ISSN: 2366-7109

WORKING PAPERS OF AGRICULTURAL POLICY

AGRICULTURAL POLICY
WORKING PAPER SERIES
WP2021-02

Metamodeling: a useful tool for applying innovative simulation techniques in agricultural economics

Ding Jin

Department of Agricultural Economics
University of Kiel

The Agricultural Working Paper Series is published by the
Chair of Agricultural Policy at the University of Kiel.
The authors take the full responsibility for the content.

Ding Jin

Metamodeling: a useful tool for applying innovative simulation techniques in agricultural economics

Department of Agricultural Economics
University of Kiel

Kiel, 2021
WP2021-02
http://www.agrarpol.uni-kiel.de/de/publikationen/working-papers-of-agricultural-policy

About the authors:
Ding Jin is research assistant and doctoral candidate at the departement for agricultural policy at Insitute for Agricultural Economics of Kiel.
Corresponding author: djin@ae.uni-kiel.de

Abstract

Computational simulation models are widely used in the field of agricultural economics for a variety of tasks, particularly for evidence-based policy analysis. Despite the substantial and continuing growth of computing power and speed, the growing complexity together with the implicit nature of the simulation models, on the one hand, still lead to high computational costs in applying the models along with great difficulties in the parameter specification where data availability and parametrization constraints for empirical calibration problems are notably challenging. On the other hand, they also limit the uses of simulation models in many other aspects such as integration into other research frameworks like policy optimization coupled with uncertainty analysis. In this paper, we attempt to systematically and comprehensively introduce the metamodeling technique and investigate several metamodel types in terms of accuracy, computational time, variable importance and potential practical applications.

1. Introduction

Computational simulation models, such as partial or general equilibrium models or Agent Based Models (ABMs), are widely used in the field of agricultural economics for a variety of tasks, particularly for evidence-based policy analysis (Birur et al., 2007; Diao et al., 2012; Manson and Evans, 2007; Rasch et al., 2017). The demands for creating large-scale dynamic models with highly disaggregated sectors and regions or developing the existing models with more components such as land, water, energy and sustainability or linking economic and ecological models to construct extensively interlinked models or diversifying agent heterogeneity drive the increase in complexity of these models. Despite the substantial and continuing growth of computing power and speed, the growing complexity together with the implicit nature of the simulation models, on the one hand, still lead to high computational costs in applying the models along with great difficulties in the parameter specification where data availability and parametrization constraints for empirical calibration problems are notably challenging. On the other hand, they also limit the uses of simulation models in many other aspects such as integration into other research frameworks like policy optimization coupled with uncertainty analysis which is receiving increasing attention as Manski stated that exact predictions of policy outcomes are routine, while expressions of uncertainty are rare (Manski, 2018).

As a specific example, let's suppose that we want to maximize a welfare function $U(z)$ subject to θ where z is an output variable of an economic-ecological simulation model that is represented by $T(z, \theta) = 0$ and θ is an input parameter of the model that is assumed to have a direct impact on z. Without any further restrictions, deriving an analytical solution to the maximization problem is complicated or at least tedious and time-consuming because $T(z, \theta) = 0$ is an implict representation of the relationship between θ and z. Likewise, due to the same reason, finding solutions to problems mentioned above is tricky.

Researchers have been increasingly tapping into this gap. For example, Storm et al. have stated in their review paper that the Machine Learning (ML) methods have the potential to address both computational demands of complex simulation models and their calibration (Storm et al., 2020). Moreover, they have metioned that the metamodeling method offers opportunities to solve these problems. In the metamodeling literature, the ML methods are considered as an important element of the entire framework, which is called the metamodel (see Section 2). Therefore, in this paper, we are going to dive deep into the metamodeling method and explore its potentials from a practical point of view. Metamodeling is also known as response surface modeling or surrogate modeling (Kleijnen and Sargent, 2000). The method aims at approximating the mapping between inputs and outputs of the underlying simulation model with a simplified model which enables us to circumvent the simulation model's black-box nature and ease the computational cost. The

statistical approximation models are called metamodels (Blanning, 1975; Kleijnen, 1975) and the ultimate goal of metamodeling is to construct metamodels that can approximate the Input-Output (I/O) relationship with an acceptable level of precision so that they can be used to understand and develop the simulation models further or replace them for other research purposes. The method has been extensively used in other disciplines, such as engineering (Simpson et al., 1997; Barthelemy and Haftka, 1993; Jaroslaw and Raphael T, 1996) and natural science (Razavi et al., 2012; Gong et al., 2015; Mareš et al., 2016). In recent years, the method has drawn the interests of researchers from the economic field as well: Ruben and van Ruijven have applied the approach to bio-economic farm household models to analyze the potential impact of agrarian policies on changes in land use, sustainable resource management and farmers' welfare (Ruben and van Ruijven, 2001); Villa-Vialaneix et al. have compared eight metamodels for the simulation of N2O fluxes and N leaching from corn crops (Villa-Vialaneix et al., 2012); Yildizoglu et al. have applied two popular metamodel types to two well known economic models, Nelson and Winter's industrial dynamics model and Cournot oligopoly with learning firms, to conduct sensitivity analysis and optimization respectively (Yildizoglu et al., 2012).

With a metamodel being constructed to approximate the I/O relationship, we can replace the simulation model in the aforementioned optimization problem and derive an analytical solution. Analogously, it can help us solve other problems as well. However, to the author's knowledge, there have been no comprehensive metamodeling studies in the field of agricultural economics. Thus, in this paper, we attempt to systematically and comprehensively introduce the metamodeling technique and investigate several metamodel types in terms of accuracy, computational time, variable importance and potential extensions to demonstrate how to use them in practical applications and the features of various metamodel types as well as to explain how they can help us solve the aforementioned problems. The paper is organized hereafter as, Section 2: a general introduction of the metamodeling framework, Section 3: a comprehensive review of the metamodels that are considered in this study, Section 4: a brief introduction of the simulation model that is used in the paper and analysis of the results, Section 5: conclusion.

2. Framework of metamodeling

A simulation model is often developed to study a real problem for a specific purpose. Such a model, deterministic or stochastic, is applied to approximate the true I/O relationships.

In the field of agricultural economics, a good case in point is the Computable General Equilibrium (CGE) model and the corresponding CGE-based policy analysis. CGE models are broadly used as policy analysis instruments that can comprehensively represent the economy-wide effects of potential policy interventions (Henning et al., 2018). However, given the internal complexity, it is often viewed as a black-box (Böhringer et al.,

2003) that does not allow to identify relationships between input parameters[1] and output variables[2] explicitly. Besides, the implicit nature of model relationships and high computational intensity due to the growing complexity hinder the possibility of performing many tasks such as empircal calibration of large-scale dynamic models, integration into other research frameworks such as policy optimization coupled with uncertainty analysis and etc.

To overcome the limitation of applying simulation models, researchers have come up with the idea of producing a model that can approximate the I/O relationships of the underlying simulation model but has a more simplified form. Such a model is named as the metamodel (Blanning, 1975; Kleijnen, 1975).

Having a metamodel instead of a simulation model usually gives us several benefits (Barton, 2015; Simpson et al., 2001; Britz et al., 2009):

- Metamodels have a more simplified form that enables the users to better understand the behaviors of the simulation model;
- Metamodels can be integrated into other research frameworks such as policy optimization to solve more complex problems;
- Metamodels are faster to run and evaluate. Once they are constructed, we can easily utilize and manipulate them based on research purposes.

The method to produce metamodels is called metamodeling, also known as surrogate modeling or response surface modeling (Kleijnen and Sargent, 2000), which aims at approximating the mathmatical mapping between the inputs and outputs of the underlying simulation model.

To explain the metamodeling technique intuitively, let $(\mathbf{x}, \mathbf{y})$ represent the dataset that contains n pairs of (x_i, y_i) where $x_i = (x_i^1, ..., x_i^k)$ are the input parameters and y_i are the output variables. Thus, the simulation model is refered to as:

$$y_i = F^{sim}(x_i) \qquad i = 1, ..., n, \tag{1}$$

where F^{sim} represents the simulation model.

Furthermore, with x_i and y_i, we can fit a metamodel which can be formulated as:

$$\hat{y}_i = f^{meta}(x_i) \qquad i = 1, ..., n, \tag{2}$$

[1]In this paper, input parameter is also refered to as input, factor, or independent variable because it has different names in different fields. Therefore, we use them interchangeably.

[2]In this paper, output variable is also refered to as output, response, or dependent variable because it has different names in different fields. Therefore, we use them interchangeably.

where f^{meta} represents the metamodel that we utilize to approximate the I/O relationship of the underlying simulation model and $\hat{y}_i$ are the predicted values of the outputs using x_i.

Having introduced the general idea of metamodeling, we should dive into the three essential elements of this technique:

- Design of Experiments (DoE)[3];
- Metamodel;
- Validation.

DoE refers to the sampling method which we use to obtain the simulation sample (to get $\mathbf{x}$) for running the simulations (to get $\mathbf{y}$) because we need to train the metamodels before we employ them in practical applications. The training is performed with the help of the simulation data that are generated by running the simulation sample in the simulation model. The sample for simulation are produced by DoE which includes two categories: the classical experimental design and the space-filling experimental design (see Figure 1). The former places multiple sample points at the boundaries and the center of the parameter space to minimize the influence of the random errors from the stochastic simulation models. However, Sacks et al. have argued that this is not the case for deterministic simulation models where systematic errors prevail and therefore the space-filling experimental designs should be employed to replace the classical ones (Sacks et al., 1989). Among popular space-filling designs, such as the orthogonal arrays (Owen, 1992), uniform designs (Fang et al., 2000) and Latin Hypercube designs (Sacks et al., 1989), Latin Hypercube design enjoys great popularity due to its ability to generate uniformly distributed sample points with ideal coverage of the parameter space as well as the flexibility with the number of the sample points (Sacks et al., 1989).

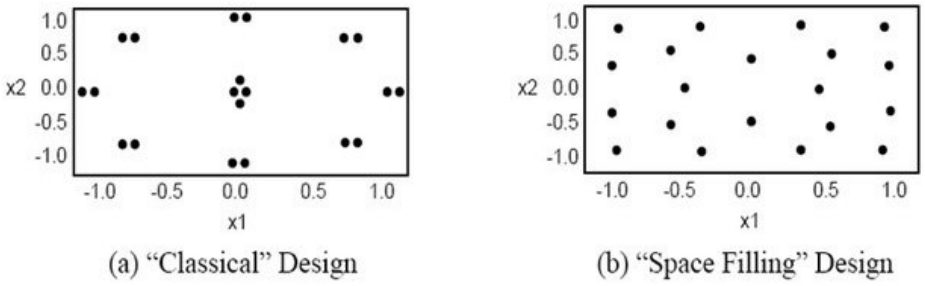

Figure 1: Classical and Space-filling Design. (Simpson et al., 2001)

[3]DoE is a statistical method of drawing samples in computer experiments (Dey et al., 2017).

Metamodel refers to the statistical method that we use to approximate the I/O relationships of the underlying simulation model. In the next section, we will give a brief introduction of the metamodel types that are selected and compared in this paper.

The process of constructing a metamodel can be called the training process. Moreover, in the training process of certain metamodel types (see Section 3 for details), a tuning process is necessary to adjust the hyper-parameters in the pursuit of optimal predictive ability. More specifically, for some metamodel types, there exist some parameters that affect how the predictions are made but cannot be estimated. These parameters are refered to as hyper-parameters. Practically, a k-fold cross validation approach is applied to tune the hyper-parameters which are selected on a grid search. It works in this way: the training sample is randomly partitioned into k folds (subsets); $k-1$ folds are used to train the metamodel and the other fold is used to make predictions; the averaged prediction accuracy is used to determine the best hyper-parameters on a grid search.

Validation refers to the evaluation criteria such as prediction accuracy and others in order to compare the performance of various metamodels and determine if the metamodels hold an acceptable level of quality. To assess the quality of a trained metamodel, we often make use of two samples: the training sample and the test sample. The training sample includes the sample points that are used to train the metamodel whereas the test sample contains sample points that are used to validate the metamodel that has been trained. An important difference is that there are sample points in the test sample that the training sample doesn't include. The importance of having a test sample is that we will use the metamodel as the replacement of the simulation model and therefore it is necessary to evaluate the generalization quality of it. In other words, we not only want the metamodel to have good predictions on the training sample but also want it to be able to predict values close to real values even for points which are not in the training sample. The metrics that we use to assess the validation results are summarized in Section 4.

