Computers and Operations Research 119 (2020) 104941

Contents lists available at ScienceDirect

Computers and Operations Research

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

Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management

Babak Abbasi[a,*], Toktam Babaei[d], Zahra Hosseinifard[b], Kate Smith-Miles[c], Maryam Dehghani[a]

[a] College of Business and Law, RMIT University, Melbourne, VIC 3000, Australia
[b] Faculty of Business and Economics, The University of Melbourne, Parkville, VIC 3010, Australia
[c] School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, Australia
[d] School of Mathematical Sciences, Queensland University of Technology, Brisbane, QLD 4000, Australia

ARTICLE INFO

Article history:
Received 19 May 2019
Revised 16 March 2020
Accepted 16 March 2020
Available online 18 March 2020

Keywords:
Data mining
Large-Scale optimization
Machine learning
Perishable inventory management
Neural networks
CART
k-NN
Blood supply chain

ABSTRACT

Practical constrained optimization models are often large, and solving them in a reasonable time is a challenge in many applications. Further, many industries have limited access to professional commercial optimization solvers or computational power for use in their day-to-day operational decisions. In this paper, we propose a novel approach to deal with the issue of solving large operational stochastic optimization problems (SOPs) by using machine learning models. We assume that decision makers have access to facilities to optimally solve their large-scale optimization model for some initial and limited period and for some test instances. This might be through a collaborative project with research institutes or through short-term use of high-performance computing facilities. We propose that longer term support can be provided by utilizing the solutions (i.e., the optimal value of the actionable decision variables) of the stochastic optimization model from this initial period to train a machine learning model to learn optimal operational decisions in the future. In this study, the proposed approach is employed to make decisions on transshipment of blood units in a network of hospitals. We compare the decisions learned by several machine learning models with the optimal results obtained if the hospitals had access to commercial optimization solvers and computational power, and with the hospital network's current empirical heuristic policy. The results show that using a trained neural network model reduces the average daily cost by about 29% compared with current policy, while the exact optimal policy reduces the average daily cost by 37%. Although optimization models cannot be fully replaced by machine learning, our proposed approach while not guaranteed to be optimal can improve operational decisions when optimization models are computationally expensive and infeasible for daily operational decisions in organizations such as not-for-profit and small and medium-sized enterprises.

1. Introduction

Optimization has been the centerpiece of many machine learning algorithms and has been a catalyst in machine learning theoretical development. In turn, insights from machine learning (ML) in optimization are at a relatively embryonic stage and are yet to make a comparable impact (Xu et al., 2016). In this research, we aim to extend further the application of machine learning in optimization by challenging machine learning models to generate the solutions achieved from solving complex stochastic optimization models. Optimization under uncertainty is an appealing problem in both academia and industry (Powell, 2016). However, solving such problems in practice is often computationally expensive. Mathematical modeling has been the main approach to solve large-scale practical optimization problems over the last decades, with powerful commercial solvers now available.

Benefiting from this capability requires industries to be equipped with strong mathematical and optimization knowledge as well as appropriate commercial tools and computational power, creating significant costs in terms of computational and human resources, which are not always easy to justify, especially for some not-for-profit industries, small and medium-sized enterprises (SMEs), and startups. Further, even if organizations have the expertise and computational power to solve the optimization problem, the requirement of accessing near real-time solutions

* Corresponding author.
E-mail addresses: babak.Abbasi@rmit.edu.au (B. Abbasi), Toki.babaei@qut.edu.au (T. Babaei), zahra.h@unimelb.edu.au (Z. Hosseinifard), smith-miles@unimelb.edu.au (K. Smith-Miles), maryam.dehghani@rmit.edu.au (M. Dehghani).

https://doi.org/10.1016/j.cor.2020.104941
0305-0548/

can hinder organizations in implementing the optimization models. For instance, to make efficient decisions in blood inventory management or generate optimal schedules for operating theaters, hospitals would ideally be in a position to capitalize on recent research advances and advanced computational facilities (Dehghani et al., 2019; Freeman et al., 2015; Zhu et al., 2018). The unfortunate reality, however, is that budgetary limitations typically restrict continuing access to the expertise required to design and solve the mathematical models and access to the required computational facilities, leaving little choice but for such industries to rely on their intuition to make decisions.

Indeed, the main motivation presented in the literature for using machine learning in solving optimization problems is related to the required computational time of solving large-sized instances of optimization problems, where in practice the solutions are needed almost immediately (Larsen et al., 2018; Lodi et al., 2019). In a similar vein, this research is motivated by such computational considerations. In our case study, solving some instances of the blood transshipment stochastic optimization problem takes more than 20 min. using a commercial solver on a desktop computer for a network of four hospitals (Dehghani et al., 2019). The computational time may be even longer using non-commercial solvers. This makes the stochastic optimization model almost impractical to be used in daily operations because 20 min. delay in dispatching delivery cars is significant and becomes crucial in emergency situations. In addition, there may be cases in which there are many more hospitals in the network, which also increases the computational time of solving the stochastic optimization model. The proposed approach requires expertise in both machine learning and optimization in the development phase, but not in the daily implementation. During the development phase, the optimization model is solved by accessing supercomputer facilities via a short-term collaboration with consultants or research institutes to prepare the training data set. Thus, the machine learning approach enables non-research organizations to benefit from daily decision support more readily than if a traditional stochastic optimization problem was continually solved in daily operations.

In this study, we propose a methodology that uses predictive machine learning models to address the above issues in practice. Machine learning models aim to infer appropriate relationships for predicting dependent variables (outputs) in terms of independent variables (inputs). We train several machine learning models to generate solutions (outputs) of a stochastic optimization model (with inputs from model parameters), when trained on an exact optimization solver output. In stochastic optimization models, such as two-stage models, the first-stage decisions are immediately actionable, while the second-stage (scenario indexed) decisions help build an understanding of robustness but are not critical to the immediate action. Therefore, we train the machine learning models to generate the actionable decisions of a stochastic optimization model. For example, in the stochastic optimization model for the transshipment of blood units (with maximum shelf life of 11 days) between four hospitals, there are 136 actionable decisions (i.e. order and transshipment quantities for today) while considering 100 scenarios with a planning horizon of 7 days; this adds 95,200 ($136 \times 100 \times 7$) decision variables (as scenario indexed decision variables) into the model, which are not immediately actionable and are used to take into account the demand uncertainty for the current decision stage. Thus, the decision-makers only need to know today's decisions and the model will be run again the following day based on the state of the system after implementing today's decisions and realizing today's true demand. In addition, in such stochastic optimization models, several auxiliary variables are required to express the constraints. We analyze the performance of the most common machine learning models in terms of their ability to mimic the solutions obtained from a commercial solver, and then as an independent decision-making agent that reliably emulates the optimization model. We argue that this simulation–optimization framework (Deist et al., 2018) will alleviate the aforementioned challenges and will benefit industry by lowering the overall decision and operational costs compared with current empirical heuristic approaches.

