Using Sensitivity Analysis and Visualization Techniques to Open Black Box Data Mining Models

Paulo Cortez[a], Mark J. Embrechts[b]

[a] *Centro Algoritmi, Departamento de Sistemas de Informação, Universidade do Minho,*
Campus de Azurém, 4800-058 Guimarães, Portugal
[b] *Department of Industrial and Systems Engineering*
Rensselaer Polytechnic Institute, CII 5129, Troy, NY 12180 - USA

Abstract

In this paper, we propose a new visualization approach based on a Sensitivity Analysis (SA) to extract human understandable knowledge from supervised learning black box data mining models, such as Neural Networks (NN), Support Vector Machines (SVM) and ensembles, including Random Forests (RF). Five SA methods (three of which are purely new) and four measures of input importance (one novel) are presented. Also, the SA approach is adapted to handle discrete variables and to aggregate multiple sensitivity responses. Moreover, several visualizations for the SA results are introduced, such as input pair importance color matrix and variable effect characteristic surface. A wide range of experiments was performed in order to test the SA methods and measures by fitting four well-known models (NN, SVM, RF and decision trees) to synthetic datasets (five regression and five classification tasks). In addition, the visualization capabilities of the SA are demonstrated using four real-world datasets (e.g., bank direct marketing and white wine quality).

Keywords: Sensitivity Analysis, Visualization, Input Importance, Supervised Data Mining, Regression, Classification.

Email addresses: `pcortez@dsi.uminho.pt` (Paulo Cortez), `embrem@rpi.edu` (Mark J. Embrechts)

1. Introduction

Data Mining (DM) aims to extract useful knowledge from raw data. Interest in this field arose due to the advances of Information Technology and rapid growth of business and scientific databases [15]. These data hold valuable information such as trends and patterns, which can be used to improve decision making [30]. Two important DM tasks are classification and regression. Both tasks use a supervised learning paradigm, where the intention is to build a data-driven model that learns an unknown underlying function that maps several input variables to one output target.

Several learning models/algorithms are available for these tasks, each one with its own advantages. In a real-world setting, the value of a supervised DM model may depend on several factors, such as predictive capability, computational requirements and explanatory power. Often, it is important to have DM models with high predictive capabilities on unseen data. Computational effort and memory requirements are particularly relevant when dealing with vast datasets or real-time systems. This work focuses primarily on the explanatory power aspect, which relates to the possibility of extracting human understandable knowledge from the DM model. Such knowledge is important to determine if the obtained model makes sense to the domain experts and if it unveils potentially useful, interesting or novel information [15][4]. Increasing model interpretability allows for better understanding and trust of the DM results by the domain users [28] and this is particularly relevant in critical applications, such as control or medicine.

There is a wide range of “black box” supervised DM methods, which are capable of accurate predictions, but where obtained models are too complex to be easily understood by humans. This includes methods such as: Neural Networks (NN) (e.g., multilayer perceptrons and radial basis-functions) [18], Support Vector Machines (SVM) and other kernel-based methods [10], and ensembles, including Random Forests (RF) [2], where multiple models are combined to achieve a better predictive performance [11]. Recent examples of successful applications of these black box methods are: network intrusion detection using NN [16], wine quality prediction using SVM [7] and text sentiment classification (e.g., positive/negative movie-review identification) using ensembles of SVM and other DM methods [34].

To increase interpretability from black box DM models, there are two main strategies: extraction of rules and visualization techniques. The extraction of rules is the most popular solution [29][26][23]. However, such extraction is often based on a simplification of the model complexity, hence leading to rules that do not accurately represent the original model. For instance, a pedagogical technique was adopted in [27] within the intensive-care medicine domain to extract the relationships between the inputs and outputs of a NN classifier using a decision tree. While producing more understandable rules, decision trees discretize the classifier separating hyperplane, thus leading to information loss. Regarding the use of visualization techniques, the majority of these methods address aspects related to the multidimensionality of data and the use of visualization for black box DM models is more scarce [21]. Regarding the latter approach, some graphical methods were proposed, such as: Hinton and Bond diagrams for NN [9]; showing NN weights and classification uncertainty [31]; and improving the interpretability of kernel-based classification methods [5]. Yet, most of these graphical techniques are specific to a given learning method or DM task.

Our visualization approach to open DM models is based on a Sensitivity Analysis (SA), which is a simple method that performs a pure black box use of the fitted models by querying the fitted models with sensitivity samples and recording the obtained responses [25]. Thus, no information obtained during the fitting procedure is used, such as the gradient of the NN training or importance attributed to the splitting variable of a RF, allowing its universal application. In effect, while initially proposed for NN, SA can be used with virtually any supervised learning method, such as partial least squares [12] and SVM [7].

In [20], a computationally efficient one-dimensional SA (1D-SA) was proposed, where only one input is changed at the time, holding the remaining ones at their average values. Later, in [13] a two-dimensional SA (2D-SA) variant was presented. In both studies, only numerical inputs and regression tasks were modeled. Moreover, SA has been mostly used as a variable/feature selection method, where the method is used to select the least relevant feature that is deleted in each iteration of a backward selection [25][12][5][7].

The use of SA to open black box models was recognized in [20] but more explored in [13], [21] and [8]. In [13], the proposed 2D-SA was used to show the effects of two input variables on the DM model, with the importance of these pair of inputs being measured by the simple output range measure. In [21], a genetic algorithm was used to search for interesting output responses related with one (2D plot) or two input (3D plot) variables. Yet, the study was focused on visualizing the individual predictions of an ensemble of models, where the intention was to check if the distinct individual predictions were similar, in conjunction with other criteria, such as the simpler output range measure. More recently, a Global SA (GSA) algorithm was presented in [8], capable of performing a F-dimensional SA for both regression and classification tasks, although with a high computational cost.

In this paper, we extend and improve our previous work [8], leading to a coherent SA framework capable of handing any black box supervised model, including ensembles, and applicable to both classification and regression tasks. The main contributions are:

i) we present three novel and computationally efficient SA methods (DSA, MSA and CSA), comparing these with previous SA algorithms (1D-SA [20] and GSA [8]);

ii) we propose a new SA measure of input importance (AAD), test it against three other measures, and present a more informative sensitivity measure pair for detecting 2D input relevance;

iii) we adapt the SA methods and measures for handling discrete variables and classification tasks;

iv) we propose novel functions for aggregating multiple sensitivity responses, including a 3-metric aggregation for 1D regression analysis and a fast aggregation strategy for input pair (2D) analysis;

v) we present new synthetic datasets (four regression and five classification tasks) for evaluating input importance;

vi) we present useful visualization plots for the SA results: input importance bars, color matrix, variable effect characteristic curve, surface and contour;

vii) we explore three black box (NN, SVM and RF) and one white box (decision tree) models to test the SA capabilities and show examples of how SA can open the black box in four real-world tasks.

The paper is organized as follows. First, we present the SA approaches, visualization techniques, learning methods and datasets adopted in Section 2. Then, in Section 3 the proposed methods are tested in both synthetic and real-world datasets. Finally, conclusions are summarized in Section 4.

2. Materials and methods

2.1. Sensitivity methods

A supervised DM model is fit to a dataset, or training data, made up of N examples of M input variables and one output target (y). Let $\hat{y}$ denote the value predicted by the model for one example or data sample ($\mathbf{x}$) and let P be the function used to build the model's responses, i.e., $\hat{y} = P(\mathbf{x})$. In general, the sensitivity methods work by varying an input variable $\mathbf{x}_a$ through its range with L levels, under a regular sequence from the minimum to the maximum value. Let x_{a_j} denote the j-th level of input $\mathbf{x}_a$. For example, if $L = 5$ and $\mathbf{x}_a$ ranges within $[0, 1]$, then $x_{a_j} \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$. In this paper, we analyze five sensitivity methods, where the last three are novel algorithms:

1D-SA [20] – This method works by considering a given baseline vector $\mathbf{b}$. Typically, $\mathbf{b}$ contains the mean or median values of each input, although any other vector can be used. For instance, input values of an interesting example or mean values for a particular cluster, such as patients younger than eighteen. Then, it cycles through all $\{\mathbf{x}_a : a \in \{1, ..., M\}\}$ inputs. For each input, L input examples are built using all $\mathbf{b}$ values except $\{x_{a_j} : j \in \{1, ..., L\}\}$. Let the respective model responses be denoted by $\hat{\mathbf{y}}_a = \{\hat{y}_{a_j} : j \in \{1, ..., L\}\}$, where $\hat{y}_{a_j}$ represents the response for x_{a_j}. These responses are stored and used to compute a sensitivity measure of the input or for visualization purposes. The computational complexity is $\mathcal{O}(M \times L \times P)$.

GSA [8] – Similar to 1D-SA, except that this method uses a set of F features that simultaneously vary with L levels. The number of simultaneous sensitivity variables ($\#F$) can range from 1 (equal to 1D-SA) to M (in such case a M-th SA dimensionality, denoted here as MD, is used). When compared with 1D-SA, GSA ($\#F > 1$) is more suited to capture interactions of inputs. However, GSA requires more computation, with a complexity of $\mathcal{O}(L^{\#F} \times P)$.

Data-based SA (DSA) – Similar to 1D-SA, except that this method uses several training samples instead of the baseline vector. The idea is to capture input interactions (as GSA) but with less computational effort. First, the SA dataset is composed of N_s random samples taken from the original dataset. In the SA dataset, all $\mathbf{x}_a$ values are replaced by x_{a_j} and the respective responses are stored. Then, the previous step is repeated using a different j value. This procedure is repeated for all input variables, thereby resulting in a complexity of the order $\mathcal{O}(M \times L \times N_s \times P)$, where N_s is the length of the training samples.

Monte-Carlo SA (MSA) – Similar to DSA, except that the N_s random samples are not taken from the dataset but are instead built from a uniform distribution, in a continuous (MSA$_c$) or discrete space (MSA$_d$, where L different levels are considered for all inputs). The computation effort is same as DSA: $\mathcal{O}(M \times L \times N_s \times P)$.

Cluster-based SA (CSA) – First, this method stores all predicted values for the whole dataset, and then cycles through all inputs. For each input $\mathbf{x}_a$, L clusters are defined, according to a very fast procedure, where a regular sequence with $L+1$ levels, from the minimum to maximum of the $\mathbf{x}_a$ values is created. Next, L ordered clusters are defined, where each cluster contains all data samples whose $\mathbf{x}_a$ values are within two levels of the regular sequence. For example, when $L = 5$ and $\mathbf{x}_a \in [0,1]$ the sequence $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ is used to create 5 clusters, where the first cluster contains the data samples whose $\mathbf{x}_a$ values are within $[0.0, 0.2[$. Finally, the sensitivity values ($\hat{y}_{a_j}$) are set as the predictions related with the examples that belong to cluster j. Typically, CSA is faster than DSA, since the complexity is $\max(\mathcal{O}(N \times P), \mathcal{O}(M \times L))$.

