Simulation Modelling Practice and Theory 92 (2019) 62–81

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling

Mert Edali[*,a,b], Gönenç Yücel[a]

[a] Department of Industrial Engineering, Bogazici University, Bebek, Istanbul 34342, Turkey
[b] Department of Industrial Engineering, Yildiz Technical University, Besiktas, Istanbul 34349, Turkey

ARTICLE INFO

Keywords:
Agent-based modeling
Metamodeling
Random forest
Sequential sampling
Active learning
Output categorization

ABSTRACT

Agent-based modeling is an effective way of understanding and analyzing complex adaptive phenomena. In this respect, discovering the relationship between inputs and outputs of agent-based models is the ultimate way of providing insights into understanding the dynamics of the system being modeled. Therefore, there are many approaches in the literature to clarify these relationships including sampling and metamodeling. Emphasizing the weaknesses and disadvantages of current methods, we present a metamodel-guided sequential sampling technique which combines random forests and uncertainty sampling. Experimental results on two well-known agent-based models show that the presented technique yields metamodels of higher accuracy compared to metamodels trained with randomly selected input–output data. Contrary to the previous studies emphasizing only the improvement in the metamodel accuracy, we also focus on input parameter combinations selected by the sequential sampling technique, and we observe that sequential sampling is able to capture the boundaries of tipping point behaviors as well as the points exhibiting counter-intuitive behavior, thus, potentially aiding verification, validation, and understanding of agent-based models. Additionally, we propose a novel two-step method for the categorization of the agent-based model outputs prior to metamodel training, which helps the analyst to decide on whether preserving numerical model outputs or continuing the metamodel training procedure with qualitative categorical agent-based model outputs.

1. Introduction

Agent-based modeling has emerged as a way of modeling and analyzing complex adaptive systems and has been efficiently used in a broad range of domains such as economics [28,74], ecology [32], sociology [53,73], archeology [5,35], and military [17]. The main motivations behind using agent-based modeling are mainly to understand the dynamics of the system and to observe how the system behaves in the presence of interventions (e.g., new policies, parameter changes) [81]. In this respect, discovering the relationship between inputs and outputs of agent-based models is the ultimate way of providing insights into understanding the dynamics of the system being modeled.

The most naive way to uncover the regularities between inputs and outputs of agent-based models is manual model exploration, where the exploration process is guided by an analyst or an expert. Although this kind of exploration process is the most straightforward one and does not require advanced technical background, it is very likely to be biased and may leave some “interesting” parts of the input space unexplored [48]. A more systematic way of discovering input–output relationship is to employ well-established

* Corresponding author at: Department of Industrial Engineering, Bogazici University, Bebek, Istanbul 34342, Turkey.
E-mail address: medali@yildiz.edu.tr (M. Edali).

https://doi.org/10.1016/j.simpat.2018.12.006
Received 10 October 2018; Received in revised form 14 December 2018; Accepted 29 December 2018
Available online 31 December 2018
1569-190X/

sampling techniques from the design of experiment (DoE) literature such as random sampling, full and fractional factorial designs, and central composite design. Once the input parameter combinations are obtained by utilizing one of these sampling techniques, they are fed to the simulation model and outputs are generated. Then, the resulting input–output data is analyzed with the help of a statistical tool such as first- and second-order linear regression and ANOVA. However, the credibility of the results obtained from these tools depends on the validity of the assumptions such as (i) low order interactions among model input parameters, (ii) low order relationship between input parameters and output, and (iii) normally-distributed residuals [48,67]. These assumptions are mostly violated in the case of agent-based models since they exhibit highly nonlinear emergent behaviors such as tipping points, path dependencies, and regime shifts. Therefore, the analyst must carefully implement these techniques and should check whether the assumptions are violated or not [14,48]. Alternatively, the machine learning literature presents a broad set of tools, which can address the limitations of both manual and DoE-based techniques, to support model exploration in a systematic way. Among these tools, supervised learning algorithms are capable of building a nonlinear mapping from simulation model inputs to the outputs by learning a functional relationship from the input–output data obtained from the simulation model [4,26]. The learned (potentially nonlinear) function can be considered as an approximate representation of the input–output relationship of the original simulation model, and is also known as a metamodel, a surrogate model, an emulator, or a response surface [41,43]. The main motivations behind the use of metamodels can be summarized as follows [39,44]:

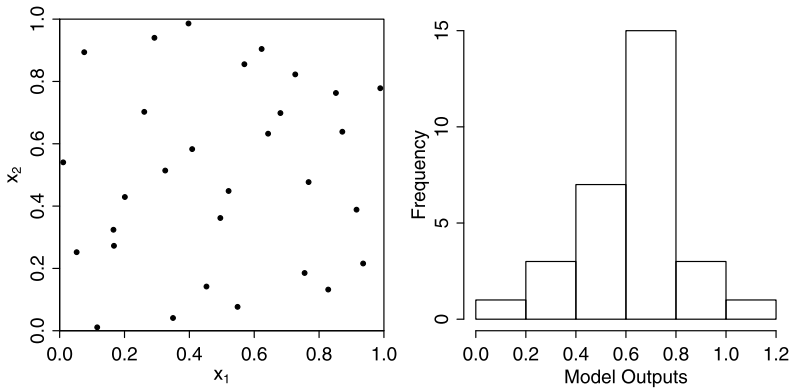
- *Understanding*: An interpretable metamodeling technique may further provide insights into the dynamic problem by improving the understanding of input–output relationships of a simulation model. For example, estimated coefficients of input parameters in a valid linear regression metamodel can reveal the direction of the effect of a simulation input parameter on the output.
- *Prediction*: If the metamodel is proven to be accurate, it can be used to generate output predictions for a given set of input parameter combinations. This feature of metamodeling helps the analyst to save time especially if the runtime of the original simulation model is significantly high.
- *Optimization*: By combining optimization techniques (exact or heuristic methods) and metamodeling, it is possible to find input parameter combinations which maximize or minimize a certain feature of the output (e.g., fit to historical data, terminal value) of the original simulation model (e.g., Barton and Meckesheimer [7]).
- *Validation and verification*: Metamodels also serve as a tool for validating and verifying simulation models. Since a metamodel mimics the input–output relationship of the simulation model, we expect that all the interactions and relationships captured by the metamodel should be consistent with what is known about the real system. However, any contradiction between the metamodel and the real system serves as a warning signal, but is not a conclusive result about the invalidity of the original simulation model since a metamodel may miss some inner-dynamics of the simulation model due to its simplified nature.

In this study, we primarily focus on two of the aforementioned aspects of metamodeling; *understanding the system* and *output prediction*. One central issue that conditions the benefit from a metamodeling practice is the choice of approach with the appropriate output type. Most frequently used metamodeling approaches produce continuous output. Such approaches may be appropriate when the distribution of the model output being analyzed is unimodal. However, an agent-based model's output of interest may exhibit a multimodal distribution and/or a categorical nature, especially when there is a tipping point in its behavior space. In such cases, a metamodel that generates continuous estimates is very likely to perform poorly as a representation of the original model. In such cases, a metamodeling approach with categorical output may capture the discrete jumps in the simulation model's output much better. Therefore, an analyst requires apriori information about whether the nature of a simulation model's output distribution is unimodal, or multimodal in order to get the maximum benefit from a metamodeling exercise. In the latter case, if the analyst opts for using a categorical metamodel, setting appropriate thresholds for specifying categories is another challenge. We consider the absence of a formal procedure both for assessing the multimodality of the output distribution and defining appropriate thresholds for categorization as an important gap, and propose a novel two-step procedure which can assist the model analysts (Section 2).

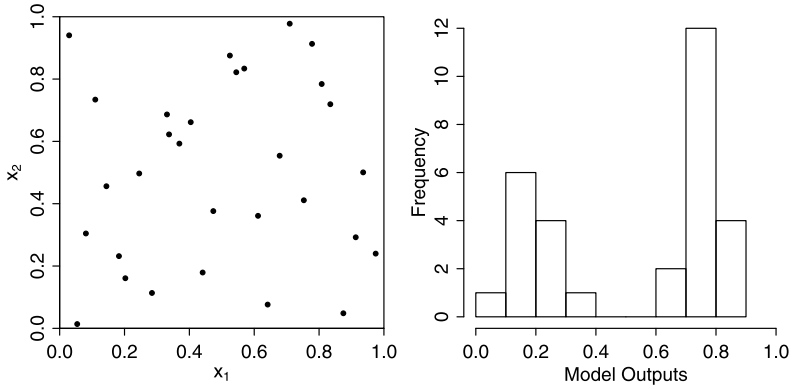
After selecting the metamodel output type (i.e., continuous or categorical), the analyst needs a systematic and automated sampling technique which allows to construct a metamodel accurately capturing the input–output dynamics of an agent-based model. A convenient way of collecting input (and the resulting simulation model output) data is to employ sampling techniques such as Latin hypercube sampling [58], which generate a predefined number of input parameter combinations with certain properties. These type of sampling techniques are also called "one-shot" designs since they do not consider adding additional samples to the training set after fitting the metamodel. These approaches may require a large number of sample points to obtain a metamodel with the desired accuracy. Considering this limitation, the analyst can alternatively employ sequential sampling techniques, which iteratively expands a comparatively small one-shot sample by processing the information gained by both the previously collected sample points and the metamodel [20,25]. Compared to one-shot techniques, sequential sampling techniques yield metamodels with the desired accuracy with a smaller sample size. Besides, sequential sampling process may point out input parameter combinations yielding counter-intuitive simulation model behavior, and boundary points between different regions of the model's behavior space since they focus the sampling process towards the challenging regions (i.e., regions with input parameter combinations whose outputs are hard to predict) in the behavior space. In that respect, we propose a metamodel-assisted sequential sampling technique for agent-based models in this study, which is shown to have advantages over one-shot designs regarding both precision and efficiency (Section 3).

In summary, the purpose of this paper is threefold: (i) to offer a novel and well-grounded approach for categorizing the outputs of agent-based models, (ii) to develop a metamodel-guided sequential sampling technique, and (iii) to discuss the potential benefits of metamodel-based sequential sampling technique for the model analysts by applying it on two well-known agent-based models.

The remainder of this article is organized as follows: Sections 2 and 3 summarize the technical background of output



(a) Latin hypercube sampling from the input space (left) and unimodal distribution of the model outputs (right).



(b) Latin hypercube sampling from the input space (left) and bimodal distribution of the model outputs (right).

Fig. 1. Two hypothetical examples for unimodal and bimodal distribution of simulation model outputs.

categorization and sequential sampling technique, respectively. Section 4 presents the proposed procedure, the experimental design, and the agent-based models selected for the experimentation. Section 5 gives results of the experiments and provides observations based on the results. Section 6 discusses the potential extensions of the proposed approach. Finally, Section 7 concludes the study.

