Article

Data Augmentation and Feature Selection for Automatic Model Recommendation in Computational Physics

Thomas Daniel [1,2,*] , **Fabien Casenave** [1] , **Nissrine Akkari** [1] and **David Ryckelynck** [2]

1   SafranTech, Rue des Jeunes Bois, Châteaufort, 78114 Magny-les-Hameaux, France; fabien.casenave@safrangroup.com (F.C.); nissrine.akkari@safrangroup.com (N.A.)
2   Centre des matériaux (CMAT), MINES ParisTech, PSL University, CNRS UMR 7633, BP 87, 91003 Evry, France; david.ryckelynck@mines-paristech.fr
*   Correspondence: thomas.daniel@mines-paristech.fr

Abstract: Classification algorithms have recently found applications in computational physics for the selection of numerical methods or models adapted to the environment and the state of the physical system. For such classification tasks, labeled training data come from numerical simulations and generally correspond to physical fields discretized on a mesh. Three challenging difficulties arise: the lack of training data, their high dimensionality, and the non-applicability of common data augmentation techniques to physics data. This article introduces two algorithms to address these issues: one for dimensionality reduction via feature selection, and one for data augmentation. These algorithms are combined with a wide variety of classifiers for their evaluation. When combined with a stacking ensemble made of six multilayer perceptrons and a ridge logistic regression, they enable reaching an accuracy of 90% on our classification problem for nonlinear structural mechanics.

1. Introduction

Classification problems can be encountered in various disciplines such as handwritten text recognition [1], document classification [2], and computer-aided diagnosis in the medical field [3], among many others. In numerical analysis, classification algorithms are getting increasingly more attention for the selection of efficient numerical models that can predict the behavior of a physical system with very different states or under various configurations of its environment [4–11]. Classifiers have been used as reduced-order model (ROM) selectors in [4,5,8,9,11] in computational mechanics, enabling the computation of approximate solutions at lower cost by replacing a generic high-fidelity numerical model by a specific (or local) ROM adapted to the simulation's context. Reduced-order modeling [12,13] consists in identifying an appropriate low-dimensional subspace on which the governing equations are projected in order to reduce the number of degrees of freedom of the solution. In [11], the combination of a classifier with a dictionary of local ROMs has been termed *dictionary-based ROM-net*. Such approaches are promising numerical methods using both physics equations and a collection of latent spaces to compute approximations of solutions lying in nonlinear manifolds.

Dictionary-based ROM-nets use a physics-informed automatic data labeling procedure based on the clustering of numerical simulations. Due to the cost of numerical simulations, training examples for classification are limited in number. Moreover, the dimensionality of input data can be very high, especially when dealing with physical fields discretized on a mesh (e.g., finite-difference methods [14], finite-element method [15], and finite-volume method [16]) or with bond graphs modeling engineering systems [17].

When classification data are high-dimensional, dimensionality reduction techniques can be applied to reduce the amount of information to be analyzed by the classifier. For

classification problems where the dimension of the input data is higher than the number of training examples, dimensionality reduction is crucial to avoid overfitting. In addition, when considering physical fields discretized on a mesh, the dimension of the input space can reach 10^6 to 10^8 for industrial problems. In such cases, the input data are too hard to manipulate, which dramatically slows down the training process for the classifier and thus restrains the exploration of the hyperparameters space, as it requires multiple runs of the training process with different values for the hyperparameters. Applying data augmentation techniques to increase the number of examples in the training set is also impossible, as it would cause memory problems. Therefore, dimensionality reduction is recommended not only for reducing the risk of overfitting, but also for facilitating the training phase and enabling data augmentation.

Feature selection [18] aims at decreasing the number of features by selecting a subset of the original features. It differs from *feature extraction*, where new features are created from the original ones (e.g., Principal Component Analysis (PCA), and more generally encoders taken from undercomplete autoencoders [19]). Feature selection can be seen as applying a mask to a high-dimensional random vector to get a low-dimensional random vector containing the most relevant information. It is preferred over autoencoders when interpretability is important [20]. Furthermore, contrary to undercomplete autoencoders trained with the mean squared error loss, most feature selection algorithms do not intend to find reduced features enabling the reconstruction of the input: features are selected for the purpose of predicting class labels, which makes these algorithms more goal-oriented for supervised learning tasks.

Among the existing feature selection algorithms, univariate filter methods consist in computing a score for each feature and ranking the features according to their scores. The score measures how relevant a feature is for the prediction of the output variable. If N_f is the target number of features, then the N_f features with the highest scores are selected, and the others are discarded. The major drawback of univariate filter methods is that they do not account for relations between the selected features. The resulting set of selected features may then contain redundant features. To address this issue, the *minimum redundancy maximum relevance* (mRMR) algorithm [21,22] tries to find a trade-off between relevance and redundancy. However, for very large numbers of features like in computational physics, evaluating the redundancy is very computationally demanding. Fortunately, working on physics data provides other possibilities to define a redundancy measure. In this paper, we propose a new feature selection algorithm suitable for features coming from the same physical quantity but corresponding to different points in a space-time discretization. It is assumed that this physical quantity, defined as a function of space and/or time, has some smoothness properties. This is often the case in physics, where the physical quantity satisfies partial differential equations and boundary conditions. In [23], it is shown that the solution of Poisson's equation on a Lipschitz domain in $\mathbb{R}^3$ with a L^2 source term and Dirichlet or Neumann boundary conditions is continuous. Poisson's equation is well known in physics, and can be found, for example, in electrostatics, in Gauss's law for gravity, in the stationary heat equation, and in the stationary particle diffusion equation. If the features of a random vector contain the discretized values of a smooth function of space and time, then their correlations are related to their proximities on the space-time grid. The approach presented in this paper is depicted as a geostatistical variant of mRMR algorithm, in the sense that it consists in modeling the redundancy as a function of space and time.

Once the dimension of the input space is reduced, another challenge of the classification problems encountered in computational physics must be addressed: the lack of training data. *Data augmentation* refers to techniques aiming at enlarging the training set by generating new examples from the original ones. For image classification, many class-preserving operations can be used to create new images, such as translations; rotations; cropping; scaling; and changes in colors, brightness, and contrast. Unfortunately, these common techniques cannot be used when considering physics data. For this type of

data, new examples can be generated using generative adversarial networks (GAN [24]; see in [25] for the use of deep convolutional GANs in computational fluid dynamics). However, training GANs is quite complex in practice and may also be made more difficult by the lack of training examples. More simply, new data can be generated by convex combinations of the original examples. SMOTE [26] takes convex combinations of input data with their nearest neighbors in the input space. ADASYN [27] uses the same idea but focuses more on examples that are hard to learn, i.e., those having examples of a foreign class in their neighborhoods. Both data augmentation algorithms use k-nearest neighbors algorithm and thus compute Euclidean distances in the input space. When working on high-dimensional physics data, this approach may suffer from the curse of dimensionality [28]. In addition, defining neighborhoods with the Euclidean distance in the input space is not always appropriate, as dictionary-based ROM-nets use physics-aware dissimilarities to label the data, such as distances on the primal variable or on a quantity of interest. The data augmentation algorithm developed in this article consists in growing sets around original examples by incrementally adding nearest neighbors in terms of the dissimilarity measure used for the automatic data labeling procedure. These sets are used to generate new data by convex combinations. Contrary to SMOTE and ADASYN, the risk of generating new data with wrong labels is controlled by checking that the convex hulls of the growing sets do not contain any example belonging to a foreign class.

In sum, the contributions of this paper are motivated by difficulties encountered in our previous work on ROM-nets [11]. These difficulties are inherent to classification tasks on simulation data and can be summarized in three main issues:

- the lack of training data due to the expensive data labeling procedure involving simulations with a high-fidelity model (risk of overfitting),
- the high dimensionality of input data (risk of overfitting), and
- most common data augmentation techniques are not applicable to physics data.

The feature selection and data augmentation strategies introduced in this paper are developed to tackle these difficulties. Classification problems encountered in computational physics are described in Section 2. Section 3 presents the classification problem studied in this paper. The feature selection algorithm is described in Section 4 and is shown to efficiently remove irrelevant and redundant features. Section 5 presents the data augmentation algorithm, which successfully generates a large amount of new data with correct labels. Section 6 evaluates both algorithms in conjunction with 14 different classifiers. On our classification task, the average accuracy gain due to data augmentation is 4.98%. Using ensemble methods on classifiers combined with our algorithms enables reaching a classification accuracy of 90%. Finally, Section 7 explains to what extent our algorithms can be applied to other types of problems.

2. Classification in the Context of Numerical Modeling

2.1. Classification: A Brief Review

Supervised learning is the task of learning the correspondence between input data X and outputs Y from a training set of input–output pairs $\{(x_i, y_i)\}_{1 \leq i \leq N}$. Supervised machine learning problems fall into two categories: regression problems, for which the outputs take continuous values, and classification problems, consisting in the prediction of categorical labels. This paper focuses on the latter, with the additional assumptions that X is a continuous multivariate random variable having a probability density function $p_X : \mathcal{X} \to \mathbb{R}_+$, and that any observation $x \in \mathcal{X}$ is associated to a single label y. The discrete random variable Y follows a categorical distribution (or multinoulli distribution) whose probability mass function is defined by

$$\forall y \in \mathbb{R}, \quad p_Y(y) = \sum_{k=1}^{K} \mathbb{P}_Y(k) \delta(y - k) \tag{1}$$

where K is the number of categories (or classes), δ is the Dirac delta function, and $\mathbb{P}_Y(k)$ denotes the probability of the event $Y = k$ for a given label $k \in [\![1; K]\!]$. The labeled training data are drawn from the joint probability distribution $p_{X,Y}$, called the *data-generating distribution*. As X is continuous and Y is discrete, $p_{X,Y}$ is a mixed joint density and can be obtained with the formula

$$p_{X,Y}(x,y) = p_Y(y)\, p_{X|Y}(x \mid y) = \sum_{k=1}^{K} \mathbb{P}_Y(k)\delta(y-k)p_{X|Y}(x \mid y) \qquad (2)$$

with $p_{X|Y}$ being the class-conditional probability distribution.

In the present paper, we are interested in single-label multiclass problems. Therefore, the classification problem considered here reads: *given an integer $K \geq 2$ and a training set $\{(x_i, y_i)\}_{1 \leq i \leq N} \subset \mathcal{X} \times [\![1; K]\!]$, train a classifier $\mathcal{C}(\,.\,; \theta) : \mathcal{X} \to [\![1; K]\!]$ to assign any observation $x \in \mathcal{X}$ to the correct class, with θ denoting the parameters of the classifier.* However, reaching the highest possible accuracy on the training set is not the objective to be pursued, as it usually leads to *overfitting*. Indeed, the classifier is supposed to be applied to new unseen data, or *test* data, after the training phase. Therefore, the generalization ability of the classifier is at least as important as its performance on the training set. A classifier with high capacity (ability to learn classes with complex boundaries, related to model complexity) perfectly fits training data but is very sensitive to noise, leading to high test error and thus overfitting. On the other hand, a classifier with low capacity can produce smaller error gaps between training and test predictions, but such a classifier may not be able to fit the data, which is called *underfitting*. This dilemma is known as the *bias-variance trade-off*: low model capacity leads to high bias, while high model capacity leads to high variance.