3. Review of Metamodels

In the literature, there is a variety of metamodel types, see Dey et al. 2017; Simpson et al. 2001; Chen et al. 2006 for nice reviews. Based on the popularity in the practical applications, we have selected six types to assess and compare in terms of prediction performances and other aspects. We will provide a brief description of the metamodels studied in this section: polynomial model (Forrester et al., 2008), Kriging model (Cressie, 1993), multivariate adaptive regression splines (Friedman et al., 1991), support vector regression (Vapnik, 2013), random forest regression (Breiman, 2001) and artificial neural network (Smith, 1993). The polynomial model is a parametric model (Myers et al., 2016) that has explicit structure and specification whereas the others are non-parametric models (Yildizoglu et al., 2012; Kleijnen, 2015) that do not depend on assumptions in terms of

model specification and determine the I/O relationship of the underlying simulation model using experimental data.

3.1. Polynomial Models

The polynomial models are the most basic metamodels. In this paper, we introduce polynomial models that have specific orders which are determined by the user. As a specific example, a second-order polynomial model is given as follows:

$$y = \beta_0 + \sum_{h=1}^{k} \beta_h x_h + \sum_{h=1}^{k} \sum_{g \geq h}^{k} \beta_{h,g} x_h x_g + \epsilon, \tag{3}$$

where $x_1, ..., x_k$ are the k factors of the model, y is the response and ϵ is the error term. The coefficients β are usually derived through a linear regression based on least squares estimation. The main reasons for the popularity of polynomial models in comparison to the non-parametric metamodels are:

- Polynomial models have simple forms, which are easy to understand and manipulate;
- Polynomial models require low computational efforts;
- Polynomial models can be easily integrated into other research frameworks.

In this paper, we consider two polynomial models: a first-order polynomial model, recorded as *LM1*, and a second-order polynomial model, recorded as *LM2*. In practical applications, there are cases where even higher orders are used in the model when analysts have good reasons to do so. In the current context, we will not go into that direction.

In the determination of the form for *LM1* and *LM2*, we rely on expert opinions to select the input parameters (see Section 4.2 for details). As for *LM1*, we include all the main effects of the input parameters. As for *LM2*, we include all the main effects, a subset of all the two-way interaction effects, and all the quadratic effects. Moreover, we perform a preliminary stepwise selection based on the AIC criterion (Akaike, 1974) to screen the terms of *LM2* in order to select an optimal subset of covariates in the training process to avoid the potential overfitting problem[4]. There are various programming languages with the built-in functions to perform this selection process. In this paper, we use the "stepAIC" function from the R package MASS (Venables and Ripley, 2002).

In the construction of *LM1* and *LM2*, there are no hyper-parameters that we need to play around in the tuning process.

[4]In statistics, overfitting is the production of a model that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably.

3.2. Kriging Models

The Kriging model is based on the pioneering work of Danie Krige and introduced as a metamodel for deterministic simulation models in Sacks et al. 1989.

In this paper, we study two Kriging models in the context of deterministic simulation models: the Ordinary Kriging, recorded as *OK*, and the Universal Kriging, recorded as *UK*. For readers who are interested in the Stochastic Kriging for random simulation models, we refer to Kleijnen 2015. As a specific example, a Universal Kriging has the following form:

$$y = f(x) + M(x) \tag{4}$$

where x and y represent the factors and response of the model, $f(x)$ represents the assumed global trend and $M(x)$ is a stochastic process that refers to the localized deviations from the global trend[5]. $M(x)$ is assumed to be a weakly stationary process with zero mean and covariance matrix $\Sigma = \tau^2 R$ where τ^2 is the process variance and R is the correlation matrix whose (i, j) element is the correlation between points x_i and x_j[6], namely, $R_{ij} = Corr[M(x_i), M(x_j)]$. In Kriging, the correlations are determined by the distances between the points, that means, the closer the points x_i and x_j are to each other, the higher the correlation between them is. This idea is represented by the following correlation function which computes the correlation of points x_i and x_j using a Gaussian kernel[7]:

$$Corr[M(x_i), M(x_j)] = exp(-\frac{1}{2}\sum_{h=1}^{k}\frac{1}{\psi_h^2}(x_{i,h} - x_{j,h})^2), \tag{5}$$

where h represents the h^{th} factor of each point and ψ_h quantifies the relative importance of this factor meaning that a higher ψ_h represents a higher contribution of factor x_h to the correlation between the points, in other words, a higher importance of factor x_h to the response.

The Kriging models use a linear predictor and predict the new point x_0 as a linear function of the n old points.

$$\hat{y}_{x_0} = \sum_{i=1}^{n}\lambda_i y_i, \tag{6}$$

where $y_i = F^{SIM}(x_i)$ is the simulation response of the i^{th} old point x_i and λ_i refers to the weight of it in the prediction. The Kriging is often called a spatial estimator because λ_i decreases as the distance between the new point x_0 and the old point x_i increases. To determine the optimal weights λ_i^*, the Kriging uses the best linear unbiased predictor

[5]Figure 2 shows an Ordinary Kriging which has a constant trend.

[6]x_i and x_j refer to the i^{th} and j^{th} points in the simulation sample respectively.

[7]There are also other kernel functions that we can use in the computation of correlations, such as linear, exponential, etc (Kleijnen, 2015).

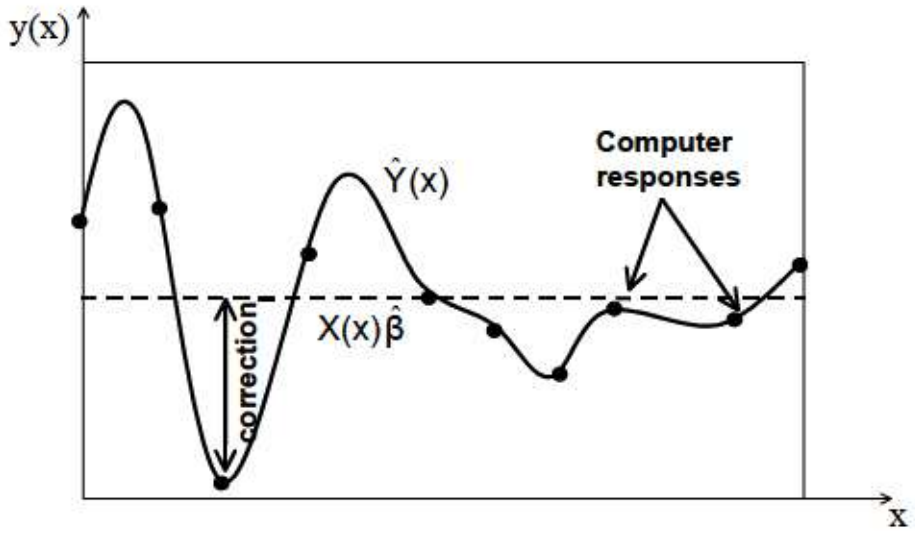

Figure 2: An Ordinary Kriging model (Jourdan, 2005).

(BLUP) as a criterion which minimizes the mean squared error of the predictor:

$$min \quad MSE[\hat{y}_{x_0}] = min \quad E[\hat{y}_{x_0} - y(x_0)]^2. \tag{7}$$

Following the derivations in Kleijnen 2015, we can obtain:

$$\hat{y}_{x_0} = f(x_0) + \sigma(x_0)^\intercal \Sigma^{-1}(y - f(x_0)), \tag{8}$$

where we have unknown parameters β (in the trend function) estimated via GLS (Kleijnen, 2015) as well as ψ and τ^2 that are estimated using the maximum likelihood method:

$$\begin{aligned} l(\mu, \tau^2, \psi) = \quad & -\ln[(2\pi)^{n/2}] \\ & -\tfrac{1}{2}\ln[\det\Big(\tau^2 R(\psi)\Big)] - \tfrac{1}{2}(y - f(x)1)^\intercal [\tau^2 R(\psi)]^{-1}(y - f(x)) \quad with \quad \psi \geq 0, \end{aligned} \tag{9}$$

where det refers to the determination of a matrix.

Some of the advantages and disadvantages of the Kriging model can be listed as:

- In comparison to polynomial models, Kriging models can handle nonlinear and irregular I/O relationships much better;
- Kriging models are by nature exact interpolators and therefore can predict the training sample with certainty;
- A potential disadvantage of the Kriging models is the implementation which is more time-consuming due to the optimization difficulty in the maximum likelihood estimation of the parameters (Kleijnen, 2015).

In this paper, we implement the Kriging model with the help of the R packages

DiceDesign and DiceEval (Dupuy et al., 2015). In addition, we want to mention the "nugget" effect (see Kleijnen 2015 for detailed explanations) which is essentially an error term added to Equation (4). Adding the "nugget" effect often reduces the numerical problems in the estimation of the Kriging models and makes them smooth but lose the interpolating property.

In the construction of *OK* and *UK*, there are two hyper-parameters to adjust in the tuning process:

- The kernel function that determines how the correlations are computed;
- The optimization method that is used in the estimation of ψ and τ^2.

The kernel functions and optimization methods are summarised in Table 8 in the appendix. For the details of them, we refer to Kleijnen 2015 and the accompanying documentations of the R packages DiceDesign and DiceEval (Dupuy et al., 2015).

3.3. Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines, recorded as *MARS*, was first developed by Friedman (Friedman et al., 1991). It is a nonparametric statistical method that approximates the I/O relationships of the underlying simulation model by adaptively selecting a set of basis functions. A *MARS* model has the following form:

$$y = \sum_{m=1}^{M} a_m B_m(x), \tag{10}$$

where a_m is the coefficient of the basis function $B_m(x)$ and $B_m(x) = 1$ when $m = 1$, namely, the first basis function is a constant. Basis functions can be written as follows:

$$B_m(x) = \prod_{p=1}^{P_m} [s_{p,m}(x_{v(p,m)} - t_{p,m})]_+^q \tag{11}$$

where P_m is the number of interaction order in the m^{th} basis function, $s_{p,m} = \pm 1$, $x_{v(p,m)}$ is the v^{th} factor, $1 \leq v(p,m) \leq k$ and $t_{p,m}$ is a knot location corresponding to $x_{v(p,m)}$. The subscript "+" means the function is a truncated power function that is often called a hinge function:

$$[s_{p,m}(x_{v(p,m)} - t_{p,m})]_+^q = \begin{cases} [s_{p,m}(x_{v(p,m)} - t_{p,m})]^q, & s_{p,m}(x_{v(p,m)-t_{p,m}}) > 0 \\ 0, & \text{otherwise.} \end{cases}$$

In the construction of a *MARS* model, the algorithm performs a forward selection and a backward pruning. In the forward selection, the algorithm recursively splits the training sample by adding a pair of hinge functions, which maximize the reduction of

the residual sum of squares. To be more specific, the algorithm iteratively searches for a proper knot location for a factor and splits the training sample at this knot location using hinge functions, which yields a regression equation for each of the two subsets of the training sample. The pair of hinge functions can enter the model directly or they can be multiplied by an existing basis function that is already in the model and become new basis functions. This forward selection stops when a predefined maximum number of basis functions is reached or if the reduction of the residual sum of squares is less than 0.001. In the backward pruning, a generalized cross validation (GCV) approach (Craven and Wahba, 1978) is applied to remove the basis functions which have the least contribution to the reduction of residual sum of squares. The purpose of the backward pruning is to reduce the complexity of the *MARS* model and avoid the potential overfitting problem.

In the paper, the *MARS* is produced with the help of the R packages "earth" and "caret" (Milborrow., 2020; Kuhn, 2020). Additionally, in the construction of *MARS*, there are two hyper-parameters to adjust in the tuning process:

- *degree*: the maximum order of interaction in the basis functions;
- *nprune*: the predefined maximum number of basis functions to retain in the model.

As described in section 4.3, a k-fold cross validation approach is applied to tune the two hyper-parameters *degree* and *nprune* which are selected on a grid search (see Table 9 in the appendix).

3.4. Suppport Vector Regression

Support vector regression, recorded as *SVR*, is an extension of the support vector machine (Boser et al., 1992) which was originally developed to address the classification problems. Vapnik extended the idea to regression problems (Vapnik, 1995).

A *SVR* model has the following form:

$$y = wx + b, \tag{12}$$

where x and y are the factors and response of the model while w and b represent the coefficients. In the literature, there are several methods that have been proposed to determine the coefficients (Steinwart and Christmann, 2008). In this paper, we still follow the original approach that utilizes the $\epsilon - insensitive$ loss function to determine the coefficients for the model:

$$L_\epsilon = \sum_{i=1}^{n} \max(|\hat{y}_i - y_i| - \epsilon, 0), \tag{13}$$

where $\hat{y}_i$ refer to the predicted values. Such a loss function pays no attention to errors which are smaller than ϵ and deems them as acceptable while focuses only on the errors larger than ϵ.

To determine the coefficients, the algorithm minimizes the loss function L_ϵ while penalizes the complexity of the estimated function, which is reflected as follows:

$$\underset{w,b}{arg\,min}\;\; L_\epsilon + \frac{1}{C}\|\mathbf{w}\|^2, \tag{14}$$

where $\|\mathbf{w}\|^2$ represents the regularization term that controls the complexity of the estimated function and C is the regularization parameter. Parameter C regulates the tolerance of the points which lie outside the accepted error margin. If C decreases, it means that we give more tolerance to the points outside because we allow for bigger errors in exchange for less complex estimated model. On the contrary, if C increases, it means that we accept more complexity of the estimated model so as to avoid bigger errors. It is worth mentioning that if we allow for high complexity, the model can usually make precise predictions on the training sample but poor predictions on a new sample (a typical overfitting problem). Thus, it is always a compromise that we have to make between the complexity of the model and the tolerance of the errors.