2. Multimodality detection and output categorization

In order to elaborate on the distinction of unimodality and multimodality of behavior, we present two hypothetical output distributions in Fig. 1. In the first case (Fig. 1a), the output is observed to be centered around a single output value, and can be claimed to have a continuous nature. In the second case (Fig. 1b), the system seems to have two different modes; i.e., in the first mode model output is centered around 0.2, and in the second mode the output is relatively higher. More importantly, the output does not seem to vary continuously between the low-valued region and the high-valued region.

In the first case, a continuous-valued metamodel may "predict" the model output as a function of the input parameters relatively easily. However, the bimodal nature and the discrete jump from low-valued region to the high-valued region in the second has potential to be troublesome for such a metamodel. In that case, it may be more important to predict whether the model will generate an output in the "low" or "high" modes. In this way, the problem is transformed into a classification problem rather than a regression problem, and one needs to specify the thresholds for labeling a model output as "low", or "high".

Our multimodality detection and output categorization procedure involves two main steps: First, we use Silverman's test [72] to detect multimodality in the output distribution. If multimodality is detected (for example, Fig. 1b) by the test procedure, we categorize the outputs by utilizing Mean Shift algorithm [16,31]. The resulting clusters are considered as classes (categorical model outputs), and the metamodel is trained with a dataset having input parameter combinations and corresponding class labels (e.g., "low" and "high"). However, if the output distribution is not multimodal, the original (numerical) simulation model outputs are

preserved, and the metamodel is trained with the original dataset having input parameter combinations and corresponding numerical model outputs.

2.1. First step: Silverman's test

Silverman [72] proposes a hypothesis testing procedure based on kernel density estimation to explore the number of modes (peaks) of a density function. Suppose that the probability density function of the simulation model outputs $y_1, y_2, \ldots, y_n$ is $f(\cdot)$. The kernel density estimate $\hat{f}(\cdot)$ of $f(\cdot)$ is calculated as follows

$$\hat{f}(y) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{y - y_i}{h}\right), \tag{1}$$

where $K(\cdot)$ is the kernel function with bandwidth h [4]. Here, h controls the smoothness of the estimated distribution, and thus, the number of modes of the density; as h increases, the number of the modes of $\hat{f}(y)$ decreases, and vice versa. Silverman [72] utilizes Gaussian kernel in his study Eq. (2):

$$K(u) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{u^2}{2}\right) \tag{2}$$

The hull hypothesis H_0 of the Silverman's test states that the function $f(\cdot)$ has at most j modes and the alternative hypothesis H_1 states that $f(\cdot)$ has more than j modes, where k is the actual number of modes of $f(\cdot)$ [29]:

H_0: $k \leq j$

H_1: $k \geq j + 1$

where $j = 1, 2, \cdots$.

Silverman [72] proves that the number of modes of $\hat{f}(\cdot)$ is a decreasing function of h. In other words, as h increases, the number of modes of $\hat{f}(\cdot)$ decreases. Therefore, it is possible to define $h_{crit,\ j}$ indicating the critical bandwidth value h beyond which $\hat{f}(\cdot)$ has at most j modes: If $h > h_{crit,\ j}$, $\hat{f}(\cdot)$ has less than or equal to j modes, and if $h < h_{crit,\ j}$, $\hat{f}(\cdot)$ has greater than or equal to $j + 1$ modes [29].

To calculate the p-value, Silverman [72] first generates a smoothed bootstrap sample $y_1^*, y_2^*, \ldots, y_n^*$ from $y_1, y_2, \ldots, y_n$. Then, based on the smoothed bootstrap sample, he calculates a new kernel density estimate $\hat{f}^*(\cdot)$ by using $h_{crit,\ j}$. This procedure is repeated m times, and the proportion of the cases where $\hat{f}^*(\cdot)$ has at least $j + 1$ modes is considered as the p-value. In this respect, a smaller p-value indicates that most of the m repetitions yields $\hat{f}^*(\cdot)$ with at most j modes, which also shows that $h_{crit,\ j}$ is significant. This indicates that $h_{crit,\ j}$ should be high in order to force the density function to have j modes, and this points out that $\hat{f}(\cdot)$ has more than j modes. If p-value is less than a predetermined significance level α, the null hypothesis H_0 is rejected.

To determine the number of modes of the simulation model outputs, we conduct Silverman's test for successive values of j starting from $j = 1$ until H_0 is failed to be rejected. Then, j is the number of the modes of $f(\cdot)$. However, for high values of j (especially if $j \geq 5$), we suggest visually checking the distribution of the outputs.

After finding that the number of modes is greater than or equal to 2, the analyst can proceed to the second step. If not, the analyst can skip the second step and directly proceed to metamodel fitting with the dataset having input parameter combinations and the resulting numerical model outputs.

2.2. Second step: Mean shift algorithm

Mean Shift algorithm is a mode-seeking and clustering approach which utilizes kernel density estimate $\hat{f}(\cdot)$ of a dataset. The algorithm locates the modes (i.e., peaks) of a kernel density estimate by iteratively shifting each data point to one of the modes. Then, the data points converging to the same peak of the density function are grouped into the same cluster [19,31,60].

Mean shift term, which is given in Eq. (3), is derived from the gradient estimate of $\hat{f}(\cdot)$ and its direction is towards the direction of the maximum increase [19]:

$$m(y) = \frac{\sum_{i=1}^{n} K\left(\frac{y - y_i}{h}\right) y_i}{\sum_{i=1}^{n} K\left(\frac{y - y_i}{h}\right)} - y \tag{3}$$

Algorithm 1 presents the Mean Shift algorithm, which has four main inputs: a kernel function K, a kernel bandwidth h, a dataset $Y = \{y_1, y_2, \ldots, y_n\}$, a threshold ε to check the convergence. The algorithm shifts each data point y_i to the weighted average of its neighbors until mean shift term, $m(y_i)$, is less than ε. The algorithm returns the cluster centers (i.e., peaks of the density function) and the members (i.e., data points) of each cluster c_i, $i = 1, \cdots, j$ by putting data points converging to the same peak to the same cluster [60]. It has proven in the literature that each data point y_i converges to the nearest mode of the kernel density estimate $\hat{f}(\cdot)$ [19].

Mean Shift algorithm requires the selection of a kernel function K and a kernel bandwidth h. We use Gaussian kernel since the first step (Silverman's test) also uses the same kernel function. As the bandwidth value h of a density function with j modes, we use the

Input: K, h, Y, ε
Output: $C = \{c_i : i = 1, \ldots, j\}$ /* j is the number of modes */
1: **for** $i = 1$ **to** n **do**
2: **while** $m(y_i) > \varepsilon$ **do**
3: $y_i \leftarrow m(y_i) + y_i$
4: **end while**
5: **end for**
6: Determine the cluster centers and the members of each cluster $c_i, i = 1, \ldots, j$.

Algorithm 1. Mean Shift.

average of $h_{crit,\ j}$ and $h_{crit,j-1}$, which is equidistant from both of the critical values, since $h_{crit,\ j}$ and $h_{crit,j-1}$ are both transition points between modes $j \leftrightarrow j + 1$ and $j - 1 \leftrightarrow j$, respectively.

3. Sequential sampling

One of the most extensively used sampling techniques for metamodel fitting is Latin hypercube sampling [58] due to its space-filling and projection properties. These properties allow to evenly sample from the input space and generate representative input parameter combinations from different subspaces of the input space. Projection property is achieved by dividing each input parameter into n equally-probable distinct intervals and then samples are randomly drawn so that there is only one sample in each interval for each input parameter [40]. However, it is not guaranteed that every Latin hypercube sample has a good space-filling property (see, for example, Crombecq et al. [21] and Sheikholeslami and Razavi [71]). Therefore, there are many attempts in the literature to improve the space-filling property of Latin hypercube samples. These improvements are mostly achieved by (i) maximizing the minimum distance among the sample points (i.e., input parameter combinations) or (ii) minimizing the pairwise correlations between input parameters (for a brief survey of these studies, please see Sheikholeslami and Razavi [71]). However, although they are improved in the sense of space-filling, these sampling designs are called “one-shot” since they are generated with a fixed number of input parameter combinations prior to metamodel training [21,25,71]. In practice, there is no established way of determining the sample size prior to metamodel fitting to obtain a desired accuracy. Besides, aiming to generate a sample with space-filling property may be an expensive effort for large regions of homogeneous behavior. Therefore, as an alternative to “one-shot” designs, sequential sampling techniques (also known as adaptive sampling and active learning) are extensively used in the literature [25,47,51,82]. Contrary to beginning the training process with a large “one-shot” sample, sequential sampling strategies iteratively expand the sample set size and yield metamodels of higher accuracy with comparatively smaller training sets by utilizing the knowledge obtained in the earlier sampling and metamodel training iterations.

Sequential sampling techniques can be input- or output-oriented. In the input-oriented sequential sampling, outputs obtained from the simulation model (or from the metamodel) are ignored, and the sampling process focuses on the space-filling and projection properties of the input samples. However, output-oriented techniques consider the insights provided by the simulation model or the metamodel for the selection of the new samples [71]. While both of the approaches aim to increase the accuracy of the metamodel in an iterative manner, output-oriented sequential sampling also gives valuable insights about both the simulation model and the system since most of these strategies consider output prediction uncertainty for selecting/generating new input parameter combinations [25]. Naturally, these newly selected/generated input parameter combinations mostly point out the boundaries between different types of model behavior (numerically or categorically) since metamodels generally have difficulty in predicting the outputs of input parameter combinations located around these boundaries.

In supervised learning, a machine learning model (in our case, a metamodel) is trained on a preferably large labeled dataset (i.e., a dataset that includes a set of features and resulting outputs) to predict the outputs of unlabeled data (i.e., a dataset that only includes a set of features). However, in some problem domains such as image classification and speech recognition, it is hard to obtain labeled data. In other words, instances must be labeled by an oracle (e.g., a human annotator) and this turns out to be a potentially time-consuming and expensive process [70]. Similarly, in the simulation domain, evaluating an input parameter combination on the simulation model can also be computationally costly.

Contrary to passive learning where the machine learning model trained on a static labeled data as mentioned above, the machine learning model can choose the training data to be collected further in active learning (i.e., sequential sampling): First, the model is trained on a limited number of labeled instances. Then, the model selects unlabeled data to be labeled by an oracle (mostly a human annotator), and the labeled data is added to the training set. This process iteratively expands the training set yielding more accurate prediction models compared to models trained on a randomly selected data with equal size. In addition, active learning also yields a prediction model with a similar accuracy that of the prediction model trained with passive learning by using a comparatively smaller training set [52,70]. In simulation metamodeling, the metamodel is trained on a dataset consisting of input parameter combinations and their respective outputs (i.e., labeled data). Then, the metamodel selects unlabeled input parameter combinations where the metamodel is highly unsure about their outputs. These parameter combinations are then evaluated on the simulation model and then added to the training set of the metamodel.

