ADAPTIVE SAMPLING WITH FEATURE ELIMINATION FOR AGENT-BASED MODELS

by

Pelin Yurdadön

B.S., Industrial Engineering, Middle East Technical University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2020

ACKNOWLEDGEMENTS

The last three years have thought me a lot while bringing me at this point. I value this study as it reminds me to be brave and embrace my desires. Finally, I am glad to contribute to academic literature.

I would like to put my gratitude towards all people helping this dissertation happen with their constant care and support. Countless times I found myself stuck in a detail keeping me apart from moving on. I want to thank my advisor Gönenç Yücel, who patiently put me on track every single time I was lost and always supported me. His insights and guidance carried this thesis on a certain level. Also, I sincerely thank Mert Edali for his intellectual contribution, friendship and never-ending support even from overseas. Additionally, I would like to thank my jury members, Mustafa Gökçe Baydoğan and Uzay Çetin, who came together for me in a very short amount of time regardless of these very unprecedented times.

Throughout these three years, my path crossed with wonderful people. Mehmet, Zehra, Ömer, Arifcan, and my ex-partners are only some of them. I am deeply grateful to them as they were huge emotional and intellectual support for me. I must especially thank Esra. Esra, you are my luck and treasure. We went through various adventures together and I am sure we will continue to enjoy dancing on the waves of life together.

Last but not least, I must express my greatest gratitude to my family; Lale, Mehmet, Pınar, and Fiko. Despite all emotional roller-coasters I made them go through, I always feel their love and support for me. I am lucky to have them and I love them a lot. I am also grateful to Şivegül and Nermin for always being by my side.

I gratefully acknowledge the financial support of TÜBİTAK through the domestic scholarship program 2210-A.

ABSTRACT

ADAPTIVE SAMPLING WITH FEATURE ELIMINATION FOR AGENT-BASED MODELS

In this thesis, we propose an efficient and user-friendly metamodeling procedure simultaneously incorporating adaptive sampling with feature elimination in order to successfully capture the relationships between the parameters and the output of simulation models in the presence of insignificant model parameters with respect to an output of interest. In this procedure, Random Forest metamodel is utilized. While adaptive sampling efficiently yields the metamodel with a high quality training set, feature elimination promotes the performance and efficiency of the metamodeling procedure by reducing complexity and dimensionality due to insignificant parameters. In this respect, the proposed metamodeling procedure is also potentially applicable to high-dimensional simulation models. For illustrative purposes, the proposed procedure is applied to an agent-based segregation model. It is observed that adaptive sampling strategy can generate metamodels with higher predictive performance compared to input-oriented and random sampling strategies. Yet, the metamodel accuracy is considerably influenced by the *mtry* parameter of Random Forest metamodel and the presence of the insignificant simulation model parameters. However, experimental results show that the proposed procedure successfully alleviates the drawbacks of insignificant simulation parameters on the metamodel and sampling by eliminating them, and reduces the dependency of the metamodel performance to the *mtry* parameter. As a result, the proposed metamodeling procedure stands out as a robust and efficient way to produce a metamodel that successfully approximates the dynamics embedded in the simulation model. The proposed procedure can be applied by researchers to reduce the time required for comprehensive model analyses, to gain awareness about the model behaviors and to detect the model parameters conditioning the model behavior.

ÖZET

ETMEN-TABANLI BENZETİM MODELLERİ İÇİN UYARLANABİLİR ÖRNEKLEME VE DEĞİŞKEN SEÇİMİ

Bu tezde, incelenen çıktıya göre önemsiz olan model parametrelerinin varlığında, model parametreleri ile model çıktısı arasındaki ilişkileri başarılı bir şekilde yakalamak için uyarlanabilir örnekleme ve öznitelik eleme stratejilerini eşzamanlı kullanan verimli ve kullanıcı dostu bir metamodelleme prosedürü öneriyoruz. Bu prosedürde, Rassal Orman metamodeli kullanılmakta. Uyarlanabilir örnekleme, metamodeli eğitmek için yüksek kaliteli bir veri setini verimli bir şekilde oluştururken; öznitelik eliminasyonu, önemsiz parametreler nedeniyle oluşan karmaşıklığı ve boyutsallığı azaltarak metamodelleme prosedürünün performansını ve verimini arttırır. Bu bağlamda, önerilen metamodelleme prosedürü potansiyel olarak çok parametreli simülasyon modelleri için de uygulanabilir. Örnekleme amacıyla, önerilen prosedür etmen-tabanlı bir Ayrışma modeline uygulandı. Uyarlanabilir örnekleme stratejisinin, girdi odaklı ve rastgele örnekleme stratejilerine kıyasla daha yüksek tahmin performansına sahip metamodeller üretebileceği gözlendi. Yine de, metamodelin doğruluğu Rassal Orman metamodelinin *mtry* parametresi ve önemsiz simülasyon parametrelerinin varlığından önemli ölçüde etkilenir. Ancak, deney sonuçları, önerilen prosedürün önemsiz simülasyon parametrelerinin metamodel ve örnekleme üzerindeki dezavantajlarını başarılı bir şekilde ortadan kaldırdığını ve metamodel performansının *mtry* parametresine bağımlılığını azalttığını göstermektedir. Sonuç olarak, önerilen metamodelleme prosedürü, simülasyon modeline gömülü dinamikleri başarılı bir şekilde yaklaşıklayan bir metamodel üretmek için gürbüz ve verimli bir yol olarak öne çıkıyor. Önerilen prosedür, kapsamlı model analizleri için gereken süreyi azaltmak, model davranışları hakkında farkındalık kazanmak ve model davranışını düzenleyen model parametrelerini tespit etmek için araştırmacılar tarafından uygulanabilir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

C	Set of labeled training candidates
c	Set of unlabeled training candidates
$elim\_iter$	Iteration number within the feature elimination procedure
h	Number of instances selected
I	Iteration budget
$iter^*$	Iteration at which feature elimination procedure is activated
$iter$	Iteration number within the metamodeling procedure
k	Number of features
L	Number of decision trees
M	Metamodel
mm	Mtry multiplier
$mtry$	Random Forest parameter defining the number of variables randomly sampled at each split
N	Number of instances
$ntree$	Random Forest parameter defining the number of trees in the random forest
p^*	A fraction determining the number of features considered for elimination
p	A predefined fraction determining the maximum number of features that can be eliminated at once
R	The number of replications
T	Training set
U	Unlabeled data set
u	The number of unlabeled instance in U
v_u	Uncertainty value
$\hat{y}$	Predicted simulation model output
y	Actual simulation model output

LIST OF ACRONYMS/ABBREVIATIONS

ABM	Agent-Based Model (i.e. Agent-Based Simulation)
AdS	Adaptive Sampling
AdS&Fe	Adaptive Sampling with Feature Elimination
AdS&Fe_mm1	Adaptive Sampling with Feature Elimination where *mm* equals to 1
AdS&Fe_mm2	Adaptive Sampling with Feature Elimination where *mm* equals to 2
AdS_cvar	Adaptive Sampling based on coefficients of variation
AdS_range	Adaptive Sampling based on range
AdS_sd	Adaptive Sampling based on standard deviation
CART	Classification and Regression Trees
DSM	Segregation Model with Dummy Parameters
LASSO	Least Absolute Shrinkage and Selection Operator
LHS	Latin Hypercube Sampling
MSE	Mean Squared Error
NRFE	Nonrecursive Feature Elimination
OOB	Out-of-bag
OsS	One-shot Sampling
QBC	Query-by-Committee
RdS	Random Sampling
RF	Random Forest model
RFE	Recursive Feature Elimination
RMSE	Root Mean Square Error
RSS	Residual Sum of Squares
SM	Segregation Model
SVM	Support Vector Machine
VIM	Variable Importance Measure

1. INTRODUCTION

In the last couple of decades, agent-based modeling (ABM) has been used to comprehend various phenomena in many domains [1, 2]. In consistence with its broad use, Arroyo *et al.* [3] have already touched on the rising popularity of ABM using empirical data. The promising potential of ABM lies behind its power to represent the dynamics of complex adaptive systems and to monitor the changes in their behaviors with respect to interventions.

In complex adaptive systems, autonomous components interact and adapt at individual or population levels [2]. Therefore, the dynamic behavior of the system roots from the heterogeneity and autonomy of its components. Macal and North [2] emphasize that agent-based simulation can be distinguished with its capability to model heterogeneous agents and revealing the self-organization of agents. Agents can learn from experiences and adapt their behaviors in a feedback mechanism through interactions with other agents. Also, Lempert [4] describes agent-based simulation as a significantly capable tool in simulating social systems since it can reveal difficult-to-show phenomena without any need for mathematical formalisms. As a result, ABM stands out as a promising tool to gain insights about complex systems.

Yet, the exploration and analysis of simulation models can be challenging due to the nonlinear interactions among model variables and the number of parameters. Nonlinear relationships between variables may lead to rich system behaviors with several tipping points and counter-intuitive dynamics. Detecting all of behaviors of a complex system and identifying their causes may demand exhaustive efforts. Furthermore, the scale of the output data of ABMs can easily become difficult-to-handle [3, 5] to perform analyses. Considering that the complexity of the model and the volume of the model outputs tend to increase with the number of parameters in the model, it becomes even more challenging to find out the parameter combinations conditioning specific behaviors in high-dimensional models. Lee *et al.* [1] mention that the large-

scale output data brings about challenges in the detection of interesting results and the assessment of model behaviors under different conditions. Therefore, additional tools such as statistical methods and machine learning techniques are suggested to enhance model exploration and analysis [1, 3, 5]. Also, the sufficient number of simulations to perform model exploration analysis tends to increase with the number of parameters. As the parameter space enlarges, more parameter value combinations require to be evaluated with simulations, and the total simulation time increases. As a result, the exploration and analysis of simulation models become easily complicated and computationally costly.

The metamodeling with adaptive sampling emerges as a potentially fruitful approach to deal with such problems and enhance the model exploration and analysis. A metamodel serves as a time-efficient and simplified model of any comprehensive model that relates input variables to output variables. Metamodeling is mainly based on the trade-off between accuracy and time-efficiency. Crombecq *et al.* [6] explain that metamodeling provides a fast and satisfactory approximation to handle the long computational time of an underlying model. In the context of simulation models, metamodels can easily represent approximate dynamics of the model without baring with long simulation execution times to reveal accurate dynamics. Edali and Yücel [7] summarize the main purposes of metamodeling of ABMs in relation with model exploration, prediction, optimization, and validation and verification. Metamodeling can be used to understand the input-output dynamics, predict the output for a parameter value combination in a computationally efficient way without a significant loss in accuracy, find the optimal parameter combinations leading to maximum (or minimum) level of a specific feature of the output, and validate and verify the simulation model. In all cases, training a metamodel that can successfully capture the behavior of the model plays a significant role.

One of the ways to promote the performance of a metamodel is to feed the metamodel with an informative data set. In this sense, adaptive sampling can supply the metamodel with useful information that can continuously improve the performance

of the metamodel. On top of that, the size of data set required to successfully train the metamodel tends to decrease as the quality of the data set grows. In adaptive sampling (i.e. active learning), the initially small training set continuously grows with the introduction of new data instances in an iterative procedure. The selection of the new instances is guided by the previous iterations of the procedure so that a small but high-quality data set can be composed. To be clear, the increased quality of the training set can compensate for the negative effect of the decrease in the size of the training set on the metamodel performance. Therefore, adaptive sampling contributes to the time-efficiency of the metamodeling applications by reducing the number of simulations needed for training set construction. Furthermore, by monitoring the selected instances during the adaptive sampling procedure, Edali and Yücel [7] show that adaptive sampling can highlight the boundaries between different behaviors of a simulation model. Moreover, since the smart selection of data points generally leads to considerable improvements in metamodel performance with limited data, adaptive sampling can evade the requirement for a large data set and avoid high computation costs.

In literature, studies conducted by Edali and Yücel [7] and Lamperti *et al.* [8] stand out as comprehensive applications of the metamodeling approach with an adaptive sampling method for ABMs with two different motivations. While Edali and Yücel [7] aim to facilitate the exploration of model behavior by focusing on understanding the relationship between parameters and output variables and achieving accurate output predictions, Lamperti *et al.* [8] intend to propose an efficient model validation and parameter verification method. These applications show that metamodeling with adaptive sampling is a promising tool to explore and analyze an agent-based simulation model efficiently. Yet, Edali and Yücel [7] discuss that as the number of parameters in an ABM increases, the execution time of the whole process of metamodeling with adaptive sampling goes up exponentially due to the curse of dimensionality. Also, the large number of parameters can increase the complexity of the metamodel and negatively influence the efficiency of the metamodel. In their study, van der Hoog [9] draws attention to the need for complexity reduction of the metamodel for the sake of the efficiency of their application.

Moreover, in high-dimensional complex models, some parameters of the model can be redundant or slightly influential depending on the model output monitored. The presence of such insignificant parameters unnecessarily complicates the model exploration and analysis with respect to a specific model output. They cause extra computational cost by triggering the curse of dimensionality and generating excessively large simulation output data. Due to the vast multi-dimensional parameter space and large-scale output data caused by insignificant parameters, it becomes more challenging to detect significant parameters conditioning the dynamic behavior of the system.

Consequently, alternative metamodeling approaches for complex ABMs that can successfully deal with a large number of parameters are yet to be developed. In this study, we aim to propose an efficient metamodeling procedure for ABMs considering its applicability also for high-dimensional ABMs. To overcome the drawbacks of the presence of insignificant parameters and enhance the efficiency and accuracy of the procedure, adaptive sampling and feature elimination methods are simultaneously utilized with metamodeling.

The objective and the extent of this study are presented clearly in the following section. While the background research is shared in the third section, a new metamodeling approach with adaptive sampling and feature elimination is introduced in the fourth section. Afterward, an application of the proposed metamodeling procedure to a well-known agent-based simulation model is shown and the results are discussed in the final sections.

2. PROBLEM DESCRIPTION & OBJECTIVES

Recent studies show that metamodeling with adaptive sampling has emerged as a capable tool to reduce computational burden and promote the interpretability of the input-output relationships. Yet, the efficiency of such methods for high-dimensional models can suffer from the curse of dimensionality. The number of instances required to train a successful metamodel, the computational cost of training the metamodel, and the design space of the model exponentially increase with the number of features [10]. Hence, the construction time of the training set and the execution time of a metamodeling approach with adaptive sampling are dramatically influenced by the number of parameters in a simulation model. Moreover, gaining insight into input-output relationships and identifying main triggers leading to a specific output become more challenging in high-dimensional models due to the vast parameter space and large-scale output data [1, 5]. Finally, as mentioned previously, some parameters of a large model can be redundant with respect to a model output monitored. Their presence further causes inefficiency and inconvenience in exploring and analyzing the behaviors of model due to their contribution to the curse of dimensionality.

In light of previous discussions, our objective is to propose an efficient metamodeling approach combined with adaptive sampling and feature elimination that can facilitate model exploration and analysis while dealing with the complexity and inefficiency induced by insignificant parameters. An adaptive sampling method accompanies metamodeling in order to achieve computational efficiency and contribute to model performance. Besides, eliminating insignificant features can improve the metamodeling and adaptive sampling processes by reducing both the complexity of the metamodel and dimensionality of the parameter space. As insignificant features are reduced from the metamodel, adaptive sampling is conducted on a space with lower dimensions. Hence, adaptive sampling gradually focuses more on the significant features to efficiently enhance the performance of the metamodel. Consequently, a refined metamodel without insignificant features can disclose the model dynamics better and

potentially faster. Elimination of insignificant features also promotes the use of the proposed metamodeling procedure for high-dimensional simulation models.

To be clear, the main motivations behind the efficient metamodel training are:

- To gain insights into the emergent behavior of the model by satisfactorily summarizing the input-output dynamics with the metamodel.
- To reduce the computational cost due to the large number of simulations required to train metamodel successfully.
- To reveal important parameters considerably restricting the dynamic behavior of the model by eliminating parameters that do not have a crucial impact on the model behavior.

In consistence with the motivations, this study has several potential benefits. Firstly, researchers (such as model analysts) can perform model exploration and analysis more efficiently with the guidance of the proposed procedure. The resulting metamodel can point out the important model parameters and successfully mimic dynamic relationships between model parameters and the output. The proposed metamodeling procedure can further encourage the use of high-dimensional complex ABMs by reducing the complexity and speeding up metamodeling process. On top of that, if the metamodel is accurate enough, it can also serve as a preliminary testing ground to observe the effects of interventions in policy-making. Finally, in the context of model validation and verification, the proposed procedure can provide insights about the effective ranges and the effects of model parameters.

As a result, to develop an efficient metamodeling procedure preferred by modelers/analysts with different motivations, in this study, we will mainly focus on

- acquiring a metamodel with high predictive power by correctly addressing the nonlinear relationships between parameters and the output represented in the simulation model,

- maintaining a user-friendly application with a minimum requirement for additional processes by offering a self-sufficient and compact metamodeling procedure,
- achieving a fast application,
- providing a reasonable level of interpretability about parameters and model dynamics by identifying significant parameters.

3. BACKGROUND

Barton [11] considers the choice of the metamodel, the selection of data points to train the metamodel, the performance assessment of metamodel fitted to the experimental data, and the adequacy of the metamodel among the major issues in a metamodeling approach. To ensure the accuracy and efficiency of a metamodeling approach with adaptive sampling, the selection of a proper metamodel and sampling strategy is of great importance. As Kleijnen [12] warns a computationally demanding metamodel may negatively influence the efficiency of adaptive sampling, the coherence between the metamodel and sampling strategy should be considered.

Also, the identification of significant features (i.e. input variables, model parameters) in a metamodeling approach deserves attention. A carefully adapted feature selection strategy can enhance the efficiency of the metamodeling approach by reducing the dimensionality and complexity. Also, it can promise the metamodel with higher accuracy as it guides the metamodeling approach to focus on significant features. On the other hand, extra computations the feature selection strategy demands may cause efficiency loss.

As a result, it is important to carefully select and coherently utilize the metamodel, sampling strategy, and feature selection strategy to achieve a sufficiently accurate and efficient metamodel of a simulation model. The following subsections introduce and compare various metamodels, sampling strategies, and feature selection strategies. The most appropriate candidates for the metamodel, sampling strategy, and feature selection strategy are discussed in detail.

3.1. Metamodeling

Kleijnen [13] describes a metamodel as an approximate model of the input-output relationships embedded in a simulation model. Metamodels explicitly represent the

functional relationship between inputs and outputs in a computationally efficient way since they provide approximate results [11]. Rather than replicating a simulation multiple times for a parameter value combination to obtain an accurate output, a metamodel can be used to produce a sufficiently accurate output in a relatively short time.

Various metamodeling applications for ABMs can be seen in the areas of simulation optimization [11,13], model exploration and analysis [7,14], model calibration [8]. All these studies rely on a successful metamodel that can capture the implicit dynamics of the underlying simulation model with minimum accuracy loss. Methods such as kriging, support vector machines (SVM), random forest (RF), boosting, and neural network are commonly utilized in metamodeling approaches in the literature.

3.1.1. Selection Criteria of Metamodels

Crombecq *et al.* [6] highlights that the possible functional relationships embedded in a data set should be searched wisely so that the function mimicking the model behavior most accurately can be found. For instance, decision trees search for the best function by splitting the data set recursively while linear regression models try to find a linear function minimizing the residuals. Therefore, the capability of a metamodel to address the implicit relationships in a data set plays an important role to achieve a successful approximation of a simulator.

Also, the purpose of applying a metamodeling procedure can influence the choice of the metamodel. For instance, a hands-free metamodeling application would prioritize a learner that does not demand extensive parameter tuning and data processing before application. If gaining insights about the mechanisms of the dynamic behavior of the simulation model is sought, then the metamodel is expected to interpretable. If the computational efficiency of the metamodeling procedure is desired, metamodels requiring heavy computation times like neural networks can be avoided. Regarding these discussions, the following points are considered in selecting an appropriate metamodel to use in this study:

- High predictive power: ABMs reveal nonlinear relationships between parameters and outputs and interactions between variables constrained by parameters. To construct successful metamodels of ABMs, these aspects of ABMs should be properly addressed. Therefore, machine learning methods such as RF, boosting, neural network and SVM can be advantageous due to their ability to reflect nonlinear dynamics in a data set.
- Fast application: The computational cost of metamodels tends to increase with the size of the training set and the number of features existing in the data set. Therefore, the time-efficiency of the metamodel used in the procedure considerably influences the overall speed of a metamodeling procedure. Especially, in an adaptive sampling design where the metamodel is re-trained iteratively, the computational cost of metamodel training can become more visible. If an efficient metamodel is not employed, any changes in the training set can incur significant time cost in the metamodeling procedure.
- User-friendly application: The target user of the proposed procedure consists of mainly modelers and analysts from different backgrounds and with various expertise. They can use our procedure to explore a model fast and easily and manipulate their analysis strategies. Hence, proposing a user-friendly design while maintaining a satisfactory prediction performance seems crucial to promote the use of the proposed procedure. We consider two main aspects of this issue, which are hands-free application and self-sufficient application. Regarding hands-free application, it is more convenient to select a metamodel that does not demand heavy efforts for parameter tuning and that can perform well on various data sets without requiring data preprocessing. Secondly, metamodels with additional features such as variable importance estimates and prediction error estimates may contribute to the self-sufficiency of the proposed procedure. Easy and cheap computation of such estimates during metamodel training can reduce the need for additional methods to perform adaptive sampling and feature selection strategies.
- Interpretability: It is aimed that the proposed metamodeling procedure serves for acquiring a reasonable level of understanding of the underlying dynamics of the simulation model. At first sight, decision trees and linear models stand out as

they justify their predictions by establishing easily tractable rules about the relationships between inputs and outputs. However, decision trees may have stability and overfitting problems, and linear models are not even among the alternatives due to the aforementioned accuracy concern. On the other hand, the loss of interpretability of more complicated techniques such as neural networks, boosting, RF, or SVM can be restored up to some level by different rule extraction techniques [7,15]. Among these techniques, RF and boosting stand out as they easily estimate reliable variable importance scores and contribute to the interpretability of the model without any need for extra computations.

