Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, eds.

AUTOML APPROACH TO CLASSIFICATION OF CANDIDATE SOLUTIONS FOR SIMULATION MODELS OF LOGISTIC SYSTEMS

Ilya Jackson
Josué C. Velázquez-Martínez

Center for Transportation & Logistics
Massachusetts Institute of Technology
1 Amherst Street
Cambridge, MA, USA

ABSTRACT

This paper proposes a framework based on automated machine learning for the classification of candidate solutions for logistic and industrial discrete event simulation models. The proposed framework aims to augment the capabilities of a decision-maker to make critical yes-no decisions quickly with confidence and nearly perfect accuracy. The framework is based on a combination of artificial neural networks and a genetic algorithm. Such that the genetic algorithm orchestrates both neural architecture search and hyperparameter optimization. The paper demonstrates that the classifiers obtained through the proposed procedure can learn and generalize complex nonlinear relations within the discrete-event simulation models of subsystems vitally crucial for food supply chains and classify a candidate solution as profitable or not.

1 INTRODUCTION

Real-world dynamic systems, as a rule, require operational decisions on a repetitive basis. A manifold of such systems can be found in the context of supply chain management, logistics, and production. Decision-makers work under constant tough pressure to make critical decisions quickly, requiring a decision-support system to arrive at a yes-no conclusion with confidence and nearly perfect accuracy. Therefore, prediction in general and binary classification, in particular, are core elements of decision-making (Agrawal et al. 2018). However, two possible obstacles arise. First, predictive models, especially state-of-the-art, require the sheer amount of data that the physical system, even stuffed with sensors and IoT devices, cannot provide. Second, the complexity, dimensionality, and stochasticity of real-world systems frequently make analytic approaches unreliable. Discrete event simulation (DES) can be used to overcome both obstacles. DES model allows one to model complex stochastic systems in the most straightforward form, namely in the form of an executable computer program. DES models can substitute the modeled physical asset and be applied for system performance prediction and control policy formulation. Besides, DES models are a perfect source of synthetic data, which can be obtained at a far lower cost and in a shorter time. Furthermore, it is worth emphasizing that desirable experiments with the real system are not feasible in many cases due to physical or legislation constraints, associated risks, or simply because a system in necessary configurations does not yet exist.

However, if a DES model is complex, realistic, and detailed, the computational burden can be substantial (Barton 2015). In this case, it is reasonable to take advantage of an alternative model known as a metamodel. Roughly speaking, a metamodel is a "model of the model" built to approximate the output of interests of the original DES with a specified degree of accuracy. However, insofar as a yes-no decision regarding a candidate solution is a primary consideration for a plethora of problems in logistics and supply chain management, it appears to be reasonable not to approximate an exact numerical outcome of a

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

candidate solution but to classify it according to a particular criterion (e.g., profitability or feasibility). Machine learning models in general and artificial neural networks in particular are capable of performing such classifications. However, there is a significant challenge that cannot be ignored. Although machine learning algorithms are universal and task-unspecific, they are sensitive to various design considerations: feature selection and extraction, model selection, neural architecture search, and hyperparameter optimization (Hutter et al. 2019). This search for a decent model through laborious fine-tuning and customization requires the participation of qualified specialists. Automated machine learning (AutoML) has an immense potential to make advanced machine learning accessible to domain scientists, which can be considered as "democratization" of the field. In this regard, this paper proposes an AutoML framework for the classification of candidate solutions for logistic and industrial DES models.

The ongoing COVID-19 pandemic has tested food systems to their limits and revealed vulnerabilities in food supply chains from farm to fork. The UN Food and Agriculture Organization predicts that the number of people suffering from food insecurity could rise significantly. Namely, since the world is globalized and interconnected as never before, the combination of movement and trade restrictions caused by the pandemic can disrupt global food supply chains, severely exacerbating existing food shortages. In this regard, this paper focuses on DES models of subsystems vitally crucial for food supply chains. Namely, a grocery retailer's inventory control system operates with perishable products and a production-inventory system with Markov-modulated demand typical for canning and beverage industries.

The coming text is structured as follows. Section 2 presents the related work and highlights the novelty. Section 3 sheds light on the methodology behind the approach. Section 4 contains the description of the DES models under consideration and experimental results. Section 5 discusses the results, provides possible interpretations, and outlines shortcomings. Section 6 summarizes the results and provides a concluding statement. Since the reproducibility of experiments is a pivot of the scientific method, all the source-code of DES models and the developed framework can be found in the GitHub repository (Metainventory 2021).

2 RELATED WORK AND NOVELTY

A suggested approach is closely related to the concept of metamodeling with artificial neural networks. The term metamodel was introduced and further popularized by Blanning (1975). The term indicates an application of an alternative cheaper-to-compute model to mimic the input-output mapping of DES or other complex simulation models. Among many metamodeling techniques, the most frequently used and praised in the literature are response surface methodologies, kriging models, radial basis functions. Nevertheless, the rising popularity of deep-learning sparked a surge of interest in metamodeling with artificial neural networks. Metamodeling using artificial neural networks can be viewed as a regression problem. However, if data generated by a DES model is labeled according to specific criteria, for example, the profitability of a candidate solution, metamodeling can be alternatively considered a binary classification problem.

