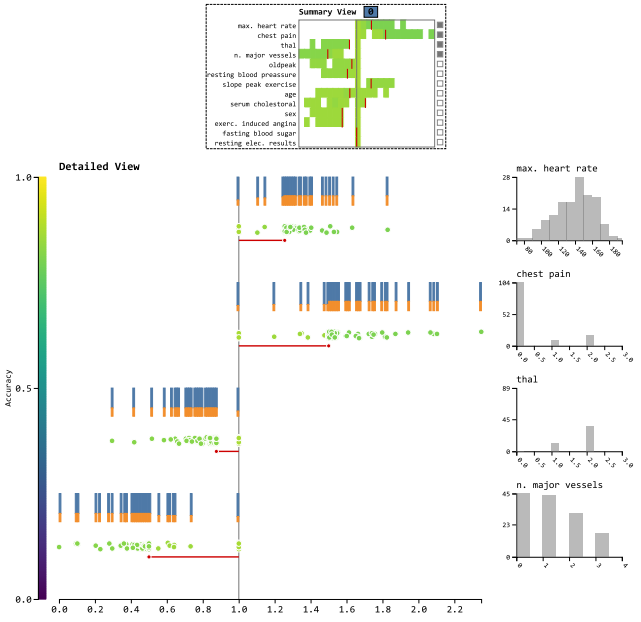

Figure 11: Detailed View for patients with healthy hearts. Our technique correctly shows that the classifier was able to learn that a healthy heart has a moderate heart rate while lower severity in chest pain.

The pattern perceived for unhealthy hearts (class 1) is nearly the opposite of the ones perceived for healthy hearts, as shown in Figure 12. Although the four most important features of class 1 are the same four most important features of class 0, there is a change in the ordering. For the case of `chest pain`, the feature weights are lower than one since unhealthy hearts present more severe chest pain, encode by greater values. The second most important feature, `max. heart rate` shows how unhealthy hearts present a beat rate greater than healthy hearts. Then, the narrowing of blood

vessels, which is related to the number of major vessels, is usually due to arteriosclerosis, a common arterial disease in which increased areas of degeneration and cholesterol deposit plaques form on the inner surfaces of the particles blocking the blood flow (Gersh, 2000; Khatibi and Montazer, 2010). Finally, one must understand the organization of the dataset and the field in which the problem is inserted before taking any particular assumption. That is, besides preventing analysts from defining erroneous hypothesis, carefully analyze the results returned by our algorithm will help one to understand the decisions of the model and the structures of the dataset, such as data mismatch. For example, after receiving a high score in a classification task, one could analyze and understand if there is bias in the features learned by the models.

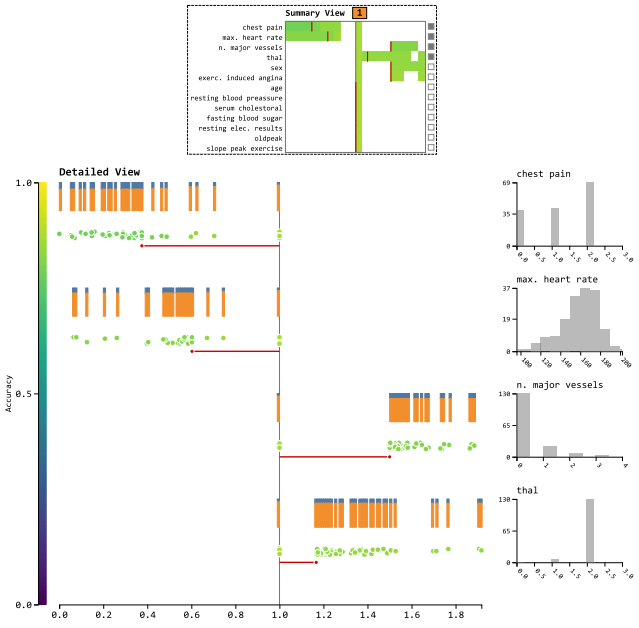

Figure 12: As discussed for patients with healthy hearts, our approach consistently shows that the classifiers assigned the data samples as unhealthy when they have higher `max. heart rate` and severe `chest pain`. Besides that, unhealthy hearts also seem to present a lower number of visible vessels.