Based on these four concerns, ensemble methods such as RF and boosting seem to be the most adequate metamodel candidates for this study. The efficient and successful use of RF in [7] and boosting in [8] confirm that both are relatively fast and good performing learners. Furthermore, they can be easily utilized in adaptive sampling methods such as query-by-committee and uncertainty sampling. Also, they can contribute to the feature elimination process by estimating variable importance. Such additional benefits of the ensemble methods promote the self-sufficiency of the metamodeling procedure.

Apart from the common strengths of RF and boosting, RF is widely used in prominent feature elimination studies and in many studies related to RF importance measure [16–21]. Besides, Verikas *et al.* [22] point out the high performance of RF by saying that the number of cases where RF is outperformed by alternative techniques is less than the number of cases where it outperforms. Regarding its popularity in feature elimination and variable importance literature, and its competitive performance among alternative machine learning techniques, RF stands out as the strongest candidate for the metamodel in the frame of this study. The advantages of RF metamodel is further discussed below in detail.

3.1.2. Advantages of Random Forest

Random forest (RF) is a widely used ensemble learning method consisting of different and independent decision trees. While benefiting from the tree predictions with low bias, RF also alleviates the overfitting risk of a single tree and provides stable predictions. Verikas *et al.* [22] show the popularity of RF by listing a substantial number of studies applying RF in various domains. This implies that RF can be adapted to data sets with different characteristics and perform successfully in many of them. Besides, RF is intensively utilized in many feature selection and adaptive sampling algorithms. Hence, the technical capabilities, predictive performance, and potential uses of RF lie behind its popularity.

First of all, RF can grasp nonlinear relationships between inputs and output variables, and interactions between input variables through recursive binary splitting. RF can perform well in the presence of categorical and continuous variables in the same data set, correlated variables, noise variables or missing data [16]. Hence, the need for preliminary processes such as encoding the categorical variables, removing correlated variables or filling the missing data before the training is not necessary. Moreover, it can handle a large number of variables with limited training data even when the number of variables exceeds the number of instances in the data set [16, 21]. Since RF is an ensemble method, its predictions are generally more stable and accurate compared to a single tree learner [23]. On top of that, RF can perform well even with little effort to tune its parameters [16, 24]. The number of trees grown in RF (*ntree*) and, especially, the number of randomly selected splitting variable candidates on the nodes of the trees of RF (*mtry*) are the most influential parameters in its predictive performance.

In addition to its technical capabilities and satisfactory predictive performance, RF can be easily applied in different approaches since it can provide variable importance estimates, prediction error estimates, and tree-based predictions. For example, in literature, highly cited several feature selection strategies [16, 21, 25] depend on variable importance estimates of RF. Similarly, prediction error estimate of RF can be helpful

especially when splitting the available data set into training and test sets is not viable due to scarcity of the data. Finally, tree-based predictions of RF can be compatibly used with adaptive sampling strategies such as query-by-committee.

Overall, these features bring quite a fame to RF in many scientific studies and various domains. Thanks to its popularity, it is expected that the target users of this study such as researchers, analysts and modelers can easily become familiar with RF and learn to apply the RF metamodel with a small effort even though their knowledge in machine learning is limited. Also, RF can be easily applied in well-known programming languages such as R and Python through many reliable libraries and most of the resources are easily accessible. The availability of a large number of RF applications and information about its technical details can encourage the target users such as analysts and modelers to employ the metamodeling approach utilizing RF metamodel.

3.1.3. Random Forest

Random forest (RF) consists of de-correlated decision trees so that low bias and low variance in output prediction could be achieved at the same time by simply taking the average of the individual predictions over the trees for regression problems and by selecting the most frequent class predicted by individual trees for classification problems. Each tree in RF is trained on different bootstrap samples. In bootstrapping, N many instances are randomly selected with replacement from the original data set with N many instances. While all of the instances are unique in the original data set, the same instances might be chosen several times in a bootstrap sample. Therefore, each bootstrap sample remains different from the original data set. According to statistical findings, as Breiman (2001) mentions, about two-third of the instances in the original data set is chosen on average in bootstrapping. In other words, each tree in RF is trained on randomly chosen two-thirds of the original data set. If there are *ntree* many trees in RF, then *ntree* many different bootstrap samples are generated to train each tree. Therefore, the variety of trees is maintained.

In addition to bootstrapping, the selection of splitting variables contributes to variety in RF. Splitting variable at a node of a tree is chosen from a random set of variables. Breiman [26] explains that the randomness in determining the set of potential splitting variables serves to minimize the correlation between trees and to improve accuracy. At each node of a tree, among randomly chosen *mtry* many variables, the most explanatory variable is selected to further grow the tree. In general, *mtry* is chosen smaller than the number of variables so that a diverse set of trees can be grown and even the interaction effects of less significant variables with other variables might be captured. Strobl *et al.* [20] explain that random selection of splitting variables might lead to the global optimality of RF despite sub-optimal splits. Breiman [26] also mentions that this selection method increases the robustness of RF to outliers and noise, and the speed of RF compared to bagging and boosting.

Bootstrapping and the selection of the splitting variable from a random subset of variables stand out as the most effective characteristics of RF in terms of providing unique uncorrelated trees. Also, the fully-grown structure of RF trees contributes to variety in trees. While single decision tree approaches generally need pruning to handle overfitting, fully-grown trees do not cause overfitting in RF since the prediction is made based on the aggregate result of all trees. The aggregation of the tree predictions in RF is obtained by majority voting (i.e. the most frequent class is selected) for classification problems (categorical output) and by averaging tree predictions for regression problems (continuous output). The aggregation of fully-grown uncorrelated trees with high variance yields a low-bias and low-variance ensemble. The more different trees are generated; the more potential decrease in variance is gained.

Another prominent feature of RF is the assessment of the model performance without any strict need for performing cross-validation or the validation set approach. As discussed above, for a given tree, one-third of the original data set (i.e. out-of-bag data set, OOB data) is not used in training the tree. This allows OOB data to be utilized as a substitute for a test set. Predictions of the instances in OOB data can be calculated by the trained tree. Given that RF is an ensemble of *ntree* many trees,

overall *ntree* many diverse OOB data sets exist, and each instance of the original data set is included in about *ntree*/3 many OOB data sets. This means that *ntree*/3 many predictions are obtained for each instance in the original data set. Predictions for each instance are aggregated by taking the average or majority vote for regression and classification, respectively. After the calculation of OOB predictions for each instance, overall OOB MSE (for regression) and classification error (for classification) over the original data set are computed and named as OOB error. According to empirical evidence, Breiman [26] states that OOB error can be used as an accurate substitute for a test error. Yet, Breiman also warns that the small number of *ntree* may lead to overestimation of the error rate. As long as *ntree* is sufficiently large, OOB error can be used as an unbiased estimator of a test error.

Each decision tree in RF divides variable space linearly by recursive splitting. Each decision rule points out a different partition of the variable space. Therefore, nonlinear relationships between input and output variables can be pointed out with different decision rules. In RF, each tree constructs different decision rules according to different subsets of the data. On the other hand, RF can cover the whole variable space and capture the interactions among variables by gathering different decision trees. From another perspective, the strict decision boundaries of a decision tree are relaxed in RF. Hence, unlike a single decision tree, RF is unable to provide decision rules due to the aggregation of tree predictions. As a consequence, RF shows better performance than a single decision tree at the expense of losing interpretability. Still, the trees of RF can be analyzed, and their rules can be accessed if needed. Also, with rule extraction techniques, interpretability loss of RF can be recovered up to some point [27].

Even without rule extraction, RF is not a completely black-box approach. There are two main variable importance measures used by RF to assess the effect of the variables on the output. The first measure is called permutation importance measure. The importance of a variable is measured by the change in prediction accuracy after the values of the corresponding variable in OOB data sets are randomly shuffled while the values of the other variables are kept the same. Basically, after replacing the

true values of a variable with random noise, OOB predictions are calculated again. If the shuffled values do not cause a significant loss in prediction accuracy, the variable is considered irrelevant. If the variable is significant, a noticeable decrease in the accuracy of OOB predictions is expected. Prediction error on OOB data is recorded as the misclassification error for classification and MSE for regression, and the errors are averaged over all trees in RF. Hence, the permutation importance measure is described with %MSE for regression and misclassification error for classification. The second measure is based on impurity reduction resulted from splitting on the feature. The magnitude of a decrease in node impurity determines the importance of a variable. Consistently, the importance is measured as the mean decrease in the Gini index for classification and the mean decrease in RSS (Residual Sum of Squares) for regression. Yet, Gini index-based importance measure is shown to be biased for classification if the number of levels of categorical variables or the scales of the continuous variables varies [20]. Moreover, in the presence of high correlations between variables, Strobl *et al.* [20] remark that RF-based variable importance measures show a tendency towards correlated variables. Although these findings raise questions on the reliability of RF importance measures, they can still provide useful importance scores, if used carefully.

3.2. Sampling Strategies

In addition to selecting a proper metamodel, the selection of a sampling strategy feeding the metamodel influences the performance of a metamodeling procedure. Both Barton [11] and Crombecq *et al.* [6] touch on the importance of data set to yield metamodels with sufficient predictive power. Large data sets generally contribute to achieving sufficiently accurate metamodels. However, when data generation is computationally costly, Salle and Yıldızoğlu [14] draw attention to the trade-off between the size of the training data set and the predictive performance of the metamodel. The high cost of collecting and evaluating instances can be a disincentive to obtain a large data set. To compensate for the potential accuracy loss due to reducing the size of a data set, promoting its informativeness through smart sampling methods gains an impor-

tant role. Adaptive sampling (i.e. active learning, sequential sampling)[1] is one of such methods providing useful instances for the training set to obtain successful metamodels at a relatively small computational cost. Since this study focuses on metamodeling of a simulation model, data sets are basically composed by parameter value combinations with their corresponding output value. For this reason, all sampling methods aims to obtain simulation parameter configurations from the parameter space.

When a large data set is available or the computational cost of obtaining a large set is affordable, one-shot approaches can be utilized. In a one-shot approach, a training data set is selected at once and the metamodel is trained on this data set. Note that the metamodel is not involved in the construction of training set. To increase the efficiency of one-shot sampling, space-filling methods can be employed. Space-filling designs such as Latin hypercubes (LHS) and Quasi-random Sobol sampling (based on Sobol' sequences) are easily applicable and provide parameter configurations (i.e. input variable configurations) successfully covering the whole parameter space. LHS selects parameter configurations from mutually exclusive and exhaustive intervals with equal probability corresponding to each parameter (i.e. input variable) [13]. LHS randomly combines the selected values of each parameter to determine parameter value combinations while maintaining one-dimensional projection properties. It adopts a noncollapsing design. Therefore, a change in the number of parameters does not influence the distribution of instances on one-dimensional projections and parameter configurations remain unique. Yet, since parameter values are combined randomly, a correlation between the columns of LHS can be observed. To improve the space-filling property of LHS across the parameter space, optimization-assisted LHS algorithms are utilized. Sheikholeslami and Razavi [28] mention two widely used LHS algorithms where the minimum distance between instances is maximized (maximin LHS) and the correlations between the pairs of the columns of the data matrix are minimized. Like LHS, Kucherenko *et al.* [29] explain that Sobol sampling aims to provide uniformly distributed data points in a fast way. Even a small data set generated by Sobol sam-

[1] Although adaptive sampling, active learning, and sequential sampling are slightly different notions, they all imply iterative model training. Therefore, in this study, we consider them interchangeable.

pling can provide uniform-looking filling over the parameter space. Based on several uniformity criteria, they state that Sobol sampling provides better data sets compared to LHS unless an optimized LHS is used. Yet, the findings in [29] favor Sobol Sampling over optimized LHS in terms of sampling efficiency and uniformity of data points, especially when a large number of data points are selected from a high-dimensional parameter space.

Nevertheless, one-shot sampling methods applying Sobol sampling and LHS can cause inefficiency. Sheikholeslami and Razavi [28] mention that generally an unnecessarily large sample size is selected to alleviate the risk of undersampling of the parameter space since it is required to specify the sample size before metamodel training. In case of undersampling in LHS, either a new, larger data set should be generated at a significant computational cost or new samples are simply added to the current data set at a cost of distorting the Latin hypercube structure of the training set. Although incremental increase in the data set is much easier in Sobol sampling compared to an optimized LHS [29], the selection and evaluation of additional data points to overcome undersampling still require extra effort. Besides, the presence of insignificant parameters worsens the efficiency of one-shot sampling methods by enlarging the parameter space. As the parameter space enlarges, both the complexity of the method and the sample size required to sufficiently cover the parameter space increase. Hence, insignificant parameters cause unnecessary computational cost by complicating the one-shot sampling process.

Unlike one-shot designs, in adaptive sampling, an initially small training set is iteratively built up with the addition of new instances. These instances are expected to enhance the metamodel performance. Adaptive sampling aims to reduce the overall training set size required to train a successful metamodel by using historical information of the data and metamodels. The predictive performance of the metamodel is efficiently promoted with a comparatively small but highly informative data set. Depending on the purpose of metamodeling, input-oriented or output-oriented adaptive

sampling methods[2] can be utilized. Sheikholeslami and Razavi [28] describe input-oriented sampling as model-free since the selection of the next instances is not guided by the metamodel output. The main concern of input-oriented sampling is to enhance the coverage of the parameter space through iterative sample selection. On the other hand, in output-oriented sampling, new instances are selected based on their potential to improve the metamodel performance. Instead of equally sampling over all areas of the parameter space, output-oriented sampling mostly tends to sample from the areas difficult to predict their outputs such as the boundaries between different model behaviors [7]. Therefore, the model exploration also benefits from the output-oriented sampling [7, 13].

As a result, one-shot sampling methods are generally computationally more expensive than adaptive sampling methods as they require sufficiently large data sets to include parameter value combinations from the key regions of the parameter space. Besides, the preselection of the sample size carries the risk of oversampling and undersampling, both of which increase the marginal computational cost of one-shot designs. On the other hand, adaptive sampling methods can provide more accurate metamodels at a smaller computational cost compared to one-shot sampling. Furthermore, unless insignificant parameters are detected prior to the sampling, the adverse effect of insignificant parameters on the efficiency of the one-shot method cannot be avoided. However, adaptive sampling methods can be engaged with feature elimination strategies to reduce complications induced by insignificant parameters.

3.2.1. An Overview of the Sampling Strategies

Among adaptive sampling methods, output-oriented sampling methods serve better for our aim to explore and understand model dynamics in this study. Firstly, they can provide insights about model dynamics by pointing out tipping points and behavior boundaries. Secondly, they avoid computational cost due to the evaluation of

[2]Such differentiation between adaptive sampling methods is also made by Crombecq *et al.* [6]. They use exploration and exploitation based sequential sampling as equivalent to input-oriented or output-oriented sequential sampling, respectively.

parameter configurations with easily predictable outputs. Therefore, we will focus on output-oriented sampling and we will use the term of adaptive sampling to easily refer to output-oriented sampling.

In the simulation domain, although it is possible to easily obtain a large number of parameter value combinations (i.e. unlabeled instances), their evaluation in a simulation model can be quite costly. Considering that exploration and analysis of simulation models generally demand a large number of simulation experiments replicated several times, reducing the total number of evaluations in the simulation model is important to achieve a cost-efficient metamodeling process. Adaptive sampling methods compile up the knowledge obtained by previously trained metamodels and selected instances so that small and concise data sets are obtained to train successful metamodels. They satisfactorily contribute to the cost-efficiency of a metamodeling approach by improving the quality of the training set with respect to its size.

In adaptive sampling designs, firstly, a learner (i.e. metamodel) is trained on a relatively small training set. Then, the learner is used to identify the most useful parameter value combinations (i.e. unlabeled instances). Selected parameter value combinations are evaluated in the simulation model and their outputs are obtained so that they are added to the training set. Hence, three main issues should be discussed: the selection of instances to initially train the metamodel, the acquisition of unlabeled instances, and the identification and selection of the most informative unlabeled instances.

An adaptive sampling method is initialized with the selection of a relatively small data set to train the metamodel for the first time. Since adaptive sampling compiles the information gain starting from the first iteration, the initial training data influences both the metamodel performance and the quality of the final training set. Assuming that little or nothing is known about the regions of parameter space and model behavior, Crombecq *et al.* [30] state that a sampling method with space-filling property can convey significant information by providing unlabeled instances from various regions

of the parameter space. Therefore, both Sobol sampling and optimized LHS methods can be easily implemented to obtain the initial training set.

Following the metamodel training on the training set, informative unlabeled instances are specified according to the feedbacks from the metamodel. Unlabeled instances are acquired based on a query scenario. Settles [31] mentions three main scenarios, which are membership query synthesis, stream-based selective sampling, and pool-based sampling. In membership query synthesis, the learner generates unlabeled data points de novo rather than selecting them from the parameter space. On the other hand, stream-based selective sampling emerges as an alternative to membership query synthesis especially when the distribution of parameter combinations over parameter space is non-uniform. The main assumption here is that the cost of obtaining unlabeled instances is negligible. Instead of constructing queries, the learner determines whether to request the label of each unlabeled instance sampled one by one from a real data distribution. Lastly, pool-based sampling assumes that only a small set of labeled instances (training set) is available, whereas there is a large pool of unlabeled instances. The learner evaluates all unlabeled instances in the pool and selects the best query. Among these three scenarios, pool-based sampling seems to fit better to the simulation context. Since the evaluation of parameter combinations in a simulation model is costly in terms of time, the size of the training set is limited while generating a pool of unlabeled instances is comparatively cheaper. Consistently, pool-based sampling is also utilized in some metamodeling studies applied to simulation models such as [7, 8]. However, it is important to generate a pool sufficiently representing the parameter space. If the pool is biased or small, pool-based adaptive sampling procedure may induce a selection bias in the training set and strictly intervenes in the metamodel training. To overcome such problems, space-filling sampling methods (such as LHS and Sobol sampling) can be used to construct the pool.

In addition to the acquisition of unlabeled instances, selection of the most useful unlabeled instances requires careful attention. Crombecq *et al.* [30] agree that spending time to select the best parameter value combinations rather than immediately querying

the labels of ordinary parameter value combinations is favorable due to the high cost of evaluations by an oracle (i.e. a simulation model, in simulation domain). Settles [31] provides an overview of six general frameworks to select the best queries. First of all, according to uncertainty sampling, unlabeled data points whose predictions by the metamodel are the least certain are assumed to be the most informative. Uncertainty sampling can be applied to both classification and regression settings. The uncertainty of a prediction can be measured based on several approaches such as the difference between the predicted class probabilities or entropy in classification problems, or variance in the prediction in regression problems. On the other hand, query-by-committee (QBC) consults a committee of learners and identifies the most informative unlabeled instances by monitoring the highest disagreement among committee learners. Disagreement implies the uncertainty in the output prediction. Uncertainty value of an unlabeled instance can be evaluated as, for instance, the entropy of the predicted class probabilities over committee members or the difference between class votes of committee members for classification problems, and the variance in output predictions of committee members for regression problems.

Both uncertainty sampling and QBC are interested in finding informative instances. However, density-weighted methods also consider input distribution so that selected unlabeled instances are both informative and representative of the input space.[3] In case the distribution of unlabeled instances over the input space is not uniform, informative unlabeled instances are more likely to be selected from densely populated areas of the input space. So far, the informativeness of an unlabeled instance is mainly implied by the uncertainty of a model (i.e. a learning algorithm) about its predicted label. However, the informativeness of unlabeled instances can be assessed according to their expected effect on the model (i.e. model parameters of the learner or metamodel) regardless of their predicted labels. To be clearer, unlabeled instances leading to the maximum expected change in the model due to unveiling their labels are selected to query. However, the model error does not necessarily improve with this strategy [23].

[3]Input space implies the all possible combinations of input variables. In the context of this study, since each variable in the data sets corresponds to a parameter of a simulation model, input space and parameter space can be used interchangeably.

To ensure the improvement in the model performance, unlabeled instances can be evaluated based on their expected contribution to reducing the model error, instead of their influence on the model. Unlabeled instances that provide the highest expected error reduction are selected. Since each unlabeled instance should be handled separately and the learner should be trained over and over, the computational cost of this strategy can be significantly high. Instead of minimizing the expected model error, one can be interested in maximizing the reduction in expected variance. Indeed, reducing expected variance can also ensure a decrease in model error since other components of error such as bias and noise are not influenced by the model. In this framework, unlabeled instances minimizing the expected variance for the model are selected to query their labels.

Overall, the choice of a proper query selection strategy is not independent of the distribution of the parameter value combinations in the parameter space, the type, and the computation speed of the learning model. If the distribution of parameter value combinations is non-uniform, a query strategy framework considering global properties of the parameter space such as density-weighted methods, expected error reduction methods, or expected variance reduction methods can be preferable. Uncertainty sampling and QBC do not pay attention to the distribution of parameter value combinations on the parameter space. Unless the input distribution is uniform, they can keep querying outliers and low-density areas. However, expected error reduction framework is quite costly, especially when the learner is computationally demanding, since it requires re-training of the learner. Similarly, the utilization of variance reduction framework is limited due to the binding assumptions on the model class (i.e. the type of the learner). Depending on the formulation of the query selection criteria and the type of learner, the computational complexity of variance reduction framework can be overwhelming.