In this study, four common machine learning models' classification and regression tree (CART), k-nearest neighbors (k-NN), random forest (RF), and multilayer perceptron (MLP) artificial neural network (ANN) are considered to learn the relationship hidden between the input parameters defining the optimization problem and the optimal value of its decision variables. These models have been frequently employed in many tasks such as parameter estimation (Abbasi et al., 2010; 2008) and forecasting problems (Londhe and Charhate, 2010). Moreover, they have been used extensively in the operations research literature as optimization methods themselves (see Smith, 1999 and Smith and Gupta, 2000). Maier and Dandy (2000) and Burke and Ignizio (1992) reviewed the applications of ANNs in forecasting and operations research fields. Smith (1999) surveyed the application of neural networks in combinatorial optimization problems. Paliwal and Kumar (2009) reviewed the applications of neural networks in various areas including engineering and manufacturing. Here, however, we must draw a distinction between neural networks used to find their own approximate solutions to optimization problems as a heuristic, and our aim to use them to predict the solutions of an exact (non-heuristic) solver.

Bengio et al. (2018) review recent attempts by both the machine learning and the operations research communities to use machine learning to solve combinatorial optimization problems. These attempts can be classified into five streams:

1. Machine learning providing additional information to assist solution strategies, such as learning which algorithm is likely to perform better on an instance of the problem before making an attempt to solve it. For example, Smith-Miles (2009) proposes a framework to formulate the algorithm selection problem as a learning problem and Kruber et al. (2017) use machine learning to estimate if applying a Dantzig–Wolf decomposition will be effective on a mixed-integer program before solving.
2. Machine learning accelerating the performance of an algorithm. For example, machine learning can be used to find an initial feasible solution or to facilitate branching/pricing decisions in the branch and bound algorithm (Lodi and Zarpellon, 2017; Vaclavik et al., 2018).
3. Machine learning improving the robustness of solutions. For example, improving the quality of solutions and the computational time of solving optimization models whose constraints are highly corrupted or noisy (Xu et al., 2016).
4. Machine learning to produce the optimal or close to optimal solutions of all or a subset of decision variables of a combinatorial optimization problem (Larsen et al., 2018). Our research falls into this fourth category.
5. Machine learning predicting the optimal objective value of a combinatorial optimization problem. For example, Fischetti and Fraccaro (2019) uses machine learning models to predict the optimal objective function of a wind farm layout optimization problem.

A comparison of the recent literature describing studies in terms of their inputs and outputs is presented in Table 1. Table 1 is comprehensive in terms of research in stream 4; however, for other streams, it only includes recent studies. Stream 4 contains several recent studies that apply machine learning algorithms to obtain the solution of some well-known combinatorial optimization problems, such as traveling salesman problem (TSP) and knapsack problem. Vinyals et al. (2015) use neural networks to find

Table 1
Comparison of recent literature applying machine learning to solve optimization problems.

Stream	Reference	Application	Inputs	Outputs	ML method
1	Kruber et al. (2017)	Classical combinatorial optimization problems	Instance characteristics (e.g., number of variables)	Recommended decomposition approach to be used	k-NN
2	Vaclavik et al. (2018)	Time-division multiplexing, Nurse rostering	Problem features	Upper bound for the iterations in branch and price algorithm	Feature selection and regression
3	Xu et al. (2016)	Linear program (LP) but not specific application	Noisy parameters of LP	Estimated important parameters	Principal component analysis (PCA)
4	Larsen et al. (2018)	Load planning problem	Parameters of the problems	Aggregated value of decision variables (tactical level decision)	Logistic/Linear regression, ANN
4	Mossina et al. (2019)	Multi-label classification problem	Parameters of the problems	Solutions of a subset of decision variables	SuSPen meta-algorithm
4	Lodi et al. (2019)	Facility location problem	Parameters of the in hand and reference problems	Solutions of a subset of decision variables	Tree, ANN, Logistic regression, Naive Bayes
4	Vinyals et al. (2015)	Sequence of points (applicable in convex hull, TSP, and Delaunay triangulations)	Coordinates of points	Sequences of points	ANN
4	Bello et al. (2016)	TSP, Knapsack problem	Coordinates of points, The value and weights of items	Sequences of points, List of items	ANN
4	**This paper**	Blood supply chain	Inventory levels	Transshipment and orders decisions	ANN, CART, RF, k-NN

the convex hull of a finite number of points, compute Delaunay triangulations and predict detailed descriptions of solutions for the TSP. Bello et al. (2016) use a neural network approach to achieve close to optimal solutions for a TSP problem with up to 100 nodes and for a knapsack problem with up to 200 items. Larsen et al. (2018) train a neural network to provide the solution of a stochastic optimization problem where the training data were obtained by solving the deterministic version of an integer linear model.

In terms of using machine learning in making operational decisions, our paper is closest in spirit to the work of Larsen et al. (2018). However, Larsen et al. (2018)'s approach predicts descriptions of the solution (i.e., the tactical decisions) and does not determine the value of operational decision variables in the model. For example, in the case of load planning problem, their proposed approach determines how many containers from each type will be utilized in the final solution but does not pinpoint how many of each type will be loaded to a specific railcar (railcars are not identical). The main reason for predicting the description of the solution rather than the solution itself is that their aggregated training data set is obtained from solving a deterministic version of the problem that assumes availability of all information. However, in practical scenarios, due to the stochastic nature of the problem only the values of some features are available at the time of decision-making. Thus, the machine learning models are trained to predict some descriptions of solutions on the basis of available features. Their proposed approach is relevant to the cases where tactical and operational planning problems are interrelated. However, in many operational decision-making situations, such as the blood inventory management problem considered in this paper, tactical decisions cannot be used to generate optimal operational decisions. Larsen et al. (2018) noted that solution of load planning, which is part of an online booking system, is required immediately, and solving the mathematical model, which may take several minutes, is not practical. Therefore, using machine learning to predict aggregate decision variables in an online booking system is considered more practical. In contrast, our approach provides detailed operational solutions as the training data set is obtained by solving the stochastic optimization model where the parameters of the training data set have a similar structure to those expected in the decision-making phase. Larsen et al. (2018) considered regression and neural network models, and left the comparison of their model with the "ground truth" stochastic optimization model for future research. In our study, we consider a wider range of common machine learning methods, and not only compare the results with the results of the optimization stochastic model but also compare the results with the empirical-based decisions used in practice. Furthermore, our application is different, and the development of a general framework has been motivated by the context of this practical case study.