There are two main issues in an active learning approach: (i) how does it ask queries for labeling? and (ii) how does it select an unlabeled data for labeling? The first question is related to the selection of unlabeled data generation strategy, and in this context, we

employ *pool-based sampling* which assumes that there is a large set of unlabeled data while the amount of labeled data is comparatively small. This assumption holds for our case since we also have a small training set and the cost of generating a pool of unlabeled data is very small. The second question refers to the selection of the most informative unlabeled data which yields the highest increase in accuracy of the metamodel if it is added to the training set with its label. For unlabeled data selection, we employ *uncertainty sampling*[1] [49]. In this setting, the machine learning model selects the unlabeled data whose label is the hardest to predict. To quantify the level of uncertainty, we use *margin sampling* for metamodels with categorical model outputs, where we select an unlabeled instance $\boldsymbol{x}^*$ such that

$$\boldsymbol{x}^* = argmin_x(Pr\{\hat{y} = c_1|\boldsymbol{x}\} - Pr\{\hat{y} = c_2|\boldsymbol{x}\}) \tag{4}$$

where c_1 and c_2 are assumed to be the first and the second most probable class labels of $\boldsymbol{x}^*$ and $\boldsymbol{x}$ is an instance from the pool of unlabeled data. In other words, the metamodel fails to make a clear distinction between the two most probable categories of the simulation input parameter combination $\boldsymbol{x}^*$, and it is assumed that the best improvement in the predictive performance of the metamodel will be achieved if $\boldsymbol{x}^*$ is added to the training set with its true label. For metamodels with continuous outputs, the uncertainty measure is the variability (e.g., variance) of the predictions.

4. Proposed procedure and experimental design

4.1. Metamodeling approach

Metamodel training basically involves constructing a mapping from simulation model inputs to outputs. Therefore, many supervised learning techniques can be effectively used to construct simulation metamodels. These techniques include artificial neural networks [3,15,36,46], kriging [41,66], support vector regression [18,26,83], random forests [76], multivariate adaptive regression splines [11,30], radial basis functions [37,59], first- and second-order linear regression models [24,42,54]. There are several studies in the literature which compare different subsets of these techniques based on different criteria such as accuracy, robustness, transparency (i.e., interpretability), and efficiency (i.e., runtime) [18,50,75,76]. Among these studies, Clarke et al. [18], Li et al. [50], and Villa-Vialaneix et al. [76] conclude that support vector regression yields the most accurate metamodels compared to other techniques. Besides, based on their empirical comparison, Villa-Vialaneix et al. [76] also conclude that accuracy of random forest is competitive compared to that of support vector regression.

The first candidate technique for metamodeling is support vector regression due to its high accuracy in representing nonlinear systems. Support vector regression technique utilizes kernel functions to approximate highly nonlinear systems, which generates accurate metamodels but decreases the interpretability of resulting prediction functions. Therefore, support vector machines is considered as a "black-box" technique [6]. In other words, the resulting prediction function is not directly interpretable (i.e., does not provide insights into the input–output relationship) although it is very accurate in predicting unlabeled instances [26]. Therefore, there are many techniques offered in the literature to increase the transparency of support vector machine models by extracting comprehensible rules (e.g., IF-THEN rules) from the prediction function (for a detailed survey, please see Barakat and Bradley [6]). However, rule extraction techniques add another abstraction layer between the real system and its representation (i.e., metamodel), which may result in loss of information.

The second potential candidate is random forest technique to capture input–output relationships of agent-based models. A random forest (RF) is an ensemble of J individual decision trees $\{g_j, j = 1, 2, \cdots, J\}$ each of which is trained on a dataset obtained by bootstrapping. Each decision tree, g_j, is fully grown by using CART algorithm [13]. Besides being trained on different bootstrap training sets, only a random subset of features is selected as candidate features for splitting at each node. These properties of random forests reduce the variance and make them competitive in classification and regression [4,12]. In classification, a class prediction is returned from each tree g_j and the most frequent class is assigned as the label of the (unlabeled) test instance (i.e., majority voting). In regression, the mean of the outputs obtained from each tree is returned as the point estimate.

Compared to support vector regression, random forest technique generates more interpretable metamodels. Since a random forest (RF) is an ensemble of individual decision trees all of which can be represented as a set of IF-THEN rules, a rule selection algorithm, which directly captures the most accurate and comprehensive set of rules, can be utilized without adding any additional abstraction level. These techniques also consider rule length [57,62] or propose rule pruning [22] to increase the comprehensibility of selected rules in case of larger trees (i.e., trees with high number of leaf nodes). In other words, interpretability of a random forest metamodel can be maintained despite its complexity. However, rule extraction from RFs is beyond the scope of this study. Interested readers are referred to Deng [22], Mashayekhi and Gras [56], [57], Phung et al. [62], and Adnan and Islam [1].

Another advantage of random forest over support vector machines is that the prediction model provides class probability estimates for categorical model outputs, which enables us to employ *uncertainty sampling*. In an RF with J individual decision trees $\{g_j, j = 1, 2, \cdots, J\}$, class probability estimates for an unlabeled sample $\boldsymbol{x}$ (in our case, a simulation input parameter combination) can be obtained by (for class c, $c = 1, 2, ..., C$)

$$Pr\{\hat{y} = c|\boldsymbol{x}\} = \frac{1}{J}\sum_{g_j \in J} I(\hat{y}_j(\boldsymbol{x}) = c) \tag{5}$$

[1] For a survey of alternative unlabeled data selection techniques, please see Settles [70].

where $\hat{y}_j(\boldsymbol{x})$ is the class prediction returned from tree g_j for unlabeled instance $\boldsymbol{x}$. $I(\cdot)$ is an indicator function which returns 1 if its input argument is true and returns 0 otherwise [8]. These class probability estimates provide useful information to the analyst by giving an intuition about the outputs of input parameter combinations without running the original simulation model. Although obtaining probability estimates is not possible for continuous outputs, predictions returned by each tree can be summarized by using some data dispersion measures such as variance and standard deviation. Probability estimates (categorical output case) and dispersion measures (continuous output case) also form a basis for sequential sampling procedure. For example, in the context of RF, the uncertainty measure for continuous output case is the standard deviation of the predictions returned from each tree g_j [55]

$$\sqrt{\frac{\sum_{g_j \in J} \left(\hat{y}(\boldsymbol{x}) - \hat{y}_j(\boldsymbol{x})\right)^2}{J-1}} \tag{6}$$

where,

$$\hat{y}(\boldsymbol{x}) = \frac{1}{J} \sum_{g_j \in J} \hat{y}_j(\boldsymbol{x}) \tag{7}$$

For the categorical output case, *uncertainty sampling* (*margin sampling*, in specific) helps the trees of the RF to correctly locate the axis-parallel splits by selecting the input parameter combinations where the disagreement in the prediction is the highest among the trees. Therefore, we naturally expect that the selected input parameter combinations by the *margin sampling* will lie on the boundaries connecting different categorical agent-based model outputs. Similar arguments are also valid for the continuous output case, where the disagreement among the trees is measured by the standard deviation of the estimates obtained from each tree of the RF (Eq. 6).

Due to the abovementioned advantages over support vector machines, we use random forest metamodeling technique in this study. Our proposed procedure, which utilizes RFs as the metamodeling technique, is depicted in Fig. 2. We first generate a set of n unlabeled instances by utilizing maximin Latin hypercube sampling [9,23,38]. Then, we feed these instances to the simulation model and obtain the corresponding outputs by replicating several times and averaging them. In the third step, we run Silverman's test to detect whether the simulation model outputs exhibit multimodal distribution. If the distribution is not multimodal, we solve a regression problem (i.e., construct the metamodel with continuous outputs) and utilize sequential sampling by considering the variability of the outputs returned by each tree of the RF (Eq. 6). If the distribution is multimodal, we first run Mean Shift algorithm (Algorithm 1) and determine the class labels of each instance. Finally, we solve a classification problem (i.e., construct the metamodel with categorical outputs) and utilize sequential sampling by *margin sampling* technique (Eq. 4).

4.2. Experimental design

We conduct a comparative analysis of metamodel-guided sequential sampling and random sampling in the case of metamodel training. For this purpose, we generate different initial training sets each having N instances. After generating initial training sets, we utilize our two-step procedure for the outputs of each training set. If the distribution of the outputs turns out to be unimodal, we do not run Mean Shift algorithm and preserve the numerical model outputs. If not, we run Mean Shift algorithm to determine the class labels of each instance in the training set. Then, we run the sequential sampling procedure (Algorithm 2) for each training set independently. Basically, Algorithm 2 expands the initial training set (L) of metamodel (M) in a predefined number of iterations (I) by selecting h unlabeled instances from unlabeled data pool (U) at each iteration. The selected unlabeled instances are then evaluated on the simulation model (S). If the outputs of the metamodel (M) are numerical, they are directly added to the training set (L) with their corresponding input parameter combinations. If not, we run Mean Shift algorithm to determine the class labels and add them to the training set (L). At termination, the procedure returns two outputs: L^* and M^*, which are the final (expanded) labeled training set and the final metamodel, respectively. Note that L is expanded throughout iterations and M is retrained at the end of each iteration. Besides, accuracy of M on test set (T) is reported at the end of each iteration. If there is a randomness in the training procedure, the sequential sampling procedure is replicated by R times.

In our experiments, U includes 500 unlabeled instances. Initially, each L includes 30 labeled instances (i.e., $N = 30$), and the number of instances in the test set (T) is 500. Since number of iterations (I) is selected as 14, and batch size (h) is selected as 5, the final training set (L^*) includes 100 labeled instances at termination. Due to the randomness of RF metamodel training procedure, we replicate each sequential sampling procedure 10 times (i.e., $R = 10$) for each training set and report the accuracy at each iteration j averaged over these 10 replications.

In order to show the benefits of sequential sampling technique in terms of metamodel accuracy, we also perform a random sampling procedure, which is given in Algorithm 3. The only procedural difference between random and sequential sampling is that, in the random sampling procedure, h number of random samples are added to the training set (L) instead of selecting the most uncertain samples.

For illustrative purposes, we select two widely available models from NetLogo[2] models library, which are Segregation and Traffic Basic. For both models, we follow similar experimental procedures. We first generate 10 different initial training sets, each of which include 30 parameter combinations (i.e., $N = 30$) sampled using maximin Latin hypercube sampling. We evaluate each parameter combination on the simulation model by replicating 30 times, and then we take the average of these 30 replications as the output.

[2] NetLogo is an agent-based modeling and programming environment [79].