2.2. Sensitivity measures of input importance

All SA methods described previously (Section 2.1) query a fitted DM model in order to obtain a set of sensitivity responses ($\hat{\mathbf{y}}_a$). These responses can be used to measure input importance. The rational is that a relevant input ($\mathbf{x}_a$) should produce substantial output changes when varying its input levels. Such input relevance is quantified by using a sensitivity measure.

In [20], three sensitivity measures were presented for continuous outputs, namely range (S_r), gradient (S_g) and variance (S_v). In this paper, we also propose the purely new Average Absolute Deviation (AAD) from the median (S_d), which is a measure of data dispersion that is less sensitive to outliers when compared with the variance. For input $\mathbf{x}_a$, the four measures are:

$$\begin{array}{llr} S_r = \max(\hat{y}_{a_j} : j \in \{1, ..., L\}) - \min(\hat{y}_{a_j} : j \in \{1, ..., L\}) & \text{(range)} \\ S_g = \sum_{j=2}^{L} |\hat{y}_{a_j} - \hat{y}_{a_{j-1}}|/(L-1) & \text{(gradient)} \\ S_v = \sum_{j=1}^{L} (\hat{y}_{a_j} - \overline{y}_a)^2/(L-1) & \text{(variance)} \\ S_d = \sum_{j=1}^{L} |\hat{y}_{a_j} - \widetilde{y}_a|/L & \text{(AAD)} \end{array} \quad (1)$$

where $\overline{y}_a$ and $\widetilde{y}_a$ denote the mean and median of the responses, respectively. The gradient is the only measure that is dependent on the order of the sensitivity responses. Such property may be a disadvantage when addressing nominal input variables, whose level change values may lead to non-smooth response changes, or random sample sensitive methods (e.g., MSA or DSA).

For all measures, the higher the value, the more relevant is the input $\mathbf{x}_a$. Thus, the relative importance (r_a) can be given by [7]:

$$r_a = \varsigma_a / \sum_{i=1}^{M} \varsigma_i \tag{2}$$

where ς_a is the sensitivity measure for $\mathbf{x}_a$ (e.g., range). In this work, the importance values are denoted by vector $\mathbf{r} = (r_1, r_2, ..., r_M)$.

For demonstration purposes, Figure 1 shows sensitivity measures for five hypothetical response curves: A, B, C, D and E. The flat curve (A) produces a zero value for all measures. However, the graph shows different rankings for the other curves. According to the range, B, C and D curves have an equal influence in the responses and higher than curve E ("saw" shape). The gradient ranks higher both D and E, followed by both C and B. The variance favors B, C and D, followed by E. Finally, the average deviation ranks C and D at first place, followed by B and E. Intuitively, curve D should denote an higher influence than E. Also, the average effect of curve C should be higher when compared with curve B. Under this rational, the absolute deviation is the best choice.

In both [13] and [21], the importance of an input pair of variables was measured using a simple measure: the global range of the output responses produced by the pair. However, this simple measure may led to misleading conclusions. For instance, when considering the synthetic psin regression task (from Section 2.7, Equation 6), the overall range is the same for the responses related to pairs (x_2, x_3) and (x_1, x_4) (Figure 2). Yet, x_4 has a null effect in psin and only x_1 is contributing for the overall range (left of Figure 2). In contrast, both x_2 and x_3 inputs affect the response (right of Figure 2), thus this pair produces a more interesting response when compared with the previous one.

Let $\{\hat{\mathbf{y}}_{(a_i,b_j)} : i \in \{1, ..., L\}, j \in \{1, ..., L\}\}$ denote the sensitivity responses related with $L \times L$ changes of input pair $(\mathbf{x}_a, \mathbf{x}_b)$. Let ς_{a_i} and ς_{b_j} represent the sensitivity measures (e.g., range) computed over $\hat{\mathbf{y}}_{a_i}$ and $\hat{\mathbf{y}}_{b_j}$, respectively, where $\hat{\mathbf{y}}_{a_i}$ denotes all responses when $\mathbf{x}_a$ is set to its i-th level. Instead of the global range, we propose the novel and more informative pair $(\varsigma_a, \varsigma_b)$:

$$(\varsigma_a, \varsigma_b) = (\textstyle\sum_{j=1}^{L} \varsigma_{b_j}/L, \sum_{i=1}^{L} \varsigma_{a_i}/L) \tag{3}$$

In the psin example, the range due to x_4 is $\varsigma_{x_4} = 0.0$ (left of Figure 2), thus confirming the null effect of x_4 variable.

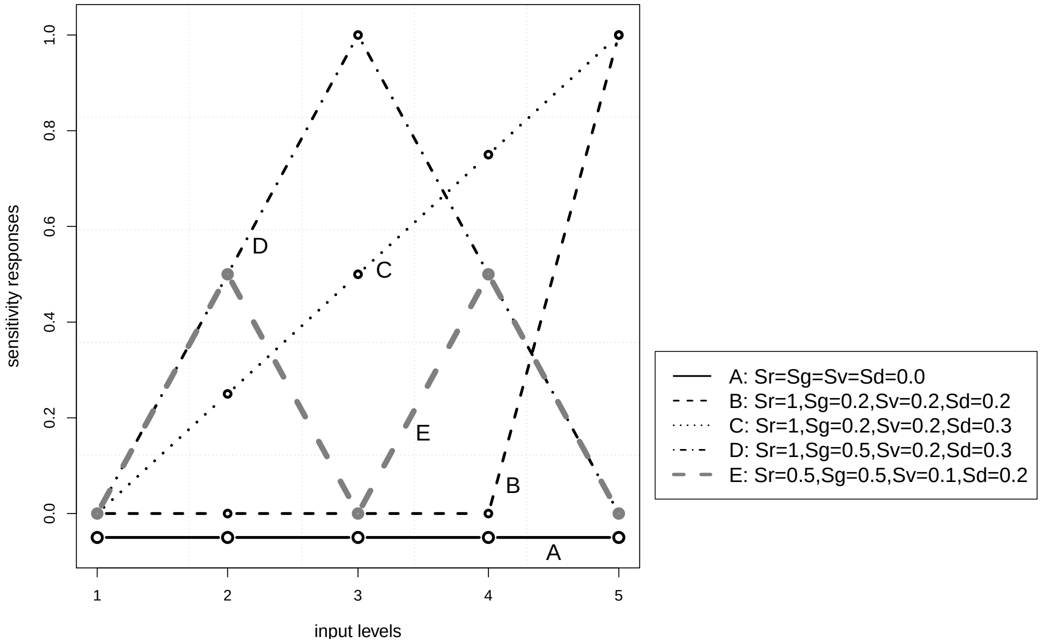

Figure 1: Example of the sensitivity measures for five response curves.

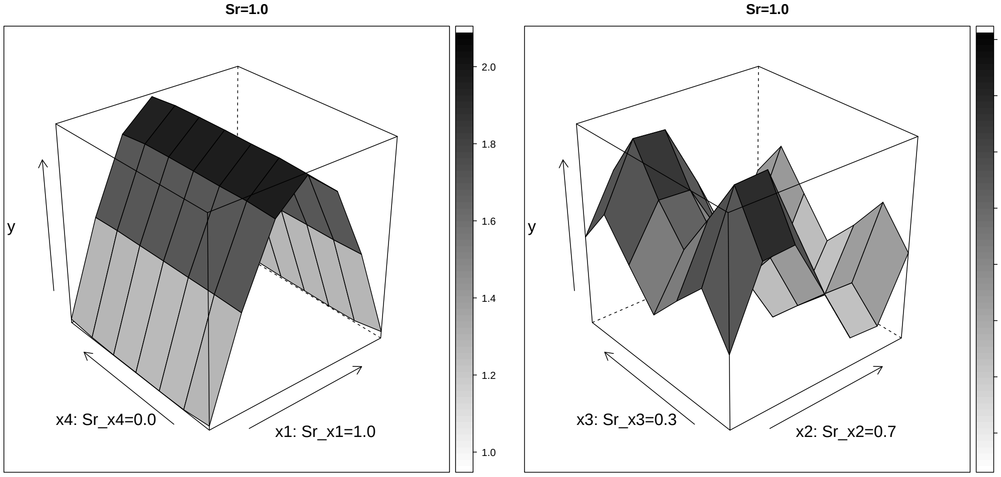

Figure 2: Variable effect characteristic surface for the psin task and pairs (x_1, x_4) (left) and (x_2, x_3) (right).

2.3. Discrete attributes

The sensitivity methods and measures of Sections 2.1 and 2.2 were devised for continuous variables. Since discrete variables are common in DM tasks (e.g., classification) we address them in this section. A discrete variable is often classified as ordered, if the variable values can be put in a scale (e.g., {low, medium, high}), or nominal, if there is no order in the values (e.g., {red, blue, yellow}). Dealing with ordered variables is simpler, as they can be encoded into a numeric ordered scale $\{1, 2, ..., G\}$ (G is the number of classes), which is treated as the continuous case.

To handle nominal input variables, we propose the following. For methods that require a baseline vector (i.e., 1D-SA and GSA), use the mode, i.e., most common value, as the baseline value. Also, when varying input $\mathbf{x}_a$ and for all sensitivity methods, all levels ($L_{\mathbf{x}_a} = G$) or only the most frequent ones ($L_{\mathbf{x}_a} = \min(G, L)$) can be used, where $L_{\mathbf{x}_a}$ denotes the number of levels used for the input $\mathbf{x}_a$.

When the target is nominal, there are two main modeling possibilities: outputting class labels ($\{c_1, ..., c_G\}$) or probabilities, where $\hat{\mathbf{y}} = (p_1, ..., p_G)$ such that $\sum_{i=1}^{G} p_i = 1$ and p_c denotes the probability of class c. To compute the sensitivity measures, our methods make use of the latter approach. Thus, in case of pure classification, we propose the transformation of the class labels into probabilities using the popular One-of-G transformation, where one binary variable is assigned to each class. For the color example, the transformation is: red→(1,0,0); blue→(0,1,0); yellow→(0,0,1). Similarly to the total Area Under of the receiver operating characteristic Curve (AUC) calculation for multi-class tasks [14], the sensitivity measures are first computed for each individual class (c) and then a weighted average is performed in order to compute the global sensitivity measure:

$$\varsigma_a = \sum_{c \in \{c_1, \ldots, c_G\}} f(c) \varsigma_a(c) \tag{4}$$

where $\varsigma_a(c)$ is the sensitivity measure (e.g., AAD) for attribute $\mathbf{x}_a$ and output class c, and $f(c)$ is the class c frequency in the dataset.

