Feature Interaction Selection for High-Dimensional Experimental Data

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Di Bo

May 2024

Dedication

I would like to dedicate this paper to my dearest family.

Acknowledgments

I would like to thank my advisor, Dr. Hwangbo. You are an excellent professor. I am amazed by your ideas. I cannot finish my dissertation without your help!

Special thanks to Dr. Stephanie TerMaath for providing the data set and valuable insights. Thank you to Dr. Vinit Sharma and Corey Arndt for providing suggestions.

Thank you for all committee member's valuable feedback and comments.

Thank you to my boyfriend, you always try to help me.

Thank you to Bobby and Mary Kay. You are supportive to international students.

Thank you to Rui Zhou and Tongtong Li. You are the best friends in Knoxville.

Thank you to Dian Zheng and Yiling Xie. You are my mental garden.

Abstract

In a material development process, discerning the effect of material properties and their interactions on material behaviors is critical to achieving the desired functionality of a material. This causal analysis often involves a small experimental dataset arranged in a high dimension and is challenged by the curse of dimensionality. Feature selection can alleviate such a challenge by producing a short list of features that are significant, but identifying significant feature interactions is very challenging. In this proposal, we propose a couple of approaches that can evaluate and determine important interactions, including a randomized subspace-based model (RSM), feature subspace selection (FSS), and XGBoost. The core idea of RSM and FSS is to randomly generate low-dimensional subsets of a feature set and use such a subset, referred to as a subspace, as a basic modeling and significance evaluation unit. The significance of each potential subspace is evaluated, and a significant subspace is integrated into a base model to form an ensemble predicting the response. The experimental results also provide valuable insights into the damage process of the hybrid material structure. We upgrade the subspace generation and exploration by using XGBoost. The subspace is evaluated by its contribution to the response. When applied to the analysis of a composite/metal hybrid structure exhibiting complex progressive damage failure under loading, our methods show advantages in response modeling and feature selection compared to other machine learning-based alternatives.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 General Introduction

High-dimensional data analysis has gained more attention with the recent breakthrough in computing, and it has a great demand in studying complex systems [76, 64]. The high dimensionality of a dataset can be beneficial in terms of the fluency of information, but the analysis of a high-dimensional dataset is not an easy task due to the curse of dimensionality [1]. As the dimensionality of a dataset increases, the volume of the feature space increases dramatically, and the data therein become dissimilar and sparse in many ways, which restricts the capability of general data modeling strategies [53]. A common solution to this problem is to evaluate a data-driven model in a meaningful low-dimensional space by identifying a subset of features that can represent the entire data with a minimal loss of information, e.g., through dimensionality reduction [72].

In general, dimensionality reduction can be classified by two types of methods, feature extraction, and feature selection. Feature extraction projects the feature space into a low-dimensional intrinsic space and extracts features spanning the intrinsic space [34]. The projection creates new features that are different from the original ones with physical meanings, so the usage of feature extraction is limited in an experimental study where original physical features need to be controlled. Feature selection, on the other hand, examines the significance of existing physical features and determines which feature to keep based on the significance evaluation [72]. This produces a smaller subset of features, pinpointing which design parameters are among the existing ones to test in subsequent experiments. However, common feature selection approaches are not capable of identifying significant feature interactions, often leading to the exclusion of key parameters that are insignificant on their own but essential in conjunction with others. It is known that discovering important interactions is very challenging, and only a few studies are available along this direction [1].

In this proposal, we propose novel methods that can identify significant feature interactions based on random sampling and ensemble modeling. We model a regression problem as an ensemble of multiple base learners where each base learner is defined over a lower-dimensional input space only; we will refer to this reduced-dimensional physical space as a subspace. The proposed methods are similar to general feature selection approaches

as they select subsets of the original features to achieve dimensionality reduction. However, different from others in the literature relying on a single subset of features, the proposed methods leverage multiple distinct subsets and accordingly multiple models to evaluate a function. To balance model accuracy and computational complexity, we apply a random generation of subspaces and selectively choose important subspaces. By construction, an important subspace informs that not only the features therein but also potential interactions between them are significant. We prescribe the subspaces to be low-dimensional, considering that a high-order interaction exceeding a certain order is rarely significant in practice. By keeping all subspaces at a low dimension, functional evaluation is always performed at a low-dimensional space without the risk of suffering from the curse of dimensionality.

We first develop the randomized subspace-based model (RSM) to establish the basis for important interaction selection. RSM imposes a fixed dimension on all subspaces to simplify the random exploration of subsets of the feature space. The quality of prediction is used to determine whether a subspace is significant, or equivalently, whether to include a particular subspace in the ensemble model. To improve the overall structure and computations, we also develop feature subspace selection (FSS). FSS allows subspaces with varying dimensions by applying backward feature elimination to a chosen subspace. To enhance the subspace selection process, we leverage double selection criteria involving statistical hypothesis tests as well as prediction quality-based selection. The termination of the algorithm is adapted accordingly to manage the volume of the computations that were supposed to increase as a consequence of the additional benefits. Finally, considering the effectiveness of tree-based models in interaction modeling, we apply XGBoost [14] to replace a fully randomized search by a directional random search, which also alleviates computational demands significantly. In a tree-based model, a tree is created by multiple features, naturally forming a subspace. Once a set of significant subspaces are available from the implementation of XGBoost, this information is used to create a partially connected layer in a deep neural network (DNN) structure to take advantage of DNN's prediction capability and at the same time facilitate the learning process through the additional information.

Design and analysis of layered hybrid structures is a great challenge in terms of the multiple choices of materials and configurations, including the large number and ranges

of input features. This vast parameter space is hard to explore via physical testing and computation due to the analysis time required. RSM and FSS are applied to overcome this drawback and provide an accurate and efficient analysis to computationally characterize the parameter space and utilize limited physical testing to define parameters. The proposed reduced order model can rapidly explore the parameter space to customize and optimize layered designs. The specific problem of metal and composite layered structures was chosen due to its complexity and potential interactions among input parameters.

Chapter 2

Randomized Subspace-based Model (RSM)

2.1 Introduction

With the recent breakthrough in computing, a real-world data analysis tends to involve voluminous high-dimensional data sets for modeling various systems with great complexity [46, 76]. An increased dimensionality can provide more information about an underlying system, but analyzing a high-dimensional data set is not an easy task due to the curse of dimensionality [37]. That is, for a fixed sample size, data tend to be sparser in space, so signals are much weaker but noises have far greater impacts, leading to improper analysis outcomes. A common solution to this problem is to evaluate a data-driven model in a meaningful low-dimensional space by identifying a subset of features that can represent the entire data with a minimal loss of information, e.g., through dimensionality reduction or variable selection [37].

Dimensionality reduction extracts inherent features from high-dimensional data and relocates the data in a low-dimensional space constructed by the extracted features, alleviating the curse of dimensionality [63]. However, existing dimensionality reduction techniques suffers from the lack of interpretability as the extracted features are "artificial" having obscure connections to the original "physical" variables [63]. In engineering applications, identifying important physical variables is crucial for reducing the number of experimental runs required to optimize a system of interest. In this context, the usage of dimensionality reduction is limited. As an alternative, variable selection, also referred to as subset selection or feature selection, can distinguish important physical variables by selecting a subset of the original variables that are significant [66]. Finding an exact optimal subset is computationally intractable, so variable selection typically involves a heuristic search [20]. A simple heuristic such as a greedy approach cannot ensure the quality of the final model, and a complex heuristic such as genetic algorithm is computationally expensive. In addition, while applying a heuristic search, important interactions between physical variables can easily be missed.

This paper develops a randomized subspace-based modeling for the purpose of alleviating the curse of dimensionality and providing valuable insight about an underlying system for system optimization. To this end, we model a regression problem as an ensemble of

multiple base learners where each base learner is defined over a lower-dimensional input space only; we will refer to this reduced-dimensional physical space as a subspace. The subspace-based modeling is similar to general variable selection approaches, as it selects subsets of the original variables to achieve dimensionality reduction. However, it differs from them as it leverages multiple distinct subsets and accordingly multiple models to evaluate a function, extending the idea of stepwise regression and stacking. To balance between model accuracy and computational complexity, we randomly generate subspaces and selectively choose important subspaces. By construction, an important subspace informs that not only the variables therein but also the potential interactions between them are significant. We prescribe the subspaces to be a fixed (low) dimension considering that a high-order interaction exceeding a certain order is rarely significant in practice. By keeping all subspaces at a low dimension, functional evaluation is always performed at a low-dimensional space without a risk of suffering from the curse of dimensionality.

The potential of this method to significantly impact our characterization and understanding of advanced engineering systems is demonstrated using the challenging problem of identifying the most influential material properties on damage tolerance for a layered hybrid structure and formulating a reduced order model based on these most sensitive parameters. This example was chosen due to the high dimensional parameter space and wide range of parameter values. Additionally, the high fidelity model used to predict damage tolerance requires a prohibitive amount of computational time to characterize just a small number of parameters in a narrow subspace of the parameter value ranges. The approach presented herein enables a novel method to rapidly reduce and characterize this vast and complex parameter space to solve a challenging physics-based structural mechanics problem.

The remainder of this article is organized as follows. Section 2.2 reviews relevant studies on dimensionality reduction and important variable selection. Section 2.3 presents the proposed subspace-based method by describing model structure, random generation and selective extraction of subspaces, and overall learning process. Section 2.4 illustrates feature selection and prediction capability of the subspace-based modeling by using popular toy problems and a publicly available data set. Section 2.5 demonstrates the benefits of the proposed method in comparison with others for modeling the damage tolerance of a hybrid

material and discusses its potential usages for structural design, analysis, and optimization. Section 2.6 concludes the paper and discusses future work.

2.2 Literature Review

In the past decades, dimensionality reduction has been common in many applications involving a large number of variables, including digital photographs, speech recognition, financial marketing and bioinformatics [63]. Dimensionality reduction techniques transform high-dimensional data into a meaningful reduced-dimensional representation, ideally close to its intrinsic dimensions [63]. The intrinsic dimensions of data are the minimum features needed to account for the observed properties of the data [21].

Traditional dimensionality reduction techniques include principal component analysis (PCA), independent component analysis (ICA), and multidimensional scaling (MDS). PCA is one of the most popular linear reduction techniques and dates back to Karl Pearson in 1901 [43]. This technique tries to find orthogonal directions that account for as much variance of data as possible. Due to its attractive advantages of minimal information loss and generation of uncorrelated dimensions, PCA is still popular for use in a broad range of application areas [79]. For example, Li et al. [35] applied PCA to evaluate energy security of multiple East Asian countries with respect to vulnerability, efficiency, and sustainability. Salo et al. [49] proposed a network anomaly detection method based on a reduced dimensionality achieved by combining information gain and PCA. On the other hand, ICA tries to extract independent pieces of information from high-dimensional data, mainly for the purpose of source blind separation that has been popular in signal processing [27]. ICA can be useful in other areas of study; for example, Sompairac et al. [55] discussed the benefits of ICA to unravel the complexity of cancer biology, particularly in analyzing different types of omics data sets. Different from PCA and ICA, MDS is a nonlinear dimensionality reduction technique. It provides a useful graphical representation of data based on the similarity information of individual data points. MDS is a common technique for analyzing network-structured data as presented in Saeed et al. [48] that applied MDS to wireless networks localization.

Over the recent decades, many nonlinear or nonparametric dimensionality reduction techniques have been proposed [33, 50]. Kernel PCA is a reformulation of traditional linear PCA constructing feature space through a kernel function [51]. Choi et al. [16] found PCA was inefficient and problematic for modeling a nonlinear system but kernel PCA effectively captured nonlinearity while achieving dimensionality reduction. Xu et al. [68] proposed a defect prediction framework that combined kernel PCA and weighted extreme learning machine to extract representative data features and learn an effective defect prediction model. Tenenbaum et al. [58] proposed an isometric feature mapping (Isomap) technique that was based on classical MDS but sought to retain the intrinsic geometry of data sets as captured in the geodesic manifold distances. It achieves the goal of nonlinear dimensionality reduction by using easily measured local metric information to learn the underlying global geometry of a data set. Hinton and Salakhutdinov [29] suggested using deep autoencoder networks while transforming high-dimensional data into low-dimensional codes by training a multilayer neural network with a small central layer. They presented that the deep autoencoder networks outperformed PCA with a proper selection of initial weights. Belkina et al. [8] introduced t-distributed stochastic neighbor embedding (t-SNE), and Gisbrecht et al. [22] presented kernel t-SNE as an extension of t-SNE to a parametric framework, which enabled explicit out-of-sample extensions. They demonstrated that kernel t-SNE yielded satisfactory results for large data sets. McInnes et al. [40] proposed uniform manifold approximation and projection (UMAP) constructed by a theoretical framework based on Riemannian geometry and algebraic topology. Compared to t-SNE, UMAP arguably preserves more of the global structure. Since UMAP does not have computational restrictions on embedding dimension, it can be easily used for general dimensionality reduction.

Even with the theoretical and methodological advance of the dimensionality reduction techniques, they cannot pinpoint which variables are important among the existing variables; instead, they extract inherent features that are artificial. Determining important physical variables is a critical task in many experimental studies, including those for structural mechanics [59], environmental science, and bioinformatics [66]. This is because the knowledge of important variables can suggest which variables to test and optimize for subsequent experiments to reduce the number of experimental trials and hence expedite the overall

experimental procedure. In this context, variable selection (or feature selection) that extracts a subset of existing variables can be a good alternative to general dimensionality reduction since it is capable of maintaining the original variable structure while reducing the dimensionality.

Feature selection methods are typically classified into three groups of filter methods, wrapper methods, and embedded methods [12]. Filter methods evaluate each variable based on some ranking criteria, such as Pearson correlation coefficient [7] or mutual information [62]. Wrapper methods, evaluating different subsets of variables by using a learning algorithm, generally provide better performance compared to filter methods [69]. To explore potential subsets of variables, an exhaustive search requires considering 2^p different feature combinations when there are p variables, which is impractical. Instead, sequential forward selection [67] and sequential backward selection [39] algorithms that apply greedy search have been used broadly. More recently, sequential floating forward selection [45] and adaptive sequential floating forward selection [54] have been proposed to alleviate poor performance of the greedy search. However, these sequential selection methods still suffer from a nesting problem caused by the nature of the greedy search, so the selection outcome is generally far from the optimum. To overcome the nesting problem, heuristic search methods, e.g., one based on genetic algorithm [2], have been proposed. These wrapper methods, however, require an extensive amount of computation and are prone to overfitting [12]. Embedded methods aim to alleviate the computational cost of wrapper methods by integrating feature selection and model learning into a single process [69]. In other words, some ranking criteria, including max-relevancy min-redundancy [44] and the weights of a classifier [41], are used for the initial feature reduction, and then wrapper methods are applied for the final feature selection.

These days, machine learning methods, such as random forest (RF) [32, 13, 75] and neural network (NN) [42] have been used to measure variable importance and rank variables based on model's prediction accuracy, which can be easily extended for feature selection [38, 17]. Gregorutti et al. [24] explored the usage of RF in the presence of correlated predictors. Their results motivated the use of the recursive feature elimination algorithm that uses the permutation importance measure as a ranking criterion. Chen et al. [13]

compared results from three popular data sets (bank marketing, car evaluation database, human activity recognition using smartphones) with and without feature selection based on multiple methods, including RF and recursive feature elimination. The experimental results demonstrated that RF achieved the best performance in all experimental groups. Zhou and Hooker [75] considered split-improvement, a popular feature importance measure for tree-based models, for an RF model to determine variable importance. As split-improvement suffers from bias towards continuous features, they resolved this issue and demonstrated the effectiveness of their solution approach both theoretically and empirically. On the other hand, Olden et al. [42] compared nine variants of artificial neural network (ANN) for quantifying variable importance from simulated data. For this comparison, the true importance of variables was known and used for verification. Their results showed the connection weight approach provided the best accuracy and it was the only method that correctly identified the rank of variable importance. Liu [37] presented a deep Bayesian rectified linear unit (ReLU) network for high-dimensional regression and variable selection. The result showed their method outperformed existing classic and other NN-based variable selection methods.

Although these variable selection methods provide an efficient tool to extract important variables, they primarily focus on the importance of individual variables, with little consideration for the importance of feature interactions. In a general experimental problem, however, knowledge about the significance of feature interactions is a critical factor for designing experiments. Instead of merely finding important individual variables, our approach aims to identify critical subspaces, each of which presents which variable combination as a whole (including interactions) is important. For the exploration of potentially important subspaces, a randomized search is employed to improve model accuracy and computational efficiency relative to a greedy search and a metaheuristic search, respectively. Furthermore, our method leverages multiple subsets of variables (subspaces) for model construction instead of relying on a single subset as the existing methods do, which can provide a more flexible model. The details of the proposed method will be discussed in the subsequent sections.

2.3 Subspace-Based Modeling and Critical Subspaces

In this section, we develop subspace-based modeling for solving a general supervised learning problem (in particular, a regression problem) and identifying critical variables and interactions. The data set of interest contains data pairs of $\{\mathbf{x}_i, y_i\}_{i=1}^n$ where $\mathbf{x}_i$ is a p-dimensional input vector and y_i is the corresponding response. For subspace-based modeling, we form a subspace, a space spanned by a subvector of input $\mathbf{x}$, determine the significance of a subspace for modeling response, and use such a critical subspace as a basic unit for model building. A critical subspace implies that the variables forming the space and their interactions are important altogether, so it naturally pinpoints important variables and interactions. In what follows, we describe the proposed model structure, a randomized search for subspace generation, a significance evaluation of a subspace, and the overall model learning process.

2.3.1 Subspace-based Model

Considering a multi-inputs, single output regression problem, we estimate a function $f : \mathbb{R}^p \to \mathbb{R}$ that relates p-dimensional input $\mathbf{x}$ and output y as $y = f(\mathbf{x}) + \varepsilon$ where ε is an additive noise. We model the unknown function f as an additive mixture of subfunctions $g_j : \mathbb{R}^k \to \mathbb{R}$ for $j = 1, \ldots, J$ defined in a lower dimensional space of dimension $k \ll p$:

$$y = f(\mathbf{x}) + \varepsilon \approx g_1(\mathbf{z}_1) + g_2(\mathbf{z}_2) + ... + g_J(\mathbf{z}_J) + \varepsilon, \tag{2.1}$$

where $\mathbf{z}_j$ is the jth subvector of $\mathbf{x}$ of size k. Each $\mathbf{z}_j$ for $j = 1, \ldots, J$ takes different components of $\mathbf{x}$. For example, suppose $k = 3$ and $\mathbf{x} = \Big[x_1, x_2, \ldots, x_p\Big]^T$ where $p > 20$. Then, we may have $\mathbf{z}_1 = \Big[x_3, x_8, x_{12}\Big]^T$ and $\mathbf{z}_2 = \Big[x_7, x_{11}, x_{20}\Big]^T$. We define a reduced-dimensional space spanned by each subvector $\mathbf{z}_j$ as a subspace, and g_j estimates the response y within a subspace formed by $\mathbf{z}_j$. As such, in Eq. (2.1), we model the function f as a mixture of subspace-based models. This ensemble extends the idea of stepwise regression [31] and stacking [70] as different subsets of the features are used to build distinct models (g_j) predicting the response.

The advantage of the subspace-based modeling is obvious. By evaluating y in low-dimensional subspaces, there is no risk of suffering from the curse of dimensionality, which is not true when evaluating y in a single high-dimensional space. Different from general dimensionality reduction methods, the subspace-based modeling does not require extracting artificial dimensions, but the reduction of dimensionality is achieved by forming low-dimensional physical spaces. One major shortcoming of the subspace-based modeling is its incapability of modeling interactions of an order higher than k. However, in many real-world problems, a high-order interaction is almost negligible in modeling response. From a preliminary study, we found that $k = 3$, i.e., modeling up to 3-factor interactions, produced decent predictions; see Appendix B for more details.

The key to the success in the subspace-based modeling lies in how to form the subspaces, i.e., how to generate the subvectors of $\mathbf{z}_j$ for $j = 1, \ldots, J$. For an exhaustive search, if we assume $p = 41$ and $k = 3$, the number of all subspace candidates is $\binom{41}{3} = 10660$. Evaluating models for this many subspaces and determining whether to include each of them requires a considerable amount of computation (still, much smaller compared to 2^{41} for an exhaustive search in the absence of the proposed model structure in Eq. (2.1)). This requires an efficient strategy for subspace exploration, and in the next section, we discuss how to generate subspaces and how to determine critical subspaces.

2.3.2 Subspace Generation and Extraction

To explore a broad area covering potential subspace configurations, we generate subspaces randomly. In particular, at each draw, we randomly choose k variables out of p available and evaluate whether to include this randomly generated subspace into the model or not. The sampling process is without replacement, but we allow duplicated selection of a single variable at multiple draws. In other words, an input variable x_1 could be a part of subspace $\mathbf{z}_1$, and the variable can be chosen again at later draws for $\mathbf{z}_j$ for $j > 1$. This is to ensure that multiple interactions associated with a key variable are not lost.