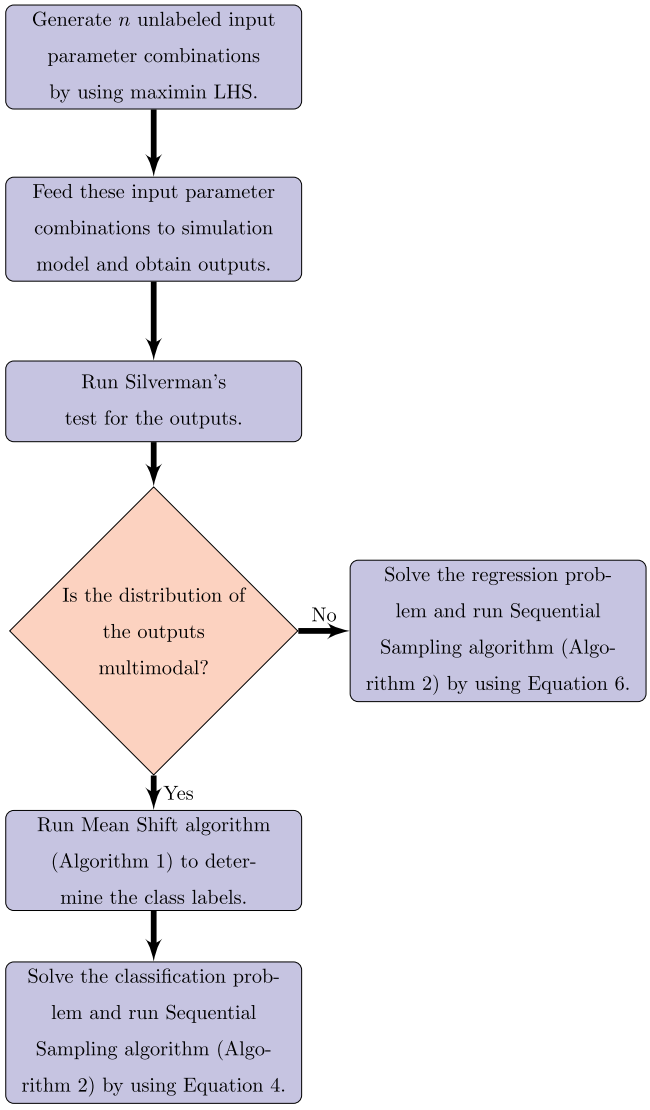


Fig. 2. Flowchart of the proposed approach.

4.2.1. First model: Segregation

We use Segregation model [68] implemented in NetLogo [77] as the first experimental case. The model assumes that people from two different groups (for example, from two different ethnicities or political views) live in a "city" represented by a two-dimensional space. Each individual wants that at least a certain proportion of his neighbors to be from his own group. Otherwise, they consider themselves unhappy and seek to move to an unoccupied space where they can be potentially happy to live in. The model shows how this simple choice mechanism of residents can result in a segregated society. To do so, the model incorporates two different parameters: density of the city (*density*) and the minimum fraction of the neighbors that are wanted to be alike (*%-similar-wanted*). Fig. 3 shows three examples of the spatial model output for a fixed *density* parameter (75%) with three different *%-similar-wanted* values. We observe that as *%-similar-wanted* increases, society becomes more segregated. We are interested in *%-similar* as the model output, which is a measure of the severity of the segregation. For example, in Fig. 3a, *%-similar* is 60.4%. In Fig. 3b, *%-similar* is 87.1%. As expected, society is extremely segregated in Fig. 3c; *%-similar* is 99.8%.

4.2.2. Second model: Traffic basic

Traffic jam formation without any physical blockage on a road, namely *phantom jam*, has been extensively studied in the literature, and it is shown that it emerges as a result of the interactions among the agents, drivers of cars [80]. Agent-based models by Resnick [64] and Wilensky [78] allow to experiment to observe how traffic congestions occur without any physical obstruction on a road [65].

Our second experimental model is Traffic Basic, which is a ready-to-use model that can be found in NetLogo models library [79].

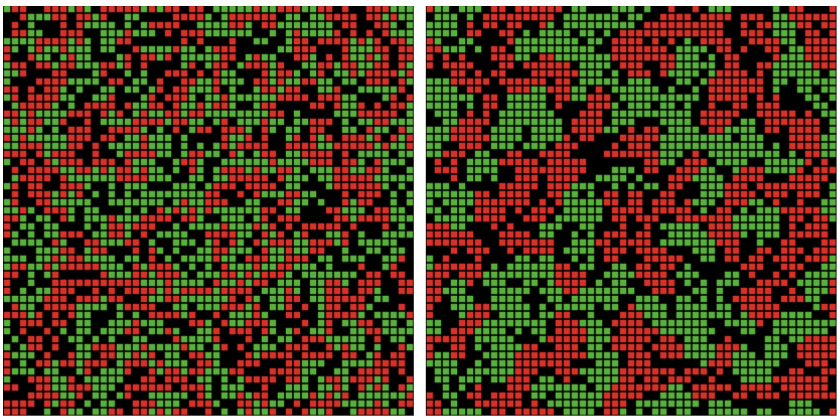
Input: U, L, T, h, M, I, S, R
Output: L^*, M^*

for $i = 1$ to R **do**
  for $j = 1$ to I **do**
    Obtain the uncertainty value using Equation 4 (for categorical output case) or Equation 6 (for continuous output case) for all instances in U using M.
    Select h instances with the highest uncertainty. Name this selected set as $U^{'}$.
    $U \leftarrow U \setminus U^{'}$
    Obtain the outputs of $U^{'}$ with S.
    $L \leftarrow L \cup U^{'}$
    Retrain M with L.
    Report accuracy of M on T.
  end for
end for
$L^* \leftarrow L$
$M^* \leftarrow M$

Algorithm 2. Sequential Sampling Procedure.

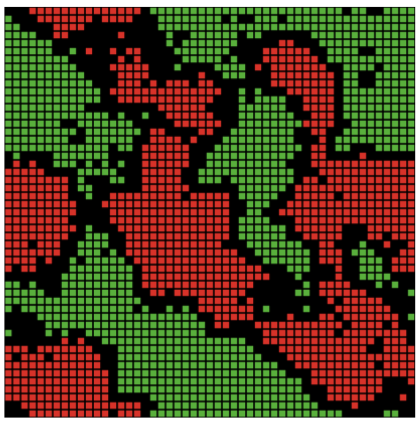
Input: L, h, T, M, I, S, R
Output: L^*, M^*
for $i = 1$ to R **do**
  for $j = 1$ to I **do**
    Generate h random instances for labeling. Name this set as U'.
    Obtain the outputs of U' with S.
    $L \leftarrow L \cup U'$
    Retrain M with L.
    Report accuracy of M on T.
  end for
end for
$L^* \leftarrow L$
$M^* \leftarrow M$

Algorithm 3. Random Sampling Procedure.



(a) *%-similar-wanted* = 25% and *%-similar* = 60.4%

(b) *%-similar-wanted* = 50% and *%-similar* = 87.1%



(c) *%-similar-wanted* = 75% and *%-similar* = 99.8%

Fig. 3. Spatial outputs of Segregation model for different values of *%-similar-wanted*.

This model simulates the flow of cars moving on a single-lane road by utilizing three different parameters: (i) *number of cars*, (ii) *deceleration*, and (iii) *acceleration*. Initially, each car is randomly located on the road, and a speed value between 0.1 and 1.0 is randomly assigned to each car. For each car, the maximum possible speed is 1.0 and the minimum possible speed is 0. At each iteration, each car decelerates when they detect a car being ahead and accelerates otherwise. In order to observe how the model parameters influence the traffic flow, one of the cars is randomly selected, and its speed is monitored during a simulation run. As a representation of the traffic flow, we select the model output as the average speed of the randomly selected car over 1000 iterations.

5. Results

5.1. Segregation model

Segregation model serves as an appropriate case where the simulation model outputs do not exhibit a multimodal distribution. Outputs of all of the 10 different training sets each involving 30 model instances (i.e., input parameter combinations) exhibit a unimodal distribution at $\alpha = 0.05$ level. p-values for the test H_0: $k \leq 1$ againts H_1: $k \geq 2$ are presented in Table 1. Therefore, we keep the original continuous outputs and solve a regression problem for all of the training sets. However, we note that lower p-values given in Table 1 also indicate a potential multimodality in the outputs of Segregation model (e.g., Training Set 3 and 10). Since the results of Silverman's test depend on the initial training set, these variations in p-values are not surprising if the simulation model outputs demonstrate a "weak" multimodality as in the case of Segregation model. If p-value is close to significance level α, the analyst can use classification or regression approach depending on the context of the analysis.

Fig. 4 shows the Root Mean Square Error (RMSE) values for both sequential sampling (active learning) and random sampling throughout iterations, and Table 2 shows initial and terminal RMSE values of RF metamodels trained with random and sequential sampling. We observe that sequential sampling technique is superior to random sampling in terms of RMSE values for all of the initial training sets. In other words, sequential sampling technique, which adds instances where the metamodel is highly uncertain about their outputs, is a superior to random sampling for metamodel training. Another important advantage of sequential sampling is that the reduction in RMSE values is independent from the initial RMSE value; terminal RMSE values are around 2.95 although initial RMSE values can be high, which also shows the robustness of sequential sampling. However, in random sampling side, the gain obtained depends on the initial RMSE value; if initial RMSE is high, terminal RMSE is also high.

Numerical results clearly show that sequential sampling technique generates more accurate metamodels at termination. Besides, we observe that metamodels trained with sequential sampling reach the terminal accuracy of metamodels trained with random sampling at the very early iterations. For example, for Training Set 1, the terminal RMSE value for the metamodel trained with randomly selected data is 4.9470. We observe that the metamodel trained with sequential sampling procedure reaches the RMSE value of 4.7240 at the sixth iteration. In other words, a metamodel trained with a dataset having 60 instances, half of which is selected by sequential sampling, performs better than a metamodel trained with a dataset having randomly selected 100 instances. In addition, sequential sampling also contributes to enhance the understanding of the original simulation model. For example, in Fig. 5, we observe that the input parameter combinations selected by the sequential sampling procedure concentrated on specific areas of the input space, especially around *%-similar-wanted* = 80%. This indicates that the RF metamodel needs a "high-resolution picture" of these input subspaces to accurately learn the input–output relationship of the simulation model. A careful inspection of these subspaces points out the input parameter combinations where the simulation model dramatically changes its behavior.

When we inspect the added points through sequential sampling in Fig. 5, we see that Segregation model exhibits a tipping point behavior around *%-similar-wanted* = 80% for *density* ≥ 20%. This observation is in line with that of presented by Railsback and Grimm [63], which manually detects the tipping point around *%-similar-wanted* = 75%. If 40% ≤ *%-similar-wanted* ≤ 80%, the society is extremely segregated for all *density* values (i.e., *density* ∈ [10, 90]). For *%-similar-wanted* ≥ 80%, the segregation in the society disappears since all of the residents are always "unhappy" and constantly move from one place to another. Newly added points also show that even a smaller *%-similar-wanted* value can result in a segregated society if *density* is low since residents freely move to low-density areas of the city where they are isolated from the residents of the other group.