5.3. Diabetes

In this final case study, we interpret CatBoost (Dorogush et al., 2018) classifier's prediction on a dataset containing 768 data samples of patients described by eight medical features concerning the presence of diabetes: 268 data samples of patients with diabetes (class 0) and 500 data samples of non-diabetic patients (class 1). The features used for classification are: `preg` (number of times pregnant), `plas` (plasma glucose concentration after 2 hours in an oral glucose tolerance test), `pres` (diastolic blood pressure), `skin` (triceps skin fold thickness (mm)), insulin (2-hour serum insulin (mU/ml)), `mass` (body mass index (weight in kg/(height in m)^2), `pedi` (diabetes pedigree function), `age` (age in years). As we discussed for the previous case study, we do not rely on the similarity view to extract information of this dataset since it consists in a binary classification.

Figure 13 shows the result for both classes: patients with diabetes 0 and patients without diabetes 1. Here, we focus on the three most important features returned by our technique for both of the classes, in which three importance ordering is the same: plasma glucose concentration (`plas`), body mass index (`mass`), diabetes pedigree function (`pedi`). Recalling that plasma glucose concentration (or simply blood sugar level) is a well-known indicator of prediabetes when the levels are high (Abdul-Ghani and DeFronzo, 2009), our approach consistently shows how a healthy patient (class 0) must increase his blood sugar levels to become diabetic – notice that it consistently identified this feature as the most important one. Looking at the distribution of values for `plas` for patients without diabetes, there is a concentration of around 100. Further, the second most important feature also shows that increased body fat will result in patients with a higher probability of having diabetes. Interestingly, such a result is consistent with the literature since an increase in body fat is generally associated with an increase in the risk of metabolic diseases such as Type 2 diabetes mellitus (Bays et al., 2007). Finally, the diabetes pedigree function feature (`pedi`) also illustrates the applicability of our visualization design and method of model explanation. Since such a feature measures the likelihood of diabetes based on family history, it is clear that higher pedigree function valuers will induce more chances for patients to be classified as diabetic.

Interestingly, looking at the feature perturbations of class 1, the algorithm consistently imposed weight that would result in the levels of features shown by healthy patients. That is, normal levels for `plas`, `mass`, and `pedi`.

6. Numerical evaluation

In this section, we compare our proposal against two well-established techniques at their ability to retrieve important features. The techniques were evaluated using the Keep Absolute (Lundberg et al., 2020) metric, which measures the impact of selected features on the model accuracy. In this case, the impact is measured by adding the most important features from the dataset and training the model. Figure 14 exemplifies the strategy for the Keep Absolute metric, note that as more features are added, the accuracy of the model on cross-validation (5 fold) setting increases. Also, to further validate our methodology, we evaluate it against feature selection algorithms (Permutation Importance (Altmann et al., 2010), ANOVA (Pedregosa et al., 2011), Mutual Information (Ross, 2014), and Recursive Feature Selection (Guyon et al.)) using the same metric.

Table 2 shows that our technique was able to give the best results only for the **Wine** and **Iris** datasets. The latter, however, is a relatively simple data set that do not require much robustness from the techniques, as seen in the results. Although our technique was not able to uncover the best ordering of features according to the **Keep Absolute** metric, it presented better results than all of the feature selection algorithms for most of the dataset (for **Vertebral** dataset, how-