We determine the significance of a randomly generated subspace by evaluating the percentage reduction in prediction error measured by root-mean-square error (RMSE). To evaluate out-of-sample error and avoid possible overfitting, we apply 5-fold cross validation

(CV) for error estimation. This 5-fold CV is applied to the data set assigned for model learning (excluding testing portion), and the data set is split into 5 small subsets of the data. Each subset serves as a validation data set whereas the remaining four subsets are collectively used to train a model. In this way, we generate 5 different train/validation data sets, as shown in Fig. 2.1.

To test the significance, a temporary subspace-based model (g_t) for a randomly generated subspace ($\mathbf{z}_t$) will be added to the model, and the prediction error of this extended model will be evaluated in terms of the out-of-sample RMSE. The temporary subspace-based model, g_t where t denotes the current iteration of the subspace generation, is learned for each of the five different train data sets, and the prediction error of the extended model is calculated by using the corresponding five validation data sets. This will produce $RMSE_{t,q}$ for $q = 1, \ldots, 5$ for each train/validation split at iteration t. To be specific, the RMSE is calculated as

$$RMSE_{t,q} = \sqrt{\frac{1}{n_{v_q}} \sum_{\{\mathbf{x}_i, y_i\} \in \mathcal{V}_q} (y_i - \tilde{f}_t(\mathbf{x}_i))^2}, \tag{2.2}$$

where $\mathcal{V}_q$ is the validation data set for the qth train/validation split and n_{v_q} is the number of observations in the validation data set. $\tilde{f}_t(\mathbf{x}_i) = \sum_{j \in \mathcal{J}^*} g_j(\mathbf{z}_j) + g_t(\mathbf{z}_t)$ where $\mathcal{J}^*$ is the set of indices for the selected subspaces by the previous iteration, $t-1$. The CV score is then the average of the five prediction errors, i.e., $CV_t = \sum_{q=1}^{5} RMSE_{t,q}/5$.

We use the percentage reduction of the prediction error for the selection of a critical subspace and also for the termination of the iterative search for new subspaces. The percentage error reduction is defined as $\Delta e = (CV^* - CV_t)/CV^*$ where CV^* is the minimum (best) CV score obtained until the previous iteration. The initial value of CV^* is calculated from a model that only includes a constant term (the average of the five sample means of y_i's in each training data set). We consider a subspace significant if $\Delta e > \eta$ where η is a preset selection threshold. This means that a subspace that reduces the prediction error at least by a certain percentage will be included in the model; otherwise, this subspace is disregarded. The search for a new subspace continues until the percentage reduction Δe becomes less than a termination threshold, τ. If none of the selection criterion and the termination criterion are

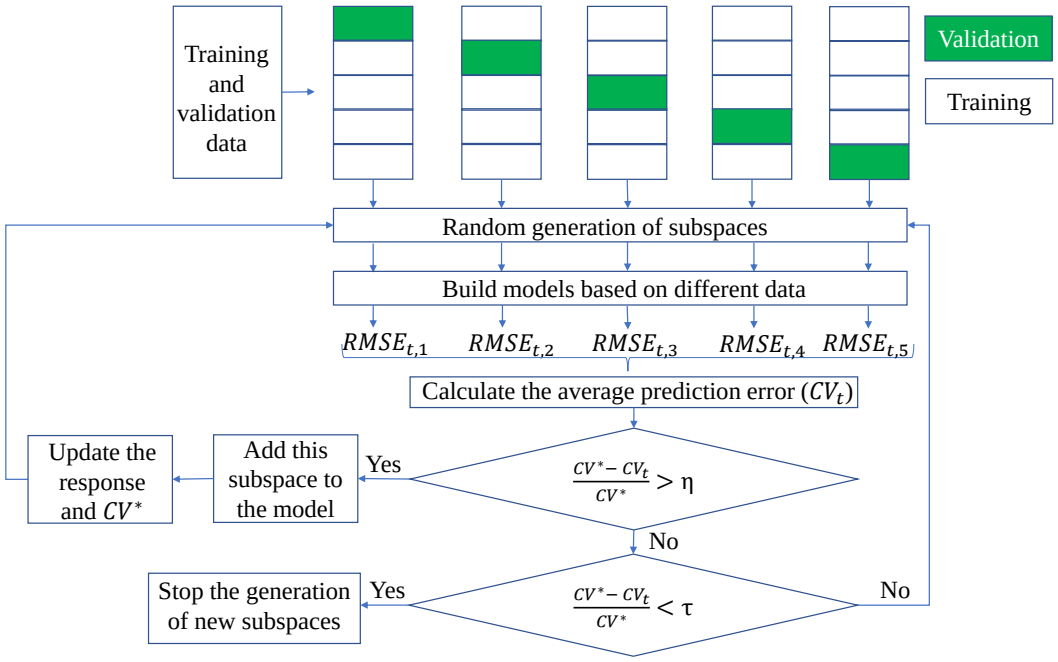

Figure 2.1: A flow chart describing the process of subspace generation and extraction

met, the iterative search continues for the next possible critical subspace. Fig. 2.1 illustrates the overall process of the subspace generation and extraction.

2.3.3 Subspace Model Learning and Hyperparameter Selection

This section describes how to estimate each subspace-based model g_t. For this estimation, we use a randomly generated subvector, $\mathbf{z}_t$ as predictors and the current residual calculated before the iteration t as a response to estimate, i.e., using $\tilde{y}_{t,i} = y_i - \sum_{j \in \mathcal{J}^*} g_j(\mathbf{z}_{j,i})$ for $i = 1, \ldots, n$ for response. In other words, we consider a regression problem written as

$$\tilde{y}_{t,i} = g_t(\mathbf{z}_{t,i}) + \tilde{\varepsilon}_{t,i}, \tag{2.3}$$

where $\mathbf{z}_{t,i}$ is a subvector of $\mathbf{x}_i$, and $\tilde{\varepsilon}_{t,i}$ is a modified noise for $i = 1, \ldots, n$.

To learn the function g_t, we use support vector machine (SVM) as a base learner. For a regression analysis, it is also known as support vector regression (SVR). To train a model based on SVR, we use "ϵ-insensitive" loss function characterized by a flexible tube of ϵ radius. Under this loss function, penalty is assigned only to the points outside the tube and is proportional to the distance from the tube, but no penalty is applied to the points within the tube [5]; see Eq. (2.5).

For SVR, the function g_t can be expressed as a linear combination of basis functions, h_m for $m = 1, \ldots, M$, as

$$g_t(\mathbf{z}_t) = \sum_{m=1}^{M} \beta_m h_m(\mathbf{z}_t) + \beta_0, \tag{2.4}$$

where β_m for $m = 1, \ldots, M$ is a coefficient of each basis function and β_0 is a constant. To estimate g_t, we minimize

$$\begin{aligned} H(\boldsymbol{\beta}, \beta_0) &= \sum_{i=1}^{n} V_\epsilon(\tilde{y}_{t,i} - \hat{g}_t(\mathbf{z}_{t,i})) + \frac{\lambda}{2} \sum_{m=1}^{M} \beta_m^2 \\ \text{where } V_\epsilon(r) &= \begin{cases} 0, & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise,} \end{cases} \end{aligned} \tag{2.5}$$

where $\boldsymbol{\beta} = \Big[\beta_1, \ldots, \beta_M\Big]^T$ is a coefficient vector, and λ is a regularization parameter. The minimizer of Eq. (2.5) provides the estimate of g_t in the form of

$$\hat{g}_t(\mathbf{z}) = \sum_{i=1}^{n} \alpha_i K_t(\mathbf{z}, \mathbf{z}_{t,i}), \tag{2.6}$$

where α_i for $i = 1, \ldots, n$ is a Lagrangian dual variable for the Lagrangian primal shown in Eq. (2.5), and $K_t(\cdot, \cdot)$ is a kernel function that represents the inner product of the unknown basis functions, h_m for $m = 1, \ldots, M$ [27].

To fully specify the estimate of g_t, some hyperparameters need to be determined. This includes the regularization parameter, λ (related to the cost parameter in a typical SVM), the radius of the tube, ϵ, for the loss function, and some kernel related parameters. For this hyperparameter learning, we apply grid search based on generalized cross validation (GCV) criterion. The proposed method already employs 5-fold CV for the selection of critical subspaces and the termination of the iterative search, so another layer of out-of-sample prediction will complicate the data structure, and the training process may suffer from the lack of usable data points. While measuring in-sample error based on training data only, GCV provides a convenient approximation to the leave-one-out CV [27], so it is a proper criterion for the hyperparameter learning of the proposed method. In general, GCV is calculated as

$$GCV(\hat{f}) = \frac{1}{n} \sum_{i=1}^{n} \Big[\frac{y_i - \hat{f}(\mathbf{x}_i)}{1 - \text{trace}(\mathbf{S})/n)}\Big]^2, \tag{2.7}$$

where $\mathbf{S}$ is an $n \times n$ hat matrix that performs a linear projection of $\mathbf{y} = \Big[y_1, \ldots, y_n\Big]^T$ to achieve the estimate of $\mathbf{y}$, i.e., $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$. For the proposed subspace-based model in Eq. (2.1), deriving this expression of linear projection is not straightforward. Theorem 2.1 shows a good approximation of $\mathbf{S}$ that can be derived under the assumption of a squared loss function.

Theorem 2.1. *Assuming a squared loss function, i.e., $V_\epsilon(r) = r^2$, the estimate of the response $\mathbf{y}$ can be expressed as $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y} = (\sum_{j=1}^{J} \mathbf{S}_j)\mathbf{y}$ where $\mathbf{S}_j = \mathbf{S}'_j(\mathbf{I} - \sum_{l=0}^{j-1} \mathbf{S}_l)$, $\mathbf{S}'_j = (\mathbf{K}_j + \lambda\mathbf{I})^{-1}\mathbf{K}_j$, $\mathbf{S}_0 = \mathbf{0}$, $\mathbf{I}$ is an $n \times n$ identity matrix, and $\mathbf{K}_j$ is an $n \times n$ kernel matrix with $\{\mathbf{K}_j\}_{i,i'} = K_j(\mathbf{z}_{j,i}, \mathbf{z}_{j,i'})$ for $i, i' = 1, \ldots, n$.*

Proof. See Appendix A. □

As implied in Theorem 2.1, we do not allow distinct hyperparameters for each g_j estimation as the GCV calculation is based on the full model. Instead, we assume the hyperparameters to be the same across all g_j for $j = 1, \ldots, J$. We apply a grid search for the hyperparameter learning, which is to keep hyperparameters at certain levels, add critical subspaces based on this hyperparameter setting, and evaluate the GCV value for the resulting full model (after the termination of subspace search). The set of hyperparameters and the corresponding subspace-based model that minimize the GCV criterion in Eq. (2.7) will be chosen as an optimal model.

Because the GCV calculation based on the result of Theorem 2.1 is complicated due to the recursive format, we also consider a simpler version of the trace calculation. For approximation, we replace $\mathbf{S}_j$ by $\mathbf{S}'_j$. The following describes two alternatives we propose for the GCV calculation:

A1. $\text{trace}(\mathbf{S}) = \sum_{j=1}^{J} \text{trace}(\mathbf{S}_j)$ based on Theorem 2.1.

A2. $\text{trace}(\mathbf{S}) = \sum_{j=1}^{J} \text{trace}(\mathbf{S}'_j)$ for simplification.

The performance of the two alternatives will be compared and discussed in Section 2.5.

2.3.4 Overall Algorithm

For a clear illustration of the entire learning process, Alg. 1 shows the step-by-step procedures of the proposed subspace-based modeling. To build a model, we use a data set that is dedicated to model learning; if testing is needed (as in Section 2.5), we split the original data set into two distinct data sets, one for model learning and another for testing. From the model learning data set, we apply 5-fold CV to split training data and validation data where validation data are required to determine the significance of subspaces and the convergence of the algorithm. After the data split, we produce a model that includes a single constant term by using training data only and calculate the prediction error from the validation sets to initialize CV^*. Each hyperparameter setting will be evaluated once a full model is established, i.e., once all critical subspaces are determined and added into the model. For

Algorithm 1: Learning process of the randomized subspace modeling

$\mathcal{D}$: data set available for model training and validation; k: predetermined dimension of subspaces; η: selection threshold; τ: termination threshold Trained model with optimal hyperparameters Apply 5-fold CV to split given data $\mathcal{D}$ into training ($\mathcal{T}_q$) and validation data ($\mathcal{V}_q$) for $q = 1, \ldots, 5$;
Build a constant model, i.e,. $\hat{y} = \bar{y}$, by using data in $\mathcal{T}_q$ and use this model to predict responses in the corresponding $\mathcal{V}_q$ for $q = 1, \ldots, 5$;
Initialize CV^* as the prediction error of the constant model (the average of 5 RMSEs obtained from $\mathcal{V}_q$ for $q = 1, \ldots, 5$);
Create a grid for assessing different sets of hyperparameters;
for *each level combination of hyperparameters* **do**
    $t \leftarrow 0$; $\mathcal{J}^* \leftarrow \varnothing$; $\mathcal{Z} \leftarrow \varnothing$;
    repeat
        $t \leftarrow t + 1$;
        Randomly draw a k-dimensional subspace, i.e., randomly sample k integers from $\{1, \ldots, p\}$ and store the resulting k-tuple of indices in s;
        By using the randomly drawn subspace, fit a model in Eq. (2.3) for each $\mathcal{T}_q$ and accordingly, predict $\tilde{y}_{t,i}$ for each $\mathcal{V}_q$ for $q = 1, \ldots, 5$;
        Compute $\tilde{f}_t(\mathbf{x}_i)$ and calculate $RMSE_{t,q}$ and CV_t;
        if $\Delta e > \eta$ **then**
            $CV^* \leftarrow CV_t$;
            $\mathcal{J}^* \leftarrow \mathcal{J}^* \cup \{t\}$;
            $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{s\}$;
        end
    until $\Delta e < \tau$;
    Use the entire model learning data ($\mathcal{D}$) to build a full model as in Eq. (2.1) by using the selected set of subspaces, $\mathbf{z}_j$ for $j \in \mathcal{J}^*$ where $\mathbf{z}_j$'s are constructed based on the k-tuples in $\mathcal{Z}$;
    Calculate GCV (either A1 or A2) for the current hyperparameter setting by using the full model;
end
Determine the optimal hyperparameters that minimize GCV;
Return the full model with the minimum GCV (this includes the information of $\mathcal{J}^*$ and $\mathcal{Z}$);

each fixed level set of hyperparameters, we generate a subspace randomly and build an SVR model to estimate g_t. The quality of a subspace is evaluated via a 5-fold CV error of CV_t and its percentage reduction Δe. If a subspace passes the selection criterion, it will be added into the model. This process of generating and evaluating a random subspace will continue until the termination criterion is met. Once the algorithm terminates, the GCV value will be computed according to Eq. (2.7). After completing all iterations for the grid search, we find the optimal hyperparameter values and apply this optimal setting to the entire model learning data set (training and validation) to build a final model. This final model will be evaluated by a separate test set for comparison with other methods in Section 2.5. Please note, this entire procedure can be easily adapted to a classification problem by choosing proper loss functions and error measures that are suitable for classification.

2.4 Comparative Experiments

Prior to applying our method to the hybrid structure data set, in this section, we use three other data sets to evaluate the performance of the proposed method in terms of prediction and feature selection capability. For the first two data sets, the true importance of features and interactions is known, so they are used to assess the feature selection capability. The other data set involves a large-scale feature space where feature selection is essential for predicting response; we investigate prediction accuracy and robustness through this data set. To demonstrate the effectiveness of the proposed method, we compare its performance with other possible alternatives. The methods subject to comparison include various wrapper methods and machine learning-based methods. Metaheuristic-based methods are not considered here due to their high computational requirement.

2.4.1 Data Description

The existing studies of feature selection typically concern classification problems and often use the MONK's problem [18] to assess algorithm performance [60]. Three data sets are available for this problem, and for each, different feature combinations are used to generate a response variable. Among the three, we use MONK 1 and MONK 3 data sets but not

MONK 2. For MONK 2 data set, all features and 2-factor interactions are designed to be relevant to the response, so the benefit of feature selection achieving dimensionality reduction cannot be clearly shown with this data set. Both MONK 1 and MONK 3 have 6 features and a single response, and they have 556 and 554 data points, respectively. One thing to note is these data sets were designed for a classification problem and the response is a binary variable taking 0 or 1 for two class labels; for the complete data generation process, please refer to Thrun et al. [60].

Another data set we consider in this section is gene data from Golub et al. [23]. This data set contains 7130 predictors with 72 data records, so it has vast feature space. Instead of exploring over all predictors, we choose 30 predictors based on a relevancy measure to keep the dimensionality manageable but still large enough. Specifically, we calculate the correlation coefficient between each predictor and the response and select 10 predictors from each of three groups of strong, medium, and low relevancy in terms of the linear relationship. This problem is also for classification where the response is a categorical variable with three class labels of 0, 1, and 2.

As we consider a regression problem, we apply the variable transformation of the response variable so that the data sets can be more suitable for regression analysis. In particular, we create a new response variable y from the original categorical response y' as $y = 10y' + v$ where v is a normal random error with mean of 0 and standard deviation of 5. The same treatment is applied to all three data sets we consider, including MONK1, MONK3, and gene data. The scale of 10 and the standard deviation of 5 are chosen as this setting allows non-separable data from the classification perspective and thereby prevents the regression task from being too trivial.

2.4.2 Implementation Detail

By design, the randomized subspace modeling includes a 5-fold CV for the selection of critical subspaces. When comparing prediction performance with other methods, another layer of a 5-fold CV is implemented to have separate test data sets. The similar treatment is applied to other methods being compared. One layer of a 5-fold CV is used to split test data sets, and another layer of a 5-fold CV is used to learn hyperparameters, if applicable.

The prediction error is measured by RMSE, and the average and standard deviation of 5 RMSE's are compared. For MONK data sets, they are used to justify the selection of significant features and interactions, there is no need for separate test data, so all data points are used for model learning and feature selection. The other methods subject to comparison include linear regression (LR), lasso regression (Lasso), ridge regression (Ridge), principal component regression (PCR), partial least squares regression (PLS), RF, k-nearest neighbors (k-NN), SVR, and NN to cover various linear and nonlinear modeling commonly used for feature selection. Before applying any method, we normalize the data to prevent any scale-relevant errors.

To implement the randomized subspace modeling, we set the selection and termination thresholds to $\eta = 1\%$ and $\tau = 0.001\%$, respectively, based on the results from preliminary studies; see Appendix B. The dimension of subspaces, k, is set to either 2 or 3 depending on the problem we consider. For the base learner of the subspace-based modeling (SVR), a kernel needs to be chosen among multiple possible options, including linear, polynomial (POLY), radial basis function (RBF), and sigmoid kernels. From preliminary experiments, we found kernels capturing linearity, such as linear and POLY kernels, provided better performance. In this regard, we choose POLY kernel, $K(\mathbf{u}, \mathbf{v}) = (\gamma \mathbf{u}^T \mathbf{v} + \delta)^d$ where γ, δ, and d, respectively, denote the scale, offset, and degree of polynomials, and set $\gamma = 1/k$, $\delta = 0$, and $d = 1$ (found optimal). Since γ had little impact on the final prediction error, we use the default value of an R function for SVR fitting. Considering that we use normalized data set, it is natural to have $\delta = 0$ provide the optimal performance. Although $d = 1$ yields the optimal result, the POLY kernel is chosen over the linear kernel as it is more flexible and capable of modeling nonlinearity with different parameterization. On the other hand, SVR learning itself involves some parameters, ϵ and $C := 1/\lambda$. We test three levels for each parameter, i.e., $\epsilon = 0.01, 0.1, 0.5$ and $C = 1, 2, 5$, and this generates nine distinct parameter settings among which an optimal setting is determined. When applying a grid search for this hyperparameter learning, if there exists a poor level combination, the designed termination criterion may not work properly. To prevent iterating more than what is required for the exhaustive search, we force the search process to stop after a certain number of iterations. Assuming three-dimensional subspaces, for the MONK's data sets involving 6 features, we

use 20 ($\binom{6}{3} = 20$) for the maximum number of iterations. Similarly, for the gene data, we use 4060 ($\binom{30}{3} = 4060$) to limit the number of iterations.

Excluding NN and the proposed method, all methods are executed in R by using `train()` function in the `caret` package. Chen et al. [13] show `varImp()` function (with RF) is an efficient tool for calculating variable importance, so we use the same function to measure variable importance for other methods. NN is implemented in Python by using `keras` package. To make the result comparable to the existing variable selection methods based on NN [42, 37], we test over multiple NN structures, one or two hidden layer(s), and three activation functions of linear, sigmoid, and ReLU. We also evaluate various hyperparameter options including the numbers of neurons, batch size, and epochs. With all this variation, we pick an NN model with the lowest prediction error for the result comparison. For NN, variable importance is calculated by the Connection Weight Approach, referring to the experimental results in Olden et al. [42]. Since all these methods rely on variable importance scores for feature selection, we list the first twenty variables with high importance score for each 5-fold learning and find the common variables selected shown in the 5 lists to report critical individual variables.