2.4. Sensitivity response aggregation functions

For 1-SA, there is only one sensitivity value for a given input variable level (x_{a_j}). Thus, the sensitivity measures of Section 2.2 can be directly applied over the sensitivity responses ($\hat{\mathbf{y}}_a$). However, the remaining sensitivity methods (e.g., GSA) produce several sensitivity values ($\hat{\mathbf{y}}_{a_j}$) for each input level. In such cases, an aggregation function needs to be set prior to the computation of the sensitivity measures.

In [8], all GSA sensitivity responses were first averaged according to each input level, i.e., only $j \in \{1, ..., L\}$ distinct $\overline{y}_{a_j}$ values were stored and then fed into the importance calculations (Equation 1). Yet, this simple aggregation function can lead to information loss. For instance, the left of Figure 3 demonstrates the sensitivity responses when using a full 4D SA for a regression task. When changing the input levels, the average sensitivity values (diamond points) are kept constant, although other statistics (e.g., maximum and minimum) undergo a substantial change. This phenomenon is less likely to occur in classification, as shown in the right of Figure 3, since the minimum and maximum values tend to be stable (set at minimum of 0.0 and maximum of 1.0 values).

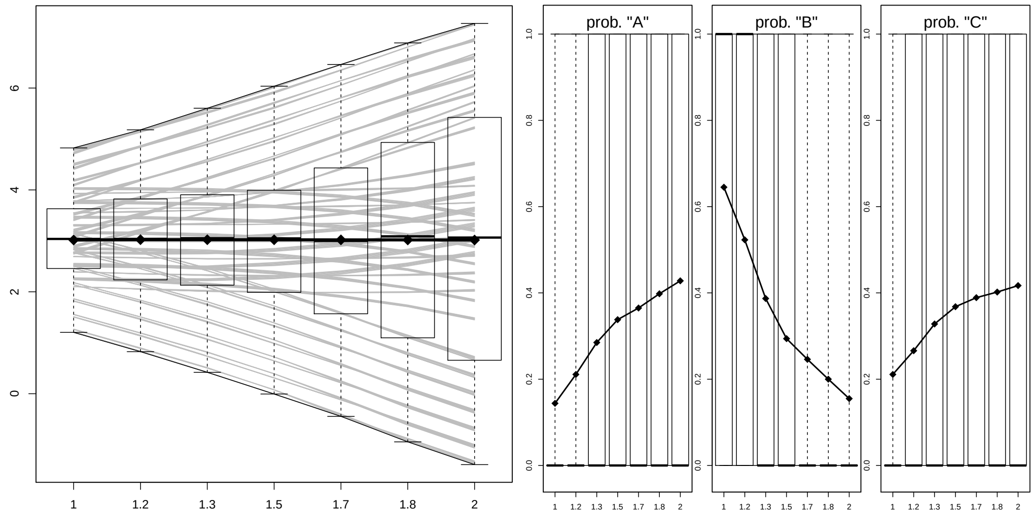

Figure 3: Variable effect characteristic curves for the regression int2 (left) and classification int2-2c (right) tasks (described in Section 2.7). The x-axis denotes the 7 different levels for the first input x_1 and the $y-$axis shows the sensitivity responses, in terms of: diamond points – average values; box plots – e.g., minimum and maximum values; and grey lines – individual data point sensitivity values).

To solve this problem, we propose a novel multi-statistic aggregation method for regression tasks, which averages three sensitivity measures, related with the statistics: minimum (min), average (avg) and maximum (max), over the $\hat{\mathbf{y}}_{a_j}$ values:

$$\begin{array}{lcl}
\hat{\mathbf{y}}_{\min(a)} & = & (\min(\hat{\mathbf{y}}_{a_1}), ..., \min(\hat{\mathbf{y}}_{a_L})) \\
\hat{\mathbf{y}}_{\mathrm{avg}(a)} & = & (\mathrm{avg}(\hat{\mathbf{y}}_{a_1}), ..., \mathrm{avg}(\hat{\mathbf{y}}_{a_L})) \\
\hat{\mathbf{y}}_{\max(a)} & = & (\max(\hat{\mathbf{y}}_{a_1}), ..., \max(\hat{\mathbf{y}}_{a_L})) \\
\varsigma_a & = & (\varsigma_{\hat{\mathbf{y}}_{\min(a)}} + \varsigma_{\hat{\mathbf{y}}_{\mathrm{avg}(a)}} + \varsigma_{\hat{\mathbf{y}}_{\max(a)}})/3
\end{array} \tag{5}$$

where $\varsigma_{\hat{\mathbf{y}}}$ denotes the sensitivity measure for each statistic (min, avg and max). For classification, we adopt the simpler average (1-metric) aggregation method over the sensitivity measure (using Equation 4).

Aggregation is also required when several responses are available for the sensitivity of an input pair $(\mathbf{x}_a, \mathbf{x}_b)$. For example, this can occur with GSA when $\#F > 2$. In such case, and for classification, we adopt the average aggregation function, over $\hat{\mathbf{y}}_{(a_i,b_j)}$, while the similar three statistic (min, avg and max) approach is proposed for regression.

A 2D $(L \times L)$ SA can also be applied to the DSA or MSA methods, although it can require a substantial computational effort. As a fast alternative, we propose a simpler aggregation strategy that combines CSA with the previous methods. We denote such strategy as $(\mathbf{x}_a, \mathbf{x}'_b)$, where the $\mathbf{x}'_b$ values are estimated but not taken directly from the sensitivity dataset. First, N_s responses values that are related with the first input $\mathbf{x}_a$ are selected. Next, the estimated input levels for the remaining input are set to $\mathbf{x}'_b = (x'_{b_1}, ..., x'_{b_L})$. Then, the responses for a given pair of values, $\hat{\mathbf{y}}_{(a_i,b_j)}$, are set to all the responses related with the first input level (x_{a_i}) and whose $\mathbf{x}_b$ values, as they occur in the DSA or MSA dataset, are closer to x'_{b_j}, within a threshold of $t_b \times (x'_{b_2} - x'_{b_1})$, where t_b is a tolerance constant. After obtaining the $\hat{\mathbf{y}}_{(a_i,b_j)}$ responses, then we can apply the same aggregation methods suggested for GSA.

2.5. Visualization techniques

Consider that a supervised learning model that was designed to achieve a satisfactory prediction accuracy under the smallest set of M input variables (e.g., use of feature selection). After applying the SA methods, several 2D and 3D visualization plots can be used to open the black box (examples of these plots are shown in Section 3.3).

Regarding input relevance, the **input importance bar plot**, can be used to show 1-D input importances (Equation 2), sorted from the highest to the lowest values. We propose a **color matrix** for the visualization of the sensitivity measures related with an input pair. Such technique was initially suggested in [13], where a dark coloring, denoting an interesting pair, was applied proportionally to the overall pair sensitivity measure. Yet, since we use a two dimension measure (Equation 3), the dark coloring is adapted to be proportional to the sum of these measures ($\varsigma_a + \varsigma_b$), provided that both ς_a and ς_b are above a given threshold (t_{cm}).

To present the average impact of a given input $\mathbf{x}_a$ in the model, we suggest the use of the **Variable Effect Characteristic (VEC) curve**, which plots the x_{a_j} values (x-axis) versus the $\hat{y}_{a_j}$ responses (y-axis). Between two consecutive x_{a_j} values, the VEC plot uses a line for continuous values and a horizontal segment for categorical data. To enhance the visualization analysis, nominal $\mathbf{x}_a$ values can be sorted according to the average $\hat{y}_{a_j}$ response. Also, several VEC curves can be plotted in the same graph. In such case, the x-axis should be scaled (e.g., within [0,1]) for all $\mathbf{x}_a$ values. For regression, and when there are several sensitivity responses for a given input level ($\hat{\mathbf{y}}_j$), a box plot can be used for each x_{a_j} level, as shown in Figure 3. For classification, and similarly to the Receiver Operating Characteristic (ROC) analysis, a VEC plot can be built for each target class (right of Figure 3). Moreover, when using multiple runs, the distinct VEC curves can be averaged vertically.

Finally, to show the impact of a pair of inputs ($\mathbf{x}_a$,$\mathbf{x}_b$), we propose the use of a **VEC surface** (e.g., Figure 2) and **contour plot**. The VEC surface provides more detail when compared with the contour plot, yet, a good interpretability is dependent on a correct adjustment of the 3D axis/angle representation. For regression, and when multiple responses are obtained for two input levels ($\hat{\mathbf{y}}_{(a_i,b_j)}$), three VEC surfaces or contour plot can be drawn, related with the min, avg and max aggregation functions. For classification, a VEC surface or contour plot can be drawn for each class.

2.6. Supervised learning methods

We explore three black box and popular DM models: NN, SVM and RF, as implemented in the rminer library of the R tool [6].

The NN is based on an ensemble that computes the average of the predictions of N_r multilayer perceptrons, all with different initial random weights and a hidden layer of H neurons with logistic functions. For regression, the output neuron uses a linear function. For a binary classification, there is one output neuron with a logistic function. Under multi-class tasks ($G > 2$), there are G linear output neurons and the softmax function is used to transform these outputs into class probabilities. The predicted class is given by the highest probability. The training (BFGS algorithm) of each network is stopped when the error slope approaches zero or after a maximum of M_e epochs.

The SVM uses a Gaussian (or RBF) kernel with the parameter γ. The SVM is fit using the sequential minimal optimization (SMO) learning algorithm. When modeling regression datasets, the ϵ-insensitive cost function adopted. For classification, the model can be set to estimate classes or class probabilities. For multi-class tasks, the one-against-one approach is used [33].

RF is an ensemble of a large number of T unpruned decision trees [2]. Each tree is based in a random feature selection from bootstrap training samples and the RF predictions are built by averaging the outputs of the T trees. For RF, rminer adopts the randomForest R package with all its default values (e.g., $T = 500$) except m, the number of inputs randomly selected at each decision tree node.

Before feeding the NN and SVM models, the nominal inputs are encoded using a One-of-G transformation, while the real inputs are standardized to a zero mean and one standard devision. The NN and RF hyperparameters, H and m, are set using a simple grid search. Regarding SVM, a more sophisticated search was adopted for the two (classification) or three (regression) hyperparameters, by using a two-level uniform design search [19]. For a given hyperparameter set of values, a cross-validation is applied over the training data and the value with the lowest estimation error is selected.

2.7. Data

We address synthetic and real-world datasets. Synthetic data from well defined benchmark functions are used to evaluate the sensitivity performances, because in real-world applications it is difficult to know in advance how a given input truly affects the target variable. Real-world data are used to demonstrate the sensitivity visualization capabilities and consists of two classification and two regression datasets from the UCI repository [1]: **bank** marketing, contraceptive method choice (**cmc**), **servo** and white wine quality (**wwq**). The bank data consists of 4521 samples related to a direct marketing campaign of a Portuguese bank [24]. The goal is to predict ({yes,no}) if a client will subscribe to a term deposit based on 16 continuous (e.g., age) and categorical (e.g., marital status) attributes. The cmc goal is to predict the contraceptive method type ({no use, long-term, short-term}) based on 9 socio-economic characteristics of Indonesian women (e.g. wife's age) and it includes 1473 instances. The servo dataset corresponds to a nonlinear task related to rise time of a servomechanism, including 167 examples and 4 inputs (2 nominal and 2 continuous). Finally, the wwq dataset contains 4898 wine entries and the regression goal is to predict human taste preferences, within a scale ranging from 3 – poor quality to 9 – excellent quality, based on 11 analytical continuous inputs (e.g., alcohol) [7]. All datasets are publicly available at: `http://www3.dsi.uminho.pt/pcortez/data`.