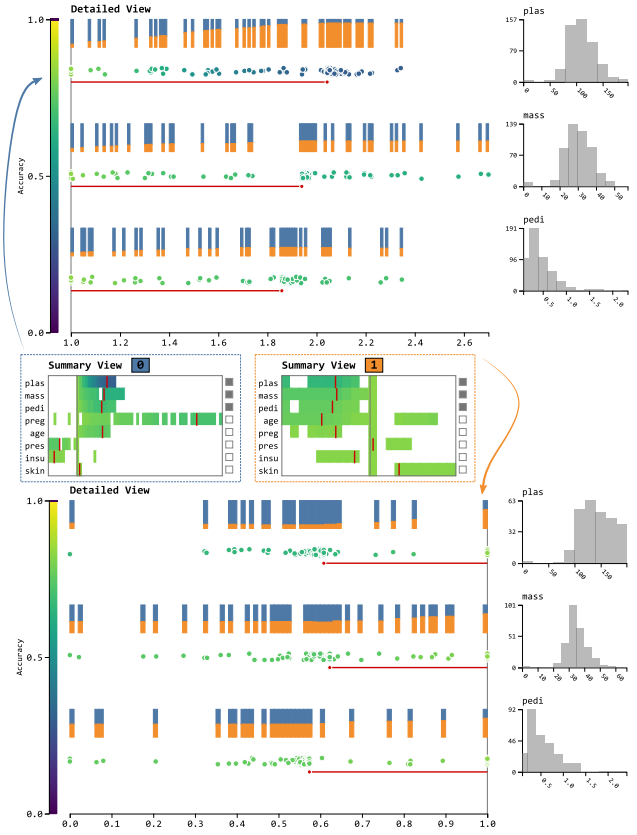

Figure 13: Detailed View showing that the classifier gave the same importance ordering for both non-diabetic and diabetic patients. However, we can see that the weights generated by the PSO algorithm are consistent with classes. For instance, non-diabetic patients (class 0 - blue) need to have `plas` and `mass` increased to be classified as diabetic.

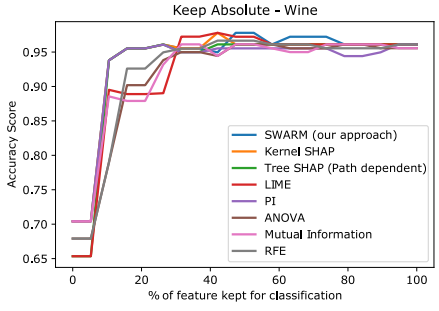

Figure 14: F1-Score when varying the number of features kept for classification.

ever, our technique presented a difference of only 0.0009 to the best result). Focusing on the model explanation techniques, our method was able to provide explanations that resulted in better feature ordering based on importance better than at least one method for all of the datasets (of course, excluding **Iris** dataset). From this numerical analysis, we could see that our method performs very similar to well-established methods in the literature. Besides that, looking at the differences among the scores in Table 2, it is possible to note that there are only a few differences in the ordering imposed by the algorithms.

As seen for XGBoost Classifier in Table 2, we also evaluated the techniques by using Random Forest Classifier. Table 3 summarizes the results. We can see from the table that although our technique was not able to present the best results for all of the datasets, it presented the second-best score for *Indian Liver*, *Heart*, and *Breast Cancer*. Finally, the significative difference was only reported for the *Wine* dataset.

To illustrate the stability of our method in returning stable results compared with other techniques, we show the rankings of each method according to XGBoost Classifier and Random Forest Classifier. Based on the results for XGBoost Classifier in Table 5, our method has a mean ranking of 2.33 (underlined) while losing only for TreeSHAP (in bold) with a mean ranking of 2.17. Finally, the results in Table 4 for Random Forest Classifier show that our method got the third position with 0.17 behind the first two methods (KernelSHAP and LIME, both with a mean ranking of 3.00). These results confirm the stability of the proposed technique.

7. Discussions

In this work, we presented a novel model interpretation approach using Particle Swarm Optimization. Our method, defined as a global interpretation approach, can be employed as a local approach by feeding one instance at a time to our PSO implementation. The main strength of our method is its simplicity and ability to be adapted to various performance scores and optimization functions, which is a strength of our method over the state of the art algorithms LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017).

In the numerical evaluation, our method performed better than feature selection algorithms on tasks that measure the performance on defining the proper importance ordering of the input features. Although it was not able to uncover the best results for all of the datasets used in training, our method provided stable results on higher positions.

One particularly important thing to mention is the need for preprocessing for some datasets to use the approach proposed with PSO perturbation weights. For instance, a ϵ value needs to be added to those datasets where there are zero entries. Such a preprocessing step is important since our explanations consist of using instructions in the form $w \times X_{ij}$, and having $X_{ij} = 0$ will make the particles of the PSO algorithm to stay still and not make any progress. Adding ϵ to X will solve such an problem – here we used $\epsilon = 1$. Notice that this preprocessing is only applied to help the algorithm, and the visualization aspects are created based on the original data.

Another preprocessing step is necessary due to the patterns of perturbation used to compute the explanations. For example, consider a z-score normalization on the dataset columns. If an important characteristic to define a class relates to the signal of a particular feature, our perturbation mechanism will not change classifications since the perturbations will not change signals. So that, we resolve such an issue by adding the value $x = |min(X), 0|$ – where X is the input dataset – to the input dataset, which will force all of

Table 2
Area Under the Curve (AUC) for Keep Absolute metric. The curve is calculated by varying the number of features kept for classification – note that the best scores (closest to one) are highlighted in bold.