Table 1
p-values for all training sets of Segregation model (H_0: $k \leq 1$ against H_1: $k \geq 2$) for $m = 1000$ repetitions.

Training set	p-value
1	0.819
2	0.123
3	0.052
4	0.097
5	0.318
6	0.378
7	0.136
8	0.160
9	0.089
10	0.070

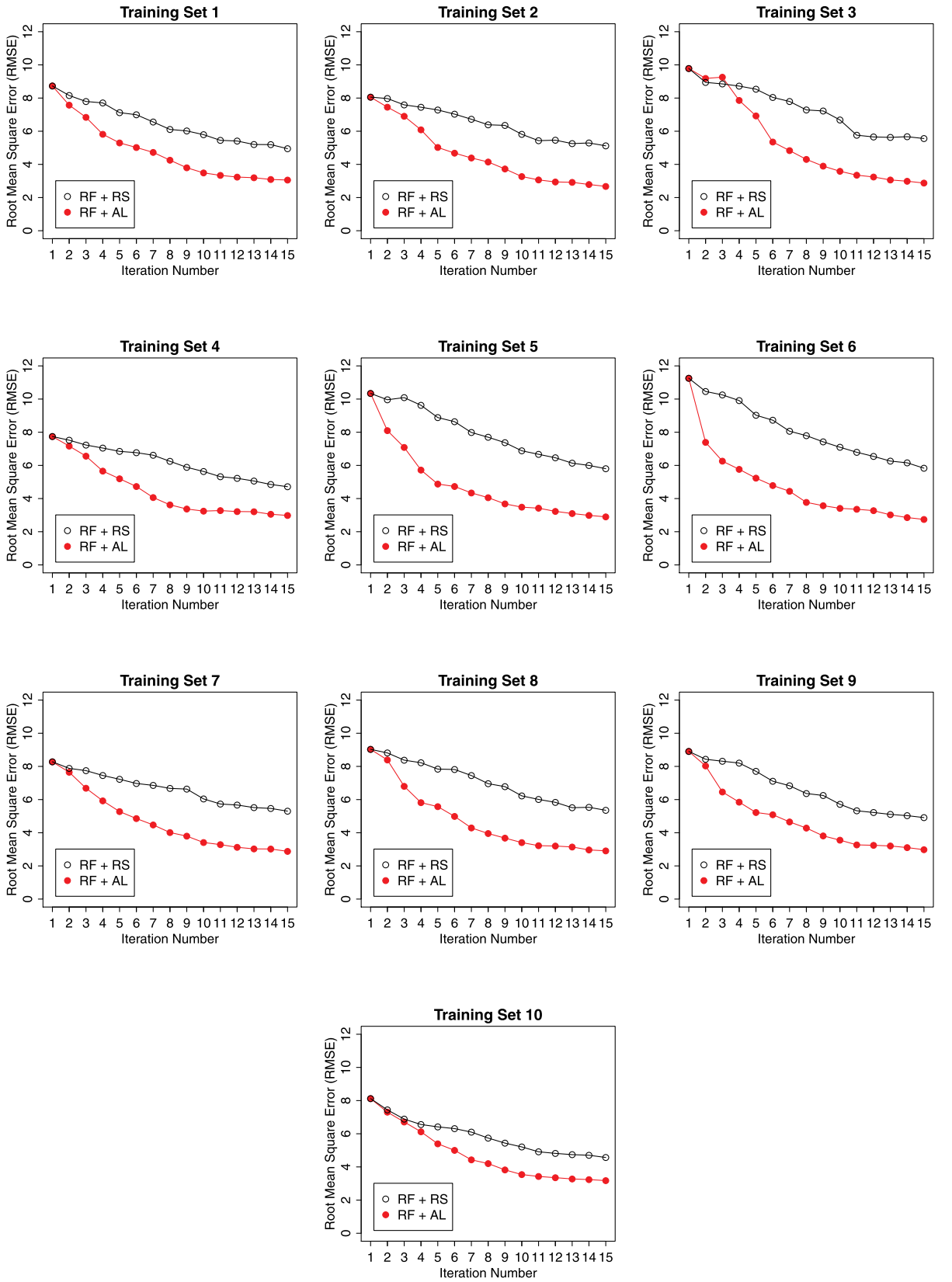


Fig. 4. Errors of two different methods for Segregation model: Random Forest with Random Sampling (RF + RS) and Random Forest with Active Learning (RF + AL) for 10 different initial training sets.

Table 2
Segregation model: Initial and terminal RMSE values of RF metamodels trained with random and sequential sampling for 10 different initial training sets.

Training set	Random forest with random sampling		Random forest with sequential sampling	
	Initial RMSE	Terminal RMSE	Initial RMSE	Terminal RMSE
1	8.7288	4.9470	8.7288	3.0570
2	8.0550	5.1195	8.0550	2.6727
3	9.7829	5.5589	9.7829	2.8754
4	7.7429	4.7161	7.7429	2.9824
5	10.3405	5.8018	10.3405	2.8981
6	11.2553	5.8315	11.2553	2.7427
7	8.2785	5.3026	8.2785	2.8780
8	9.0289	5.3549	9.0289	2.9080
9	8.9048	4.9142	8.9048	2.9796
10	8.1206	4.5704	8.1206	3.1754

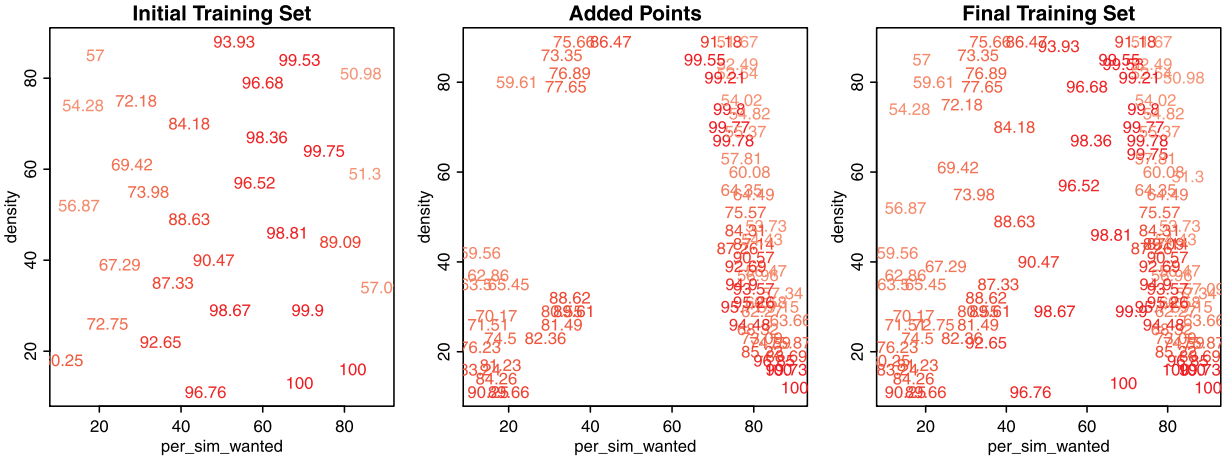


Fig. 5. Initial training set, newly added training points through sequential sampling, and final training set for the most accurate RF metamodel for continuous output (*RMSE* = 2.3835). Numerical values show the *%-similar* values of input parameter combinations. The darker the color is, the higher the *%-similar* value is.

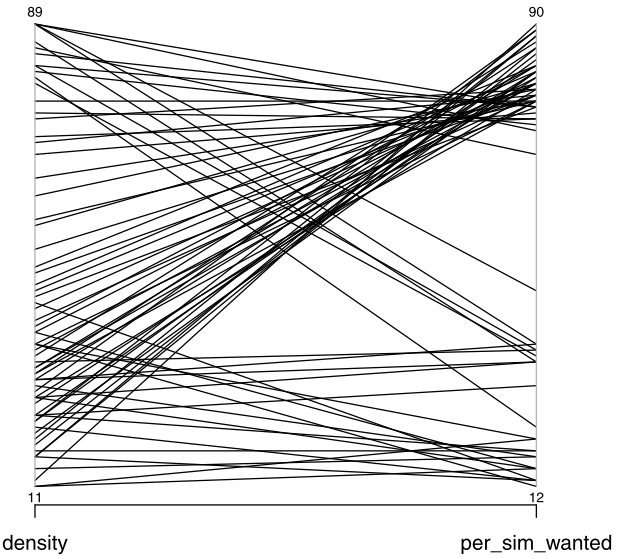


Fig. 6. Parallel coordinates plot of the newly added points through sequential sampling for Segregation model.

Table 3
p-values for all training sets of Traffic Basic model for $m = 1000$ repetitions.

Training set	p-values for H_0: $k \leq 1$ against H_1: $k \geq 2$	p-values for H_0: $k \leq 2$ against H_1: $k \geq 3$
1	0.010	0.657
2	0.014	0.512
3	0.024	0.603
4	0.023	0.444
5	0.023	0.387
6	0.058	0.927
7	0.032	0.716
8	0.010	0.492
9	0.015	0.118
10	0.018	0.623

For simulation models having more than two input parameters, parallel coordinates plot is more useful to inspect the newly added instances. A parallel coordinates plot is a visualization tool to present high-dimensional data where each dimension is represented by a vertical axis, and an instance is represented by connecting the locations of each value in each dimension with a line. Fig. 6 shows an example of parallel coordinates plot for Segregation model. This plot clearly shows that for higher values of *%-similar-wanted*, the metamodel is highly uncertain about the simulation model outputs for nearly all values of *density*. These points correspond to the input subspace where the original simulation model dramatically changes its behavior with small changes in input.

5.2. Traffic basic model

For Traffic Basic model, among 10 different initial training sets each involving 30 model instances (i.e., input parameter combinations), outputs of only one training set (i.e., Training Set 6) exhibits a unimodal distribution at $\alpha = 0.05$ level (Table 3). For the other training sets, the model exhibits a bimodal behavior each of which can be labeled as "low-speed" and "high-speed". Fig. 7 shows an example case where black crosses and red circles indicate the simulation model outputs that can be classified as "low-speed" and "high-speed", respectively. Therefore, we categorize each numerical model output by utilizing Mean Shift algorithm and solve a classification problem for each training set. Cluster centers obtained from Mean Shift algorithm (i.e., peak values of each mode) for each class are given in Table 4.