2.4.3 Experimental Results

From the MONK's problem, we investigate if feature selection methods can accurately identify important features (and interactions) and also compare the running time of feature selection alternatives. Among 6 variables available in the MONK data sets, i.e., a_i for $i = 1, \ldots, 6$, (a_1, a_2) and a_5 are the significant features in MONK 1 where the parentheses signify interactive features. In MONK 3, (a_2, a_5) and (a_4, a_5) are designed to be significant. Since there are three significant variables in MONK 1, we set the dimension of subspaces to 3, i.e., $k = 3$. MONK 3, on the other hand, includes two interactive features of dimension of 2, so we set $k = 2$ for this data set.

For MONK 1, the critical subspace captured by our method is (a_1, a_2, a_5). The ideal selection is to choose (a_1, a_2) and a_5 separately. The proposed method imposes a fixed dimension on subspaces, and this limits its flexibility in selecting features. Still, the selected three-dimensional subspace involves all target features, including the interaction between

a_1 and a_2. Among all alternatives, only the proposed method and RF identify the correct variables for significant features. Other methods miss one or two important variables. Please note, the other methods including the RF-based feature selection are not capable of specifying any significant interaction. The presented results merely show their selection of individual variables. For MONK 3, our method correctly selects the two sets of interactive features, i.e., (a_2, a_5) and (a_4, a_5). In addition, the proposed method is the only method identifying all significant variables while the others miss one important variable, a_4. The result of the feature selection and computing time is shown in Table 2.1.

As can be expected, linear models do not require a large amount of computation; running each of them can be completed in 3 seconds for this particular analysis. Among nonlinear models, k-NN and SVM with a polynomial kernel achieve comparable computational time marking slightly greater than 3 seconds. The other nonlinear models require more computations in general. Our method's running time is greater than RF's but still comparable in scale; also much less than NN's computational time. Although the proposed method requires more computations, it is worthwhile considering the feature selection results. Understanding the system is very important is practical application. In this filed, our method is a powerful tool.

The comparison of prediction capability from the gene data is shown in Table 2.2. For this particular data set, in general, nonlinear models perform slightly better than linear counterparts in terms of the average of 5 test errors calculated from the 5-fold cross validation. Although NN has relatively poor accuracy, this can be attributed to the lack of data records; note that there are only 72 observations in this data set. The standard deviations of each method are calculated from 5 test errors. As we can see, the standard deviation are similar between linear and nonlinear models. Among all the alternatives, k-NN obtains the lowest average prediction error with a relatively high standard deviation while LR achieves the smallest standard deviation with the highest prediction error. On the other hand, the proposed method attains not only the lowest prediction error but also the lowest standard deviation. This result demonstrates that the proposed method, subspace-SVM, not only has strength in feature selection but also attains decent predictions.

Table 2.1: Comparison of the feature selection results and running time for the MONK's Problem

Method	MONK 1		MONK 3	
	Sig. features	Time (s)	Sig. features	Time (s)
LR	a_5, a_6	2.81	a_2, a_5	2.07
Lasso	a_5	2.36	a_2, a_5	2.37
Ridge	a_5, a_6	2.31	a_2, a_5	2.39
PCR	a_5	2.96	a_2, a_5	2.18
PLS	a_5	2.28	a_2, a_5	2.3
RF	a_1, a_2, a_5	38.38	a_2, a_5	37.57
k-NN	a_5	3.30	a_2, a_5	2.33
SVM (RBF)	a_5	12.63	a_2, a_5	12.55
SVM (POLY)	a_5	4.07	a_2, a_5	4.06
NN	a_2, a_5	720.00	a_2, a_5	840.00
Subspace-SVM	(a_1, a_2, a_5)	**51.50**	(a_2, a_5), (a_4, a_5)	**60.72**
Ground Truth	(a_1, a_2), a_5		(a_2, a_5), (a_4, a_5)	

Table 2.2: Comparison of the average and standard deviation of RMSE's from 5-fold CV evaluation (gene data)

Method	Average	Std. dev.
LR	11.91	1.22
Lasso	9.92	1.54
Ridge	9.59	1.44
PCR	9.55	1.62
PLS	10.28	1.72
RF	9.43	1.75
k-NN	9.41	1.54
SVM (RBF)	9.5	1.38
SVM (POLY)	9.55	1.40
NN	10.87	1.69
Subspace-SVM	**9.41**	**1.21**

2.5 Damage Tolerance Analysis of Hybrid Materials

Design and analysis of layered hybrid structure is challenged by the many possible choices of materials and configurations. This vast parameter space is impractical to explore through physical testing and is computationally prohibitive due to the analysis time required due to the large number and ranges of input parameters. The proposed method is applied to overcome this limitation and provides an efficient approach to computationally characterize the parameter space and informs limited physical testing to define parameters. The reduced order model can then be used to rapidly explore the parameter space to customize and optimize layered designs. The specific problem of metal and composite layered structures was chosen due to its complexity and fit for the demonstration objectives.

The major purpose of this section is to determine important variables for modeling the damage tolerance of layered hybrid structures. In the absence of prior knowledge about variable importance, we compare the selection outcome of the proposed method with those of other alternatives. To ensure the quality of models used for feature selection, we compare the prediction accuracy of the models in addition to variable selection results. For more effective comparison, 5-fold CV is used for this analysis. Since the other methods focus on the importance of "individual" variables, we extract a common set of important variables obtained from the 5-fold CV implementation for comparison purpose.

2.5.1 Finite Element Analysis

Numerical simulation using a validated finite element model (finite element analysis) is an efficient method to design and predict the damage tolerance of a layered hybrid structure for varying material properties. However, given the nearly unlimited choices of material combinations and stacking sequences, it is not possible to explore every possible design and optimize for all possible material and configuration parameters. Therefore, identification of the most influential parameters is necessary to limit the design parameter space to obtain a computationally tractable solution. The case study layered hybrid configuration is an aluminum plate with a co-cured bonded quasi-isotropic E-glass/epoxy composite overlay. The composite overlay consists of 8 layers of multiple lamina types (see Table 2.3) resulting

Table 2.3: Composite patch stacking sequence

	E-glass fabric	Fabrication style
1	Hexcel 7781	0°/90° Stain weave
2	Vectorply E-BX 1200	± 45° Stitch
3	Vectorply E-LT 1800	0°/90° Stitch
4	Vectorply E-BX 1200	± 45° Stitch
5	Vectorply E-BX 1200	± 45° Stitch
6	Vectorply E-LT 1800	0°/90° Stitch
7	Vectorply E-BX 1200	± 45° Stitch
8	Hexcel 7500	0°/90° Plain weave

in a high dimensional number of material parameters needed for characterization and input into the finite element model.

This layered multi-material model was loaded under four point bending to engage progressive damage in the structure through multiple damage mechanisms. Damage tolerance was evaluated by the total energy absorbed by the structure (an output parameter that is calculated during analysis and readily extracted using an automated approach). This model captures multiple, interacting damage mechanisms including the plastic deformation in aluminum, shear plasticity in each lamina, the intralaminar fracture of each lamina, delamination within the patch and disbond at the interface. Evaluation using this total damage energy provides a distinct and measurable result for the development of a reduced order predictive model and evaluation of influential parameters and their interactions.

A 3D high fidelity finite element model explicitly captures each layer in the hybrid structure as well as the interfaces [28]. Each fabric layer (lamina) is explicitly modeled, and cohesive elements are included between each layer to capture delamination between plies. Each lamina is individually modeled with continuum shell elements (SC8R). A cohesive damage model is implemented for each lamina using a VUMAT user subroutine. Cohesive elements with a triangular traction-separation law are used to detect the interlaminar damage and are also included at the metal/composite interface to capture disbond between the metal and resin. The aluminum substrate is modeled with solid elements (C3D8R). Loading and support pins are modeled as rigid bodies to create the boundary and loading conditions. The numerical simulations are executed in the FE code ABAQUS. This physics-based model was validated under four point bend loading through physical testing (Fig. 2.2) .

To predict the total energy absorption of the layered hybrid structure, the 41 parameters characterizing the material properties needed for input into the finite element model for the multiple layers are used as predictors (See Table 2.4 for details). Latin hypercube sampling is applied to sample the parameter space based on the mean and standard deviation of the parameter ranges. For this case study, completing a single analysis generating a single data record takes an average of 3 hours depending on the parameter combination. Due to the computational time required to analyze the finite element model, we conducted 200 analyses producing a predictor matrix of size 200×41 and a response vector of size 200. With

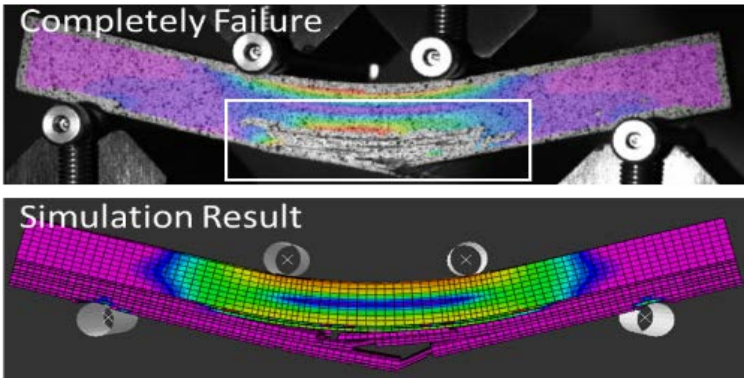

Figure 2.2: Comparison of experimental (top) and numerical results (bottom) at failure under four point bending

Table 2.4: Variables for modeling damage tolerance of a hybrid metal structure (the Resin in this table is "Resin (M1002 with M2046 Hardener)") [28]

Variable	Corresponding parameter
x_1 (σ_y)	Yield Stress of Al-5456
x_2 (n)	Strain Hardening Exponent of Al-5456
x_3 (σ_{ss})	Nominal Stress First/Second Direction of Resin between Laminate plies
x_4 (E_{Al})	Young's Modulus of Al-5456
x_5 (P)	Power Term in Shear Hardening Equation for Laminates
x_6 (σ_{nn})	Nominal Stress Normal-only Mode of Resin between Laminate plies
x_7 (X_{7781})	Tensile strength of the Laminae Reinforced with Hexcel 7781
x_8 (G_{1200})	Intralaminar Fracture Toughness of Laminae Reinforced with EBX 1200
x_9 (ϵ^{pl}_{max})	Maximum Shear Plastic Strain for Laminates
x_{10} (G_{II})	Shear Mode Fracture Energy First/Second Direction of Resin between Laminate plies
x_{11} (α_{12})	Shear Damage Parameter for Laminates
x_{12} (E_{1800})	Young's Modulus of Laminae Reinforced with ELT 1800
x_{13} (G_I)	Normal Mode Fracture Energy of Resin between Laminate plies
x_{14} (X_{7500})	Tensile strength of the Laminae Reinforced with Hexcel 7500
x_{15} (σ_{nni})	Nominal Stress Normal-only Mode of Resin between metal/composite interface
x_{16} (v_{Al})	Poisson's Ratio of Al-5456
x_{17} (E_{7500})	Young's Modulus of Laminae Reinforced with Hexcel 7500
x_{18} (σ_{ssi})	Nominal Stress First/Second Direction of Resin between metal/composite interface
x_{19} (X_{1800})	Tensile strength of the Laminae Reinforced with ELT 1800
x_{20} $(B-K_i)$	Mixed Mode Behavior for Benzeggagh-Kenane of Resin between metal/composite interface
x_{21} (E_{1200})	Young's Modulus of Laminae Reinforced with EBX 1200
x_{22} (G_{1800})	Intralaminar Fracture Toughness of Laminae Reinforced with ELT 1800
x_{23} $(\tilde{\sigma}_y)$	Effective Shear Yield Stress for Laminates
x_{24} (X_{12})	Shear Strength of Resin for all Lamina
x_{25} (B)	Strength Coefficient of Al-5456
x_{26} (v_{7500})	Poisson Ratio of Laminae Reinforced with Hexcel 7500
x_{27} (X_{1200})	Tensile strength of the Laminae Reinforced with EBX 1200
x_{28} (v_{1200})	Poisson Ratio of Laminae Reinforced with EBX 1200
x_{29} (v_{1800})	Poisson Ratio of Laminae Reinforced with ELT 1800
x_{30} (G_{7500})	Intralaminar Fracture Toughness of Laminae Reinforced with Hexcel 7500
x_{31} (E_{7781})	Young's Modulus of Laminae Reinforced with Hexcel 7781
x_{32} (v_{7781})	Poisson Ratio of Laminae Reinforced with Hexcel 7781
x_{33} (G_{7781})	Intralaminar Fracture Toughness of Laminae Reinforced with Hexcel 7781
x_{34} (G_{12})	Shear Modulus of Laminate for Laminates
x_{35} (d^{max}_{12})	Maximum Shear Damage for Laminates
x_{36} (C)	Coefficient in Shear Hardening Equation for Laminates
x_{37} (E_{nn})	Elastic Modulus of Resin between Laminate plies
x_{38} $(B-K)$	Mixed Mode Behavior for Benzeggagh-Kenane of Resin between Laminate plies
x_{39} (E_{nni})	Elastic Modulus of Resin between metal/composite interface
x_{40} (G_{Ii})	Normal Mode Fracture Energy of Resin between metal/composite interface
x_{41} (G_{IIi})	Shear Mode Fracture Energy First/Second Direction of Resin between metal/composite interface

more analyses, it is possible to generate a larger data set, and this can further improve the prediction quality. However, from a practical perspective, we aim to develop a method that works well even with a small data set, so we compare alternatives based on this small data set.

2.5.2 Feature Selection Outcomes

In this section, we compare the proposed method with the same alternatives used in Section **??** and apply the implementation settings discussed in Section 2.4.2. For this particular data set, the proposed method creates and evaluates 3-dimensional subspaces, $k = 3$, and limits the maximum number of iterations to 10000 ($\binom{41}{3} = 10660$). Among 9 different hyperparameter level settings, this hard threshold for termination becomes active only for a single level combination, so the termination criterion described in Section 2.3.2 generally works well (on average, having 3154 iterations). The optimal hyperparameter values of the proposed method vary a little with each model learning set (five of such). Table 2.5 shows the optimal values selected based on the two GCV criteria. From the table, it is shown, in general, $\epsilon = 0.01$ and $C = 5$ are found optimal.

The optimal models based on the result in Table 2.5 are compared with other alternatives as shown in Table 2.6. It is noticed that, in general, linear models perform better than nonlinear models for the material damage tolerance prediction. This applies to both the average and standard deviation of the prediction errors. Among the linear methods, PLS regression produces the lowest average prediction error while the regularized methods (lasso and ridge) show comparably decent performance. Although LR attains the smallest standard deviation, its average prediction error is relatively higher than the other methods. For nonlinear methods, only SVM with a polynomial kernel and NN provide an average RMSE comparable to that of the linear counterpart. The two methods also show a similar level of robustness of the estimator (through standard deviation). The proposed subspace-based method achieves the best performance among all the methods. The model based on A1 GCV criterion is comparable to the best linear methods, and the model based on A2 outperforms all other methods with respect to the average and standard deviation of the RMSE's. This

Table 2.5: Optimal parameters selected for each of the five model learning/test data split

		Learn/test 1	Learn/test 2	Learn/test 3	Learn/test 4	Learn/test 5
A1	ϵ	0.01	0.01	0.01	0.01	0.01
	C	5	5	5	5	2
A2	ϵ	0.1	0.01	0.01	0.01	0.01
	C	5	5	5	5	2

Table 2.6: Comparison of the average and standard deviation of RMSE's from 5-fold evaluation

Method	Average	Std. dev.
LR	12.61	1.05
Lasso	12.02	1.45
Ridge	11.98	1.20
PCR	12.21	1.64
PLS	11.89	1.15
RF	14.70	3.00
k-NN	15.27	3.51
SVM (RBF)	17.49	3.77
SVM (POLY)	12.01	1.20
NN	12.45	1.13
Subspace-SVM (A1)	11.99	1.12
Subspace-SVM (A2)	**11.64**	**0.94**

indicates that the proposed methods provide a model that is not only more accurate but also more robust and stable.

Although A1 applies the outcome of Theorem 2.1 exactly, it is based on the assumption of a squared loss function. As such, A1 is also an approximation of the exact GCV criterion. Since ϵ-insensitive loss function used in Eq. (2.5) penalizes errors out of the ϵ-tube proportionally to the distance from the tube, the calculation based on a squared loss function that more harshly penalizes larger errors may not guarantee the optimal performance. In A2, simplifying the hat matrix, i.e., dropping the recursive term of $(\mathbf{I} - \sum_{l=0}^{j-1} \mathbf{S}_l)$, removes dependency on other $\mathbf{S}_j$'s. This can reduce the variance of the estimator producing better results. In addition, from the computational perspective, A2 does not require storing and adding all the previous $\mathbf{S}_j$'s but just requires calculating the trace of $\mathbf{S}'_j$ for each subspace model estimation, so it is computationally more efficient than A1.

By construction, the randomized subspace model extracts critical subspaces. However, none of the other methods has this capability. To assess the variable selection capability of the randomized subspace method, we instead compare which individual variables are selected as an important predictor. As illustrated in Fig. 2.3, we first generate a set of all individual variables included in the selected subspaces for each 5-fold evaluation (used at the testing level), $\mathcal{X}_q$ for $q = 1, \ldots, 5$. Then, we extract common variables included in all five sets of important individual variables for comparison. Similarly, for all other methods, we find a set of important individual variables from each 5-fold evaluation and select the common variables from the sets, for the consistency of the comparison procedure.

The result of important individual variable selection is shown in Table 2.7. In the table, certain variables are identified as important variables by all methods while some are selected by only a few methods. For illustration purpose, we exclude all other variables that are not selected by any of the methods. According to the variable selection result, PCR and NN missed a variable (x_8 and x_3, respectively) that is chosen by all others. On the other hand, our method captures all variables that are found important by all the other methods, such as x_1, x_2, x_4 and x_7 as well as x_8 and x_3. Variables identified by the majority of the other methods (e.g., x_5 and x_{12}) are marked as important in our method. Certain variables rarely identified by the others (e.g., x_9 and x_{11}) are not included in the set of important variables

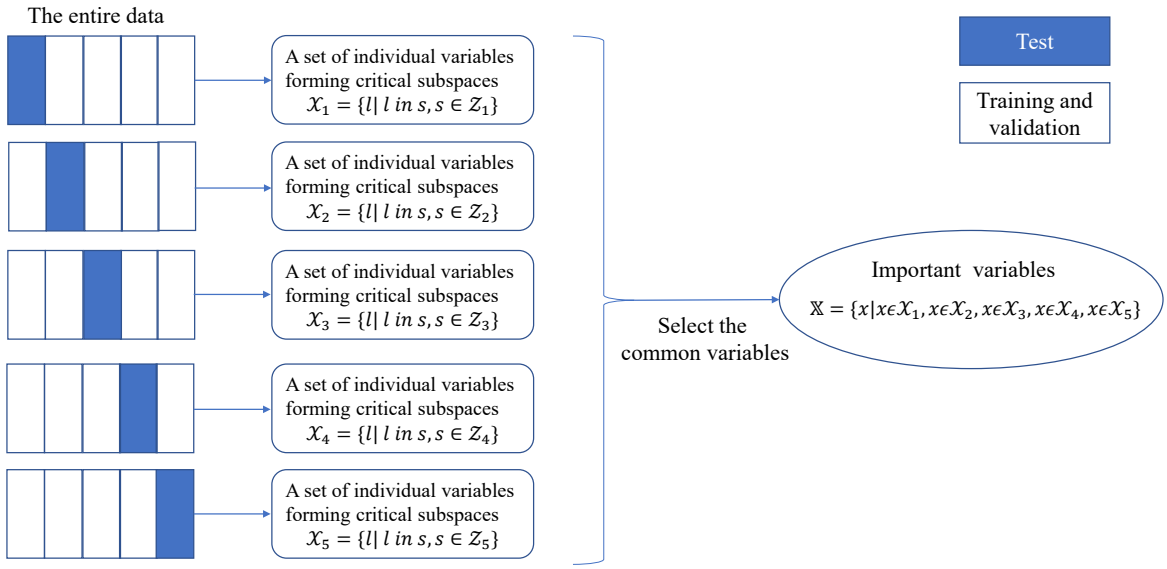

Figure 2.3: A procedure of selecting important individual variables

Table 2.7: Important variables selected by all alternatives; the alternatives with poor prediction errors are dimmed with gray color.

Method	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
Linear	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓								
Lasso	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓						
Ridge	✓	✓	✓	✓	✓	✓	✓	✓			✓		✓	✓				
PCR	✓	✓	✓	✓			✓					✓	✓	✓	✓	✓		
PLS	✓	✓	✓	✓	✓		✓	✓		✓		✓						✓
RF	✓	✓	✓	✓	✓		✓	✓					✓			✓	✓	
k-NN	✓	✓	✓	✓			✓	✓				✓	✓	✓	✓		✓	
SVM (RBF)	✓	✓	✓	✓			✓	✓				✓	✓	✓	✓		✓	
SVM(POLY)	✓	✓	✓	✓			✓	✓				✓	✓	✓	✓		✓	
NN	✓	✓		✓	✓	✓	✓	✓				✓						
Our method	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓						

from our method. In a few cases, our method selects a variable not chosen by the majority (such as x_6 and x_{10}) and does not select a variable voted by the majority (x_{13} and x_{14}). Note here that, however, according to Table 2.6, the prediction quality of RF, k-NN, and SVM with an RBF kernel is not quite acceptable. By excluding the variable selection results from these methods, our method is, in fact, capable of determining important and unimportant variables, of which results well align with the results of the remaining methods. In summary, this comparison result demonstrates that our method effectively extracts important variables.