	Vertebral	Indian Liver	Heart	Wine	Breast Cancer	Iris
SWARM	0.8071	0.6687	0.8172	**0.9413**	0.9639	**0.9615**
Kernel SHAP	0.8047	0.6672	0.8108	0.9393	0.9628	**0.9615**
Tree SHAP (PD)	0.8056	**0.6692**	0.8194	0.9338	**0.9650**	**0.9615**
LIME	0.8074	0.6658	0.8197	0.9253	0.9645	**0.9615**
PI	**0.8080**	0.6636	**0.8209**	0.9333	0.9637	**0.9615**
ANOVA	0.7849	0.6632	0.7940	0.9187	0.9510	**0.9615**
M. Information	0.7933	0.6603	0.7922	0.9239	0.9505	**0.9615**
RFE	0.7960	0.6585	0.7795	0.9246	0.9617	0.9560

Table 3
Area Under the Curve (AUC) for Keep Absolute metric and Random Forest Classifier.

	Vertebral	Indian Liver	Heart	Wine	Breast Cancer	Iris
SWARM	**0.8155**	0.6756	0.8178	0.9208	0.9563	0.9541
Kernel SHAP	0.8128	**0.6820**	0.8165	**0.9433**	0.9537	0.9535
LIME	0.7931	0.6723	**0.8212**	0.9332	0.9559	0.9551
PI	0.8144	0.6666	0.8115	0.9283	**0.9566**	0.9551
ANOVA	0.7916	0.6681	0.7897	0.9282	0.9490	0.9534
M. Information	0.7978	0.6652	0.7906	0.9332	0.9474	0.9551
RFE	0.7966	0.6706	0.7740	0.9407	0.9514	**0.9602**

the entries to be positive.

Finally, while we demonstrated our technique ability to explain classifiers' predictions, it can be easily modified to work with regression models. In this case, the PSO algorithm can be used to produce perturbation weights that would maximize an error metric (e.g., mean squared error) while reducing the distance between the position with no perturbation ($w = 1$).

Table 4
Ranking for the results presented in Table 2.

	Vertebral	Indian Liver	Heart	Wine	Breast Cancer	Iris	Mean ranking	St.d.
SWARM	3	2	4	1	3	1	2.33	1.21
Kernel SHAP	5	3	5	2	5	1	3.50	1.76
Tree SHAP (PD)	4	1	3	3	1	1	**2.17**	1.33
LIME	2	4	2	5	2	1	2.67	1.51
PI	1	5	1	4	4	1	2.67	1.86
ANOVA	8	6	6	8	7	1	6.00	2.61
M. Information	7	7	7	7	8	1	6.17	2.56
RFE	6	8	8	6	6	8	7.00	1.10

Table 5
Ranking for the results presented in Table 3.

	Vertebral	Indian Liver	Heart	Wine	Breast Cancer	Iris	Mean ranking	St.d.
SWARM	1	2	2	7	2	5	3.17	2.32
Kernel SHAP	3	1	3	1	4	6	**3.00**	1.90
LIME	6	3	1	3	3	2	**3.00**	1.67
PI	2	6	4	5	1	2	3.33	1.97
ANOVA	7	5	6	6	6	7	6.17	0.75
M. Information	4	7	5	3	7	2	4.67	2.07
RFE	5	4	7	2	5	1	4.00	2.19

7.1. Limitations

The main limitation of our work is the run-time execution since we have to execute the PSO algorithm for each pair of feature/class. Although we plan to investigate faster optimization algorithms in further works, a subset of the data used to feed our algorithm could also be used to decrease the execution time. Besides that, for a dataset with higher dimensionality, we could use a subset of the most important features.

Another limitation of our work is related to the need for preprocessing steps as discussed above, which could break the desire patterns introduced by users or during dataset acquisition. In future works, we plan to investigate alternatives to reduce the dependency on these steps. For example, instead of multiplying a feature by an optimized value, each data sample could receive a random perturbation (negative or positive) created using a Gaussian function with an optimized kernel. So, less important features could receive a strong perturbation without a relevant change in the model's performance. On the other hand, an essential feature would cause a loss of performance even with a weak perturbation.

7.2. Implementation

Our technique was implemented using Python, while the prototype system with the visualization uses D3.js (Bostock et al., 2011). We also created a Python API (the API will be available after publication) with lightweight visualizations similar to presented here so our approach can be employed by users (such as in notebooks).