Regarding the synthetic data, four synthetic functions are proposed together with an adaptation of the Friedman #1 model [17]. These benchmark functions are realistic nonlinear targets and each one captures a different characteristic that is to be tested. Let x_1, x_2, ..., x_M denote a set of independent input variables, from a specified random distribution, y the dependent target and $N = 1000$ is the number of training samples. The five synthetic functions are listed below:

$$\begin{array}{rll}
y = & 1/2\sin(\pi\frac{x_1}{2N}) + 1/4\sin(\pi\frac{x_2}{2N}) + 1/6\sin(\pi\frac{x_3}{2N}) & \text{(ssin)} \\
y = & (1+\sin(\pi\frac{x_1}{N}))(1+0.25\sin(\pi\frac{2x_2}{N}))(1+0.125\sin(\pi\frac{3x_3}{N})) & \text{(psin)} \\
y = & x_1^2\sin(\pi\frac{2x_2}{N}) + 2x_3 & \text{(int2)} \\
y = & q(x_1 > 0.5)(5 + 2.5q(x_2 > 0.5) + 1.25q(x_3 > 0.5)) & \text{(tree)} \\
 & +q(x_1 \leq 0.5)(2.5q(x_4 > 0.5) + 1.25q(x_5 > 0.5)) & \\
y = & 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 & \text{(fri1)}
\end{array} \tag{6}$$

where $q(Q)$ is a function that returns 1 if the query Q is true, else $q(Q)$ returns 0. In preliminary experiments, we tested three different distributions for the random number generation: uniform, normal and beta. Because all distributions led to similar sensitivity results, we only use the simpler random uniform function, to simplify the analysis. The five synthetic regression tasks are:

Sum of sin (ssin) – The aim is to mimic an additive response of independent nonlinear inputs. The dataset contains $M = 4$ inputs, each uniform in $[0, 1000]$. Only 3 inputs influence the target, under the theoretical relative importances of $(0.55, 0.27, 0.18, 0.00)$, e.g., $0.55 = 0.5/(0.5 + 0.25 + 0.125)$.

Product of sin (psin) – This is a multiplicative model with nonlinear inputs, with a decreasing influence and thus should be more challenging than ssin. Similarly to ssin, there are $M = 4$ inputs uniform in $[0, 1000]$. Due to the multiplicative effect, it is harder to infer about the true input relative importances. The first input produces a response within $[1, 2]$

and range=1, which is multiplied by the effect of x_2 (within $[0.75, 1.25]$, range=0.5) and then multiplied by x_3 (range=0.25). Under this rational, x_1 is more relevant than x_2, which is more relevant than x_3.

Interaction of two variables (int2) – This function was specifically designed to challenge the a 1D-SA, since it uses a nonlinear interaction of the first two variables. A total of $M = 4$ inputs are used, where x_1, x_3, x_4 are uniform in $[1, 2]$ and x_2 is uniform in $[0, 1000]$. Moreover, x_2 produces a wave (up and down) effect, with an average value of 0. Again, it is difficult to assess the true input relevances, although the pair (x_1, x_2) produces a range of 8, and thus should be more relevant than x_3 (range=2).

Decision tree (tree) – The intention is to mimic a simple regression tree, with a total of $M = 10$ inputs uniform in $[0, 1]$. This dataset was also set to defy the 1D-SA, since the first input (x_1) sets two different branches and the remaining inputs only are used in one of these branches. The theoretical input importances are (0.57,0.14,0.07,0.14,0.07,0,0,0,0,0). For instance, for x_2, $0.14 = (2.5 \cdot 0.5)/(5 + 2.5 \cdot 0.5 + 1.25 \cdot 0.5 + 2.5 \cdot 0.5 + 1.25 \cdot 0.5))$.

Friedman #1 (fri) – There are $M = 10$ inputs (uniform in $[0, 1]$), although only the first five are relevant. This function contains some interesting properties: there is an interaction between x_1 and x_2 and both have the same importance; x_1, x_2 and x_3 produce a nonlinear effect; and x_4 is twice as important as x_5. Similarly to psin and int2, the true input importances are difficult to be estimated, although the pair (x_1,x_2), with a range of 10, should affect the target more than x_3 (range of 5).

For a synthetic classification, we use variants of the ssin and int2 functions. The aim is to test different: outputs (class labels or probabilities), number of classes (2, 3 or 8) and inputs (real and nominal). The five synthetic classification tasks are:

ssin-2c and **ssin-2p** – similar to ssin, except that the target is transformed into the class labels {"A","B"}, under the rule "A" if $y < 0.6$, else "B". In total, this leads to 477 "A" and 523 "B" samples and the same data is used by ssin-2c and ssin-2p. The difference is that ssin-2c models pure class labels while ssin-2p handles class probabilities.

ssin-n2p – similar to ssin-2p, except that the aim is to test nominal inputs, thus all inputs suffer a leveling transform into 10 random shuffled labels within {"A","B",...,"J"}. For instance, all input values of $x_1 < 100$ are labeled "B" and all $100 \leq x_1 < 200$ values are labeled "J".

int2-3c and **int2-8p** – similar to int2, except that the target is transformed into: int2-3c – output label modeling with three classes {"A","B","C"}, under the rule "A" if $y < 0.2$ (310 samples), else "B" if $y < 3.8$ (341 samples), else "C" (349 samples); int2-8p – probability modeling with eight classes, using the vector $(0, 1, 2, 3, 4, 5, 6)$ as the separation limit values to assign the classes. The int2-8b classes are biased, e.g., class "A" contains 200 samples while class "E" contains 49.

Given the rules applied, it is difficult to assess the true theoretical input importances for the synthetic classification tasks. Thus, we opt for the following criteria: ssin based tasks – the input importance ranking should be first x_1, second x_2, third x_3; int2 based tasks – the same as int2, i.e., the pair (x_1, x_2) should be more relevant than x_3, also all x_1, x_2 and x_3 should affect the target. In both ssin and int2, x_4 should have a null impact.

3. Experiments and results

3.1. Predictive results

All reported experiments were conducted using the rminer library[1] (e.g., function `Importance`) and the R tool [6]. For NN, the settings are $N_r = 7$ and $M_e = 100$. The grid search ranges are: $H \in \{1, 2, \ldots, 10\}$ (NN); $\gamma \in \{2^{-15}, \ldots, 2^3\}$ and $C \in \{2^{-5}, \ldots, 2^15\}$ (SVM classification), $\gamma \in \{2^{-8}, \ldots, 2^0\}$, $C \in \{2^{-1}, \ldots, 2^6\}$ and $\epsilon \in \{2^{-8}, \ldots, 2^{-1}\}$ (SVM regression); $m \in \{1, ..., 10\}$ (RF). A total of 10 searches were used for NN and RF, while the SVM uniform design method required 13 searches. An internal 3-fold cross-validation was applied (using only training data) to obtain the estimation error. To assess the predictive capabilities for each method (i.e., test error), 10 runs of a k-fold cross-validation procedure were applied. For the smaller datasets, related to all synthetic data and servo, a k=10 fold validation was applied. For the larger real-world datasets (bank, cmc and wwq), and similar to [7], the more computationally reasonable 5-fold setup was adopted. For pure multi-class classification, the error metric is the overall classification accuracy (ACC) [32]. When modeling class probabilities, we adopt the global

[1] `http://cran.r-project.org/web/packages/rminer/`

Area Under the Curve (AUC), which weights the area of each ROC class according to its prevalence in the data [14]. For regression, the Mean Absolute error (MAE) was used [32].

Table 1 shows the best models for the synthetic datasets, according to the average test error (last column, shown in terms of the mean value and respective 95% t-student confidence intervals). For each model, the median hyperparameter is also shown in brackets. Overall, high quality predictive results were achieved. For all synthetic tasks except fri1, the results are almost perfect, with MAE near 0.0 and ACC/AUC near 1.0. For the real-world data, interesting performances were obtained. For the bank data, the average AUC is higher than 0.9, which is often considered a high quality discrimination. The average global AUC, for the three cmc classes, is higher than 0.7, corresponding to a good discrimination, and this performance is similar to what was achieved in [6]. For servo, the MAE value is low (0.22) and corresponds to a Relative Absolute Error (RAE) of 19% (i.e., around five times better than the error given by the naive average predictor). For wwq, the MAE value of 0.450 is quite small when considering the range of 6.0 for the output target and identical to the value that was obtained in [7].

Table 1: Best models and predictive test errors for synthetic (regression, classification) and real-world datasets.

Task	**Model**	**Test Error**
ssin	SVM ($\tilde{\gamma} = 2^{-3.0}$,$\tilde{C} = 2^{0.75}$,$\tilde{\epsilon} = 2^{-8.0}$)	MAE=0.001±0.000
psin	SVM ($\tilde{\gamma} = 2^{-2.0}$,$\tilde{C} = 2^{6.87}$,$\tilde{\epsilon} = 2^{-8.0}$)	MAE=0.014±0.000
int2	SVM ($\tilde{\gamma} = 2^{-2.0}$,$\tilde{C} = 2^{6.87}$,$\tilde{\epsilon} = 2^{-8.0}$)	MAE=0.011±0.000
tree	RF ($\tilde{m} = 10$)	MAE=0.019±0.001
fri1	NN ($\tilde{H} = 10$)	MAE=0.296±0.009
ssin-2c	SVM ($\tilde{\gamma} = 2^{-6}$,$\tilde{C} = 2^{10}$)	ACC=0.987±0.001
ssin-2p	SVM ($\tilde{\gamma} = 2^{-6}$,$\tilde{C} = 2^{10}$)	AUC=0.999±0.000
ssin-n2p	SVM ($\tilde{\gamma} = 2^{-8.25}$,$\tilde{C} = 2^{10}$)	AUC=0.994±0.000
int2-3c	NN ($\tilde{H} = 7$)	ACC=0.969±0.002
int2-8p	NN ($\tilde{H} = 9$)	AUC=0.994±0.000
bank	RF ($\tilde{m} = 5$)	AUC=0.916±0.001
cmc	NN ($\tilde{H} = 7$)	AUC=0.735±0.002
servo	NN ($\tilde{H} = 7$)	MAE=0.220±0.014
wwq	SVM ($\tilde{\gamma} = 2^{0.0}$,$\tilde{C} = 2^{1.63}$,$\tilde{\epsilon} = 2^{-2.75}$)	MAE=0.450±0.004

3.2. Sensitivity results

In this section, we evaluate the sensitivity methods. To simplify the explanation and analysis, just one model for each task was considered. In order to do so, the best models of Table 1 were retrained using all data and the median hyperparameter values (e.g., H = 10 for fri1). Then, the sensitivity methods and visualization techniques (Section 3.3) were applied to the retrained models.