With regard to the physical interpretation of these results, the total damage energy can be described as the sum of the energies contributed by 5 specific damage mechanisms [4]. [4] performed a comprehensive characterization and evaluation of the total damage energy behavior as well as the individual damage mechanisms. They determined that x_1, x_2, x_3, and x_4 were the most influential parameters by a considerable margin. For the loading case investigated, plastic deformation of the metal layer was significantly the largest contributor to the total energy with the corresponding significant parameters of x_1, x_2, and x_4 for this damage mechanism. Meanwhile, x_3 was a highly influential parameter for both the shear plasticity in the laminate and interlaminar fracture, which were the next two highest contributors to the total energy. Almost all methods accurately identified these parameters as significant. The next two most influential parameters from a physical perspective are x_5 and x_{10} which some of the methods, including ours, captured and others missed. x_{13} and x_{14} were not physically determined to contribute to any of the individual damage mechanics, however, were determined as significant by some of the other methods investigated. Our method correctly did not identify these two parameters as influential.

Discussion of the other parameters determined as significant but of less influence becomes more complex and dependent on the sampling method. As determined by [4], not only does the total energy vary throughout the parameter space, but also the contributions of the damage mechanisms. Therefore, the significance of some parameters varies depending on the location in the parameter space according to which damage mechanisms govern at that location. For example, x_{11} is not one of the most significant parameters, but is identified as influential in both interlaminar fracture and interlaminar delamination and x_9 is not one of the most significant parameters, but is identified as influential in both shear plasticity in the

laminate and interface disbond. These parameters could be significant to the total energy in local subspaces while not as influential globally. Alternatively, x_6 is not a significant parameter on its own from a physical perspective, but may contribute through parameter interaction, an effect requiring future investigation.

Besides the popular machine learning methods, we also compare our variable selection result with that of the elementary effects method [28] in Table 2.8. The result of the elementary effects method was derived from a data set generated by one-factor-at-a-time approach, and the method did not extract common variables from 5-fold CV and did not record the top 20 important variables (instead, reported the top 10 important variables). Albeit this is not a valid comparison as there are other aforementioned factors affecting the variable selection, we aim to compare our finding to the existing variable selection for this specific problem. In Table 2.8, there are four common variables selected by our method and the elementary effects method. The elementary effects method does not mark some variables found important by the majority of the machine learning-based methods, including x_2, x_3, x_7, and x_8, and it identifies some variables that are never selected by others, such as x_{24}, x_{31}, x_{33}, and x_{41}. Without knowing the true importance of the variables, we cannot draw a solid conclusion, but our methods result well aligns with the result of other alternatives whereas the existing selection from the elementary effects method is a bit far from the majority vote. Our method adds x_2, x_3, x_5, x_6, x_7, and x_8 as important variables and drops x_{11}, x_{17}, x_{24}, x_{31}, x_{33}, and x_{41} to and from the existing variable selection.

2.5.3 Critical Subspace Analysis

One of the major novelties of randomized subspace modeling is the capability of extracting critical subspaces, equivalently, identifying important physical variables and interactions among the variables. In Section 2.5.2, the comparison result demonstrates the quality of the important variable selection of the proposed method. Note here that the result of the important (individual) variable selection was derived from critical subspaces identified in multiple model learning/testing data splits. This ensures the quality of the critical subspace selection to some extent.

Table 2.8: Important variables selected by elementary effects method and our method

Method	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{17}	x_{24}	x_{31}	x_{33}	x_{41}
Elementary effects method	✓			✓						✓	✓	✓	✓	✓	✓	✓	✓
Our method	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓					

To determine critical subspaces for the given data set, we used the entire data for model learning without leaving any separate data for testing. The final model is constructed by 11 distinct critical subspaces, as shown in Table 2.9. We observe that these subspaces contain most of the important individual variables found in Table 2.7, except x_6, while each subspace excluding the sixth subspace includes at least one or two important individual variable(s). Assuming that an interaction formed by individually important variables is more significant than an interaction between others, we can argue that the interactions formed by (x_1, x_{12}), (x_3, x_8), (x_1, x_7), (x_5, x_{10}), and (x_2, x_3) are significantly important.

There are some variables that are not identified as important individual variables but included in the critical subspaces. These variables by themselves may not have significant impact on the damage tolerance, but their interactions with others could influence the response. In the current form of the proposed method, we cannot validate if any interaction of variables in a subspace is significant. We will investigate this aspect as one of future study. For a reference, if only one variable is added to the model at a time ($k = 1$), the prediction error is 12.28 with the standard error of 0.84. By ignoring the interactions formed by multiple variables, the prediction becomes worse. This supports, at least to the minimum extent, that some of the interactions we modeled through subspaces improve the prediction, and hence they are considered significant. Meanwhile, this determination of critical interactions improves our understanding of the complex engineering problem by identifying behaviors that may be outliers to our current level of understanding and characterization.

2.6 Conclusion

In this paper, we propose the randomized subspace modeling to alleviate challenges in high-dimensional data analysis and provide valuable insight about an underlying system by identifying important physical variables and interactions. The proposed method leverages an ensemble of multiple models derived from critical subspaces, reduced-dimensional physical spaces. The critical subspaces are generated by a randomized search and evaluated by a cross-validated selection criterion. With this structure, the proposed method shows its superiority over other alternatives in modeling a regression problem and identifying important variables.

Table 2.9: Critical subspace selected by our method

	Critical subspace	Important individual variables
1	x_{33}, x_4, x_{13}	x_4
2	x_{12}, x_{19}, x_{14}	x_{12}
3	x_{12}, x_1, x_{16}	x_1, x_{12}
4	x_8, x_3, x_{15}	x_3, x_8
5	x_{27}, x_7, x_1	x_1, x_7
6	x_{28}, x_{19}, x_{18}	/
7	x_{19}, x_{29}, x_2	x_2
8	x_5, x_{14}, x_{10}	x_5, x_{10}
9	x_{34}, x_3, x_{11}	x_3
10	x_{23}, x_4, x_{15}	x_4
11	x_9, x_2, x_3	x_2, x_3

Specifically, from the analysis of hybrid material's damage tolerance, we derive the following conclusions:

1. Compared to other methods commonly used for variable selection (e.g., RF, NN, and Lasso), our method attains the lowest prediction error for this complex problem both in mean and standard deviation, demonstrating its competitiveness in high-dimensional prediction tasks.

2. The important variable selection result of the proposed method well aligns with the majority of other alternatives, verifying its variable selection capability. More importantly, our method identifies critical subspaces capturing not only important physical variables but also significant interactions, which is beneficial to experimental designs for broad engineering problems.

Chapter 3

Feature Subspace Selection (FSS)

3.1 Introduction

High-dimensional data analysis has gained more attention with the recent breakthrough in computing, and it is essential for analyzing various complex systems [76, 64]. One particular example is material development and customization processes where a vast set of potentially-relevant material design parameters are tested to achieve desired material behaviors. Without prior knowledge of the importance of the parameters, this process typically requires a considerable number of experiments to study the causal effects of the design parameters on the material response [71]. Comprehensive exploration of the parameter (feature) space is not feasible in practice, so simple approaches relying on rough exploration of the space, such as space filling approaches, are often used to generate a sequence of experimental runs [6]. For a material development process, such an experimental design tends to produce a small-sample, high-dimensional dataset, necessitating efficient high-dimensional analysis to understand the causal effects.

The high dimensionality of a dataset can be beneficial in terms of the fluency of information, but the analysis of a high-dimensional dataset is not an easy task due to the curse of dimensionality [1]. As the dimensionality of a dataset increases, the volume of the feature space increases dramatically, and the data therein become dissimilar and sparse in many ways, which restricts the capability of general data modeling strategies [53]. A common solution to this problem is to represent high-dimensional data in a low-dimensional space by removing redundant and noisy information, known as dimensionality reduction [72].

In general, dimensionality reduction can be classified by two types of methods, feature extraction and feature selection. Feature extraction projects the feature space into a low-dimensional intrinsic space and extracts features spanning the intrinsic space [34]. The projection creates new features that are different from the original ones having physical meanings, so the usage of feature extraction is limited in an experimental study where original physical features need to be controlled. Feature selection, on the other hand, examines the significance of existing physical features and determines which features to keep based on the significance evaluation [72]. This produces a smaller subset of features, pinpointing which design parameters to focus in subsequent experiments. However, common feature

selection approaches are not capable of identifying significant interactions between features, often leading to the exclusion of parameters that are insignificant on their own but essential in conjunction with others. It is known that discovering important interactions is very challenging, and only a few studies are available along this direction [1].

Recently, [9] proposed a randomized subspace-based model (RSM) randomly generating low-dimensional feature subspaces, evaluating significance of the subspaces, and integrating significant subspaces to model the response. Different from other feature selection approaches deriving a single subset of features considered significant as a whole, RSM produces multiple low-dimensional feature subsets and presents each as a significant interaction, producing a list of important interactions. To accommodate multiple feature subspaces, multiple models (one for each subspace) are learned and integrated as an ensemble to form a full model. In this paper, we extend the idea of RSM and propose a feature subspace selection (FSS) approach which is designed to identify significant physical features and interactions.

Although RSM has such a unique aspect, it has some limitations. First, RSM imposes fixed dimensionality on all subspaces, e.g., three-dimensional subspaces. This is not desired because a randomly generated subspace, albeit selected, may include some insignificant features merely to meet the dimensionality requirement. To define feature interactions more realistically, in this paper, we design each subspace with varying dimensionality. Meanwhile, RSM regards a randomly drawn subspace as a significant interaction if adding this subspace reduces the model's prediction error by a certain percentage. Depending on the level of prediction error at the time of evaluation, this strategy may result in including an insignificant subspace in the model. To address this issue, we apply double selection criterion leveraging statistical hypothesis tests as well as predictions. In addition, the deficient selection of RSM as well as its fixed dimensionality requirment make the termination of the algorithm difficult sometimes, necessitating a hard threshold to stop the algorithm after a fixed number of iterations. We redesign the termination process to make the entire learning process with the new additions run even faster than the original RSM. Throughout this paper, we compare various aspects of RSM and FSS, including prediction accuracy, interaction selection capability, and computational time.

To demonstrate the practical importance of the proposed FSS, a case study investigating the damage tolerance of a layered composite/metal structure is performed. Design and analysis of layered hybrid structures is challenged bymaterial and configuration selection which encompasses large number and ranges of input features. This vast parameter space is hard to explore via physical testing and computation due to the analysis time required. FSS is applied to overcome this drawback and provide an accurate and efficient analysis to computationally characterize the parameter space and utilize limited physical testing to define parameters. The proposed reduced order model can rapidly explore the parameter space to customize and optimize layered designs. The specific problem of metal and composite layered structures was chosen due to its complexity and potential interactions among input parameters.

The rest of this article is organized as follows. Section 3.2 reviews relevant studies on dimensionality reduction techniques with more focus on feature selection. Section 3.3 presents the proposed FSS by describing a model structure, model learning, subspace selection, and algorithm termination. Section 3.4 shows the proposed method's feature (interaction) selection capability through an experiment using common benchmark datasets. Section 3.5 demonstrates the benefits of the proposed method for modeling the damage tolerance of a hybrid material, in comparison with others. Section 3.6 concludes the paper and discusses future work.

3.2 Literature review

With the explosive increase in the number of applications dealing with high-dimensional datasets, dimensionality reduction has become common in many areas, including bioinformatics, chemistry and speech signal analysis [63, 56]. Dimensionality reduction can be split into two branches of feature extraction and feature selection depending on the characteristics of reduced dimensions.

Feature extraction, often perceived as dimensionality reduction itself, is an unsupervised learning technique projecting high-dimensional data into a low-dimensional space that minimizes information loss [34]. The projection involves variable transformation which

produces a new set of features spanning a low-dimensional space. In general, it is difficult to interpret these newly-created artificial features and their relations to the original physical features [63]. This limits the usage of feature extraction for an experimental study, especially for the process of designing an experiment and optimizing experimental response [59]. For this reason, feature selection preserving the original features while reducing dimensionality can be a good alternative for analyzing high-dimensional systems of science and engineering experiments.

Feature selection has gained broad attention in the field of machine learning and data mining in recent years [1, 77]. Feature selection aims to improve prediction performance, provide faster and more effective predictors, and bring a better understanding of an underlying process [25]. In general, there are three types of methods for feature selection: filter methods, wrapper methods, and embedded methods. Filter methods apply ranking criteria, such as Pearson correlation coefficient [25] and mutual information [7], to select the most influential features. Filter methods are well known for their simplicity and success in practical applications [12]. Wrapper methods explore various feature subsets and find the optimal one by leveraging a learning algorithm. Since wrapper methods use a learning algorithm minimizing errors in response modeling, they achieve better performance compared to filter methods [69]. To alleviate computational complexity in determining subsets to evaluate, sequential selection algorithms or heuristic search algorithms are used [12]. Often, sequential selection algorithms lack accuracy and suffer from a nesting problem, and heuristic search algorithms involve an unnecessarily massive amount of computation [25]. Embedded methods aim at reducing the computation time of wrapper methods by integrating feature selection and model learning into a single process [69]. Embedded methods try to compensate for the drawbacks in the filter and wrapper methods [12].

Unlike traditional wrapper methods that apply a hard selection of features, some recent studies employ a soft selection of features by calculating variable importance scores from machine learning models, such as lasso regression [61, 52], random forest [13] and artificial neural network (ANN) [42]. [52] apply a lasso technique to corporate credit ratings. They conclude that variable selection models, such as the proposed one based on lasso, can obtain superior estimations and improve predictive ability thanks to their ability to uncover the

underlying structure of a problem. [13] suggest using random forest for calculating variable importance and selecting features based on the calculated importance. While using multiple popular datasets, they show the random forest-based approach outperforms recursive feature elimination leveraging a permutation importance measure as a ranking criterion. [42] generate simulated data for which the true variable importance is known and compare nine variants of ANN assessing variable importance by using the simulated data. Their experimental results show a connection weight approach produces the lowest error among all alternatives, and the approach correctly estimates the true ranking of variable importance.

Although the field of feature selection has made great progress, there are only a few papers trying to identify significant feature interactions [1]. Zhao and Liu [74] consider feature interactions in a subset selection process by employing an information-theoretic feature ranking mechanism. The method relies on a filter algorithm named INTERACT which selects relevant features while implicitly exploring the presence of feature interactions. They suppose feature interactions exist if a removal of some variables in a set makes the relevance of the class label with a feature or a feature subset decrease. Zeng et al. [73] redefine the redundancy and interaction of features in the frame of information theory and devise an interaction weight factor reflecting the redefined information and accordingly an interaction weight based feature selection (IWFS) algorithm to address feature interactions. To find targeted features, including relevant and interacting features, the IWFS algorithm applies forward selection based on the interaction weight factor. Their method obtains the best average accuracy among five algorithms (including the INTERACT) when applied to real-world datasets.

The aforementioned studies regarding feature interaction all produce a single subset of features that includes all important features and potential interactions. In this regard, their selection of important interactions is implicit as it is hard to point out which features interact with which and which do not, lacking the practical importance. On the other hand, by using a single set of features when modeling a response, the corresponding model is not free of the curse of dimensionality, especially when the set involves many features. To the best of our knowledge, Bo et al. [9] is the only study producing multiple feature subsets, each of which clearly dictates the interaction relation among features.

Bo et al. [9] perceive a feature subspace as a basic modeling unit. They apply a randomized search to generate a potential subspace, evaluate the significance of a subspace by assessing prediction errors, and integrate the selected subspaces in a boosting fashion while continuously updating predictions until convergence. They show the benefit of the subspace-based modeling in prediction and feature selection by using multiple datasets, and their method produce a list of significant subspaces, equivalently, a list of significant interactions. As stated earlier, however, RSM has some limitations in subspace dimensionality, subspace selection process, and algorithm termination process. We will describe how the proposed FSS tries to overcome these limitations by highlighting the differences between FSS and RSM whenever relevant.

3.3 Feature subspace selection

This section develops and introduces step-by-step procedures of FSS. In the following, we discuss an ensemble model structure combining multiple subspace-based models, strategies for subspace exploration, model learning process based on support vector machine (SVM), critical subspace selection, and overall algorithm implementing the proposed method.

3.3.1 Subspace-based ensemble model

Suppose a dataset containing data tuples of $\{\mathbf{x}_i, y_i\}_{i=1}^n$ is given where $\mathbf{x}_i$ is a p-dimensional feature vector and y_i is response. We assume a function $f : \mathbb{R}^p \to \mathbb{R}$ maps a p-dimensional input $\mathbf{x}$ to a single-dimensional output y as $y = f(\mathbf{x}) + \varepsilon$ where ε is an additive noise. We model the function f as an additive mixture of subfunctions $g_j : \mathbb{R}^{d_j} \to \mathbb{R}$ for $j = 1, \ldots, J$ defined in a lower dimensional space of dimension d_j, as

$$y = f(\mathbf{x}) + \varepsilon \approx g_1(\mathbf{z}_1) + g_2(\mathbf{z}_2) + \cdots + g_J(\mathbf{z}_J) + \varepsilon, \tag{3.1}$$

where $\mathbf{z}_j$ is the jth subvector of $\mathbf{x}$ of size d_j; we refer to a vector space associated with each $\mathbf{z}_j$ as a subspace. The dimensionality of all J subspaces are restricted by an upper bound of a predefined dimensionality $d \ll p$ such that $d_j \leq d$ for $\forall j = 1, \ldots, J$. In practice, a high-order

interaction is often found insignificant, so we rule out subspaces of which dimension exceeds a preset value. Different from [9] assuming $d_1 = d_2 = \ldots = d_J = d$, we allow a flexible dimensionality for each subspace.

In Eq. (3.1), each $\mathbf{z}_j$ will take different components of $\mathbf{x}$. Suppose $d = 3$ and $\mathbf{x} = [x_1, \ldots, x_p]^T$ where $p > 20$. When $J = 3$, the subspaces used to build the model in Eq. (3.1) can possibly include, for example, vector spaces formed by $\mathbf{z}_1 = [x_3, x_8, x_{12}]^T$, $\mathbf{z}_2 = [x_{19}]^T$, and $\mathbf{z}_3 = [x_7, x_{16}]^T$ with the respective dimensionality of 3, 1, and 2 which are less than or equal to $d = 3$. By construction, each subspace can model up to the d_j-th order interaction with the maximum possible order being d.

To integrate the subfunctions g_j, we apply a boosting strategy. In particular, we include a new subspace (and the associated subfunction) in the model if it is significant in improving the prediction of the current model and iteratively repeat this procedure to form the model in Eq. (3.1). For this purpose, a new subspace will predict the errors of the most recent model. As shown in Fig. 3.1, when no subspace is selected yet, a newly-generated subspace predicts the initial response y. Once a subspace $\mathbf{z}_1$ is selected as it being significant, the response is updated to $r_1 := y - \hat{g}_1(\mathbf{z}_1)$ and remains the same until another subspace $\mathbf{z}_2$ is selected. After $\mathbf{z}_2$ is selected, the response is updated to $r_2 := r_1 - \hat{g}_2(\mathbf{z}_2) = y - \hat{g}_1(\mathbf{z}_1) - \hat{g}_2(\mathbf{z}_2)$. This procedure continues until a termination criterion is met. In summary, the jth subspace estimates a model expressed as

$$r_{j-1} = g_j(\mathbf{z}_j) + \varepsilon_j, \tag{3.2}$$

where $r_{j-1} := y - \sum_{l=1}^{j-1} \hat{g}_l(\mathbf{z}_l)$ for $j = 2, \ldots, J$ and $r_0 := y$. ε_j denotes the error structure of the jth model.

3.3.2 Subspace model learning

At iteration t of the subspace search process, a randomly generated subvector, $\tilde{\mathbf{z}}_t$ is given. We build a model by using this subvector and test if it is a good candidate for the jth subspace. This implies that $j - 1$ subspaces have been already selected, so the response of the model is the residuals after having the $j - 1$ subspaces in the full model. In other words,

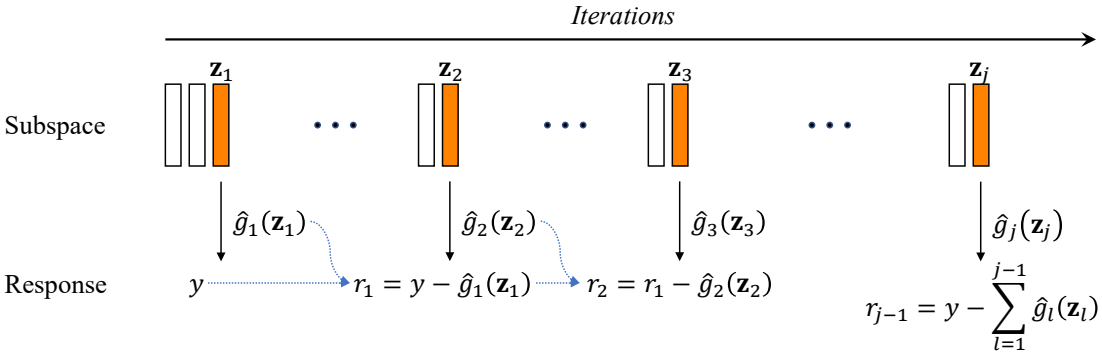

Figure 3.1: Boosting strategy to form an ensemble of subspace-based models: the empty boxes denote subspaces found insignificant whereas the orange boxes show those selected as significant subspaces.

we use r_{j-1} as a response to estimate as shown in

$$r_{j-1} = g_t(\tilde{\mathbf{z}}_t) + \varepsilon_t. \tag{3.3}$$

Once $\tilde{\mathbf{z}}_t$ is selected as a significance subspace, $\mathbf{z}_j$ and g_j are updated by $\tilde{\mathbf{z}}_t$ and g_t, respectively, becoming a part of the model in Eq. (3.1).