For a given observation $x \in \mathcal{X}$, probabilistic classification algorithms estimate the membership probabilities $\mathbb{P}_{\text{model}}(y \mid x; \theta)$ for each class $y \in [\![1; K]\!]$. The classifier $\mathcal{C}$ returns the index of the class with the highest membership probability:

$$\mathcal{C}(x; \theta) = \underset{y \in [\![1;K]\!]}{\arg\max}(\mathbb{P}_{\text{model}}(y \mid x; \theta)) \qquad (3)$$

The parameters θ must be optimized to minimize the *expected risk* $\mathcal{J}(\theta)$ defined by

$$\mathcal{J}(\theta) = \mathbb{E}_{(X,Y)\sim p_{X,Y}}[L(\mathcal{C}(X; \theta), Y)] \qquad (4)$$

where L is the per-example loss function quantifying the error between the predicted class $\mathcal{C}(X; \theta)$ and the true class Y. However, as the true data-generating distribution $p_{X,Y}$ is unknown, the expected risk must be estimated by computing the expectation with respect to the empirical distribution $\widehat{p}_{X,Y}$:

$$\widehat{p}_{X,Y}(x,y) = \frac{1}{N}\sum_{i=1}^{N} \delta(x - x_i, y - y_i) \qquad (5)$$

Therefore, the training process consists in minimizing the *empirical risk*:

$$\widehat{\mathcal{J}}(\theta) = \mathbb{E}_{(X,Y)\sim\widehat{p}_{X,Y}}[L(\mathcal{C}(X; \theta), Y)] = \frac{1}{N}\sum_{i=1}^{N} L(\mathcal{C}(x_i; \theta), y_i) \qquad (6)$$

This is known as the *empirical risk minimization* (ERM) principle [29]. Common choices for the function L are the hinge loss (defined for multiclass problems in [30]) used by support vector machines (SVMs), and the log loss or negative log-likelihood

$$L(\mathcal{C}(x; \theta), y) = -\log\, \mathbb{P}_{\text{model}}(y \mid x; \theta) \qquad (7)$$

that is widely used for classifiers based on artificial neural networks (ANNs) and for logistic regression. When L is the negative log-likelihood, the objective function $\widehat{\mathcal{J}}(\theta)$ is

the cross-entropy loss and the optimal set of parameters θ^* minimizing $\widehat{\mathcal{J}}$ is the maximum likelihood estimator [31]. Usually, a regularization term is added to the empirical risk to penalize the model complexity in order to reduce overfitting.

The boundaries between classes in the input space are called decision boundaries. Linear classifiers are classification algorithms for which the decision boundaries are defined by linear combinations of the features of X. Linear classifiers are appropriate when the classes are linearly separable in $\mathcal{X}$, which means that the decision boundaries correspond to portions of hyperplanes. Linear classifiers include logistic regression [32–34], linear discriminant analysis (LDA [31]), and the linear support vector classifier (linear SVM [35,36]).

Many algorithms exist for nonlinear classification problems, each of them having its own advantages and drawbacks. As a kernel method, the linear SVM is extended to nonlinear classification problems using the *kernel trick* based on Mercer's theorem [37]. Artificial neural networks [38,39] (see in [40] for a historical review) have become very popular due to their performances in numerous classification contests. Decision trees (e.g., CART algorithm [41]) and naive Bayes classifiers [42,43] are well-known for their interpretability. Other nonlinear classifiers include the k-nearest neighbors algorithm (kNN [44]) and quadratic discriminant analysis (QDA [31]). In [45], the most common classifiers are compared on eleven binary classification problems. Short reviews of classification algorithms can be found in [46,47].

Usually, combining several models to form a meta-estimator results in more robust predictions and reduces overfitting. This idea is used in ensemble methods such as bagging (or bootstrap aggregating) [48], feature bagging (or random subspace method) [49], stacking [31,50], boosting (including the well-known AdaBoost algorithm [51,52]), gradient boosting [53–56], and voting classifiers based on either a majority vote or a soft vote (technique known as ensemble averaging [57]). Random forests [58] combine bagging and feature bagging to build an ensemble of decision trees. All these methods are designed to reduce overfitting. For instance, when using ensemble averaging, the final estimates for membership probabilities are obtained by averaging predictions of several classifiers or instances of a classifier, which results in a regularization of the boundaries between the classes.

2.2. Classification for Numerical Simulations

Classification algorithms have recently found applications in numerical simulations, and more specifically for the selection of numerical models adapted to the context of the simulation. In this case, the class labels are used to identify the models.

Applications to turbulence modeling in computational fluid dynamics can be found in [7,10]. In large eddy simulations (LES; see in [59]), the Navier–Stokes equations are filtered to avoid resolving small-scale turbulent structures whose effects are taken into account either by sub-grid scale models (explicit LES closures) or via the dissipation induced by numerical schemes (implicit LES). In [7], sub-grid statistics obtained from direct numerical simulations enable training a fully connected deep neural network to switch between different explicit LES closures at any point of the grid. This classifier is reused in [10], this time for switching between different numerical schemes in implicit LES. In both cases, the classifier is used to increase the accuracy of numerical predictions.

The idea of locally switching between different simulation strategies can also be found in [6] for the multiscale modeling of composite materials. In the multilevel finite-element method (FE^2 [60]), the quantities of interest at every integration point of the macroscopic finite-element mesh are given by a microscopic finite-element computation of an elementary cell representing the material's microstructure. The multi-fidelity surrogate model presented in [6] relies on two surrogate models replacing the microscopic finite-element model: a reduced-order model taken from [61] and an artificial neural network based regression model. At each integration point of the macroscopic mesh, the classifier (a fully connected network) analyzes the effective strains and predicts whether the error

of the regression model would be acceptable, enabling the selection of either the purely data-driven regression model or the more sophisticated physics-driven ROM. This time, automatic model recommendation by a classifier is used to adapt the model complexity and reduce the computation time.

In [8,9], optimal classification trees (OCTs [62]) are used as *model selectors* in a data-driven physics-based digital twin of an unmanned aerial vehicle (UAV). The OCTs enable the update of the digital twin according to sensor data by selecting a model from a predefined model library. In this context, the training procedure for the classifier corresponds to an inverse problem. Indeed, training examples are generated by running simulations with all the models in the library and evaluating their predictions at the sensors' locations. Therefore, for a given model $y \in [\![1;K]\!]$, the data x are obtained by means of numerical simulations performed with y. This corresponds to the forward mapping. The classifier must learn the inverse mapping giving y as a function of x. In this example, data labeling is straightforward: the label of a training example x is given by the index y of the model which was used to generate x. It is also noteworthy that generating training examples is not too expensive, because numerical simulations are performed with reduced-order models obtained by the Static-Condensation Reduced-Basis-Element method (SCRBE [63–66]). In this application, automatic model recommendation gives the UAV the ability to dynamically evaluate its flight capability and replan its mission accordingly.

Another example of classifier used to accelerate numerical simulations can be found in [4]. Contrary to the works in [8,9], the data labeling procedure relies on the clustering of simulation data. In this framework, the model library is made of cluster-specific Discrete Empirical Interpolation Method (DEIM) [67] models that are faster than the high-fidelity model. The high-fidelity model computes a prediction u_i for each input x_i in the database $\{x_i\}_{1 \leq i \leq N}$, resulting in a dataset $\{u_i\}_{1 \leq i \leq N}$ on which a clustering algorithm is applied. The predicted variable u is the discretization of a continuous field on a finite-element mesh, thus living in a high-dimensional space. To avoid the so-called *curse of dimensionality* [28], a DEIM-based feature selection technique is used before applying k-means clustering [68]. Alternatively, the clusters can be obtained with a variant of k-means using the DEIM residual as clustering criterion. Then, for a given training example x_i, the class label y_i is defined by the index of the cluster that u_i is assigned to. In the exploitation phase, when dealing with test data, the best DEIM model is selected by a nearest neighbor classifier. The input data given to the classifier are either parameters of the problem or the variable u obtained at the previous time increment. A similar methodology is described in [5], where the concept of model library is termed *model dictionary*, which is the terminology adopted in this paper. The model dictionary is made of hyper-reduced-order models [69], and the input data $\{x_i\}_{1 \leq i \leq N}$ are images of a mechanical experiment. The dimensionality of simulation data is reduced by Principal Component Analysis (PCA) before using k-means clustering. A convolutional neural network [70] is trained to return class labels without computing the intermediate variable u in order to avoid time-consuming operations. This classifier is an approximation of the *true classifier* $\mathcal{K}$ returning the correct label for any input x.

3. Definition of the Classification Problem

Notations: The j-th feature of a random vector X is the real-valued random variable denoted by X^j. Its observations are denoted by x^j, or x_i^j when indexing is necessary, for example, when considering training data. When X is obtained by discretizing a random field on a mesh, the feature X^j corresponds to the value taken by the random field at the j-th node. In the numerical application presented in this work, a random temperature field is considered. The spatial coordinates of the j-th node are stored in a vector $\boldsymbol{\xi}_j \in \mathbb{R}^3$. The categorical variable Y indicates which model should be used.　□

In this paper, input data $\{x_i\}_{1 \leq i \leq N}$ correspond to several instances or *variabilities* of a physical field discretized on a mesh. Let $\mathcal{N}$ be the number of nodes in the mesh. If the physical field is scalar and defined at the nodes, then each observation x_i is a vector of

$\mathbb{R}^{\mathcal{N}}$. For relatively small problems, $\mathcal{N}$ is in the order of 10^4 to 10^5. For some industrial problems, $\mathcal{N}$ can be in the order of 10^6 to 10^8. The dataset $\{x_i\}_{1 \leq i \leq N}$ may come from experiments, numerical simulations, statistical models, or a combination of them, and contains from 10^2 to 10^4 observations. It is assumed that all features of all observations are known, contrary to some classification tasks in other disciplines encountering the problem of missing values. This assumption is clearly satisfied when data come from numerical simulations or statistical models. For experimental data, numerous techniques provide space-distributed measurements that can be projected onto the mesh, such as particle image velocimetry [71] in fluid dynamics, digital image correlation [72] and photoelastic experiments [73] in solid mechanics, and temperature-sensitive paints [74] measuring surface temperatures.

The framework considered in this paper is the same as in [11] for ROM-nets, where the input variabilities are supposed to be used for an uncertainty propagation study in a physics problem $\mathcal{P}$, for which a high-fidelity model m_{HF} is available. The physics problem $\mathcal{P}$ is a time-dependent problem. As the high-fidelity model is too computationally expensive, dictionary-based ROM-nets have been introduced to reduce the computation time by means of a reduced-order model dictionary and a classifier playing the role of a model selector. The dictionary-based ROM-net is trained on the available dataset $\{x_i\}_{1 \leq i \leq N}$. For a given observation x_i, the class label y_i indicates the most appropriate model in the dictionary to be used for fast simulations with limited errors with respect to the high-fidelity model m_{HF}. Class labels are obtained by the following data labeling procedure.

- **Step 1:** For each observation x_i in the dataset, use the high-fidelity model m_{HF} to solve a simplified version $\mathcal{P}'$ of the physics problem $\mathcal{P}$ (for example, the problem $\mathcal{P}'$ can consist in solving $\mathcal{P}$ for a few time increments only). The primal solution of $\mathcal{P}'$ computed for x_i is denoted by u_i. It consists of a collection $\{u_i^n\}_{1 \leq n \leq n_t}$ of n_t fields defined on the mesh, with n_t being the number of time increments in problem $\mathcal{P}'$.

- **Step 2:** Given $\{u_i\}_{1 \leq i \leq N}$, compute the dissimilarity matrix $\delta \in \mathbb{R}^{N \times N}$ with the following formula:

$$\delta_{ij} = \delta(x_i, x_j) = d_{\mathrm{Gr}(\infty,\infty)}\left(\mathrm{span}(\{u_i^n\}_{1 \leq n \leq n_t}), \mathrm{span}(\{u_j^n\}_{1 \leq n \leq n_t})\right)$$

  with $d_{\mathrm{Gr}(\infty,\infty)}$ being the Grassmann metric defined in [75]. The coefficient δ_{ij} is a dissimilarity measure between x_i and x_j.