The first attempt of metamodeling automation can be traced back to Wang (2005), who applied a genetic algorithm (GA) to discover a neural architecture capable of approximating the vessel design problem. Nezhad and Mahlooji (2014) meticulously studied metamodels based on artificial neural networks for realistic inventory control DES models under (s, S) policy. A metamodel based on an artificial neural network was successfully applied to perform a metamodel-based optimization of an order-picking system (Dunke and Nickel 2020). The authors concluded with the statement that fidelity and flexibility of neural networks with acyclic structure make them attractive for metamodeling and metamodel-based optimization of complex industrial systems. Owoyele et al. (2021) combined GA with machine learning techniques for the metamodeling of computational fluid dynamics simulations. GA was applied for automated model selection and hyperparameter optimization. A recent paper demonstrated the efficiency of a combination of artificial neural networks and GA for metamodeling automation of logistic and production systems (Jackson 2021).

The framework proposed in this paper inherits the core ideas of its predecessors. However, unlike the predecessors, the framework automates both neural architecture search and hyperparameter optimization.

Besides, special attention is paid to such overfitting preventing techniques as regularization and dropout as well as optimizers with momentum term.

3 METHODOLOGY

This section demonstrates how metamodeling can be considered from a binary classification point of view and describes the AutoML framework for deriving a model capable of performing such classifications.

3.1 Metamodeling as a classification problem

A complex industrial system simulation model can be viewed as a nonlinear input-output map represented by a "black-box" function *y = f(x)*, where *x* stands for the vector of inputs and y is the output. Since the output is stochastic in cases covered by this paper, the average value estimated based on *n* replications is an output of specific interest $\bar{y} = n^{-1}\sum_{i=1}^{n} f(x)$. The exact value of *n* can be derived using the method based on confidence intervals proposed by Byrne (2013).

$$n = \left(\frac{z_{\alpha/2}}{\zeta} CV\right)^2. \tag{1}$$

where *n* is a minimum number of simulation runs to achieve desired confidence interval width ζ (expressed as a multiplier by the mean) for the stochastic output with a coefficient of variation *CV*.

A dataset $\tau = \{x_i, \bar{y}_i\}_{i=1}^{N}$ can be obtained by computing corresponding simulation outputs for inputs generated in the feasible range. In this context, the classic metamodeling is the process of deriving a deterministic function $\Phi(x)$ that can be fit by the data from τ and approximate the averaged output of the original simulation with the sufficient degree of accuracy $\|\Phi(x) - \bar{y}\| < \varepsilon$, where ε is a positive value, small enough for the problem under consideration. Alternatively, metamodeling can be viewed as a binary classification problem. Since the output of simulation models considered in this paper corresponds to the net profit, the dataset τ can be labeled depending on whether the input x_i entails positive net profit or not.

$$label_i = \begin{cases} 1, if\ y_i > 0 \\ else\ 0 \end{cases}. \tag{2}$$

After profitable candidate solutions are labeled as 1 and the rest as 0, a new labeled dataset $\tilde{\tau} = \{x_i, label_i\}_{i=1}^{N}$ is obtained and can be further used for binary classification. In these settings, the inputs can be classified as either leading to profit or not by minimizing the binary cross-entropy function, also known as a log-loss score:

$$CE = -\frac{1}{N}\sum_{i=1}^{N} label_i \log_2\big(Prob(label_i)\big) + (1 - label_i)\log_2(1 - Prob(label_i)).$$

Table 1 demonstrates the confusion matrix taking into account the context of the problems under consideration. The confusion matrix allows one to visualize the performance of a supervised learning algorithm. Each row of the matrix stands for the instances in a predicted class, while each column represents the instances in an actual class

Table 1: The confusion matrix in the context of the considered problems.

Confusion matrix		**Actual class**	
		Profitable (1)	Non-profitable (0)
Predicted class	Profitable (1)	True positive (*tp*)	False positive (*fp*)
	Non-profitable (0)	False negative (*fn*)	True negative (*tn*)

Each row of the confusion matrix corresponds to the instances of a predicted class, whereas columns stand for the instances of an actual class. Calculating true positive (*tp*), false positive (*fp*), false negative (*fn*), true negative (*tn*) instances, a plethora of performance estimation metrics can be calculated, including accuracy, precision, recall, and F_1*-score*:

$$accuracy = \frac{tp+tn}{tp+tn+fp+fn}.$$
$$precision = \frac{tp}{tp+fp}.$$
$$recall = \frac{tp}{tp+fn}.$$
$$F_1 = \frac{2*precision*recall}{precision+recall}.$$

3.2 AutoML framework for classifying candidate solutions

A multilayer perceptron (MLP) equipped with regularization and dropout layers is chosen as a baseline model. Several reasons can support this choice. First, MLP is the most general feedforward architecture, and other feedforward networks can be considered its specific case, making it especially suitable if there is a lack of information regarding the structure of the problem. Second, the universal approximation theorem recently extended by Hanin (2019) guarantees that MLP equipped with nonlinear activation functions can, in principle, learn and generalize nonlinear relations between simulation variables. Third, the presence of regularization and dropout layers serves as an efficient overfitting-preventing mechanism (Wager et al. 2013).