As a base learner, we use support vector regression (SVR) to estimate the function g_t. By applying basis expansion, g_t can be expressed as

$$g_t(\tilde{\mathbf{z}}_t) = \sum_{q=1}^{Q} \beta_q h_q(\tilde{\mathbf{z}}_t) + \beta_0, \tag{3.4}$$

where h_q and β_q for $q = 1, \ldots, Q$ are basis functions and the corresponding coefficients, respectively. β_0 is a constant. Among different loss functions available for SVR, we use the "ϵ-insensitive" loss function to learn the model [5], so we minimize

$$H(\beta_0, \beta_1, \ldots, \beta_Q) = \sum_{i=1}^{n_T} V_\epsilon(r_{j-1,i} - \hat{g}_t(\tilde{\mathbf{z}}_{t,i})) + \frac{\lambda}{2} \sum_{q=1}^{Q} \beta_q^2 \tag{3.5}$$

where $V_\epsilon(e) = |e| - \epsilon$ but zero if $|e| < \epsilon$ and n_T is the size of a training dataset. The subscript i for $r_{j-1,i}$ and $\tilde{\mathbf{z}}_{t,i}$ is used to indicate each data point in the training dataset. The minimizer of Eq. (3.5) produces an estimate of g_t expressed as

$$\hat{g}_t(\tilde{\mathbf{z}}) = \sum_{i=1}^{n_T} \alpha_i K_t(\tilde{\mathbf{z}}, \tilde{\mathbf{z}}_{t,i}). \tag{3.6}$$

where α_i for $i = 1, \ldots, n_T$ is a Lagrangian dual variable for the Lagrangian primal shown in Eq. (3.5), and $K_t(\mathbf{a}, \mathbf{b}) = \sum_{q=1}^{Q} h_q(\mathbf{a}) h_q(\mathbf{b})$ is a kernel function [27].

For the usage in subspace selection and hyperparameter tuning, we approximate the effective degrees of freedom of the SVR estimator. For this approximation, we assume a squared loss, i.e., $V_\epsilon(e) = e^2$, and under this assumption, the minimizer of Eq. (3.5) provides

$$\hat{\mathbf{r}}_{j-1} = (\mathbf{K}_t + \lambda \mathbf{I})^{-1} \mathbf{K}_t \mathbf{r}_{j-1}. \tag{3.7}$$

where $\mathbf{K}_t$ is the gram matrix of size $n_T \times n_T$, i.e., $\{\mathbf{K}_t\}_{i,i'} = K(\tilde{\mathbf{z}}_{t,i}, \tilde{\mathbf{z}}_{t,i'})$ for $i, i' = 1, \ldots, n_T$. $\mathbf{I}$ is the $n_T \times n_T$ identity matrix, and $\mathbf{r}_{j-1} = [r_{j-1,1}, \ldots, r_{j-1,n_T}]^T$ is the residual vector that is used as a response for this subspace-based model. Since the estimator can be expressed as a linear projection, its effective degrees of freedom (edf) can be derived as

$$\mathrm{df}(\hat{\mathbf{r}}_{j-1}) = \mathrm{tr}((\mathbf{K}_t + \lambda\mathbf{I})^{-1}\mathbf{K}_t), \tag{3.8}$$

where $\mathrm{tr}(\cdot)$ calculates the trace of a given matrix. We define $\mathbf{S}'_t := (\mathbf{K}_t + \lambda\mathbf{I})^{-1}\mathbf{K}_t$ for future reference.

3.3.3 Subspace exploration and selection

Identifying significant feature subspaces is challenging as there are a substantial number of potential subspaces to evaluate, e.g., $2^p - 1$ subspaces for a complete exhaustive search. Keeping the dimension of a subspace less than or equal to d can significantly lower the number to $\sum_{l=1}^{d} {}_pC_l$, but an exhaustive search of these many candidates is still computationally intensive. Bo et al. [9] draw a subspace from a set of ${}_pC_d$ candidates (as the dimensions are fixed at d) in a completely randomized fashion, which shows its benefits for prediction and important subspace selection. For subspace generation, we apply the exactly same procedure while keeping the dimensionality of a subspace constant at d.

Selection criteria

Whenever a random subspace is generated, we evaluate its significance and whether to include this subspace into the model or not. Bo et al. [9] consider a randomly generated subspace significant if the addition of this subspace reduces prediction error by more than a preset value in percentages. The preset threshold serves as a tuning parameter which is subject to estimation through a grid search. Although the error is evaluated in a cross validation (CV) framework, the procedure lacks of statistical justification in defining the significance. In this paper, similar to Bo et al. [9], we apply a k-fold CV for a robust estimate of the significance. However, the significance of a d-dimensional feature subspace will be determined by statistical hypothesis testing.

As shown in Fig. 3.2, each randomly generated subspace will be tested via k different training/validation data splits. This implies that there are k hypothesis tests to assess, one for each data split. To derive a unique conclusion from multiple hypothesis tests, we apply the Bonferroni correction [15]. To test k different sets of hypotheses for a given type I error, α, the Bonferroni correction suggests testing each individual hypothesis at a significance level of α/k [47]. For example, suppose we use 5-fold CV where $k = 5$ and set a desired type I error level to $\alpha = 0.05$. By applying the Bonferroni correction, an individual hypothesis testing can be conducted at the significance level of $0.05/5 = 0.01$. There is a chance that a subspace is tested significant in one-fold testing, but it is not in the rest data-fold testings. According to the Bonferroni correction approach, we need to regard this subspace significant as long as it is tested significant at least once. To prevent this happening and select strongly significant subspaces only, we add another criterion; the inclusion of this subspace must reduce prediction error on average relative to the model without this subspace. Different from Bo et al. [9], this criterion does not involve any tuning parameter as the threshold value is set to zero. In summary, there are two selection criteria for subspace selection based on hypothesis testing and prediction error, which will be referred to as double selection criteria.

Hypothesis tests

Suppose, at iteration t, $\tilde{\mathbf{z}}_t$ is randomly generated, and the corresponding subspace-based model, g_t, is learned. By the previous iteration, there are $j-1$ subspaces added into the base model, i.e.,

$$\text{Model 1}: \quad \hat{y} = \hat{f}(\mathbf{x}) = \hat{g}_1(\mathbf{z}_1) + \cdots + \hat{g}_{j-1}(\mathbf{z}_{j-1}). \tag{3.9}$$

If no subspace has been selected yet, we assume $\hat{y} = \bar{y}$. Now, by adding a temporary model learned from $\tilde{\mathbf{z}}_t$ to Eq. (3.9), we have

$$\text{Model 2}: \quad \hat{y} = \hat{f}(\mathbf{x}) = \hat{g}_1(\mathbf{z}_1) + \cdots + \hat{g}_{j-1}(\mathbf{z}_{j-1}) + \hat{g}_t(\tilde{\mathbf{z}}_t). \tag{3.10}$$

To evaluate the significance of the temporary subspace-based model, we use F statistic for comparing the two models in Eq. (3.9) and (3.10). By using Theorem 1 in [9], we can approximate the effective degrees of freedom for the two models as follows: Suppose a

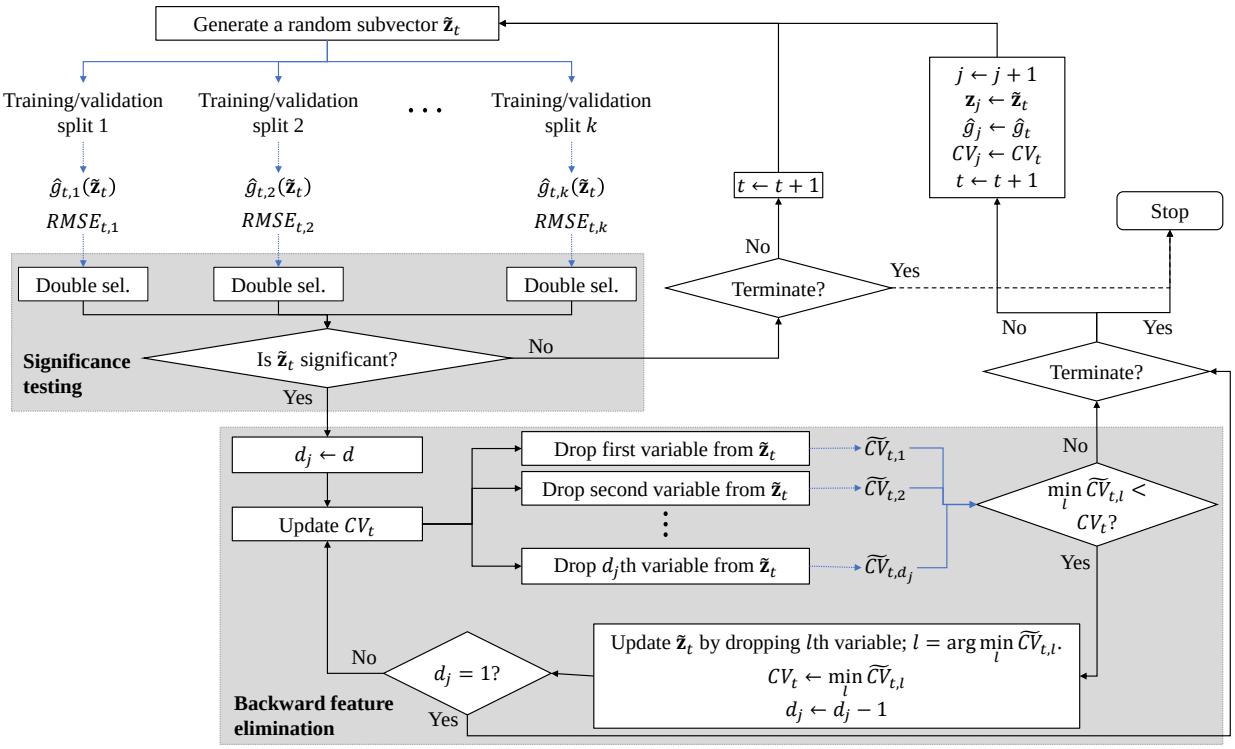

Figure 3.2: Schematic illustration of the process of feature subspace selection

squared loss function, i.e., $V_\epsilon(e) = e^2$, is used to learn subspace-based models. Let $\mathbf{S}'_j := (\mathbf{K}_j + \lambda\mathbf{I})^{-1}\mathbf{K}_j$ denote the linear projection matrix for the jth (selected) subspace-based model, $\hat{g}_j$. By defining $\mathbf{S}_j = \mathbf{S}'_j(\mathbf{I} - \sum_{l=0}^{j-1}\mathbf{S}_l)$ and $\mathbf{S}_0 = \mathbf{0}_{n\times n}$, the edf of Model 1 and Model 2, respectively, can be approximated as $\sum_{l=1}^{j-1}\mathrm{tr}(\mathbf{S}_l)$ and $\mathrm{tr}\{(\mathbf{I}-\mathbf{S}'_t)\sum_{l=1}^{j-1}\mathbf{S}_l\}+\mathrm{tr}(\mathbf{S}'_t)$.

Proof. See Appendix C. □

In Theorem 3.3.3, the expression $\mathbf{S}_j$ is recursive, so the calculation of the edf requires to store the matrix $\sum_{l=1}^{j-1}\mathbf{S}_l$ at all time and update it whenever a subspace is added to the base model. To simplify the computation and alleviate data storage, we further approximate the two models as $\hat{\mathbf{y}} = (\sum_{l=1}^{j-1}\mathbf{S}'_l)\mathbf{y}$ and $\hat{\mathbf{y}} = (\sum_{l=1}^{j-1}\mathbf{S}'_l + \mathbf{S}'_t)\mathbf{y}$ for the edf calculation purpose only. [9] discuss and show applying this approximation for hyperparameter learning produces more accurate predictions. With this approximation, we have By approximating Model 1 and Model 2 as $\hat{\mathbf{y}} = (\sum_{l=1}^{j-1}\mathbf{S}'_l)\mathbf{y}$ and $\hat{\mathbf{y}} = (\sum_{l=1}^{j-1}\mathbf{S}'_l + \mathbf{S}'_t)\mathbf{y}$, the edf of the two models can be calculated, respectively, as $\sum_{l=1}^{j-1}\mathrm{tr}(\mathbf{S}'_l)$ and $\sum_{l=1}^{j-1}\mathrm{tr}(\mathbf{S}'_l) + \mathrm{tr}(\mathbf{S}'_t)$. According to Proposition 3.3.3, for each addition of a subspace, we need to update only the trace of the linear projection matrix calculated in the corresponding iteration and store the sum of traces instead of storing the sum of matrices.

For general regression problems, the F statistic for comparing two models is defined by $F = \frac{(RSS_r - RSS_f)/(df_r - df_f)}{RSS_f/df_f}$ where RSS_r and RSS_f is the the residual sum of squares (RSS) of the reduced model (Model 1) and full model (Model 2), respectively, and df_r and df_f are the corresponding degrees of freedom. The RSS measures in-sample error, but as we apply CV, we consider out-of-sample error to form an F statistic while assuming prediction errors are normally distributed. In particular, we use the root mean square error (RMSE) for its usage in other places as well and define the F statistic based on Proposition 3.3.3 as

$$F = \frac{(RMSE_r^2 - RMSE_f^2)/\mathrm{tr}(\mathbf{S}'_t)}{RMSE_f^2/(n_V - \sum_{l=1}^{j-1}\mathrm{tr}(\mathbf{S}'_l) - \mathrm{tr}(\mathbf{S}'_t))}, \tag{3.11}$$

where RMSE_r and RMSE_f are the RMSE of the reduced model and full model, respectively. n_V is the size of validation dataset in the k-fold CV framework. By using this F statistic,

we calculate the p-value for the hypothesis testing as

$$p\text{-value} = P(F_{\text{tr}(\mathbf{S}'_t), n_V - \sum_{l=1}^{j-1} \text{tr}(\mathbf{S}'_l) - \text{tr}(\mathbf{S}'_t)} > F) \tag{3.12}$$

where $F_{\text{tr}(\mathbf{S}'_t), n_V - \sum_{l=1}^{j-1} \text{tr}(\mathbf{S}'_l) - \text{tr}(\mathbf{S}'_t)}$ is an F random variable with two degrees of freedom of $\text{tr}(\mathbf{S}'_t)$ and $n_V - \sum_{l=1}^{j-1} \text{tr}(\mathbf{S}'_l) - \text{tr}(\mathbf{S}'_t)$. If the p-value is less than α/k, we conclude the temporary subspace is significant for the current test.

Feature elimination and subspace selection

If a subspace is considered significant based on the double selection criteria, it is selected as a subspace candidate, and the features constructed from this subspace are individually evaluated for their significance. If a temporary subspace is not found significant, we proceed to the next iteration and generate another random subspace until the termination of the algorithm.

Since the subspace generation relies on a random search, a subspace candidate, albeit significant as a whole, may include insignificant features. Given a subspace candidate, we sequentially eliminate insignificant features, if any, by using an out-of-sample prediction error calculated via k-fold CV. We use RMSE to measure the prediction error and define the CV score of the subspace candidate, CV_t, as the average of k RMSEs. From the d-dimensional subspace candidate $\tilde{\mathbf{z}}_t$, features therein are dropped one by one, each forming a unique $(d-1)$-dimensional subvector. Each of such d subvectors is used to build a model replacing $\hat{g}_t(\tilde{\mathbf{z}}_t)$ in Eq. (3.10), and the corresponding CV score is calculated as $\widetilde{CV}_{t,l}$ for $l = 1, \ldots, d$ by taking the average of RMSEs calculated from the k data splits. If $\min_l \widetilde{CV}_{t,l} > CV_t$, i.e., dropping each feature does not improve prediction, the candidate subspace $\tilde{\mathbf{z}}_t$ is nominated as the jth critical subspace, $\mathbf{z}_j$. Otherwise, if $\min_l \widetilde{CV}_{t,l} < CV_t$, we drop the l^*th feature from $\tilde{\mathbf{z}}_t$ such that $l^* =_l \widetilde{CV}_{t,l}$, replace $\tilde{\mathbf{z}}_t$ by this reduced subset, and update $\hat{g}_t$ and CV_t accordingly. We repeat this process until $\min_l \widetilde{CV}_{t,l} > CV_t$ or the dimension of the (updated) candidate subspace becomes 1; see Figure 3.2 for the iterative procedure. Once $\mathbf{z}_j$ is determined, we update $\hat{g}_j$ and CV_j by the $\hat{g}_t$ and CV_t of the selected subspace, respectively.

3.3.4 Termination and overall algorithm

The FSS is an iterative process. At each iteration, we generate a random subspace, build a model using the random subspace (Section 3.3.2), determine whether to include the random subspace (Section 3.3.3), and eliminate insignificant feature(s) in the subspace (Section 3.3.3). The iterations continue until a termination criterion is met. For the termination criterion, different from [9] using a naive percentage reduction, we adopt the "absolute" percentage reduction of the CV score, which yields a faster and more effective convergence of the algorithm. The absolute percentage reduction is defined as

$$\Delta = \frac{|CV_{j-1} - CV_t|}{CV_{j-1}}. \tag{3.13}$$

Again, we assume we have $j-1$ subspaces selected at iteration t. The overall algorithm will terminate when $\Delta < \tau$ where τ is a termination threshold. In this paper, we set the τ at a higher value compared to [9] for a faster convergence. We use $\tau = 0.0001$; see Section 3.4.1 for more detail.

This entire learning process requires hyperparameter tuning for better model performance. Recall that subspace model learning requires an application of CV for subspace selection as described in Section 3.3.3, so we split a given dataset into multiple training and validation datasets. This allows evaluating the significance of each subspace based on out-of-sample prediction performance. For hyperparameter learning, on the other hand, we use the entire data and assess different hyperparameter settings with respect to in-sample errors calculated by generalized cross validation (GCV). This implies that each hyperparameter setting is evaluated once a full model is built, i.e., after all critical subspaces are selected and added into the full model. Given a vector of all hyperparameters, $\boldsymbol{\theta}$, the GCV is calculated according to

$$GCV(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(\mathbf{x}_i;\boldsymbol{\theta})}{1 - \sum_{j=1}^{J}\mathrm{tr}(\mathbf{S}'_j)/n)}\right]^2 \tag{3.14}$$

where $n = n_T + n_V$ is the size of the entire dataset. The step-by-step procedure is summarized in Alg. 2.

Algorithm 2: Learning process of the randomized subspace modeling

Split given data into k sets of training ($\mathcal{T}_q$) and validation ($\mathcal{V}_q$) to apply k-fold CV;
Generate a set of experiments testing hyperparameters;
for *each hyperparameter settings* **do**
    $t \leftarrow 0$; $j \leftarrow 0$;
    repeat
        $t \leftarrow t + 1$;
        Generate a random subspace $\tilde{\mathbf{z}}_t$;
        while $q \in \{1, \ldots, k\}$ **do**
            Use $\mathcal{T}_q$ and estimate $\hat{g}_t(\tilde{\mathbf{z}}_t)$ as discussed in Section 3.3.2;
            Predict the response for $\mathcal{V}_q$ and calculate an RMSE.
        end
        Apply subspace selection process described in Section 3.3.3;
        if $\tilde{\mathbf{z}}_t$ *is selected* **then**
            $j \leftarrow j + 1$; $\mathbf{z}_j \leftarrow \tilde{\mathbf{z}}_t$; $\hat{g}_j \leftarrow \hat{g}_t$;
        end
    until $\Delta < \tau$;
    Use the entire data ($\mathcal{T}_q \cup \mathcal{V}_q$ for any q) and build a full model in Eq. (3.1) by using the selected subspaces, $\mathbf{z}_j$ for $j = 1, \ldots, J$;
    Calculate GCV;
end
Determine the set of optimal hyperparameters minimizing GCV;

3.4 Comparison of feature interaction selection

In this section, we evaluate the feature interaction selection capability of FSS by using MONK data [18], a common benchmark dataset used in feature (interaction) selection. In MONK data, important and interactive features are known *a priori*, and the data generation process can be found in [60]. There are three datasets available: MONK 1, MONK 2, and MONK 3. MONK 1 and MONK 3 are suitable for this evaluation since they include all of relevant, interactive, and redundant features. However, in MONK 2, all features are important and interactive with each other, so the benefits of feature interaction selection for dimensionality reduction are limited. As such, we illustrate the selection results from MONK 1 and MONK 3 datasets only. Both datasets have 6 features and a single response which is a binary variable taking 0 or 1 for two class labels. They have 556 and 554 data points, respectively. Since the ground truth information of important features and interactions is given, there is no need

for a separate test dataset. Therefore, we use all available data points for model learning and feature interaction selection.