- **Step 3:** k-medoids clustering [76–78] is applied to the dissimilarity matrix δ. In this paper, we consider $K = 4$ clusters. The label $y_i = \mathcal{K}(x_i) \in [\![1; K]\!]$ is given by the index of the cluster containing u_i.

This procedure gives $N = 1000$ examples of input–label pairs $\{(x_i, y_i)\}_{1 \leq i \leq N}$. This dataset is split into a training set, a validation set, and a test set with cardinalities 600, 200, and 200, respectively, enabling the supervised training and evaluation of a classifier $\mathcal{C}$. For the sake of simplicity, the labeled data are renumbered so that the $N_{\text{train}} = 600$ first input–output pairs $\{(x_i, y_i)\}_{1 \leq i \leq N_{\text{train}}}$ form the training set on which the feature selection and data augmentation algorithms presented in this paper are trained.

In this work, the physics problem $\mathcal{P}$ is a temperature-dependent mechanical problem. The structure is made of an elasto-viscoplastic material whose behavior depends on the local value of the temperature field [79]. The random variable X is a random vector representing the evaluation of the random temperature field on a finite-element mesh containing $\mathcal{N} = 42,445$ nodes (see Figure 1). The structure is subjected to centrifugal forces and pressure loads. The random temperature fields are generated by a stochastic model described in [11], where ten fluctuation modes are randomly combined and superposed to a reference temperature field. The realizations of the random temperature field are continuous and always satisfy the heat equation. Modeling random fields as random combinations of deterministic spatial functions is quite common when studying stochastic partial differential equations [80–82], because a random field can be approximated by truncating its Karhunen–Loève expansion [83].

As already stated, the main contributions of this paper are a feature selection strategy and a data augmentation algorithm adapted to the specificities and difficulties of classification problems encountered when training dictionary-based ROM-nets. Concerning feature selection, the main focus is on the fast quantification of features redundancy by taking advantage of the type of input data. Concerning data augmentation, in addition to the constraints that have already been mentioned, it is likely that transforming an input example x_i substantially modifies the intermediate variable u_i, and thus the class label y_i might no longer be relevant for the transformed input. Avoiding this situation is crucial to ensure that the augmented data are correctly labeled. Our algorithms are applicable under the assumptions that the random vector X derives from a random field whose realizations are continuous with probability one (sample path continuity, see Definition 2.1 in [84]) and belong to a convex domain $\mathcal{X}$ related to physics constraints. Last, a comparison of various classification algorithms is conducted to put into perspective the choice made in [11] to use an ensemble of deep neural networks trained with different architectures and loss functions.
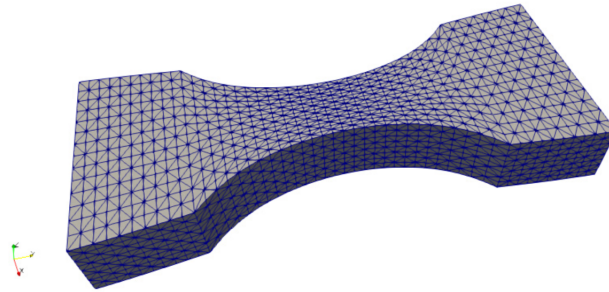

Figure 1. Finite-element mesh of the structure considered in this paper.

Remark 1. *Another strategy would consist in using a regression algorithm for the classification task. Indeed, as our data labeling procedure is based on clustering, the classification problem could be replaced by a regression problem for the prediction of dissimilarities $\{\delta(x, \tilde{x}_k)\}_{1 \leq k \leq K}$ for $x \in \mathcal{X}$, with $\tilde{x}_k$ being the medoid of the k-th cluster. Given these distances for a new observation x, the class label is obtained by taking the integer $k \in [\![1; K]\!]$ associated to the smallest dissimilarity $\delta(x, \tilde{x}_k)$. However, the data augmentation algorithm presented in this paper is not compatible with regression algorithms. For this reason, this paper focuses on classifiers rather than regressors.*

4. Feature Selection

4.1. Feature Selection Based on Mutual Information

We recall that a projection π is a linear map satisfying $\pi \circ \pi = \pi$, with $\circ$ denoting function composition. It is entirely defined by its kernel and its image, which are complementary: given two complementary vector subspaces V_1 and V_2, there is a unique projection π whose kernel is V_1 and whose image is V_2, namely, the projection onto V_2 along V_1. For more details about projections, see in [85], pages 385 to 388. Let us now give a formal definition of a *feature selector*:

Definition 1. *(Feature selector) Let V be a finite-dimensional real vector space. Given a basis $\mathcal{B} = (e_i)_{1 \leq i \leq \dim(V)}$ of V and a set of integers $S \subset [\![1; \dim(V)]\!]$, the feature selector $\pi_{S,\mathcal{B}} : V \to V$ is the projection whose image is $\mathrm{span}(\{e_i\}_{i \in S})$ and whose kernel is $\mathrm{span}\left(\{e_i\}_{i \in [\![1; \dim(V)]\!] \setminus S}\right)$.*

When the choice of the basis $\mathcal{B}$ is obvious, the notation $\pi_{S,\mathcal{B}}$ is simply replaced by π_S. In practice,

$$\forall (\lambda_i)_{1 \leq i \leq \dim(V)} \in \mathbb{R}^{\dim(V)}, \quad \pi_S \left(\sum_{i=1}^{\dim(V)} \lambda_i e_i \right) = \sum_{i \in S} \lambda_i e_i \tag{8}$$

Therefore, from a numerical point of view, one can interpret the feature selector as linear map $\pi_S : V \to \mathrm{span}(\{e_i\}_{i \in S})$, which enables reducing the size of the vector representing $\pi_S(x)$ for $x \in V$. In this way, applying a feature selector π_S to a vector of $\mathbb{R}^{\mathcal{N}}$ consists in masking its features whose indexes are not in S, which gives a reduced vector in $\mathbb{R}^{|S|}$ where $|S|$ denotes the number of elements in S. Feature selection algorithms build the set S by searching for the most relevant features for the prediction of the output variable Y. For this purpose, the mutual information can be used to quantify the degree of the relationship between variables.

Definition 2 (Mutual information [86], eq. 8.47, p. 251). *Let Z^1 and Z^2 be two real-valued random variables with joint probability distribution $p_{1,2}$ and marginal distributions p_1 and p_2. The mutual information $I(Z^1, Z^2)$ is defined by*

$$I\left(Z^1, Z^2\right) = \int_{\mathbb{R}^2} p_{1,2}(z^1, z^2) \log\left(\frac{p_{1,2}(z^1, z^2)}{p_1(z^1) p_2(z^2)}\right) dz^1 dz^2 \tag{9}$$

The mutual information measures the mutual dependence between two random variables. Contrary to correlation coefficients, the information provided by this score function is not limited to linear dependence. The mutual information is non-negative, and equals to zero if and only if the random variables are independent. Given Equation (2), replacing Z^1 by a feature X^i of X and Z^2 by Y gives

$$I\left(X^i, Y\right) = \sum_{k=1}^{K} \mathbb{P}_Y(k) \int_{x^i \in \mathbb{R}} p_{X^i|Y}(x^i|k) \log\left(\frac{p_{X^i|Y}(x^i|k)}{p_{X^i}(x^i)}\right) dx^i \tag{10}$$

The mutual information can be used to quantify the redundancy of a set of features S with cardinality $|S|$ and its relevance for predicting Y:

Definition 3 (Relevance [22], eq. 4, p. 2). *Let $X = (X^i)_{1 \le i \le \mathcal{N}}$ be a multivariate random variable, and let Y be a discrete random variable. The relevance of a reduced set $S \subset [\![1; \mathcal{N}]\!]$ of features of X for predicting Y is defined by*

$$D(S, Y) = \frac{1}{|S|} \sum_{i \in S} I(X^i, Y) \tag{11}$$

Definition 4 (Redundancy [22], eq. 5, p. 2). *Let $X = (X^i)_{1 \le i \le \mathcal{N}}$ be a multivariate random variable. The redundancy of a reduced set $S \subset [\![1; \mathcal{N}]\!]$ of features of X is defined by*

$$R(S) = \frac{1}{|S|^2} \sum_{i,j \in S^2} I(X^i, X^j) \tag{12}$$

The *minimum redundancy maximum relevance* (mRMR) algorithm [21,22] builds the set S by maximizing $D(S, Y) - R(S)$, which is a combinatorial optimization problem. For this type of optimization problem, a brute-force search is intractable, because the number of solution candidates is too large. Instead, mRMR searches for a sub-optimal solution by following a greedy approach. First, the feature having the highest mutual information with the label variable Y is selected. Then, the algorithm follows an incremental procedure: given the set S_{m-1} obtained at iteration $m-1$, form the set S_m such that

$$S_m = S_{m-1} \cup \left\{ \underset{i \in [\![1; \mathcal{N}]\!] \setminus S_{m-1}}{\arg\max} \left(I(X^i, Y) - \frac{1}{m-1} \sum_{j \in S_{m-1}} I(X^i, X^j) \right) \right\} \tag{13}$$

This incremental procedure stops when m reaches the target number of features N_f. A review of feature selection algorithms based on mutual information can be found in [87].

4.2. A Geostatistical Variant of mRMR Feature Selection

When training dictionary-based ROM-nets, the number of features of the random vector X scales with the number of nodes $\mathcal{N}$ in the mesh. In particular, the number of features is exactly $\mathcal{N}$ if X is the nodal representation of a scalar field. Therefore, there are too many features to compute all redundancy terms $I(X^i, X^j)$. However, one can estimate the redundancy terms thanks to the proximities of the features on the mesh. Indeed, X is a regionalized variable: in our example, we recall that $\xi_i \in \mathbb{R}^3$ denotes the position of the i-th node in the mesh, and that the feature X^i corresponds to the value taken by a random temperature field at ξ_i. If two points ξ_i and ξ_j of the mesh are close to each other, the corresponding features X^i and X^j are likely to be correlated and thus redundant because of the smoothness of the temperature field. This idea is also valid when considering physical variables discretized in time.

In this paper, the random temperature field is modeled by a Gaussian random field [84] as in [11], which is a common and simple approach when modeling uncertainties on a physical field. As a consequence, X is a Gaussian random vector and the mutual information $I(X^i, X^j)$ has a simple formula involving the correlation coefficient:

Property 1 (Mutual information of two correlated Gaussian random variables [86], eq. 8.56, p. 252). *Let (X^1, X^2) be a Gaussian random vector. The mutual information $I(X^1, X^2)$ reads*

$$I(X^1, X^2) = -\frac{1}{2}\ln\left(1 - \rho^2\right) \tag{14}$$

where ρ denotes the correlation between X^1 and X^2.

This property implies that, for Gaussian random fields having isotropic correlation functions ρ (the correlation function $\rho(\xi, \xi')$ of a random field is isotropic if it only depends on the distance $||\xi - \xi'||_2$), the mutual information $I(X^i, X^j)$ only depends on the distance $||\xi_i - \xi_j||_2$. A wide variety of isotropic correlation functions are given in [84]. More generally, as Equation (14) is an increasing function of ρ^2, any isotropic upper (resp. lower) bound of the squared correlation function gives an isotropic upper (resp. lower) bound of the mutual information.