Despite the mentioned advantages, MLP is sensitive to a manifold of design considerations related to architecture, optimizers, regularization constants, and other hyperparameters. As it has been demonstrated by Mazzawi et al. (2019), a neural architecture search and hyperparameter optimization conducted using evolutionary computations can rival and even outperform human data scientists and machine learning experts. The proposed AutoML procedure for deriving an MLP classifier is driven by GA distinguished by Gray-code chromosome representation, uniform crossover, and tournament selection. This recipe is not accidental and can be justified by evidence from seminal papers. Because of space-efficiency, a binary representation is an apparent choice. However, chromosome representations vary significantly even among different binary schemes. In the case of the reflected Gray coding scheme, only one bit has to be flipped in order to reach the neighboring value. This property postulates gradualism, which is necessary to overcome problems related to large Hamming clefts (Caruana and Schaffer 1988). In the uniform crossover, the number of crossover points is not constant, and flipping a bit is made independently for every single bit in a chromosome. In this regard, the crossover operator will less likely split up the fit building blocks. Additionally, the uniform crossover can potentially decrease the chance of premature convergence (Back et al. 2018). The tournament selection allows one to fine-tuned the convergence by varying the tournament size (Miller and Goldberg 1995).

In high-level programming interfaces like Keras, artificial neural networks can be assembled from prebuilt building blocks that include layers, activation functions, optimizers, along with corresponding hyperparameters that include learning rate, momentum term, and regularization terms (Chollet 2018). If this information is encoded into a binary chromosome $a \in A$, GA can orchestrate the morphism of MLP architecture and hyperparameters searching for such a chromosome $a^* \in A$ that produces the fittest classifier $\Phi(a^*, x)$. Fitness in this context is measured by the average *F1-score* estimated using the k-fold cross-validation. Figure 1 demonstrates a complete pipeline from simulation replication to the derivation of the fittest classifier.

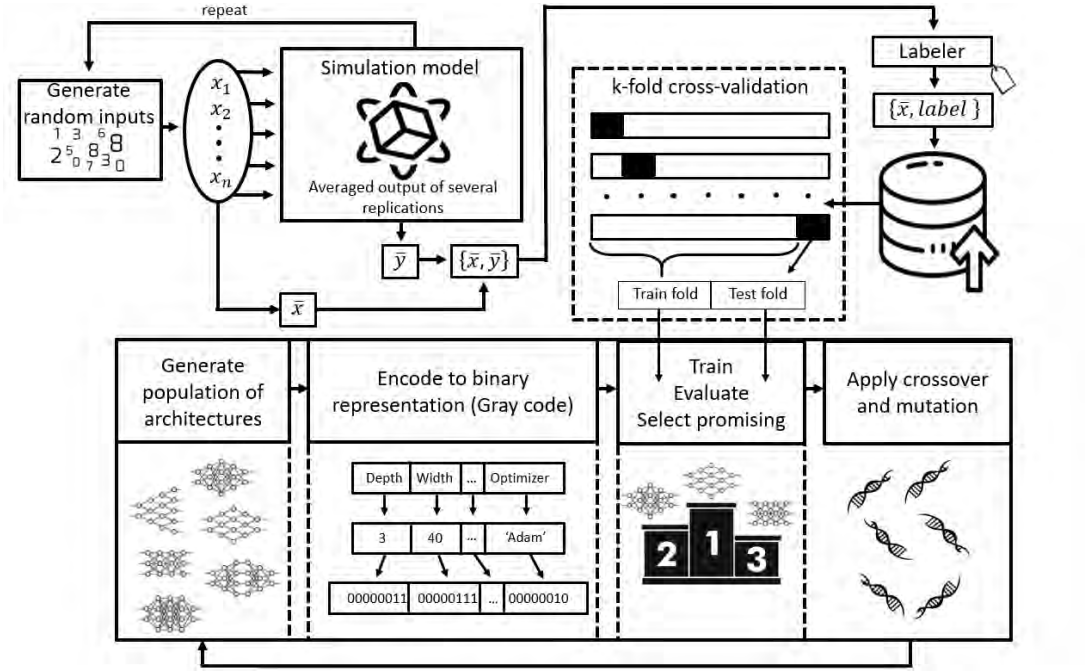

Figure 1: AutoML framework for driving the MLP capable of classifying candidate solutions.

The procedure starts with generating a random vector of inputs in the feasible range. In order to generate one observation, a DES model is replicated with randomly generated input n times. An exact value of n is calculated using equation (1). The obtained output is averaged and labeled according to equation (2). By repeating this procedure in a loop, a dataset is synthesized. The dataset is split by training and test subsamples using k-fold cross-validation. Such that every MLP-based classifier is expected to predict the label for such vectors of inputs that have not been encountered during the training phase.

3.3 Simulations under consideration

The industrial systems under consideration are represented as DES models and implemented in Python 3.7.

3.3.1 Perishable inventory system

The first DES model under consideration corresponds to the inventory control of a grocery retailer that operates with perishable products. The model is referred to as PIS (perishable inventory system) further in the text. The complete description of the model can be found in (Jackson 2019a). The inventory control system operates with multiple products under the constrained total inventory capacity. Such that inbound exceeding the available capacity is unfeasible and penalized. See Figure 2 for example.