As can be inferred from the binary response, MONK data are designed for classification problems and hence used to compare methods developed for classification. Since our method is established for regression problems, some adjustment needs to be made to the original MONK datasets. In this regard, we create a new response y from the original one, y', as $y = 10y' + v$ where v is a normal random error with a mean of 0 and a standard deviation of 5. This variable transformation allows some overlaps of the response values between the two class labels preventing this regression task from being too trivial.

3.4.1 Experimental setup

When implementing FSS, we set the maximum dimensionality of subspaces to 2 as it provides better performance in terms of prediction errors for the MONK datasets. The significance level of multiple hypothesis testing is set to 0.05 ($\alpha = 0.05$), and the termination threshold to 0.01% ($\tau = 0.0001$), also based on preliminary experimental outcomes. Compared to Bo et al. [9], this termination threshold is relaxed to terminate the algorithm faster with a smaller number of iterations. For the base learner of SVR, we use the polynomial (POLY) kernel among various kernel choices available, as it provides a better fit to the hybrid material data that will be discussed in Section 3.5. For any feature vector $\mathbf{u}$ and $\mathbf{v}$, the POLY kernel is expressed as $K(\mathbf{u}, \mathbf{v}) = (\gamma \mathbf{u}^T \mathbf{v} + \delta)^{\text{deg}}$ where γ, δ, and deg denote the scale, offset, and degree of polynomials, respectively. For this kernel, we set $\gamma = 1/d_j$ (d_j is the dimension of each subspace) and $\delta = 0$ while we allow three options for $\text{deg} = 1, 2, 3$. Meanwhile, SVR learning itself involves hyperparameters of ϵ and $C := 1/\lambda$, and we consider five and three levels for these parameters, respectively, i.e., $\epsilon = 0.01, 0.05, 0.1, 0.2, 0.5$ and $C = 0.01, 0.1, 1, 10, 100$. We optimize the three parameters of deg, ϵ, and C by applying a grid search for 75 distinct level combinations of these parameters. In particular, we select a level combination that minimizes GCV in Eq. (3.14).

3.4.2 Experimental result

As stated earlier, we compare feature selection results of the proposed method with other benchmark methods. For the benchmark, we consider some linear methods, such as linear regression (LR), lasso regression, ridge regression, principal component regression (PCR), and partial least squares regression (PLS); and non-linear or nonparametric methods, including SVM, RF, k-nearest neighbors (k-NN), and ANN; and a feature interaction selection method, IWFS. Whenever hyperparameter tuning is needed for these methods, we apply grid search based on 5-fold CV so that we can obtain the best result from each method. We follow the exactly same setting used in [9], so please see [9] for more detail.

For MONK 1, the target features are $(a_1, a_2), a_5$ where parentheses represent interactive features. In other words, three features of a_1, a_2, and a_5 are significant out of six available and the interaction between a_1 and a_2 is also significant. In MONK 3, (a_2, a_5) and (a_4, a_5) are designed to be significant.

The feature selection results from all methods under consideration are shown in Table 3.1. For MONK 1, among all alternatives, RF, IWFS, RSM, and FSS identify the important features of a_1, a_2, and a_5 while others fail to determine the significant features correctly. Yet, RF searches for only important individual features without pointing out if any interaction is significant, and hence provides no information about important interactions. IWFS, RSM and FSS can identify important individual features and interactive features. For example, IWFS identifies the important and interactive features as (a_1, a_2, a_5), indicating that the three features selected are important with some significant interaction, but which among the three interact is not clearly stated. This is a natural consequence of its design; by construction, IWFS cannot extract multiple sets of important interactions, so it ends up producing a single set that includes all important features and interactions. On the other hand, RSM is capable of identifying multiple subsets of important interactions, but as it imposes a fixed common dimensionality on each subset, it cannot separately identify two sets of (a_1, a_2) and a_5 with different dimensionality. By construction, FSS can identify multiple subsets with different dimensionality, and as shown in the table, this is the only method that can correctly identify important features and interactions.

Table 3.1: Comparison of the feature selection results and running time for the MONK's problem

Method	MONK 1	MONK 3
LR	a_5, a_6	a_2, a_5
Lasso	a_5	a_2, a_5
Ridge	a_5, a_6	a_2, a_5
PCR	a_5	a_2, a_5
PLS	a_5	a_2, a_5
RF	a_1, a_2, a_5	a_2, a_5
k-NN	a_5	a_2, a_5
SVM (RBF)	a_5	a_2, a_5
SVM (POLY)	a_5	a_2, a_5
ANN	a_2, a_5	a_2, a_5
IWFS	$(a_1, a_2,\ a_5)$	(a_2, a_4, a_5)
RSM	$(a_1, a_2,\ a_5)$	(a_2, a_5), (a_4, a_5)
FSS	$(a_1, a_2), a_5$	(a_2, a_5), (a_4, a_5)
Ground Truth	(a_1, a_2), a_5	(a_2, a_5), (a_4, a_5)

The similar results can be observed from MONK 3 implementation. Here, all machine learning-based methods except RSM and FSS fail to identify all important features of a_2, a_4, and a_5 by missing a_4. This implies that a_4 may not be strongly significant on its own but is significant in the form of an interaction with a_5. Similar to the case of MONK 1, IWFS identifies a single subset that includes all important individual and interactive features, suggesting a_2, a_4, and a_5 are significant altogether. Obviously, this can cause some confusion as it implies a_2 and a_4 can be potentially interactive, which is not true. Now, for MONK 3, the two important interactions of (a_2, a_5) and (a_4, a_5) are with the same dimensionality which is two. Under this setting, RSM can identify the correct interactions as desired. Please note, however, in a real practical dataset, it is hard to expect all important interactions are defined in the same dimensionality. For this implementation as well, FSS can pinpoint the target features and interactions correctly.

In summary, IWFS can extract important and interactive features, but it cannot inform which features have a significant interaction. RSM is able to point out the truly important individual and interactive features when the interactions are defined on the same dimensionality. FSS does not need this prerequisite because it automatically adapts the dimensionality of each subspace to the given data by dropping insignificant features.

3.5 Case study: hybrid material experiments

In this section, we apply FSS to a hybrid material dataset for modeling the damage response behavior of the material. In particular, we use hybrid metal/composite structure data obtained from a finite element model; see [9] for more detail. This dataset includes 41 variables (features) determining the properties of the multi-layer hybrid structure and 5 variables (responses) describing the damage response of the structure. The 41 variables can be classified into three groups of Metal, Composite, and Interface, and each of them has 5, 31, and 6 features, respectively. Among the 5 response variables, one aggregates all the others presenting the overall damage response behavior. We use the aggregated response as the response of our model. There are 200 observations in the dataset.

Since the modeling involves predictions of the damage response, we first compare the prediction performance of FSS with other alternatives considered in Section 3.4 (excluding IWFS as it does not build a prediction model). Then, we use FSS to identify important features and interactions from the dataset. Through this process, we highlight the difference between FSS and RSM and investigate the effect of every change made over RSM, including changes in algorithm termination, subspace selection based on hypothesis testing, and backward feature elimination.

For the comparison of prediction performance, we apply 5-fold CV splitting the given dataset into 5 different pairs of training/validation and test datasets. Note, we use another layer of 5-fold CV for the selection of significant subspaces and hyperparameter tuning where we split the training/validation portion of the data into separate datasets for training and validation. The same treatment is applied to other methods. To be specific, while the testing portion of the data are not accessible in the phase of model learning, the remaining portion assigned for training and validation, say model learning data, will be used to train a model (training portion) and learn hyperparameters (validation portion) via 5-fold CV. For both hyperparameter tuning and testing error computation, we use root mean squared errors (RMSEs; Chai and Draxler [11]). On the other hand, when identifying significant features and interactions, we use the entire dataset for training and validation since there is no prior information about the true significance and hence there is no instance to test. For all these implementations, we set the subspace dimension for RSM and the maximum subspace dimension for FSS (d) to 3, and degree of polynomial is set to 1 based on preliminary experimental results.

3.5.1 Prediction performance

For the comparison of prediction performance, as there are some methods involving randomness, including RF, ANN, RSM, and FSS, we use 11 random seeds and compare the average results of these methods with the remaining others. RSM and FSS use random numbers to draw random subspaces, and each complete run of the two methods terminates within 25,000 subspace searches in all cases. To exclude any advantage these methods could take from having common subspaces over different random seeds and make the multi-seed

testing fully independent of each other, we let the 11 random seeds start from 1 with an increment of 25,000. For a fair comparison, we use the exact same seeds for RF and ANN.

Table 3.2 shows the comparative results of the prediction performance. In general, linear models perform better than nonlinear models for this particular dataset in terms of the average and standard deviation of the 5-fold CV errors. Among the linear models, PLS provides the lowest average prediction error, and Lasso and Ridge also produce decent outcomes. While the standard deviation of PLS and Ridge is similar, that of Lasso is noticeably larger. Although LR obtains the smallest variance, its predictability is relatively weaker than other linear methods.

For the nonlinear methods, SVM with POLY kernel and ANN achieve comparable performance to that of the linear counterpart. RSM produces the best performance among all methods in terms of predictability (except PLS) and the robustness of the estimator (associated with the standard deviation). Although the average prediction error of FSS is a little higher than RSM, it is still one of the best methods concerning both the average and standard deviation of the prediction errors. This observation is very interesting since we expected to see better predictions from FSS by eliminating insignificant features from randomly generated subspaces. One possible reason is that the preset dimensionality of subspaces is relatively low, i.e., three, so any functional evaluation in the subspaces is performed with dense data where an additional dimension, albeit noisy, can possibly improve the predictability. Yet, this observation is specific to the data considered in this paper, so further investigation with other datasets will be needed as a future study, especially for those with significant interactions in higher orders. Still, for the purpose of selecting important features and interactions, removing any insignificant feature from a randomly generated feature subset is definitely desired; we showed some examples earlier in Section 3.4.2 and further discuss this issue in later sections.

On the other hand, as FSS has a more relaxed termination criterion, it tends to terminate early with less exploration of random subspaces, which can also influence the prediction performance. Please note that, compared to RSM, FSS imposes more restrictive selection criteria and has an additional module for backward feature elimination, which in general will require more search iterations and more computational time. Despite of that, the relaxation

Table 3.2: Comparison of the average and standard deviation of RMSE's from 5-fold evaluation

Method	Average	Standard deviation
LR	12.61	1.05
Lasso	12.02	1.45
Ridge	11.98	1.20
PCR	12.21	1.64
PLS	11.89	1.15
RF	14.52	2.76
k-NN	15.27	3.51
SVM (RBF)	17.49	3.77
SVM (POLY)	12.01	1.20
ANN	12.57	1.44
RSM	11.95	0.99
FSS	**12.08**	**1.06**

of the termination criterion renders less iterations and smaller computational time for each complete run of the algorithm for a given hyperparameter setting, as presented in Table 3.3. Each value shown in the table is an averaged quantity over multiple hyperparameter settings. From the 11-seed testing, the overall average number of iterations decreases noticeably from 2822 to 522 by using FSS. In addition, the overall average computational time decreases from 78 seconds to 53 seconds, even with additional functionalities.

Table 3.4 shows more details of the four methods of RF, ANN, RSM, and FSS by presenting the individual seeds' results. The average of the mean prediction errors of RF is 14.52, and the average of the standard deviations is 2.76. RF does not provide comparably good predictions for any test instance, and all standard deviations are larger than 2. ANN has the average of the means as 12.57 and the average standard deviation as 1.44. The results of Test 7 and 10 present decent performance, but the prediction errors are generally higher than those of the best models in Table 3.2 while having one instance with very poor predictions (Test 3). For ANN, all standard deviations are greater than 1 with the highest standard deviation being 3.31 (Test 3). RSM and FSS produce decent predictions in all test instances while having the average of the mean errors of 11.95 and 12.08, respectively. They also achieve stable predictions having the average of the standard deviations as 0.99 and 1.06, respectively. It is worth mentioning that FSS performs better than RSM in 5 test instances out of 11 overall (Test 2, 6, 7, 9, 10).

This result demonstrates FSS provides decent and stable predictions when compared to other alternatives although RSM achieves slightly better performance. In addition, RSM and FSS effectively control the randomness opposed to RF and ANN with relatively poor prediction since the subspace selection criteria guarantee the quality of randomly generated subspaces. It is also shown that this result holds regardless of the random seeds.

3.5.2 Selection of important individual features

In the absence of information about truly important features and interactions for the hybrid material dataset, we first extract important individual features from FSS and compare them with a selection result of [3]. [3] analyze the exact same dataset and compute the total index, the total contribution of one input to the output variance to determine the significance of

Table 3.3: Comparison of computational performance between RSM and FSS, evaluated from 11 random seeds. The running time is in the unit of seconds.

Test		1	2	3	4	5	6	7	8	9	10	11	Avg.
Number of iterations	RSM	3154	2392	2638	2598	3237	2962	2877	3314	2706	2126	3037	2822
	FSS	622	403	723	480	538	420	529	447	478	631	466	522
Running time	RSM	87	63	74	75	95	81	81	88	72	57	82	78
	FSS	61	39	76	53	54	42	59	43	42	66	50	53

Table 3.4: The mean and standard deviation of 5-fold based RMSE's computed from different random seeds

Test		1	2	3	4	5	6	7	8	9	10	11	Avg.
RF	Mean	14.52	14.56	14.52	14.54	14.53	14.51	14.50	14.50	14.50	14.50	14.51	14.52
	S.D.	3.11	2.86	2.79	2.75	2.71	2.69	2.68	2.69	2.69	2.67	2.69	2.76
ANN	Mean	12.37	12.29	14.44	12.93	12.48	12.44	12.14	12.37	12.37	11.94	12.48	12.57
	S.D.	1.45	1.17	3.31	1.36	1.03	1.22	1.44	1.27	1.54	1.02	1.06	1.44
RSM	Mean	11.64	12.39	12.01	11.7	11.41	11.78	12.25	11.41	12.27	12.41	12.25	11.95
	S.D.	0.94	0.98	0.99	0.83	1.04	0.94	1.45	0.78	0.85	1.11	1.08	0.99
FSS	Mean	11.98	11.91	12.43	12.18	12.37	11.64	12.23	11.67	12.23	11.96	12.29	12.08
	S.D.	1.39	0.88	0.87	0.80	0.93	1.08	1.57	1.10	1.31	0.78	0.93	1.06

each individual feature; the result is shown in Table 3.5. From their analysis, they find only the first 4 parameters of x_1, x_4, x_3 and x_2 are significant for their reduced order model, and adding additional variables does not improve the model significantly.

Although FSS is capable of extracting important individual features if the final dimension of a subspace becomes one after feature elimination, it does not directly produce a list of important individual features by its construction. Therefore, in order to compare the selection result of important individual features, we extract common individual features observed in selected subspaces from all 5-fold CV iterations. For example, for x_1 to be an important individual feature, there must be at least one subspace including x_1 in every 5-fold CV iteration. As shown in Table 3.5, this extraction of common features yields a selection of 5 features: x_1, x_2, x_3, x_4, and x_7.

By comparison, FSS captures all critical features selected in [3] and excludes all those found insignificant excluding x_7. The fact that the two completely different methods ([3] focuses more on physical knowledge about the material and FSS is fully data-driven) show agreeable selection results implies that both methods are effective in capturing important individual features.

Although the significance of x_7 is still debatable, the advantages of FSS become evident when compared to the selection result of RSM. As shown in Table 3.6, RSM extracts 10 features among which 4 are found significant and 6 insignificant in [3]. Among the 6 potentially insignificant features, four of them (x_5, x_6, x_8, x_{12}) are selected as a part of subspaces in FSS but removed during the feature elimination process. It is likely that some of these features are just included along with other key features as a consequence of random subspace generation, but RSM fails to get rid of these features.

3.5.3 Selection of important feature interactions

In Section 3.5.2, we maintain 5-fold CV structure to analyze multiple (training and validation) datasets for the purpose of extracting important individual features. In this section, however, we use the entire data including 200 observations to identify critical feature interactions.

Table 3.5: Selection results of important individual features by using total index [3] and FSS

Features	Total index	Features selected in FSS
x_1	0.37	✓
x_4	0.13	✓
x_3	0.13	✓
x_2	0.11	✓
x_{10}	0.03	
x_5	0.03	
x_7	0.02	✓

Table 3.6: Important individual features selected by RSM and FSS

Method	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
RSM	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓
FSS	✓	✓	✓	✓			✓					

Fig. 3.3 illustrates the important feature interactions selected by FSS and compares them with those identified by RSM. We use a color code distinguishing features in different groups of Metal, Composite, and Interface to highlight the types of interactions, i.e., within group or between groups. In the figure, the differences between RSM and FSS are clearly presented. RSM identifies 11 critical subspaces for each of which has a constant dimensionality of 3. FSS considers 7 distinct subspaces significant, and the selected subspaces have different dimensionality.

From the FSS result, setting the maximum dimension of a subspace to 3 seems sufficient to capture important interactions in general as most subspaces are defined in less than 3 dimensions. By considering the selection result collectively with that shown in Table 3.5 for individual important features, three variables of x_1, x_4, and x_7 can be considered significant on their own. While x_2 and x_3 themselves are significant, their impact can further increase when interacting with other features. The features in two subspaces of (x_8, x_{10}, x_{20}) and (x_6, x_9) are not significant individually, but their interactions can influence the damage tolerance response. In summary, three individual features (x_1, x_4, and x_7), two 2-factor interactions ((x_2, x_{19}) and (x_6, x_9)), and two 3-factor interactions ((x_3, x_{11}, x_{41}) and (x_8, x_{10}, x_{20})) are found significant.

Regarding the three feature groups, FSS includes 3 key features from Metal group, 8 features (including 1 key feature) from Composite group, and two features (x_{20}, x_{41}) from Interface group. Considering that the 3 features in Metal group are also significant on their own, this result well aligns with the physical understanding of the damage tolerance behavior where Metal layer has the most significant impact followed by Composite with the minimum impact from Interface group [3].

On the contrary, the selection result of RSM is relatively hard to interpret. There are plenty of potentially insignificant features, and some of them need to be dropped. For example, the seventh subspace in RSM is (x_2, x_{19}, x_{29}), and the exact same subspace is evaluated in FSS as the first selected subspace. In FSS, the feature x_{29} is removed from the subspace after applying the feature elimination step. On the other hand, it is worth noting that RSM includes more subspaces in the final model. Whether each subspace is significant

	RSM			FSS		
1	x_{33}	x_4	x_{13}	x_2	x_{19}	
2	x_{12}	x_{19}	x_{14}	x_{41}	x_3	x_{11}
3	x_{12}	x_{16}	x_1	x_8	x_{10}	x_{20}
4	x_8	x_3	x_{15}	x_7		
5	x_{27}	x_7	x_1	x_4		
6	x_{28}	x_{19}	x_{18}	x_1		
7	x_{19}	x_2	x_{29}	x_9	x_6	
8	x_5	x_{14}	x_{10}			
9	x_{34}	x_3	x_{11}			
10	x_{23}	x_4	x_{15}			
11	x_9	x_2	x_3			

Metal | Composite | Interface

Figure 3.3: Critical feature interactions selected by RSM and FSS

as a whole is questionable, but it suggests that there can be more of potentially significant interactions.

There are two possible causes why FSS has a smaller number of subspaces selected. First, FSS uses double selection criteria along with feature elimination, imposing more restrictive selections of subspaces. Some subspaces selected in RSM may not be sufficiently significant under the FSS criteria. It is also possible that, by keeping unnecessary noises in the model (due to the constant dimensionality), RSM may need additional subspaces to offset the effects of the noises. The second possible cause is that the subspace exploration in FSS is not sufficient, especially with the relaxed termination criterion. However, even with more iterations of the algorithm, there is no guarantee that the subspace generation based on random draws covers enough portion of potential subspaces. To address this, ultimately, a better strategy for subspace generation will be needed; we leave this as a future study.

3.6 Conclusion

In this paper, we propose a feature subspace selection (FSS) approach envisioning its practical usage in many experiment-based studies for simultaneous identification of important features and interactions. The unique aspect of the proposed approach is the selection of multiple distinct subsets of features, represented by subspaces, which differentiates it from other relevant studies providing a single subset of features that includes important features and interactions altogether. This aspect endows a more detailed understanding of the system of interest by explicitly pinpointing which feature interacts with which among others selected. Although there are some wrapper-type methods that compute the strength of feature interactions in a pair-wise fashion, to the best of our knowledge, the proposed approach is among the first filter-type methods that also enable the identification of important interactions up to a desired order. The only exception is RSM in [9] for which potential feature subsets are randomly generated with fixed dimensionality. To improve the nature of random subset generation and allow flexible dimensionality in selected feature subsets, we leverage double selection criteria involving hypotheses tests and embed a feature

elimination module to remove any insignificant feature included in the model as a result of the random generation.