For the example studied in this paper, Figure 2 shows that the mutual information $I(X^i, X^j)$ decreases as the corresponding distance $||\xi_i - \xi_j||_2$ increases. Therefore, our feature selection algorithm builds a metamodel $\tilde{I}$ replacing $I(X^i, X^j)$ by a function of the distance $||\xi_i - \xi_j||_2$, which drastically reduces the computational cost of mRMR algorithm for our particular problem. First of all, one must build a design of experiments (DOE) to select a few terms $I(X^i, X^j)$ to be computed exactly. The metamodel $\tilde{I}$ is calibrated to fit the corresponding precomputed redundancy terms. Then, mRMR feature selection is applied by replacing $I(X^i, X^j)$ with $\tilde{I}(||\xi_i - \xi_j||_2)$. The feature selection algorithm is described in Algorithm 1. We call this algorithm *geostatistical mRMR*, as geostatistics is the branch of statistics that deals with regonalized variables. A stopping criterion is added to the incremental procedure used in mRMR, enabling an automatic selection of the number of features to be kept for the classification task: the algorithm stops when the value of $\underset{i\in[\![1;\mathcal{N}]\!]\backslash S_m}{\arg\max}\left(I(X^i, Y) - \frac{1}{m}\sum_{j\in S_m}\tilde{I}\left(\left\|\xi_i - \xi_j\right\|_2\right)\right)$ has not changed much during a number of iterations. A condition on the mutual information $I(X^i, Y)$ can also be added to avoid selecting quasi-irrelevant features. It should be noted that the number of selected features does not depend on the number of nodes $\mathcal{N}$ in the mesh. In addition, for stage 1 of Algorithm 1, computing all the terms $\left\|\xi_{j_1} - \xi_{j_2}\right\|_2$ of the matrix of pairwise mesh nodes distances is not necessary: only a few lines of this matrix corresponding to randomly selected nodes are evaluated, which is sufficient to build the DOE. In other words, one computes the distances between a few nodes and all the mesh nodes.
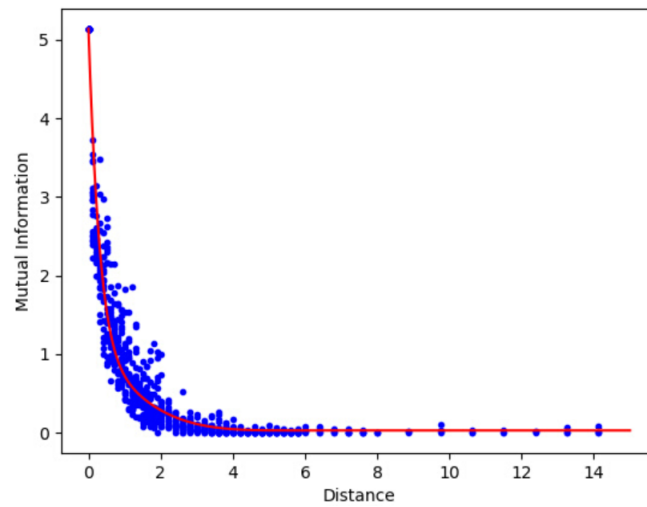
Figure 2. Mutual information $I(X^i, X^j)$ as a function of the distance $\|\xi_i - \xi_j\|_2$.

Algorithm 1 Geostatistical mRMR

Input: training set $\{(x_i, y_i)\}_{1 \leq i \leq N_{\text{train}}}$, set of mesh nodes $\{\xi_i\}_{1 \leq i \leq \mathcal{N}}$, stopping criterion.
Output: set of selected features.

1: **Stage 1 (design of experiments):**
2: 　　Select distance values r_j.
3: 　　For each r_j, draw n_j pairs of mesh nodes (ξ_{j_1}, ξ_{j_2}) such that $\left\|\xi_{j_1} - \xi_{j_2}\right\|_2 \approx r_j$.
4: **Stage 2 (metamodel for redundancy terms):**
5: 　　Compute the mutual information $I(X^i, X^j)$ for each pair selected in Stage 1.
6: 　　Train a metamodel $\tilde{I}$ such that $I(X^i, X^j) \approx \tilde{I}\left(\left\|\xi_i - \xi_j\right\|_2\right)$.
7: **Stage 3 (compute relevance terms):**
8: 　　Compute $I(X^i, Y)$ for all $i \in [\![1; \mathcal{N}]\!]$.
9: **Stage 4 (greedy feature selection):**
10: 　　$S_1 := \underset{i \in [\![1; \mathcal{N}]\!]}{\arg\max}\ I(X^i, Y)$
11: 　　$m := 1$
12: 　　**while** stopping criterion not satisfied **do**
13: 　　　　$S_{m+1} := S_m \cup \{ \underset{i \in [\![1; \mathcal{N}]\!] \setminus S_m}{\arg\max} \left(I(X^i, Y) - \frac{1}{m} \sum_{j \in S_m} \tilde{I}\left(\left\|\xi_i - \xi_j\right\|_2\right) \right) \}$
14: 　　　　$m := m + 1$
15: 　　**end while**
16: **return** S_m

Remark 2. *A parallel can be drawn between our feature selection strategy and hyper-reduction methods [69,88–90] used to accelerate complex nonlinear problems in physics (see in [91] for design optimization and [92] for large-scale simulations). Hyper-reduction methods aim at finding a reduced set of integration points in the finite-element mesh that is sufficient to predict the behavior of the physical system. The constitutive equations are solved on this reduced integration domain only, while the values of quantities of interest at the remaining integration points can be recovered with the Gappy-POD [93]. In short, hyper-reduced solvers make predictions from a reduced number of points in a mesh, like the classifiers used in this paper do when combined with the geostatistical mRMR. Although the objectives are different, both hyper-reduction and geostatistical mRMR feature selection benefit from the properties of physics data to reduce the complexity of numerical tasks.*

4.3. Numerical Results

The red curve on Figure 2 corresponds to the metamodel estimating redundancy terms. In this example, we choose

$$\tilde{I}(r) = I_\infty + \gamma_1 (r_1 - r)^{\alpha_1} H(r_1 - r) + \gamma_2 (r_2 - r)^{\alpha_2} H(r_2 - r) \tag{15}$$

where H is the Heaviside step function and $I_\infty, \gamma_1, \gamma_2, r_1, r_2, \alpha_1, \alpha_2$ are calibration parameters that are adjusted manually. In the DOE, the step between distances r_j is smaller for small distances, in order to better capture the evolution of the mutual information in its high gradient regime. The number n_j of pairs of nodes separated by a distance of r_j selected in the DOE also depends on r_j: as higher variances were expected for small distances, n_j decreases when r_j increases. In total, 749 terms $I(X^i, X^j)$ are computed, which takes 5.12 s using Scikit-learn [94]. Building the DOE takes only 0.33 s. Then, the greedy procedure takes 303 s and selects 87 features among the 42,445 original ones. The first iteration is the longest one with 276 s, because it includes the computation of all the relevance terms $I(X^i, Y)$. As a comparison, the original mRMR algorithm takes 6469 s to compute 7 iterations only. We did not let mRMR algorithm go further, as the per-iteration computation time grows with the iteration number. For a fair comparison, our implementations of mRMR and stages 3 and 4 of the geostatistical mRMR are the same except that redundancy terms are evaluated with Scikit-learn for mRMR and with the function $\tilde{I}$ for the geostatistical mRMR.

Table 1 compares the relevance $D(S, Y)$, the true redundancy $R(S)$, the approximate redundancy $\tilde{R}(S)$ estimated with $\tilde{I}$, the true cost function $D(S, Y) - R(S)$, and the approximate cost function $D(S, Y) - \tilde{R}(S)$ for three different feature selection strategies:

- the geostatistical mRMR feature selection (Algorithm 1), selecting a set S^* of features;
- a univariate filter algorithm selecting the features with the highest mutual information (MI) scores $I(X^i, Y)$. This algorithms finds a set S^{MI} maximizing the relevance for a given cardinality; and
- a purely geometric feature selection algorithm, randomly selecting the first feature and adding features in a greedy manner so that the distance to the closest point ξ_i, $i \in S_m$ is maximized. This algorithm tends to select a set S^G of well-distributed features in order to get a low redundancy for a given cardinality.

As the geostatistical mRMR automatically selected 87 features, the two other approaches are applied with $|S^G| = |S^{MI}| = 87$ as a target. Table 1 shows that the relevance of the set S^* selected by our algorithm is in the same order of magnitude as the relevance of the set S^{MI}. Its redundancy is in the same order of magnitude as the redundancy of the set S^G. These results show that the geostatistical mRMR algorithm does have the desired behavior: it selects a subset of features S^* with high relevance and low redundancy. Figure 3 shows the features selected by the three different algorithms. The classification accuracies of several classifiers using the reduced features S^* are given in the last section of the article.

Table 1. Evaluation of the geostatistical mRMR feature selection algorithm.

Algorithm	$D(S, Y)$	$\tilde{R}(S)$	$R(S)$	$D(S, Y) - \tilde{R}(S)$	$D(S, Y) - R(S)$
Geostatistical mRMR (S^*)	0.0460	0.0816	0.1111	-0.0356	-0.0651
MI-based filter (S^{MI})	0.0671	0.9794	0.8129	-0.9124	-0.7458
Geometric filter (S^G)	0.0090	0.0788	0.1072	-0.0699	-0.0982

Remark 3. *The geometric feature selection algorithm gives rather good results in terms of the cost function, but it does not mean that it is an appropriate approach. Indeed, one can see that the relevance of S^G is very low, as this algorithm does not use any information concerning the classification problem.*
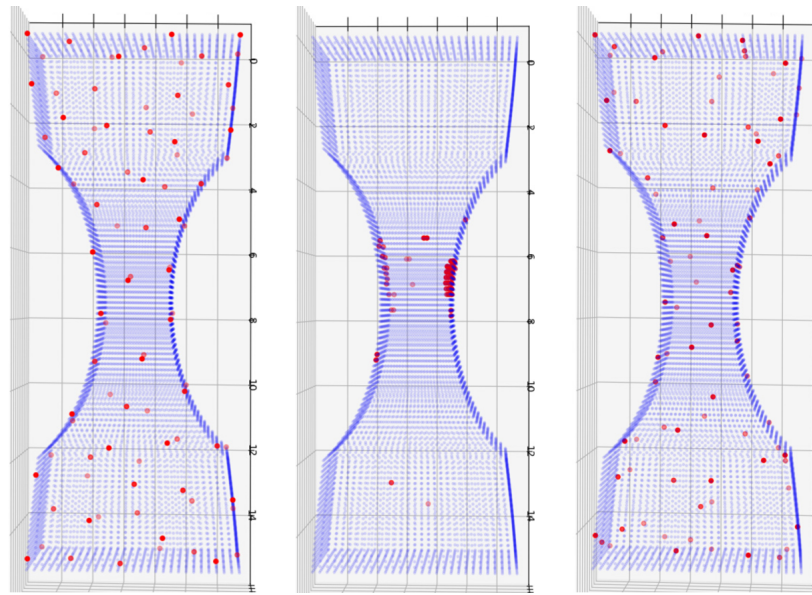
Figure 3. Red dots indicate the selected features. From the left to the right: geometric feature selection, MI-based feature selection, and geostatistical mRMR.