Fischetti and Fraccaro (2019) used machine learning to estimate the optimal objective value of the offshore wind farm layout (deterministic) optimization problem. They used linear regression, neural networks (NNs), and support vector regression models (SVR) instead of the optimization model, which requires significant computational time, to obtain the potential value of a new site (taking about 10 hours to solve the optimization model) where many sites (e.g., 1000 sites) are needed to be compared. Mossina et al. (2019) used the SuSPen meta-algorithm as a machine learning tool to predict a subset of decision variables in the multi-label classification problem that are set as the parameters of the formulated mixed-integer linear optimization model. This step is called blocking decision variables. Then, the resultant smaller sub-model is solved using optimization solvers. Lodi et al. (2019) used a binary classifier and a regressor to predict whether a new instance of an optimization problem, derived from a reference instance, will share all or a part of its optimal solution with the solution of the reference instance. They translated this information into an additional constraint given to a solver, which accelerated solving the optimization model. They applied their proposed approach to the facility location problem and highlighted that such an approach can lead to good solutions in a reduced amount of time.

In terms of using machine learning to directly acquire the detailed solution, our approach is analogous to Vinyals et al. (2015) and Bello et al. (2016). However, the combinatorial optimization problems tackled in Vinyals et al. (2015) and Bello et al. (2016) were deterministic. Further, they only examine a neural network approach, while we utilize a variety of machine learning approaches and embed a simulation-based model selection component in our methodology to select the best method for practical use. To the best of our knowledge, machine learning models have not been used as a decision support tool to generate detailed decisions in a supply chain optimization model.

Our case study to illustrate the general framework is the case of a hospital blood transshipment problem in a network of hospitals. Each hospital must make daily decisions about the quantities of blood of various ages to order. This can be shipped from either a central blood bank (CBB) or surplus blood from other hospitals to meet expected demand for blood transfusion recipients. The optimal decisions must consider simultaneous minimization of oversupply (wastage) and undersupply (shortage), and minimization of the operational costs of ordering, transshipment, and holding of inventory. This is a complex stochastic problem of optimal inventory control with perishable items, where handling the stochastic modeling and the large-scale nature of the problem present challenges (see Goyal and Giri, 2001 and Bakker et al., 2012 for reviews).

Given the difficulties of this problem, it is not surprising that the standard industry practice is a practical heuristic policy, typically to order such that the total inventory is maintained each day at a multiple of the average daily demand (e.g., two or three times the average demand), with transshipment rules that require small hospitals to transship units of blood to larger hospitals once the units pass a certain threshold of residual shelf life. For example, units with remaining shelf life of 6 days may be transshipped to larger hospitals in the hope that they will be used before expiry. Dehghani and Abbasi (2018) investigated the above heuristic policy when there are two hospitals in the network to determine the best value of the threshold. Recognizing that this heuristic is far from optimal, however, Dehghani et al. (2019) have also modeled and solved the blood transshipment problem as a two-stage stochastic optimization model, with over 129,000 decision variables and more than 100,000 constraints for a four-hospital network, offering significant transshipment cost savings for optimal decisions. However, only 136 decision variables (i.e., the first-stage decisions) are implemented for each day as the operational decision and the remaining decision variables (i.e., second-stage decisions) are used to capture uncertainty in the model. In addition, Dehghani et al. (2019) reported that solving the developed stochastic optimization problem using the commercial solver for some instances may take more than 20 minutes on a desktop computer, which is not desirable in practice for daily decisions, especially considering the travel times for transshipment between hospitals in large geographically dispersed locations, such as the situation in Australia. Given that the implementation and maintenance of this powerful but expensive solution (in terms of cost and time) is beyond the reach of most not-for-profit industries, SMEs, and startups, it motivates the question of whether the developed optimization model could provide the kind of training data required to help machine learning models emulate their performance to provide a cheaper proxy decision-maker in the longer term. We propose a general framework to answer this question and provide evidence via our case study of whether data-driven models can support cost-effective decision-making in not-for-profit sectors. Given the current state-of-the-art model for optimal transshipment decisions in the blood supply chain is highly complex, the value of our proposed methodology will be well established by considering its application to this important case study.

This paper makes the following primary contributions:

- It proposes a general framework to use machine learning models as a proxy of large-scale optimization models (which are expensive to be available in practice) in day-to-day operations.
- It sheds light on the business value of predictions made by the machine learning models in mathematical modeling based decision-making. This offers some guidance as to whether deploying machine learning models is economically sensible.
- It applies the proposed methodology in the blood supply chain to make decisions on transshipment and orders.
- It compares the results of implementing the machine learning models with the case of implementing the optimization model and compares the results with the practiced policy based on intuition.
- It investigates the accuracy of four well-known machine learning models in making decisions on the quantity of blood order and transshipment.

The remainder of this paper is organized as follows. Section 2 presents the general methodology and algorithm. Section 3 presents the case study for blood supply chain management and the optimization solver, and describes the hospital data and model training data. Section 4 presents the results after discussing some implementation details of the machine learning models, including a comparison of machine learning models and their success in emulating the solutions of the exact optimization solver. The results also compare the long-term costs of using the machine learning models, compared with the exact solutions if commercial solvers and expertise were available, and compare the cost savings with the current hospital heuristic policy. Finally, Section 5 concludes the paper and discusses directions for future research.

2. Methodology

In this study, we aim to use machine learning models to closely emulate the optimal solutions from a complex solver for decision-making. For instance, in our case study, the machine learning models are expected to learn the relationship between hospital inventory levels and the optimized orders and transshipment quantities in a specific supply chain. Nevertheless, the proposed methodology is generic, and can be applied to any problem where a complex optimization model and solver has been developed and it is desired to replace the solver with a learned machine learning model that has been trained to output an optimized decision for a given input vector summarizing the optimization problem. In this study, we focus on cases where the problem is modeled as a two-stage stochastic optimization. A two-stage stochastic optimization in operational decision-making, considering $\Gamma = |\mathcal{T}|$ as the planning horizon, is defined as:

$$\min_{x,y^{\xi,t}} c^T x + \sum_{t\in\mathcal{T}} \sum_{\xi\in\Xi} p_\xi q_{\xi,t}^T y^{\xi,t} \tag{1}$$

$$\text{s.t.: } Ax \geqslant b$$

$$Q_{\xi,t} x + W_{\xi,t} y^{\xi,t} \geqslant d_{\xi,t}, \quad \forall \xi \in \Xi, \forall t \in \mathcal{T}$$

$$x \in X, y^{\xi,t} \in Y^\xi, \quad \forall \xi \in \Xi, \forall t \in \mathcal{T}$$