In the computation of w and b, Vapnik has derived the solution using the Lagrangian and Karush-Kuhn-Tuchker conditions (Vapnik, 1995) and Keerthi et al. have demonstrated that it is a classical quadratic optimization problem that can be explicitly solved (Keerthi et al., 2001). For readers who are interested in the details, we refer to these two papers.

In the paper, the *SVR* is produced with the help of the R package "e1071" (Meyer et al., 2020). Additionally, in the construction of *SVR*, there are three hyper-parameters to adjust in the tuning process:

- The ϵ of the loss function;
- The regularization parameter C;
- The γ parameter of the Gaussian kernel used in solving the optimization problem (Keerthi et al., 2001).

As described in section 4.3, a k-fold cross validation approach is applied to tune the hyper-parameters ϵ, C and γ which are selected on a grid search (see Table 10 in the appendix).

3.5. Random Forest

Random forest regression, recorded as *RF*, is developed based on two methods: classification and regression tree (Breiman et al., 1984), also known as *CART*, and bootstrap

aggregating regression trees (Breiman, 1996), also known as *Bagging*.

CART constructs a basic regression tree (see Figure 3) by partitioning a training sample into smaller subsets and fitting a constant for each subset. The partitioning is achieved by successive binary partitions based on the different predictors. The algorithm begins with the entire training sample, say S, and searches every distinct value of factors to find the predictor and its split value that partitions S into two subsets, R_1 and R_2, such that the overall sum of squared errors is minimized:

$$min \;\; SSE = \sum_{i \in R_1} (y_i - c_1)^2 + \sum_{i \in R_2} (y_i - c_2)^2 \tag{15}$$

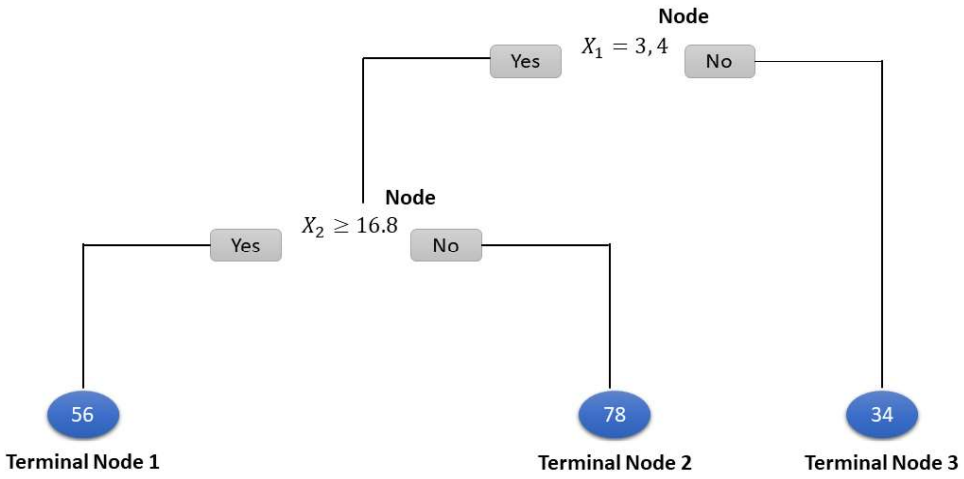

Figure 3: The structure of a basic regression tree with three terminal nodes.

The partitioning is carried out from top to bottom and any partitioning will not affect the previous partitioning. Having found the first split, the process will be conducted on each of two resulting subsets and it will continue until some stopping criteria is reached. Thus, a basic regression tree with multiple nodes[8] is constructed. Then the prediction of a new point is made by assessing it at each node and proceeding to the terminal nodes[9] which contain the predicted values. Besides, in order not to have a complex regression tree that is likely to lead to an overfitting problem, there is a pruning process that allows us to control the complexity by making trade-offs between complexity and prediction accuracy. For readers who are interested in the pruning process, we refer to Breiman et al. 1984. The advantage of *CART* is that it generates basic regression trees which are easy to use

[8] A node is represented by a predictor and its split value.
[9] A terminal node is where the fitted constant locates.

and interpret while the disadvantage of it is that these basic regression trees have high variances which could lead to unstable predictions. Because of this, it leads us to an upgrade of the basic regression trees, namely, *Bagging*.

Basically, *Bagging* combines and averages the predictions across multiple basic regression trees[10]. Averaging across multiple trees reduces the variability of any single tree and overfitting, which improves predictive performance. In practice, *Bagging* is performed in the following process:

1. Creating a certain number of bootstrap samples from the training sample. Bootstrapped samples allow us to create many slightly different subsets of the training sample;

2. For each bootstrap sample, train a basic regression tree without the pruning process;

3. Averaging individual predictions from each regression tree to create an overall predicted value.

Despite the improvements that *Bagging* brings, the regression trees produced by *Bagging* are not completely independent of each other owing to the fact that all of the original factors are considered at every split of every tree. The problem is known as tree correlation which could reduce the overall performance of the model (Breiman, 1996).

Random forests (Breiman, 2001) are built on the same fundamental principles as *CART* and *Bagging* and aim at improving the prediction accuracy further by minimizing the amount of correlation among regression trees. This can be achieved by injecting more randomness into the tree-growing process which can be practically done by only considering a subset of factors at every split of every tree to find the predictors and their split values.

Besides, Zhang et al. have found out that random forests often fail in extrapolating problems (Zhang et al., 2019). Therefore, in our paper, we analyze two types of random forests. The first one is a random forest for extrapolating[11], recorded as *RF1*, and the second one is a random forest for interpolating[12], recorded as *RF2*. In the paper, the random forest is produced with the help of the R package "ranger" (Wright and Ziegler, 2017). Additionally, in the construction of random forest, there are two hyper-parameters to adjust in the tuning process:

- *mtry*: number of factors to consider at each split to find the predictor and its split value;

[10] In *Bagging*, the construction of basic regression trees will not include the pruning process for the sake of reducing computational time.

[11] Extrapolating means that the paramenter space of the test sample is larger than that of the training sample.

[12] Interpolating means that the paramenter space of the test sample is the same as that of the training sample.

- *node_size*: minimum number of sample points within the terminal nodes, this hyper-parameter controls the complexity of the trees. Smaller *node_size* allows for more complex trees.

As described in section 4.3, a k-fold cross validation approach is applied to tune the hyper-parameters *mtry* and *node_size* which are selected on a grid search (see Table 11 in the appendix). In addition, there is another hyper-parameter that we can tune in the process which is the number of the trees *ntree* in the final random forest, but it is not a very sensitive hyper-parameter and we simply follow the rule-of-thumb by using 500 trees.

3.6. Artificial Neural Network

Proposed in the 1940s as a simplified model of the elementary computing unit in the human cortex, artificial neural networks (Ripley, 1994; Bishop et al., 1995), recorded as *ANN*, have since been an active research area. It is an information processing model that is composed of a large number of highly interconnected processing elements known as the neurons to learn to solve problems by examples. An artificial neural network is a complex adaptive system, which means that it has the ability to change its internal structure by adjusting the weights in the network.

To understand the structure of an artificial neural network and how it works, we begin with its basic element: a neuron. The structure of a neuron is illustrated in Figure 4. The neurons are crucial elements because they are the places where the calculations happen, or to put it in a fancy way, where the model learns.

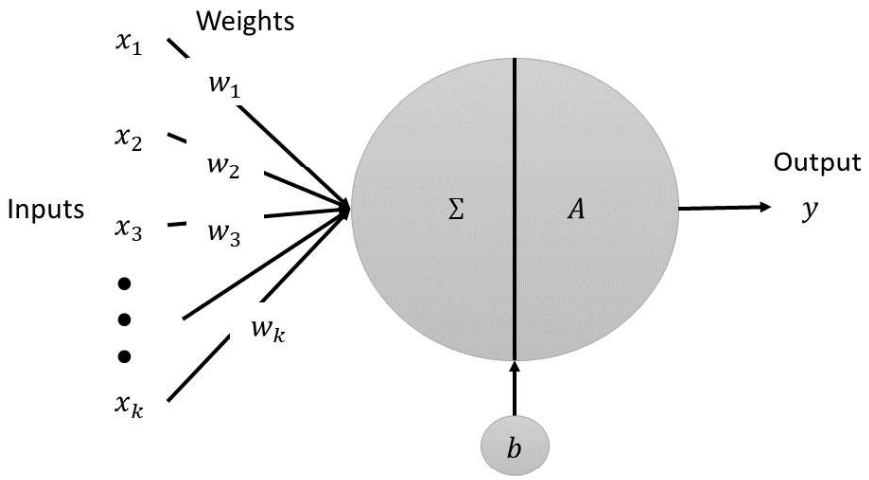

Figure 4: The structure of a neuron.

The inputs are connected to the neuron and each input has an initial weight. Then the neuron calculates a weighted sum of its inputs, adds a bias and decides whether it should be activated or not:

$$y = A(\sum_{h=1}^{k} w_h x_h + b), \tag{16}$$

where $\sum_{h=1}^{k} w_h x_h + b$ is passed to the activation function A (Awad and Khanna, 2015) which can be considered as a "gate" between the inputs feeding the current neuron and its output going to the next layer. Besides, it helps the network learn complex relationships in the data by adding non-linearity into the calculation. The bias b shifts the activation function to the right or the left, which has a similar role to the constant of a regression model that moves the regression line to fit the prediction with the data better.

An artificial neural network is a collection of multiple layers of neurons. Figure 5 shows the structure of a one-hidden-layer artificial neural network. The layer that contains the inputs is called the input layer, the layer that contains the outputs is called the output layer, and the layer that contains the neurons is called the hidden layer. In each neuron on the hidden layer, the calculation mentioned above is executed. Then the results from each activated neuron is passed further to the neuron on the output layer where the same calculation is carried out again. Thus, we get the predicted outputs. For regression problems, the activation function in the neuron on the output layer is an identity function which means that the predictions are a linear combination of the values from the activated neurons on the hidden layer.

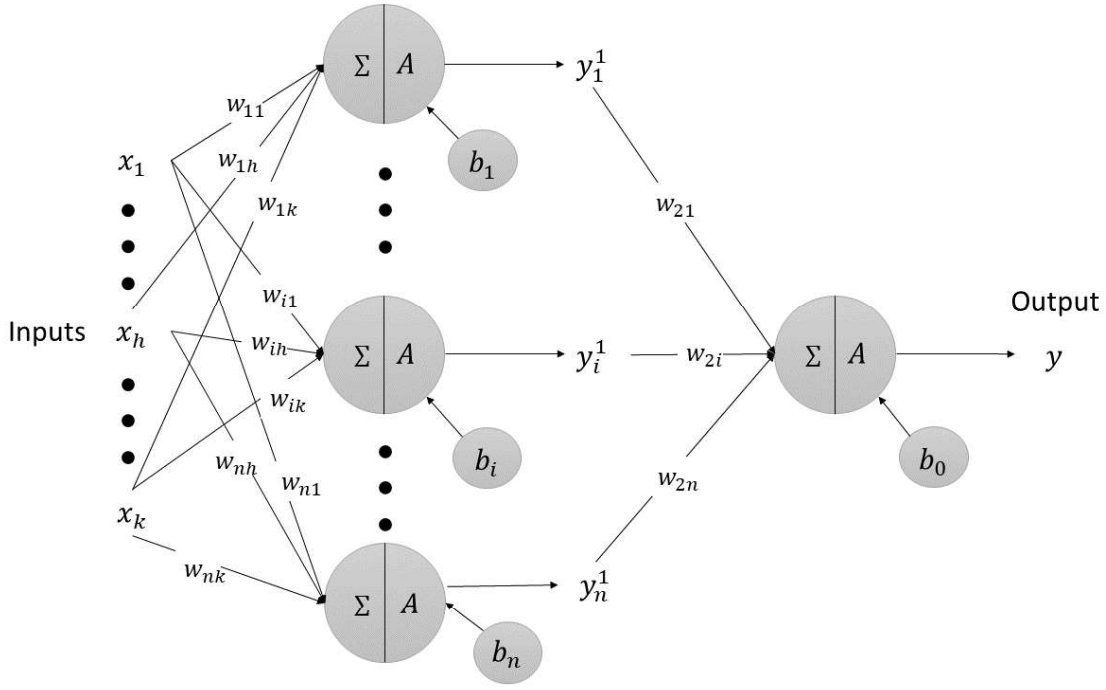

Figure 5: An artificial neural network with one hidden layer.