5. Data Augmentation

5.1. Pure Sets

Definition 5 (Convex set [95], p. 10)**.** *Let V be a real vector space. A non-empty set $\mathcal{S} \subset V$ is convex if*

$$\forall (x_1, x_2) \in \mathcal{S}^2, \ \forall \lambda \in [0;1], \quad \lambda x_1 + (1 - \lambda) x_2 \in \mathcal{S} \tag{16}$$

Definition 6 (Convex combination [95], p. 11)**.** *Let $\{x_i\}_{1 \leq i \leq n}$ be a finite set of elements of a real vector space V. A convex combination of $\{x_i\}_{1 \leq i \leq n}$ is a vector $x \in V$ such that*

$$\exists \, (\lambda_i)_{1 \leq i \leq n} \in \mathbb{R}_+^n \quad | \quad \sum_{i=1}^n \lambda_i = 1 \quad and \quad x = \sum_{i=1}^n \lambda_i x_i \tag{17}$$

Definition 7 (Convex hull of a set [95], p. 12)**.** *Let V be a real vector space and $\mathcal{S}$ a non-empty set included in V. The convex hull or convex envelope $\mathcal{E}(\mathcal{S})$ of $\mathcal{S}$ is the smallest convex set containing $\mathcal{S}$. Equivalently, the convex hull $\mathcal{E}(\mathcal{S})$ can be defined as the set of all convex combinations of all finite subsets of $\mathcal{S}$.*

Property 2 (Image of a convex hull by a linear map)**.** *Let V and W be two real vector spaces, and let $\mathcal{L} : V \to W$ be a linear map. Let $\mathcal{S}$ be a non-empty set included in V. Then,*

$$\mathcal{L}(\mathcal{E}(\mathcal{S})) = \mathcal{E}(\mathcal{L}(\mathcal{S})) \tag{18}$$

Proof. Let $z \in \mathcal{E}(\mathcal{L}(\mathcal{S}))$. Following the definition of a convex hull, there exists $n \in \mathbb{N}^*$ such that

$$\exists \, (w_i)_{1 \leq i \leq n} \in \mathcal{L}(\mathcal{S})^n, \ \exists \, (\lambda_i)_{1 \leq i \leq n} \in \mathbb{R}_+^n \quad | \quad \sum_{i=1}^n \lambda_i = 1 \text{ and } z = \sum_{i=1}^n \lambda_i w_i \tag{19}$$

For all $i \in [\![1; n]\!]$, as $w_i \in \mathcal{L}(\mathcal{S})$, there exists $v_i \in S$ such that $w_i = \mathcal{L}(v_i)$. By linearity of $\mathcal{L}$:

$$z = \sum_{i=1}^n \lambda_i \mathcal{L}(v_i) = \mathcal{L} \left(\sum_{i=1}^n \lambda_i v_i \right) \in \mathcal{L}(\mathcal{E}(\mathcal{S})) \tag{20}$$

so $\mathcal{E}(\mathcal{L}(\mathcal{S})) \subset \mathcal{L}(\mathcal{E}(\mathcal{S}))$. The other inclusion can be shown using exactly the same arguments. Thus, $\mathcal{L}(\mathcal{E}(\mathcal{S})) = \mathcal{E}(\mathcal{L}(\mathcal{S}))$. $\square$

This property has a very simple yet important consequence for the data augmentation algorithm presented in this paper.

Property 3. *Let V and W be two real vector spaces, and let $\mathcal{L} : V \to W$ be a linear map. Let $\mathcal{S}$ be a non-empty set included in V. Then, for all $x \in V$:*

$$\mathcal{L}(x) \notin \mathcal{E}(\mathcal{L}(\mathcal{S})) \Rightarrow x \notin \mathcal{E}(\mathcal{S}) \tag{21}$$

Proof. By contraposition, $x \in \mathcal{E}(\mathcal{S}) \Rightarrow \mathcal{L}(x) \in \mathcal{L}(\mathcal{E}(\mathcal{S})) = \mathcal{E}(\mathcal{L}(\mathcal{S}))$. $\square$

Our data augmentation strategy uses this property in the particular case where the linear map is a projection. As a reminder, the notation $\mathcal{K}$ stands for the true classifier assigning any input x to a single label $y \in [\![1; K]\!]$. Before giving the description of the algorithm, let us introduce the definition of *pure sets* in a labeled dataset and a characterization theorem:

Definition 8 (Pure set). *Let n be a positive integer, and let $\mathcal{S} = \{x_i\}_{1 \leq i \leq n}$ be a finite set of elements of a real vector space V labeled by $\mathcal{K}$. Let $\mathcal{S}_{\mathcal{I}} = \{x_i\}_{i \in \mathcal{I} \subset [\![1;n]\!]}$ be a non-empty subset of $\mathcal{S}$. The set $\mathcal{S}_{\mathcal{I}}$ is pure in $\mathcal{S}$ if $\mathcal{K}(\mathcal{S} \cap \mathcal{E}(\mathcal{S}_{\mathcal{I}}))$ is a singleton, which means that the set $\mathcal{S}_{\mathcal{I}}$ is pure in $\mathcal{S}$ if all of the points of $\mathcal{S}$ that belong to the convex hull of $\mathcal{S}_{\mathcal{I}}$ have the same label.*

Let $\mathcal{S} = \{x_i\}_{1 \leq i \leq n}$ be a finite set of elements of a finite-dimensional real vector space V labeled by integers $\{y_i\}_{1 \leq i \leq n}$ in $[\![1; K]\!]$, with $K \leq n$. For all $k \in [\![1; K]\!]$, C_k denotes the set of elements of $\mathcal{S}$ labeled by k:

$$C_k = \{x_i \in \mathcal{S} \mid y_i = k\} \tag{22}$$

For any subset S_k of C_k with cardinality $|S_k|$, $\hat{A}_{S_k} \in \mathbb{R}^{\dim(V) \times |S_k|}$ denotes the matrix whose columns contain the coordinates of the elements of S_k. The matrix denoted by A_{S_k} is obtained by adding a row of ones below the matrix $\hat{A}_{S_k}$, giving a matrix of size $(1 + \dim(V)) \times |S_k|$.

Theorem 1 (Pure set characterization). *Let S_k be a subset of C_k with cardinality $|S_k|$. The set S_k is pure in $\mathcal{S}$ if and only if for all x in $\mathcal{S} \setminus C_k$ the linear system:*

$$A_{S_k} w = \begin{pmatrix} x \\ 1 \end{pmatrix} \tag{23}$$

has no non-negative solution $w \in \mathbb{R}_+^{|S_k|}$.

Proof. Let $x \in \mathcal{S} \setminus C_k$. Equation (23) has no non-negative solution if and only if

$$\nexists\, w \in \mathbb{R}_+^{|S_k|} \quad | \quad \sum_{i=1}^{|S_k|} w_i = 1 \text{ and } \hat{A}_{S_k} w = x \tag{24}$$

$$\iff x \notin \mathcal{E}(S_k) \tag{25}$$

which ends the proof. $\square$

Corollary 1 (Pure set testing). *Let S_k be a subset of C_k with cardinality $|S_k|$, and let $\mathcal{L} : V \to W$ be a linear map, where W is a finite-dimensional real vector space. If for all x in $\mathcal{S} \setminus C_k$ the linear system*

$$A_{\mathcal{L}(S_k)} w = \begin{pmatrix} \mathcal{L}(x) \\ 1 \end{pmatrix} \tag{26}$$

has no non-negative solution in $\mathbb{R}_+^{|S_k|}$, then S_k is pure in $\mathcal{S}$.

Proof. Equation (26) characterizes the purity of $\mathcal{L}(S_k)$ in $\mathcal{L}(\mathcal{S})$ (Theorem 1), which implies that S_k is pure in $\mathcal{S}$ (Property 3). □

Figure 4 illustrates the concept of pure sets. On this figure, the set C_1 is made of all the elements represented by dots, while the crosses form the set $C_2 = \mathcal{S} \setminus C_1$. On the left, the subset formed by the six black dots is pure since its convex hull delimited by dashed lines contains only dots. The subset made of the six black dots on the right-hand side of the figure is not pure because of the presence of a cross in its convex hull. Equation (23) has a non-negative solution when using the coordinates of this cross in its right-hand side.
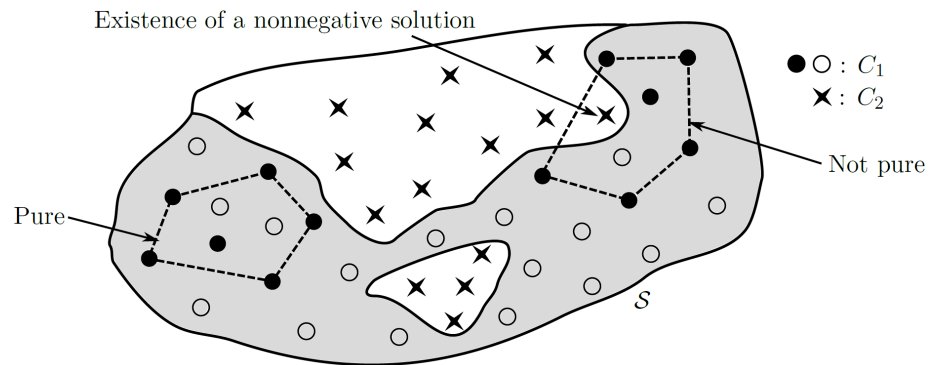

Figure 4. Illustration of the concept of pure sets on a binary classification problem.

5.2. The Data Augmentation Algorithm

The objective is to generate new data points $x \in \mathcal{X}$ in a given class $y \in [\![1; K]\!]$ from the preexisting observations in that class. To this end, one must apply class-preserving transformations on the training examples. New examples can be created by taking convex combinations of some subsets of the training set, for example. One way of controlling the risk that newly generated examples have wrong labels is to take convex combinations of subsets only if they are pure. Indeed, if the k-th class C_k contains a set S_k that is pure in the training set, one can expect that the probability $\mathbb{P}(Y = k \mid X \in \mathcal{E}(S_k))$ is high enough to get new examples of class C_k by drawing samples in $\mathcal{E}(S_k)$. In addition, the third Hadamard well-posedness condition states that the solution of a physics equation changes continuously with respect to the parameters of the problem. In the neighborhood of a point x_0 belonging to a pure set S_k, the primal solution u stays in the neighborhood of the solution u_0 obtained with x_0 and is thus likely to have the same label. Therefore, the objective of our algorithm is to find pure sets in the training set in order to generate new examples by convex combinations with a limited risk of getting incorrectly labeled examples. The pure sets detected by the algorithm are listed in a matrix $\mathcal{S}$ such that $\mathcal{S}[k, i]$ contains the indices of the training points forming the i-th pure set of the k-th class. The pure sets are grown from different starting points or seeds in the training set by iteratively adding the seeds' nearest neighbors in terms of the precomputed dissimilarity measure δ used for clustering in the data labeling procedure. The growth stops before losing the purity of the subsets. However, checking the purity in the high-dimensional input space can cause difficulties, even when training the data augmentation algorithm after a first dimensionality reduction like in this paper. For this reason, the algorithm checks the purity after having applied a feature selector π_S with a small random subset of features S containing d features. Let us apply Property 3 with $V = W$ being the input vector space containing $\mathcal{X}$ and with the linear map $\mathcal{L}$ being the feature selector π_S. As Property 3 states, if no point of $\pi_S\left(\{x_m\}_{1 \leq m \leq N_{\text{train}}} \setminus C_k\right)$ belongs to the convex hull of $\pi_S\left(\{x_m\}_{m \in \mathcal{S}[k,i]}\right)$, then the set $\mathcal{E}\left(\{x_m\}_{m \in \mathcal{S}[k,i]}\right)$ does not contain any point labeled with $k' \neq k$. As a set can lose its purity after projection, the algorithms tries $p_{\max}$ random feature selectors π_S before considering that the set is not pure. In practice, the purity of $\pi_S\left(\{x_m\}_{m \in \mathcal{S}[k,i]}\right)$ in $\pi_S\left(\{x_m\}_{1 \leq m \leq N_{\text{train}}}\right)$

is numerically tested by solving a non-negative least squares (NNLS [96]) problem. If for all points $x \in \{x_m\}_{1 \leq m \leq N_{\text{train}}} \setminus C_k$ the inequality

$$\min_{w \in \mathbb{R}_+^{|\mathcal{S}[k,i]|}} ||A_{\pi_S(\{x_m\}_{m \in \mathcal{S}[k,i]})} w - \tilde{\pi}_S(x)||_2 \geq \varepsilon_{\text{DA}} ||\tilde{\pi}_S(x)||_2 \tag{27}$$

is satisfied with $\tilde{\pi}_S(x) = (\pi_S(x)^T \ 1)^T$ and with ε_{DA} being the tolerance of the data augmentation algorithm, then Corollary 1 implies that $\{x_m\}_{m \in \mathcal{S}[k,i]}$ is pure in $\{x_m\}_{1 \leq m \leq N_{\text{train}}}$. Algorithm 3 describes the data augmentation algorithm. It calls Algorithm 2 to find n well-distributed seeds per class before growing pure sets. It is noteworthy that using few pure sets to generate many examples would increase the distribution gap [97] between augmented data and original data. To avoid this issue, one had better use many well-distributed seeds to distribute data augmentation efforts between the pure sets and thus limit the divergence between the augmented distribution and the true data-generating distribution.