We start by testing the simpler 1D-SA on the ssin dataset. Table 2 shows the respective relative input importance values. For this dataset, although the number of levels (L) varied from 2 to 20, the same importance values were obtained. In the input importance tables shown in this paper, **bold** denotes empirical (r_a) values that confirm the theoretical analysis (r_{ta}) performed in Section 2.7. Thus, **bold** is used when $|r_{ta} - r_a| < 0.05$ and/or if ranking of the inputs makes sense.

Table 2: 1D-SA importance values (R_a) for ssin task.

Measure	Levels	Input Importances
range	$L \in 2, 3, ..., 20$	(**0.55,0.27,0.18,0.00**)
gradient	$L \in 2, 3, ..., 20$	(**0.54,0.27,0.18,0.00**)
variance	$L \in 2, 3, ..., 20$	(0.74,0.18,0.08,**0.00**)
AAD	$L \in 2, 3, ..., 20$	(**0.55,0.27,0.18,0.00**)

All sensitivity measures perfectly capture the theoretical importance values, except for the variance (S_v), which assigns more weight to the first input (x_1). Since a similar behavior was found for other synthetic datasets, we discard the analysis of the variance measure for the remainder of this paper.

In 1D-SA, the influence of the number of levels (L) depends on the dataset tested. Often, there is a flat effect after a few levels, i.e., increasing the value of L produces only a slight change in the input importance values [20]. This effect is shown in Figure 4 (when $L \geq 6$), which plots the relative importance values, according to three sensitivity measures, of the most relevant input for psin (x_1). Since it makes sense to use an odd number of levels, to capture the average $\mathbf{x}_a$ value, and as a reasonable trade-off between the obtained effect and computational complexity, $L = 7$ is used in the remainder of this paper.

There are datasets which do not present differences in terms of relative importance when different sensitivity measures are applied (e.g., ssin, tree).

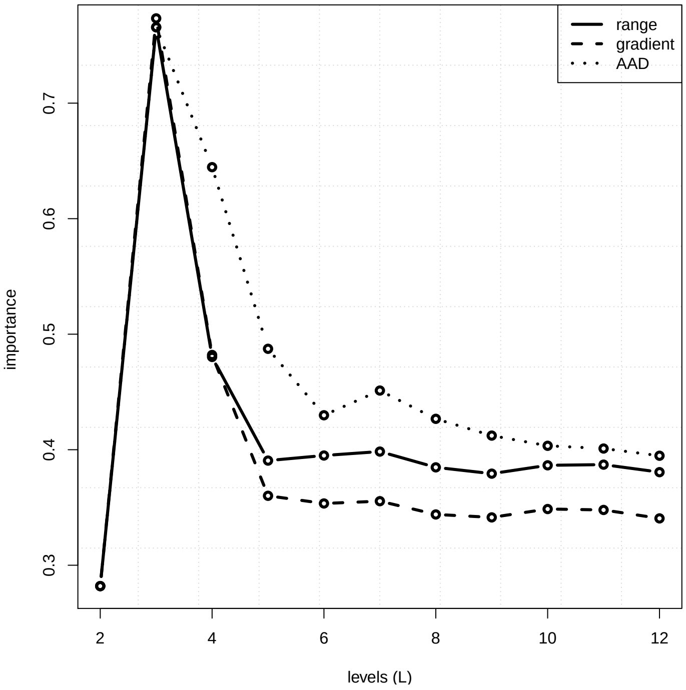

Figure 4: Effect of increasing the number of levels on psin for the relative importance of x_1 (r_1).

In other cases, such as shown in Table 3, the sensitivity measure leads to distinct importance values. For psin, AAD gives a higher importance for x_1 when compared with range or the gradient. More interesting is that for the classification tasks (ssin-2c and ssin-2p), AAD is the only measure that ranks x_1 as the most important input. Taking into account Table 3 and the theoretical advantages exposed in Section 1, we opt for using AAD through the remainder of this paper.

Table 3: 1D-SA ($L = 7$) importance values (R_a) for psin, ssin-2c and ssin-2p tasks.

Task	**Measure**	**Input Importances**
psin	range	(**0.41**,**0.36**,**0.23**,**0.01**)
	gradient	(**0.36**,**0.32**,**0.31**,**0.01**)
	AAD	(**0.46**,**0.36**,**0.18**,**0.01**)
ssin-2c	range	(0.33,0.33,0.33,**0.00**)
	gradient	(0.33,0.33,0.33,**0.00**)
	AAD	(**0.43**,0.29,0.29,**0.00**)
ssin-2p	range	(0.33,0.33,0.33,**0.00**)
	gradient	(0.33,0.33,0.33,**0.00**)
	AAD	(**0.40**,**0.32**,**0.28**,**0.00**)

To study the influence of the DM model, Table 4 shows the input importance values for two synthetic datasets (ssin and psin) and distinct models. We tested three DM models (SVM, NN and RF) and two slight variations of the best hyperparameters for SVM and NN. The obtained results show that, once a good predictive capability is achieved, the quality of the sensitivity approach is robust and not dependent on a particular model type or hyperparameter configuration, since similar importances were achieved.

The performance of the SA methods over all synthetic tasks is analyzed in Tables 5 and 6). For the multiple response SA methods (e.g., GSA) and the regression tasks, the three-metric aggregation method described in Section 2.4 is used, while for the remaining cases the simpler average aggregation is adopted. For task int2-n3p, the number of input levels is set to $L_{\mathbf{x}_a} = \min(10, 7) = 7$, thus the three least frequent levels from each input were discarded from the analysis. Two different DSA versions were tested, DSA using all training data ($N = 1000$) and using only 1% random samples ($N_s = 10$) from the data. The same $N_s = 10$ value was adopted for MSA$_d$ (MSA with discrete random sampling). For the stochastic sampling methods

Table 4: 1SA ($L = 7$, AAD) importance values (R_a) for the ssin and psin tasks.

Task	**Model**	**Error (MAE)**	**Input Importances**
ssin	SVM ($\gamma = 2^{-3.0}$,$C = 2^{0.75}$,$\epsilon = 2^{-8.0}$)	0.001±0.000	(**0.55,0.27,0.18,0.00**)
	SVM ($\gamma = 2^{-5.0}$,$C = 2^{1.75}$,$\epsilon = 2^{-7.0}$)	0.001±0.000	(**0.54,0.27,0.18,0.00**)
	NN ($H = 10$)	0.001±0.000	(**0.55,0.27,0.18,0.00**)
	NN ($H = 8$)	0.001±0.000	(**0.55,0.27,0.18,0.00**)
	RF ($m = 3$)	0.012±0.000	(**0.58,0.27,0.15,0.00**)
psin	SVM ($\gamma = 2^{-2.0}$,$C = 2^{6.87}$,$\epsilon = 2^{-8.0}$)	0.014±0.000	(**0.46,0.36,0.18,0.00**)
	SVM ($\gamma = 2^{-4.0}$,$C = 2^{5.87}$,$\epsilon = 2^{-7.0}$)	0.067±0.000	(**0.46,0.39,0.13,0.02**)
	NN ($H = 10$)	0.047±0.001	(**0.49,0.38,0.13,0.01**)
	NN ($H = 8$)	0.055±0.002	(**0.49,0.41,0.09,0.01**)
	RF ($m = 3$)	0.075±0.000	(**0.43,0.41,0.16,0.01**)

the average value of 20 runs is shown. Due to the high memory and computational requirements of GSA, the analysis is limited to the first seven inputs ($\#F = 7$) for tree and fri1.

The 1D-SA performance is of high quality for three regression tasks (ssin, psin and fri1) and also captures the performance criteria defined for two classification problems (ssin-2p and int2-8p). Yet, in the remaining cases, 1D-SA presents several flaws: the importance of x_1 is not detected for the int2 tasks (the only exception is int2-8p), both x_4 and x_5 have a null effect on tree; x_2 and x_3 have an equal importance in ssin-2c; and x_3 has a null effect on ssin-n2p. Turning to the cluster-based method (CSA), it presents some interesting results, in particular for classification, outperforming 1D-SA for int2 (when measuring the importance of x_1), tree (for x_4 and x_5), and several classification tasks (e.g., ssin-2c). However, CSA fails in detecting the most relevant input in psin. Also, CSA tends to give a considerable (e.g., ≥ 0.05) impact to the null effect inputs. All remaining methods (GSA, DSA and MSA), capture the theoretical analysis of the input importances. In particular, it should be stressed that even when only N_s =1% of the samples are used, both stochastic methods (DSA and MSA) obtained high quality results. Moreover, in general, the results obtained by these three methods (GSA, DSA and MSA) are quite similar, confirming that the true input importances were captured.

The proposed sensitivity methods also work for "white box" models. To

Table 5: Importance values for the regression synthetic tasks and best models from Table 1.

Task	**Method**	**Input Importances**
ssin	1D-SA	(**0.55,0.27,0.18,0.00**)
	GSA ($\#F = 4$)	(**0.54,0.27,0.18,0.02**)
	DSA ($N = 1000$)	(**0.54,0.27,0.18,0.01**)
	DSA ($N_s = 10$)	(**0.54,0.27,0.18,0.0**)*
	MSA$_d$ ($N_s = 10$)	(**0.54,0.27,0.18,0.01**)*
	CSA	(**0.52,0.24,0.18**,0.05)
psin	1D-SA	(**0.46,0.36,0.18,0.01**)
	GSA ($\#F = 4$)	(**0.53,0.32,0.14,0.00**)
	DSA ($N = 1000$)	(**0.52,0.33,0.15,0.01**)
	DSA ($N_s = 10$)	(**0.52,0.33,0.15,0.01**)*
	MSA$_d$ ($N_s = 10$)	(**0.53,0.32,0.15,0.01**)*
	CSA	(0.39,0.40,**0.16**,0.05)
int2	1D-SA	(0.04,**0.63,0.32,0.00**)
	GSA ($\#F = 4$)	(**0.21,0.53,0.24,0.02**)
	DSA ($N = 1000$)	(**0.24,0.52,0.23,0.01**)
	DSA ($N_s = 10$)	(**0.24,0.51,0.25,0.00**)*
	MSA$_d$ ($N_s = 10$)	(**0.23,0.51,0.26,0.01**)*
	CSA	(**0.19,0.57,0.16**,0.07)
tree	1D-SA	(0.64,0.23,0.13,0.00,0.00,**0.00,0.00**)
	GSA ($\#F = 7$)	(**0.57,0.15,0.07,0.14,0.07,0.00,0.00**)
	DSA ($N = 1000$)	(**0.57,0.14,0.07,0.14,0.07,0.00,0.00**)
	DSA ($N_s = 10$)	(**0.57,0.15,0.07,0.14,0.07,0.00,0.00**)*
	MSA$_d$ ($N_s = 10$)	(**0.57,0.15,0.08,0.13,0.07,0.00,0.00**)*
	CSA	(0.52,**0.13,0.07,0.13,0.06,0.02,0.02**)
fri1	1D-SA	(**0.25,0.25,0.13,0.24,0.13,0.00,0.00**)
	GSA ($\#F = 7$)	(**0.17,0.16,0.18,0.33,0.15,0.00,0.01**)
	DSA ($N = 1000$)	(**0.17,0.18,0.17,0.32,0.14,0.00,0.01**)
	DSA ($N_s = 10$)	(**0.20,0.21,0.15,0.29,0.14,0.00,0.00**)*
	MSA$_d$ ($N_s = 10$)	(**0.19,0.19,0.16,0.30,0.14,0.00,0.00**)*
	CSA	(**0.14,0.14,0.12,0.22,0.12**,0.05,0.05)

$\star$ - mean values over 20 runs (all confidence intervals are within $[0.00, 0.02]$).