In equation (1), x is known as the first-stage decision variables and $y^{\xi,t}$ is called the second-stage decision variable, which is defined based on each scenario ($\xi \in \Xi$). Γ is the planning horizon; p_ξ is the probability of scenario ξ; c, $q_{\xi,t}$, $d_{\xi,t}$, $Q_{\xi,t}$ and $W_{\xi,t}$ are parameters of the model; X and Y^ξ define the domain of decision variables. The number of second-stage decision variables increases by the number of scenarios and the length of the planning horizon, and often the problem becomes large very quickly in many real applications with a large number of scenarios and long planning horizons. However, the number of actionable decision variables in such operational decision-making (i.e., the first-stage decision variables) is limited. Hence, it seems viable to design a mechanism to learn to produce the values of the first-stage decision variables of a two-stage stochastic optimization problem. This can be achieved by developing machine learning models that can be trained according to the optimal solutions generated from (1) to act later in the absence of model (1). The first sub-figure in Fig. 1 shows the input–output of a machine learning model for predicting the solution of an optimization model. The second sub-figure in Fig. 1 presents the

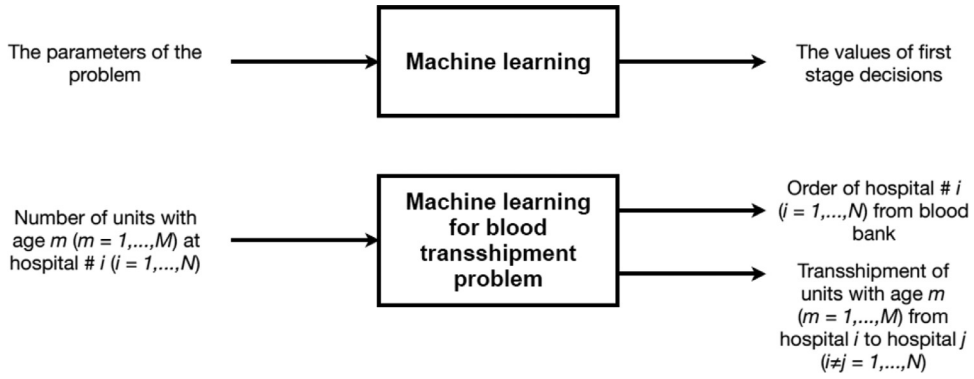

Fig. 1. Prediction of solution of an optimization problem via machine learning. The second sub-figure shows the input–output of the machine learning model in the case study in this paper.

Table 2
The mapping of decision variables and key parameters in Model (1) to the blood transshipment problem.

Decision variables and parameters in generic Model 1	Definition	Correspondence in Transshipment model	Relevance to machine learning (ML) models
x	First stage decision variables (actionable decisions)	The order quantity of each hospital and the transshipment quantities between hospitals in the current period	Used in developing the ML models;generated by ML models once trained
$y^{\xi,t}$	Second stage decision variables (scenario based decisions used to capture uncertainty)	The order quantity of each hospital in the next $\|\mathcal{T}\|$ periods under scenario ξ	Not used in ML models; ML models do not requires demand scenarios
$Q_{\xi,t}$, $W_{\xi,t}$ and $d_{\xi,t}$	Parameters of the model	Parameters to define the inventory flow, demand fulfilment and perishability of blood units under each scenario	Not used in ML models
c, p_{ξ} and $q_{\xi,t}$	Parameters of the model	c and $q_{\xi,t}$ are used to compute the holding, outdate, shortage and transshipment costs. p_{ξ} is the probability of each blood demand scenario	Not used in ML models
X and Y^{ξ}	Domain of decision variables	Domain of order and transshipment quantities which are positive integer numbers	Not used in ML models

input–output of the machine learning in the blood transshipment problem, which is the case study in this paper.

Table 2 shows the definitions of the decision variables and parameters of Model (1) in the blood transshipment problem. It also shows which variables are used in developing the machine learning models.

Algorithm 1 outlines the major steps in our proposed methodology. In our methodology, first, the optimization problem needs to be formulated. In many practical cases in operations management, the model will naturally be a two-stage or multistage stochastic integer linear program or a dynamic program because of the need to regularly rerun the model as the environment changes to obtain reliable operational decisions. After formulation, the optimization model is solved for a sample of instances to prepare the required training data. This training data set can be constructed using formal design of experiments methodologies or a simulation–optimization framework in a rolling horizon scheme. In the simulation–optimization framework, the model solves for each simulation iteration to determine the required decisions; then, the uncertainties are observed after forwarding the simulation to the system to the time of the next decision-making point. Therefore, the training data set consists of 1) the input parameters, which resemble what is known as the state of the system in the dynamic programming framework, and 2) the output (target) values, which are the corresponding solutions obtained from solving the optimization model. In the next step, the training data set is partitioned and used to develop machine learning models, such as ANN, k-NN, and CART, that can predict optimization solver solutions based on optimization input parameters. Then, in a laboratory setting (e.g., simulation–optimization framework), the models are used to make decisions, and the performance of each model is recorded on the basis of the effectiveness of implementing the model's decisions. For example, one can examine the average and standard deviation of system daily cost (or profit) when any of the developed machine learning models are used. Finally, the best model based on their performance in the simulation study is selected to be used in practice.

In the following subsections, we review the background of the candidate machine learning models before implementing their models for the case study.

2.1. Artificial neural network

ANNs are basically universal function approximators that learn the relationship between independent variables and dependent variables after executing a learning process from training data. The major difference between ANNs and statistical approaches such as regression is that ANNs make no assumptions about the statistical distribution of the data (Bishop, 1995), and therefore are more useful in many practical situations. Further, because of their nonlinear nature, ANNs perform very well in modeling complex data patterns where the functional form is nonlinear. ANNs are composed

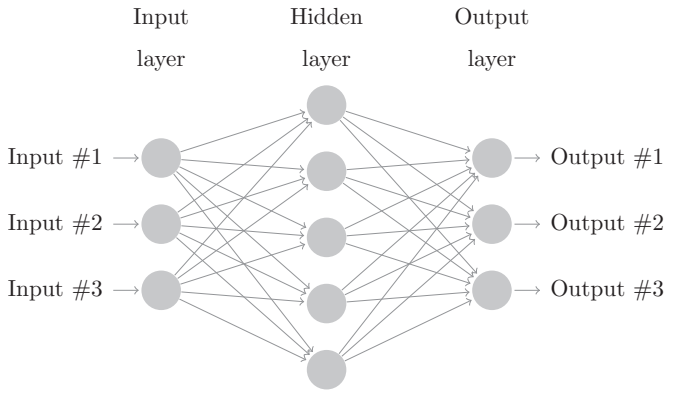

Fig. 2. An example of multilayer perceptron neural network structure with three inputs, one hidden layer, and three outputs. The bias input thresholds are not shown.