In addition to the query selection strategy, the number of queries selected at once can differ in adaptive sampling applications. In sequential sampling, data points are selected one by one. However, this can be time-consuming when parallel labeling is

available or the time cost of training the metamodel is not ignorable. On the other hand, in active batch learning, a group of data points is selected each time. After the labels of all group members are obtained, they are added to the training set together and the metamodel is trained. Since batch sampling requires a smaller number of metamodel training and allows parallel labeling, it can be more efficient. Yet, the selection of the group members should be properly handled. While the most informative data points can be easily set as a group, more comprehensive approaches incorporating diversity among the members or density measures can be utilized in forming groups [23].

3.2.2. Advantages of Query-by-Committee

In this study, due to the time cost of evaluations in the simulation model, we would like to generate a small initial training set. However, prior information about the relationships between model parameters and the output is not available before metamodel training. Hence, uniformly distributed parameter value combinations over the parameter space are expected to provide the metamodel with comprehensive information from various regions of the parameter space [30, 32].

In a uniformly populated parameter space, the risk of spending a long time on outliers and uninhabited areas of parameter space is not present for QBC and uncertainty sampling. Since QBC and uncertainty sampling can be easily applied without demanding heavy computations, they become even more advantageous. Nevertheless, the proper query selection strategy should be chosen in coherence with the metamodel for the sake of the efficiency and accuracy of the metamodeling approach.

A random forest (RF) metamodel facilitates the utilization of QBC as it is already an ensemble of diverse decision trees. Also, QBC can be employed in a pool-based sampling scenario. Since the predictions of individual trees for each unlabeled instance in the pool can be easily obtained, the most informative unlabeled instances can be identified very fast based on the disagreement within the committee. The disagreement can be simply computed as standard deviation, range, or coefficients of variance

in predicted outputs over individual trees for continuous output, whereas entropy of predicted class probabilities or difference in class votes over the trees can be calculated for categorical output. Since QBC can be employed both in regression and classification settings, QBC promotes the applicability of the metamodeling approach for a wide range of simulation models. Moreover, Seung *et al.* [33] show that QBC can perform well even with two committee members. Due to its robustness, QBC is likely to work successfully with the default hyperparameters of the RF metamodel without an urgent need for tuning. As a result, QBC emerges as a competent framework to detect informative unlabeled instances.

3.2.3. Query-by-Committee

Query-by-committee (QBC) is a query filtering algorithm based on an ensemble (i.e. committee) of learners trained on the same data set. Seung *et al.* [33] explain that the choice of a query depends on the principle of maximal disagreement. In other words, the information gain of a query can be estimated according to the level of disagreement between learners of a committee. The high level of disagreement points out the query leading to the most informative unlabeled instance. After the selection of an unlabeled instance and acquisition of its output value through an oracle, the labeled instance is added to the training set. Updating training set is followed by the re-training of the committee repeatedly.

In their application, Seung *et al.* [33] screen a set of random inputs until finding a query that conditions the highest disagreement among committee learners instead of constructing such a query. Their algorithm only considers the expected information gain of unlabeled instances where information gain is linked to the uncertainty of the predictions made by committee learners. QBC algorithm does not consider any information which can be extracted from the unlabeled data set such as input density. Indeed, Borisov *et al.* [23] warn that QBC can be obsessed with unlabeled instances from outliers and low-density areas in the unlabeled data set since it is not sensitive to the input distribution and global properties of the inputs. On the other hand,

Freund *et al.* [34] apply a heuristic considering global properties. They identify the most informative queries according to their expected contribution to the reduction of the prediction error. However, they conclude that the heuristic is not efficient. As a result, global properties can be introduced to QBC by compromising the computational efficiency and simplicity of the method.

In consistence with the selection method of informative queries, the computation of the informativeness of an unlabeled instance can vary in QBC applications. For example, Seung *et al.* [33] identify the most informative queries according to the difference between the number of positive and negative classes. The small difference implies informative unlabeled instances. Similarly, Borisov *et al.* [23] mention entropy of the class probabilities predicted by learners of the committee as an example measure for informativeness. In their study, Borisov *et al.* [23] also consider weighted class probabilities to determine the most informative unlabeled instances by using the information about class proportions in the labeled data set. Indeed, the type of predicted output can affect the informativeness measure. For instance, while selection criteria mentioned by Seung *et al.* [33] and Borisov *et al.* [23] can be evaluated in classification settings, the standard deviation, coefficient of variance or range of the predictions of the committee members can be considered in regression settings.

Lastly, even though in their application Seung *et al.* [33] consider querying one instance at a time such that the training set is incremented one by one after each query, batch learning can be another option where more than one instance is queried and added to the training set together. Borisov *et al.* [23] state that batch learning can significantly contribute to efficiency in both labeling efforts and computation time.

3.3. Feature[4] Selection Strategies

Feature selection (i.e. variable selection, feature subset selection) refers to the process of selecting a good subset of features to use in constructing a predictive learning model in supervised learning. Depending on the data set and the goal of applying feature selection, different feature sets can be of interest. Since feature selection serves to reduce the dimensionality of a data set, it is mostly applied when there are many features in a relatively limited data. For this reason, feature selection is a popular application in areas of research dealing with high-dimensional data sets such as bioinformatics [16, 19, 25], neuroimaging [35] and landslide mapping [36].

In the context of metamodeling of a simulation model, feature selection can be applied to find out the significant model parameters with respect to a specific output variable. Therefore, three main objectives of variable selection stated by Guyon and Elisseeff [25] are also valid for the selection of the significant simulation parameters in a metamodeling study. Firstly, variable selection promotes computational efficiency of the metamodel by reducing measurement and storage requirements, and accelerating metamodel training and utilization. In addition, variable selection can provide improvement in prediction performance of the metamodel by discarding insignificant variables (i.e. insignificant simulation model parameters), enabling the metamodel to focus on the significant variables and mitigating curse of dimensionality. Lastly, variable selection may help revealing the embedded relationships in the data through better comprehension. Even data visualization can be applicable with fewer variables left.

In many prominent feature selection studies, based on the discussion about relevance and usefulness of features [37], the objectives of variable selection are generally considered from two perspectives: interpretation and prediction [16, 17, 25, 38]. While finding all relevant features serves for the better interpretation of the output, suffi-

[4]Guyon and Elisseeff [25] differentiate feature from variable ("raw" input variable) when feature is constructed and it cannot be explicitly computed (for example, after kernel transformations). However, in this study both terms are used interchangeably and both correspond to model parameters of a simulation model.

ciently informative features can be used to build a good predictive model [25, 38]. For example, if finding all symptoms of a disease serves for the interpretation objective, identifying important symptoms sufficient to diagnose the disease addresses prediction objective. In this study, feature selection is employed to find the simulation model parameters which condition the model behavior. Identification of such significant parameters contributes to both the interpretation and the successful prediction of the model behavior. Although there could be other parameters slightly affecting the output, these parameters do not play a significant role in determining the output value and their identification does not considerably promote the interpretability of the output.

3.3.1. Variable Importance Measure

Feature selection algorithms can assess the importance of features using variable importance measures (VIM). VIMs are utilized to quantify the relation between variables and the output based on several considerations.

In Wei *et al.* [39], the authors explain seven successful VIMs: the difference-based VIMs, parametric regression and related VIMs, nonparametric regression techniques (like random forest based VIMs), hypothesis test techniques, variance-based VIMs, moment-independent VIMs and graphic VIMs. Each measure can serve different purposes of use. For example, the difference-based VIMs and RF-based VIMs can assess the importance of variables depending on their influence on the output. On the other hand, some VIMs such as variance-based VIMs, moment-independent VIMs and graphic VIMs can evaluate variable importance by extracting the contribution of the uncertainty of variables from the uncertainty of the output. Lastly, the other VIMs can be employed if the dependence between input variables and the output determines the significance of the variables. As a result, the selection of a proper variable importance measure is highly affected by the purpose of quantifying the relation between input variables and the output. (For more detailed explanations, the comprehensive review conducted by Wei *et al.* [39] can be explored. However, a detailed discussion over variable importance measures is beyond the scope of this study.)

Moreover, the choice of variable importance measure is not independent of the data characteristics. In other words, the number of features and variety in their scale of measurement, existence of correlation between features and absence of some part of the data (i.e. missing data) can have impact on determining the proper variable importance measure.

Obviously, RF-based VIMs stand out as powerful candidates in the scope of this study. First of all, instead of using an external VIM which most likely increases both computational and practical burden, a variable importance measure already computed by the metamodel (RF) is more preferable. Secondly, RF-based VIMs can be applied to small data sets with high number of features and they can be calculated for both categorical and continuous features easily. Thirdly, since RF-based VIMs are widely studied in literature, their drawbacks and strengths are mostly revealed. This brings benefits in two aspects. Variable importance scores computed by a well-known application can be examined thoroughly, and the results can be improved by taking required precautions. Also, the popularity of RF may help the analyst to understand the feature importance measurement and encourage to check their findings about feature importance easily when they apply our proposed metamodeling procedure.

3.3.2. Random Forest VIMs

Breiman's RF based on CART trees can compute two variable importance measures, which are Gini index-based (i.e. Mean Decrease in Impurity) and permutation-based VIMs (i.e. Mean Decrease in Accuracy). Gini-based measure considers impurity reduction gained by a variable to determine its importance level. The total decrease in node impurity from splitting on a variable is averaged across all RF trees to calculate the importance score of a variable. Node impurity is measured with the Gini index and Residual Sum of Squares (RSS), respectively for classification and regression. Since splitting variables in RF are selected from a random subset of variables based on their performance in impurity reduction, Gini index-based VIM is also related to the frequency of splits from a variable. On the other hand, permutation-based

measure utilizes OOB data to assess the effect of features on the accuracy of the RF model. The prediction error (error rate for classification, Mean Squared Error (MSE) for regression) on OOB data of each tree is calculated and used as the reference error value. Then, for each feature, the values for the feature on OOB data are shuffled and prediction error on the new data is calculated again. Prediction accuracy is expected to deteriorate after shuffling the values of a significantly important feature. Finally, the average of the difference between the prediction errors on these two data sets over all RF trees is used to estimate the importance score of the corresponding feature.

Strobl *et al.* [40] criticize the reliability of Gini index-based VIM (when RF is built with CART trees). They introduce variable selection bias in the individual classification trees of an RF to describe the bias embedded in the selection of splitting variable. Variable selection bias towards categorical variables, especially with large number of categories, and continuous variables with missing values exists because such variables have higher chance to be a splitting variable in decision trees of an RF [20, 40]. Since Gini index-based VIM tends to favor variables frequently split on, the plausibility of Gini index-based VIM decreases in the presence of missing data and categorical variables. Unless the data is complete and all the features are either continuous or categorical with the same number of categories, Gini index-based VIM is subjected to variable selection bias in classification problems.

On the other hand, permutation-based VIM mitigates variable selection bias in classification problems. Therefore, it can be safely used not only in regression problems but also in classification problems. Consistently, permutation-based importance scores are widely used in prominent feature selection studies using RF such as [16–18, 21]. Nevertheless, permutation-based VIM is affected by the number of randomly selected splitting variable candidates (*mtry* parameter of RF). Since *mtry* can considerably change the tree structure and control the variety among trees, variable importance calculations can be dramatically affected by *mtry* in both regression and classification settings. Furthermore, Strobl *et al.* [20] underline the presence of a bias towards correlated features in RF based VIMs. Permutation-based VIM becomes more sen-

sitive to *mtry* if correlations between variables exist. Permutation-based importance scores of correlated variables are overestimated especially when *mtry* is small [20]. Also, correlated variables might mask each other's importance, since they carry similar information [26].

In order to overcome the bias towards correlated variables and ensure the stability of the variable importance measure, Strobl *et al.* [20] propose a new RF-based VIM called conditional VIM. Unlike Gini index-based VIM and permutation-based VIM, conditional VIM is calculated by another version of RF constructed with conditional inference trees. They show that conditional VIM provides more consistent and bias-free importance scores for both categorical and continuous features even if some features are correlated. However, calculation of conditional VIM requires a tree structure different from Breiman's RF. Moreover, the computation of conditional VIM requires more time than permutation VIM. As the number of features increases, computational cost of conditional variable importance increases as well.

In the context of this study, correlations between variables do not emerge since each variable in the data sets refers to an independent parameter of a simulation model. Hence, the bias towards correlated variables does not appear in calculations of permutation-based VIM and Gini-index based VIM. Moreover, the data sets used in this study do not suffer from missing information as they are generated through simulation runs. Nevertheless, the reliability of Gini-index based VIM is still questionable due to variable selection bias towards categorical features in classification problems. Since permutation-based VIM can overcome the selection bias and produce reliable importance estimates both in classification and regression problems, it stands out as a better measure than Gini-index based VIM. In addition to its reliability, permutation-based VIM is more cost-efficient and more compatible with the proposed metamodeling procedure compared to conditional VIM.

Lastly, in Strobl *et al.* [40], the authors find out that the bootstrap sampling with replacement itself causes bias in variable importance. In their classification ex-

periments, they observe that both Gini index-based and permutation-based VIMs are influenced by the sampling bias, and the effect is more dominant for variables with many categories. Although the sampling bias is present for all of RF-based VIMs unless subsampling is conducted without replacement, their simulation studies show that the effect of sampling with replacement is much weaker than variable selection bias.

3.3.3. Feature Selection Algorithms

Feature selection algorithms are generally discussed in three categories: filter, wrapper, and embedded methods. Guyon and Elisseeff [25] explain that filter methods are applied as preprocessing steps before constructing a predictive model. Basically, they are independently utilized to provide relevant features for a predictive learning algorithm [41]. Statistical measures such as t-test, Anova, and Pearson correlation coefficient can be used to rank and filter features [35]. Unlike filter methods, wrappers utilize the predictive learning algorithm itself to find a good subset of features. Therefore, the learning algorithm is expected to provide better accuracy with the set of features selected by wrapper methods [37]. However, wrapper methods are generally computationally more expensive than filter methods. Since the learning algorithm is used to search over the feature space and re-evaluated for each subset of features, wrapper methods are generally utilized with efficient search strategies such as recursive feature elimination methods. Lastly, in embedded methods, variable selection is applied in the training of the learning model and manipulates the learning model to provide a subset of important variables. Least absolute shrinkage and selection operator (LASSO) can be considered as one of the embedded methods [35]. Embedded methods are generally faster and computationally more efficient than wrapper methods.

Most feature selection algorithms which inspired this study apply wrapper algorithms utilizing RF [16, 17, 19, 21]. They use permutation-based VIM to assess the importance of features and they aim to find the best set of variables by re-training RF with different subsets of features. They either try to find the smallest set of features yielding a competitive prediction performance or try to identify all relevant features to

the outcome of interest for interpretation purposes. While Diaz-Uriarte and Alvarez de Andrés [16] and Jiang *et al.* [19] focus on the former, Genuer *et al.* [17] propose two separate methods to address each objectives. Differently, Hapfelmeier and Ulm [38] aim to develop a method to satisfy both prediction and interpretation objectives.

Feature selection strategies can be grouped into two categories: feature introduction strategies (i.e. ascending selection) and feature elimination strategies (i.e. backward elimination, descending selection). While feature introduction strategies try to find a set of features yielding the best predictive performance by adding new features to an initially very small set of features, feature elimination strategies generally start with the whole set of features and try to reduce the size of the set. Among previously mentioned feature selection studies, Genuer *et al.* [17] perform two separate stepwise variable introduction strategies. They firstly compute the importance scores of the variables. In order to find all the relevant features, the nested subsets of variables are formed by adding variables one by one starting from the most important variables. In their second strategy, to build a successful predictive model, they sequentially introduce each variable and check the performance of the new set of features by monitoring reduction in error. In both strategies, Genuer *et al.* [17] consider OOB error to find the best set set of features. Yet, they suggest utilizing cross validation or a test set to assess the predictive performance of the learning model. Unlike Genuer *et al.* [17], Svetnik *et al.* [21], Jiang *et al.* [19], and Diaz-Uriarte and Alvarez de Andrés [16] apply feature elimination strategies. They calculate permutation-based VIM for all features and iteratively eliminate a specific proportion of the features according to their importance scores. After elimination procedure terminates, they choose the set of features yielding the best learning model performance. While Diaz-Uriarte and Alvarez de Andrés [16] monitors the progress in OOB errors to find a set of features with the minimum error and uses .632+bootstrap method to estimate the prediction error, Svetnik *et al.* [21] perform cross-validation both to estimate test error and to select the best feature set. Since Jiang *et al.* [19] deal with two data sets, they calculate a combined measure incorporating OOB error on the training set and prediction error on the other data set, and select a feature set with the lowest combined value.

Feature elimination strategies also differ in updating the importance ranking of the features after elimination starts. In recursive feature elimination (RFE), importance scores of features are computed at every step and elimination decision is made based on the updated importance ranking. On the other hand, in NRFE, importance scores are calculated in the beginning of the process only once and importance ranking based on the scores is used throughout the elimination process. In other words, the elimination order of features does not change during the process. Since Svetnik *et al.* [21] claim that recalculation of importance scores may lead to overfitting and higher test error, Svetnik *et al.* [21] and Diaz-Uriarte and Alvarez de Andrés [16] apply NRFE. However, Jiang *et al.* [19] recalculates the importance scores after each elimination step so that elimination is guided by the new order of the genes.

3.3.4. RFE vs NRFE

Svetnik *et al.* [21] claim that RFE is more prone to overfitting. According to their experiments, NRFE performs better than RFE. On the other hand, Gregorutti *et al.* [18] argue that Svetnik *et al.* [21] do not consider correlated features and support their claim with simulation studies. In contrast to Svetnik *et al.* [21], Gregorutti *et al.* [18] state that RFE achieves better results than NRFE. Since the correct importance score of a variable is masked due to the presence of its correlated variables, RFE enables variables to update their importance scores as their correlated variables are eliminated. Moreover, Gregorutti *et al.* [18] do not observe overfitting problem with RFE.

In the context of this study, correlations between variables are unlikely to occur unless correlated parameter value combinations of the simulation model are intentionally selected while generating the initial data set. For this reason, both RFE and NRFE can be used. However, the potential of overfitting problem with RFE deserves attention.

Nevertheless, all of the studies mentioned previously perform their strategies on stationary data sets. The size of the data sets used in these studies does not change

during the application of feature elimination. Therefore, all studies can calculate reliable importance estimates in the beginning of their applications. The change in the importance scores results mainly from the updated set of features. Unlike these studies, in our study, we consider a procedure where adaptive sampling and feature selection strategies are applied simultaneously. The procedure is initialized with a relatively small training set and the size of the set is incremented step by step. Depending on the size of the initial training set and the number of instances added to the set at each step, the importance ranking of the feature can dramatically change in the earlier steps of the procedure. Besides, the quality of the training set is expected to increase as the new informative instances are added to the training set. Therefore, to estimate reliable importance scores, it is important to wait until a training set with sufficient quality is obtained before applying a feature elimination algorithm.

In this application frame, NRFE does not benefit from the information carried by new instances selected for the training set at each step of adaptive sampling. However, this may not be an important issue. After the training set is qualified to yield reliable importance estimates, we do not expect drastic changes in importance ranking of features. For this reason, NRFE does not necessarily suffer from ignoring the progress of the training set. On the other hand, RFE uses the most recent information embedded in the training set to estimate variable importance scores. Although Svetnik *et al.* [21] highlight overfitting risk of RFE, Gregorutti *et al.* [18] provide counterarguments. Therefore, RFE may not lead to overfitting in the context of this study. As a result, rather than immediately deciding on which elimination strategy should be used, we will compare their performances by conducting experiments.

4. PROPOSED APPROACH

In light of previous discussions, it is clear that the collaboration and coherence between metamodel, sampling strategy, and feature selection strategy can promote the accuracy and efficiency of a metamodeling approach. While metamodel is used to extract the embedded relationships in a data set, adaptive sampling ensures the quality of the data set. In the simulation context, since the data set is generated via a simulation model, the metamodel, indeed, is used to approximate the model dynamics. Feature selection strategy, on the other hand, can alleviate the complexity in metamodel training and adaptive sampling induced by insignificant model parameters with respect to the monitored output. Hence, we propose an efficient metamodeling approach combining adaptive sampling and feature selection simultaneously to obtain a concise and sufficiently accurate metamodel of a simulation model.

4.1. Selected Methods and Strategies

The approach initializes with the generation of a small training set. Fang *et al.* [32] state that a uniform design is essential to detect various model behaviors without any prior knowledge about the parameters and the output. Therefore, a space-filling sampling method with uniformity property like an optimization-assisted LHS and Sobol sampling should be utilized in obtaining the training set. Since LHS and its variants are widely used in metamodeling studies for simulation models [7, 13, 14], we decided to employ maximin LHS.

The training set obtained from a complex simulation model includes nonlinear relationships between features and the output. Successfully addressing nonlinearity in the training set is vital to achieve high performance and to capture the dynamics of the simulation model. Moreover, an efficient automation of the metamodel, adaptive sampling, and feature selection strategy is important to obtain a fast and self-sufficient metamodeling procedure. In this respect, RF metamodel stands out due to its relatively

fast and straightforward application with QBC-based adaptive sampling and feature elimination strategies. Besides, RF can successfully perform on various data sets with minimal preprocessing and hyperparameter tuning efforts compared to many other metamodel candidates. Hence, an RF-based metamodeling procedure can achieve a user-friendly and compact design while maintaining high predictive performance.