Fig. 8 comparatively shows the accuracy values of metamodels trained with sequential sampling and random sampling throughout iterations. Since we are dealing with categorical outputs, the predictive performance of the metamodels is measured by accuracy, which is the proportion of the instances in the test set correctly predicted by the metamodel. Table 5 shows initial and terminal accuracy values of RF metamodels trained with random and sequential sampling. Similar to the continuous output case, uncertainty-based sequential sampling procedure performs better than random sampling for categorical model outputs. For all of the initial training sets, the terminal accuracy of metamodels trained with sequential sampling procedure is higher than that of trained with random sampling. Besides, we observe that metamodels trained with sequential sampling technique reach the terminal accuracy of metamodels trained with random sampling at the very early iterations.

Compared to Segregation model, the explanatory power of the sequential sampling is not very obvious since the selected samples do not reflect a regular pattern (Fig. 9). However, the input parameter combinations and their resulting outputs (i.e., "average speed"

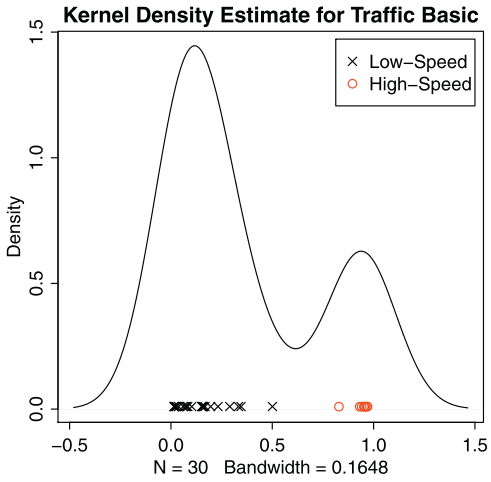


Fig. 7. Kernel density estimate for the outputs of Traffic Basic model. The black crosses and red circles indicate the simulation model outputs that can be classified as "low-speed" and "high-speed", respectively.

Table 4
Peak values of the modes for all training sets of Traffic Basic model.

Training set	Peak 1 (Low-Speed)	Peak 2 (High-Speed)
1	0.0998	0.9330
2	0.1078	0.9251
3	0.1172	0.9384
4	0.0934	0.9437
5	0.1030	0.9047
6	0.0913	0.9014
7	0.1037	0.8776
8	0.1061	0.9309
9	0.1012	0.9329
10	0.1091	0.9200

and "class") show that low *number of cars*, high *acceleration* and low *deceleration* values result in a fast-moving traffic (class 2). Newly added samples also point out some counter-intuitive cases such that the traffic flow is slower even though *number of cars* is low. When we carefully inspect those cases, we observe that the variability of simulation model outputs over 30 replications is very high. For example, for input parameter combination, *number of cars* = 5, *acceleration* = 0.0006392, and *deceleration* = 0.03286, the mean and standard deviation of model outputs over 30 replications are 0.4687 and 0.1283, respectively, with a maximum output value of 0.8560 and a minimum output value of 0.3116. Intuitively, we expect that this input parameter combination will result in a "high-speed" mode. However, we observe that the model is very sensitive to randomly assigned initial conditions (e.g., initial position and initial speed of each car), which lead to diverse model outputs throughout replications. A similar case is also possible when the *number of cars* is high; for input parameter combination, *number of cars* = 41, *acceleration* = 0.006918, and *deceleration* = 0.004356, the mean and standard deviation of model outputs over 30 replications are 0.4864 and 0.2972, respectively, with a maximum output value of 0.9013 and a minimum output value of 0.1925. We can conclude that, for the extreme values of *number of cars*, Traffic Basic model can exhibit counter-intuitive behavior, and these cases are captured by the sequential sampling technique. These parameter combinations indicate the cases whose outputs deviate from the regular and expected behavioral pattern, potentially aiding the verification and validation of the agent-based model.

6. Discussions

Our experimental analysis shows that the coupled use of multimodality detection/output categorization and sequential sampling is very effective in accurately capturing the input–output relationships of agent-based simulation models. Besides, sequential sampling technique reveals the instances which do not obey the input–output regularity captured by the metamodel.

Although our approach is devised for single-output analysis, it is possible to extend it for the analysis of multiple model outputs. The first solution is to handle each output separately without modifying our approach. Despite its simplicity, this approach does not utilize the potential linear/nonlinear associations among different model outputs, which may result in individual metamodels with lower accuracy.

A more sophisticated method is to use multivariate extensions of the approaches used for the univariate (single) output case. For multimodality detection, model outputs must be projected to a one-dimensional space since Silverman's test only deals with univariate distributions. For this purpose, several projection methods were proposed in the literature (e.g., Ahmed and Walther [2], Hahn and Foster [33]) to use Silverman's test to detect multimodality in multivariate distributions. If multimodality is detected, Mean Shift algorithm can be used to determine class labels without any modification since it does not assume univariate distributions. Since Mean Shift algorithm reduces the multivariate continuous outputs to a set of univariate class labels, the existing random forest-assisted sequential sampling technique can be utilized for classification. Alternatively, Edmonds et al. [27] and Patel et al. [61] utilize *k*-means clustering technique to cluster multiple outputs of an agent-based model prior to metamodel fitting. However, the main drawback of *k*-means is that the user must specify the number of clusters beforehand. However, in our two-step procedure, number of clusters is statistically determined by Silverman's test. This prevents a subjective selection of the number of clusters. Besides, *k*-means clustering algorithm can stuck in a local minimum, generating clusters which may not capture the underlying clustering structure even the clustering tendency is high [34].

If the multivariate output distribution is unimodal, multi-output regression techniques, including random forests of multi-target decision trees [45] and multivariate random forests [69], can be used to relate model input parameters to multiple outputs. The reader is referred to Borchani et al. [10] for a survey of multi-output regression techniques.

We demonstrated the proposed approach on two models (i.e., Segregation and Traffic Basic) that are widely known in the agent-based modeling community. Considering the number of procedures and input parameters, these two models can be considered basic ones. However, it should be noted that when considered as black-box transformation functions between their input parameters and outputs, both models demonstrate complex features like tipping points and nonlinear responses. As it is the behavioral complexity that matters when it comes to the issue of accuracy in metamodels rather than number of procedures or parameters, these two models serve as good proxies for evaluating the potential accuracy on even larger models with similar behavioral features. Being said that, growing number of input parameters naturally has significant efficiency implications. Although none of the building blocks of the

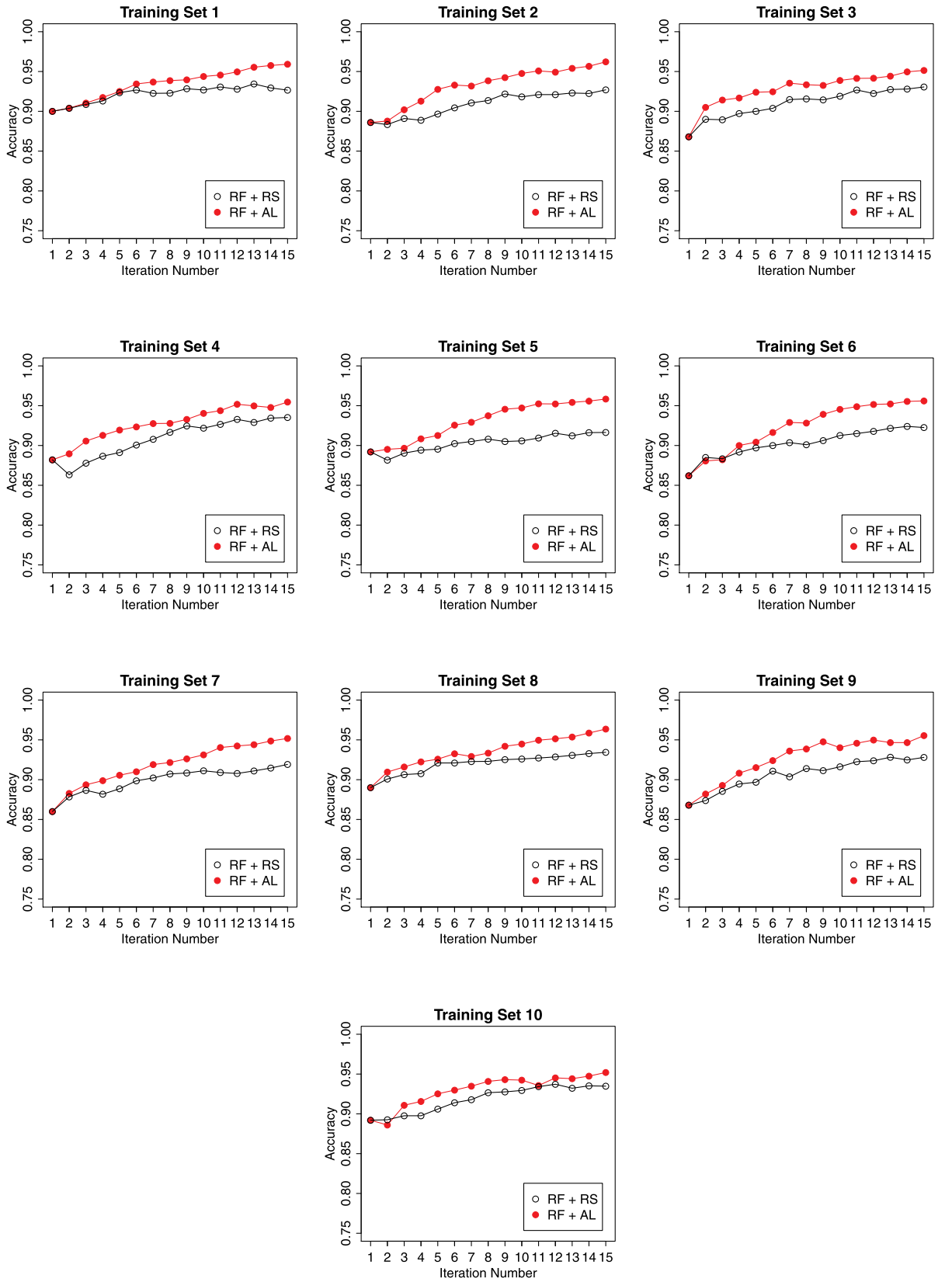


Fig. 8. Accuracies of two different methods for Traffic Basic model: Random Forest with Random Sampling (RF + RS) and Random Forest with Active Learning (RF + AL) for 10 different initial training sets.

Table 5
Traffic Basic model: Initial and terminal accuracy values of RF metamodels trained with random and sequential sampling for 10 different initial training sets.