Remark 4. *Realizations of the random variable X belong to a convex domain $\mathcal{X}$ related to physics constraints. When considering surface random temperature fields defined on the boundaries of a solid, $\mathcal{X}$ is a hypercube consisting of all the fields taking values between zero Kelvin degree and the material's melting point. These random fields can be used as Dirichlet boundary conditions for the nonlinear heat equation. The assumption of a linear thermal behavior is added when considering three-dimensional random temperature fields defined inside the solid, so that the set $\mathcal{X}$ remains convex when adding the constraint that the random field must satisfy the heat equation. More generally, convex combinations respect physics constraints defined by linear operators, such as linear partial differential equations and Dirichlet, Neumann, and Robin boundary conditions.*

Algorithm 2 Seeds selection for data augmentation. Note: all the dissimilarities have already been computed in the data labeling procedure.

Input: training set $\{(x_i, y_i)\}_{1 \leq i \leq N_{\text{train}}}$, class label k, class center $\tilde{x}_k$, dissimilarity matrix δ, target number of seeds n, preselection parameters $(\varepsilon_1, \varepsilon_2) \in [0; 1]^2$.
Output: List l_k of n indices of seeds candidates for the k-th class.
1: **Stage 1 (filter the data):**
2: Find the minimum dissimilarity δ_{ref}^k separating the class center $\tilde{x}_k$ from a point belonging to another class.
3: Remove points having neighbors belonging to foreign classes within a distance of $\varepsilon_1 \delta_{\text{ref}}^k$.
4: Remove isolated points having no neighbor within a distance of $\varepsilon_2 \delta_{\text{ref}}^k$.
5: $\mathcal{I}_k :=$ set of the indices of the remaining points in class k.
6: **Stage 2 (maximin greedy selection):**
7: Initialize l_k with the index of the class center $\tilde{x}_k$.
8: **for** $i \in [\![2; \min(n, |\mathcal{I}_k| - 1)]\!]$ **do**
9: $\quad j := \underset{l \in \mathcal{I}_k \setminus l_k}{\arg\max} \ \underset{m \in l_k}{\min} \ \delta_{lm}$
10: $\quad$ Append j to l_k.
11: **end for**
12: **return** l_k

Algorithm 3 Data augmentation algorithm

Input: training set $\{(x_i, y_i)\}_{1 \leq i \leq N_{\text{train}}}$, dissimilarity matrix δ, per-class number of seeds n, maximum number of pure set testings $p_{\max}$, dimension d of subspaces for pure set testings, number of augmented data N_{DA}.

Output: augmented data $\{(\tilde{x}_i, \tilde{y}_i)\}_{1 \leq i \leq N_{DA}}$ and matrix $\mathcal{S}$ listing pure sets.

  1: **Stage 1 (find pure sets in the training set):**
  2: **for** $k \in [\![1; K]\!]$ **do**
  3:   Apply Algorithm 2 to get the list l_k of n indices of seeds candidates.
  4:   **for** $i \in [\![1; n]\!]$ **do**
  5:     $\mathcal{S}_1 := \{l_k[i]\}$
  6:     neighbors := argsort($\delta[l_k[i], :]$)
  7:     $j := 1$
  8:     setPurity := True
  9:     **while** setPurity **do**
 10:       $\mathcal{S}_{j+1} := \mathcal{S}_j \cup \{\text{neighbors}[j]\}$
 11:       $j := j + 1$
 12:       $p := 1$
 13:       Select a random subset S of d features.
 14:       **while** $\{\pi_S(x_m)\}_{m \in \mathcal{S}_j}$ is not pure in $\{\pi_S(x_m)\}_{1 \leq m \leq N_{\text{train}}}$ and $p \leq p_{\max}$ **do**
 15:         Select a new random subset S of d features.
 16:         p := p+1
 17:       **end while**
 18:       **if** $p = p_{\max} + 1$ **then**
 19:         setPurity := False
 20:       **end if**
 21:     **end while**
 22:     $\mathcal{S}[k, i] := \mathcal{S}_{j-1}$
 23:   **end for**
 24: **end for**
 25: **Stage 2 (generate new data):**
 26: Generate N_{DA} random convex combinations $\{\tilde{x}_i\}_{1 \leq i \leq N_{DA}}$ of the pure sets listed in $\mathcal{S}$. Convex combinations $\tilde{x}_i$ of the pure set described in $\mathcal{S}[k, j]$ are labeled by $\tilde{y}_i = k$.
 27: **return** $\{(\tilde{x}_i, \tilde{y}_i)\}_{1 \leq i \leq N_{DA}}$ and $\mathcal{S}$

5.3. Numerical Results

Linear discriminant analysis (LDA), commonly used for classification tasks, can also be used for supervised dimensionality reduction by projecting the data onto the subspace maximizing the between-class variance, as explained in [31]. For the classification problem presented in this paper, the training data are visualized in the two-dimensional subspace obtained by LDA in Figure 5. Although this subspace is the one that best separates the classes, one can see that the training examples do not form well-separated groups. For this reason, testing the purity of subsets of training data before generating new examples by convex combinations is necessary to reduce the risk of getting incorrectly labeled augmented data.
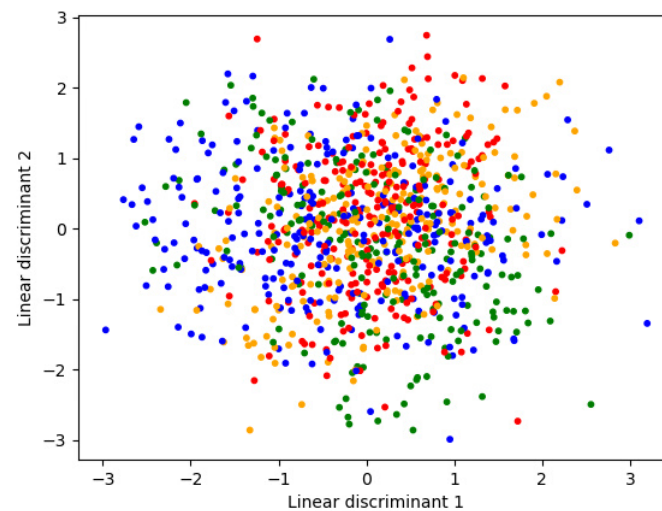
Figure 5. Data visualization in the 2D subspace maximizing the separation between classes (supervised linear dimensionality reduction using linear discriminant analysis (LDA)).

The data augmentation algorithm finds about 60 pure sets per class with an average population of five training examples, using random subspaces of dimension 5 to test the purity. Note that two pure sets are merged only when one is included in the other, as the union of two pure sets is not always pure. The computation time for the data augmentation training phase is 40 minutes. Once pure sets have been found, one can generate as many augmented examples as necessary. Generating 5400 examples to multiply the size of the training set by 10 takes less than a second. Among the augmented data, 400 examples are taken for the evaluation of the data augmentation algorithm. The data labeling procedure involving numerical simulations is applied for these 400 examples in order to estimate the percentage of incorrectly labeled data. It turns out that none of these examples is incorrectly labeled, which validates the algorithm for our problem. The benefits of data augmentation for the classification task are evaluated in the final section.

6. Validation of Our Feature Selection and Data Augmentation Algorithms

6.1. Classification Performances of Various Classifiers

In this section, 14 different classifiers are trained and evaluated on our classification problem. To evaluate whether the features selected by geostatistical mRMR are relevant for classification purposes, each classifier is tested twice: once in combination with the geostatistical mRMR and once with principal component analysis (PCA) with 10 modes. As the random temperature fields derive from a Gaussian random field involving only 10 modes, the database obtained after applying PCA contains all the information of the original data. Each combination of one of the 14 classifiers with PCA or feature selection is trained twice: once on the true training set containing $N_{\text{train}} = 600$ examples, and once on the augmented training set made of 6000 examples.

All the classifiers are trained with Scikit-learn [94], except multilayer perceptrons (MLPs; i.e., fully-connected feedforward deep neural networks) and radial basis function networks (RBFNs) which are trained with PyTorch [98]. We train the RBFNs in a fully supervised manner with Gaussian radial basis functions, which means that the parameters of the radial basis functions are learned by gradient descent like the weights of the fully-connected layers. In addition, we use only one hidden layer for RBFNs, as these artificial neural networks generally have shallow architectures, as explained in [99]. Deeper architectures have been tested for MLPs, with dropout [100], batch normalization [101], ReLU activation functions, and with the number of hidden layers ranging from 2 to 8 with a maximum of 500 neurons per layer. The architectures and the values of some hyperparameters such as the learning rate for Adam optimizer [102], batch size, number of epochs, and dropout rate are calibrated by evaluating the classifier on the validation set after each training

attempt. Scikit-learn's MLP classifier has also been tested; it is called *simple MLP* in this paper, because its architecture is only made of fully-connected layers and does not include dropout nor batch normalization. All the classifiers based on artificial neural networks are trained with Tikhonov regularization (or L^2 regularization of the network's parameters) and early stopping [19]. Logistic regression is trained with elastic net regularization [103] consisting in a weighted average of L^1 and L^2 penalties of the model's parameters. Kernels used for support vector machines (SVMs) are obtained by combining several polynomial kernels with different hyperparameters. Kernel design could be optimized using multiple kernel learning algorithms [104], but we simply build our kernels by evaluating different combinations on the validation set, just as when we look for a good architecture for artificial neural networks. For all of the classifiers using regularization terms in their loss functions, namely, neural networks, SVMs, and logistic regression, hyperparameters such as the regularization strength (weight of the regularization term in the loss function) or the elastic net mixing coefficient are also calibrated using the validation set. For tree-based classifiers, model capacity is controlled by adjusting the maximum depth of the tree and the minimum number of samples at a leaf node. Given the instability of decision trees and their known tendency to overfit, our analysis includes random forests, as well as AdaBoost and gradient boosting with decision trees as base estimators, whose hyperparameters are calibrated on the validation set. Finally, this comparative study also includes the k-nearest neighbors classifier whose number of nearest neighbors must be calibrated, and three generative classifiers that have (almost) no hyperparameter to calibrate, namely, Gaussian naive Bayes, LDA and QDA classifiers.

The classification accuracies on test data are given in Table 2 for classifiers trained with PCA and in Table 3 for those trained with feature selection. Of course, this ranking is specific to the classification problem presented in this paper, no general conclusion can be drawn from this particular numerical application. On this classification problem, when using augmented data in the training phase, the highest test accuracy reached with linear classifiers is 43.5%, obtained with the linear SVM combined with PCA. The fact that k-nearest neighbors classifiers barely exceed 50.0% of accuracy on this problem is related to an observation that was made in [11]: there is no simple correlation between the Euclidean distance and the physics-informed dissimilarity measure used in dictionary-based ROM-nets. MLPs get the best results, reaching 87.0% of accuracy when combined with our data augmentation and feature selection algorithms. Interestingly, quadratic discriminant analysis (QDA) gives excellent results while having no hyperparameter to tune, contrary to the two other families of classifiers obtaining the best results: MLPs and multiple kernel SVMs. This makes QDA the best compromise between accuracy and training complexity for this specific classification task.