The performance of the metamodel is further improved by adding informative instances to the training set. For this purpose, a pool-based approach is employed to access a wide range of instances without output value (unlabeled instances) and informative instances are chosen from the pool of unlabeled instances according to QBC framework. Now that unlabeled instances correspond to parameter value combinations of the simulation model, obtaining parameter value combinations does not require simulation model evaluations. Hence, the computational cost of generating such a pool is generally quite low. To properly utilize pool-based approach and to ensure QBC is well-supplied, the pool should resemble the actual parameter space. Since the parameter space of the simulation model is uniformly populated, an efficient space-filling method such as maximin LHS can be also used to select unlabeled instances for the pool. The uniform design of the pool enables QBC and uncertainty sampling to identify informative unlabeled instances from different areas of the parameter space. Unlike uncertainty sampling, QBC considers the variety in the predictions of different learners. Therefore, QBC can detect informative instances that a single learner would miss. Besides, RF metamodel has a convenient structure to utilize QBC and a pool-based approach and QBC are compatible with each other. For these reasons, we decided to apply the QBC framework. Accordingly, the most informative unlabeled instances are implied by the highest disagreement between the trees of RF. Disagreement can be computed by various measures such as entropy of the predicted class probabilities or difference in class votes of trees for classification setting. Similarly, standard deviation, coefficient of variance, and range in tree predictions corresponding to each parameter combination can be used to measure disagreement in the regression setting. We will compare the performance of these three measures in Section 5.4.

In order to limit the negative effects of insignificant features on the metamodel performance and adaptive sampling, feature elimination strategies are promising. Besides, they can be adapted to the proposed metamodeling procedure thanks to their iterative nature. Regarding the discussions over RFE and NRFE, we will compare their performances by conducting experiments. In general, feature elimination strategies are guided by VIMs to make elimination decisions. In this study, since the correlation between features (i.e. independent parameters of a simulation model) does not exist, permutation-based VIM appears to be a reliable importance measure. Also, considering that it is easily accessible with RF metamodel, permutation-based VIM is used to estimate the importance scores of features.

4.2. Adaptive Sampling and Feature Elimination Procedures

The proposed metamodeling procedure consists of two main sub-procedures, called adaptive sampling procedure and feature elimination procedure. While the adaptive sampling procedure feeds the metamodel with informative instances and ensures the quality of the training set, the feature elimination procedure enables the metamodel to focus on significant features and allows sampling to perform on a space with lower dimensions. The main flow in the proposed procedure is represented in Figure 4.1.

Initially, a relatively small set of parameter value combinations are generated by maximin LHS. The output values of the corresponding parameter combinations are obtained via simulation runs. The size of a data set required to sufficiently cover the parameter space tends to increase with the number of parameters. Yet, Crombecq *et al.* [6] warn that its size should be small enough to benefit from adaptive sampling in constructing the training set. After the initial training set (T) is constructed, RF metamodel (M) is trained and the initialization prior to the adaptive sampling procedure is completed.

After the initialization phase, the adaptive sampling procedure is applied iteratively throughout the metamodeling procedure until a predefined number of iterations

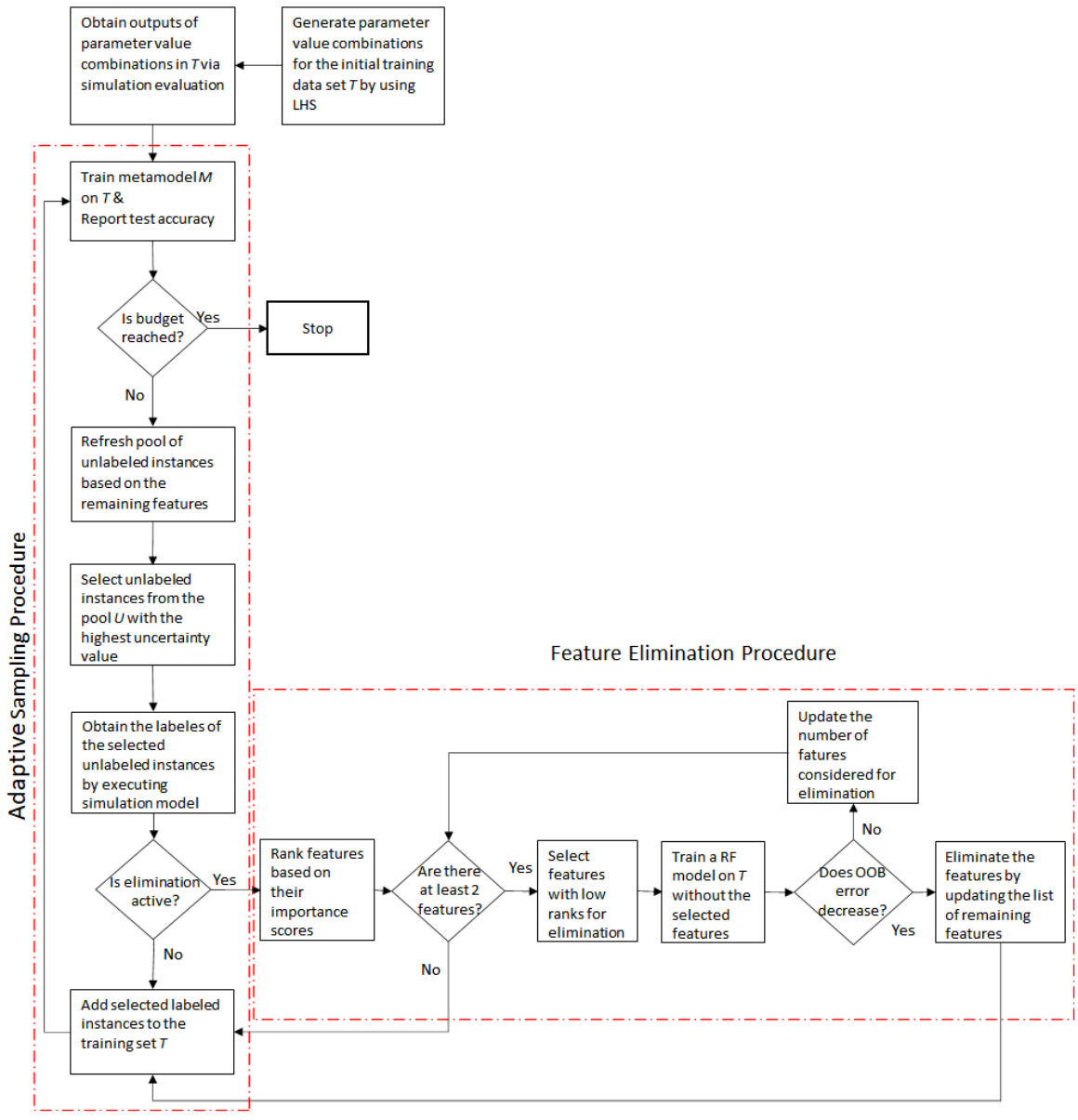

Figure 4.1: Flowchart of the proposed metamodeling procedure (RFE).

(iteration budget, I) is reached. The size and the quality of the training set gradually increase with the addition of new instances to the training set at each iteration. In order to ensure that the quality of the training set is sufficient to estimate reliable importance scores for features, the adaptive sampling procedure is performed alone until a preselected iteration ($iter^*$). When the metamodeling procedure reaches $iter^*$, the feature elimination procedure is activated. Starting from $iter^*$, the feature elimination procedure accompanies the adaptive sampling procedure until the end as long as feature elimination is meaningful. The overall procedure is terminated when the iteration budget is reached.

4.2.1. Adaptive Sampling Procedure

The adaptive sampling procedure is applied throughout the proposed metamodeling procedure. At each iteration, a specific number of the most informative instances (h) is picked from a pool of unlabeled instances (U). The labels of instances in the selected set (c) are queried to the simulation model. Following the assignment of their labels, they join the training set at the end of the iteration. If the feature elimination procedure is active at the current iteration, feature elimination is applied before the new instances are added to the training set. In either case, the effect of the adaptive sampling is seen at the next iteration after the metamodel is trained on the updated training set. The steps of the adaptive sampling procedure are shown in Figure 4.2.

Initially, the dimensionality of the parameter space is potentially large due to the insignificant parameters. When the number of parameters is large and the sample size is relatively small, Kleijnen [13] warns that LHS does not cover the parameter space enough to train the metamodel well. To sufficiently cover the high dimensional parameter space, a much larger pool of unlabeled instances is required. The dimensional complexity and the requirement for a large number of unlabeled instances cause maximin LHS to perform at a high computational cost [29]. In this sense, a traditional pool-based approach [7, 8] where a very large pool is generated at the beginning with both significant and insignificant parameters appears to be potentially inefficient.

Hence, in this procedure, a relatively small pool of unlabeled instances is renewed at every iteration. Since the insignificant features are gradually eliminated by the feature elimination procedure, the complexity in maximin LHS decreases and its efficiency increases over iterations. Moreover, since new pools are sampled based on remaining features without reducing the size of the pools, the informativeness of the pools tends to increase.

Until the iteration budget is reached, the selection of new instances continues, and the training set is enlarged. Rather than defining a dynamic stopping criterion, we let the user free to decide on the length of the proposed procedure depending on the available computational power.

Initialization: Train RF metamodel (M) on the initial training set (T).
Record the RMSE value of M.

1: Generate a pool of unlabeled instances (U) with maximin LHS.

2: Obtain the predictions of individual trees in M for each instance in U. Assuming M consists of L decision trees, L many predictions for each instance are obtained.

3: For each instance in U, compute the disagreement measure, i.e. uncertainty value, based on a selected metric such as coefficients of variation.

4: Select h instances subjected to the highest disagreement between the trees of M to obtain unlabeled training candidates (c) by using Eq. 4.1;

$$c = \underset{u}{\operatorname{argmax}}\, v_u,\ u \in U \tag{4.1}$$

5: Obtain labels of the training candidates through simulation runs.

6: Update T by adding labeled training candidates (C) to T, after feature elimination procedure.

Figure 4.2: Adaptive Sampling Procedure.

4.2.2. Feature Elimination Procedure

Feature elimination procedure can be applied earliest when the training data becomes large enough to correctly order the variables according to their importance scores. For convenience, the first iteration at which feature elimination procedure can be utilized in the proposed procedure ($iter*$) is predetermined.[5] Once the proposed metamodeling procedure arrives at $iter*$, the feature elimination procedure is activated. Afterward, as long as the number of remaining features (k) is enough, the feature elimination procedure is implemented at the next iterations. When only significant features are left, the feature elimination procedure does not make an elimination decision.

The feature elimination procedure is guided by the permutation-based importance scores of features. If NRFE is utilized, importance scores are calculated only when the feature elimination procedure is activated. These scores are permanently used for the rest of the procedure. In case RFE strategy is employed, the importance scores of the features are updated at each iteration ($iter$).

The feature elimination procedure follows an iterative strategy to point out the insignificant features. In the elimination procedure, initially a specific fraction (p) of features is considered for elimination. These features are selected based on their importance scores. However, if the elimination of initially selected features does not lead to a satisfactory expected change in OOB error of the metamodel, a smaller set of the features is reviewed for elimination at the next iteration of the elimination procedure ($elim\_iter$). The number of features considered for elimination exponentially decreases at each $elim\_iter$ until Cond. 4.2 holds;

$$OOB\,error_{elim\_iter} < OOB\,error_{elim\_iter-1} * (1 + o) \tag{4.2}$$

where $OOB\,error_{elim\_iter-1}$ is OOB error calculated at the previous $elim\_iter$ and o is the allowed increase rate in OOB error defined by the user. Exponential reduction

[5]Detailed discussions on the selection of a proper $iter*$ are made based on several experiments in Subsection 5.5.2.

allows for decreasing the dimensionality faster. At the same time, the attention to the significance of each remaining feature increases step by step. When a subset of insignificant features is found, their elimination is realized by updating the feature list so that the metamodeling procedure continues with the remaining features. Otherwise, the elimination procedure is terminated and none of the features is eliminated in this iteration of the metamodeling procedure (*iter*). The steps of the feature elimination procedure are summarized in Figure 4.3.

Table 4.1: Number of features considered for elimination at each *elim_iter* based on p, when the number of features is 50 in the beginning.

		elim_iter				
		1	**2**	**3**	**4**	**5**
p	**0.2**	10	2	-	-	-
	0.5	25	12	6	3	1
	0.8	40	32	25	20	16

The speed of the elimination procedure is controlled by the predetermined parameter p, which determines the number of features considered for elimination. Table 4.1 illustrates the number of features considered for elimination at each *elim_iter* for different p values in a single application of feature elimination procedure. As p increases, larger sets of insignificant features are evaluated for elimination. Therefore, for a large p value, the elimination procedure becomes more aggressive. Yet, if the number of insignificant features is only a few, it requires more *elim_iter* to find the small set of insignificant features. Considering that an RF learner is trained at each *elim_iter*, the frequent training of RF learner causes inefficiency in the elimination procedure. On the other hand, the procedure terminates faster with a small p. Nevertheless, a small p can result in partial elimination of insignificant features especially when the number of insignificant features is large. The feature elimination procedure is applied again at the next iterations of the metamodeling procedure to eliminate all insignificant features. In the worst case, the proposed metamodeling procedure may not achieve to eliminate all of the insignificant features due to the limited iteration budget. As a result, de-

Given that $iter^* \leq iter$ and $2 \leq k$;

Calculate permutation-based importance scores if RFE is used. Otherwise, calculate only when $iter = iter^*$;

Set $elim\_iter$ to 1;

$check\_elim \leftarrow TRUE$;

while $check\_elim$ **do**

  Calculate the elimination fraction by using Eq. 4.3, and the number of features considered for elimination by using Eq. 4.4;

$$p^* = p^{elim\_iter} \tag{4.3}$$

$$k' = \lfloor k_{iter} * p^* \rfloor \tag{4.4}$$

  where p is a predefined fraction determining the maximum number of features that can be eliminated at once and k_{iter} is the number of features at $iter$.

  Train M on T without k' features having the lowest importance scores;

  if Cond. 4.2 holds **then**

    Update k_{iter} by eliminating the selected k' features from the feature set;

    $check\_elim \leftarrow FALSE$;

  else

    if all features are significant **then**

      $check\_elim \leftarrow FALSE$;

    else

      $elim\_iter = elim\_iter + 1$;

    end if

  end if

end while

Continue with the adaptive sampling procedure.

Figure 4.3: Feature Elimination Procedure.

pending on the number of insignificant features, different p values can be preferred. Unless a prior information about the number of insignificant features is available, it is safer that p should be neither very large nor very small to ensure the efficiency and the performance of the metamodeling procedure.

The feature elimination procedure makes elimination decisions before the new informative instances selected by the adaptive sampling procedure are added to the training set. For this reason, the feature elimination procedure is not influenced by the adaptive sampling procedure performed at the same iteration of the proposed procedure. The combined effect of feature elimination and adaptive sampling can be seen at the next iteration of the metamodeling procedure at the earliest.

5. EXPERIMENTAL APPROACH

Two main experimental designs are considered to find out the sampling strategy yielding the best metamodel performance, and to evaluate the performance of the proposed metamodeling procedure with feature elimination. The performances of adaptive sampling methods are compared with the performances of one-shot sampling and random sampling in the first design. Then, the best sampling strategy is utilized in the proposed procedure in the second design, and the overall performance of the proposed procedure with feature elimination is evaluated. The second design also enables to separate the contributions of adaptive sampling and feature elimination to the metamodel performance.

The experiments are performed with a well-known agent-based simulation model, which is segregation model (SM) [42]. The simulation runs are executed on Netlogo [43] version 6.0.4. The metamodeling procedure is performed in R.

5.1. Simulation Model

Segregation model (SM) depicts a very common social phenomenon with only two parameters. Despite its simplicity, the model is capable to show complex system behaviors. Hence, the metamodeling experiments with SM provide valuable insights about the potential performance of metamodeling procedures applied on a higher-dimensional simulation models with similar complex behaviors [7]. Moreover, since SM is widely studied in literature, all possible dynamics of the model are already revealed. With prior knowledge of the system behavior, the results of experiments can be analyzed better.

Despite its convenient use in the experiments, SM is a concise model only with two parameters and both of them are significant. Therefore, the proposed metamodeling procedure with feature elimination is applied to a variant of SM (DSM). DSM is built

by introducing four insignificant parameters (i.e. dummy parameters) to SM. The influence of insignificant parameters of a simulation model on sampling and metamodel performance, and the contribution of their elimination to the overall metamodeling procedure are analyzed through SM and DSM.

5.1.1. Segregation Model

The basic segregation model [42] involves two groups of people in a specified area. For example, these groups can be formed based on ethnicity, supported football team or religion while the specified are can be a city, a neighborhood, or a class. According to the model scenario, if people can live beside a certain number of people from their own group, they feel happy. To seek happiness, people keep moving to unoccupied places in the specified area. The simulation continues until everyone is happy. The model reveals the effect of individuals' choices on society based on their preferences for similarity and spatial restrictions [44].

As mentioned previously, SM has two parameters called *density* and *%-similar-wanted*. While *density* is related to the physical conditions such as population and the size of the area, *%-similar-wanted* resembles the intolerance level of people to people from the other group. Both parameters are represented with percentages. When *density* is set to a high value, the neighborhood is represented as a crowded place and there are only a few empty spaces in the neighborhood. In order to describe a place with full of unoccupied places where people can move easily, a small density value is chosen. On the other hand, *%-similar-wanted* expresses people's desire to live close to neighbors from their own group. It is represented as the minimum fraction of neighbors from their own group in order for people to be happy. Setting *%-similar-wanted* high implies the high intolerance between groups in the society while low *%-similar-wanted* resembles a more peaceful society. The output variable monitored to measure the level of segregation is *%-similar*. *%-similar* is computed as dividing the sum of the number of same-group neighbors of each person by the sum of the number of all neighbors of each person.

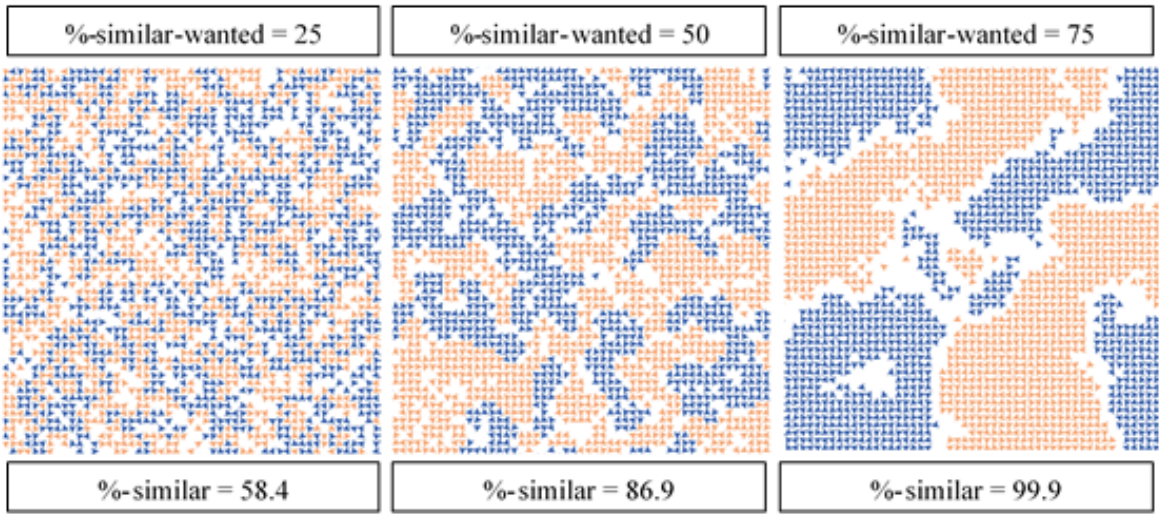

Figure 5.1: Segregation in the society based on different *%-similar-wanted* values.

In general, an increase in *%-similar-wanted* is expected to result in a rise in *%-similar* since people would be more eager to live near the people from the same group. However, the interaction of model variables gives birth to various system behaviors for different levels of *density* and *%-similar-wanted*. Firstly, as long as *density* is low enough (roughly under 15), independent of *%-similar-wanted* value, *%-similar* generally reaches values higher than 80. Secondly, if *density* is higher than 15, the value of *%-similar-wanted* starts to play an active role. Unless *%-similar-wanted* is above 75, *%-similar* tends to increase as *%-similar-wanted* keeps rising, in general. Thirdly, *density*, on the other hand, has a little effect on holding back *%-similar* from its increasing trend with *%-similar-wanted*. Finally, only when *%-similar-wanted* is very small (less than 30), high *density* has an observable negative effect on *%-similar*. To sum up, specific thresholds for the parameters determine the effect of parameters on the output and result in sharp changes in the model behavior.

Figure 5.1 shows the segregation in society visually for different *%-similar* values. Each point represents a person from one of two groups depending on its color (blue vs orange). *density* is set to 80 and by changing *%-similar-wanted* values, the levels of segregation corresponding to achieved *%-similar* are represented. On the figure, it can

be seen that high *%-similar* generally addresses highly segregated society while lower *%-similar* points out the existence of small-sized local groups.

5.1.2. Segregation Model with Dummy Parameters

With the introduction of four dummy parameters, the new segregation model (DSM) includes six parameters in total. While *%-similar-wanted* and *density* still condition the model behavior, the other four parameters have little or no effect on the output. Among dummy parameters, *budget-multiplier-dummy* and *density-multiplier-dummy* are completely redundant parameters which do not influence any variables in the model. On the other hand, the other two parameters, i.e. *noise-dummy* and *tick-limit*, have a very small impact on the output. While the value of *noise-dummy* is directly added to the output, *tick-limit* indirectly affects the output by constraining the simulation run time. In order to make sure these dummy parameters remain hardly relevant, *noise-dummy* is restricted between 0.00001 and 0.0001, and *tick-limit* is forced to be between 90 and 110. The upper limit for *tick-limit* is higher than the simulation run length limit (100 time steps) so that it becomes completely irrelevant in some of the simulations.

Since the four dummy parameters barely affect the output, SM and DSM are expected to show very similar model behaviors when the same pair of *%-similar-wanted* and *density* is selected. In other words, no matter what the values of dummy parameters are, as long as the same value combination of *%-similar-wanted* and *density* is selected, the outputs of SM and DSM are expected to be almost the same. Hence, DSM enables us to monitor the effect of the presence of insignificant parameters with respect to a selected output on the metamodel performance and adaptive sampling. Also, the influence of the feature elimination on the metamodel performance can be analyzed by experimenting with DSM.