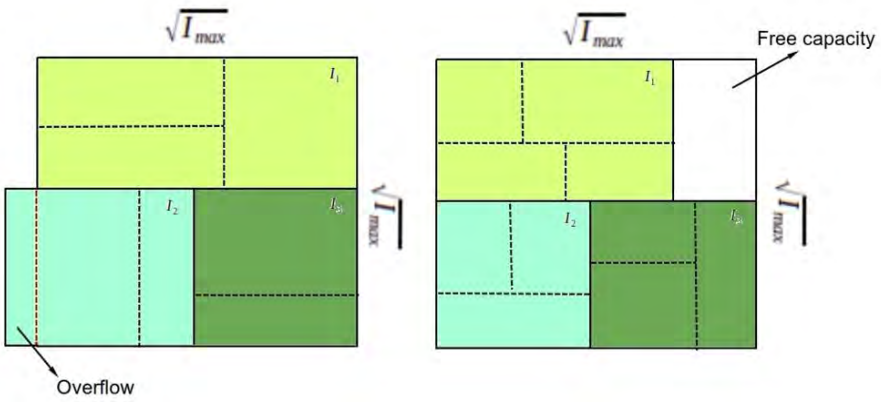

Figure 2: The limited storage capacity can be illustrated as the square with side length equal to the square root of the maximum storage capacity $\sqrt{I_{max}}$.

The model incorporates realistic perishability mechanics. See Figure 3 for example. Products that the inventory control system operate with constitute a sequence $P = (p_1, p_2, \ldots, p_n)_{n\in\mathbb{N}^+}$. Timings of discrete events constitute a sequence $T = (t_1, t_2, \ldots, t_n)_{n\in\mathbb{N}^+}$. In order to model perishability realistically, the storage is denoted as a sequence of independent batches replenished at different moments of time $S_t^p = (s_1^{p,t}, s_2^{p,t}, \ldots, s_n^{p,t})_{n\in\mathbb{N}^+}$. Such that for each S_t^p there is a corresponding sequence of days to expiration $E_t^p = (e_1^{p,t}, e_2^{p,t}, \ldots, e_n^{p,t})_{n\in\mathbb{N}^+}$. For every single product at each instance of time, there is an inventory level $\sum_{i=1}^{n} S_t^p$. The following equation models how the days to expiration decrease over time:

$$E_{t+1}^p = \varepsilon(E_t^p, t_i, t_{i+1}) = (e_0^{p,t} - (t_{i+1} - t_i), \ldots, e_n^{p,t} - (t_{i+1} - t_i)).$$

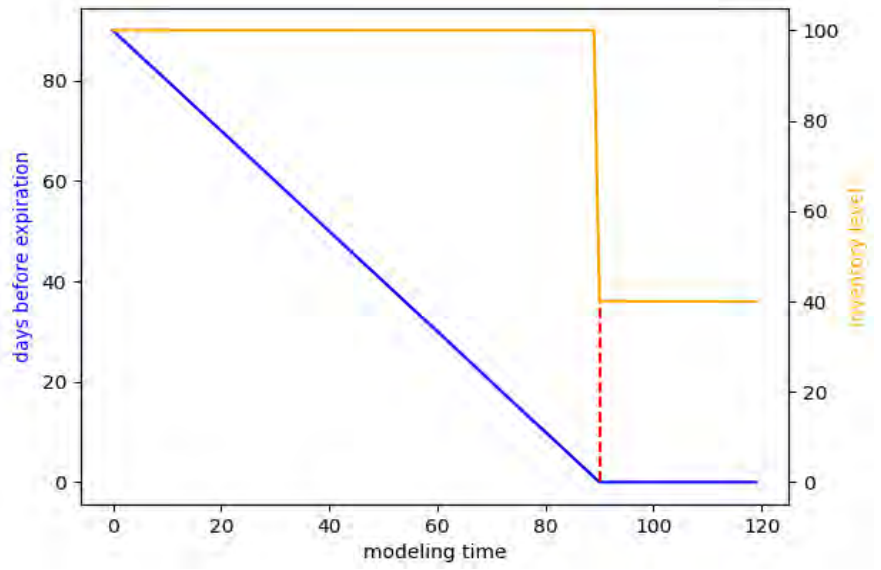

Figure 3: Illustration of the perishability mechanics.

For all products in the inventory system, there is a pair of control parameters (Q^p, r^p) that postulates a replenishment policy. As soon as the current inventory level reaches the reorder point r^p, namely, the condition $I_t^p \leq r^p$ is satisfied, a new batch of the size Q^p is ordered. The system operates in a stochastic

environment. In the following numerical example, demand size and replenishment lead times are normally distributed, and demand interarrival time is distributed exponentially.

Five composite costs are considered: ordering costs, inventory costs, backorder costs, overflow fee, and recycling fee. Ordering cost includes both purchase price and transportation cost and costs for associated logistics (loading, packaging, unloading). The principle of cut-off point quantity discount is adopted. Such that discount is given only to the extent that the order exceeds a cut-off point. Depending on the context, unit inventory cost can consist of handling costs and opportunity loss associated with the "frozen capital". In this context, the "frozen capital" is the lost possibility of using the capital for other purposes. Inventory cost is constant and set for a unit of modelling time. Every single out-of-stock is considered to be associated with a loss of business reputation. When a product is backordered, a customer is directly stimulated to search for a substitute. Therefore, every out-of-stock event is related to a constant fee. Overflows (exciding the total inventory capacity) are also penalized by a constant fee. In real world, such expenses are associated with reverse logistics. Therefore, when a batch perishes, a similar penalty related to reverse logistics of expired goods arises (a batch must be sent for recycling). Taking into account that each unit of product is sold by a constant price. Total net profit of the system is treated as the simulation output. Figure 4 demonstrates the dynamics of commodity and financial flows.