Although PCA perfectly describes the input data in this example, the geostatistical mRMR feature selection algorithm enables reaching higher accuracies with some classifiers. Not only does it behave as the original mRMR when selecting features, but it also gives satisfying results when combined with a classifier. Concerning data augmentation, Tables 2 and 3 show that our algorithm significantly improves classification results. The accuracy gain due to data augmentation is 4.98% on average and ranges from −2.5% to 10.5%, increasing the accuracy in 25 cases out of 28.

Table 2. Test accuracies of different classifiers with dimensionality reduction by principal component analysis (PCA), with and without data augmentation (DA).

Classifier	Acc. with DA	Acc. without DA
Stacking (6 MLPs and logistic regression)	89.5%	–
Ensemble averaging (6 MLPs)	89.0%	–
Multilayer perceptron	86.5%	81.5%
Simple multilayer perceptron	85.0%	79.5%
Quadratic discriminant analysis	76.0%	70.0%
Multiple kernel support vector machine	73.0%	68.0%
Radial basis function network	63.5%	62.0%
k-nearest neighbors	51.0%	46.0%
AdaBoost	50.5%	52.5%
Gradient-boosted trees	49.5%	48.0%
Random forest	45.0%	47.5%
Linear support vector machine	43.5%	33.0%
Gaussian naive Bayes	38.5%	31.5%
Penalized logistic regression	38.5%	28.0%
Decision tree	34.0%	36.5%
Linear discriminant analysis	33.5%	29.0%

Table 3. Test accuracies of different classifiers with dimensionality reduction by feature selection (FS), with and without data augmentation (DA).

Classifier	Acc. with DA	Acc. without DA
Stacking (6 MLPs and logistic regression)	90.0%	–
Ensemble averaging (6 MLPs)	89.0%	–
Multilayer perceptron	87.0%	81.0%
Simple multilayer perceptron	84.0%	80.0%
Quadratic discriminant analysis	77.5%	70.5%
Multiple kernel support vector machine	72.5%	66.0%
Random forest	69.0%	63.0%
AdaBoost	68.5%	63.0%
Gradient-boosted trees	68.0%	58.5%
Radial basis function network	62.5%	60.0%
Decision tree	55.5%	43.5%
k-nearest neighbors	50.0%	47.0%
Linear support vector machine	40.5%	34.5%
Gaussian naive Bayes	39.5%	34.5%
Penalized logistic regression	37.0%	29.0%
Linear discriminant analysis	32.5%	29.0%

6.2. How to Further Improve Classification Performances?

Ensemble methods can be used to reduce overfitting and increase the accuracy on test data. In addition, it enables recycling different variants of a classifier that the user has trained for different hyperparameters. Using ensemble averaging with classifiers trained on the augmented dataset with feature selection, we manage to combine six MLPs with different architectures to reach an accuracy of 89.0%. When stacking these MLPs with a ridge logistic regression analyzing the predicted membership probabilities, we get an accuracy of 90.0%. Following the same procedures with six MLPs trained on the PCA representation of the data, ensemble averaging (resp. stacking with ridge logistic regression) gives an accuracy of 89.0% (resp. 89.5%). In addition to ensemble learning methods, one can also use random noise injection to increase noise robustness, as explained in [19].

7. Applicability to Other Problems

The feature selection and data augmentation algorithms introduced in this paper can either be used together like in the previous section, or they can be used separately and combined with other algorithms. For instance, the data augmentation algorithm can be used in conjunction with any dimensionality reduction technique. However, feature selection is recommended when the interpretability of input data is important. In addition, unsupervised dimensionality reduction techniques such as PCA, sparse PCA, kernel PCA, and deep autoencoders extract features that are relevant to reconstruct data after compression, while supervised dimensionality reduction techniques such as mRMR and geostatistical mRMR select features that are suitable for the specific supervised learning task that is considered.

Although this paper is motivated by difficulties encountered when training dictionary-based ROM-nets, our algorithms could be used for other computational methods involving classifiers for model recommendation, such as the LDEIM [4]. The nature of the models in the dictionary, the underlying physics describing the problem, and the way input data are labeled do not matter. It is important to emphasize that our data augmentation algorithm is dedicated to classification problems, whereas our feature selection algorithm could be applied to regression problems too as the definition of the relevance $D(S, Y)$ can be extended to continuous output variables Y. As a consequence, the geostatistical mRMR feature selection algorithm can be applied before training a metamodel that directly predicts a quantity of interest, as long as the inputs are regionalized variables. One could also think of a classifier making qualitative predictions, such as a binary classifier saying whether the system will fail for a given configuration. Our algorithms are applicable to all these types of problems on physics data.

8. Conclusions

Classification algorithms are used in computational physics for automatic model recommendation. Such modeling strategies enable the reduction of the computation time, or the selection between models with different physics when one wants to improve the accuracy of numerical predictions. This article deals with the specificities of the classification problems encountered in computational physics, and more particularly for dictionary-based ROM-nets. These classification problems generally have the three following issues: the lack of training data, their high dimensionality, and the non-applicability of common data augmentation techniques to physics data. To tackle these difficulties, two algorithms are proposed. The first one is a geostatistical variant of the mRMR feature selection algorithm, enabling the identification of a reduced set of relevant but non-redundant features for high-dimensional regionalized variables. The second one is a data augmentation algorithm controlling the risk of generating new examples with wrong labels by finding pure subsets in the training set. The performances and benefits of these algorithms are illustrated on a classification problem for which 14 classifiers are evaluated.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial Neural Network
DA	Data Augmentation
DEIM	Discrete Empirical Interpolation Method
DOE	Design of Experiments
ERM	Empirical Risk Minimization
FS	Feature Selection
GAN	Generative Adversarial Network
HF	High-fidelity
kNN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
LDEIM	Localized Discrete Empirical Interpolation Method
LES	Large Eddy Simulation
MI	Mutual Information
MLP	Multilayer Perceptron
mRMR	Minimum Redundancy Maximum Relevance
NNLS	Non-negative Least Squares
OCT	Optimal Classification Tree
PCA	Principal Component Analysis
POD	Proper Orthogonal Decomposition
QDA	Quadratic Discriminant Analysis
RBFN	Radial Basis Function Network
ROM	Reduced-Order Model
SCRBE	Static-Condensation Reduced-Basis-Element method
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle

References

1. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
2. Baharudin, B.; Lee, L.; Khan, K.; Khan, A. A Review of Machine Learning Algorithms for Text-Documents Classification. *J. Adv. Inf. Technol.* **2010**, *1*. [CrossRef]
3. Kourou, K.; Exarchos, T.; Exarchos, K.; Karamouzis, M.; Fotiadis, D. Machine learning applications in cancer prognosis and prediction. *Comput. Struct. Biotechnol. J.* **2015**, *13*, 8–17. [CrossRef] [PubMed]
4. Peherstorfer, B.; Butnaru, D.; Willcox, K.; Bungartz, H. Localized Discrete Empirical Interpolation Method. *SIAM J. Sci. Comput.* **2014**, *36*. [CrossRef]
5. Nguyen, F.; Barhli, S.; Muñoz, D.; Ryckelynck, D. Computer vision with error estimation for reduced order modeling of macroscopic mechanical tests. *Complexity* **2018**. [CrossRef]
6. Fritzen, F.; Fernández, M.; Larsson, F. On-the-Fly Adaptivity for Nonlinear Twoscale Simulations Using Artificial Neural Networks and Reduced Order Modeling. *Front. Mater.* **2019**, *6*, 75. [CrossRef]
7. Maulik, R.; San, O.; Jacob, J.; Crick, C. Sub-grid scale model classification and blending through deep learning. *J. Fluid Mech.* **2019**, *870*, 784–812. [CrossRef]
8. Kapteyn, M.; Knezevic, D.; Willcox, K. Toward predictive digital twins via component-based reduced-order models and interpretable machine learning. *AIAA Scitech 2020 Forum* **2020**. [CrossRef]
9. Kapteyn, M.; Willcox, K. From Physics-Based Models to Predictive Digital Twins via Interpretable Machine Learning. *arXiv* **2020**, arXiv:2004.11356
10. Maulik, R.; San, O.; Jacob, J. Spatiotemporally dynamic implicit large eddy simulation using machine learning classifiers. *Phys. D Nonlinear Phenom.* **2020**, *406*, 132409. [CrossRef]
11. Daniel, T.; Casenave, F.; Akkari, N.; Ryckelynck, D. Model order reduction assisted by deep neural networks (ROM-net). *Adv. Model. Simul. Eng. Sci.* **2020**, *7*. [CrossRef]
12. Quarteroni, A.; Rozza, G. *Reduced Order Methods for Modeling and Computational Reduction*; Springer: New York, NY, USA, 2013.
13. Keiper, W.; Milde, A.; Volkwein, S. *Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing*; Springer International Publishing: New York, NY, USA, 2018.
14. Smith, G. *Numerical Solution of Partial Differential Equations: Finite Difference Methods*; Clarendon Press: Oxford, UK, 1985.
15. Ern, A.; Guermond, J. *Theory and Practice of Finite Elements*; Springer: New York, NY, USA, 2013.

16. Versteeg, H.; Malalasekera, W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*; Pearson Education Limited: London, UK, 2007.

17. Borutzky, W. *Bond Graph Modelling of Engineering Systems: Theory, Applications and Software Support*; Springer: New York, NY, USA, 2011.

18. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [CrossRef]

19. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.

20. Janecek, A.; Gansterer, W.; Demel, M.; Ecker, G. On the Relationship Between Feature Selection and Classification Accuracy. *J. Mach. Learn. Res.* **2008**, *4*, 90–105.

21. Ding, C.; Peng, H. Minimum redundancy feature selection from microarray gene expression data. In Proceedings of the 2003 IEEE Bioinformatics Conference, Stanford, CA, USA, 11–14 August 2003; pp. 523–528. [CrossRef]

22. Peng, H.; Long, F.; Ding, C. Feature Selection Based On Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1226–1238. [CrossRef]

23. Hua, D. A regularity result for boundary value problems on Lipschitz domains. *Annales de la Faculté des Sciences de Toulouse Mathématiques* **1989**, *5*, 325–333. [CrossRef]

24. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *arXiv* **2014**, arXiv:1406.2661.

25. Akkari, N.; Casenave, F.; Perrin, M.; Ryckelynck, D. Deep Convolutional Generative Adversarial Networks Applied to 2D Incompressible and Unsteady Fluid Flows. In *Intelligent Computing, Proceedings of the 2020 Computing Conference*; Springer: New York, NY, USA, 2020; pp. 264–276. [CrossRef]

26. Chawla, N.; Bowyer, K.; Hall, L.; Kegelmeyer, W. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res. JAIR* **2002**, *16*, 321–357. [CrossRef]

27. He, H.; Bai, Y.; Garcia, E.; Li, S. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328. [CrossRef]

28. Bellman, R. *Adaptive Control Processes*; Princeton University Press: Princeton, NJ, USA, 1961.

29. Vapnik, V. *Statistical Learning Theory*; Wiley-Interscience: New York, NY, USA, 1998.

30. Crammer, K.; Singer, Y. On the Algorithmic Implementation of Multiclass Kernel-Based Vector Machines. *J. Mach. Learn. Res.* **2002**, *2*, 265–292.

31. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed.; Springer: Berlin, Germany, 2009.