The number of hidden layers determines the depth of an artificial neural network. A neural network can be "shallow", meaning it has an input layer, only one hidden layer that processes the inputs, and an output layer that provides the final output of the model. A Deep Neural Network (DNN) commonly has between 2-8 additional layers of neurons. Researches from Goodfellow et al. and other experts suggest that neural networks increase in accuracy with the number of hidden layers (Goodfellow et al., 2016). Moreover, the deeper the artificial neural network, the more capable it is of approximating complex relationships, while however, the higher the demand is for the data, which will make the compuation particularly time-consuming.

During the learning process the weights are determined by minimizing the mean squared error:

$$P = \frac{1}{2}\sum_{i=1}^{n} \|\hat{\mathbf{y}}_\mathbf{i} - \mathbf{y}_\mathbf{i}\|^2, \tag{17}$$

where P is not a quadractic function of w and the existence of a global minimum is not garanteed. Thus, gradient descent based approximation algorithms have been utilized to find an approximate solution, where the gradient of w is calculated by the back-propagation principle (Werbos, 1974).

Moreover, Krogh and Hertz have introduced a penalization strategy, called weight decay, to tackle the problem of overfitting, which aims at minimizing the following performance function (Krogh and Hertz, 1992):

$$P = \frac{1}{2}\sum_{i=1}^{n} \|\hat{\mathbf{y}}_\mathbf{i} - \mathbf{y}_\mathbf{i}\|^2 + C\|\mathbf{w}\|^2, \tag{18}$$

where C is the penalization parameter.

The learning process of artificial neural network models is an iterative process. The algorithm compares the predicted values with the true values and record the information of the errors. Then the information is propagated backwards to all the activated neurons on the hidden layer and dependent on the contribution of each activated neuron to the prediction, a corresponding correction is made to reduce the error.

In our paper, we will analyze two types of artificial neural networks. The first one is an artificial neural network with one hidden layer and the penalization parameter, recorded as *ANN1*. The second one is an artificial neural network with multiple hidden layers (in this paper, we train a three-hidden-layer artificial neural network model) but without the penalization parameter, recorded as *ANN2*. *ANN1* is constructed with the help of the R package "nnet" (Fritsch et al., 2019) while *ANN2* is constructed with the help of the R package "neuralnet" (Venables and Ripley, 2002). In the construction of *ANN1*, there are two hyper-parameters in the tuning process: the number of neurons on the hidden layer Q_1 and the penalization parameter C while in the construction of *ANN2* there are three hyper-parameters in the tuning process: the number of neurons on the three hidden

layers Q_1, Q_2 and Q_3.

As described in section 4.3, a k-fold cross validation approach is applied to tune the hyper-parameters which are selected on a grid search (see Table 12 and Table 13 in the appendix).

4. Simulation and Results

4.1. Simulation Model

In this paper, we use the recursive-dynamic CGE model for Senegal of the International Food Policy Research Institute (IFPRI) developed by Löfgren et al. (2002); Diao and Thurlow (2012), and the corresponding 2015 social accounting matrix (SAM) constructed by Randriamamonjy and Thurlow (2019).

Given that the focus of our study is to compare the predictive ability and demenstrate the features of various metamodel types, we simplify the model structure by aggregating the accounts of the original SAM (see B.1 for more details). We also select specific functional forms and closures of the CGE model such that they reflect long-term adjustment possibilities of the Senegalese economy (see B.2 for more details).

4.2. Inputs and Outputs (Sampling and Simulation)

Two output variables are studied in this paper: income (represented by GDP per capita, Z_{11}) and poverty (represented by national poverty headcount rate, Z_{14}). Moreover, we select ten years (2016-2025) as the simulation horizon and consider the linear growth rates as the responses for the metamodels:

$$z = \frac{Z^{2025} - Z^{2016}}{Z^{2016}}, \tag{19}$$

where Z^{2016} represents the simulated value of year 2016, Z^{2025} represents the simulated value of year 2025 and z is the linear growth rate.

The degree of SAM aggregation and selected functional forms as well as closures define a set of CGE parameters that have direct or indirect impact on z.

There are two groups of inputs parameters that we will study in the paper: parameters of policy intervention and uncertain non-policy CGE parameters. They refer to the main driving forces affecting z in the simulation.

As parameters of policy intervention, we consider the growth of TFP (Total Factor Productivity) induced by public investment in the following sectors:

- crop production (**crop**);
- other agriculture: forestry, fishing, and livestock (**oagr**);

- processing of agricultural products (**agrib**);
- other industrial production (**oind**);
- public goods and services (**pub**);
- private-sector services (**prserv**).

As a set of uncertain non-policy CGE parameters that can have additional significant impact on z, we consider parameters of the CGE equations that describe:

- production side of the economy (growth of primary production factors (capital, labor, agricultural land) and elasticities of substitution);
- trade and other iterations with the Rest of the World (world market prices, elasticities of substitution and transformation, current account balance).

Model parameters of the equations that describe consumption side of the economy are excluded from the analysis and assumed to be fixed (certain). All in all, total number of CGE input parameters is 33.

We use different sources, estimates, and assumptions to assign initial values (μ_h) for all CGE parameters (see Section B.3 for more details). Then, for 33 CGE parameters that are assumed to be uncertain, we additionally specify values of the standard deviations (σ_h). Wherever possible, we used historical data to estimate σ_h (see Section B.4 for more details); for those parameters that do not have historical data available, we use rule of thumb and assign $\sigma_h = \frac{1}{3}\mu_h$.

We generate two latin hypercube samples for our analysis: the training sample and the test sample. In the training sample, each parameter is sampled within $[\mu_h - 2*\sigma_h, \mu_h + 2*\sigma_h]$ and in the test sample each parameter is sampled within the same range except for the TFP parameters which are drawn within $[\mu_h - 3*\sigma_h, \mu_h + 3*\sigma_h]$, namely, the test sample contains sample points which do not exist in the parameter space that is represented by the training sample.

With the simulation sample generated, we run the simulation and collect the simulated outputs z.

4.3. Training, Tuning and Testing

According to the idea of validation introduced in Section 2, we generate two samples for the analysis: the training sample with the size of 4000 and the test sample with the size of 1000. The training sample is used to train the metamodels and the test sample is used to evaluate the trained metamodels. Additionally, we want to make an extra analysis of the relationship between the size of the training sample and the predictive ability of the metamodels, thus we further extract some subsets of the orginal training sample. Finally,

we have training samples with sizes 4000, 2000, 1000, 500, 200 and 100, each of them being the subset of the previous one.

For each metamodel type, each output and each training sample size, we do the following:

1. Train the metamodel with the training sample;
2. Predict the outputs for the test sample using the trained metamodel;
3. Assess the prediction accruacy and other properties.

Importantly, in the step of training the metamodels, we need to tune the hyper-parameters to get the optimal prediction accuracy. To explain how the tuning process works, let's take the *MARS* model for example which has two hyper-parameters, *degree* and *nprune*, to adjust in the tuning process. Firstly, a grid containing various combinations of values of the two hyper-parameters is generated. Secondly, the metamodel is trained using each combination from the grid by means of the k-fold cross validation method[13]. Thirdly, the combination that gives the best prediction accuracy is kept and used to make predictions for the test sample. Finally, the predicted results are compared across all metamodel types.

The tuning process is applied to all metamodel types except for *LM1*, *LM2*, *OK* and *UK*. As for the polynomial models, they do not have hyper-parameters to tune. As for the Kriging models, it is due to the high computational cost that instead of the cross validation we apply a simple validation strategy, namely, we use the training sample to train *OK* and *UK* while the test sample is used to select the best hyper-parameters on a grid search in the tuning process.

In the assessment of prediction accuracy and the variability of the predictions, we follow Villa-Vialaneix et al. and use the following metrics:

- the Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{z}_i - z_i)^2, \tag{20}$$

  where z_i represent the simulated output values in the test sample and $\hat{z}_i$ represent the predicted output values of the metamodel.

- the R^2 coefficient:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(z_i - \hat{z}_i)^2}{\sum_{i=1}^{n}(z_i - \bar{z})^2}, \tag{21}$$

[13]In this paper, we use 10 folds in the cross validation.

where $\bar{z}$ is the mean of the outputs in the test sample.

- the standard deviation and the maximum of the squared errors are also computed so that we can have an insight on the variability of the prediction.

4.4. Results and Discussion

In this section, we compare the results from four aspects. Section 4.4.1 compares the prediction accuracy of all metamodel types. Section 4.4.2 compares the computational time for training and prediction. Section 4.4.3 discusses the variable importance and understanding of the metamodels. Section 4.4.4 discusses the potential extensions of the metamodels. Section 4.4.5 summarizes the results.

4.4.1. Accuracy

The prediction performance of the metamodels are summarized in Figure 8 and Figure 9 as well as in Table 1 to Table 4. The two figures show the evolution of R^2 for z_{11} and z_{14}. Table 1 and Table 3 show results of MSE which evaluate the prediction accuracy while Table 2 and Table 4 display the results of the maximum and standard deviation of the squared errors which assess the variability of the performance with respect to z_{11} and z_{14} whose distributions are presented in Figure 6 and Figure 7 where the orange line refers to the mean and the blue line refers to the median of the simulated values in the test sample.

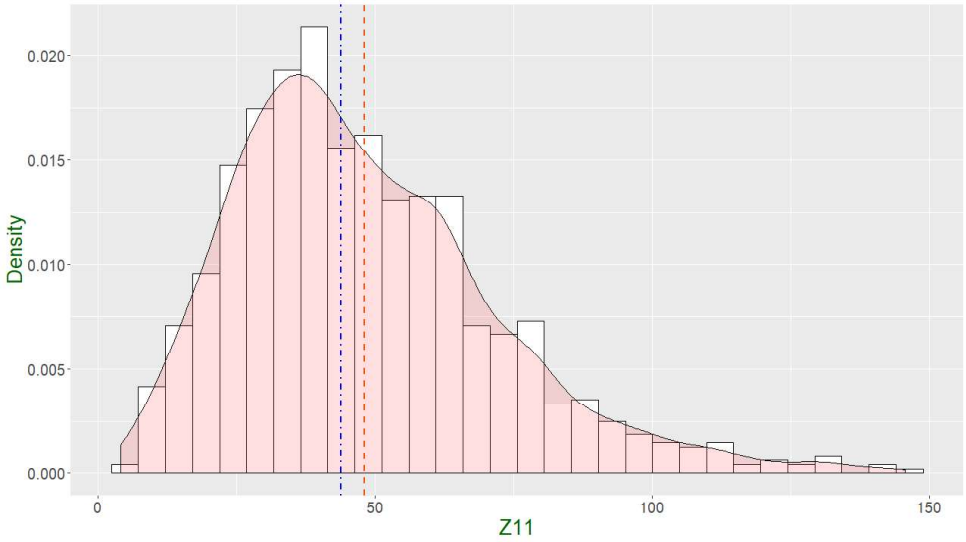

Figure 6: Distribution of z_{11}.

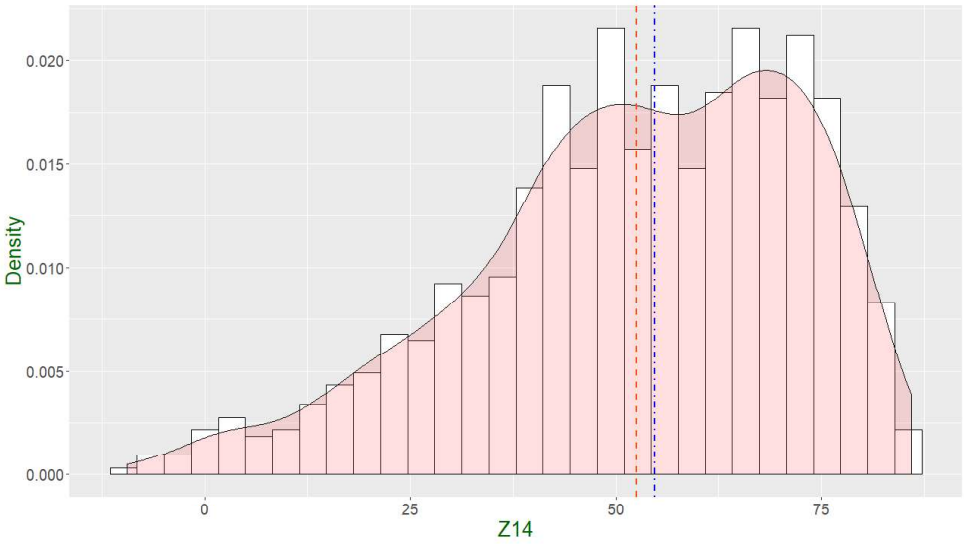

Figure 7: Distribution of z_{14}.