Table 6: Importance values for the classification synthetic tasks and best models from Table 1.

Tasks	**Method**	**Input Importances**
ssin-2c	1D-SA	(**0.43**,0.29,0.29,0.00)
	GSA ($\#F = 4$)	(**0.74,0.16,0.10,0.00**)
	DSA ($N = 1000$)	(**0.60,0.24,0.16,0.00**)
	DSA ($N_s = 10$)	(**0.61,0.23,0.16,0.0**)*
	MSA$_d$ ($N_s = 10$)	(**0.61,0.23,0.15,0.01**)*
	CSA	(**0.6,0.21,0.14**,0.06)
ssin-2p	1D-SA	(**0.40,0.32,0.28,0.00**)
	GSA ($\#F = 4$)	(**0.74,0.16,0.10,0.00**)
	DSA ($N = 1000$)	(**0.60,0.24,0.16,0.00**)
	DSA ($N_s = 10$)	(**0.59,0.26,0.15,0.01**)*
	MSA$_d$ ($N_s = 10$)	(**0.62,0.24,0.13,0.01**)*
	CSA	(**0.60,0.20,0.14**,0.05)
ssin-n2p	1D-SA	(0.97,0.03,0.00,**0.00**)
	GSA ($\#F = 4$)	(**0.78,0.12,0.08,0.02**)
	DSA ($N = 1000$)	(**0.64,0.20,0.14,0.02**)
	DSA ($N_s = 10$)	(**0.64,0.19,0.15,0.02**)*
	MSA$_d$ ($N_s = 10$)	(**0.66,0.19,0.12,0.02**)*
	CSA	(**0.59,0.20,0.13**,0.07)
int2-3c	1D-SA	(0.00,**0.64,0.36,0.00**)
	GSA ($\#F = 4$)	(**0.12,0.78,0.10,0.00**)
	DSA ($N = 1000$)	(**0.20,0.63,0.17,0.01**)
	DSA ($N_s = 10$)	(**0.17,0.63,0.20,0.00**)*
	MSA$_d$ ($N_s = 10$)	(**0.21,0.60,0.17,0.01**)*
	CSA	(**0.16,0.60,0.16**,0.08)
int2-8p	1D-SA	(**0.16,0.42,0.41,0.01**)
	GSA ($\#F = 4$)	(**0.20,0.67,0.13,0.00**)
	DSA ($N = 1000$)	(**0.26,0.60,0.14,0.00**)
	DSA ($N_s = 10$)	(**0.27,0.49,0.21,0.02**)*
	MSA$_d$ ($N_s = 10$)	(**0.23,0.51,0.24,0.02**)*
	CSA	(**0.22,0.55,0.14**,0.09)

$\star$ - mean values over 20 runs (all confidence intervals are within $[0.00, 0.05]$).

illustrate this, we used a decision tree model to fit the tree (regression) and sin-2c (classification) datasets. The model was trained using the CART algorithm [3], as implemented in the R tool. Table 7 presents the average test error for such model, when applying 10 runs of the 10-fold validation, revealing a good fit. The table also shows the input importance values, as given by the DSA method when applied to a decision tree fit with all data. For both datasets, the input importance measures captured by DSA confirm the theoretical analysis. Moreover, the input ranking given by DSA matches the same ranking, from top to bottom, as given by the decision trees (Figure 5).

Table 7: DSA ($L = 7$) importance values (R_a) for a decision tree model.

Task	**Test Error**	**Input Importances**
tree	MAE=0.018±0.001	(**0.57,0.14,0.07,0.14,0.07,0.00,0.00**)
ssin-2c	ACC=0.936±0.003	(**0.72,0.20,0.08,0.00**)

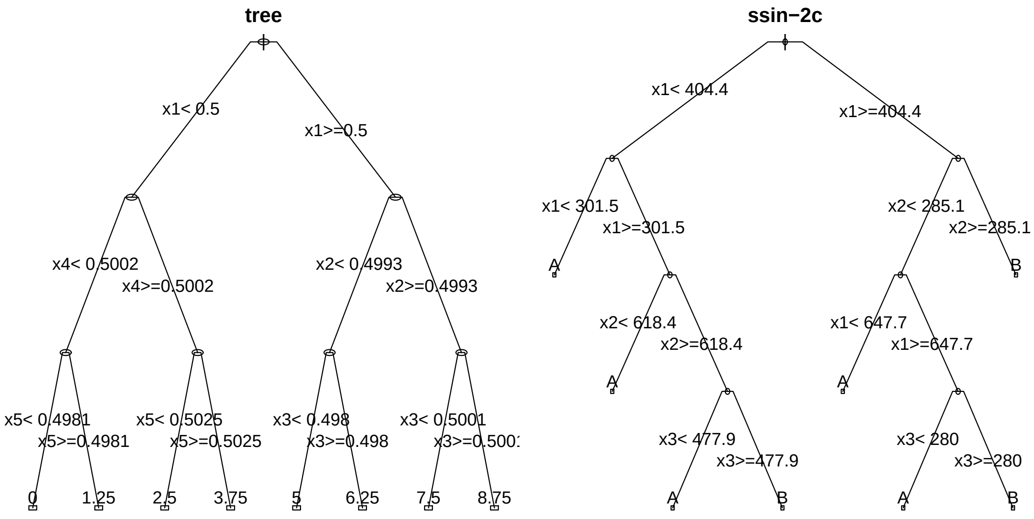

Figure 5: Decision tree model for the tree and ssin-2c datasets.

Turning to the sensitivity of a pair of inputs, Table 8 shows the AAD sensitivity measures (ς_a, ς_b) for psin task. The psin was selected as an interesting task for this analysis, since the output depends of nonlinear interactions between all the first three inputs (x_1, x_2 and x_3). Two sensitivity approaches were used. The first approach uses the full 4D-GSA, while the second one uses DSA with all training samples ($N_s = 1000$) and the estimated sensitivity $(\mathbf{x}_a, \mathbf{x}'_b)$ pair. For the $\mathbf{x}'_b$ estimation, we set a tolerance of $t_b = 0.1$ (other values were also tested, such as 0.2, but similar results were achieved). For both approaches, the average of the three-metric aggregation function was adopted (Section 2.4). Table 8 clearly shows the true input pair importances, with the most interesting pairs being (x_1, x_2), (x_1, x_3), (x_2, x_3), for both sensitivity approaches and under a decreasing ranking of interest. Also, for GSA, the results are symmetric. For instance, the sensitivity for the pairs (x_1, x_2) and (x_2, x_1) are (0.32,0.19) and (0.19,0.32), respectively. Moreover, the results achieved for the computationally faster DSA method are quite similar to the ones obtained by GSA. The main difference is that the null effect input (x_4), which is clearly detected by GSA but not by DSA for the pairs $(\mathbf{x}_a, x'_4)$. This behavior is similar to what was measured by CSA under the 1D-SA but in the 2D case it is not problematic. First, because the input pair ranking for DSA is similar to the one given by GSA. Second, because the null effect input is still detected under the $(x_4, \mathbf{x}'_b)$ sensitivity measures.

Table 8: Sensitivity measures for psin task and the pairs $(\mathbf{x}_a, \mathbf{x}_b)$ and $(\mathbf{x}_a, \mathbf{x}'_b)$.

Method		x_1	x_2	x_3	x_4
4D GSA $(\mathbf{x}_a, \mathbf{x}_b)$	x_1		(0.32,0.19)	(0.32,0.08)	(0.32,0.00)
	x_2	(0.19,0.32)		(0.19,0.08)	(0.19,0.00)
	x_3	(0.08,0.32)	(0.08,0.19)		(0.08,0.00)
	x_4	(0.00,0.32)	(0.00,0.19)	(0.00,0.08)	
DSA $(\mathbf{x}_a, \mathbf{x}'_b)$	x_1		(0.32,0.20)	(0.32,0.09)	(0.33,0.04)
	x_2	(0.20,0.31)		(0.20,0.10)	(0.21,0.05)
	x_3	(0.09,0.31)	(0.09,0.22)		(0.09,0.08)
	x_4	(0.00,0.30)	(0.00,0.23)	(0.00,0.13)	

3.3. Visualizations for the sensitivity analysis

In this section, we show several examples of the proposed visualization techniques (Section 2.5) to open the black box for the real-world tasks. In the experiments, we used a 2.66 GHz Intel Core i7 processor, under the operating system Mac OS X 10.6.8. Due to computational limitations, the full $\#F$D GSA cannot be computed, since the number of input attributes is too high. For example, the full 16D GSA requires the prediction of $\approx 7^{16}$

examples for the bank task. Therefore, a more reasonable DSA algorithm was applied, using all the training samples, to avoid using multiple runs, and all levels (e.g., $L_{\text{month}} = 12$ for bank). The DSA execution times were: 34 seconds for bank, 6 for cmc, 2 seconds for servo and 139 seconds for wwq. The respective sensitivity results were aggregated as described in Section 2.4, using the threshold t_b=0.1 for the input pair $(\mathbf{x}_a, \mathbf{x}'_b)$ sensitivity estimation.

Figure 6 shows two examples of how the input importances can be plotted, under a 1D (bank) and 2D (wwq) sensitivity. For bank data, the most relevant input is the **duration** of the marketing phone/cellular contact, followed by the number of days that passed from a previous campaign (**pdays**) and the **month** of the marketing contact. The 2D analysis considers all $(\mathbf{x}_a, \mathbf{x}_b)$ input pairs, coloring each ($a - x$-axis, $b - y-$axis) square according to what is proposed in Section 2.5, under the AAD measure and with $t_{cm} = 10\%$. From the matrix, it is clear that the most interesting input pair is (x_{11} – **alcohol**, x_6 – **free sulfur dioxide**).