8. Conclusion

As machine learning algorithms take over traditional approaches to solve problems, their reliability becomes important, even more in applications where wrong decisions can lead to serious issues. Understanding a model's decisions is now an important process of the development and execution of machine learning strategies, as a way to assess if such decisions make sense to domain experts.

In this work, we proposed a novel model-agnostic approach to interpret any classification algorithm by using particle swarm optimization. Throughout several case studies, we validated our approach on its ability to explain classifiers' decisions – we also showed that the results presented here make sense with the domain literature. Finally, our methodology was also numerically evaluated on its ability to retrieve an ordering to the input features according to their importance. The results showed that our method can return very stable orderings with good results for all of the datasets – a result that only one state of the art method was able to provide.

In future works, we plan to investigate our methodology on regression tasks, as well as using other optimization algorithms to find the perturbation weights. Besides that, since SHAP approximate Shapley values, we also want to investigate nature-inspired optimization techniques to approximate them.

Acknowledgements

This work was supported by Fundação de Amparo à Pesquisa (FAPESP) – grant #2018/17881-3, and by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) – grant #88887.487331/2020-00.

References

, 2019. National heart, lung, and blood institute. Harrison's Principles of Internal Medicine. URL: https://www.nhlbi.nih.gov/health-topics/thalassemias.

Abdul-Ghani, M.A., DeFronzo, R.A., 2009. Plasma glucose concentration and prediction of future risk of type 2 diabetes. Diabetes care 32, 194–198.

Altmann, A., Tolosi, L., Sander, O., Lengauer, T., 2010. Permutation importance: a corrected feature importance measure. Bioinform. 26, 1340–1347.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.R., 2010. How to explain individual classification decisions. J. Mach. Learn. Res. 11, 1803–1831.

Balagopalan, A., Novikova, J., Rudzicz, F., Ghassemi, M., 2018. The effect of heterogeneous data for alzheimer's disease detection from speech. ArXiv abs/1811.12254.

Bansal, G., Nushi, B., Kamar, E., Weld, D.S., Lasecki, W.S., Horvitz, E., 2019. Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff, in: AAAI.

Bays, H.E., Chapman, R.H., Grandy, S., 2007. The relationship of body mass index to diabetes mellitus, hypertension and dyslipidaemia: comparison of data from two national surveys. International journal of clinical practice 61, 737–747.

Bostock, M., Ogievetsky, V., Heer, J., 2011. D3 data-driven documents. IEEE Transactions on Visualization and Computer Graphics 17, 2301–2309.

Breiman, L., 2001. Random forests. Machine Learning 45, 5–32.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system, Association for Computing Machinery, New York, NY, USA. p. 785–794.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1800–1807.

Clavien, G., Alberti, M., Pondenkandath, V., Ingold, R., Liwicki, M., 2019. Dnnviz: Training evolution visualization for deep neural network, in: 2019 6th Swiss Conference on Data Science (SDS), pp. 19–24.

Craven, M.W., Shavlik, J.W., 1995. Extracting tree-structured representations of trained networks, in: NIPS.

Datta, A., Sen, S., Zick, Y., 2016. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems, in: 2016 IEEE Symposium on Security and Privacy (SP), pp. 598–617.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. URL: http://arxiv.org/abs/1810.04805.

Dorogush, A.V., Ershov, V., Gulin, A., 2018. Catboost: gradient boosting with categorical features support. ArXiv abs/1810.11363.

Doshi-Velez, F., Kim, B., 2017. Towards a rigorous science of interpretable machine learning. arXiv URL: https://arxiv.org/abs/1702.08608.

Doshi-Velez, F., Kim, B., 2018. Considerations for evaluation and generalization in interpretable machine learning.

Dua, D., Graff, C., 2017. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S., 2017. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118.

Gersh, B.J., 2000. Mayo Clinic Heart Book. 2nd ed., HarperCollins.

GJameson JL, e.a., 2019. Disorders of hemoglobin. Harrison's Principles of Internal Medicine. URL: https://www.nhlbi.nih.gov/health-topics/thalassemias.

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., Cristianini, N., . Gene selec-

tion for cancer classification using support vector machines, in: Machine Learning, p. 2002.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.

Hinterreiter, A., Ruch, P., Stitz, H., Ennemoser, M., Bernard, J., Strobelt, H., Streit, M., 2020. Confusionflow: A model-agnostic visualization for temporal analysis of classifier confusion. IEEE Transactions on Visualization and Computer Graphics , 1–1.