When applied to MONK data, common datasets used for evaluating feature selection capability, FSS is the only approach correctly capturing the important features and interactions among all alternatives considered. In a case study analyzing the damage tolerance of a hybrid material, FSS produces competitive predictions in comparison to other benchmark models. Although RSM performs slightly better in prediction, the advantages of FSS are clearly demonstrated in the selection of important individual features and important interactions. In particular, the selection result of FSS well aligns with that of [3] resulting from physics-oriented analysis, suggesting both methods are capable of identifying important features for this particular dataset.

For future works, FSS needs to be tested over a variety of datasets to further verify and test its prediction performance and feature (interaction) selection capability. In addition, there is a definite need for an improvement of the subspace exploration process, i.e., subspace generation. The current structure based on a pure random search is limited while lacking the coverage of the search space. It is desired to have a more strategic scheme that can improve the effectiveness and efficiency of the random search.

Chapter 4

XGBoost Evaluated Feature Selection

4.1 Introduction

Recent advancements in computing have intensified the focus on high-dimensional data analysis, crucial for dissecting complex systems like material development [57]. In material customization, an extensive array of material design parameters are evaluated to attain desired macro-behaviors. Lacking initial parameter significance knowledge, this typically involves numerous experiments to understand their causal impact on material responses [10]. Given the impracticality of exhaustive parameter space exploration, simpler methods like space filling are employed for experimental sequencing (Bang et al., 2018). Material development often entails data collection through physical tests or high-fidelity modeling, leading to datasets that are small in sample size yet high in dimensions. Efficient high-dimensional analysis is therefore essential for discerning causal effects.

High-dimensional datasets offer rich information, but their analysis is challenging due to the 'curse of dimensionality' [1]. As dimensions increase, data within the feature space becomes more disparate and sparse, limiting traditional data modeling techniques [19]. A common remedy is dimensionality reduction, which simplifies high-dimensional data into a more manageable form by removing extraneous and noisy data [26].

Dimensionality reduction encompasses two methods: feature extraction and feature selection. Feature extraction transforms the data into a lower-dimensional space, creating new features unlike the original physical ones, and limiting its use in experiments requiring control of physical features [30]. Conversely, feature selection evaluates and retains significant physical features, focusing experiments on these key parameters (Zebari et al., 2020). However, this method often overlooks critical feature interactions, omitting parameters that, although individually insignificant, are crucial collectively.

[9] introduced a novel approach with the randomized subspace-based model (RSM) and feature subspace selection (FSS), generating and evaluating low-dimensional feature subspaces. Unlike typical feature selection methods that derive a single significant feature subset, RSM and FSS identify multiple significant interactions. These two methods use an ensemble of models for each subspace to create a comprehensive model. This paper extends

their concept with feature selection, aiming to pinpoint significant physical features and their interactions.

Despite RSM's uniqueness, it has limitations. It enforces a fixed dimensionality on all subspaces, possibly including irrelevant features to meet this requirement. In Chapter 3, we proposes varying subspace dimensionality for a more realistic interaction representation. Additionally, RSM's reliance on prediction error reduction for subspace significance may inadvertently include insignificant subspaces. We address this by integrating statistical hypothesis tests with prediction criteria. Furthermore, we streamline the termination process, enhancing the algorithm's speed compared to RSM. To validate FSS's effectiveness, we apply it to a case study on the damage tolerance of layered composite/metal structures. The vast parameter space of these structures, combined with the impracticality of exhaustive testing, highlights the need for an efficient analysis method. FSS's application in this context demonstrates its capability to rapidly explore parameter spaces and optimize layered hybrid designs.

Although these two methods provide insights into important subspaces (features and interactions), the RSM and FSS search mechanism is inefficient. To improve the search mechanism, we consider XGBoost. The tree-based method is known for interaction selection. If we use the tree-based method, we can achieve model building and subspace selection at the same time. More details can be found in the methodology section.

4.2 Literature Review

The recent surge in handling high-dimensional datasets across various disciplines, including bioinformatics, chemistry, and speech signal analysis, has escalated the significance of dimensionality reduction methods [63, 56]. This process of reducing dataset complexity is primarily categorized into two methodologies: feature extraction and feature selection, each distinguished by their approach towards the dimensions they reduce.

Feature extraction, commonly associated with the essence of dimensionality reduction, entails an unsupervised learning approach. This method projects data from high-dimensional spaces to lower-dimensional spaces, focusing on minimizing the loss of critical information

[34]. However, this technique encounters interpretability challenges, particularly when it comes to the newly formed features post-transformation, which often do not correspond directly to the original physical variables. This interpretability issue limits the application of feature extraction in experimental research, where understanding the relationship between original variables is essential [63, 59].

Given these challenges, feature selection emerges as a viable alternative. This approach retains the original dataset features while reducing its dimensionality, making it particularly beneficial for analyzing complex scientific and engineering systems. The methodologies for feature selection are diverse, and primarily classified into three categories: filter methods, wrapper methods, and embedded methods. Filter methods utilize ranking criteria like the Pearson correlation coefficient and mutual information for selecting influential features and are celebrated for their straightforwardness and practicality [25, 7]. Wrapper methods, in contrast, systematically search through various feature combinations using a learning algorithm to identify the most effective subset, often yielding more precise results but with increased computational demands [69]. Embedded methods aim to merge the benefits of filter and wrapper methods by incorporating feature selection directly into the model training process, offering a balanced approach in terms of efficiency and effectiveness [12].

Despite the progress in feature selection, the challenge of detecting significant interactions between features remains a complex area [1]. [74] emphasizes the importance of identifying interacting features to enhance the predictive accuracy and comprehensibility of machine learning models, particularly in domains like computational biology, biomedicine, and web services. To address this, the authors propose INTERACT, an algorithm designed to select relevant features while implicitly exploring feature interactions. INTERACT employs a backward elimination method that removes features with low or no consistency contribution (c-contribution), a metric defined based on the inconsistency rate among data instances. The algorithm works by initially ranking features using symmetrical uncertainty and then evaluating each feature based on its c-contribution, removing those below a certain threshold. [36] proposed method, Interaction Gain-Recursive Feature Elimination (IG-RFE), is designed to evaluate the importance of features by considering both their relevance to the class label and their interaction with other features. The relevance is measured using symmetrical

uncertainty, while the interaction among features is assessed through normalized interaction gain. The IG-RFE method iteratively eliminates less important features based on these metrics. [65] introduces a novel feature selection method that emphasizes the analysis of relevance, redundancy, and interaction among features. The proposed method, termed Max-Relevance and Max-Interaction (MRMI), aims to select the most relevant and interactive features by using a "maximum of the maximum" approach. The MRMI algorithm's uniqueness lies in its ability to address relevance, redundancy, and interaction simultaneously, distinguishing it from other feature selection methods.

Innovative techniques such as the INTERACT algorithm, which uses information theory for feature ranking, and the Interaction Weight based Feature Selection (IWFS) algorithm have been developed to address this challenge. These methodologies aim to discern not only pertinent features but also their interactions within datasets [74, 73]. However, these approaches typically generate a singular subset of features, which may obscure detailed interactions between specific features.

To overcome these limitations, [9] have proposed a distinctive methodology that generates multiple feature subsets, each elucidating different feature interactions. This approach involves a randomized search for identifying potential feature subspaces, assessing their significance through prediction error analysis, and amalgamating these subspaces using a boosting technique. This process not only aids in better prediction and feature selection across various datasets but also in articulating a list of significant subspaces, or in other words, a catalog of crucial feature interactions. Nonetheless, this method has certain shortcomings in terms of the dimensionality of subspaces, the process of subspace selection, and the termination criteria of the algorithm. The forthcoming discussion will detail how our proposed Feature Subset Selection (FSS) method aims to address these challenges, highlighting the distinctions between FSS and the method proposed by [9].

4.3 Feature Subspace Selection by XGBoost

This section describes how we identify and determine important subspaces, including important features and interactions, by using XGBoost. We first build an XGBoost model

and extract all trees used for modeling response. Each tree is composed of a subset of features, and this subset naturally forms a subspace that captures interactions of relevant features. Given the list of trees (subspaces), we evaluate the impact of each tree on final prediction and rank all the trees based on the magnitude of their impact. Similar to the practice of determining significant principal components in PCA, we quantify the error reduction achieved by each tree starting from the one with the highest impact and generate a plot similar to scree plot (will be referred to as subspace scree plot hereafter) to determine significant subspaces. The detailed procedures are explained in the following sections.

4.3.1 XGBoost Model

XGBoost, introduced by [14], leverages a base learner of a regression tree, a great structure for capturing the information of feature interactions. While applying a gradient boosting scheme, XGBoost seeks a feature subset (tree) that is likely more important in modeling response, providing a more efficient way to explore feature subspaces compared to a random search applied in [9].

Suppose a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ with p features and a single response is given, i.e., $\mathbf{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}$. XGBoost integrates K additive functions in ensemble to predict the response as

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), f_k \in \mathcal{F} \tag{4.1}$$

where $\mathcal{F} = \{f(\mathbf{x}) = \omega_{q(\mathbf{x})}\}$ is a set of regression trees that maps the p-dimensional input to one of the leaves, i.e., $q : \mathbb{R}^p \to L$ where $L = \{1, 2, \ldots, T\}$ and T is the total number of leaves in a tree. $\omega \in \mathbb{R}^T$ is a weight vector associated with each leaf of the tree determining a predicted value for an input assigned to each leaf. Every f_k corresponds to a distinct tree structure q_k and its associated leaf weights w_k. To acquire the set of functions employed in the model, we minimize the following regularized objective.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \tag{4.2}$$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda||\omega||^2$. Here l is a differentiable convex loss function measuring the difference between the prediction $\hat{y}_i$ and the output y_i. The second term Ω penalizes the model's complexity, specifically the regression tree functions. The supplementary regularization term facilitates smoothing the final learned weights to prevent overfitting. In essence, the regularized objective leans towards selecting a model that employs simple yet predictive functions.

The tree ensemble model in Eq. (4.2) involves functions as parameters, making it unsuitable for optimization using traditional methods in Euclidean space. Instead, the model is trained using a boosting strategy. Formally, let $\hat{y}_i^{(t)}$ represent the prediction of the i-th instance at the t-th iteration; we introduce f_t to minimize the following objective.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t), \tag{4.3}$$

This implies that we systematically incorporate f_t, prioritizing the addition that maximally enhances our model based on Eq. (4.2).

In [9], once the subspace can meet the selection criteria (out-of-sample error and F Statistics), the new subspace is considered a critical subspace. The final model is trained in an additive manner, adding each critical subfunction (trained by each corresponding critical subspace). In XGBoost, the new subfunction f_t, improving the model most, is greedily added. Although the selection criteria are different, XGBoost shares the same idea of subspace generation and subspace selection. In [9], we use randomized search to generate subspace. In XGBoost, subspace is a single decision tree. For subspace selection, in [9], if a subspace meets the selection criteria, we add the subfunction (trained by the subspace) to the final model, while in XGBoost, the subfunction is greedily added, that is, the subfunction improving the model most is added. In addition, XGBoost adds regularization terms to the model to avoid overfitting.

The whole data is split into train, validation, and test datasets. We build an XGBoost model using the train dataset and apply a grid search for hyperparameter tuning. The out-of-sample error (validation dataset) is used to obtain the optimal hyperparameter. The final model for tree extraction is built using the optimal hyperparameter and train plus

validation dataset. All trees used in the model are recorded. These trees automatically include important features and interactions, called subspace in our method.

4.3.2 Significant Subspace Selection

By fitting an XGBoost model, we obtain the information of tree structures, i.e., q_k for $k = 1, \ldots, K$. Given the presence of p features, i.e., $\mathbf{x} = (x_1, \ldots, x_p)$, from each tree structure, we extract an index set of split variables $\mathcal{S}_k \subset \{1, \ldots, p\}$ such that $|\mathcal{S}_k| < p$ for $\forall k$. We accordingly define $\mathbf{z}_k \in \mathbb{R}^m$ for $m < p$ as a subvector of $\mathbf{x}$ that includes features associated with $\mathcal{S}_k$. The subvector $\mathbf{z}_k$ naturally spans a subspace of the p-dimensional feature space. In this section, we will quantify an importance score of each $\mathbf{z}_k$ and determine significant $\mathbf{z}_k$ among all sub-feature vectors selected by the XGBoost model.

We consider an XGBoost model in Eq. (4.1) using the entire feature set $\mathbf{x}$ as a baseline for computing the importance score. For $k = 1, \ldots, K$, we remove the features in $\mathbf{z}_k$ from the entire set $\mathbf{x}$ including p features and use these smaller number of features, say $\mathbf{x}^{-k}$, to fit an XGBoost model. We denote the prediction errors of the two models in RMSE as $e(\mathbf{x})$ and $e(\mathbf{x}^{-k})$ accordingly. Then, we define the importance score as

$$\rho_k = \frac{e(\mathbf{x}^{-k}) - e(\mathbf{x})}{e(\mathbf{x})}, \text{ for } \forall k \tag{4.4}$$

while measuring the magnitude of deterioration in prediction as a consequence of excluding features belonging to $\mathbf{z}_k$.

Once the importance scores are calculated, we arrange them in a non-increasing order and define a set of indices matching the order. In other words, we define a totally ordered index set $\mathcal{J} = \{j | \rho_j \geq \rho_{j'} \text{ for } j < j' \text{ and } j, j' \in \{1, \ldots, K\}\}$. Similar to the scree plot, commonly used in principal component analysis to determine the number of principal components to be kept in a model [78], we produce a subspace version of the scree plot based on the set $\mathcal{J}$. We refer to this as a subspace scree plot and use it to determine the number of subspaces to be kept in a model and thereby determine significant subspaces.

To generate a subspace scree plot, we consider an index set of features $\mathcal{R}$ initially set to a null set. Starting from $j = 1$, we iteratively update $\mathcal{R}$ as $\mathcal{R} \cup \mathcal{S}_j$ and construct a subvector

of features $\mathbf{z}_j$ according to the feature index set $\mathcal{R}$. We build an XGBoost model by using $\mathbf{z}_j$ and compute the prediction error in RMSE as $e(\mathbf{z}_j)$. We plot $e(\mathbf{z}_j)$ versus j while keep increasing j by 1.

Although this plot functions similarly to a general scree plot, the change in $e(\mathbf{z}_j)$ is not always monotone (decreasing) because we use an out-of-sample prediction error for determining the values of vertical axis of the plot, $e(\mathbf{z}_j)$. Therefore, to determine how many subspaces (associated with $\mathcal{S}_j$) to include in a model, an additional decision criterion is needed. By adopting the idea of the scree plot, we generate a cut where the prediction error stabilizes. In particular, we stop adding more subspaces if the inclusion of next 5 consecutive subspaces does not improve prediction. In other words, we let $\mathcal{S}_j$ be the last index set to include if $e(\mathbf{z}_j) \leq e(\mathbf{z}_{j'})$ for $\forall j' = j+1, \ldots, j+5$.

4.4 Case Study

In this section, we implement XGBoost on a dataset comprising a layered, hybrid material, aiming to predict its damage response behavior. The dataset, derived from a high-fidelity finite element model detailed in [9], encompasses data related to a hybrid metal/composite structure. It contains 41 variables (features) that characterize the properties of the multi-layer hybrid structure and 5 variables (responses) that describe the structure's damage response. These 41 features are grouped into three categories corresponding to the materials in the layered structure: Metal, Composite, and Interface, represented by 5, 24, and 12 features respectively. Our model's response is the overall damage response behavior of the structure. The dataset comprises 1500 observations.

The dataset is partitioned into training, validation, and testing sets. We construct an XGBoost model on the training set, employing grid search for hyperparameter optimization. For each model iteration, hyperparameters are fine-tuned. Preliminary trials for hyperparameter tuning indicated that increasing the tree count to 500 did not substantially improve out-of-sample error. Hence, to balance computational efficiency, we fix the tree count at 100. The validation dataset's out-of-sample error aids in deriving the optimal hyperparameters. The final model, for extracting trees, is developed using these optimal parameters and the

combined training and validation datasets. The model's trees, which inherently incorporate key features and interactions, are meticulously documented. These form the subspaces as per our methodology. The parameters under consideration are detailed in Table 4.1.

Since we set the number of trees to 100, there are 100 subspaces after XGBoost model building. The next step is to attain the importance score of each subspace. Here we use backward feature elimination. One subspace is dropped, that is, the corresponding features are not used for the XGBoost model building. The out-of-sample error is calculated. We sorted the 100 errors. The most critical subspace is the one that weakens the model most after dropping.

The importance scores of the subspaces enable the construction of a subspace scree plot, as depicted in Fig. 4.1. This plot, by sequentially adding subspaces from most to least important, highlights a plateau in out-of-sample error reduction, indicating no new feature inclusion beyond a certain point.

From Fig. 4.1, it is difficult to decide the cutting point. Therefore, the cutting criterion is needed. If the prediction error does not show a decreasing trend up to 5 times after a certain point, then we can cut at that point. The cutting point is at the 20th subspace for Fig. 4.1. If we cut at the 9th subspace, there are decreasing trend up to 7 times. The captured subspace using the cutting point at 20th subspace is shown in Table 4.2, where RMSE is the current RMSE by adding the new subspace. As we can see, the very important individual features, x_1, x_2, x_3, x_4 are included in the trees/subspaces, verifying the efficiency of our method.

4.5 Conclusion

In this research, we delved into the complexities of high-dimensional datasets, a prominent challenge in fields like bioinformatics and speech recognition. These datasets are often plagued by the curse of dimensionality, posing significant hurdles in data analysis and interpretation. Traditional approaches to mitigate these challenges typically involve dimensionality reduction and feature selection. However, this study introduced a novel and

Table 4.1: XGBoost Hyperparameter

Hyperparameter	Values
Number of trees	100
Learning rate	0.001,0.01,0.1
Column sampling	0.8, 1
Max depth	2, 3, 6

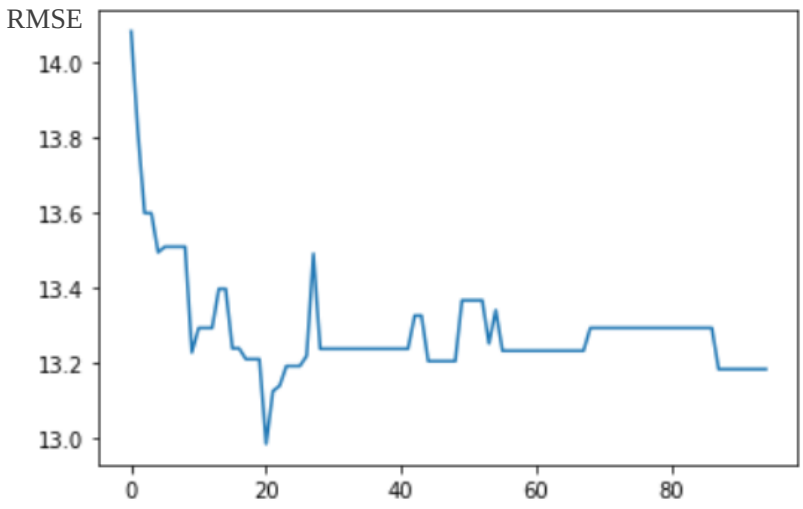

Figure 4.1: Subspace scree plot

Table 4.2: Critical Subspaces obtained by XGBoost Evaluated Feature Selection

	Subspace	RMSE
1	$x_4, x_1, x_2, x_{39}, x_3$	14.08
2	$x_4, x_1, x_2, x_{27}, x_7, x_5, x_3$	13.81
3	$x_4, x_1, x_2, x_9, x_{36}, x_3$	13.59
4	x_4, x_1, x_2, x_5, x_3	13.49
5	$x_4, x_1, x_2, x_{31}, x_{35}, x_3, x_{10}$	13.51
...	...	...
19	x_4, x_1, x_{19}, x_5	13.21
20	$x_4, x_1, x_{25}, x_7, x_{34}, x_{10}$	12.98

more effective method for feature selection using XGBoost (eXtreme Gradient Boosting), recognized for its power, scalability, and efficiency in tree-boosting systems.

Our approach capitalizes on the inherent structure of XGBoost's decision trees to adeptly identify and rank feature subspaces based on their criticality in predictive modeling. The process involved constructing an XGBoost model, extracting its decision trees, and then applying a backward elimination strategy. This strategy meticulously assessed the impact of each feature subspace's removal, with their significance determined by their influence on out-of-sample error. The resulting scree plot was instrumental in identifying the most relevant features, enhancing our understanding of data interactions and notably improving model performance.

The practical application of our method was demonstrated through a case study, underscoring its efficiency in handling high-dimensional data. Our findings not only provide a more interpretable framework for feature selection in complex datasets but also mark a significant advancement over traditional dimensionality reduction techniques.

In conclusion, this research presents a groundbreaking approach to feature selection, offering valuable insights into the interactions within high-dimensional datasets and paving the way for more effective and interpretable data analysis techniques. Our method stands as a testament to the potential of XGBoost in transforming the landscape of feature selection, promising far-reaching implications for various scientific and analytical domains.