of processing units called neurons and can generally be divided into three taxonomies (multilayer feedforward networks, recurrent networks, and mesh networks) in terms of their architecture, that is, the way neurons interact within the network and its structure (Zhang, 2018). The MLP is one of the main types of ANN with a multilayer feedforward architecture in the sense that the output signal of any neuron in the MLP architecture is calculated merely using the current input signals from the outputs of neurons in the previous layer. This contrasts, for example, with so-called recurrent networks, which incorporate previous values of input and output signals (Graves et al., 2013). The MLP consists of an input layer, one or more hidden layer, and an output layer. Fig. 2 illustrates an MLP with one hidden layer. Data enter the network through the input layer, and are fed forward through subsequent layers. Neurons in MLP layers are connected to those in the successive layer by synaptic weights. Therefore, the sum of the weighted inputs is accumulated and passed to the activation function of the destination neuron. The typically nonlinear activation function, f, computes the output of the neuron:

$$o_j = f(\sum_{i=1}^{n} \theta_{ji} x_i - I_j) \tag{2}$$

where o_j represents the output of the $j-th$ neuron, which receives an n dimensional input vector $\mathbf{x}$ and corresponding weight vector θ_{ji} coming from the preceding layer, with an external bias threshold I_j. Sigmoidal, Gaussian, hyperbolic, and piecewise linear functions are the most common types of activation function. MLP training is the process to derive the inherent complex relationship between input and output variables by adjusting the weight values of the network such that the error function generated as the difference between desired output and network output becomes minimal. It has been theoretically shown that an appropriately trained MLP can approximate any nonlinear process to a high degree of accuracy (Bishop, 1995), although care must be taken to avoid overfitting training data and ensure generalization to unseen test data.

The MLP is usually trained by variations on the original back propagation (BP) method, which is based on gradient descent or conjugate gradient descent learning methods (Bishop, 1995). These methods rely on partial derivatives of the output with respect to the weights and bias values. The weights and biases are systematically adjusted to minimize the error between the predicted output and the desired output of the neural network. The mean squared error (MSE) is commonly used as the error measure.

2.2. Classification and Regression Tree (CART)

CART algorithms are used in many applications to predict both categorical and continuous variables. It divides the input data in branches and allocates predictions of response variables for each ending branch. The CART model is a tree framework than can be used for the prediction of new unbalanced data. The CART algorithm develops binary decision trees instead of using classification functions or network architectures to generate outputs. It constructs binary trees using the inputs and threshold that yield the largest information gain at each node (Han et al., 2011). The set of rules can then be used for prediction through the repetitive application of splitting (Ravi et al., 2008). In the CART, the training data are partitioned into smaller sets. Given an input sample, the tree building process starts at a root node considering all training data. Tree splitting proceeds when the training data cover different classes. In this case, two leaf nodes are introduced according to the most distinctive feature of the data samples. The same tree-splitting mechanism applies until a full decision tree is obtained (Seera et al., 2015). The features that help to achieve the "purest" levels of such separation are considered significant in their influence on the output variable, and end up at the root and closer to root levels of the tree. A number of splitting criteria such as Gini, Symgini, twoing, ordered twoing, and maximum deviance can be used in the CART, as explained in Pasiouras et al. (2005). The MSE is also frequently used as a splitting criterion. Suppose we are growing a binary tree; the algorithm will first select an attribute and find the most predictive value of the attribute to split the data into two subsets with minimal MSE in predicted (mean) values, each data set separately.

2.3. k-Nearest Neighbor

The k-NN is a non-parametric estimation method that has been applied in various estimation and pattern recognition problems. The k-NN rule estimates the output value of an input vector based on the outputs of k similar neighbors. A similarity measure is normally calculated using some kind of distance function. Various distance functions have been employed to develop k-NN algorithms, such as Euclidean distance and Mahalanobis distance. The output of a new sample is calculated by normal or weighted averaging of its k-NNs (Dudani, 1976).

2.4. Random Forest

Random Forest is an ensemble learning method, consisting of many individual decision trees. It consolidates the two concepts of bagging and random subspace by producing a set of N regression trees, where Bootstrap sampling from the original sample set has been used to select the training set for each tree and a random subset of the original set of features is considered for partitioning at each node. This helps to diminish the correlation between the generated regression trees, and therefore averaging their predictive results is relied upon to mitigate the variance of the error.

3. Case study in blood supply chain management

To illustrate our proposed methodology and evaluate its feasibility, we focus on the operational decisions in a blood supply chain. Effective blood supply chain management is critical, mostly due to the scarcity of the resource because the blood is sourced from donors, is perishable, and cannot be stored more than a certain period after donation. The main objective in managing inventories in blood supply chains is to match supply and demand in an optimum way to minimize wastage (known as outdates) and shortage of blood units. The uncertain demand and the perishable

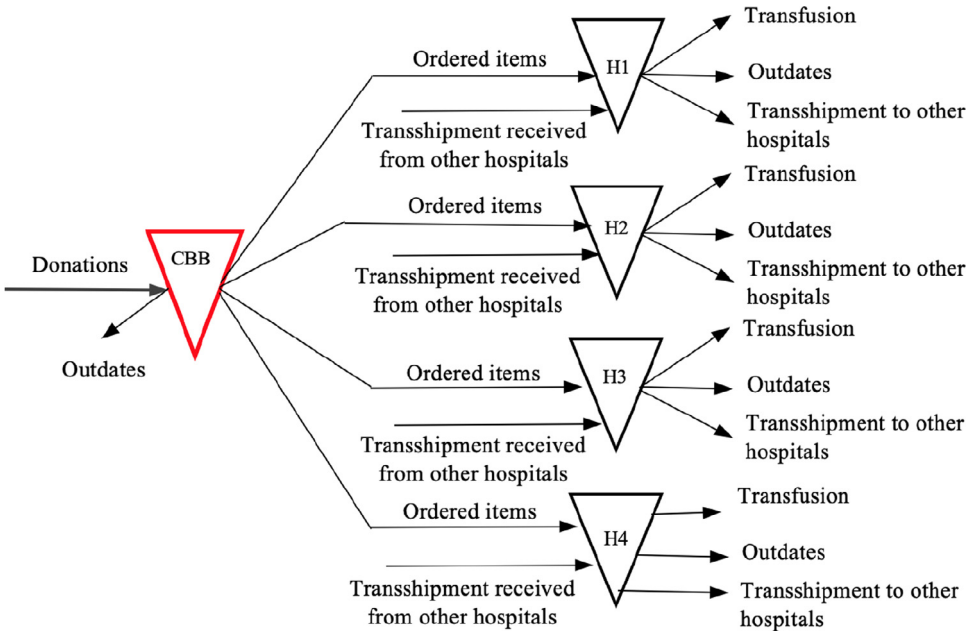

Fig. 3. A schematic demonstration of the blood supply chain network considered in this study. Each hospital maintains transshipment channels with three other hospitals.