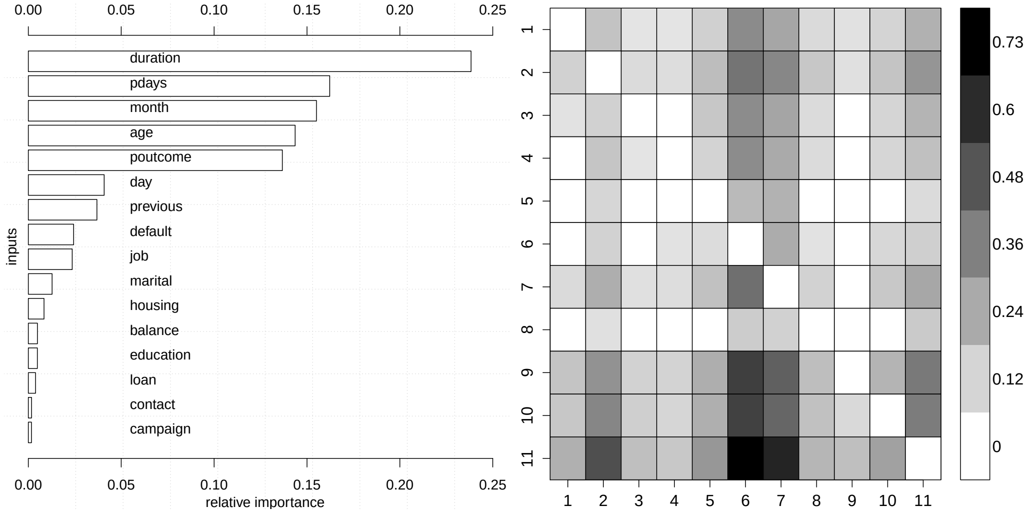

Figure 6: Bar plot with the 1D input importances for the bank data (left) and color matrix with 2D input pair sensitivity for the wwq dataset.

The individual effect of a given input is easily shown using VEC curves. For bank, Figure 7 compares the average influence of the three most relevant inputs. In the graph, the month levels were sorted in a decreasing order, according to their average influence in the target. Intuitively, the VEC curves for duration and pdays make sense: if a client keeps the call for more than

17 minutes (third level for duration) or if the previous campaign call was earlier than 400 days (third level), then the client is more willing to subscribe the term deposit. Regarding the month influence, March and October are associated with a higher probability for subscription.

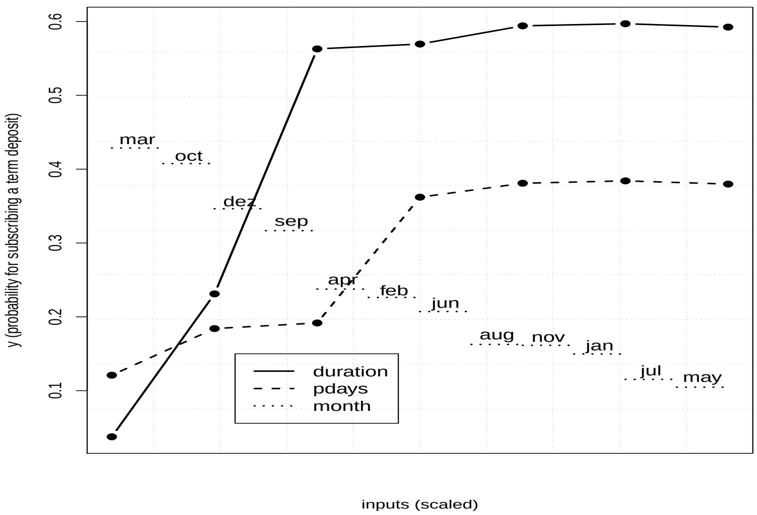

Figure 7: VEC curves for the three most relevant inputs for the bank task.

As another example of the VEC curve explanatory power, we applied the DSA method to perform a 1D analysis to the cmc model, which revealed the **wife's age** as the most relevant input (importance of 30%). The effect of this input on the choice of contraceptive method is shown in Figure 8, revealing interesting findings. For instance, on average (diamond points), younger women (up to 27 years old) tend to use short term methods, while older women (older than 38) are more likely to not use contraception. Also, age tends to produce a parabola shape effect on the probability for using long term methods, reaching its maximum effect on a value around 32 years old.

Regarding the wwq task, Figure 9 plots the VEC curves with box plots for the free sulfur dioxide (most relevant variable according to a 1D analysis) and alcohol (least relevant variable under the same analysis). In both cases, the average VEC curves (diamond points) have an almost flat shape. However, the range of the sensitivity is high (as shown by the box plots), confirming that the output depends more on input interactions rather than single input effects. For demonstration purposes, the most relevant pair according to the color matrix of Figure 6 was selected and the corresponding VEC surface

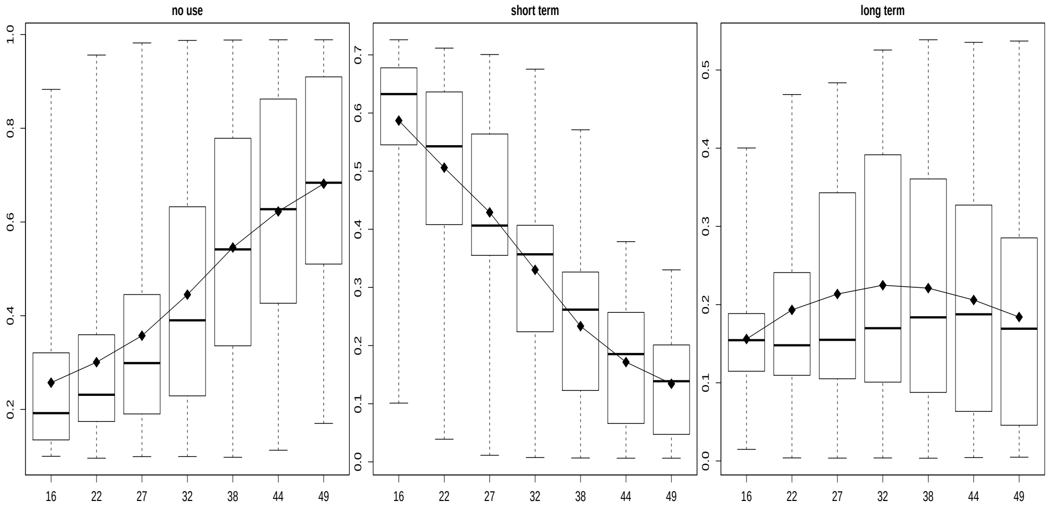

Figure 8: VEC curves with box plots for the wife's age influence (x-axis) on cmc task (y-axis) and classes "no use", "short term" and "long term".

and contour plots are shown in Figure 10. All three maximum, average and minimum functions were used when computing the color matrix. Yet, to simplify the visualization, the VEC surface and contour is only shown for the average aggregation function. For fixed surfur dioxide there are only five levels, since under the $t_{cm} = 0.1$ tolerance there were two estimated levels without data samples ($\mathbf{x}'_5$ and $\mathbf{x}'_6$). Figure 10 shows two interesting and small clusters that are close to each other, related with high quality (darkest area) and low quality (brightest area) wine. Such information is useful for wine experts, since it reveals the impact of these key variables in the tasting quality and, as argued in [7], these variables can be controlled in the production process. (e.g., alcohol levels can be increased by monitoring the grape sugar concentration before the harvest).

Another 2D analysis example is given for the servo task, with the intention to show the effect of discrete inputs. After applying DSA and a 2D analysis (under the three-metric aggregation method), we selected the input pair (**screw**, **motor**), which includes only nominal variables ($\in \{A, B, C, D, E\}$) and corresponds to the fifth most relevant pair: $(\varsigma_{\mathbf{mscrew}}, \varsigma_{\mathbf{motor}}) = (0.3, 0.3)$. The respective VEC surface and contour plots for the average aggregation function are shown in Figure 11. The plots reveal that, on average, the combination screw=E and motor=D leads to the fastest rise time, closely

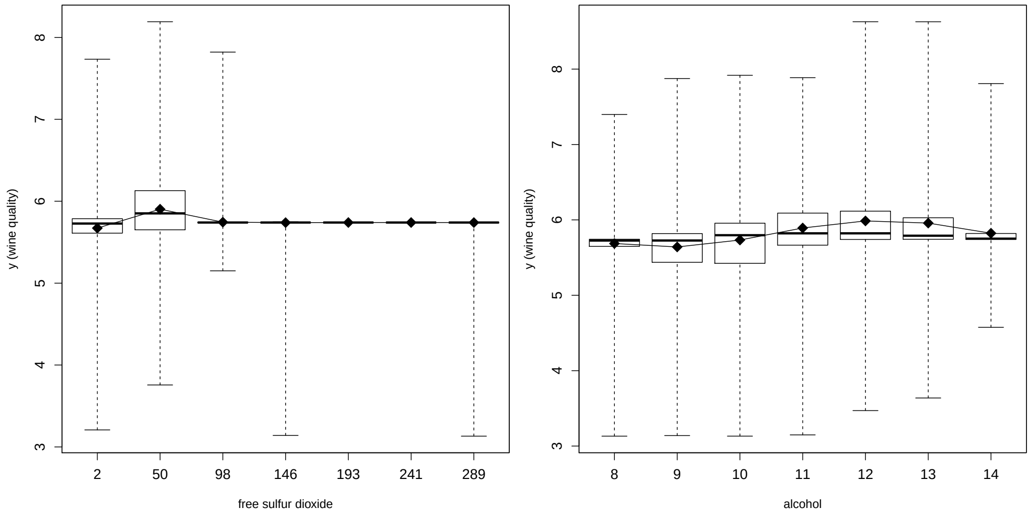

Figure 9: VEC curves with box plots for free sulfur dioxide (left) and alcohol (right).