Overall, the following points can be checked in the evaluations regarding adaptive sampling and feature elimination:

(i) If the importance scores of the parameters are assessed correctly, then four dummy parameters should be found out insignificant.

(ii) If the feature elimination procedure suggested in this study works well, four insignificant features (corresponding to the four dummy parameters of the simulation model) must be eliminated in the proposed metamodeling procedure.

(iii) Adaptive sampling must focus on *%-similar-wanted* and *density* in sample selection rather than the four insignificant features. In other words, despite the presence of the insignificant features, the information carried by a sample should be measured according to the values of significant features, that are *%-similar-wanted* and *density*.

5.2. Data Sets

In this study, since the proposed metamodeling procedure has experimented on a simulation model, data sets to train and test the metamodel are obtained through simulation runs. The parameter value combinations evaluated in the simulation model are selected by using maximin LHS. For each combination, simulation models are executed for 10 times and the average result is assigned as the output value of the corresponding parameter value combination. Also, it is observed that the output value of the model stabilizes before the 100th time step in simulation runs. Therefore, the length of the simulation runs is restricted with 100 time steps to prevent the run from lingering.

Training sets in different sizes are generated to evaluate the performance of adaptive sampling and one-shot sampling, separately. One-shot sampling is performed with a training set including 100 instances for each, whereas initial training sets for adaptive sampling are built with 50 instances. The sizes of training sets does not change for SM and DSM. Although it is better to increase the size of a data set with the number of features (i.e. parameters in the simulation model), obtaining a large data set may not be feasible due to the high computational cost of simulation evaluations. Therefore, the size of the training data sets for DSM is not increased proportionally to the number

of its parameter. Besides, in the experiments, it is observed that the current sizes of the training sets are sufficient to train successful metamodels.

Similar to the training sets, the pools of unlabeled instances include parameter value combinations obtained with maximin LHS. Pools with 100 unlabeled instances are generated in the experiments and unlabeled instances are re-selected at every iteration of the proposed metamodeling procedure.

Lastly, the performances of the metamodels are evaluated based on a test set. For SM and DSM, two separate test sets are obtained by randomly selecting parameter value combinations within the allowed ranges of each parameter. The output values for the selected parameter value combinations are evaluated in the simulation model. For both SM and DSM, the test sets include 100 instances.

Overall, data sets generated with simulation runs do not require data cleaning and corrections due to missing or wrong information. Since the features in the data sets correspond to uncorrelated and independent parameters in the simulation model, potential problems due to correlated features are not observed. Hence, data sets generated with simulation runs can be easily utilized in the proposed metamodeling procedure.

5.3. Performance Assessment

In the experiments, the performances of both adaptive sampling methods and the proposed procedure are measured on a test set. Considering that SM and DSM provide continuous output (i.e. *%-similar*), RMSE (Root Mean Squared Error) is used as the main performance measure for metamodel evaluation in the experiments. It is calculated for a data set with N instances by Eq 5.1;

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(\hat{y}_i - y_i)^2}{N}} \tag{5.1}$$

where y_i and $\hat{y}_i$ are real and predicted output values of instance i, respectively. In this study, while y_i is obtained through evaluation of instance i in the simulation model, $\hat{y}_i$ implies the estimated output of instance i by the metamodel.

Although RF provides OOB error as a prediction error estimate, OOB error does not stand out as an unbiased error estimate due to the sequential approach [16]. Moreover, a test set can be easily manipulated by the user for different motivations. Therefore, in spite of its computational cost, using a test set provides flexibility for the user and reliable error estimations to evaluate the performance better.

5.4. Comparative Study of the Sampling Strategies

In this section, a comparative analysis of adaptive sampling (AdS), one-shot sampling (OsS), and iterative random sampling (RdS) is conducted. Their influences on the metamodel performance are discussed. OsS is based on the selection of training set instances at once before metamodel training. In other words, it does not follow an iterative procedure and does not benefit from historical metamodeling and sampling information. On the other hand, RdS and AdS build the training set through iterative instance selection. Both are initialized with a small training set and at each iteration, new instances are added to the training set. Unlike OsS, RdS does not maintain a space-filling property. Although RdS initializes with a relatively small data set obtained with maximin LHS, adding random instances to the training sets violates uniform coverage of the feature space.

While RdS blindly selects random new instances without using any information throughout the iterative process, AdS compiles the information of previously selected instances and the metamodels trained to choose new instances. Therefore, AdS stands

out as a smart selection method at which instances are evaluated based on their informativeness. In AdS, highly informative instances are picked for the training set. The informativeness measure can vary in different adaptive sampling applications. Since the QBC method is utilized as the sample selection framework, the informativeness of an instance is measured based on the disagreement among the decision trees of an RF ensemble. In this study, three disagreement measures applicable to the continuous output variable (*%-similar*) are separately evaluated. These measures can be named as range-based (AdS_range), standard deviation-based (AdS_sd) and coefficients of variation-based (AdS_cvar) informativeness measures. They can be computed by Eq 5.2, Eq 5.3, or Eq 5.4;

$$range - based: \qquad v_u = \max_L \hat{y}_{ul} - \min_L \hat{y}_{ul} \tag{5.2}$$

$$standard\ deviation - based: \qquad v_u = \sqrt{\frac{\sum_{l \in L} (\hat{y}_{ul} - \hat{y}_u)^2}{L-1}} \tag{5.3}$$

$$coefficients\ of\ variation - based: \qquad v_u = \frac{\sqrt{\frac{\sum_{l \in L} (\hat{y}_{ul} - \hat{y}_u)^2}{L-1}}}{\hat{y}_u} \tag{5.4}$$

where $\hat{y}_{ul}$ is the predicted output value of instance $u \in U$ obtained by decision tree $l \in L$, and $\hat{y}_u = \frac{\sum_{l \in L} \hat{y}_{ul}}{L}$.

As a result, in this section, we focus on comparing the performance of the metamodels trained on data sets sampled with OsS, RdS, AdS_range, AdS_sd, AdS_cvar without taking feature elimination into consideration. Our analyses also aim to address the effect of the *mtry* parameter of RF and the existence of insignificant features.

5.4.1. Experimental Design

To monitor the effect of OsS on metamodel performance, 11 large training sets containing 100 instances are used while 11 small initial training sets each having 50 instances are considered for iterative sampling methods (i.e. RdS, AdS_range, AdS_sd and AdS_cvar). All sampling strategies are replicated 10 times in each of their cor-

responding training sets. Due to the randomness embedded in the metamodel itself, distinct metamodels are obtained in each replication. In other words, in the end, 110 final metamodels are obtained for each sampling strategy.

The application of OsS follows a very straightforward procedure (Figure 5.2). An RF metamodel (M) is trained on a large training set (T). After training, the accuracy of M on the test set is recorded. This process is repeated 10 times (R) in each large training set.

Inputs : R , T

1: **for** $r = 1$ to R **do**
2: Train RF metamodel (M) on T;
3: Report RMSE of M on the test set;
4: **end for**

Figure 5.2: Application of One-shot Sampling.

On the other hand, AdS and RdS introduce new instances to the training set throughout a predefined number of iterations (i.e. iteration budget, I). At each iteration, a pool of unlabeled instances (U) is generated and unlabeled training candidates (c) is composed by selecting h unlabeled instances from the pool based on the disagreement about their output predictions. Through simulation runs, the labels of the training candidates are obtained, and these labeled instances (C) are added to the training set. Then, the metamodel (M) is retrained on the training set and the RMSE value of the metamodel on the test set is recorded. Until the iteration budget is reached, this process is repeated. At the end of the final iteration, the final training set and the final metamodel utilized to mimic the simulation model are obtained.

In the experiments regarding AdS and RdS, U consists of 100 unlabeled instances. h is set to five and iteration budget is selected as 11. New instances are selected and added to the training set throughout the first 10 iterations. At the final iteration, sampling is not performed, and M is trained on the final training set with 100 instances. Therefore, the final iteration only serves to obtain the final metamodel and record its

Table 5.1: Parameter values used in the sampling strategy experiments.

Parameters in the Sampling Strategy Experiments		
I	Iteration budget	11
h	Number of instances selected in each iteration	5
u	The number of instances in U	100
mtry	Hyperparameter of RF	Tuned/Default
ntree	Hyperparameter of RF	300

performance. The parameters used in the experiments can be seen on Table 5.1. Except for *mtry*, the values of all the parameters are kept the same for SM and DSM. Since we aim to evaluate the influence of *mtry* on the metamodel performance in a generalized framework, default and tuned *mtry* values are considered in the analyses for SM and DSM. The default *mtry* value of RF is calculated by Eq. 5.5;

$$mtry_{default} = \begin{cases} \max(\lfloor \frac{k}{3} \rfloor, 1) & for\ continuous\ output \\ \lfloor \sqrt{k} \rfloor & for\ categorical\ output \end{cases} \tag{5.5}$$

where k is the number of features. Default *mtry* values equal to 1 and 2 for SM and DSM, respectively. When *mtry* values providing the best RF performance are searched, the best *mtry* value is found to be 2 for SM while it is 6 for DSM.

In SM and DSM, the dynamic behavior is conditioned mostly by *%-similar-wanted*. Although *density* is still an important model parameter, *%-similar-wanted* appears to be more dominant. Therefore, RF metamodels having more splits on *%-similar-wanted* generally provide better predictions of the model output. In SM, when *mtry* is set to 1, splits in RF are made randomly and the metamodel does not benefit from *%-similar-wanted* to the fullest. However, *%-similar-wanted* has much more chance to be a splitting node when *mtry* is selected as 2. Similarly, in DSM, when *mtry* is set to its default value, insignificant features (i.e. dummy parameters of the simulation model) have more chance to be considered in splits compared to significant

Inputs : R, T, h

$T' \leftarrow T$;
for $r = 1$ to R **do**
  for $iter = 1$ to I **do**
    Train RF metamodel (M) on T';
    Report RMSE of M on the test set;
    if $iter \neq I$ **then**
      Generate a pool of unlabeled instances (U);
      Calculate uncertainty value of each instance in U by using Eq 5.2, Eq 5.3, or Eq 5.4 based on sampling strategy in use;
      Select unlabeled training candidates (c) by choosing h instances with the highest uncertainty from U by using Eq. 4.1;
      Obtain labeled training candidates (C) by evaluating c through simulation;
      Update T' by adding C to T'; ;
    end if
  end for
end for

Figure 5.3: Application of Iterative Sampling.

features. Since only two out of six features are the significant features (*%-similar-wanted* and *density*), the RF metamodel mostly suffers from splits on insignificant features. On the other hand, when *mtry* equals to 6, the metamodel can better extract information embedded in the data set by focusing on *%-similar-wanted*. In the experiments, we confirmed that RF metamodels use *%-similar-wanted* more frequently in decision tree splits when *mtry* equals to 6. As a result, the importance level of *%-similar-wanted* considerably jumps to a higher level (Figure 5.4).

As a result, since *%-similar-wanted* is a very dominant parameter on model behavior, the best *mtry* value equals to the number of parameters in both SM and DSM.

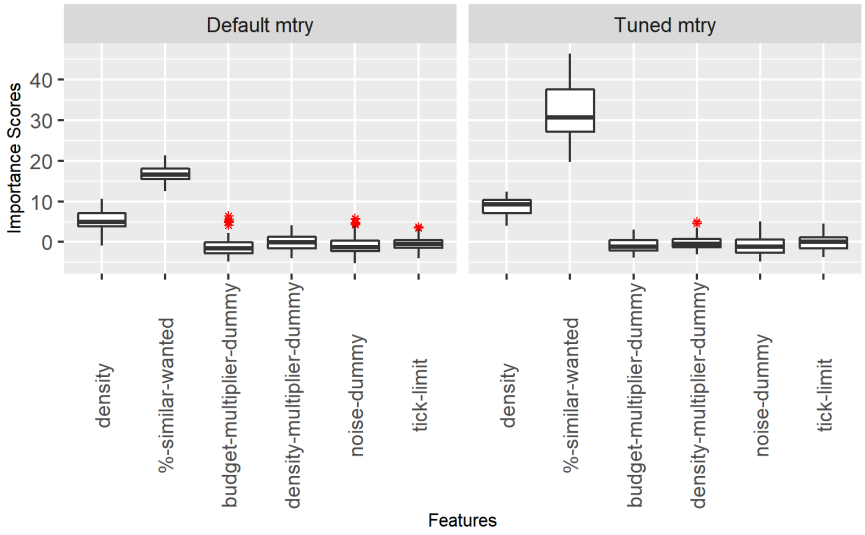

Figure 5.4: Boxplot of feature importance scores based on OsS.

For another simulation model, the best *mtry* value can be less than the number of features depending on the relative importance of the significant features. Hence, we prefer focusing on default and tuned *mtry* values to obtain more generalized insights, rather than comparing specific *mtry* values.

5.4.2. Results

To provide an overview of the performances of sampling strategies, the average RMSE values over 110 metamodels trained on 11 data sets with 10 replications are considered for each sampling strategy. In Figure 5.5 and Figure 5.6, the performances of sampling strategies over iterations are represented in terms of mean RMSE for SM and DSM, separately. Each color represents a sampling strategy. Since RdS, AdS_range, AdS_sd, and AdS_cvar follow an iterative process, their mean RMSE values change over iterations. Dots on the lines show the average of 110 RMSE values obtained at the corresponding iteration for the corresponding sampling strategy. On the other hand, OsS is represented as a horizontal line over iterations to easily compare with the other sampling strategies. The figures are displayed in two parts based on the *mtry* value used in metamodel training. While the performances obtained by RF using

default *mtry* value (1 for SM, 2 for DSM) are shown on the left-hand side, improved performances with the best *mtry* value (2 for SM, 6 for DSM) are presented on the right-hand side. Therefore, Figure 5.5 and Figure 5.6 give valuable insights into the best performing strategy and the effect of *mtry* on the performances.

In general, tuning *mtry* provides better metamodeling performance for both SM and DSM. The mean RMSE values of RF decrease with all sampling strategies. However, the positive effect of using tuned *mtry* is more visible for DSM. Due to the presence of insignificant features, default *mtry* value prevent RF from effectively using significant features in the construction of decision trees, and cause RF to miss valuable information that could be obtained from significant features. In Figure 5.6, the large gap between the performances obtained with the default and tuned *mtry* values points out the magnitude of such information loss due to the default *mtry* value used in RF for DSM.

In addition to the effect of *mtry*, sampling strategies also influence the metamodel performance. Figure 5.5 shows that all AdS strategies perform better than RdS and OsS regardless of selected *mtry*. Since they lead to sharp decreases in mean RMSEs in the early iterations, they generally obtain better data sets including fewer instances compared to OsS. For instance, AdS_sd and AdS_cvar can surpass the performance of OsS only with 75 instances and 65 instances, on average, depending on *mtry*. At the final iteration, while the mean RMSE with both AdS_sd and AdS_cvar is 4, it is 5.32 with OsS for default *mtry*. If *mtry* is tuned, the mean RMSE with AdS_sd and AdS_cvar reduces to 3.09 (by 23%) and 3.22 (by 19%). The mean RMSE with OsS decreases by 13% and becomes 4.61. On the other hand, AdS_range is affected by *mtry* the most among AdS strategies. The mean RMSE with AdS_range decreases from 4.96 to 3.32 (by 33%). Also, the pace of decrease in mean RMSE is slower with AdS_range. Therefore, although AdS_range provides better performance than OsS and RdS, on average, its performance is still dominated by AdS_sd and AdS_cvar.

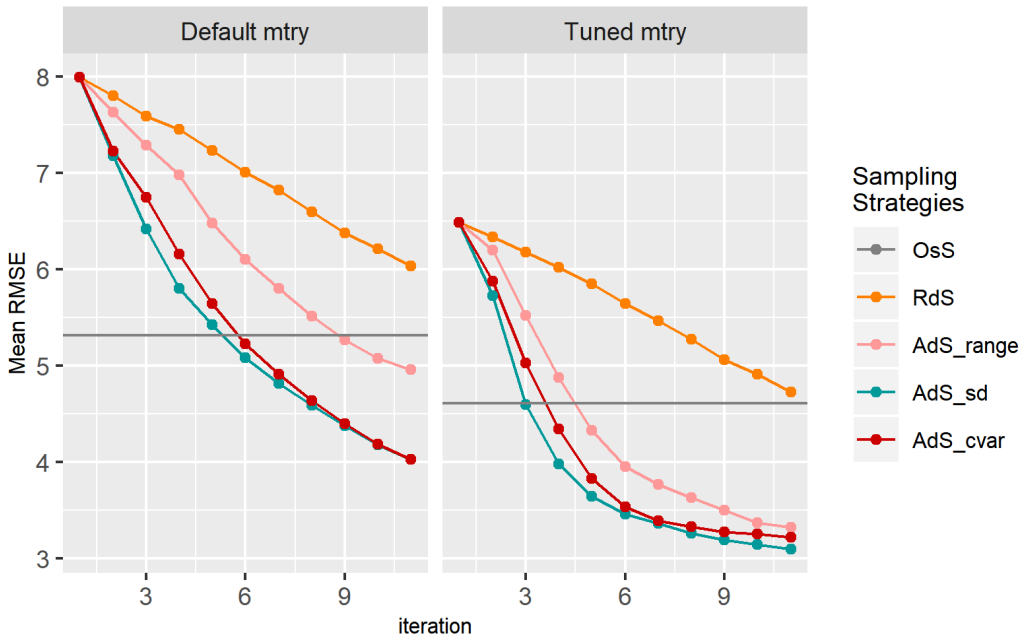

Figure 5.5: Average errors with different sampling strategies in Segregation Model.

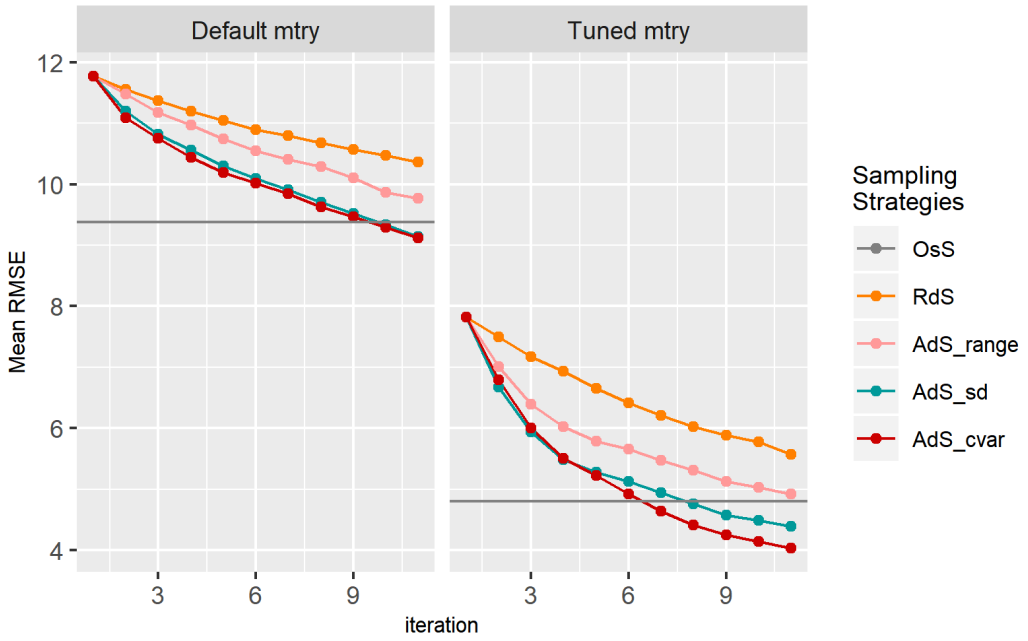

Figure 5.6: Average errors with different sampling strategies in Segregation Model with Dummy Parameters.

Moreover, a comparison between OsS and RdS provides insights into the effect of space-filling methods in metamodel performance. While OsS consists of uniformly distributed instances all over the feature space, RdS does not pay attention to maintain the space-filling property of its initial training set. Since OsS performs better than RdS on average, we can confirm that a space-filling sampling strategy such as maximin LHS is a good method to provide initial training sets for AdS applications rather than random selection of the training instances.

On the other hand, when dummy parameters exist in the simulation model, Figure 5.6 shows that the performance of all sampling strategies falls. Especially when default *mtry* value is used in the metamodels, the sampling strategies suffer from the presence of the insignificant features the most. The difference between overall performances obtained with different *mtry* values becomes deeper. However, even when the situation is at worst due to insignificant features and default *mtry*, AdS_sd and AdS_cvar perform better than OsS, on average. Although they do not improve the metamodel as fast as they do in SM, they still provide training sets with slightly better quality than OsS provides with 100 instances. While the mean RMSE with both AdS_sd and AdS_cvar is about 9.12, OsS achieves 9.38 as the average RMSE. Consistently, the utilization of the tuned *mtry* value improves the performances of AdS_sd and AdS_cvar by roughly 50%. Indeed, AdS_cvar (4.04) appears to be slightly better than AdS_sd (4.39) with tuned parameters, on average. Nevertheless, tuning *mtry* does not help AdS_range to surpass the performance of OsS in DSM. In the end, OsS outperforms AdS_range and RdS. This means that adaptive sampling (i.e. output-oriented sampling) does not necessarily perform better than an input-oriented sampling method such as maximin LHS. The identification method of informative unlabeled instances is important to obtain a competitive performance compared to OsS using a space-filling method.