Training set	Random forest with random sampling		Random forest with sequential sampling	
	Initial accuracy	Terminal accuracy	Initial accuracy	Terminal accuracy
1	0.9000	0.9266	0.9000	0.9592
2	0.8860	0.9270	0.8860	0.9622
3	0.8680	0.9306	0.8680	0.9514
4	0.8820	0.9352	0.8820	0.9546
5	0.8920	0.9164	0.8920	0.9584
6	0.8620	0.9226	0.8620	0.9560
7	0.8600	0.9192	0.8600	0.9518
8	0.8900	0.9344	0.8900	0.9636
9	0.8680	0.9280	0.8680	0.9554
10	0.8920	0.9348	0.8920	0.9520

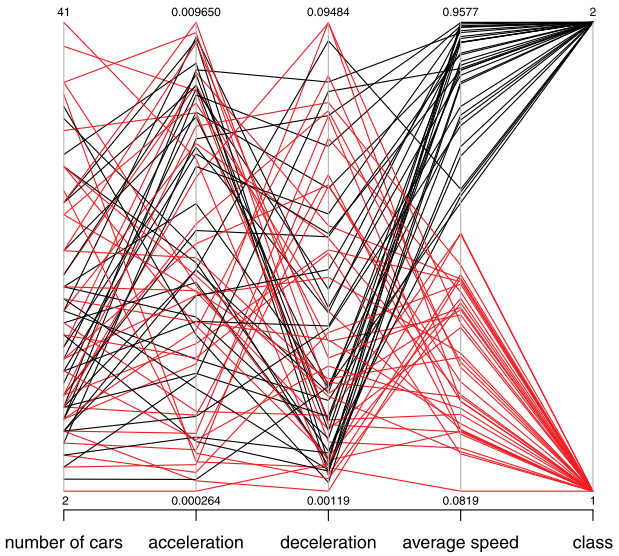


Fig. 9. Parallel coordinates plot of the newly added points through sequential sampling for Traffic Basic model (class = 1 stands for "low-speed" cases, and class = 2 stands for "high-speed" cases).

proposed approach assumes an upper limit for the number of input parameters, the number of instances required to obtain reliable and accurate metamodels grows exponentially due to the *curse of dimensionality*. However, same holds true for the conventional way of expert-driven, mainly manual model analysis. Despite its exponentially growing computation time, the proposed approach stands as an automated process that can decrease the demand for an analyst's time significantly for a task that would require an immense amount of time to execute properly to full extent.

7. Conclusions

In this paper, we first propose a two-step procedure for the categorization of agent-based model outputs. The first step of the procedure utilizes Silverman's test for multimodality detection. If the test procedure detects a multimodal distribution of the agent-based model outputs, the second step employs Mean Shift clustering algorithm by using the information gained from Silverman's test. In this way, the analyst can focus on qualitative model behaviors rather than numerical model outputs. We apply our procedure to both Segregation and Traffic Basic model outputs and, we observe that the outputs of Traffic Basic model show a significant bimodal distribution that can be categorized as "low-speed" and "high-speed" modes. However, the outputs of Segregation model do not exhibit a multimodal distribution. Therefore, in this case, preserving the original model outputs is advantageous since the model is capable of generating a variety of numerical outputs rather than naturally clustered outputs.

The abovementioned categorization procedure requires two different approaches for metamodel training: for datasets (i.e., a set of input parameter combinations and their resulting outputs) with numerical outputs, the task is to solve a regression problem and for datasets with categorical outputs, the training involves a multi-class classification problem. Considering this fact, we use random forest classification and regression technique for metamodel development. However, we use sequential sampling technique, which is

based on uncertainty sampling, to train metamodels. Experimental results show that, as stated in many publications, metamodels trained with sequential sampling technique yield higher accuracy compared to metamodels trained with random sampling. As a further contribution, we focus on the interpretation of the input parameter combinations selected by the sequential sampling procedure. We observe that selected input parameter combinations capture the boundaries of tipping points and counter-intuitive outcomes. These results are also in line with the definition of the uncertainty sampling. In this regard, metamodel-assisted sequential sampling can support the analyst for understanding the system.

Acknowledgments

Declarations of interest: none

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.simpat.2018.12.006

References

[1] M.N. Adnan, M.Z. Islam, Forex++: a new framework for knowledge discovery from decision forests, Aust. J. Inf. Syst. 21 (2017), https://doi.org/10.3127/ajis.v21i0.1539.
[2] M.O. Ahmed, G. Walther, Investigating the multimodality of multivariate data with principal curves, Comput. Stat. Data Anal. 56 (12) (2012) 4462–4469, https://doi.org/10.1016/j.csda.2012.02.020.
[3] F.M. Alam, K.R. McNaught, T.J. Ringrose, A comparison of experimental designs in the development of a neural network simulation metamodel, Simul. Model. Pract. Theory 12 (7–8) (2004) 559–578, https://doi.org/10.1016/j.simpat.2003.10.006.
[4] E. Alpaydin, Introduction to Machine Learning, MIT press, 2014.
[5] R.L. Axtell, J.M. Epstein, J.S. Dean, G.J. Gumerman, A.C. Swedlund, J. Harburger, S. Chakravarty, R. Hammond, J. Parker, M. Parker, Population growth and collapse in a multiagent model of the kayenta anasazi in long house valley, Proc. Natl. Acad. Sci. 99 (suppl 3) (2002) 7275–7279, https://doi.org/10.1073/pnas.092080799.
[6] N. Barakat, A.P. Bradley, Rule extraction from support vector machines: a review, Neurocomputing 74 (1–3) (2010) 178–190, https://doi.org/10.1016/j.neucom.2010.02.016.
[7] R.R. Barton, M. Meckesheimer, Metamodel-based simulation optimization, Handb. Oper. Res. Manag. Sci. 13 (2006) 535–574, https://doi.org/10.1016/S0927-0507(06)13018-2.
[8] M.G. Baydogan, G. Runger, Learning a symbolic representation for multivariate time series classification, Data Min. Knowl. Discov. 29 (2) (2015) 400–422, https://doi.org/10.1007/s10618-014-0349-y.
[9] B. Beachkofski, R. Grandhi, Improved distributed hypercube sampling, Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, (2002), p. 1274, https://doi.org/10.2514/6.2002-1274.
[10] H. Borchani, G. Varando, C. Bielza, P. Larrañaga, A survey on multi-output regression, Wiley Interdiscip. Rev. 5 (5) (2015) 216–233, https://doi.org/10.1002/widm.1157.
[11] D. Bozağaç, İ. Batmaz, H. Oğuztüzün, Dynamic simulation metamodeling using mars: a case of radar simulation, Math. Comput. Simul. 124 (2016) 69–86, https://doi.org/10.1016/j.matcom.2016.01.005.
[12] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, https://doi.org/10.1023/A:1010933404324.
[13] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC Press, 1984.
[14] G. ten Broeke, G. Van Voorn, A. Ligtenberg, Which sensitivity analysis method should i use for my agent-based model? J. Artif. Soc. Soc.Simul. 19 (1) (2016) 5, https://doi.org/10.18564/jasss.2857.
[15] B. Can, C. Heavey, A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models, Comput. Oper. Res. 39 (2) (2012) 424–436, https://doi.org/10.1016/j.cor.2011.05.004.
[16] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Trans. Pattern Anal. Mach. Intell. 17 (8) (1995) 790–799, https://doi.org/10.1109/34.400568.
[17] T.M. Cioppa, T.W. Lucas, S.M. Sanchez, Military applications of agent-based simulations, Proceedings of the 2004 Winter Simulation Conference, IEEE, 2004, pp. 171–180, https://doi.org/10.1109/WSC.2004.1371314.
[18] S.M. Clarke, J.H. Griebsch, T.W. Simpson, Analysis of support vector regression for approximation of complex engineering analyses, J. Mech. Des. 127 (6) (2005) 1077–1087, https://doi.org/10.1115/1.1897403.
[19] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Trans. Pattern Anal. Mach. Intell. 24 (5) (2002) 603–619, https://doi.org/10.1109/34.1000236.
[20] K. Crombecq, L. De Tommasi, D. Gorissen, T. Dhaene, A novel sequential design strategy for global surrogate modeling, Proceedings of the 2009 Winter Simulation Conference, (2009), pp. 731–742, https://doi.org/10.1109/WSC.2009.5429687.
[21] K. Crombecq, E. Laermans, T. Dhaene, Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling, Eur. J. Oper. Res. 214 (3) (2011) 683–696, https://doi.org/10.1016/j.ejor.2011.05.032.
[22] H. Deng, Interpreting tree ensembles with intrees, arXiv:1408.5456 (2014).
[23] J.L. Deutsch, C.V. Deutsch, Latin hypercube sampling with multidimensional uniformity, J. Stat. Plan Inference 142 (3) (2012) 763–772, https://doi.org/10.1016/j.jspi.2011.09.016.
[24] S. Durieux, H. Pierreval, Regression metamodeling for the design of automated manufacturing system composed of parallel machines sharing a material handling resource, Int. J. Prod. Econ. 89 (1) (2004) 21–30, https://doi.org/10.1016/S0925-5273(03)00199-3.
[25] J. Eason, S. Cremaschi, Adaptive sequential sampling for surrogate model generation with artificial neural networks, Comput. Chem. Eng. 68 (2014) 220–232, https://doi.org/10.1016/j.compchemeng.2014.05.021.
[26] M. Edali, G. Yücel, Automated analysis of regularities between model parameters and output using support vector regression in conjunction with decision trees, J. Artif. Soc. Soc. Simul. 21 (4) (2018) 1, https://doi.org/10.18564/jasss.3786.
[27] B. Edmonds, C. Little, L. Lessard-Phillips, E. Fieldhouse, Analysing a complex agent-based model using data-mining techniques, Social Simulation Conference, (2014).
[28] J.D. Farmer, D. Foley, The economy needs agent-based modelling, Nature 460 (7256) (2009) 685, https://doi.org/10.1038/460685a.
[29] N. Fischer, E. Mammen, J.S. Marron, Testing for multimodality, Comput. Stat. Data Anal. 18 (5) (1994) 499–512, https://doi.org/10.1016/0167-9473(94)90080-9.
[30] J.H. Friedman, Multivariate adaptive regression splines, Ann. Stat. (1991) 1–67.
[31] K. Fukunaga, L. Hostetler, The estimation of the gradient of a density function, with applications in pattern recognition, IEEE Trans. Inf. Theory 21 (1) (1975) 32–40, https://doi.org/10.1109/TIT.1975.1055330.
[32] V. Grimm, E. Revilla, U. Berger, F. Jeltsch, W.M. Mooij, S.F. Railsback, H.-H. Thulke, J. Weiner, T. Wiegand, D.L. DeAngelis, Pattern-oriented modeling of agent-