nature of blood units can result in overstocking or understocking, which increases the wastage and shortage costs. An effective policy to improve the efficiency of a blood supply chain is known as transshipment (Hosseinifard and Abbasi, 2018). In such a policy, hospitals make decisions to transship some of their aging blood units to other hospitals in their network, while also placing orders with the cebtral blood bank (CBB) for fresh blood. The blood supply chain problem considered in this study consists of a network of four hospitals and a CBB, with a central decision-making coordinator. All blood units dispatched from the CBB to the hospitals have a fixed shelf life (Dillon et al., 2017). A CBB supplies blood units to a large number of hospitals; for example, New South Wales state CBB in Australia supplies blood units to more than 100 hospitals. However, the network of hospitals with agreement of transshipment between them is usually small (e.g., less than four hospitals) due to the hospitals' geographical locations and differences in their management systems (Abbasi et al., 2017). Fig. 3 visualizes the decisions and movement of blood in a network comprising four hospitals. To make transshipment and order decisions, hospitals need to either rely on some fixed empirical rules (which are typically far from optimal) or solve a large and complex optimization problem. In the latter case, the decisions are made for the entire network centrally. The decision variables of the model are how many items with what age need to be transshipped to which hospitals and how many units should be ordered from the CBB, considering the demand uncertainty. The above decisions need to be made such that the total cost of the blood supply chain is minimized. The total cost consists of ordering, transshipment, holding, outdate, and shortage costs.

3.1. Data set description

The data used in this study have been obtained from a case that consists of a network of four hospitals with uncertain demands. A central system makes daily decisions on the quantity of blood units to order from the central blood bank as well as on the quantity to be transshipped between the four hospitals, where both of these decisions have to be taken before realizing the demand of the day (i.e., in the morning of each day). Without loss of generality, we develop our model for management of O-positive blood group units only, and assume that blood units have a shelf life of 11 days as soon as they arrive at a hospital from the CBB, after which they must be discarded and are labeled as outdates (wastage). Therefore, the inventory of each hospital is described as an 11 valued array, where each element indicates the number of blood units of a particular residual shelf life. Based on the above description of the data, the input variable space of our modeling problem is 44 dimensional, where each set of 11 elements provides the current inventory levels of one of the hospitals. Conversely, the output space (decisions) includes the values of orders to be requested from the CBB for each hospital (4 values), and 132 ($11 \times 3 \times 4$) values of transshipment between the inventory of one hospital and the inventories of the three other hospitals.

The data required to train machine learning models to learn the relationship between the inventories and the optimized transshipment and order values have been collected from running the mathematical model developed by Dehghani et al. (2019), which is a two-stage stochastic programming model solved to optimality over a sequence of 18,500 days.

The demand for blood products at each hospital is assumed to be uncertain and varying from day to day. For this case study, the demand values of each hospital are the same as those used in Dehghani et al. (2019), which have been taken from a zero-inflated negative binomial distribution. The justifications for choosing this distribution for the simulation have been extensively explained in Dehghani et al. (2019). Moreover, the histogram of demands for each hospital is shown in Appendix A. Note that Hospitals 1 and 2 are smaller than Hospitals 3 and 4. Based on the aforementioned description, the data that we used for building our machine learning models include 44 input attributes and 136 output attributes for 18,500 daily observations or instances.

3.2. Implementation of methodology for the case study

In our case study implementation of the proposed methodology, four of the most-cited machine learning models (MLP neural network, CART, Random Forest, and k-NN) are selected to be trained to mimic the behavior of the transshipment optimization model developed by Dehghani et al. (2019). Each model is then used to make real-time decisions on a day-to-day basis. For evaluation of