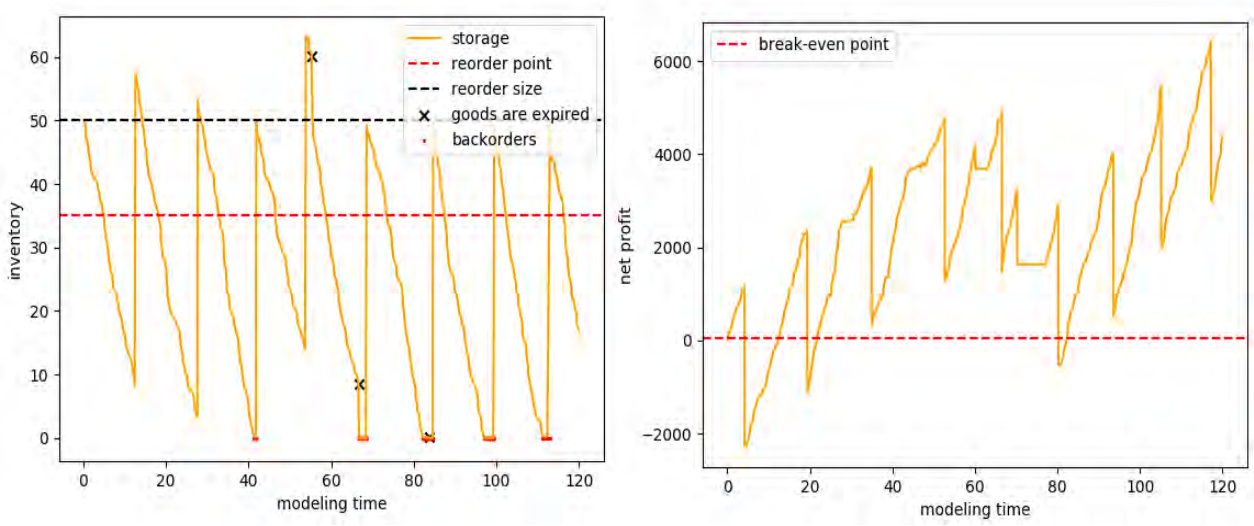

Figure 4: Dynamics of commodity and financial flows.

To summarize, the DES model is multiproduct, stochastic, incorporates lost sales, discounts, and perishability.

3.3.2 Markov-modulated production-inventory system

The second DES model corresponds to a production-inventory system with Markov-modulated demand. The model is referred to as MMPS (Markov-modulated production-inventory system) further in the text. The demand process is associated with an underlying Markov chain. Such that demand states of a Markov process represent environmental parameters. The complete description of the model can be found in (Jackson 2019b).

As in the previous case, the production-inventory system also operates with a sequence of products *P,* and the total storage capacity is limited. For each product inventory level at time t is given by a corresponding sequence $S_t = (S_1, \ldots, S_p)_{p \in P}$. Each product is produced on an individual machine with stochastic production rates measured in items per unit of time. Each production rate is distributed normally.

At any moment, incoming demand consists of all products $D_t = (D_1, \ldots, D_p)_{p \in P}$. Such demand for each particular product is a random variable under the distribution that depends on the current environmental state. Considering a set of demand states $\Pi = (1, \ldots, q)_{q \in N^+}$, the demand state is a Markov chain over Π with the corresponding transition matrix $M_p = \{m_{ij}\}$. Such that $0 \leqslant m_{ij} \leqslant 1$, and $\sum m_{ij} = 1 \; \forall i, j \in \Pi$. Figure 5 illustrates the particular case with three environmental states.

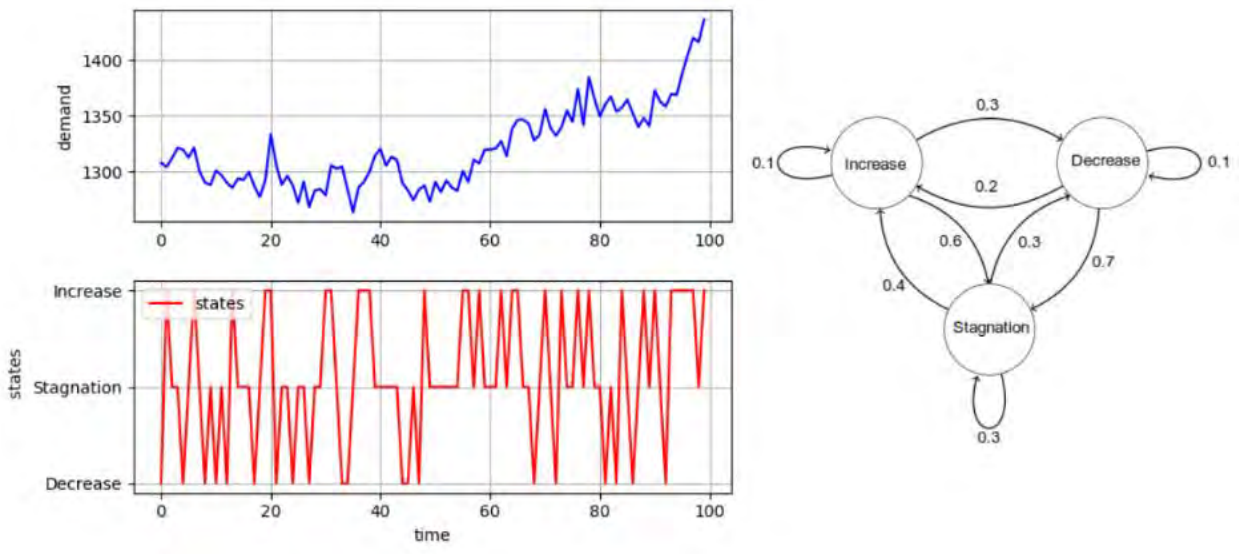

Figure 5: An example of Markovian demand for a market with three environmental states.