Figure 5.7 and Figure 5.8 provide further information about the reliability of sampling strategies. Both figures illustrate the change in RMSE of metamodels throughout the sampling procedure while the best *mtry* value is used in RF metamodel. Each

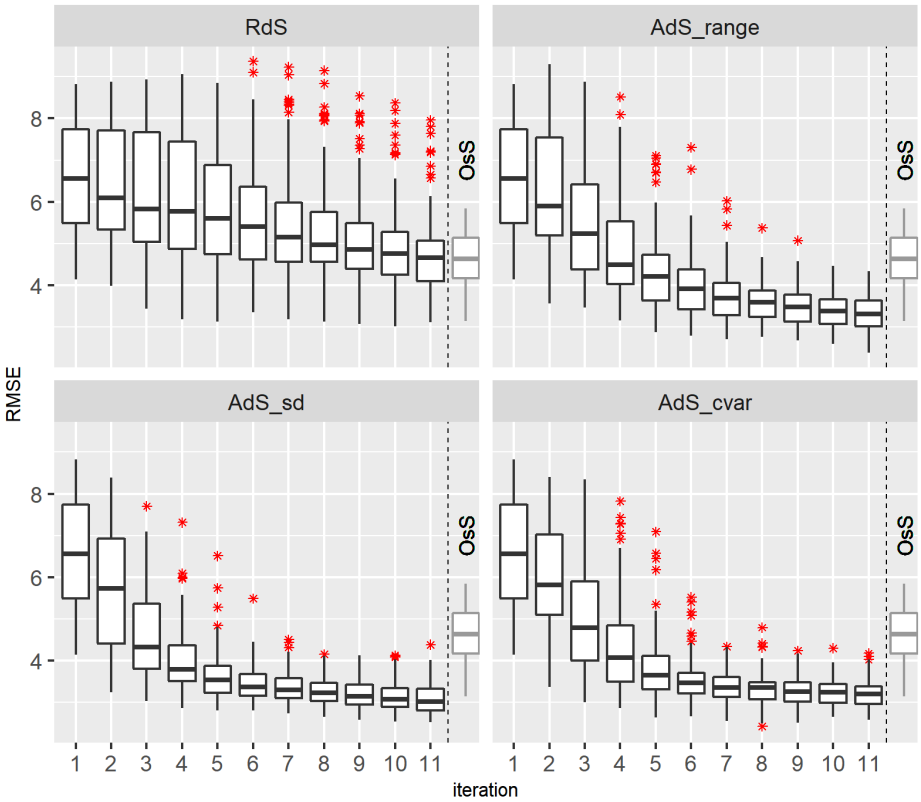

Figure 5.7: Boxplots of test errors with different sampling strategies in Segregation Model.

boxplot refers to the error distribution of 110 metamodels fed by a specific sampling strategy. To easily compare the performances of iterative sampling strategies with OsS, the boxplots of RMSE values obtained with OsS are represented next to the final boxplots of the corresponding sampling strategies. Therefore, performance differences due to the selected data set and randomness in replications can be monitored over iterations. Initially, all iterative sampling strategies have tall boxplots. This means that the performance of the metamodel mostly depends on the initial data set and randomness in the replications. However, AdS_sd and AdS_cvar achieve to shorten the length of the boxplots dramatically throughout iterations. In other words, as new instances are included in the training set, the effect of the initial data sets on the performance decreases and the performance of AdS_sd and AdS_cvar become more robust. To put it another way, regardless of the initial training set, both AdS_sd and AdS_cvar converge

to a specific RMSE level. AdS_sd and AdS_cvar generally decrease RMSE faster when the initial RMSE is high. As a result, the short boxplots of AdS_sd and AdS_cvar show their robustness. Since their boxplots are shorter and lower than the boxplot of OsS, AdS_sd and AdS_cvar are superior to OsS in terms of robustness and performance.

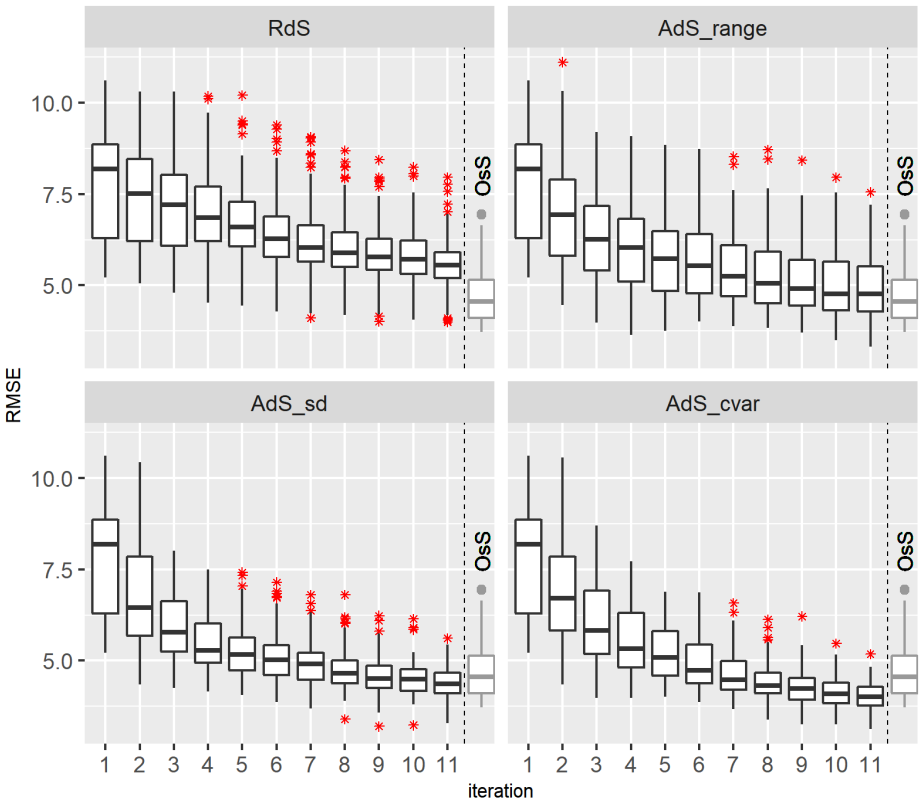

Figure 5.8: Boxplots of test errors with different sampling strategies in Segregation Model with Dummy Parameters.

Nevertheless, the presence of insignificant features negatively affects the performance of sampling strategies even when *mtry* paremeter is tuned. Although tuning *mtry* helps to compensate for the drawbacks of insignificant features, increases in the final RMSE of the metamodels trained with AdS_sd or AdS_cvar is visible (Figure 5.9).

In conclusion, AdS_cvar and AdS_sd appear to be better than OsS and other iterative sampling strategies in terms of robustness and performance. The metamodels trained with AdS_sd and AdS_cvar provide better performances with fewer training

instances compared to OsS. However, the selection of *mtry* value and the presence of the insignificant features play an important role in the performance of AdS_sd and AdS_cvar. Especially when there are insignificant features, the selection of *mtry* value becomes much more important. However, when the number of features and the size of the data set are large, a comprehensive tuning can be computationally costly. A good rule of thumb is to select an *mtry* value larger than the default value and less than the number of features. Yet, the presence of insignificant features affects the adaptive sampling process from the beginning and causes performance loss. Moreover, considering that we seek to propose a time-efficient and user-friendly metamodeling procedure, the strong dependency of the performance of the adaptive sampling procedure on the *mtry* value and the extensive need for its tuning is not desirable. As a result, we aim to eliminate the insignificant features to recover the performance loss due to insignificant features and weaken the dependency on the tuning of the *mtry* parameter of RF.

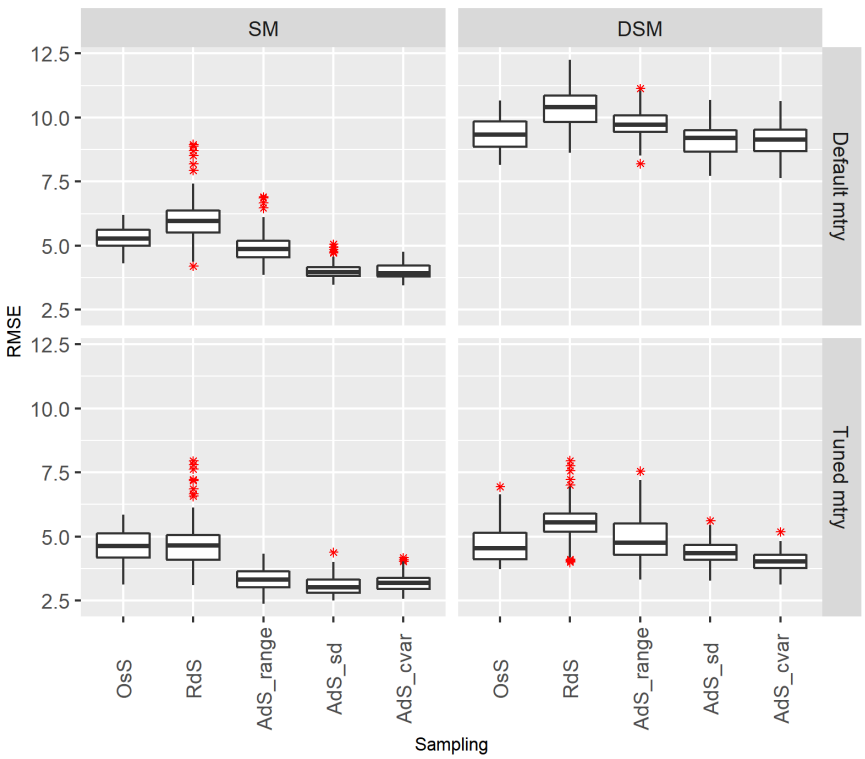

Figure 5.9: Boxplots of test errors of final metamodels obtained with different sampling strategies.

5.5. Performance of the Proposed Metamodeling Procedure

The proposed metamodeling procedure consists of two sub-procedures. After the initialization of the procedure, adaptive sampling procedure and feature elimination procedures are applied iteratively. Although the adaptive sampling procedure is employed at every iteration, the feature elimination procedure becomes active as soon as a predefined iteration ($iter^*$) is reached. To decide on which sampling strategy should be used in the proposed metamodeling procedure, previously, we analyzed the performances of different sampling strategies. The sampling experiments showed that AdS_sd and AdS_cvar provide training sets with higher quality and few instances compared to OsS. Although both AdS_sd and AdS_cvar revealed very similar performances, AdS_cvar performed slightly better in DSM with tuned *mtry*. Therefore, in this section, we use AdS_cvar in the proposed metamodeling procedure.

Moreover, as discussed in Section 3.3, we need to decide on which elimination strategy we will use in the proposed procedure. In literature, there is yet to be a consensus on whether RFE or NRFE is better. While Svetnik *et al.* [21] warn about the overfitting risk of RFE, Gregorutti *et al.* [18] claim that no overfitting is observed with RFE. Therefore, we conducted preliminary experiments to decide which elimination strategy is best to use in the proposed metamodeling procedure. After selecting the best elimination strategy, we proceed to the experiments on the proposed procedure.

In the experiments on the proposed procedure, we aim to evaluate the performance of the proposed metamodeling procedure from several aspects. Firstly, we will compare the performances of the proposed procedure with AdS_cvar applied for DSM to see whether feature elimination improves the performance of adaptive sampling. Secondly, we will examine whether or not feature elimination can overcome the drawbacks due to the insignificant features by comparing the proposed procedure with AdS_cvar applied for SM. Lastly, we touch on the importance of *mtry* in the proposed procedure.

5.5.1. Experimental Design

Since elimination can be only applied for DSM, 11 initial training sets with 50 instances and a test set generated previously with DSM are used in the experiments. These data sets are the same as we used in the previous experiments conducted to evaluate sampling strategies for DSM. The proposed metamodeling procedure is replicated 10 times in each training set.

The detailed flow of the proposed metamodeling procedure is already introduced in Chapter 4 (Figure 4.1). The pseudo-codes of the adaptive sampling procedure and feature elimination procedure are also shared previously in Figure 4.2 and Figure 4.3. Briefly, an RF metamodel (M) is trained on the initial training set (T) and the performance of M on the test set is recorded in terms of RMSE. Then, the adaptive sampling procedure is performed. A pool of unlabeled instances (U) is produced and the training candidates (c) are selected from the most informative unlabeled instances in U. The outputs of c are queried to the simulation model. If the current iteration is later than the predefined iteration to activate feature elimination ($iter^*$), the feature elimination procedure is applied before adding recently labeled training candidates (C) to the training set. To prepare for the feature elimination, the features are ranked according to their importance scores. Depending on the feature elimination strategy, the importance ranking of the features is updated either at every iteration (RFE) or only at $iter^*$ (NRFE). Within the feature elimination procedure, an iterative process is followed to decide on the features to be eliminated. Initially, a specific proportion (p) of features including the least important features are selected for elimination. An RF model is trained without the selected features and its performance in terms of OOB error is monitored. If the change in OOB error is satisfactory (i.e. Cond. 4.2 holds), the feature elimination procedure terminates and the selected features are deleted from the feature list. Otherwise, the proportion of features selected for elimination is decreased until an improvement in OOB error is achieved. If all remaining features are significant, the feature elimination process terminates without any elimination decisions. After the feature elimination process is completed, C is added to the training set. At the next

iteration, M is trained on the new training set only with the remaining features. Although new instances are selected and features to be eliminated are decided at the current iteration, the effects of these decisions are observed at the next iteration. The adaptive sampling and feature elimination procedures are iteratively applied until the iteration budget (I) is reached. At the final iteration, M is trained for the last time and its performance on the test set is recorded as the final performance. As a result, at the end of the procedure, the final training set, the final metamodel, and the significant features highly related to the output are obtained.

Table 5.2: Parameter values used in the proposed procedure experiments.

Parameters used in the Proposed Procedure Experiments		
I	Iteration budget	11
h	Number of instances selected at each iteration	5
u	The number of instances in U	100
mm	Mtry multiplier	1/2
ntree	Hyperparameter of RF	300
p	Elimination fraction	0.5
o	The allowed increase rate in OOB error	10%
iter*	The activation iteration of feature elimination procedure	5
	Elimination type	NRFE/RFE
	Sampling strategy	AdS_cvar

In the experiments, I, h, u and $ntree$ parameters are kept at the same values as they were in the previous sampling strategy experiments to compare the proposed procedure with AdS_cvar (Table 5.2). Regarding elimination type, we decide which type to use after we check the performances of both RFE and NRFE. In addition, new parameters such as the predefined iteration to activate feature elimination procedure ($iter^*$), an $mtry$ multiplier to control the value of $mtry$ parameter of RF metamodel throughout the proposed procedure (mm), a predefined fraction determining the maximum number of features that can be eliminated at once (p), and the allowed increase rate in OOB error defined by the user (o) are introduced in the proposed metamodel-

ing procedure. While the last three parameters are related to the feature elimination process, *iter** locates the feature elimination procedure within the proposed metamodeling procedure. In other words, *iter** determines at which iteration of the proposed procedure the feature elimination procedure is applied for the first time. If *iter** is set to a small number, the elimination procedure becomes active starting from the early iterations of the proposed procedure. Due to the effect of the initial training set on the metamodel performance, the importance scores of the features may not be estimated correctly in the early steps of the proposed procedure in some data sets. Therefore, small *iter** holds the risk of unreliable elimination. On the other hand, if a large number lower than iteration budget is selected as *iter**, then the elimination of all insignificant features may not be possible. Based on some analyses conducted to figure out the best *iter** value, we decided to activate the feature elimination procedure when the training set size reaches to 70 ($iter^* = 5$) in the experiments. Among parameters controlling the feature elimination procedure, *mm* is used to manipulate the *mtry* parameter of RF. The *mtry* parameter of RF metamodel is determined by

$$mtry = min(mtry_{default} * mm, k) \tag{5.6}$$

where $mtry_{default}$ is the default *mtry* value and calculated by Eq 5.5, and k is the number of remaining features. In the experiments, we compare the performances when *mm* equals to 1 and 2. Secondly, p controls the elimination speed. The maximum number of features that can be eliminated at once is determined by p depending on the number of remaining features. When p is set to 0.5, the least important half of the features are considered for elimination at first. If the set of selected features includes important features, then the set is reduced. If p is set to a larger fraction, then more than half of the features are selected for elimination. Since very large or very small p values may cause inefficiency in the elimination process, values around 0.5 are suggested for p. Consistently, in the experiments, we set p to 0.5. Lastly, o is introduced to allow the user to manipulate the final set of significant features. The total number of eliminated features tends to increase with o. Although considerably significant features are potentially left at the end of the proposed procedure, the predictive performance

of the metamodel can be negatively influenced due to the small number of remaining features. In this study, since the importance scores of significant features are clearly different from each other and much higher than the scores of the insignificant features, any changes in the elimination decisions with respect to o are not observed. However, o can be actively utilized in the metamodeling of high-dimensional simulation models having parameters with different importance scores. Indeed, a similar notion to o is also used in [16].

5.5.2. Analyses on *iter**

The proposed metamodeling procedure starts with a small training data set. As we found out in the previous experiments, in the early steps of the adaptive sampling procedures (i.e. AdS_cvar), the performance of RF metamodel is highly influenced by the training set. With additional informative instances joining to the training set at each iteration, the prediction accuracy of RF metamodel increases and its feature importance estimations become more reliable. Therefore, the importance ranking of features can be unstable, especially, at the early iterations of the procedure. Besides, the importance measure is also influenced by the *mtry* parameter of RF. If *mtry* is selected poorly, the variable importance scores can be calculated inaccurately. Hence, some significant features might be mistaken as insignificant or the reverse. After a sufficient number of training instances are collected, the additional instances would not lead to dramatic fluctuations in variable importance scores.

In Figure 5.10, the average importance scores of each feature over 10 replications are represented for every iteration of AdS_cvar applied in three specific data sets among 11 training sets. The effect of tuning the *mtry* parameter of RF is also visible. When RF is trained with default *mtry* value, it fails to distinguish significant features (such as *density*) from insignificant features in these specific data sets (data set 4, 6 and 9) in the beginning. The importance scores of *budget-multiplier-dummy*, *noise-dummy*, and *tick-limit* are estimated higher than the importance score of *density* at the first iteration, in data set 4, 6 and 9, respectively. However, starting from the second

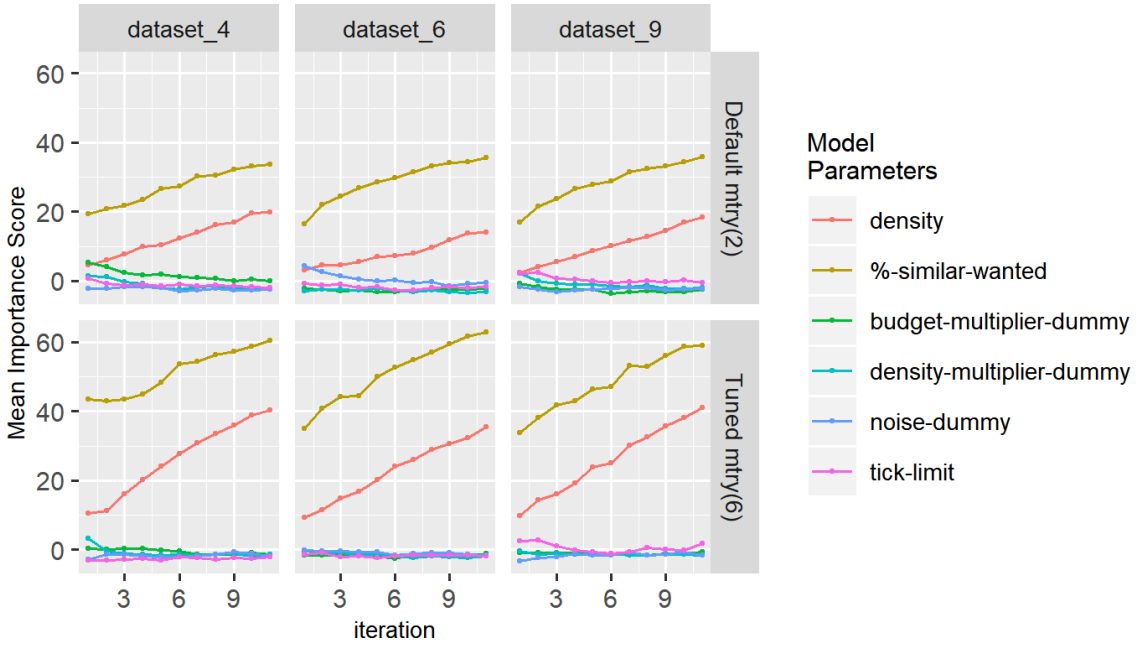

Figure 5.10: Change in importance scores of features in three initial training sets.

iteration, the importance scores of significant features show increasing trends while the importance scores of insignificant features keep the similar course. On the other hand, if the *mtry* parameter of RF is tuned, RF can successfully distinguish between significant and insignificant features even in the early steps.

However, it is desirable to come up with a rule of thumb value for *iter** independent of the *mtry* value. Moreover, in DSM, the separation between significant and insignificant features is very apparent since we manually introduced dummy parameters. For simulation models with a large number of parameters, the importance scores of their corresponding features can be close to each other and identification of significant features may require more sampling iterations. In such cases, problems regarding the reliability of the feature elimination procedure can arise if elimination is activated too early.

To suggest a general *iter** value, we searched for the first iteration at which the importance ranking of features does not dramatically change anymore. Therefore, we

first assigned features into *importance groups* according to their importance scores so that the small fluctuations in individual importance scores are ignored. Then, the change in the ranking of the importance groups is monitored. If features stay their importance groups and the ranking of the importance groups remains the same for at least two iterations, it is assumed that elimination can start. We performed this analysis on two data sets with five replications. Based on these 10 cases, it is observed that fifth iteration is appropriate to start elimination.

Nevertheless, this observation mainly depends on the number of features (k), I, and p. Therefore, a comprehensive suggestion should also consider these factors. Consider that I, p, and k are 0.5, 11, and 6, respectively. Elimination can take place at most three iterations in case five out of six features are insignificant (the number of features left: 6→3→2→1). In other words, an elimination process starting at the fifth iteration is able to eliminate all of the insignificant features and adaptive sampling can be performed with the presence of only the significant features. For any setting, *iter** can be decided by taking these considerations into account. If there are k features, and p is 0.5 then the maximum number of steps at which elimination takes place can be calculated as $E = \lceil \log_2 k \rceil$.