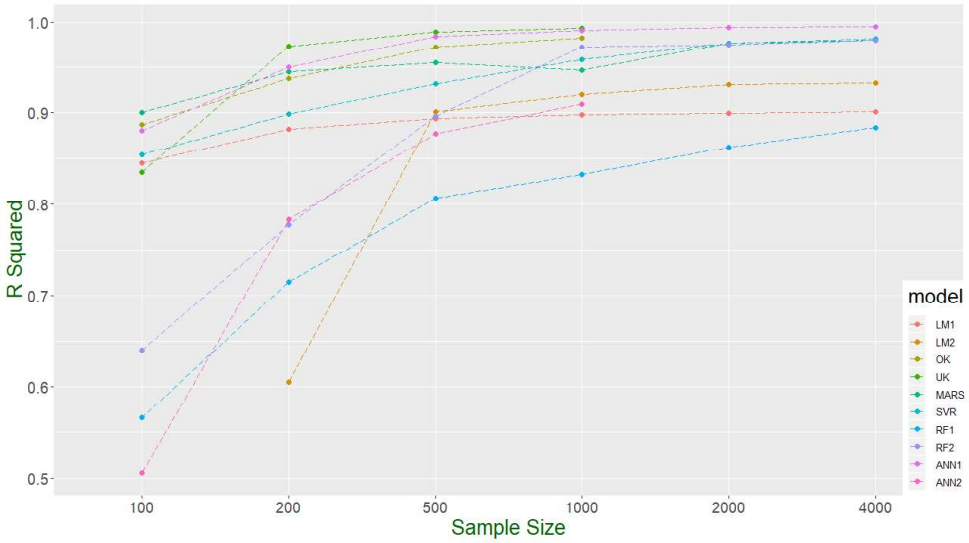

Figure 8: R^2 evolution with respect to the size of the training sample for z_{11}.

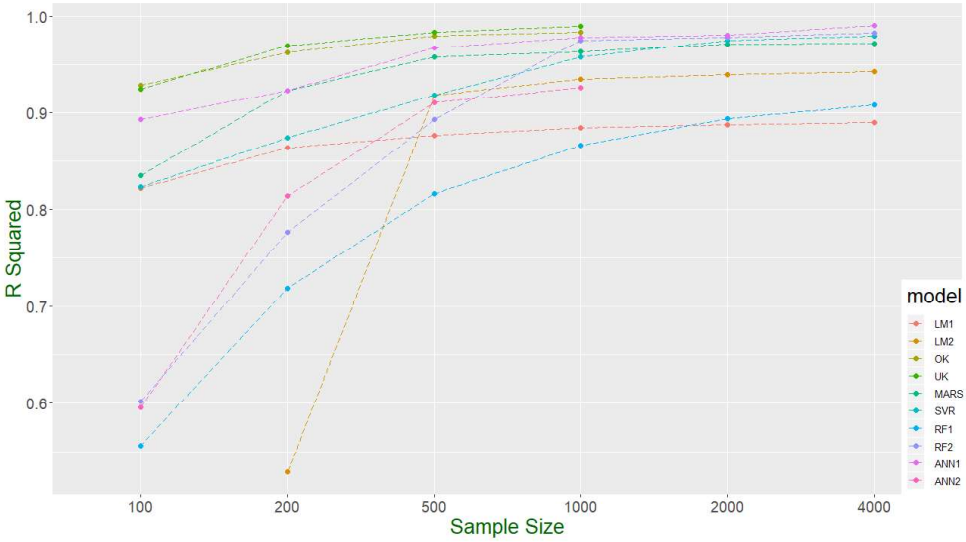

Figure 9: R^2 evolution with respect to the size of the training sample for z_{14}.

Size	LM1	LM2	OK	UK	MARS	SVR	RF1	RF2	ANN1	ANN2
100	86.98	•	63.1	92.34	55.86	81.83	243.08	201.32	66.78	277.05
200	65.87	221.23	34.73	15.11	30.8	56.87	159.8	124.46	28.05	121.07
500	59.34	55.4	15.56	5.99	24.95	38.16	108.38	58.16	9.06	68.82
1000	57.42	44.55	10.03	3.81	29.78	23.28	93.86	15.51	4.99	50.61
2000	56.2	38.51	•	•	13.15	13.03	77	14.15	3.35	•
≈ 4000	55.44	37.7	•	•	11.19	10.39	65.06	11.08	3.03	•

Table 1: MSE on the test sample for each metamodel and each size of the training sample for z_{11}. • refers to the cases that we cannot train, either because the model is over-specified (*LM2*) or the size of the training sample is so large that the estimation of the model takes too much time (*OK*, *UK* and *ANN2*).

Size	LM1	LM2	OK	UK	MARS	SVR	RF1	RF2	ANN1	ANN2
100	2489.62	•	2541.72	2701.38	1355.2	2653.9	5910.85	10325.96	2420.03	6400.36
	180.11	•	193.1	180.75	104.78	241.32	559.38	573.6	176.94	553.13
200	1917.94	6598.59	1717.58	633.81	1327.45	1776.34	4508.59	8252.96	920.93	3323.25
	152.05	390.09	119.49	43.04	80.31	161.35	357.64	398.94	70.83	267.17
500	1421.13	1143.14	846.52	211.03	1095.23	1222.32	2407.63	3130.46	269.1	1700.46
	135.81	110.31	51.07	14.66	71.49	112.65	217.7	162.7	20.14	164.66
1000	1537.62	893.91	684.01	141.16	2049.5	951.15	1833.46	1009.25	211.26	2667.25
	145.75	94.24	36.54	9.74	136.25	71.71	180.04	44.05	11.99	174.02
2000	1548.74	957.6	•	•	556.4	497.99	1399	1033.68	127.95	•
	140.33	86.7	•	•	36.27	37.8	141.93	44.68	7.97	•
≈ 4000	1662.99	908.72	•	•	682.86	578.37	1330.66	784.31	108.94	•
	143.98	86.8	•	•	39	31.19	122.34	33.44	6.61	•

Table 2: Maximum (first line) and standard deviation (second line) of the squared errors on the test sample for each metamodel and each size of the training sample for z_{11}. • refers to the cases that we cannot train, either because the model is over-specified (*LM2*) or the size of the training sample is so large that the estimation of the model takes too much time (*OK*, *UK* and *ANN2*).

Size	LM1	LM2	OK	UK	MARS	SVR	RF1	RF2	ANN1	ANN2
100	69.11	•	27.71	29.21	64.08	68.6	172.73	170.58	41.44	157.43
200	52.83	183.43	14.8	11.96	29.96	48.66	109.41	95.55	29.98	72.2
500	47.9	31.88	7.98	6.4	16.56	31.71	71.3	45.6	12.95	34.79
1000	44.97	25.44	6.42	4.19	14.28	16.45	51.97	10.72	8.72	28.64
2000	43.64	23.66	•	•	11.51	9.98	41.33	9.34	7.86	•
≈ 4000	42.75	22.44	•	•	11.14	8.07	35.69	7.35	3.85	•

Table 3: MSE on the test sample for each metamodel and each size of the training sample for z_{14}. • refers to the cases that we cannot train, either because the model is over-specified (*LM2*) or the size of the training sample is so large that the estimation of the model takes too much time (*OK*, *UK* and *ANN2*).

Size	LM1	LM2	OK	UK	MARS	SVR	RF1	RF2	ANN1	ANN2
100	1057.46	•	437.26	466.2	804.11	1443.86	1697.13	1311.17	646.56	4167.11
	106.06	•	52.19	49.96	112.38	120.54	212.95	207.76	71.92	317.84
200	1334.47	1920.91	362.37	260.71	527.08	1521.67	1408.63	811.78	528.67	1110.29
	88.11	260.64	28.34	22.04	48.59	89.56	152.35	132.31	48.48	129.42
500	1099.82	530.93	197.67	210.84	373.37	1289.35	1114.7	832.9	332.1	533.55
	79.73	54.24	14.86	12.96	26.88	63.72	106.58	89.39	21.71	59.72
1000	1016.95	458.21	193.15	202.22	190.22	386.71	957.14	148.02	225.4	572.68
	77.01	44.8	12.54	9.64	22.27	28.39	79.28	16.61	15.4	53.86
2000	952.3	473.24	•	•	298.44	264.58	789.24	134.96	169.62	•
	73.06	44.07	•	•	21	19.58	63.94	14.98	14.01	•
≈ 4000	952.67	518.2	•	•	310.55	213.81	651.14	86.17	110.04	•
	71.93	43.01	•	•	19.64	15.66	55.56	11.37	6.77	•

Table 4: Maximum (first line) and standard deviation (second line) of the squared errors on the test sample for each metamodel and each size of the training sample for z_{14}. • refers to the cases that we cannot train, either because the model is over-specified (*LM2*) or the size of the training sample is so large that the estimation of the model takes too much time (*OK*, *UK* and *ANN2*).

To assess the prediction performance, we categorize the findings into general findings that apply to all metamodel types and specific findings that are unique to each metamodel type.

We start with drawing some general conclusions:

- No models can deliver proper prediction performances when the sample size is 100.
- Even for small training sample sizes, R^2 is in some cases over 0.8. However, we want to point out that high R^2 does not necessarily mean good prediction accuracy.
- Polynomial models have a clear weakness in terms of prediction performance, particularly when large training samples are available.
- The best prediction performance in terms of MSE is always achieved with the largest training sample. For all the metamodels, we observe a clear trend that the prediction performances increase with the size of training sample rising except for one case where the *MARS* model trained with a smaller sample (500) has a better prediction accuracy than the *MARS* model trained with a larger sample (1000). Although *OK*, *UK* and *ANN2* come across computational problems due to the large size of the training sample, we can still assume that the trend will hold if the computations could proceed. Besides, we can observe that the amount of data matters a lot to machine learning models *SVR*, *RF* and *ANN*.
- In most cases, we observe that the maximum and standard deviation of the squared errors decrease while the size of the training sample increases meaning that the variability of the errors diminishes with the prediction accuracy improving except

Sample Size	z_{11}	z_{14}
200	126	136
500	59	64
1000	62	73
2000	61	79
$\approx$ 4000	76	89

Table 5: Number of covariates selected by the AIC criterion in *LM2* for z_{11} and z_{14} across various training sample sizes.

for some metamodels, such as *LM1*, *LM2* and *MARS*, where we also observe slightly different behaviors.

- Across all the cases, the most accurate prediction in terms of MSE are also the least variable in terms of the maximum and standard deviation of the squared errors.

Then, we draw some specific conclusions:

- The prediction accuracy for *LM1* improves when the size of the training sample increases from 100 to 500 and stays relatively stable afterwards which is probably owing to the fact that the specification of the metamodel is very simple. Therefore, even though more and more information is given to the model, it cannot utilize them in exchange for better predictions.

- *LM2* cannot be estimated when the size of the training sample is 100 due to over-specification. It performs very badly when the size of the training sample is 200 because the information provided by the training sample is barely enough to train the model well which affects the predictive ability. However, with the size of the training sample further rising, the prediction accuracy of *LM2* improves as well. It is also clear that for each output and size of the training sample, the prediction performance of *LM2* is better than that of *LM1* due to the fact that *LM2* has more terms in the specification which allows it to model more complex relationships and implies that the underlying I/O relationships are not simple linear relationships. The number of covariates selected by the "AIC" procedure across the two output variables with respect to various sample sizes is given in Table 5 from which we notice that the number of covariates selected is clearly larger than that in the *LM1* model (33).

- The Kriging models (*OK* and *UK*) work better with small and medium training sample sizes than other metamodels especially for *UK* as it has the best performance among all the metamodels for each training sample size from 200 to 1000. Due to the

computational limitations, for larger sizes the estimation and optimization processes of Kriging models took so long that we have to stop them.[14] We observe that for z_{11} and z_{14}, the prediction performance of *UK* is in most cases better than that of *OK* because in comparison to *OK* that only contains a constant trend, *UK* has a more complicated trend which helps in making predictions. However, for the two outputs, when the training sample size is 100, *OK* behaves better in prediction probably because the sample size is too small to produce a stable *UK* model and its predictive ability is thus poorer than *OK*. Moreover, *UK* has the same trend as *LM1*, yet the prediction accuracy is much better, which also reflects the underlying nonlinear I/O relationships.

- The *MARS* model works well with small, medium and large training samples. The prediction accuracy is better with larger training sample with one exception where for z_{11} the prediction performance is better with size 500 than that with size 1000. The prediction accuracy of *MARS* is not the best but always stays in an acceptable range. We can have a look at the *degree* and *nprune* that are selected by *MARS* with various training sample sizes which is given in Table 6. It is not surprising that with the size of the training sample increasing, the *MARS* model tends to use higher interaction orders and retain more terms in order to pursue better prediction accuracy.

	z_{11}		z_{14}	
Sample Size	*degree*	*nprune*	*degree*	*nprune*
100	3	21	1	8
200	2	30	3	23
500	2	36	2	36
1000	3	40	3	38
2000	3	40	3	40
$\approx$ 4000	3	40	3	40

Table 6: Hyper-parameters *degree* and *nprune* selected by the *MARS* for z_{11} and z_{14} across various training sample sizes.