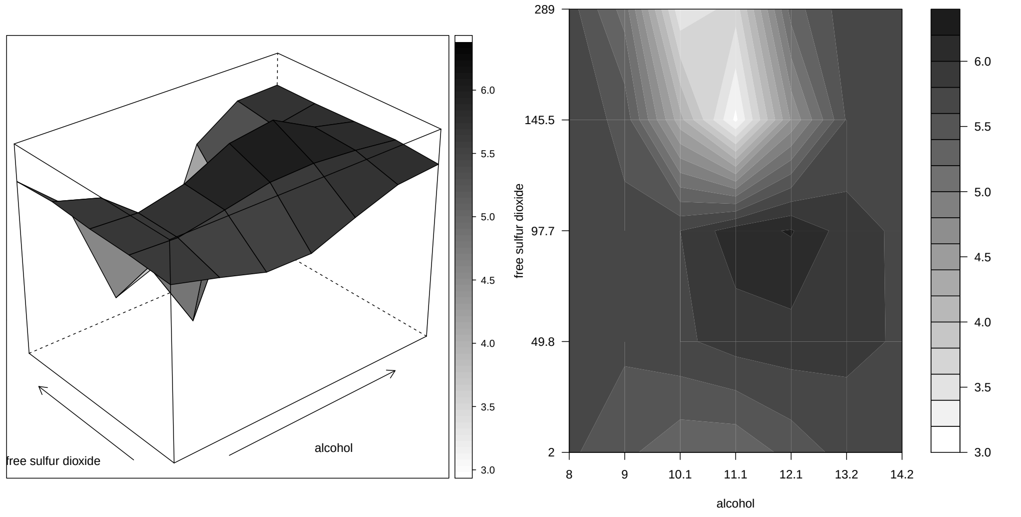

Figure 10: VEC surface and contour plots for the wwq task.

followed by the screw=E and motor=E setup. The longest rise time is related with the combination screw=A and motor=E, which clearly stands out when compared with its neighborhood (e.g., screw=A and motor=D).

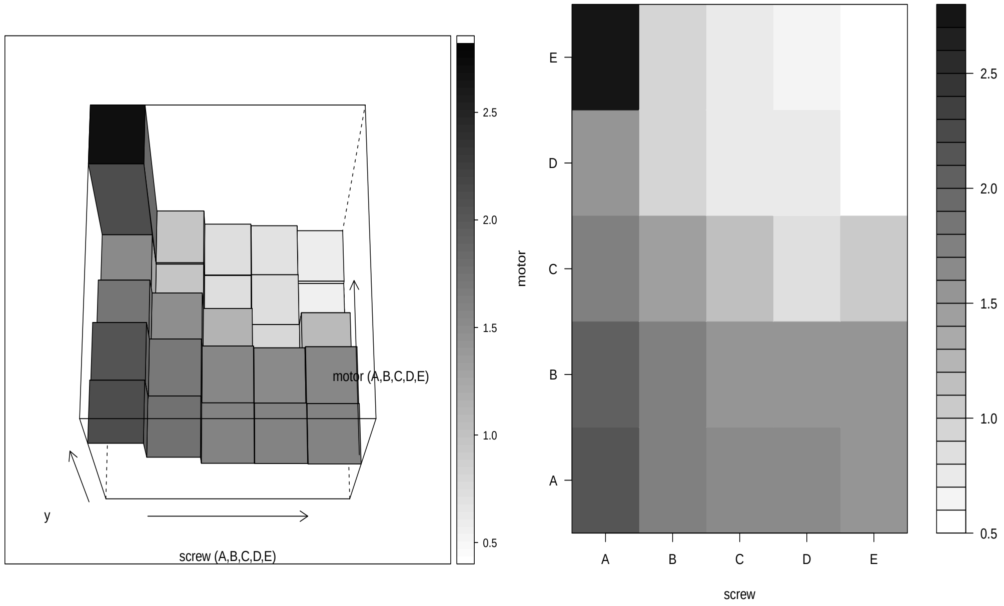

Figure 11: VEC surface and contour plots for the servo task.

4. Conclusions and future work

There are several supervised black box DM methods, such as NN, SVM and ensembles (including RF), that are capable of high quality prediction performances and thus are valuable to support decision making. Yet, the obtained data-driven models are difficult to understand by humans. Improving interpretability enhances the acceptance and understanding of these DM models by the domain users. In particular, interpretability is a key issue in critical applications, such as medicine or control.

In this paper, we propose the combination of SA methods and visualization techniques to open the black box. Since the data-driven models are treated as black boxes, and no information obtained during the fitting procedure is used, the SA methods can be applied universally to any supervised DM method. Several SA methods, measures, aggregation functions and visualization techniques (e.g., VEC surface) were proposed. The effectiveness of SA methods was assessed in several synthetic regression and classification tasks. Moreover, the capabilities of the visualization techniques were demonstrated using four real-world datasets: bank direct marketing (classification), contraceptive method choice (classification), rise time of a servomechanism (regression) and white wine quality (regression).

Given the obtained results, and as a standard approach for a real-world application, we suggest the use of the novel DSA method using all data samples, in conjunction with the AAD measure of importance. DSA is computationally reasonable, when compared with GSA, and it provides better results than the simpler 1D-SA, as it is capable of detecting input variable interactions. Moreover, for real-world datasets, DSA performs a sensitivity that is closer to the real input data distributions, when compared with MSA, which uses random uniform samples to build the sensitivity dataset. If a high number training samples is available, then the computational effort of DSA can reduced by using a smaller and random subset of the training data. Moreover, when the number of inputs is too large, such as hundreds or thousands, then, as a prior preprocessing step, a 1D-SA analysis could be used to select a smaller subset of interesting inputs, as performed in [13]. Finally, there are some application scenarios where the fitted DM model is available but not the training data, such as symbiotic data mining, which shares fitted models but not the data (due to privacy issues) among distinct users [22]. In such scenarios, the random sampling MSA method could be used as an alternative to DSA.

In the future, we will enlarge the experiments to include more real-world domains (e.g., clinical data). Also, the proposed approach will be integrated into a graphical user interface system that incorporates an interactive visualization of the SA results. For example, where users could change the selected input variables and considered levels, zoom a particular interesting area or change the orientation of a VEC 3D surface. Another promising research direction is the application of a SA approach to clustering tasks. For instance, by using a strategy similar to the classification case, where the cluster response is considered as the “output”. Finally, there is a potential to improve variable/feature selection algorithms by using the proposed measures of input relevance to guide their search (i.e., select variables to be deleted).

Acknowledgments

The work of P. Cortez was funded by FEDER, through the program COMPETE and the Portuguese Foundation for Science and Technology (FCT), within the project FCOMP-01-0124-FEDER-022674. Also, the authors wish to thank the anonymous reviewers for their helpful comments.

[1] A. Asuncion and D. Newman. UCI Machine Learning Repository, Univ. of California Irvine, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.

[2] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] L. Breiman, J. Friedman, R. Ohlsen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Monterey, CA, 1984.

[4] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. *CRISP-DM 1.0: Step-by-step Data Mining Guide*. CRISP-DM consortium, 2000.

[5] B.H. Cho, H. Yu, J. Lee, Y.J. Chee, I.Y. Kim, and S.I. Kim. Nonlinear support vector machine visualization for risk factor analysis using nomograms and localized radial basis function kernels. *Information Technology in Biomedicine, IEEE Transactions on*, 12(2):247–256, 2008.

[6] P. Cortez. Data mining with neural networks and support vector machines using the R/rminer tool. In P. Perner, editor, *Advances in Data Mining – Applications and Theoretical Aspects, 10th Industrial Conference on Data Mining*, pages 572–583, Berlin, Germany, July 2010. LNAI 6171, Springer.

[7] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[8] P. Cortez and M. Embrechts. Opening black box data mining models using sensitivity analysis. In *IEEE Symposium Series in Computational Intelligence 2011 (SSCI 2011)*, Paris, France, 4 2011.

[9] M.W. Craven and J.W. Shavlik. Visualizing learning and computation in artificial neural networks. *International Journal on Artificial Intelligence Tools*, 1(3):399–425, 1992.

[10] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other Kernel-Based Learning Methods)*. Cambridge University Press, 2000.

[11] T. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems, Lecture Notes in Computer Science 1857*, pages 1–15. Springer-Verlag, 2000.

[12] M.J. Embrechts, F. Arciniegas, M. Ozdemir, M. Momma, CM Breneman, L. Lockwood, KP Bennett, and RH Kewley. Strip mining for molecules. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 305–310. IEEE, 2002.

[13] M.J. Embrechts, F.A. Arciniegas, M. Ozdemir, and R.H. Kewley. Data mining for molecules with 2-D neural network sensitivity analysis. *International Journal of Smart Engineering System Design*, 5(4):225–239, 2003.

[14] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[15] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[16] D. Fisch, A. Hofmann, and B. Sick. On the versatility of radial basis function neural networks: A case study in the field of intrusion detection. *Information Sciences*, 180(12):2421–2439, 2010.

[17] J.H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

[18] S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition, 2009.

[19] C.M. Huang, Y.J. Lee, D.K.J. Lin, and S.Y. Huang. Model selection for support vector machines via uniform design. *Computational Statistics & Data Analysis*, 52(1):335–346, 2007.

[20] R. Kewley, M. Embrechts, and C. Breneman. Data strip mining for the virtual design of pharmaceuticals with neural networks. *IEEE Transactions on Neural Networks*, 11(3):668–679, May 2000.

[21] I. Kondapaneni, P. Kordík, and P. Slavík. Visualization techniques utilizing the sensitivity analysis of models. In *Proceedings of the 39th Conference on Winter Simulation: 40 years! The Best is Yet to Come*, pages 730–737. IEEE Press, 2007.

[22] C. Lopes, P. Cortez, P. Sousa, M. Rocha, and M. Rio. Symbiotic filtering for spam email detection. *Expert Systems with Applications*, 38(8):9365–9372, 2011.

[23] D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476, 2007.

[24] S. Moro, R. Laureano, and P. Cortez. Using data mining for bank direct marketing: An application of the crisp-dm methodology. In P. Novais et al., editor, *Proceedings of the European Simulation and Modelling Conference - ESM'2011*, pages 117–121, Guimaraes, Portugal, October 2011. EUROSIS.

[25] D.W. Ruck, S.K. Rogers, and M. Kabrisky. Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*, 2(2):40–48, 1990.

[26] R. Setiono. Techniques for extracting classification and regression rules from artificial neural networks. In D. Fogel and C. Robinson, editors, *Computational Intelligence: The Experts Speak*, pages 99–114. Piscataway, NY, USA, IEEE, 2003.

[27] Á. Silva, P. Cortez, M. F. Santos, L. Gomes, and J. Neves. Mortality assessment in intensive care units via adverse events using artificial neural networks. *Artificial Intelligence in Medicine*, 36(3):223–234, 2006.

[28] K. Thearling, B. Becker, D. DeCoste, B. Mawby, M. Pilote, and D. Sommerfield. *Visualizing Data Mining Models*, chapter 15, pages 205–222. Morgan Kaufmann, 2001.

[29] A. Tickle, R. Andrews, M. Golea, and J. Diederich. The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transactions on Neural Networks*, 9(6):1057–1068, 1998.

[30] E. Turban, R. Sharda, and D. Delen. *Decision Support and Business Intelligence Systems*. Prentice Hall, 9th edition, 2010.

[31] F.Y. Tzeng and K.L. Ma. Opening the black box-data driven visualization of neural networks. In *Proceedings of the Visualization (VIS 05)*, pages 383–390. IEEE, October 2005.

[32] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, CA, 2005.

[33] T.F. Wu, C.J. Lin, and R.C. Weng. Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005, 2004.

[34] R. Xia, C. Zong, and S. Li. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, 181(6):1138–1152, 2010.