Since adaptive sampling can perform better without insignificant features, selecting at least E iterations for adaptive sampling to be applied only with significant features sounds reasonable. Given that iteration budget (I) is larger than $2 * E$, *iter** can be computed as I–$2 * E$. For instance, in the case of DSM, since there are six features and I is 11, *iter** can be set to five (11-3*2 = 5). If there were eight features, *iter** would be three (11 – 2*4 = 3). In case *iter** is calculated as a very small number, it is advised that the iteration budget should be increased for the sake of accuracy of the metamodeling procedure.

As a result of these analyses, in the experiments, we set *iter** to five.

5.5.3. Results

Before moving to the comparative analysis of the proposed metamodeling procedure with AdS_cvar, we first share the results of experiments we conducted to figure out which elimination strategy suits the best for the proposed metamodeling procedure. In Figure 5.11, the average performance of the proposed procedure at each iteration over 11 data sets and 10 replications is represented based on the elimination strategy in use. While the feature importance scores calculated at $iter^*$ (i.e. fifth iteration) is considered for the rest of the iterations with NRFE, the importance scores of the features are updated at every iteration with RFE. Figure 5.11 clearly points out that there is no significant difference between RFE and NRFE regardless of *mm* parameter.

However, this result cannot be generalized. Due to the gap between the importance scores of significant features and insignificant features, the metamodel can easily identify the insignificant features. Similarly, since the gap among the significant features is considerably large in terms of importance scores (Figure 5.12), the importance ranking of the significant features mostly does not change throughout the proposed procedure. Therefore, NRFE and RFE perform almost the same in the experiments. Unfortunately, our observations do not clearly point out the advantages and disadvantages of both RFE and NRFE. Still, it is important to show that NRFE or RFE does not necessarily perform better than the other. We also confirm that the proposed metamodeling procedure is able to eliminate all of the four insignificant features.

On the other hand, the influence of the *mm* parameter on the performance of the proposed metamodeling procedure is visible on both Figure 5.11 and Figure 5.13. Therefore, for the rest of the experiments, we monitor the performance for two different *mm* values. We call the proposed procedure as AdS&Fe_mm1 and AdS&Fe_mm2 when *mm* is set to 1 and 2, respectively. In Figure 5.13, the change in the average performances of AdS&Fe, AdS_cvar, and OsS throughout the iterations are represented based on the *mtry* value used in AdS and OsS. Each point is calculated by taking the average RMSE of metamodels initially trained on 11 data sets and replicated 10 times.

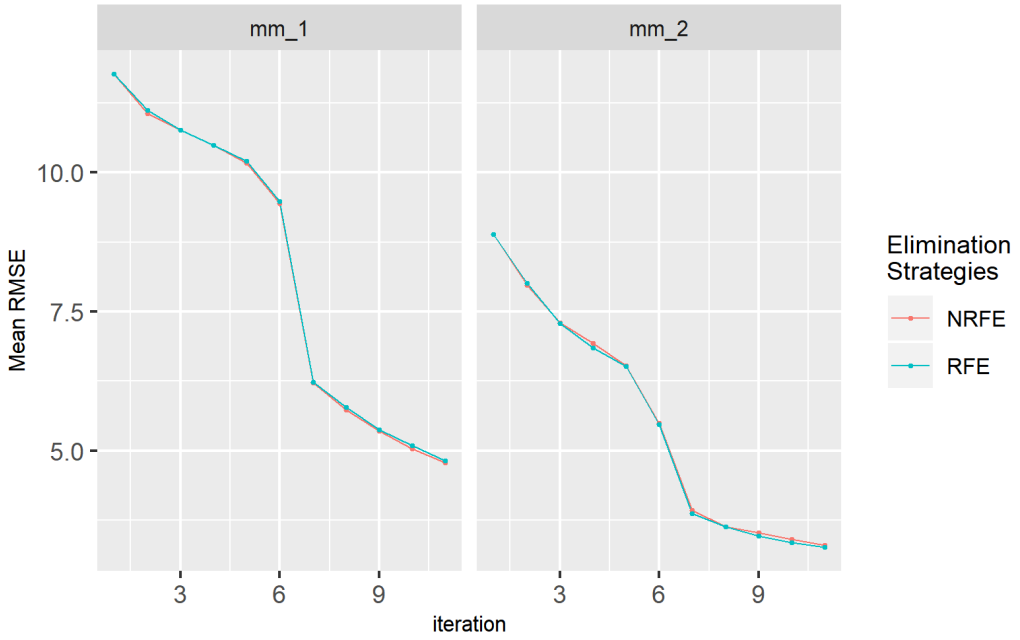

Figure 5.11: Change in the average test errors with NRFE and RFE based on *mm* parameter.

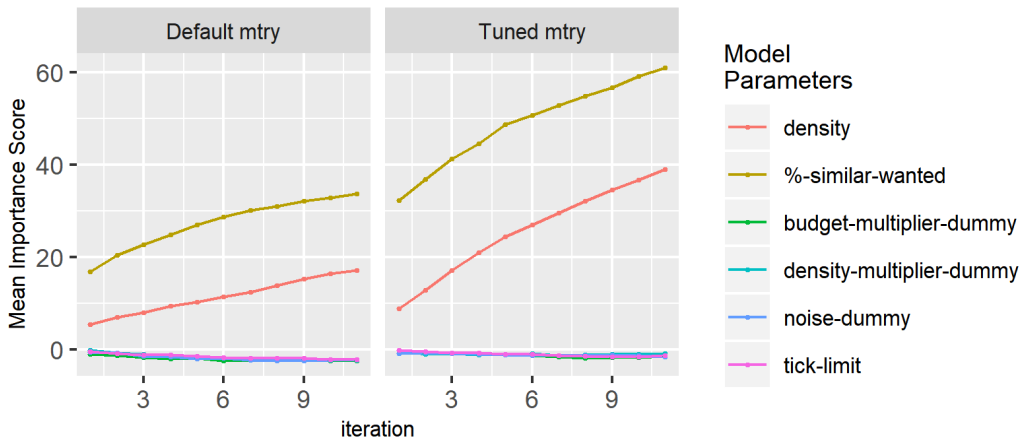

Figure 5.12: Average importance scores of simulation model parameters with AdS_cvar.

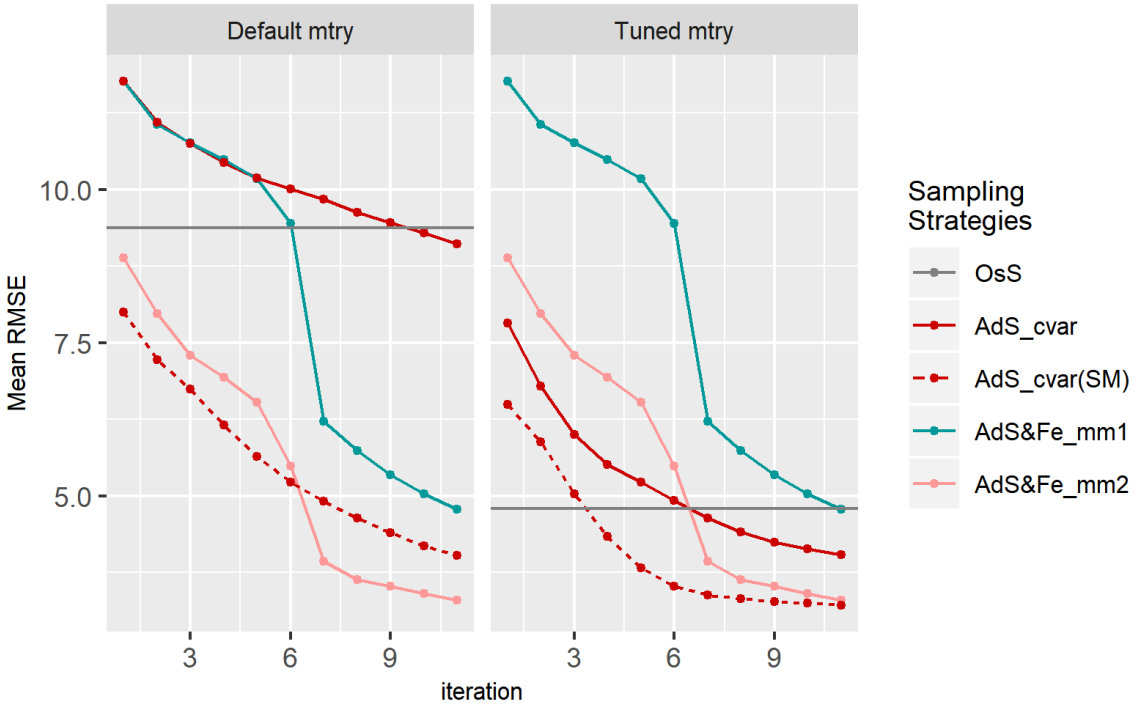

Figure 5.13: Average performances with OsS, AdS_cvar and AdS&Fe.

Figure 5.13 confirms that the elimination of insignificant parameters improves the metamodel performance substantially. The average RMSE sharply decreases with their elimination at the sixth and seventh iterations of AdS&Fe. In the end, AdS&Fe reduces the mean RMSE of metamodels by roughly 60%. Also, the comparison between AdS_cvar with default *mtry* and AdS&Fe_mm1 is meaningful to comprehend the contribution of feature elimination to metamodel performance since both utilize RF metamodel with default *mtry* value. Even though they start with the same performance (i.e. same initial training sets), AdS&Fe_mm1 achieves an accuracy level almost 1.8 times as high as AdS_cvar by getting rid of insignificant features, on average. Besides, while the performance of AdS_cvar highly depends on *mtry* due to the presence of insignificant features, AdS&Fe weakens this dependency by feeding the metamodel only with significant features.

Figure 5.13 also demonstrates that AdS&Fe performs better when *mm* is 2. The performance of AdS&Fe improves by about 30% with *mm*. Consistent with our find-

ings in the previous experiments on sampling strategies, an *mtry* value larger than the default significantly enhances the performance of the metamodel, on average. Although *mtry* tuning considerably improves the performance of AdS_cvar (red solid line), AdS&Fe_mm2 can achieve better performance. On top of that, AdS&Fe_mm2 uses initially suboptimal *mtry* (until the seventh iteration) unlike AdS_cvar using the best *mtry*. It is clear that AdS&Fe_mm2 is superior to AdS regardless of *mtry* value for DSM. Therefore, rather than spending time on tuning the *mtry* parameter of RF, AdS&Fe can easily provide higher performance than OsS and AdS only by selecting the *mm* parameter larger than 1.

Moreover, AdS&Fe_mm2 shows a very competitive performance compared to AdS_cvar applied for SM (AdS_cvar(SM)). To be clear, the dashed red lines in Figure 5.13 represent the performance of AdS_cvar when only significant features exist from the beginning. Without any tuning efforts, AdS&Fe_mm2 yields lower RMSE than AdS_cvar(SM) does, on average. In other words, AdS&Fe_mm2 mitigates the drawbacks of not tuning *mtry* parameter of RF metamodel by selecting a convenient *mm* and easily manipulating *mtry*. More importantly, AdS&Fe_mm2 performs almost as good as AdS_cvar(SM) applied with tuned *mtry*. This reveals that AdS&Fe_mm2 considerably recovers the performance loss due to the presence of insignificant features. Overall, AdS&Fe helps to overcome the negative impacts of insignificant features on sampling by discarding them and enabling the metamodel and sampling to focus only on significant features.

A more detailed comparison between AdS_cvar(SM) and AdS&Fe_mm2 can be made based on Figure 5.14. While AdS_cvar(SM) is initialized with training sets having only two significant features, initial training sets with additional four insignificant features are used for AdS&Fe_mm2. Also, while the performance of AdS_cvar(SM) is calculated by using the best *mtry* value for SM, a suboptimal *mtry* value for DSM is used to measure the performance of AdS&Fe_mm2 until the last iteration at which elimination is realized (i.e. the seventh iteration of the procedure). At the first five iterations, *mm* sets *mtry* to 4 while the best *mtry* is 6. At the sixth iteration, *mtry*

becomes 2, which is still smaller than the best *mtry*. Starting from the seventh iteration, *mm* sets *mtry* to its tuned value. Even in these conditions, AdS&Fe_mm2 reduces the RMSE value of the metamodel and provides a final performance almost as successful as AdS_cvar(SM) does. Although AdS_cvar(SM) is more successful in reducing the variance in the performance, the range of the final boxplot of AdS&Fe_mm2 can be considered small enough to ensure the robustness of the performance.

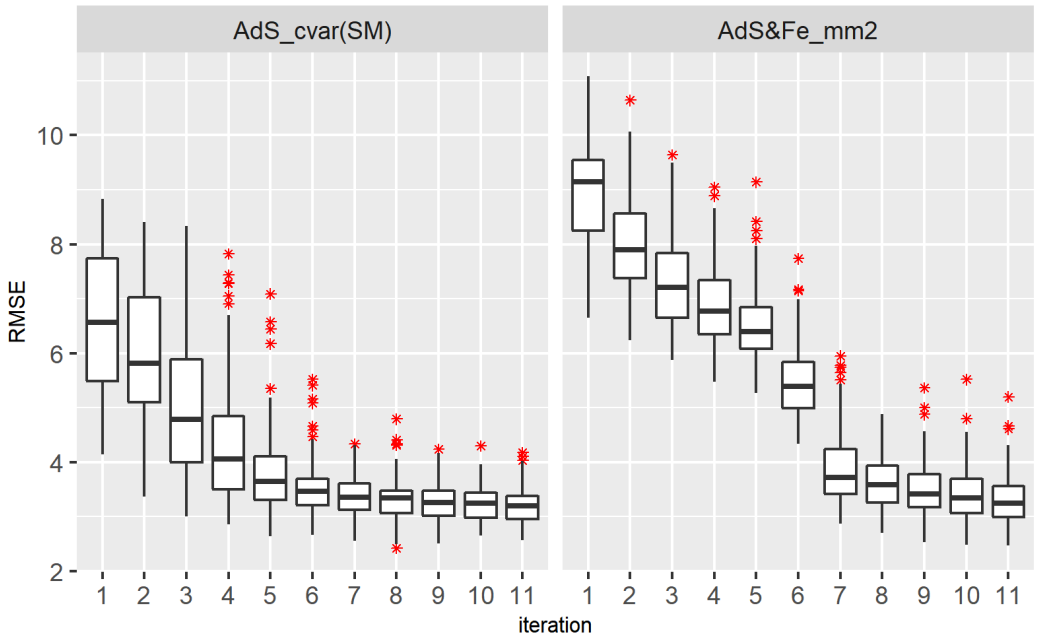

Figure 5.14: Boxplots of test errors of metamodels trained with AdS_cvar and AdS&Fe_mm2.

6. DISCUSSION

In the experiments, we evaluated the effect of various sampling strategies on metamodel training and the performance of the proposed metamodeling approach compared to the best sampling strategy selected. Besides, we discussed the impact of dummy parameters on the performance of the metamodel and sampling. We also expanded our experiments to monitor the influence of *mtry* parameter of RF.

We observed that adaptive sampling strategies such as AdS_sd and AdS_cvar generally decreases the variance in RMSE of RF metamodel over time. As the training set grows in size, the performance of the metamodel becomes independent from the initial training set. Moreover, AdS_sd and AdS_cvar achieve to provide the metamodel with a highly informative, yet relatively small training set compared to OsS. Hence, AdS_sd and AdS_cvar contribute to both robustness and the performance of the metamodel.

The comparisons between AdS_cvar and AdS&Fe point that feature elimination can further enhance the performance of the metamodel. The elimination of the insignificant features enables the metamodel to extract the information on the data set better and improves the sampling process by highlighting the significant features. Besides, the experiment confirmed that AdS&Fe is capable of distinguishing and removing dummy parameters of DSM. As a result, AdS&Fe can almost recover the performance loss caused by insignificant parameters. Moreover, robustness of AdS&Fe to *mtry* parameter of RF metamodel is stronger compared to AdS.

The metamodel accuracy is evaluated on a single test set including 100 instances for SM and DSM, separately. When we measure the metamodel accuracy in larger test sets having 200, 300, and 400 instances, we did not spot any considerable change in the results. Therefore, the performance of the metamodels can be considered unbiased regarding the size of the test set.

Nevertheless, the results of experiments clearly showed that the performance of the proposed metamodeling procedure highly depends on the *mtry* parameter of RF. Especially when there are insignificant features in the data set, *mtry* plays a more critical role in achieving a successful metamodel. Unless the *mtry* parameter of RF metamodel is selected carefully, the proposed metamodeling procedure may not be able to yield a satisfactory metamodel in terms of predictive performance. Unfortunately, tuning the *mtry* parameter at the beginning of the procedure and after each elimination can be quite computationally expensive, especially in the metamodeling of high-dimensional simulation models. Therefore, instead of frequently searching for the best *mtry* value, an *mtry* value larger than the default can also provide an RF metamodel with competitive performance. The sampling strategy experiments revealed that a sufficiently large *mtry* value enables RF to relieve the nuisance due to insignificant features and construct more decision trees based on significant features. Although the best *mtry* values for RF metamodels were equal to the number of parameters of SM and DSM, this result cannot be generalized. Besides, selecting *mtry* equal to the number of parameters, indeed, decreases the diversity of the decision trees of an RF model and reduces the potential performance gain obtained from averaging the predictions of diverse decision trees. Also, it can slow down the metamodeling procedure. For these reasons, an *mtry* value between the default and the number of features can be selected. Depending on the number of features, an *mtry* value closer to the number of features can be promising to obtain better metamodel accuracy.

Regarding the findings on *mtry*, we aimed to control the *mtry* parameter by introducing a multiplier (mm) to the proposed metamodeling procedure. Selecting mm more than one simply implies *mtry* value larger than the default. The *mtry* value can be also automatically set closer to the number of features by increasing the mm value. The experiments showed that a good choice of mm can yield successful metamodels without tuning of *mtry*. When mm is selected as 2, AdS&Fe surpasses AdS_cvar applied for DSM and almost catches up AdS_cvar applied for SM in terms of metamodel accuracy. On top of that, AdS&Fe can achieve such a competitive performance with a suboptimal *mtry* value until the seventh iteration. Based on these

findings, the proposed metamodeling procedure can considerably mitigate the need for parameter tuning to provide successful metamodels. For convenience, setting *mm* to 2 is potentially promising for regression. However, it is advised to increase *mm* according to the number of features in classification as the default *mtry* is not proportional to the number of features in classification.

In addition to *mtry*, we also examined the effect of the *ntree* parameter of RF. However, unlike *mtry*, *ntree* had a little influence on the metamodel performance. Therefore, we did not detail our experiments on the *ntree* parameter.

Settles [31] claims that diversity among h unlabeled instances selected at each iteration (i.e. training candidates) generally contributes to achieving better results. Therefore, we checked the effect of the diversity among h unlabeled instances selected at each iteration (i.e. training candidates) by conducting small-scale experiments. We monitored the change in the metamodel accuracy with respect to the number of randomly picked instances within training candidates in the proposed metamodeling procedure (AdS&Fe). It is observed that diversity within the training candidates obtained by replacing the least informative instances with randomly selected instances does not promote the performance of the metamodel. As the number of randomly picked instances increases in the training candidates, the accuracy of the metamodel decreases.

Although segregation model is a good choice to test a metamodeling approach due to its structural simplicity and behavioral complexity [27], the small number of model parameters and the clear separation between these parameters in terms of their influence on the output did not let us examine the impact of some elements of the proposed procedure. For instance, we did not observe any difference between the performances of NRFE and RFE strategies. Similarly, changing the values of some parameters of the proposed procedure such as o and p did not result in obvious differences in the proposed metamodeling procedure. However, for a high dimensional simulation model, NRFE and RFE strategies may bring about different performances and the parameters

such as o and p can potentially affect the metamodeling procedure.

		h	
		small	*large*
I	*small*	Only limited number of samples are added to the training set. The metamodeling procedure may not be able to produce a sufficiently accurate metamodel. Not all insignificant features may be eliminated.	More than enough unlabeled instances with information overlaps are selected inefficiently. The metamodel is not efficiently utilized to guide sampling. Not all insignificant features may be eliminated.
	large	The increased number of metamodel trainings and sampling processes causes potential inefficiency in the proposed procedure.	If the best performance is achieved at early iterations, extra simulation evaluations and metamodel training causes inefficiency.

Figure 6.1: The possible risks resulted from the different I and h combinations.

Apart from the discussions on the performance of the proposed metamodeling procedure, discussions on the efficiency of the proposed procedure are also worthy. The duration of a single application of the proposed procedure mostly depends on the total number of simulations executed during the procedure and the average runtime of each simulation execution. Firstly, the total number of simulations is controlled by two parameters of the proposed procedure: the iteration budget (I) and the number of the training candidates selected at each iteration (h). For the sake of the efficiency of the proposed procedure, I and h should be selected accordingly. The possible risks of the different I and h combinations are described in Figure 6.1. Secondly, the average run length of simulations is generally higher when adaptive sampling strategies are employed. A simulation run lasts until either the the stopping criterion (i.e. Everyone is happy) is fulfilled or the run length limit (100 time steps) is hit. Since adaptive sampling generally selects instances leading to dramatic changes in the output [7], the evaluation

of their outputs in the simulation model mostly requires longer time and generally terminates at the 100th time step. Therefore, the proposed metamodeling procedure produces the final data set with 100 instances in a longer time compared to a space-filling one-shot strategy. However, the proposed procedure achieves higher metamodel accuracy than OsS starting from the seventh iteration, on average. This means that the metamodel accuracy obtained with OsS can be achieved in a shorter time with the proposed procedure. From another perspective, if OsS is wanted to achieve the same the metamodel accuracy obtained with the proposed procedure, the size of the training set should be considerably increased so that informative instances are included in the training set. Since the proposed procedure can provide higher accuracy with a smaller data set, it can be said that the proposed procedure is more efficient than the one-shot sampling. Similarly, the proposed procedure achieves better performance than adaptive sampling without elimination starting from the seventh iteration, on average, although their run times do not significantly differ in the experiments. However, we expect that the proposed procedure would last shorter than adaptive sampling without elimination when applied to a high-dimensional simulation model.