- The machine learning model *SVR* is a data-intensive metamodel and therefore behaves poorly with small training sample sizes (100 and 200). Starting from medium

[14] On our PC, the process ran for more than three days and continued to run. Therefore, we manunally stopped the process.

sizes, the *SVR* model displays satisfactory and stabily increasing prediction accuracy.

- The machine learning model *RF* has shown quite different prediciton performances in terms of interpolating and extrapolating. The training samples are exactly the same for *RF1* and *RF2* while the test sample for *RF1* is the usual one and the test sample for *RF2* is the training sample with the size 1000, namely, the parameter space is the same as that of the training samples. We clearly observe that the random forest model is poor at extrapolating. This is because the random forest make predictions by using the constants from each terminal node. If the points to predict contain parameters whose ranges go beyond the ranges of the training sample, this means that the random forest model would as a result have no proper terminal nodes for these points, thus the predictions would be made by simply using the mean value of all the terminal nodes which is apparently not a good choice. However, if the parameter space of the test sample is similar to that of the training sample, as the case for *RF2*, the random forest model will behave correctly and make satisfactory predictions with medium to large training sample because the random forest model is a data-intensive machine learning technique.

- The machine learning model *ANN* has demonstrated different results in terms of the two setups: *ANN1* has one hidden layer with the penalization parameter; *ANN2* has three hidden layers. The results show that *ANN1* starts to deliver good prediction accuracies when the training sample size is 500 while for the training sample that we are able to train the *ANN2*, its prediction accuracies are very poor, in some cases even poorer than that of *LM1*. This means that *ANN1* needs at least medium-sized training sample to get satisfactory prediction accuracy and *ANN2*, if it can, needs at least large training sample sizes. Besides, the results also clearly show that having more hidden layers doesn't necessarily yield better prediction performance. In our applications, it is actually the opposite, the prediction performance of *ANN2* is worse than that of *ANN1*. It might because the data is not sufficient to train the *ANN2* well as it is a deep artificial neural network, thus it cannot deliver good predictions. Additionally, when the size of the training sample gets large enough, *ANN1* always has the best prediction performance and the smallest variability in the errors.

4.4.2. Computational Time

The computational time for training and prediction is listed in Table 7. For metamodels *OK* and *UK*, the sample size for training is 1000 and the sample size for training for other metamodels is 4000. For all the metamodels, the size of the test

sample is 1000. *LM1*, *RF1* and *ANN2* are not included in the comparison due to their poor prediction accuracies.

From Table 7, we observe no significant differences in the training time for *LM2*, *MARS*, *SVR*, *RF2* and *ANN1* among which *ANN1* has the best prediction accuracy for the two output variables. Despite the good predictive ability of *OK* and *UK* for small and medium training sample sizes, the training time is obviously longer that that of other metamodels due to the optimization process in the estimation of coefficients. On the other side, the prediction takes a short time across all the metamodels possibliy because the size of the test sample is not large.

	LM2	***OK***	***UK***	***MARS***	***SVR***	***RF2***	***ANN1***
Training and Tuning	21 mins	3 hours	2 hours	15 mins	26 mins	21 mins	32 mins
Prediction	<1 sec	6 secs	5 secs	<1 sec	<1 sec	<1 sec	<1 sec

Table 7: Training and prediction time for different metamodels.

4.4.3. Variable Importance

To evaluate the variable importance in the prediction, methods can be categorized into two groups: model-specific and model-agnostic (Fisher et al., 2019). In this paper, we focus on the model-agnostic method that does not assume anything about the model structure which allows us to apply it to any predictive model and compare variable's importance across different models.

The main idea of the method is to measure the change of prediction performance when the effect of a certain independent variable is removed. This means that the prediction performance could suffer a loss when the effect of a certain independent variable is removed and the larger the loss in the prediction performance, the more important the variable is to the prediction. In this paper, we measure the loss in the prediction performance by the increase of the Root Mean Squared Error (RMSE). Besides, the removal of the effect is realized by applying perturbations. For more details of this method, we refer to Fisher et al. 2019.

Figure 10 and Figure 11 show the results of variable importance analysis for z_{11} and z_{14} among *LM2*, *UK*, *MARS*, *SVR*, *RF2* and *ANN1* while the other three metamodels are excluded due to the poor prediction performances.

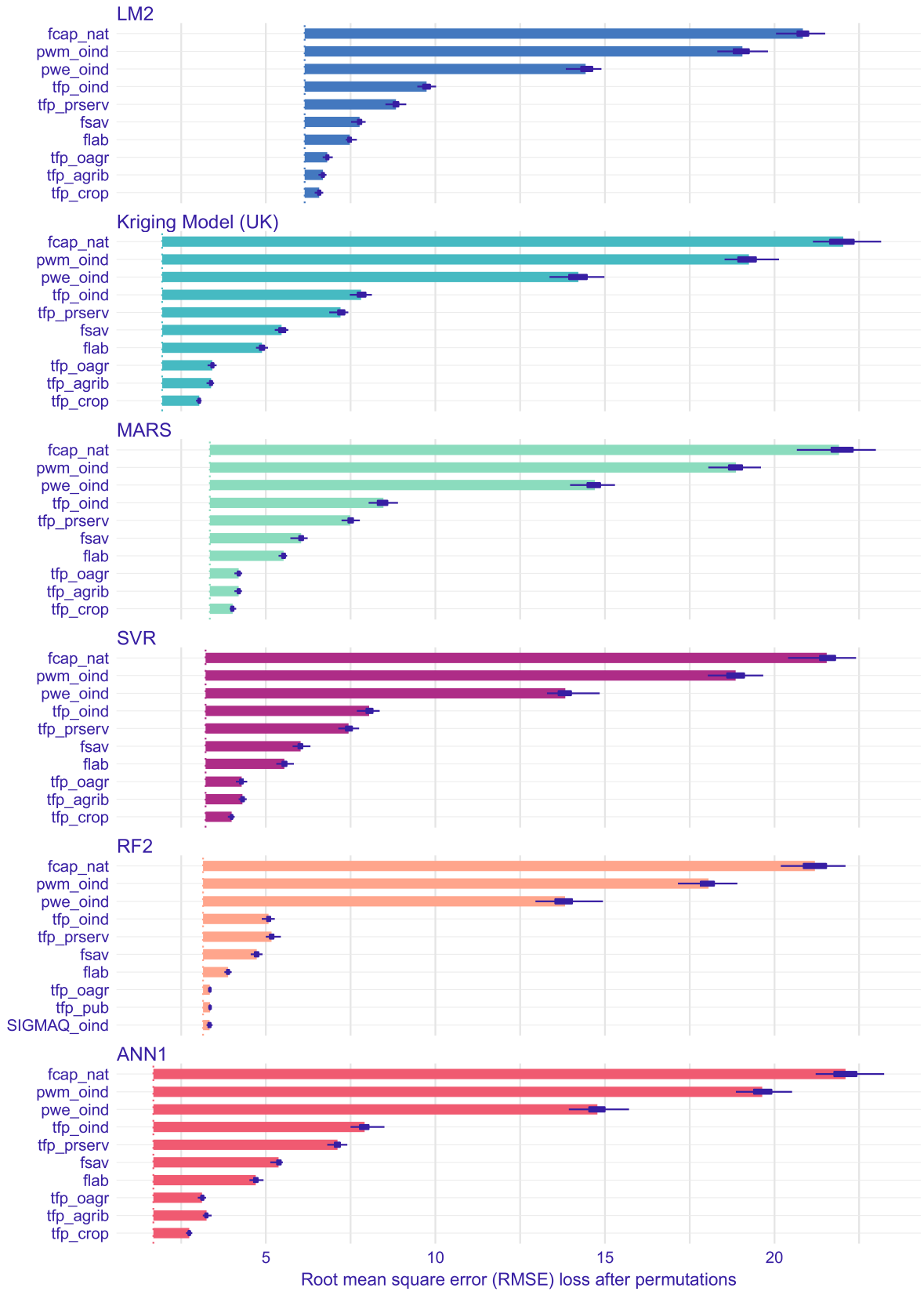

Figure 10: Variable importance analysis for z_{11}.

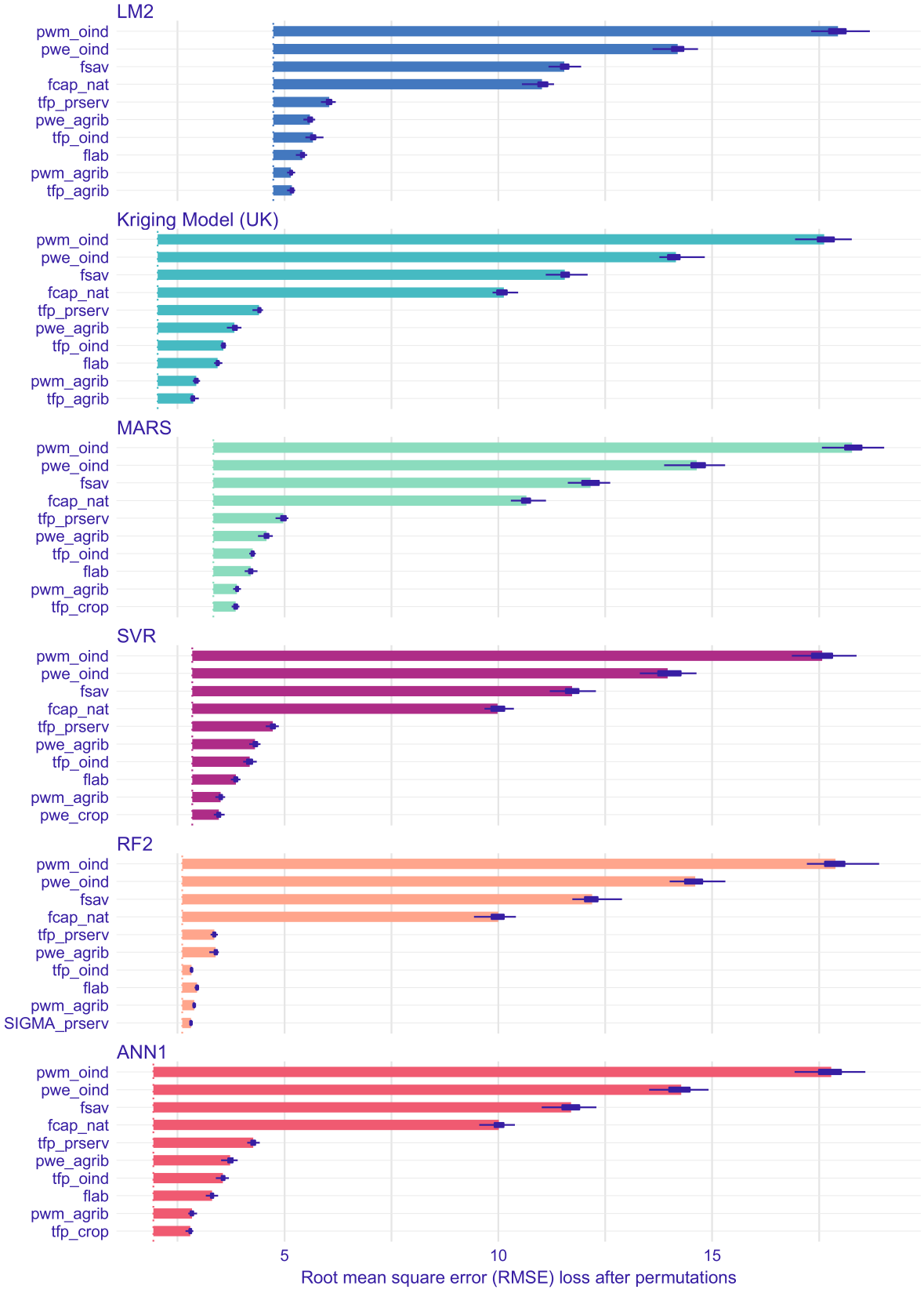

Figure 11: Variable importance analysis for z_{14}.

It shows that for the same output variables, z_{11} and z_{14}, different metamodels rank

the importance of input parameters in the same order but the magnitude of the importance (measured by the length of the bars on the figures) varies from model to model, for example, in the case of z_{11}, both *RF2* and *ANN1* rank the $flab$ (growth of land as a factor in the production) as the 7^{th} important parameter, but its importance is viewed quantitatively different by the two metamodels.

As for z_{11}, results show that the top five influential input parameters are $fcap\_nat$, pwm_oind, pwe_oind, tfp_oind and tfp_prserv. Senegal is a capital-scarce and labor-intensive country. Therefore, it is not surprising that capital growth ($fcap\_nat$) is very influential in driving the per capita GDP (z_{11}). Besides, Senegalese economy is highly dependent on trade, therefore world prices of exports (pwe) and world prices of imports (pwm) matter profoundly. Moreover, the goods from sector **oind** are far more tradable than other sectors and sector **oind** has a much wider economic linkages with other sectors. Thus, pwe_oind and pwm_oind are far more important than the two prices in other sectors. Growth of Total Factor Productivity (tfp) also matter to a certain extent and obviously, tfp in larger sectors (**oind** and **prserv**) are more important than those in smaller sectors. However, we want to mention that the difficulty of generating growth of total factor productivity in different sectors varies as well.