Hong, S.R., Hullman, J., Bertini, E., 2020. Human factors in model interpretability: Industry practices, challenges, and needs. Proc. ACM Hum.-Comput. Interact. 4.

Kahng, M., Andrews, P.Y., Kalro, A., Polo Chau, D.H., 2018. Activis: Visual exploration of industry-scale deep neural network models. IEEE transactions on visualization and computer graphics 24, 88—97.

Khatibi, V., Montazer, G.A., 2010. A fuzzy-evidential hybrid inference engine for coronary heart disease risk assessment. Expert Systems with Applications 37, 8536 – 8542.

Krause, J., Perer, A., Ng, K., 2016. Interacting with predictions: Visual inspection of black-box machine learning models, in: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ACM. p. 5686–5697.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 25. Curran Associates, Inc., pp. 1097–1105.

Kulesza, T., Burnett, M., Wong, W.K., Stumpf, S., 2015. Principles of explanatory debugging to personalize interactive machine learning, in: Proceedings of the 20th International Conference on Intelligent User Interfaces, ACM. p. 126–137.

Labelle, H., Roussouly, P., Berthonnaud, E., Dimnet, J., Obrien, M., 2005. The importance of spino-pelvic balance in l5–s1 developmental spondylolisthesis: A review of pertinent radiologic measurements. Spine 30, S27–S34.

Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R., Pfister, H., 2014. Upset: Visualization of intersecting sets. IEEE Transactions on Visualization and Computer Graphics 20, 1983–1992.

Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I., 2020. From local explanations to global understanding with explainable ai for trees. Nature Machine Intelligence 2, 2522–5839.

Lundberg, S.M., Lee, S.I., 2017. A unified approach to interpreting model predictions, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 4765–4774.

Luo, G., 2016. Automatically explaining machine learning prediction results: a demonstration on type 2 diabetes risk prediction. Health Information Science and Systems 4.

Marcilio-Jr, W.E., Eler, D.M., Garcia, R.E., Correia, R.C.M., Silva, L.F., 2020. A hybrid visualization approach to perform analysis of feature spaces, in: Latifi, S. (Ed.), 17th International Conference on Information Technology–New Generations (ITNG 2020), Springer International Publishing, Cham. pp. 241–247.

Meijer, A., Wessels, M., 2019. Predictive policing: Review of benefits and drawbacks. International Journal of Public Administration 42, 1031–1039.

Modarres, C., Ibrahim, M., Louie, M., Paisley, J.W., 2018. Towards explainable deep learning for credit lending: A case study. ArXiv abs/1811.06471.

Olsson, A.E., 2010. Particle Swarm Optimization: Theory, Techniques and Applications. Nova Science Publishers, Inc., USA.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in python. Journal of machine learning research 12, 2825–2830.

Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L., 2018. Deep contextualized word representations, in: Proc. of NAACL.

Pezzotti, N., Höllt, T., Van Gemert, J., Lelieveldt, B.P.F., Eisemann, E., Vilanova, A., 2018. Deepeyes: Progressive visual analytics for designing deep neural networks. IEEE Transactions on Visualization and Computer Graphics 24, 98–108.

Rauber, P.E., Fadel, S.G., Falcão, A.X., Telea, A.C., 2017. Visualizing the hidden activity of artificial neural networks. IEEE Transactions on Visualization and Computer Graphics 23, 101–110.

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, San Francisco, CA, USA, August 13-17, 2016, pp. 1135–1144.

Ross, B.C., 2014. Mutual information between discrete and continuous data sets 9, e87357.

Roussouly, P., Pinheiro-Franco, J.L., 2011a. Biomechanical analysis of the spino-pelvic organization and adaptation in pathology. European Spine Journal 20, 609–618.

Roussouly, P., Pinheiro-Franco, J.L., 2011b. Sagittal spino-pelvic balance is a crucial analysis for normal and degenerative spine. Eur Spine J 20, 556–557.

Smilkov, D., Carter, S., Sculley, D., Viégas, F.B., Wattenberg, M., 2017. Direct-manipulation visualization of deep networks. ArXiv abs/1708.03788.

Strumbelj, E., Kononenko, I., 2013. Explaining prediction models and individual predictions with feature contributions. Knowledge and Information Systems 41, 647–665.

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9.

Zhang, J., Wang, Y., Molino, P., Li, L., Ebert, D.S., 2019. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. IEEE Transactions on Visualization and Computer Graphics 25, 364–373.