32. Berkson, J. Application of the Logistic Function to Bio-Assay. *J. Am. Stat. Assoc.* **1944**, *39*, 357–365.

33. Cox, D. The Regression Analysis of Binary Sequences. *J. R. Stat. Soc. B Methodol.* **1958**, *20*, 215–242. [CrossRef]

34. Cox, D. *Some Procedures Connected with the Logistic Qualitative Response Curve*; John & Wiley: New York, NY, USA, 1966; pp. 55–71.

35. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]

36. Boser, B.; Guyon, I.; Vapnik, V. A Training Algorithm for Optimal Margin Classifier. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*; ACM: New York, NY, USA, 1996; Volume 5. [CrossRef]

37. Mercer, J.; Forsyth, A. XVI. Functions of positive and negative type, and their connection the theory of integral equations. *Philos. Trans. R. Soc. Lond. Contain. Pap. Math. Phys. Character* **1909**, *209*, 415–446. [CrossRef]

38. Ivakhnenko, A.; Lapa, V. *Cybernetic Predicting Devices*; CCM Information Corp.: New York, NY, USA, 1965.

39. Joseph, R.D. Contributions to Perceptron Theory. Ph.D. Thesis, Cornell University, Ithaca, NY, USA, 1961.

40. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw. Off. J. Int. Neural Netw. Soc.* **2015**, *61*, 85–117. [CrossRef] [PubMed]

41. Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. *Classification and Regression Trees*; Routledge: Boca Raton, FL, USA, 1983.

42. Maron, M.E. Automatic Indexing: An Experimental Inquiry. *J. ACM* **1961**, *8*, 404–417. [CrossRef]

43. Zhang, H. The Optimality of Naive Bayes. Available online: https://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf (accessed on 14 February 2021).

44. Cover, T.; Hart, P.. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]

45. Caruana, R.; Niculescu-Mizil, A. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*; ACM: New York, NY, USA, 2006, pp. 161–168. [CrossRef]

46. Kotsiantis, S. Supervised Machine Learning: A Review of Classification Techniques. *Informatica* **2007**, *31*, 249–268.

47. Perez-Ortiz, M.; Jimenez-Fernandez, S.; Gutierrez, P.; Alexandre, E.; Martinez, C.; Salcedo-Sanz, S. A Review of Classification Problems and Algorithms in Renewable Energy Applications. *Energies* **2016**, *9*, 607. [CrossRef]

48. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]

49. Ho, T.K. The Random Subspace Method for Constructing Decision Forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 832–844.

50. Wolpert, D. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259. [CrossRef]

51. Freund, Y.; Schapire, R. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *1*, 119–139. [CrossRef]

52. Hastie, T.; Rosset, S.; Zhu, J.; Zou, H. Multi-Class AdaBoost. Available online: https://web.stanford.edu/~hastie/Papers/samme.pdf (accessed on 14 February 2021).
53. Friedman, J. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2000**, *29*. [CrossRef]
54. Friedman, J. Stochastic Gradient Boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]
55. Mason, L.; Baxter, J.; Bartlett, P.; Frean, M. Boosting Algorithms as Gradient Descent in Function Space. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.6893&rep=rep1&type=pdf (accessed on 14 February 2021).
56. Mason, L.; Baxter, J.; Bartlett, P.; Frean, M. Boosting Algorithms as Gradient Descent. In Proceedings of the Advances in Neural Information Processing Systems 12, Denver, CO, USA, 29 November–4 December 1999; pp. 512–518.
57. Haykin, S. *Neural Networks–A Comprehensive Foundation*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 1999; pp. 351–391.
58. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
59. Meneveau, C.; Sagaut, P. *Large Eddy Simulation for Incompressible Flows: An Introduction*; Springer: Berlin/Heidelberg, Germany, 2006.
60. Feyel, F. Multiscale FE2 elastoviscoplastic analysis of composite structures. *Comput. Mater. Sci.* **1999**, *16*, 344–354. [CrossRef]
61. Fritzen, F.; Kunc, O. Two-stage data-driven homogenization for nonlinear solids using a reduced order model. *Eur. J. Mech. A Solids* **2018**, *69*, 201–220. [CrossRef]
62. Bertsimas, D.; Dunn, J. Optimal classification trees. *Mach. Learn.* **2017**, *106*. [CrossRef]
63. Phuong Huynh, D.B.; Knezevic, D.J.; Patera, A.T. A Static condensation Reduced Basis Element method: Approximation and a posteriori error estimation. *ESAIM M2AN* **2013**, *47*, 213–251. [CrossRef]
64. Eftang, J.; Huynnh, D.; Knezevic, D.; Ronqust, E.; Patera, A. Adaptive Port Reduction in Static Condensation. *IFAC Proc. Vol.* **2012**, *2*, 695–699. [CrossRef]
65. Eftang, J.; Patera, A. Port reduction in parametrized component static condensation: Approximation and a posteriori error estimation. *Int. J. Numer. Methods Eng.* **2013**, *96*. [CrossRef]
66. Smetana, K.; Patera, A. Optimal Local Approximation Spaces for Component-Based Static Condensation Procedures. *SIAM J. Sci. Comput.* **2016**, *38*, A3318. [CrossRef]
67. Chaturantabut, S.; Sorensen, D. Discrete empirical interpolation for nonlinear model reduction. In Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference, Shanghai, China, 15–18 December 2010; pp. 4316–4321.
68. MacQueen, J. Some methods for classification and analysis of multivariate observations. *Comput. Chem.* **1967**, *1*, 281–297.
69. Ryckelynck, D. A priori hyperreduction method: An adaptive approach. *J. Comput. Phys.* **2005**, *202*, 346–366. [CrossRef]
70. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; Chen, T. Recent advances in convolutional neural networks. *Patter Recognit.* **2018**, *77*, 354–377. [CrossRef]
71. Adrian, L.; Adrian, R.; Westerweel, J. *Particle Image Velocimetry*; Cambridge University Press: Cambridge, UK, 2011.
72. Chu, T.; Ranson, W.; Sutton, M. Applications of digital-image-correlation techniques to experimental mechanics. *Exp. Mech.* **1985**, *25*, 232–244. [CrossRef]
73. Mueller, H. Theory of Photoelasticity in Amorphous Solids. *Physics* **1935**, *6*, 179–184. [CrossRef]
74. Fey, U.; Egami, Y. *Transition Detection by Temperature-Sensitive Paint*; Springer: Berlin, Germany, 2007; pp. 537–552.
75. Ye, K.; Lim, L. Schubert varieties and distances between subspaces of different dimensions. *SIAM J. Matrix Anal. Appl.* **2016**, *37*, 1176–1197. [CrossRef]
76. Park, H.; Jun, C. A simple and fast algorithm for k-medoids clustering. *Expert Syst. Appl.* **2009**, *36*, 3336–3341. [CrossRef]
77. Schubert, E.; Rousseeuw, P. Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms. In *Similarity Search and Applications*; Amato, G., Gennaro, C., Oria, V., Radovanović, M., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 171–187.
78. Kaufmann, L.; Rousseeuw, P. Clustering by Means of Medoids. In *Data Analysis Based on the L1-Norm and Related Methods*; Springer: Basel, Switzerland, 1987; pp. 405–416.
79. Chaboche, J. A review of some plasticity and viscoplasticity constitutive theories. *Int. J. Plast.* **2008**, *24*, 1642–1693. [CrossRef]
80. Matthies, H.; Brenner, C.; Bucher, C.; Guedes Soares, C. Uncertainties in probabilistic numerical analysis of structures and solids—Stochastic finite elements. *Struct. Safety* **1997**, *19*, 283–336. [CrossRef]
81. Sudret, B.; Der Kiureghian, A. Stochastic Finite Element Methods and Reliability: A State-of-the-Art Report. Available online: https://ethz.ch/content/dam/ethz/special-interest/baug/ibk/risk-safety-and-uncertainty-dam/publications/reports/SFE-report-Sudret.pdf (accessed on 14 February 2021).
82. Matthies, H.; Keese, A. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 1295–1331. [CrossRef]
83. Khoromskij, B.N.; Litvinenko, A.; Matthies, H.G. Application of Hierarchical Matrices for Computing the Karhunen-Loève Expansion. *Computing* **2009**, *84*, 49–67. [CrossRef]
84. Abrahamsen, P. *A Review of Gaussian Random Fields and Correlation Functions*; Norsk Regnesentral—Norwegian Computing Center: Oslo, Norway, 1997.
85. Meyer, C. *Matrix Analysis and Applied Linear Algebra Book and Solutions Manual*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2000.
86. Cover, T.; Thomas, J. *Elements of Information Theory*; Wiley: Hoboken, NJ, USA, 2012.

87.   Vergara, J.; Estevez, P.   A Review of Feature Selection Methods Based on Mutual Information. *Neural Comput. Appl.* **2014**, *24*.
      [CrossRef]
88.   Barrault, M.; Maday, Y.; Nguyen, N.; Patera, A.   An empirical interpolation method: Application to efficient reduced-basis
      discretization of partial differential equations. *Comptes Rendus Math.* **2004**, *339*, 666–672. [CrossRef]
89.   Farhat, C.; Avery, P.; Chapman, T.; Cortial, J.   Dimensional reduction of nonlinear finite element dynamic models with finite
      rotations and energy-based mesh sampling and weighting for computational efficiency. *Int. J. Numer. Methods Eng.* **2014**,
      98, 625–662. [CrossRef]
90.   Hernandez, J.; Caicedo, M.; Ferrer, A.   Dimensional hyper-reduction of nonlinear finite element models via empirical cubature.
      Comput. Methods Appl. Mech. Eng. **2017**, *313*, 687–722. [CrossRef]
91.   Amsallem, D.; Zahr, M.; Choi, Y.; Farhat, C.   Design optimization using hyper-reduced-order models. *Struct. Multidiscip. Optim.*
      2014, *51*, 919–940. [CrossRef]
92.   Casenave, F.; Akkari, N.; Bordeu, F.; Rey, C.; Ryckelynck, D.   A nonintrusive distributed reduced-order modeling framework for
      nonlinear structural mechanics—Application to elastoviscoplastic computations. *Int. J. Numer. Methods Eng.* **2020**, *121*, 32–53.
      [CrossRef]
93.   Everson, R.; Sirovich, L.   Karhunen-Loeve procedure for gappy data. *JOSA A* **1995**, *12*. [CrossRef]
94.   Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.;
      et al.   Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
95.   Rockafellar, R. *Convex Analysis*; Princeton Landmarks in Mathematics and Physics, Princeton University Press: Princeton, NJ,
      USA, 1970.
96.   Lawson, C.; Hanson, R. *Solving Least Squares Problems*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA,
      1995; [CrossRef]
97.   He, Z.; Xie, L.; Chen, X.; Zhang, Y.; Wang, Y.; Tian, Q.   Data Augmentation Revisited: Rethinking the Distribution Gap between
      Clean and Augmented Data. *arXiv* **2019**, arXiv:1909.09148.
98.   Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An
      Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H.,
      Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019;
      pp. 8026–8037.
99.   Aggarwal, C. *Neural Networks and Deep Learning*; Springer: Cham, Switzerland, 2018. [CrossRef]
100. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.   Dropout: A Simple Way to Prevent Neural Networks
      from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
101. Ioffe, S.; Szegedy, C.   Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceed-
      ings of the 32nd International Conference on Machine Learning, Lille, France, 7–9 July 2015.
102. Kingma, D.; Ba, J.   Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
103. Zou, H.; Hastie, T.   Regularization and Variable Selection via the Elastic Net. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2005**, *67*, 301–320.
      [CrossRef]
104. Gonen, M.; Alpaydin, E.   Multiple Kernel Learning Algorithms. *J. Mach. Learn. Res.* **2011**, *12*, 2211–2268.