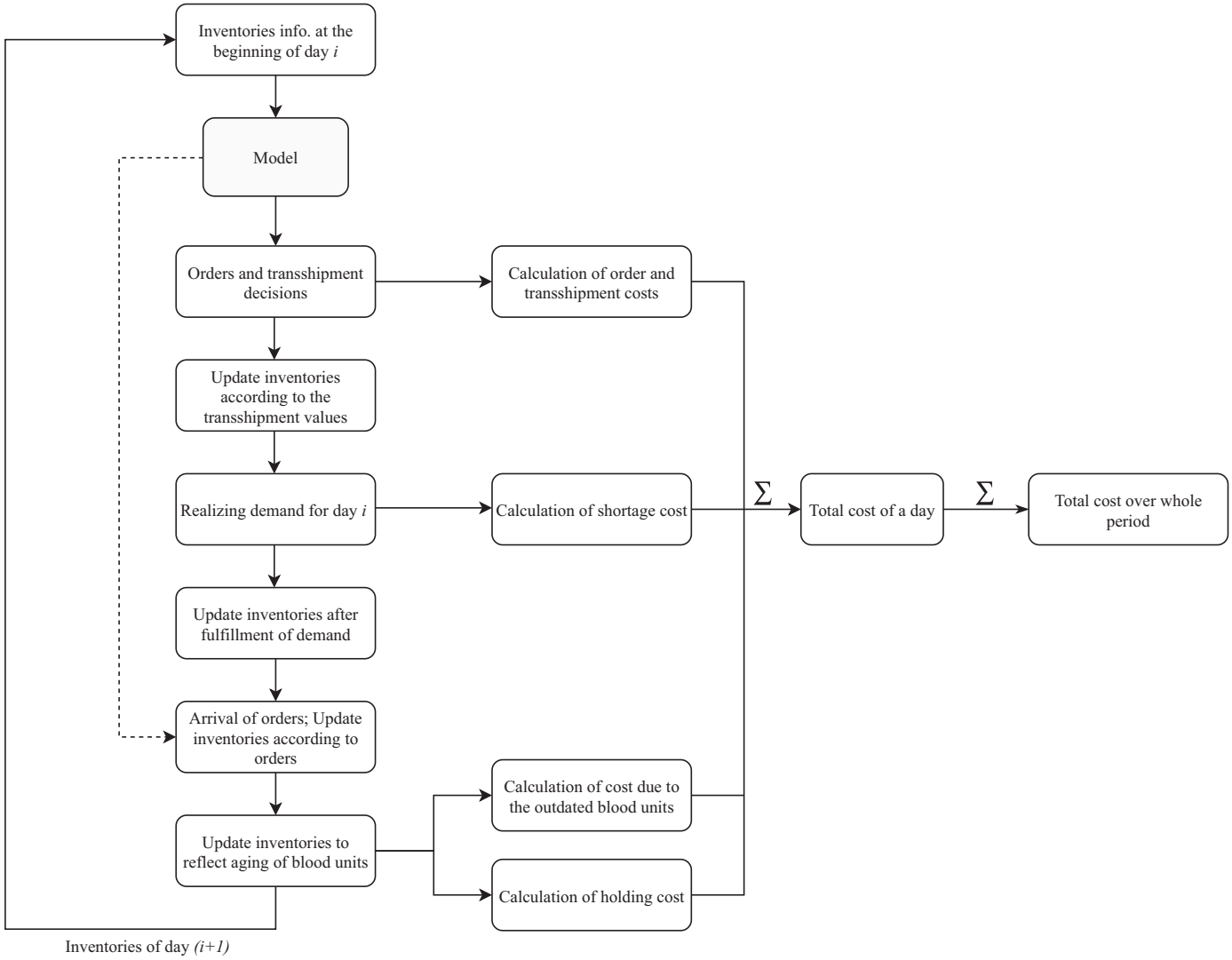

Fig. 4. Workflow diagram of simulating the blood delivery process using machine learning or TS models.

the usefulness of the models, total cost over an evaluation period can be calculated and compared with the industry's expectation and the cost of the other approaches. Details of each step including the development of the models and the simulation procedure now follow.

To evaluate performance of the models in practice, the developed machine learning models are used in a large-scale simulation study. In the case of the blood transshipment problem, the developed machine learning models generate the daily decisions on the number of blood units to be ordered as well as those to be transshipped to other hospitals in the network. The simulation is run over a period of 18,500 days, and the results of different machine learning models and those acquired from assuming availability of the optimization model are compared. The period of simulation was set according to the Dvoretzky–Kiefer–Wolfowitz (DKF) inequality (Kosorok, 2008), which computes the total number of simulations for estimating an empirical cumulative distribution of total daily cost with an error less than 0.01 with 95% confidence. Starting from the first day, the model must decide how many blood units to order from the CBB for each hospital, as well as how many blood units should be transshipped to other hospitals in the network, by examining the current inventory of the hospitals. Ordering and transshipment of blood units cost a certain value for the hospitals. Next, the inventory of each hospital is updated according to transshipped units, while the orders arrive at the end of the day. Then, each hospital realizes its demand and fulfills it using its available stock. At this stage, shortage cost will be calculated for each hospital. After that, the inventory is updated according to the arrival of orders. Finally, the inventory values are updated to reflect the aging of blood units. At this stage, the outdated blood units are discarded, and the corresponding wastage cost can be calculated. Further, the holding cost of the remaining units is also calculated. According to this procedure, each day is associated with a total cost value that consists of ordering and transshipment costs, shortage cost, and outdate cost. For evaluation of the decisions made by any decision-maker, the average of total costs and their standard deviation over the whole period are calculated. Fig. 4 represents the workflow of the whole process, where the MLP, k-NN, RF, and CART models (or any decision support model) are to be utilized as decision-makers.

3.3. Machine learning models

While the parameters of machine learning models are learned through training the models, their hyper-parameters are usually found empirically or based on prior knowledge about data. However, several works propose searching algorithms for finding the optimum value of hyper-parameters. In this work, we utilized a

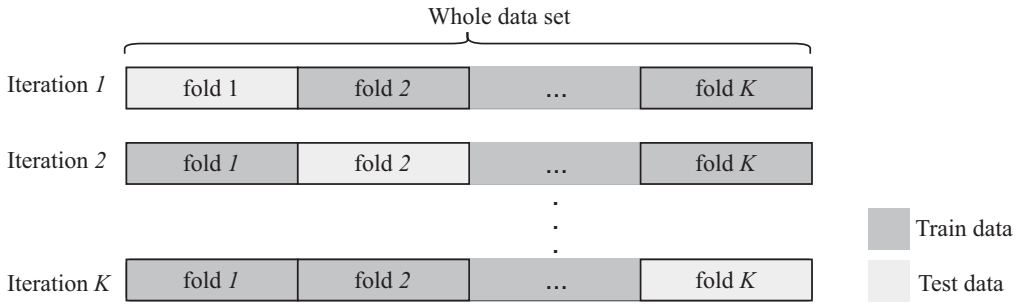

Fig. 5. Schematic representation of one cycle of *K*-fold cross-validation. The figure shows only one repetition; however, this will be repeated for several repetitions (e.g., 30) where in each repetition the folds are different as the data set is arbitrarily partitioned.

grid-search cross-validation algorithm (Hsu et al., 2003) to adjust important hyper-parameters of the machine learning models. A grid search basically establishes a full factorial design of experiments considering the given hyper-parameters. The performance of the model at each node of the grid is evaluated by running cross-validation over the entire training data set.

The cross-validation technique (depicted in Fig. 5) is commonly used in statistical and machine learning modeling and has a significant impact on the way the available data are divided to ensure good generalization of fitted models (Stone, 1974). It is particularly used to compare the generalization ability of different machine learning models (Burden et al., 1997), especially with limited training data or where there is concern of bias. In this study, a 10-fold cross-validation setup with 30 repetitions was established. This means that in each repetition the whole data set was randomly partitioned into 10 parts, and the model was iteratively trained according to nine parts (i.e., 90% of data) and tested using the remaining part (i.e., 10% of data), such that all data were used for testing once. The process was repeated 30 times to avoid bias (Bengio and Grandvalet, 2004). Specific implementation details for each of the machine learning models are now described.

3.3.1. MLP model

The input layer of the MLP comprises 44 neurons, through which start-of-day inventory values of the four hospitals were input. Two hidden layers were considered for our MLP model after a few rounds of trial and error, while the number of their neurons were found through the grid search and by further dividing train data to model and validation sets. The number of neurons in the output layer is 136, which corresponds to the orders and transshipment values. Moreover, the hidden neurons employed *tanh* function as their activation function, while the activation function of the output neurons was selected as the rectified linear unit (ReLU) function (Hahnloser et al., 2000). This function outputs 0 when its argument is negative, and otherwise returns the same value as its argument. Hence, the estimated value of each output neuron in the MLP network would be $\hat{y}$:

$$\hat{y} = max(0, o) \tag{3}$$

where o is the weighted sum of the activation values at the penultimate layer as described by Eq. 2.

The ReLU function was employed because the outputs of the problem (the transshipment and order values) can only take either 0 or positive values. To train the network (finding weights and biases; θ_i), the stochastic gradient based algorithm known as the Adam algorithm was adopted because it is a better choice for large data compared with the normal gradient descent algorithm (Kingma and Ba, 2014).