7. CONCLUSION AND FUTURE WORK

In this study, we aimed to develop an efficient and user-friendly metamodeling procedure combining adaptive sampling and feature elimination strategies so that a modeler or analyst can benefit from the proposed procedure in model analysis and exploration. Although we performed the automatized procedure on a small agent-based model, we tried to propose a general procedure also applicable to high-dimensional and complex simulation models. Indeed, we discussed each component of the proposed procedure such as the metamodel, adaptive sampling, and feature selection strategies with the underlying reasons behind the selected methods in each component. Therefore, a careful researcher can easily adapt the proposed procedure to a wide range of complex models by selecting appropriate methods.

To show the advantages of the proposed metamodeling procedure, we performed two sets of experiments. Firstly, we compared the sampling strategies in terms of metamodel accuracy and choose the best sampling strategy to use in the proposed procedure. Therefore, in the second set of experiments, we were able to easily assess the importance of feature elimination on achieving a good metamodel performance. Overall, we observed that adaptive sampling based on standard deviation and coefficients of variation perform better than one-shot sampling in terms of metamodel accuracy and reliability. The proposed metamodeling procedure can further boost the performance of the metamodel by successfully eliminating the insignificant features (i.e. dummy parameters in a simulation model). On top of that, the proposed procedure can alleviate the negative effects of insignificant features on metamodel accuracy. Furthermore, the procedure does not even require exhaustive hyperparameter tuning of RF.

The main advantage of the proposed procedure results from the simultaneous application of the adaptive sampling and feature elimination strategies. The feature elimination strategy simplifies the metamodel gradually and lets the metamodel focus on significant features. The concise metamodel can assess the informativeness of the

unlabeled instances more accurately and the sampling can feed the metamodel with more informative instances. Moreover, since the feature elimination strategy can reduce the dimensionality of the simulation model, the proposed procedure can potentially mitigate the curse of dimensionality in the metamodeling of simulation models with a large number of parameters.

Consequently, the proposed metamodeling procedure can produce a metamodel that sufficiently approximates the dynamics embedded in the simulation model. The metamodel can inform the modeler and analyst about the important parameters conditioning a specific model behavior and contributes to the interpretability of the simulation model. Moreover, the procedure provides a high-quality data set that can be used in further analyses. As a result, modelers and analysts can utilize the proposed procedure to reduce the time required for comprehensive model analysis, to gain awareness about the model behaviors and to detect the model parameters conditioning the model behavior. If the RMSE of the metamodel is small enough, the metamodel can be even used as a policy simulator.

While developing this metamodeling procedure, we focused on achieving a highly accurate metamodel without sacrificing from the efficiency and user-friendliness of the metamodeling procedure. At the same time, we sought to provide insights into the parameters and dynamics of the simulation model. Hence, our decisions regarding the selection of metamodel, sampling strategy, and feature selection strategy are mainly made based on these considerations. To encourage analysts/modelers to use the proposed procedure, we favored a self-sufficient metamodeling procedure. All the components of the proposed procedure are chosen consistently and coherently. As a result, we built the proposed metamodeling procedure with RF metamodel, QBC sampling framework, and permutation-based VIM. However, depending on the users' concerns and the characteristics of the data sets, different methods can be used instead of the ones we selected.

Similarly, the performance of the proposed procedure can be potentially further improved with alternative methods. For instance, XGBoost, which stands out with its fast and successful performance, can be replaced with RF metamodel and its variable importance measure can be utilized instead of permutation-based VIM. As future work, the variants of the proposed metamodeling procedure can be compared in terms of metamodel accuracy and efficiency of the procedure.

In this study, we did not inspect the modality of the output although Edali and Yücel [7] state that multimodality in regression settings is likely to restrict the metamodel from successfully approximating the simulation model with continuous output. However, they also found that the segregation model has a unimodal output distribution, meaning that there is no need to make any transformation in the output to obtain a successful metamodel. Yet, when the proposed metamodeling procedure is applied to a different simulation model, it is suggested to perform modality detection on the model output to obtain better metamodel performance. Similarly, if the proposed metamodeling procedure is applied to a simulation model with categorical variables, the balance between the classes of the output should be considered in metamodel training and sampling for better predictive performance. In summary, extensions regarding the multimodality and class imbalance problems can be added to the proposed procedure in the future to support the applicability of the proposed procedure to a wide range of simulation models.

Finally, the proposed metamodeling procedure should be performed on a large simulation model in the future. The proposed procedure can potentially overcome the curse of dimensionality due to the presence of the insignificant features and obtain an efficient metamodel of a high-dimensional simulation model. Due to the limited computational resources, we were not able to test the performance of the proposed procedure on a high-dimensional and large-scale simulation model. However, the extent of the benefits of the proposed metamodeling procedure can only be measured by applying the procedure to a high-dimensional simulation model.

REFERENCES

1. Lee, J.-S., T. Filatova, A. Ligmann-Zielinska, B. Hassani-Mahmooei, F. Stonedahl, I. Lorscheid, A. Voinov, J. G. Polhill, Z. Sun and D. C. Parker, "The Complexities of Agent-Based Modeling Output Analysis", *Journal of Artificial Societies and Social Simulation*, Vol. 18, No. 4, p. 4, 2015.

2. Macal, C. and M. North, "Tutorial on agent-based modelling and simulation", *Journal of Simulation*, Vol. 4, No. 3, pp. 151–162, 2010.

3. Arroyo, J., S. Hassan, C. Gutiérrez and J. Pavón, "Re-thinking simulation: A methodological approach for the application of data mining in agent-based modelling", *Computational and Mathematical Organization Theory*, Vol. 16, No. 4, pp. 416–435, 2010.

4. Lempert, R., "Agent-Based Modeling as Organizational and Public Policy Simulators", *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 99, pp. 7195–7196, National Academy of Sciences, 2002, `https://www.jstor.org/stable/3057839`.

5. Patel, M. H., M. A. Abbasi, M. Saeed and S. J. Alam, "A scheme to analyze agent-based social simulations using exploratory data mining techniques", *Complex Adaptive Systems Modeling*, Vol. 6, No. 1, pp. 1–17, 2018.

6. Crombecq, K., D. Gorissen, D. Deschrijver and T. Dhaene, "A novel hybrid sequential design strategy for global surrogate modeling of computer experiments", *SIAM Journal on Scientific Computing*, Vol. 33, No. 4, pp. 1948–1974, 2011.

7. Edali, M. and G. Yücel, "Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling", *Simulation Modelling Practice and Theory*, Vol. 92, No. 2018, pp. 62–81, 2019.

8. Lamperti, F., A. Roventini and A. Sani, "Agent-based model using machine learning surrogates", *Journal of Economic Dynamics and Control*, Vol. 90, pp. 366–389, 2018.

9. van der Hoog, S., "Surrogate Modelling in (and of) Agent-Based Models: A Prospectus", *Computational Economics*, Vol. 53, No. 3, pp. 1245–1263, 2019.

10. Liu, H., Y. Ong and J. Cai, "A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design", *Structural and Multidisciplinary Optimization*, pp. 393–416, 2017.

11. Barton, R., "Simulation Optimization using Metamodels", *Proceedings of the 2009 Winter Simulation Conference*, pp. 230–238, 2009.

12. Kleijnen, J. P. C., "Kriging metamodeling in simulation: A review", *European Journal of Operational Research*, Vol. 192, No. 3, pp. 707–716, 2009.

13. Kleijnen, J. P. C., "Regression and Kriging metamodels with their experimental designs in simulation: A review", *European Journal of Operational Research*, Vol. 256, No. 1, pp. 1–16, 2017.

14. Salle, I. and M. Yıldızoğlu, "Efficient sampling and meta-modeling for computational economic models", *Computational Economics*, Vol. 44, No. 4, pp. 507–536, 2014.

15. Bologna, G. and Y. Hayashi, "A Comparison Study on Rule Extraction from Neural Network Ensembles, Boosted Shallow Trees, and SVMs", *Applied Computational Intelligence and Soft Computing*, Vol. 2018, pp. 1–20, 2018.

16. Diaz-Uriarte, R. and S. Alvarez de Andrés, "Gene selection and classification of microarray data using random forest", *BMC Bioinformatics*, Vol. 7, pp. 1–13, 2006.

17. Genuer, R., J. M. Poggi and C. Tuleau-Malot, "Variable selection using random

forests", *Pattern Recognition Letters*, Vol. 31, No. 14, pp. 2225–2236, 2010.

18. Gregorutti, B., B. Michel and P. Saint-Pierre, "Correlation and variable importance in random forests", *Statistics and Computing*, Vol. 27, No. 3, pp. 659–678, 2017.

19. Jiang, H., Y. Deng, H. S. Chen, L. Tao, Q. Sha and J. Chen, "Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes", *BMC Bioinformatics*, Vol. 5, pp. 1–12, 2004.

20. Strobl, C., A. L. Boulesteix, T. Kneib, T. Augustin and A. Zeileis, "Conditional variable importance for random forests", *BMC Bioinformatics*, Vol. 9, pp. 1–11, 2008.

21. Svetnik, V., A. Liaw, C. Tong and T. Wang, "Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules", *Lecture notes in computer science: Multiple classifier systems*, Vol. 3077, pp. 334–343, 2004.

22. Verikas, A., A. Gelzinis and M. Bacauskiene, "Mining data with random forests: A survey and results of new tests", *Pattern Recognition*, Vol. 44, No. 2, pp. 330–349, 2011.

23. Borisov, A., E. Tuv and G. C. Runger, "Active Batch Learning with Stochastic Query-by-Forest (SQBF)", *Proceedings of Machine Learning Research*, Vol. 16, pp. 59–69, 2011.

24. Hastie, T., R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, Springer, 2009.

25. Guyon, I. and A. Elisseeff, "An introduction to variable and feature selection", *Journal of Machine Learning Research*, Vol. 3, pp. 1157–1182, 2003.

26. Breiman, L., "Random forests", *Machine Learning*, Vol. 45, pp. 5–32, 2001.

27. Edali, M., *Analysis of Agent-Based Simulation Models Through Metamodeling*, Ph.D. Thesis, Boğaziçi University, 2019.

28. Sheikholeslami, R. and S. Razavi, "Progressive Latin Hypercube Sampling: An efficient approach for robust sampling-based analysis of environmental models", *Environmental Modelling and Software*, Vol. 93, pp. 109–126, 2017.

29. Kucherenko, S., D. Albrecht and A. Saltelli, "Exploring multi-dimensional spaces: a Comparison of Latin Hypercube and Quasi Monte Carlo Sampling Techniques", *arXiv: Applications*, pp. 1–30, 2015, `http://arxiv.org/abs/1505.02350`.

30. Crombecq, K., E. Laermans and T. Dhaene, "Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling", *European Journal of Operational Research*, Vol. 214, No. 3, pp. 683–696, 2011.

31. Settles, B., "Active Learning Literature Survey", *Machine Learning*, Vol. 15, No. 2, pp. 201–221, 2010.

32. Fang, K. T., D. K. J. Lin, P. Winker and Y. Zhang, "Uniform design: Theory and application", *Technometrics*, Vol. 42, No. 3, pp. 237–248, 2000.

33. Seung, H. S., M. Opper and H. Sompolinsky, "Query by committee", *In Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp. 287–294, 1992.

34. Freund, Y., H. S. Seung, E. Shamir and N. Tishby, "Selective Sampling Using the Query by Committee Algorithm", *Machine Learning*, Vol. 28, No. 2–3, p. 133–168, 1997.

35. Mwangi, B., T. Tian and J. Soares, "A review of feature reduction techniques in Neuroimaging", *Neuroinformatics*, Vol. 12, No. 2, pp. 229–244, 2013.

36. Stumpf, A., N. Lachiche, N. Kerle and A. Puissant, "Adaptive sampling and object feature selection for landslide mapping using Random Forests", *Proceedings of GEOBIA 2012, the 4th Geobia : Geographic object - based image analysis , May 7-9, 2012 Rio de Janeiro Brazil*, pp. 273–278, Brazilian National Institute for Space Research (INPE), 2012.

37. Blum, A. and P. Langley, "Selection of relevant features and examples in machine learning", *Artificial Intelligence*, Vol. 97, pp. 245–271, 1997.

38. Hapfelmeier, A. and K. Ulm, "A new variable selection approach using Random Forests", *Computational Statistics and Data Analysis*, Vol. 60, No. 1, pp. 50–69, 2013.

39. Wei, P., Z. Lu and J. Song, "Variable importance analysis: A comprehensive review", *Reliability Engineering and System Safety*, Vol. 142, pp. 399–432, 2015.

40. Strobl, C., A. L. Boulesteix, A. Zeileis and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution", *BMC Bioinformatics*, Vol. 8, 2007.

41. Kohavi, R. and G. H. John, "Wrappers for feature subset selection", *Artificial Intelligence*, Vol. 97, pp. 273–324, 1997.

42. Wilensky, U., *NetLogo Segregation model*, Northwestern University, Evanston, IL, 1997, `http://ccl.northwestern.edu/netlogo/models/Segregation`, accessed at August 2020.

43. Wilensky, U., *NetLogo*, Northwestern University, Evanston, IL, 1999, `http://ccl.northwestern.edu/netlogo/models/Segregation`, accessed at August 2020.

44. Schelling, T. C., "Dynamic models of segregation", *The Journal of Mathematical Sociology*, Vol. 1, No. 2, pp. 143–186, 1971.

APPENDIX A: DETAILED METAMODEL PERFORMANCES OBTAINED IN THE EXPERIMENTS

The RMSE values of metamodels yielded with different sampling strategies are represented in Table A.1 and Table A.2 for segregation model (SM), and in Table A.3 and Table A.4 for segregation model with dummy parameters (DSM). In all of these tables, each column reports the average RMSE of metamodels replicated 10 times for each of initial training set. While Table A.1 and Table A.3 show RMSE values of metamodels constructed with default *mtry* (1 for SM, 2 for DSM), Table A.2 and Table A.4 demonstrate the RMSE values obtained with tuned *mtry* (2 for SM, 6 for DSM).

The first column (Initial Training Set) implies 11 large training sets for OsS and 11 small initial training set for iterative sampling strategies such as RdS, AdS_range, AdS_sd, and AdS_cvar. Since the initial training set expands with iterative sampling strategies, metamodels are trained on different final training sets in each replication. As a result, RMSE values corresponding to iterative sampling strategies incorporate 10 metamodels trained on different training set with 100 instances, whereas OsS column reports the average test performance of 10 metamodels trained on the same training set with 100 instances.

Table A.1: Results for SM when $mtry = 1$ (default $mtry$).

Initial Training Set	OsS	RdS	AdS_range	AdS_sd	AdS_cvar
1	5.0036	4.9091	4.6447	3.9445	3.9183
2	5.0368	7.2038	4.8039	3.9440	3.8493
3	6.0812	6.5074	4.7690	3.8770	3.8324
4	4.9335	6.0501	5.5081	4.0009	3.9784
5	5.1938	5.3212	4.7234	3.9025	4.0202
6	4.4699	5.7615	4.6998	3.8820	3.8049
7	5.2972	6.2891	4.9895	4.1020	4.0129
8	5.7374	6.1333	5.2652	4.2258	4.4588
9	5.4510	5.8751	5.9189	4.6852	4.4454
10	5.2928	5.8831	4.3533	3.6927	3.8991
11	5.9927	6.4139	4.8691	4.0541	4.0446
Avg	5.3173	6.0316	4.9586	4.0282	4.0240

Table A.2: Results for SM when $mtry = 2$ (tuned $mtry$).

Initial Training Set	OsS	RdS	AdS_range	AdS_sd	AdS_cvar
1	4.6383	3.9646	3.2642	2.7588	2.9400
2	3.2822	5.9589	3.3312	3.0680	3.2984
3	5.1630	5.1934	3.3115	3.0022	3.0932
4	4.1701	4.6637	3.7426	3.7143	3.7395
5	4.4143	3.7680	2.8626	2.7878	2.9742
6	3.6601	4.6300	3.1050	2.8673	2.9129
7	4.8342	5.1810	3.4110	3.1646	3.3523
8	4.2149	4.5562	3.6031	3.3266	3.4238
9	5.5445	4.7837	3.7211	3.5221	3.5055
10	5.0629	4.6890	3.2118	2.9785	3.1728
11	5.7147	4.5838	2.9188	2.8401	2.9622
Avg	4.6090	4.7248	3.3167	3.0937	3.2159

Table A.3: Results for DSM when $mtry = 2$ (default $mtry$).

Initial Training Set	OsS	RdS	AdS_range	AdS_sd	AdS_cvar
1	8.7582	10.6867	10.6787	10.1164	10.2219
2	10.3836	10.5871	9.4757	9.3275	9.4841
3	9.7926	10.1369	9.7676	9.2342	8.9752
4	8.6714	9.5538	9.9972	9.4828	9.0906
5	9.0293	10.7761	9.3063	8.6268	8.6962
6	9.7891	10.6541	9.6449	8.7959	8.9388
7	9.6213	11.5540	9.8370	9.2394	9.6207
8	9.0962	9.9926	9.2738	8.4608	8.1543
9	10.0131	10.9068	10.1084	9.2711	9.3930
10	8.7343	9.4368	9.4946	8.5358	8.6196
11	9.2831	9.7554	9.9086	9.4749	9.1039
Avg	9.3793	10.3673	9.7721	9.1423	9.1180

Table A.4: Results for DSM when $mtry = 6$ (tuned $mtry$).

Initial Training Set	OsS	RdS	AdS_range	AdS_sd	AdS_cvar
1	4.3669	5.4655	6.2960	4.8680	4.0573
2	6.6235	5.6158	5.5625	4.6363	3.9061
3	6.2359	5.3224	5.0072	4.4078	4.2208
4	4.1164	5.7961	4.7985	4.6031	4.1841
5	3.8428	5.9165	4.2643	3.9765	3.8331
6	4.1570	5.2941	3.9040	4.0827	3.7938
7	5.1022	6.5954	4.5944	4.4024	4.1616
8	4.9667	6.7536	4.0446	3.9599	3.6739
9	4.5028	5.4478	5.1625	4.3669	4.0757
10	3.8010	4.1910	4.3598	4.1174	4.0547
11	5.0657	4.9226	6.2118	4.8501	4.4379
Avg	4.7983	5.5746	4.9278	4.3883	4.0363

Table A.5 represents the RMSE values of metamodels obtained with the proposed metamodeling procedure utilizing different mm and elimination strategies for each initial training set. Each value in the table show the average RMSE values of 10 replications.

Table A.5: Results obtained with the proposed metamodeling procedure.

Initial Training Set	NRFE ($mm = 1$)	RFE ($mm = 1$)	NRFE ($mm = 2$)	RFE ($mm = 2$)
1	6.3052	5.9749	3.1565	2.9330
2	4.6108	4.7179	3.6347	3.7030
3	4.8301	4.9212	3.1505	3.1437
4	4.1988	4.3200	3.5032	3.3337
5	4.3852	4.4074	2.8887	2.8398
6	4.1600	4.2696	3.0476	3.0999
7	5.3273	5.2421	3.2780	3.2117
8	4.7146	4.6977	3.3547	3.4025
9	4.7890	5.0190	3.3290	3.3138
10	4.1567	4.3232	2.9057	2.9137
11	5.1885	5.0899	4.0264	4.0456
Avg	4.7878	4.8166	3.2977	3.2673

APPENDIX B: PSEUDO-CODE OF THE PROPOSED METAMODELING PROCEDURE

The pseudo-code of the proposed metamodeling procedure with adaptive sampling and feature elimination strategy is represented in Fig. B.1.

Inputs : $I, p, h, iter^*, o$

Outputs : T, M

Generate initial T by using maximin LHS and the simulation model;
$iter \leftarrow 1$;
while $iter \leq I$ **do**
    Train M on T with feature set f;
    Set OOB as OOB error of M;
    Report RMSE of M on test set;
    if $iter \neq I$ **then**
        Generate U by using maximin LHS considering feature set f;
        Obtain predictions made by decision trees of M for all in U;
        Calculate v_u for each instance in U by using Eq. 5.4;
        Decide c by selecting h unlabeled instances with the highest v_u by using Eq 4.1;
        Obtain C by querying the outputs of c through simulation runs;

Figure B.1: Pseudo-code of the proposed metamodeling procedure (RFE).

```
13:     if iter ≥ iter* and k ≥ 2 then
14:       check_elim ← TRUE;
15:       Set k to the number of features in feature set f ;
16:       Update rank of feature set f;
17:       elim_iter ← 1;
18:       while check_elim do
19:         Calculate k′ by using Eq. 4.4;
20:         Obtain feature set f′ by eliminating k′ features with the lowest ranks
    from f;
21:         Train M′ on T with feature set f′;
22:         Set OOB′ as OOB error of M′;
23:         if OOB′ ≤ OOB ∗ (1 + o) then
24:           f ← f′;
25:           check_elim ← FALSE;
26:         else
27:           Calculate k′_next by using Eq. 4.4;
28:           if k′_next == k′ then
29:             check_elim ← FALSE;
30:           else
31:             elim_iter ← elim_iter + 1;
32:           end if
33:         end if
34:       end while
35:     end if
36:     T ← T ∪ C
37:     iter = iter + 1
38:   end if
39: end while
```

Figure B.1: – (cont.)