Demand inter-arrival time is also a random variable under exponential distribution. According to the all-or-nothing principle, arising demands are satisfied depending on the available inventory capacity. Namely, either the composite demand is fulfilled completely or considered wholly lost. Such settings are pretty typical for cross-docking and consolidation problems.

The system operates according to quite a straightforward control rule. Namely, there is a sequence of production statuses $U_t = (U_1, \ldots, U_p)_{p \in P}$, so as $U_{t,p} \in \{True, False\} \; \forall p \in P$. For each product, there is a pair of control parameters (*Gp, Hp*), which prescribe the inventory level that must be reached to stop and resume the production process. Each resumption of production is associated with setup costs. The following function is denoted to switch the statuses and count the corresponding setups:

$$u(S_t, U_t) = \begin{cases} U_{t,p} \leftarrow False, if \; U_{t,p} = True \; and \; S_{t,p} > \; G^p \\ U_{t,p} \leftarrow True, \; Set_p \leftarrow Set_p + 1, if \; U_{t,p} = False \; and \; S_{t,p} < \; H^p \end{cases}$$

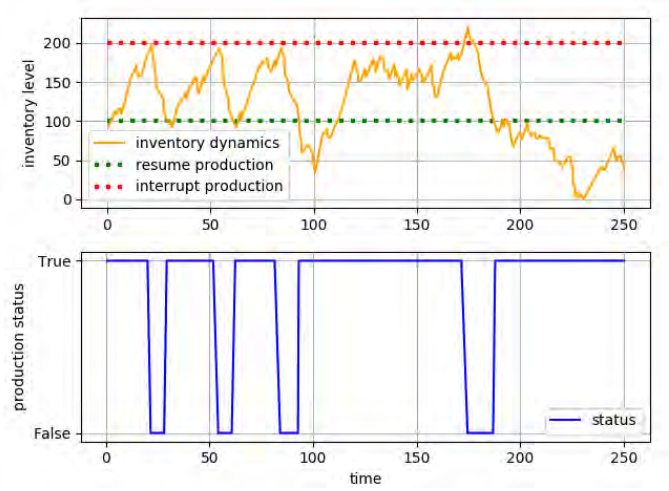

Figure 6: Inventory dynamics under the control rules.

Each product unit has production costs, handling costs, setup costs, and fees associated with lost sales and overflows. Each unit of product is sold for a constant price. The total net profit of the production-inventory system is the simulation output of special interest and subject of maximization. To summarize, the production-inventory model can be characterized as multiproduct and stochastic with lost-sales and Markovian demand.

3.4 Testing the AutoML framework

An experiment starts with generating a random vector of inputs in the feasible range for a 10-product case of the PIS model and a 4-product case for the MMPS model with three market states. The DES models contain 151 and 81 independent variables respectively. In order to generate one observation, the PIS and MMPS models are replicated with randomly generated input 35 and 44 times respectively. The obtained output is averaged and labeled. By repeating this procedure in a loop, two datasets of 600 observations are synthesized. Both datasets are split by training and test subsamples using 10-fold cross-validation. Such that every MLP-based classifier is expected to predict the label for such vector of inputs that have not been encountered during the training phase. Besides, inputs are standardized before training using Z-score. AutoML procedure is driven by GA with a tournament size of 5, crossover probability of 0.35, and mutation probability of 0.04. The evolution lasts ten generations, and each generation is populated with 40 MLP-based classifiers. Besides, the total number of trainable parameters is constrained to 20000.

In ten generations of architecture morphism, the MLP-based classifiers with the average *F1-score* of 0.96 have been derived for both PIS and MMPS simulation models (see Figure 7). It is essential to point out that for both cases, the variance of *F1-score* calculated using the 10-fold cross-validation is insignificant, which implies that the MLP-based classifiers were able to learn and generalize nonlinear input-output mapping (from the candidate-solution to the corresponding label). Due to the limited computational resource, the number of epochs is reduced to 60, which should not affect the grist of the learning process. Figure 8 demonstrates how accuracy, precision, and recall are improving during the learning process.