Chapter 5

Conclusions

This dissertation has demonstrated an innovative approach to identifying feature interactions for high-dimensional datasets. The methodologies and algorithms presented here not only underscore the complexity of feature interactions but also provide a clear pathway for their effective identification and analysis.

In conclusion, this dissertation contributes to the field of high-dimensional datasets by providing robust tools and methodologies for feature interaction identification. This contribution not only enriches academic knowledge but also has the potential to inform practical applications, leading to more informed experimental design.

Future research can build upon this foundation, exploring XGBoost and DNN. The journey of discovery is never-ending, and this dissertation marks a significant step forward in understanding and harnessing the power of feature interaction.

Bibliography

[1] Al-Tashi, Q., Abdulkadir, S. J., Rais, H. M., Mirjalili, S., and Alhussian, H. (2020). Approaches to multi-objective feature selection: A systematic literature review. *IEEE Access*, 8:125076–125096. 2, 45, 46, 48, 49, 79, 81

[2] Alexandridis, A., Patrinos, P., Sarimveis, H., and Tsekouras, G. (2005). A two-stage evolutionary algorithm for variable selection in the development of rbf neural network models. *Chemometrics and Intelligent Laboratory Systems*, 75(2):149–162. 10

[3] Arndt, C., Crusenberry, C., Butler, R., and TerMaath, S. (2023). Reduced order modeling to characterize the damage tolerance of composite/metal structure. revision in review. x, 68, 71, 72, 74, 77

[4] Arndt, C., Heng, B., Butler, R., and TerMaath, S. (2022). Nondeterministic parameter space characterization of the damage tolerance of a composite/metal structure. 38

[5] Awad, M. and Khanna, R. (2015). Support vector regression. In *Efficient Learning Machines*, pages 67–80. Apress, Berkeley, CA. 16, 53

[6] Bang, H., Shi, Y. L. Z., Hoffman, G., Yoon, S.-Y., and Selva, D. (2018). Exploring the feature space to aid learning in design space exploration. In *International Conference on-Design Computing and Cognition*, pages 195–212. Springer. 45

[7] Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550. 10, 48, 81

[8] Belkina, A. C., Ciccolella, C. O., Anno, R., Halpert, R., Spidlen, J., and Snyder-Cappione, J. E. (2019). Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature Communications*, 10(1):1–12. 9

[9] Bo, D., Hwangbo, H., Sharma, V., Arndt, C., and TerMaath, S. (2023). A randomized subspace-based approach for dimensionality reduction and important variable selection.

Journal of Machine Learning Research, 24:1–30. 46, 49, 50, 51, 54, 55, 57, 59, 61, 62, 64, 76, 79, 82, 83, 84, 86, 110

[10] Bühlmann, P. and Van De Geer, S. (2011). *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media. 79

[11] Chai, T. and Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae). *Geoscientific Model Development Discussions*, 7(1):1525–1534. 65

[12] Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28. 10, 48, 81

[13] Chen, R.-C., Dewi, C., Huang, S.-W., and Caraka, R. E. (2020). Selecting critical features for data classification based on machine learning methods. *Journal of Big Data*, 7(1):1–26. 10, 23, 48, 49

[14] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. 3, 83

[15] Cheverud, J. M. (2001). A simple correction for multiple comparisons in interval mapping genome scans. *Heredity*, 87(1):52–58. 55

[16] Choi, S. W., Lee, C., Lee, J.-M., Park, J. H., and Lee, I.-B. (2005). Fault detection and identification of nonlinear processes based on kernel pca. *Chemometrics and Intelligent Laboratory Systems*, 75(1):55–67. 9

[17] Degenhardt, F., Seifert, S., and Szymczak, S. (2019). Evaluation of variable selection methods for random forests and omics data sets. *Briefings in Bioinformatics*, 20(2):492–503. 10

[18] Dua, D. and Graff, C. (2019). UCI machine learning repository. Last accessed 16 June 2022. 20, 60

[19] Fan, J. and Lv, J. (2010). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, 20(1):101. 79

[20] Fong, S., Zhuang, Y., Tang, R., Yang, X.-S., and Deb, S. (2013). Selecting optimal feature set in high-dimensional data by swarm search. *Journal of Applied Mathematics*, 2013. 6

[21] Fukunaga, K. (2013). *Introduction to Statistical Pattern Recognition*. Elsevier, San Diego, CA. 8

[22] Gisbrecht, A., Schulz, A., and Hammer, B. (2015). Parametric nonlinear dimensionality reduction using kernel t-sne. *Neurocomputing*, 147:71–82. 9

[23] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537. 21

[24] Gregorutti, B., Michel, B., and Saint-Pierre, P. (2017). Correlation and variable importance in random forests. *Statistics and Computing*, 27(3):659–678. 10

[25] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182. 48, 81

[26] Hasan, B. M. S. and Abdulazeez, A. M. (2021). A review of principal component analysis algorithm for dimensionality reduction. *Journal of Soft Computing and Data Mining*, 2(1):20–30. 79

[27] Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, New York, NY. 8, 17, 53, 104

[28] Heng, B. and TerMaath, S. C. (2018). Prediction of damage tolerance in metallic structure repaired with a co-cured composite patch. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0225, Kissimmee, FL. ix, 29, 31, 39

[29] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507. 9

[30] Jia, W., Sun, M., Lian, J., and Hou, S. (2022). Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693. 79

[31] Johnsson, T. (1992). A procedure for stepwise regression analysis. *Statistical Papers*, 33(1):21–29. 12

[32] Kursa, M. B. (2014). Robustness of random forest-based gene selection methods. *BMC Bioinformatics*, 15(1):1–8. 10

[33] Lee, J. A. and Verleysen, M. (2007). *Nonlinear Dimensionality Reduction*. Springer Science & Business Media, New York, NY. 9

[34] Li, M., Wang, H., Yang, L., Liang, Y., Shang, Z., and Wan, H. (2020). Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction. *Expert Systems with Applications*, 150:113277. 2, 45, 47, 81

[35] Li, Y., Shi, X., and Yao, L. (2016). Evaluating energy security of resource-poor economies: A modified principle component analysis approach. *Energy Economics*, 58:211–221. 8

[36] Lin, X., Li, C., Ren, W., Luo, X., and Qi, Y. (2019). A new feature selection method based on symmetrical uncertainty and interaction gain. *Computational biology and chemistry*, 83:107149. 81

[37] Liu, J. Z. (2019). Variable selection with rigorous uncertainty quantification using Bayesian deep neural networks. In *Bayesian Deep Learning Workshop at NeurIPS*, Vancouver, Canada. 6, 11, 23

[38] Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in Neural Information Processing Systems*, 26. 10

[39] Marill, T. and Green, D. (1963). On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9(1):11–17. 10

[40] McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. 9

[41] Mundra, P. A. and Rajapakse, J. C. (2009). Svm-rfe with mrmr filter for gene selection. *IEEE Transactions on Nanobioscience*, 9(1):31–37. 10

[42] Olden, J. D., Joy, M. K., and Death, R. G. (2004). An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling*, 178(3-4):389–397. 10, 11, 23, 48, 49

[43] Pearson, K. (1901). Liii. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572. 8

[44] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238. 10

[45] Pudil, P., Novovičová, J., and Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125. 10

[46] Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Kaluri, R., Rajput, D. S., Srivastava, G., and Baker, T. (2020). Analysis of dimensionality reduction techniques on big data. *IEEE Access*, 8:54776–54788. 6

[47] Rupert Jr, G. et al. (2012). *Simultaneous Statistical Inference*. Springer Science & Business Media. 55

[48] Saeed, N., Nam, H., Al-Naffouri, T. Y., and Alouini, M.-S. (2019). A state-of-the-art survey on multidimensional scaling-based localization techniques. *IEEE Communications Surveys & Tutorials*, 21(4):3565–3583. 8

[49] Salo, F., Nassif, A. B., and Essex, A. (2019). Dimensionality reduction with ig-pca and ensemble classifier for network intrusion detection. *Computer Networks*, 148:164–175. 8

[50] Saul, L. K., Weinberger, K. Q., Sha, F., Ham, J., and Lee, D. D. (2006). Spectral methods for dimensionality reduction. *Semi-supervised Learning*, 3. 9

[51] Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319. 9

[52] Sermpinis, G., Tsoukas, S., and Zhang, P. (2018). Modelling market implied ratings using lasso variable selection techniques. *Journal of Empirical Finance*, 48:19–35. 48

[53] Sharma, N. and Saroha, K. (2015). Study of dimension reduction methodologies in data mining. In *International Conference on Computing, Communication & Automation*, pages 133–137. IEEE. 2, 45

[54] Somol, P., Pudil, P., Novovičová, J., and Paclık, P. (1999). Adaptive floating search methods in feature selection. *Pattern Recognition Letters*, 20(11-13):1157–1163. 10

[55] Sompairac, N., Nazarov, P. V., Czerwinska, U., Cantini, L., Biton, A., Molkenov, A., Zhumadilov, Z., Barillot, E., Radvanyi, F., Gorban, A., et al. (2019). Independent component analysis for unraveling the complexity of cancer omics datasets. *International Journal of Molecular Sciences*, 20(18):4414. 8

[56] Sorzano, C. O. S., Vargas, J., and Montano, A. P. (2014). A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*. 47, 80

[57] Su, Y., Shi, Q., and Wei, W. (2017). Single cell proteomics in biomedicine: High-dimensional data acquisition, visualization, and analysis. *Proteomics*, 17(3-4):1600267. 79

[58] Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323. 9

[59] TerMaath, S. C. (2018). Probabilistic multi-scale damage tolerance modeling of composite patches for naval aluminum alloys. Technical Report DTIC AD1074284,

Department of Mechanical, Aerospace, and Biomedical Engineering, University of Tennessee, Knoxville, Tennessee. 9, 48, 81

[60] Thrun, S. B., Bala, J. W., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K. A., Dzeroski, S., Fisher, D. H., Fahlman, S. E., et al. (1991). The monk's problems: A performance comparison of different learning algorithms. Technical report, Department of Computer Science, The Carnegie Mellon University, Pittsburgh, Pennsylvania. 20, 21, 60

[61] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288. 48

[62] Torkkola, K. (2003). Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438. 10

[63] Van Der Maaten, L., Postma, E., and Van den Herik, J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10:1–41. 6, 8, 47, 48, 80, 81

[64] Wan, J., Chen, H., Yuan, Z., Li, T., Yang, X., and Sang, B. (2021). A novel hybrid feature selection method considering feature interaction in neighborhood rough set. *Knowledge-Based Systems*, 227:107167. 2, 45

[65] Wang, L., Jiang, S., and Jiang, S. (2021). A feature selection method via analysis of relevance, redundancy, and interaction. *Expert Systems with Applications*, 183:115365. 82

[66] Wei, P., Lu, Z., and Song, J. (2015). Variable importance analysis: A comprehensive review. *Reliability Engineering & System Safety*, 142:399–432. 6, 9

[67] Whitney, A. W. (1971). A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 100(9):1100–1103. 10

[68] Xu, Z., Liu, J., Luo, X., Yang, Z., Zhang, Y., Yuan, P., Tang, Y., and Zhang, T. (2019). Software defect prediction based on kernel pca and weighted extreme learning machine. *Information and Software Technology*, 106:182–200. 9

[69] Xue, B., Zhang, M., Browne, W. N., and Yao, X. (2015). A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626. 10, 48, 81

[70] Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. (2018). Using stacking to average Bayesian predictive distributions (with discussion). *Bayesian Analysis*, 13(3):917–1007. 12

[71] Yördem, O., Papila, M., and Menceloğlu, Y. Z. (2008). Effects of electrospinning parameters on polyacrylonitrile nanofiber diameter: An investigation by response surface methodology. *Materials & Design*, 29(1):34–44. 45

[72] Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., and Saeed, J. (2020). A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1(2):56–70. 2, 45

[73] Zeng, Z., Zhang, H., Zhang, R., and Yin, C. (2015). A novel feature selection method considering feature interaction. *Pattern Recognition*, 48(8):2656–2666. 49, 82

[74] Zhao, Z. and Liu, H. (2009). Searching for interacting features in subset selection. *Intelligent Data Analysis*, 13(2):207–228. 49, 81, 82

[75] Zhou, Z. and Hooker, G. (2021). Unbiased measurement of feature importance in tree-based methods. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–21. 10, 11

[76] Zhu, L., Basu, S., Jarrow, R. A., and Wells, M. T. (2020). High-dimensional estimation, basis assets, and the adaptive multi-factor model. *Quarterly Journal of Finance*, 10(04):2050017. 2, 6, 45

[77] Zhu, L., Sun, N., and Wells, M. T. (2022). Clustering structure of microstructure measures. *Applied Economics and Finance*, 9(1). 48

[78] Zhu, M. and Ghodsi, A. (2006). Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, 51(2):918–930. 85

[79] Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286. 8

Appendices

A Proof of Theorem 2.1

Proof. From Eq. (2.1), we define $\mathbf{S}_j\mathbf{y}$ for $j = 1, \ldots, J$ to represent $g_j(\mathbf{z}_{j,i})$ as a linear combination of y_i's so that we have

$$\hat{\mathbf{y}} = \mathbf{S}_1\mathbf{y} + \mathbf{S}_2\mathbf{y} + \cdots + \mathbf{S}_J\mathbf{y}. \tag{1}$$

Since each g_j estimates residuals from a model including up to the previous subspace-based model (g_{j-1}), we have $\tilde{y}_{j,i} = g_j(\mathbf{z}_{j,i}) + \tilde{\varepsilon}_{j,i}$ (as in Eq. (2.3)). For an SVR with a squared loss function, the estimation can be expressed as $\widehat{\tilde{\mathbf{y}}}_j = \mathbf{S}'_j\tilde{\mathbf{y}}_j = (\mathbf{K}_j + \lambda\mathbf{I})^{-1}\mathbf{K}_j\tilde{\mathbf{y}}_j$ (see Hastie et al. 27) where $\widehat{\tilde{\mathbf{y}}}_j$ is the estimate of $\tilde{\mathbf{y}}_j$, a vector including $\tilde{y}_{j,i}$ for $\forall i$, and $\{\mathbf{K}_j\}_{i,i'} = K_j(\mathbf{z}_{j,i}, \mathbf{z}_{j,i'})$ for $i, i' = 1, \ldots, n$ for a given kernel $\mathbf{K}_j$. Since $\tilde{\mathbf{y}}_j = \mathbf{y} - \mathbf{S}_1\mathbf{y} - \mathbf{S}_2\mathbf{y} - \cdots - \mathbf{S}_{j-1}\mathbf{y}$,

$$\mathbf{S}_j\mathbf{y} := \widehat{\tilde{\mathbf{y}}}_j = \mathbf{S}'_j\tilde{\mathbf{y}}_j = \mathbf{S}'_j(\mathbf{y} - \mathbf{S}_1\mathbf{y} - \mathbf{S}_2\mathbf{y} - \cdots - \mathbf{S}_{j-1}\mathbf{y}) = \mathbf{S}'_j(\mathbf{I} - \sum_{l=1}^{j-1}\mathbf{S}_l)\mathbf{y} = \mathbf{S}'_j(\mathbf{I} - \sum_{l=0}^{j-1}\mathbf{S}_l)\mathbf{y}. \tag{2}$$

Now, for $j = 1$, the response to estimate is $\tilde{\mathbf{y}}_1 = \mathbf{y}$, so we have

$$\mathbf{S}_1\mathbf{y} := \widehat{\tilde{\mathbf{y}}}_1 = \mathbf{S}'_1\tilde{\mathbf{y}}_j = \mathbf{S}'_1\mathbf{y} = \mathbf{S}'_1(\mathbf{I} - \mathbf{S}_0)\mathbf{y}. \tag{3}$$

where $\mathbf{S}_0$ is an $n \times n$ zero matrix. Therefore, the result holds for $\forall j = 1, \ldots, J$. □

B Sensitivity Analysis

In this section, we show the sensitivity of the proposed method to some key parameters used for model construction. As the method relies on random generations of subspaces, we test the prediction capability with respect to multiple random seeds. In addition, we investigate the impact of selection and termination thresholds, i.e., η and τ, that are used for subspace selection and algorithm termination, respectively. Finally, we compare the goodness-of-fit of the proposed model for different dimensionality of subspaces, k.

To test the randomness, another 10 random seeds with an increment of 25000 are used to generate the randomized subspaces. The increment of 25000 is employed to avoid any duplicated sequences of subspace generation and fully investigate the effect of randomness; note, an exhaustive search requires evaluating $\binom{41}{3} = 10660$ subspaces for this data set. The result is shown in the Table 1. The mean of average prediction errors is 11.95 (by including the initial result) and the mean of standard deviations is 0.99 (by including the initial result). Although the average prediction errors increase a little on average, their mean is still better than other alternatives except for PLS regression; please refer to Table 2.6. This indicates that the proposed method provides decent prediction in general regardless of the random seed. All individual prediction errors are comparable to those of the linear models. In fact, there are 2 cases (Tests 4 and 7) producing lower prediction errors than the one reported in Table 2.6 and another case (Test 3) that is comparable. The standard deviations of RMSEs are also comparable to the result shown in Table 2.6. There are three cases (Test 3, 7, and 8) achieving lower standard deviation.

The result of various choices of selection and termination thresholds is given in Table 2. The proposed method is sensitive to the selection threshold η to some degree, and in the worst case, the prediction accuracy is worse compared to some nonlinear methods. As such, a careful selection of η is needed for a decent model. This also implies that the selection threshold plays an important role in controlling the uncertainty caused by the random generation of subspaces; without careful selection, the uncertainty can dominate the benefit of the random search. Although the impact could be less significant, the termination

Table 1: The average and standard deviation of RMSE's from 5-fold evaluation using different random seeds

Test	1	2	3	4	5	6	7	8	9	10
Avg.	12.39	12.01	11.7	11.41	11.78	12.25	11.41	12.27	12.41	12.25
Std. dev.	0.98	0.99	0.83	1.04	0.94	1.45	0.78	0.85	1.11	1.08

Table 2: Sensitivity of the selection and termination thresholds

Selection			Termination		
η	Average	Std. dev.	τ	Average	Std. dev.
0.001	12.03	1.28	0.000001	11.79	0.87
0.005	12.34	1.16	0.000005	12.34	1.65
0.01	11.64	0.94	**0.000010**	11.64	0.94
0.05	12.24	0.82	0.000050	12.48	1.38
0.1	12.96	1.26	0.000100	12.03	1.14

threshold τ plays a similar role. We recommend to start with some values between 1×10^{-4} and 1×10^{-6} and adjust the value depending the characteristics of a given data set.

We also vary the subspace dimension, k, and assess the prediction of models formed based on different dimensionality of subspaces. Table 3 shows the result. As the subspace dimension increases, the average prediction error decreases up to $k = 3$ and then increases. This clearly shows that the current selection of $k = 3$ provides the best prediction for this data set. The change in the dimensionality seems to have less impact compared to other parameters tested. However, please notice from Table 3, when $k = 1$, the method is only capable of choosing individually important variables without any interaction between features, and this leads to the worst performance among all dimensionality tested.

Table 3: Sensitivity of the subspace dimensions

Subspace dimension (k)	Average	Std. dev.
1	12.28	0.84
2	12.23	1.5
3	11.64	0.94
4	11.94	1.37
5	12.06	1.24

C Proof of Theorem 3.3.3

Proof. According to Theorem 1 of [9], Model 1 can be expressed as $\hat{\mathbf{y}} = (\sum_{l=1}^{j-1} \mathbf{S}_l)\mathbf{y}$, and Model 2 can be written as $\hat{\mathbf{y}} = \{\sum_{l=1}^{j-1} \mathbf{S}_l + \mathbf{S}_t'(\mathbf{I} - \sum_{l=0}^{j-1} \mathbf{S}_l)\}\mathbf{y}$. Then, the edf of Model 1 is $\mathrm{tr}(\sum_{l=1}^{j-1} \mathbf{S}_l) = \sum_{l=1}^{j-1} \mathrm{tr}(\mathbf{S}_l)$. Now, for Model 2, the edf is

$$\begin{aligned} \mathrm{tr}\left(\sum_{l=1}^{j-1} \mathbf{S}_l + \mathbf{S}_t'(\mathbf{I} - \sum_{l=0}^{j-1} \mathbf{S}_l)\right) &= \mathrm{tr}\left(\sum_{l=1}^{j-1} \mathbf{S}_l + \mathbf{S}_t' - \mathbf{S}_t' \sum_{l=1}^{j-1} \mathbf{S}_l\right) \\ &= \mathrm{tr}\left((\mathbf{I} - \mathbf{S}_t') \sum_{l=1}^{j-1} \mathbf{S}_l + \mathbf{S}_t'\right) = \mathrm{tr}\left((\mathbf{I} - \mathbf{S}_t') \sum_{l=1}^{j-1} \mathbf{S}_l\right) + \mathrm{tr}(\mathbf{S}_t') \quad (4) \end{aligned}$$

□

Vita

Di Bo was born in China. In 2020 Di moved to Knoxville, TN, to pursue her Master's degree in Statistics & Data Science and Doctorate in Industrial Engineering at the University of Tennessee, Knoxville.