As for z_{14}, the top five influential input parameters are pwm_oind, pwe_oind, $fsav$, $fcap\_nat$ and tfp_prserv. It is not surprising that pwm_oind, pwe_oind, $fcap\_nat$, tfp_prserv appear on the list because they are paramters that drive the development of the Senegalese economy and a rising economic condition eases the poverty condition. Apart from that, in the Senegal CGE model, $fsav$ (foreign savings) is modeled in a way that it could represent the variation of exchange rate and food prices, such as rice prices, are very crucial for reducing poverty, thus it is considered as a very influential parameter.

For different output variables, z_{11} or z_{14}, metamodels treat input parameters in a distinct fashion in terms of importance. For example, *ANN1* ranks $fsav$ (foreign savings) the 6^{th} for z_{11} and the 3^{rd} for z_{14}. This phenomenon is more obvious if we list more input paramters with respect to importance. Figure 12 lists the 20 important input parameters of model *ANN1* for z_{11} and z_{14}. A typical case is tfp_oagr (the Total Factor Productivity growth in sector **oagr**) which ranks the 9^{th} influential parameter for z_{11} but turns out to be hardly effective for z_{14}. Besides, it shows that the number of main drivers for prediction is not high and many parameters are hardly influential in predicting the output variables, which is an interesting observation enabling us to have more insights of the CGE models.

Additionally, we do not observe sharp differences in the ranking of importance of input parameters for z_{11} and z_{14}, namely, there are no cases where a parameter is

among the most important ones for z_{11} but turns out to be hardly important for z_{14} and vice versa. This also makes sense because the CGE model is an internally interrelated model where parameters interact with each other extensively. Besides, z_{11} and z_{14} are also related with each other to a certain degree.

The variable importance analysis is very helpful to understand the main drivers of the I/O relationships that we are interested in. Due to the implicit nature and computational cost, performing sensitivity analysis for CGE models is not straightforward. However, the variable importance analysis with the help of the metamodels gives us the possibility to gain more insights of the underlying simulation model.

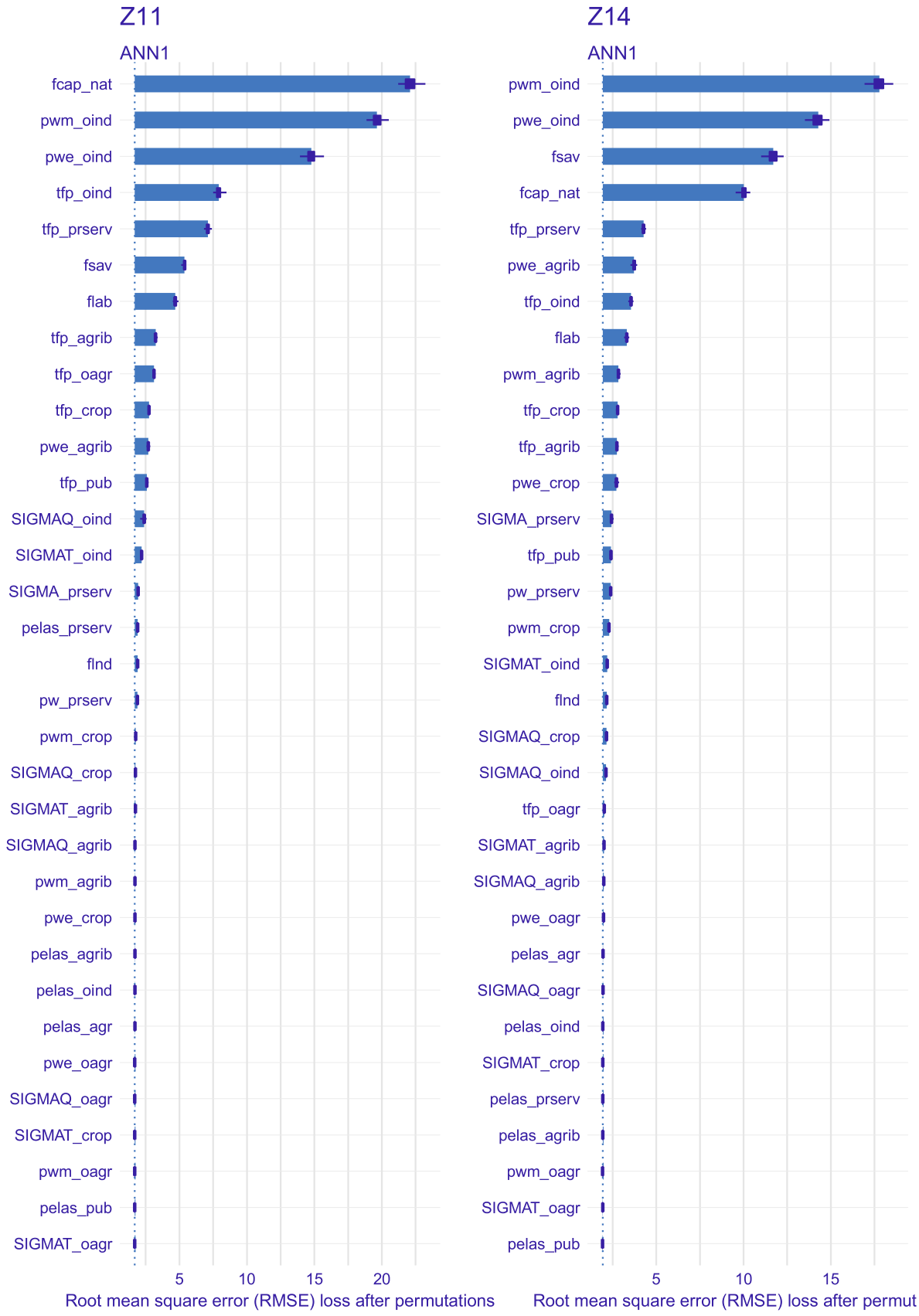

Figure 12: Variable importance analysis using *ANN1* for z_{11} and z_{14}.

4.4.4. Potential Extensions

Depending on the properties of different metamodels, they can be applied to different contexts where we can answer more complex research questions such as the ones we mentioned in Section 1. To start with, we want to emphasize a quite importance property which is the speed of prediction. It means that once a metamodel is well trained, it is exceptionally computation-efficient to be applied to any research environment. More specifically, take the *ANN1* model for example, the time for prediction of the test sample with 1000 simulation runs is only 1 second.

For example, we have applied the polynomial model to perform a baseline calibration of a large-scale dynamic CGE model Dynamic Applied Regional Trade (DART) with highly disaggregated regions and sectors[15]. The basic idea of employing metamodels to address empirical calibration problems is that we train a metamodel to approximate the underlying relationships and use it in the calibration process to obtain the "optimal" parameter values that make the simulation model produce outcomes that are close to the forecasts. Furthermore, with the metamodels, we can deal with the parametrization constraints because they can be directly added into the calibration process. In addition, Storm et al. have mentioned other possibilities for improving empircal calibration such as leveraging ideas from Generative Adversarial Nets (GANs), which is a very interesting and promising idea (Storm et al., 2020).

In addition, we have applied the polynomial model and Kriging model to perform a comprehensive policy analysis coupled with uncertainty analysis including both the impact of fundamental uncertainty regarding exogenous economic shocks and regarding the responses of the economy to these shocks as well as the impact of shocks on optimal policy choices. The basic idea is to train a metamodel and replace the simulation model with it in the optimization process to obtain an analytical solution. Moreover, an uncertainty analysis can be carried out because the probabilistic descriptions of model input parameters can be applied to metamodels so as to derive probability distributions of model outputs[16].

Furthermore, similar to the variable importance analysis we did in Section 4.4.3, metamodeling can help us conduct sensitivity analysis effectively and have more understanding of the underlying simulation model.

[15]The paper is available upon request.
[16]The paper is available upon request.

4.4.5. Summary

The results demonstrated in the sections above have given us some indications: the metamodeling techniques are able to make good approximations of the I/O relationships of the underlying CGE model at a low computational cost. Even the size of the training sample is 200, the *UK* is able to deliver a good prediction accuracy in spite of the fact that the training process is time-consuming in comparison with other metamodels. Then, with the size of the training samples increasing, the prediction performances of all the metamodels improve at different paces where machine learning metamodels tend to produce very precise predictions, among which *ANN1* clearly outperforms other metamodels. Thus, if a big training sample is not available, we can resort to Kriging or *MARS* or *ANN1* models dependent on our tolerance of the computational time and research purposes. On the contrary, if we are able to have a big training sample, the machine learning metamodels can deliver excellent prediction results while the computational costs are not very high. Moreover, polynomial and Kriging models enable us to integrate the underlying simulation models into other research frameworks to obtain analytical solutions and answer more complex questions, which opens up the door to endless possibilities.

Last but not the least, we want to point out again that the variable importance analysis in Section 4.4.3 is helpful to understand the driving forces of certain outputs which might help us have a better control of the simulation model and improve it further.

5. Conclusion

In this paper, we have conducted a systematic and comprehensive comparison analysis investigating several metamodel types in terms of accuracy, computational time, variable importance and potential extensions to demonstrate how to apply them in practical applications along with their properties. Moreover, we have pointed out some aspects where the metamodeling technique can be employed to answer complex research questions.

In summary, the metamodeling method is a useful tool for applying innovative simulation techniques and the combination of the two methods opens the door to endless possibilities.

Metamodeling is a very active research area and is constantly undergoing rapid developments. Therefore, we want to make the point that we should keep pace with this field and involve the novel methods into agricultrual economics.

References

Aguiar, A., Narayanan, B., and McDougall, R. (2016). An overview of the GTAP 9 data base. *Journal of Global Economic Analysis*, 1:181–208.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19(6):716–723.

Awad, M. and Khanna, R. (2015). *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. Springer Nature.

Barthelemy, J.-F. and Haftka, R. T. (1993). Approximation concepts for optimum structural design - a review. *Structural optimization*, 5(3):129–144.

Barton, R. R. (2015). Tutorial: simulation metamodeling. In *Proceedings of the 2015 Winter Simulation Conference*, pages 1765–1779. IEEE Press.

Birur, D., Hertel, T., and Tyner, W. (2007). Impact of biofuel production on world agricultural markets: a computable general equilibrium analysis.

Bishop, C. M. et al. (1995). *Neural networks for pattern recognition*. Oxford university press.

Blanning, R. W. (1975). The construction and implementation of metamodels. *Simulation*, 24(6):177–184.

Böhringer, C., Rutherford, T., and Wiegard, W. (2003). Computable General Equilibrium Analysis: Opening a Black Box. ZEW Discussion Paper No. 03-56, Center for European Economic Research, Mannheim.

Boogaerde, P. and Tsangarides, C. (2005). Ten years after the CFA Franc devaluation: Progress toward regional integration in the WAEMU. IMF.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

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

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Britz, W., Henning, C. H. C. A., and Henningsen, G. (2009). Modelling the impact of rural infrastructure policy on transaction costs and economic performance at the micro and macro level. Paper presented at the EU-Project ADVANCED-EVAL Workshop 'Evaluation and Modelling of Rural Development Policies: Theory and Application' in Kiel, Germany.

Chen, V. C., Tsui, K.-L., Barton, R. R., and Meckesheimer, M. (2006). A review on design, modeling and applications of computer experiments. *IIE transactions*, 38(4):273–291.

Craven, P. and Wahba, G. (1978). Smoothing noisy data with spline functions. *Numerische mathematik*, 31(4):377–403.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. John Wiley & Sons, Inc.

Dey, S., Mukhopadhyay, T., and Adhikari, S. (2017). Metamodel based high-fidelity stochastic analysis of composite laminates: A concise review with critical comparative assessment. *Composite Structures*, 171:227–250.

Diao, X. and Thurlow, J. (2012). *A recursive dynamic computable general equilibrium model*, chapter 2, pages 17 – 50. International Food Policy Research Institute (IFPRI).

Diao, X., Thurlow, J., Benin, S., and Fan, S., editors (2012). *Strategies and Priorities for African Agriculture - Economywide Perspectives from Country Studies*. International Food Policy Research Institute, Washington DC.

Dupuy, D., Helbert, C., and Franco, J. (2015). DiceDesign and DiceEval: Two R packages for design and analysis of computer experiments. *Journal of Statistical Software*, 65(11):1–38.

Fang, K.-T., Lin, D. K., Winker, P., and Zhang, Y. (2000). Uniform design: theory and application. *Technometrics*, 42(3):237–248.

Fisher, A., Rudin, C., and Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81.

Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons.

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

Fritsch, S., Guenther, F., and Wright, M. N. (2019). *neuralnet: Training of Neural Networks*. R package version 1.44.2.

Gong, W., Duan, Q., Li, J., Wang, C., Di, Z., Dai, Y., Ye, A., and Miao, C. (2015). Multi-objective parameter optimization of common land model using adaptive surrogate modeling. *Hydrology and Earth System Sciences*, 19(5):2409–2425.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Henning, C., Badiane, O., and Krampe, E., editors (2018). *Development Policies and Policy Processes in Africa*. Springer International Publishing.

IMF (2017). Senegal. Country report 17/2, International Monetary Fund.

Jaroslaw, S. and Raphael T, H. (1996). Multidisciplinary aerospace design optimization: Survey of recent developments.

Jourdan, A. (2005). *Planification d'Expériences Numériques*. PhD thesis.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., and Murthy, K. R. K. (2001). Improvements to platt's smo algorithm for svm classifier design. *Neural computation*, 13(3):637–649.

King, R. and Byerlee, D. (1978). Factor intensities and locational linkages of rural consumption patterns in Sierra Leone. *American Journal of Agricultural Economics*, 60 /2:197–206.

Kleijnen, J. P. (1975). A comment on Blanning's "Metamodel for sensitivity analysis: the regression metamodel in simulation". *Interfaces*, 5(3):21–23.

Kleijnen, J. P. (2015). Design and analysis of simulation experiments. In *International Workshop on Simulation*, pages 3–22. Springer.

Kleijnen, J. P. and Sargent, R. G. (2000). A methodology for fitting and validating metamodels in simulation. *European Journal of Operational Research*, 120(1):14–29.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

Kuhn, M. (2020). *caret: Classification and Regression Training*. R package version 6.0-86.

Lluch, C., Powell, A., and Williams, R. (1977). *Patterns in Household Demand and Saving*. Oxford University Press.

Löfgren, H., Harris, R. L., and Robinson, S. (2002). *A Standard Computable General Equilibrium (CGE) Model in GAMS*, volume Microcomputers in Policy Research 5. International Food Policy Research Institute, Washington, D.C.

Manski, C. F. (2018). Communicating uncertainty in policy analysis. *Proceedings of the National Academy of Sciences*, 116(16):7634–7641.

Manson, S. M. and Evans, T. (2007). Agent-based modeling of deforestation in southern yucatan, mexico, and reforestation in the midwest united states. *Proceedings of the National Academy of Sciences*, 104(52):20678–20683.

Mareš, T., Janouchová, E., and Kučerová, A. (2016). Artificial neural networks in the calibration of nonlinear mechanical models. *Advances in Engineering Software*, 95:68–81.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2020). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-4.

Milborrow., S. (2020). *earth: Multivariate Adaptive Regression Splines*. R package version 5.3.0.

Myers, R. H., Montgomery, D. C., and Anderson-Cook, C. M. (2016). *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons.

Owen, A. B. (1992). Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica*, pages 439–452.

Randriamamonjy, J. and Thurlow, J. (2019). (*mimeo*) 2015 Social Accounting Matrix for Senegal. A Nexus Project SAM, International Food Policy Research Institute, Washington, DC. USA.

Rasch, S., Heckelei, T., Storm, H., Oomen, R., and Naumann, C. (2017). Multi-scale resilience of a communal rangeland system in south africa. *Ecological Economics*, 131:129–138.

Razavi, S., Tolson, B. A., and Burn, D. H. (2012). Review of surrogate modeling in water resources. *Water Resources Research*, 48(7).

Ripley, B. D. (1994). Neural networks and related methods for classification. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(3):409–437.

Ruben, R. and van Ruijven, A. (2001). Technical coefficients for bio-economic farm household models: a meta-modelling approach with applications for southern mali. *Ecological Economics*, 36(3):427–441.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical science*, pages 409–423.

Simpson, T. W., Lin, D. K., and Chen, W. (2001). Sampling strategies for computer experiments: design and analysis. *International Journal of Reliability and Applications*, 2(3):209–240.

Simpson, T. W., Peplinski, J., Koch, P. N., and Allen, J. K. (1997). On the use of statistics in design and the implications for deterministic computer experiments. *Design Theory and Methodology-DTM'97*, pages 14–17.

Smith, M. (1993). *Neural networks for statistical modeling*. Thomson Learning.

Steinwart, I. and Christmann, A. (2008). *Support Vector Machines*. Springer Science & Business Media.

Storm, H., Baylis, K., and Heckelei, T. (2020). Machine learning in agricultural and applied economics. *European Review of Agricultural Economics*, 47(3):849–892.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vapnik, V. N. (1995). The nature of statistical learning. *Theory*.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

Villa-Vialaneix, N., Follador, M., Ratto, M., and Leip, A. (2012). A comparison of eight metamodeling techniques for the simulation of n2o fluxes and n leaching from corn crops. *Environmental Modelling & Software*, 34:51–66.

Werbos, P. (1974). Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

Wright, M. N. and Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17.

Yildizoglu, M., Salle, I., et al. (2012). Efficient sampling and metamodeling for computational economic models. Technical report, Groupe de Recherche en Economie Théorique et Appliquée.

Zhang, H., Nettleton, D., and Zhu, Z. (2019). Regression-enhanced random forests. *arXiv preprint arXiv:1904.10416*.

A. Summary of Hyper-parameters for Metamodels

Kernel Functions	**Optimization Method**
gauss	quasi-Newton procedure
matern5_2	genound genetic algorithm
matern3_2	
exp	
powexp	

Table 8: Hyper-parameters for tuning *OK* and *UK*.

degree	**nprune**
1	5
2	7
3	10
	13
	16
	19
	22
	25
	28
	31
	33
	36
	39
	42
	45
	48
	51
	54
	57
	60

Table 9: Hyper-parameters for tuning *MARS*.

ϵ	γ	**C**
0.1	0.001	0.5
0.01	0.01	1
	0.1	5
	1	10
	5	20
		50
		100

Table 10: Hyper-parameters for tuning *SVR*.

mtry	**node_size**
10	3
12	5
14	7
16	9
18	
21	
23	
25	
27	
30	

Table 11: Hyper-parameters for tuning *RF1* and *RF2*.

Q_1	**C**
20	0
22	0.001
24	0.01
26	0.1
28	0.5
31	
33	
35	
37	
40	

Table 12: Hyper-parameters for tuning *ANN1*.

Q_1	Q_2	Q_3
20	2	2
22	3	3
24	4	
26		
28		
31		
33		
35		
37		
40		

Table 13: Hyper-parameters for tuning *ANN2*.

B. CGE

B.1. SAM adjustments

- The original SAM by Randriamamonjy and Thurlow (2019) has regionalized production structure (5 regions) and total 462 accounts;
- the aggregated SAM used in our CGE model (Table 14) preserved the regionalized production structure and has 68 accounts;
- we also amend the original SAM such that public goods are consumed only by the government. This amendment allows us to assume that public goods and services are outside of consumers linear expenditure system (LES).

Category	Description	Label	Dimension
Activities	Crops, other agriculture, agribusiness, other industry, public, private services	acrop, aoagr, aagrib, aoind, apub, aprser	Regional
Commodities		ccrop, coagr, cagrib, coind, cpub, cprser	National
Labor	All types of labor	flab	Regional
Land	All types of land	flnd	Regional
Capital	Agriculture-related Non-agriculture-related	fcap-a fcap-n	Regional Regional
Enterprises	Distributional account (capital rents)	ent	National
Households	Urban Rural	hhd-u hhd-r	Regional Regional

Table 14: Aggregated SAM used in the CGE model (available upon request).

B.2. Functional forms and closure rules

We define functional forms and closure rules for our CGE model based on the recursive-dynamic CGE model developed by Löfgren et al. (2002) and Diao and Thurlow (2012).

B.3. Baseline CGE parameter values (vector of mean values)

- We use approximations and assumptions when the necessary estimates/forecasts are not available. For example, we use the Global Trade Analysis Project

Block	Category	Form / closure (endogenous variables)
Production	Value-added	constant elasticity of substitution (CES)
	Intermediate	Leontief
	Top of technology	Leontief
Trade	Import	CES
	Export	constant elasticity of transformation (CET)
Consumption	Households	LES
Closures	Current account	Consumer and producer price level (exchange rate is model numerraireExchange rate of CFA franc/French franc (EUR) is fixed since 1994. See Boogaerde and Tsangarides (2005) for more details)
	Factors	Fully employed and mobileCapital: 'putty-clay' assumption, see Diao and Thurlow (2012) for more details
	Government	Budget deficit
	Savings/Investment	Balanced 'S-I' closure 4, with enterprises adjusting marginal propensity to save. See Löfgren et al. (2002) for mroe details)

Table 15: Functional forms and closures of the CGE.

(GTAP) 9 database (Aguiar et al., 2016) to calculate the values of elasticity parameters;

- we use our assumption about the productivity growth of 1 % per year for all sectors in the baseline scenario for 2016-2025[17];
- because the composition of export and import commodities within defined aggregated SAM account can be different (e. g. export crops consist mostly of nuts, vegetables, and fruits, while the major import crop is rice), we distinguish between export and import elasticity parameters, and we distinguish between export and import world market prices.

[17] We prefer this modest estimate over the observed historical productivity decline in 2006-2015 and very high productivity increase forecast by the IMF (2017) for 2016-2021

Category	Used sources	Values
Total Factor Productivity (TFP) growth	Own assumption	All 1%
Growth of factors	ILO for labor; FAO for land; PWT for economywide capital	flab 3.07%; flnd 0%; economy-wide capital (fcap_e) 5.34%
Population growth	UN	hhd_u 3.72%; hhd_r 1.78%
Growth of world market prices	WB Commodities Price Forecast and FRED for CPI services in France	Export (E): crop 1.35%, oagr -0.45%, agrib 0.4%, oind 1.32%, prserv 1.39%
		Import(I): crop 0.94%, oagr 0.36%, agrib 0.91%, oind 1.39%, prserv 1.39%
Growth of BoP deficit	IMF WEO	fsav 11.17%
Production CES elast.	Based on GTAP 9	crop & oagr 0.24; agrib 1.17; oind 1.24; pub 1.26; prserv 1.41
Trade elast.	Based on GTAP 9	CET: crop 2.84; oagr 1.47, agrib 2.58, oind 1.98, prserv 1.9; CES: crop 4.29; oagr 2.27, agrib 3.39, oind 2.79, prserv 1.9
Income elasticities	Own estimates based on the household survey and King and Byerlee (1978)	Available upon request
Frisch parameters	Own estimates based on the household survey, WB WDI and Lluch et al. (1977)	Available upon request

Table 16: Baseline parameters.

B.4. Estimated variance

We use the same sources as in appendix B.3 and construct the historical sample of 10-year moving averages throughout 1980-2015 and estimate components of the multivariate Gaussian distribution.

flab	fhd	fcap_e	hhd_u	hhd_r	Ecrop	Icrop	Eoagr	Ioagr	Eagrib	Iagrib	Eoind	Ioind	prserv	fsav
0.51	0.99	1.25	0.33	0.24	2.94	4.31	3.09	2.74	2.44	2.31	6.16	4.21	1.35	7.98

Table 17: Standard deviations.

	flab	flnd	fcap_e	hhd_u	hhd_r	Ecrop	Icrop	Eoagr	Ioagr	Eagrib	Iagrib	Eoind	Ioind	prserv	fsav
flab	1														
flnd	0.43	1													
fcap_e	0.38	0.62	1												
hhd_u	0.70	0.00	-0.05	1											
hhd_r	-0.29	-0.62	-0.90	-0.01	1										
Ecrop	0.42	0.57	0.74	0.16	-0.81	1									
Icrop	0.54	0.54	0.67	0.26	-0.68	0.89	1								
Eoagr	-0.20	-0.01	0.09	0.01	-0.06	0.15	0.10	1							
Ioagr	-0.24	-0.15	-0.19	0.05	0.22	-0.06	-0.06	0.93	1						
Eagrib	0.10	0.22	0.30	0.17	-0.31	0.53	0.50	0.87	0.79	1					
Iagrib	0.45	0.53	0.69	0.31	-0.74	0.92	0.84	0.38	0.19	0.71	1				
Eoind	0.17	0.59	0.87	-0.11	-0.94	0.80	0.69	0.05	-0.23	0.28	0.71	1			
Ioind	0.18	0.59	0.85	-0.06	-0.94	0.80	0.69	0.11	-0.17	0.32	0.75	0.99	1		
prserv	0.16	-0.27	-0.78	0.45	0.64	-0.39	-0.23	-0.22	0.02	-0.19	-0.33	-0.68	-0.66	1	
fsav	0.43	0.43	0.75	0.31	-0.86	0.72	0.63	-0.01	-0.25	0.21	0.66	0.85	0.85	-0.49	1

Table 18: Correlations.