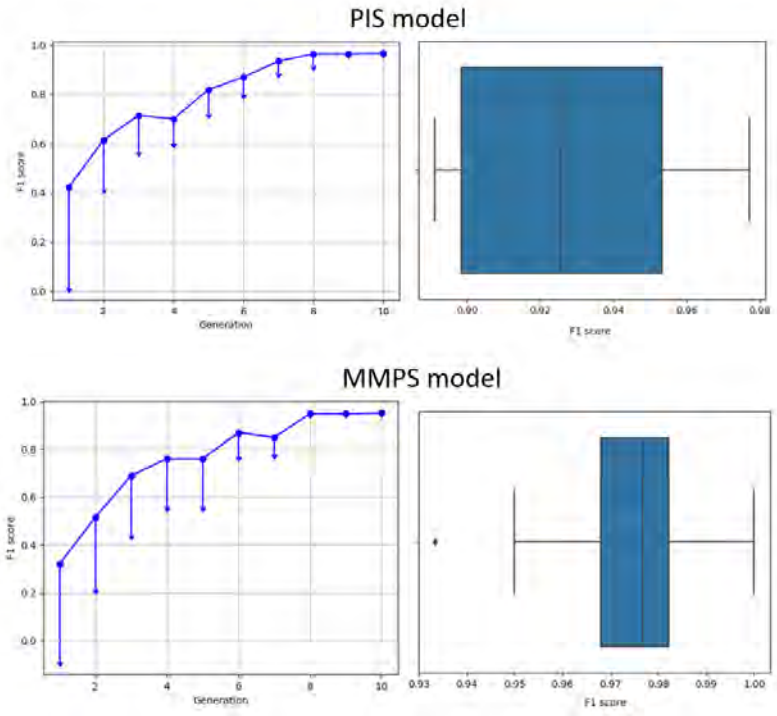

Figure 7: Evolution of the fittest classifiers.

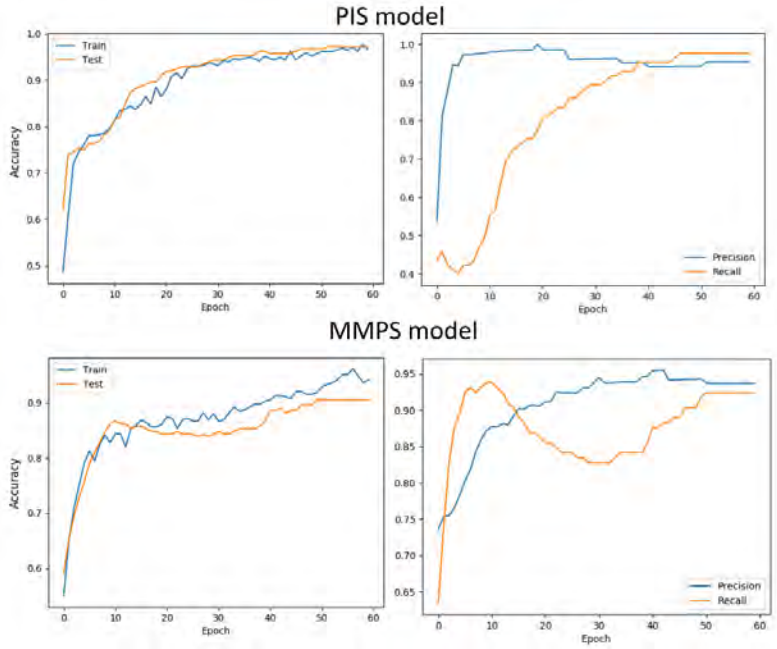

Figure 8: Accuracy, precision, and recall are improving as the classifiers learn.

Table 2 contains the specification of architecture and hyperparameters of the fittest MLP-based classifiers derived during the AutoML procedure.

Table 2: Architecture and hyperparameters of the fittest classifier.

Models	Depth	Width	Optimizer	Learning rate	Activation functions	L1	L2	Dropout
PIS	2	60	Adamax	0.004	SELU	0.005	0.002	0.05
MMPS	2	70	Nadam	0.002	SELU	0.004	0	0.05

4 DISCUSSION AND POTENTIAL LIMITATIONS

Paying attention to the components underneath the fittest classifiers, it should be emphasized both models have several important things in common. First, architectures of both MLP-based classifiers are relatively shallow (2 hidden layers). Second, both classifiers take advantage of Adam-based optimizers, emphasizing the importance of momentum in the learning process. The success of the SELU activation function can be explained by its property of self-normalization, which extirpates vanishing gradients (Kim et al. 2020). Although artificial neural networks are robust to stochastic noise in data, the MLP-based classifier will inevitably have an upper bound for accuracy. Nevertheless, the amount of noise can be reduced by increasing the number of replications of a DES model, which leads to a standard trade-off between accuracy and computational efficiency. It is also important to mention that the total number of parameters (neurons and weights) can be constrained. In other words, a potential user of the framework can decide how compact and computationally fast the resulting classifier should be.

It is also important to point out that both components within the proposed AutoML framework are strongly associated with the so-called "No Free Lunch" theorem. According to the theorem, there cannot exist an algorithm for solving all possible problems that is, in an average case, superior to any competitive one. This statement is true for both optimization driven by GA and supervised learning performed by MLP-based classifiers (Wolpert and Macready 1997). In this regard, in the general case, the question of whether the proposed approach is inferior or superior to any alternative has no sense. However, the following can be highlighted as advantages of the proposed framework:

1. Automatism. Such labor-intense and time-consuming steps as sampling, labeling, neural architecture search, hyperparameter optimization, and performance evaluation are fully automated within the proposed framework and do not require the participation of human experts.
2. Parallelism. The essential components of the proposed framework GA and MLP-based classifiers are implemented in parallel so that computations necessary for genetic operators and learning can be distributed across different processors.
3. Universality. Both GA and MLPs are universal and task-unspecific. Namely, the universal approximation theorem justifies the universality of MLPs (Cybenko 1989), and Holland's schema theorem highlights the universality behind GA (Holland, 1975).

5 CONCLUSIONS

MLP-based classifiers obtained through the proposed AutoML procedure can learn and generalize complex nonlinear relations within the DES models of subsystems vitally important for food supply chains and classify a candidate solution as profitable or not. Although the net profit has been considered as an output of interest and subject of maximization, it is essential to emphasize that the cost function incorporates substantial fees for events associated with adverse environmental implications, for example, perished goods and overproduction.