based complex systems: lessons from ecology, Science 310 (5750) (2005) 987–991, https://doi.org/10.1126/science.1116681.
[33] S. Hahn, P. Foster, Assessing the multimodality of a multivariate distribution using nonparametric techniques, COMPSTAT, Springer, 1998, pp. 329–334.
[34] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer-Verlag, 2009.
[35] S. Heckbert, Mayasim: an agent-based model of the ancient maya social-ecological system, J. Artif. Soc. Soc. Simul. 16 (4) (2013) 11, https://doi.org/10.18564/jasss.2305.
[36] R.D. Hurrion, An example of simulation optimisation using a neural network metamodel: finding the optimum number of kanbans in a manufacturing system, J. Oper. Res. Soc. 48 (11) (1997) 1105–1112, https://doi.org/10.1057/palgrave.jors.2600468.
[37] M.F. Hussain, R.R. Barton, S.B. Joshi, Metamodeling: radial basis functions, versus polynomials, Eur. J. Oper. Res. 138 (1) (2002) 142–154, https://doi.org/10.1016/S0377-2217(01)00076-5.
[38] M.E. Johnson, L.M. Moore, D. Ylvisaker, Minimax and maximin distance designs, J. Stat. Plan. Inference 26 (2) (1990) 131–148, https://doi.org/10.1016/0378-3758(90)90122-B.
[39] B. Kamiński, Interval metamodels for the analysis of simulation input–output relations, Simul. Model. Pract. Theory 54 (2015) 86–100, https://doi.org/10.1016/j.simpat.2015.03.008.
[40] A. Keane, P. Nair, Computational Approaches for Aerospace Design: The Pursuit of Excellence, John Wiley & Sons, 2005.
[41] J.P. Kleijnen, Kriging metamodeling in simulation: a review, Eur. J. Oper. Res. 192 (3) (2009) 707–716, https://doi.org/10.1016/j.ejor.2007.10.013.
[42] J.P. Kleijnen, D. Deflandre, Validation of regression metamodels in simulation: bootstrap approach, Eur. J. Oper. Res. 170 (1) (2006) 120–131, https://doi.org/10.1016/j.ejor.2004.06.018.
[43] J.P. Kleijnen, S.M. Sanchez, T.W. Lucas, T.M. Cioppa, State-of-the-art review: a userâs guide to the brave new world of designing simulation experiments, INFORMS J. Comput. 17 (3) (2005) 263–289, https://doi.org/10.1287/ijoc.1050.0136.
[44] J.P. Kleijnen, R.G. Sargent, A methodology for fitting and validating metamodels in simulation, Eur. J. Oper. Res. 120 (1) (2000) 14–29, https://doi.org/10.1016/S0377-2217(98)00392-0.
[45] D. Kocev, S. Džeroski, M.D. White, G.R. Newell, P. Griffioen, Using single-and multi-target regression trees and ensembles to model a compound index of vegetation condition, Ecol. Model. 220 (8) (2009) 1159–1168, https://doi.org/10.1016/j.ecolmodel.2009.01.037.
[46] Y. Kuo, T. Yang, B.A. Peters, I. Chang, Simulation metamodel development using uniform design and neural networks for automated material handling systems in semiconductor wafer fabrication, Simul. Model. Pract. Theory 15 (8) (2007) 1002–1015, https://doi.org/10.1016/j.simpat.2007.05.006.
[47] F. Lamperti, A. Roventini, A. Sani, Agent-based model calibration using machine learning surrogates, J. Econ. Dyn. Control 90 (2018) 366–389, https://doi.org/10.1016/j.jedc.2018.03.011.
[48] J.-S. Lee, T. Filatova, A. Ligmann-Zielinska, B. Hassani-Mahmooei, F. Stonedahl, I. Lorscheid, A. Voinov, J.G. Polhill, Z. Sun, D.C. Parker, The complexities of agent-based modeling output analysis, J. Artif. Soc. Soc. Simul. 18 (4) (2015) 4, https://doi.org/10.18564/jasss.2897.
[49] D.D. Lewis, W.A. Gale, A sequential algorithm for training text classifiers, Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Springer-Verlag New York, Inc., 1994, pp. 3–12.
[50] Y. Li, S.H. Ng, M. Xie, T. Goh, A systematic comparison of metamodeling techniques for simulation optimization in decision support systems, Appl. Soft Comput. 10 (4) (2010) 1257–1273, https://doi.org/10.1016/j.asoc.2009.11.034.
[51] H. Liu, S. Xu, X. Wang, Sequential sampling designs based on space reduction, Eng. Optim. 47 (7) (2015) 867–884, https://doi.org/10.1080/0305215X.2014.928816.
[52] Y. Liu, Active learning with support vector machine applied to gene expression data for cancer classification, J. Chem. Inf. Comput. Sci. 44 (6) (2004) 1936–1941, https://doi.org/10.1021/ci049810a.
[53] M.W. Macy, R. Willer, From factors to factors: computational sociology and agent-based modeling, Annu. Rev. Sociol. 28 (1) (2002) 143–166, https://doi.org/10.1146/annurev.soc.28.110601.141117.
[54] C.N. Madu, Simulation in manufacturing: a regression metamodel approach, Comput. Ind. Eng. 18 (3) (1990) 381–389, https://doi.org/10.1016/0360-8352(90)90060-Y.
[55] J. Maiora, B. Ayerdi, M. Graña, Random forest active learning for AAA thrombus segmentation in computed tomography angiography images, Neurocomputing 126 (2014) 71–77, https://doi.org/10.1016/j.neucom.2013.01.051.
[56] M. Mashayekhi, R. Gras, Rule extraction from random forest: the RF + HC methods, Canadian Conference on Artificial Intelligence, Springer, 2015, pp. 223–237.
[57] M. Mashayekhi, R. Gras, Rule extraction from decision trees ensembles: new algorithms based on heuristic search and sparse group lasso methods, Int. J. Inf. Technol.Decis. Mak. 16 (06) (2017) 1707–1727, https://doi.org/10.1142/S0219622017500055.
[58] M.D. McKay, R.J. Beckman, W.J. Conover, Comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 21 (2) (1979) 239–245, https://doi.org/10.1080/00401706.1979.10489755.
[59] A.A. Mullur, A. Messac, Metamodeling using extended radial basis functions: a comparative approach, Eng. Comput. 21 (3) (2006) 203, https://doi.org/10.1007/s00366-005-0005-7.
[60] J.N. Myhre, K.Ø. Mikalsen, S. Løkse, R. Jenssen, Robust clustering using a knn mode seeking ensemble, Pattern Recognit. 76 (2018) 491–505, https://doi.org/10.1016/j.patcog.2017.11.023.
[61] M.H. Patel, M.A. Abbasi, M. Saeed, S.J. Alam, A scheme to analyze agent-based social simulations using exploratory data mining techniques, Complex Adapt. Syst. Model. 6 (1) (2018) 1, https://doi.org/10.1186/s40294-018-0052-8.
[62] L.T.K. Phung, V.T.N. Chau, N.H. Phung, Extracting rule rf in educational data classification: From a random forest to interpretable refined rules, International Conference on Advanced Computing and Applications (ACOMP), IEEE, 2015, pp. 20–27, https://doi.org/10.1109/ACOMP.2015.13.
[63] S.F. Railsback, V. Grimm, Agent-Based and Individual-Based Modeling: A Practical Introduction, Princeton University Press, 2011.
[64] M. Resnick, Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds, MIT Press, 1997.
[65] A. Riener, A. Ferscha, Effect of proactive braking on traffic flow and road throughput, Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, (2009), pp. 157–164, https://doi.org/10.1109/DS-RT.2009.11.
[66] I. Salle, M. Yıldızoğlu, Efficient sampling and meta-modeling for computational economic models, Comput. Econ. 44 (4) (2014) 507–536, https://doi.org/10.1007/s10614-013-9406-7.
[67] S.M. Sanchez, T.W. Lucas, Exploring the world of agent-based simulations: simple models, complex analyses, Proceedings of the 34th Winter Simulation Conference, (2002), pp. 116–126, https://doi.org/10.1109/WSC.2002.1172875.
[68] T.C. Schelling, Dynamic models of segregation, J. Math. Sociol. 1 (2) (1971) 143–186, https://doi.org/10.1080/0022250X.1971.9989794.
[69] M. Segal, Y. Xiao, Multivariate random forests, Wiley Interdiscip. Rev. 1 (1) (2011) 80–87, https://doi.org/10.1002/widm.12.
[70] B. Settles, Active Learning Literature Survey, Computer Sciences Technical Report, University of Wisconsin–Madison, 2009.
[71] R. Sheikholeslami, S. Razavi, Progressive latin hypercube sampling: an efficient approach for robust sampling-based analysis of environmental models, Environ. Model. Softw. 93 (2017) 109–126, https://doi.org/10.1016/j.envsoft.2017.03.010.
[72] B.W. Silverman, Using kernel density estimates to investigate multimodality, J. R. Stat. Soc. (1981) 97–99.
[73] F. Squazzoni, Agent-Based Computational Sociology, John Wiley & Sons, 2012.
[74] L. Tesfatsion, Agent-based computational economics: growing economies from the bottom up, Artif. Life 8 (1) (2002) 55–82, https://doi.org/10.1162/106454602753694765.
[75] L. Van Gelder, P. Das, H. Janssen, S. Roels, Comparative study of metamodelling techniques in building energy simulation: guidelines for practitioners, Simul. Model. Pract. Theory 49 (2014) 245–257, https://doi.org/10.1016/j.simpat.2014.10.004.
[76] N. Villa-Vialaneix, M. Follador, M. Ratto, A. Leip, A comparison of eight metamodeling techniques for the simulation of n_2o fluxes and n leaching from corn crops, Environ. Model. Softw. 34 (2012) 51–66, https://doi.org/10.1016/j.envsoft.2011.05.003.
[77] U. Wilensky, Netlogo segregation model, 1997, (http://ccl.northwestern.edu/netlogo/models/Segregation).

[78] U. Wilensky, NetLogo traffic basic model, 1997, (http://ccl.northwestern.edu/netlogo/models/TrafficBasic).
[79] U. Wilensky, NetLogo, 1999, (http://ccl.northwestern.edu/netlogo/).
[80] U. Wilensky, M. Resnick, Thinking in levels: a dynamic systems approach to making sense of the world, J. Sci. Educ. Technol. 8 (1) (1999) 3–19, https://doi.org/10.1023/A:1009421303064.
[81] R.A. Williams, Lessons learned on development and application of agent-based models of complex dynamical systems, Simul. Model. Pract. Theory 83 (2018) 201–212, https://doi.org/10.1016/j.simpat.2017.11.001.
[82] F. Xiong, Y. Xiong, W. Chen, S. Yang, Optimizing latin hypercube design for sequential sampling of computer experiments, Eng. Optim. 41 (8) (2009) 793–810, https://doi.org/10.1080/03052150902852999.
[83] Q. Zhou, X. Shao, P. Jiang, H. Zhou, L. Shu, An adaptive global variable fidelity metamodeling strategy using a support vector regression based scaling function, Simul. Model. Pract. Theory 59 (2015) 18–35, https://doi.org/10.1016/j.simpat.2015.08.002.