The distinguishing features of the proposed AutoML framework include automatism, universality, and parallelism. In this regard, the framework can make advanced artificial neural network models accessible to domain scientists, which can be considered "democratization" of the field.

REFERENCES

Agrawal, A., Gans, J., and A. Goldfarb. 2018. *Prediction Machines: the Simple Economics of Artificial Intelligence.* Boston, Massachusetts: Harvard Business Press.

Back, T., Fogel, D., and Z. Michalewicz. 2018. *Evolutionary Computation 1: Basic Algorithms and Operators.* Boca Raton, Florida: CRC press.

Barton, R. 2015. "Tutorial: Simulation Metamodeling". In *Proceedings of the 2015 Winter Simulation Conference (WSC)*, December 6th-9th, Huntington Beach, CA, 1765-1779.

Blanning, R. 1975. "The Construction and Implementation of Metamodels". *Simulation* 24:177-184.

Byrne, M. 2013. "How Many Times Should a Stochastic Model Be Run? An approach based on confidence intervals". In *Proceedings of the 12th International conference on cognitive modeling*, July 12-16, Ottawa, Canada, 445-450.

Caruana, R., and J. Schaffer. 1988. "Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms". In *Proceedings of the 5th International Conference on Machine Learning*. San Mateo, CA, 153-161.

Chollet, F. 2018. *Keras: The Python Deep Learning Library.* Astrophysics Source Code Library.

Dunke, F., and S. Nickel. 2020. "Neural Networks for the Metamodeling of Simulation Models with Online Decision Making". *Simulation Modelling Practice and Theory* 99:1-18.

Hanin, B. 2019. "Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations". *Mathematics* 7(10):992.

Holland, J. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Michigan Press.

Hutter, F., Kotthoff, L. and J. Vanschoren. 2019. *Automated Machine Learning.* New York, NY: Springer.

Jackson, I. 2019a. "Simulation-Optimisation Approach to Stochastic Inventory Control with Perishability". *Information Technology & Management Science* 22:9-14.

Jackson, I. 2019b. "Neuroevolutionary Approach to Metamodeling of Production-Inventory Systems with Lost-Sales and Markovian Demand". In *Proceedings of the 19th International Conference on Reliability and Statistics in Transportation and Communication,* October 13th-16th, Riga, Latvia, 90-99.

Jackson, I. 2020. "Neuroevolutionary Approach to Metamodel-Based Optimization in Production and Logistics". In *Proceedings of the 20th International Conference on Reliability and Statistics in Transportation and Communication*, October 13th-16th, Riga, Latvia, 84-93.

Kim, D., Kim, J. and J. Kim. 2020. "Elastic Exponential Linear Units for Convolutional Neural Networks". *Neurocomputing* 406: 253-266.

Mazzawi, H., Gonzalvo, X., Kracun, A., Sridhar, P., Subrahmanya, N., Lopez-Moreno, I., Park, H., and P. Violette. 2019. "Improving Keyword Spotting and Language Identification via Neural Architecture Search at Scale". In *Proceedings of the INTERSPEECH*, September 15th-19th, Graz, Austria, 1278-1282.

Metainventory. 2021. GitHub Repository. https://github.com/Jackil1993/metainventory, accessed 30th July 2021.

Miller, B. and D. Goldberg. 1995. "Genetic Algorithms, Tournament Selection, and the Effects of Noise". *Complex systems*, 9(3):193-212.

Nezhad, A., and H. Mahlooji. 2014. "An Artificial Neural Network Meta-model for Constrained Simulation Optimization". *Journal of the Operational Research Society* 65:1232-1244.

Owoyele, O., Pal, P., Torreira, A., Probst, D., Shaxted, M., Wilde, M., and P. Senecal. 2021. "Application of an automated machine learning-genetic algorithm (AutoML-GA) coupled with computational fluid dynamics simulations for rapid engine design optimization". *International Journal of Engine Research* 22(7):2407-2764.

Wager, S., Wang, S., and P. Liang. 2013. "Dropout Training as Adaptive Regularization". In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, December 5th-10th, Lake Tahoe, Nevada, 351-359.

Wang, L., 2005. "A Hybrid Genetic Algorithm–neural Network Strategy for Simulation Optimization". *Applied Mathematics and Computation* 170(2):1329-1343.

Wolpert, D., and W. Macready. 1997. "No Free Lunch Theorems for Optimization". *Transactions on evolutionary computation* 1:67-82.

AUTHOR BIOGRAPHIES

ILYA JACKSON is a postdoctoral associate at MIT Center for Transportation and Logistics. Besides, he is an assistant professor in the Faculty of Engineering at Transport and Telecommunication Institute (Riga, Latvia). He earned his PhD in Telematics and Logistics from Transport and Telecommunication Institute. His research interests include supply chain management, applied machine learning, simulation, and metaheuristics. His email address is ilyajack@mit.edu.

JOSUÉ C. VELÁZQUEZ-MARTÍNEZ is an executive director of the Supply Chain Management Program and director of the MIT Sustainable Logistics Initiative. Additionally, he is a research scientist at the MIT Center for Transportation and Logistics, specialized in Logistics and Supply Chain Management in transportation, manufacturing, and retail industries. He earned his PhD in Industrial Engineering from Monterrey Tech, Mexico. His email address is josuevm@mit.edu.