3.3.2. CART model

For the creation of an appropriate decision tree using the CART algorithm and avoidance of over-fitting, the best maximum depth value of tree for our problem is searched using the grid-search cross-validation algorithm. The splitting criterion is selected as the MSE across the entire data set, which is the appropriate choice for regression problems.

3.3.3. k-NN model

The normalized Euclidean distance was used to implement the k-NN model. According to the cross-validated grid-search results for optimized number of nearest neighbors, the k-NN with k = 7 was created as the final K-NN model.

3.3.4. Random Forest model

For optimization of the performance of the RF model, the number of decision trees in the forest and the maximum number of levels in each decision tree were determined using the grid-search cross-validation algorithm. According to the cross-validated grid-search results for the optimized number of trees and the best maximum depth of each tree, a model with 700 trees and a maximum depth of 25 was created.

In addition to the described details, some extra steps were considered in implementing all three models to guarantee that the generated transshipment and order values were feasible. It is worth noting that this issue was addressed in the mathematical model by imposing appropriate inequality and equality constraints (Dehghani et al., 2019). The steps were as follows:

- Because the actual values of transshipments and orders are integers, the outputs of the models were rounded to be integer values.
- The transshipment value given by each model was compared with the current corresponding inventory. If it was greater than the inventory value and the inventory value was not 0, the generated transshipment value was set to equal that inventory value. If the inventory value was 0, the transshipment value generated by the model was set as 0. The percentage of infeasible transshipment values requiring this feasibility repair correction was 3.4%, 3.9%, and 5.3% for the MLP, CART, and k-NN models, respectively.

Fig. 6 schematically depicts the overall procedure of training, evaluation, and implementation of the machine learning models for making orders and transshipment decisions. The detailed steps of the simulation and cost calculation are explained in the flowchart illustrated in Fig. 4. The total cost of the whole evaluation period and the mean and standard deviation of per day costs will be reported in the results that follow.

4. Results

4.1. Machine learning model comparison

For our MLP model, it was found, after a few runs of trial and error, that two layers of hidden neurons was adequate. Then, the

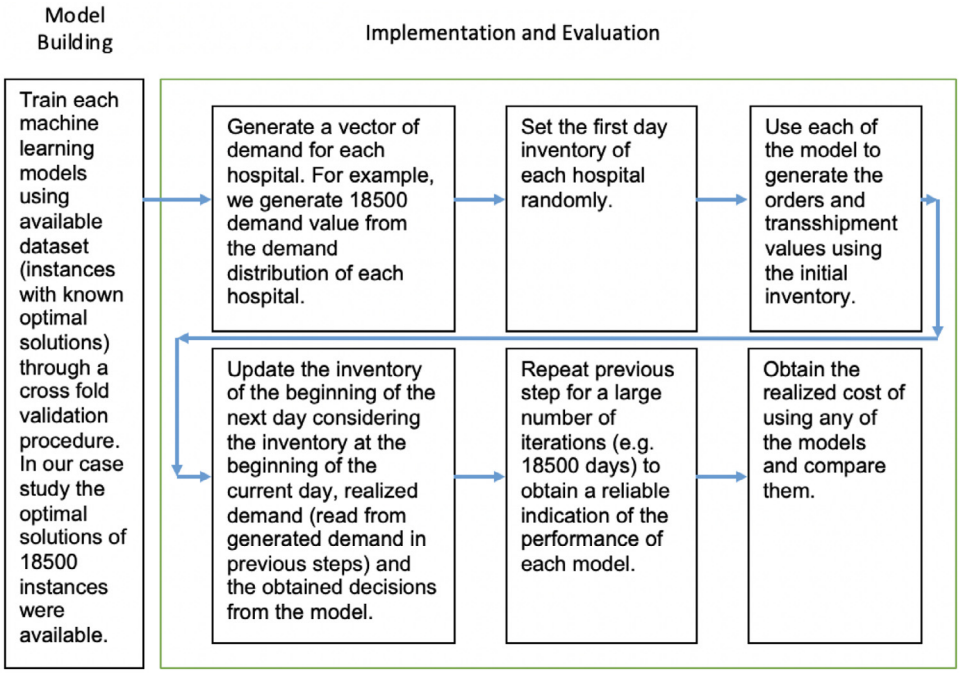

Fig. 6. The overall steps of training, implementation, and evaluation of the machine learning models of the transshipment problem.

Table 3
Grid-search cross-validation results of the MLP neural network model.

No. of neurons in hidden layers	Momentum	Mean of MSE	Standard deviation of MSE
200	0.3	0.01646	0.0030
200	0.5	0.01624	0.0027
200	0.7	0.01586	0.0021
200	1	0.01607	0.0031
300	0.3	0.01623	0.0023
300	0.5	0.01598	0.0023
300	0.7	0.01603	0.0030
300	1	0.01612	0.0028
400	0.3	0.01579	0.0030
400	0.5	0.01599	0.0029
400	**0.7**	**0.01570**	**0.0025**
400	1	0.01592	0.0026
500	0.3	0.01636	0.0030
500	0.5	0.01613	0.0029
500	0.7	0.01642	0.0025
500	1	0.01620	0.0026

number of hidden neurons and the momentum value were optimized through the grid-search cross-validation. Table 3 shows the results obtained during the search. From this table, increasing the number of hidden neurons improved the performance of the MLP model; however, increasing the number of the neurons to greater than 400 yielded only small increases in the average MSE value. Deterioration of MLP model performance can be due to overtraining of the network, because increasing the number of hidden neurons increases the complexity of the model and a complex model can be over-fitted and lose its generalization ability.

In the case of the k-NN model, the Euclidean distance implementation was utilized, while the number of neighbors, (i.e., k-value) was optimized through the grid-search cross-validation. Results of the grid search are summarized in Table 4. It is clear that $k = 7$ is the optimum value for the neighbors of our k-NN model. Similarly, Table 5 presents the grid-search results for the maximum depth of the CART. Based on the observations from this table, the tree with a maximum depth value of 10 yields the best (smallest) MSE.

To compare models, the 10-fold cross-validation results of three optimized models are considered. The same divisions into training and testing sets are adopted for all compared methods.

Table 4
Grid-search cross-validation MSE results of the k-NN model.

No. of neighbors	Mean of MSE	Standard deviation of MSE
2	0.10987	0.007
3	0.09824	0.006
5	0.09025	0.006
7	**0.08921**	**0.006**
10	0.09018	0.006

Table 5
Grid-search cross-validation MSE results of the CART model.

Max depth	Mean of MSE	Standard deviation of MSE
5	0.28080	0.010
10	**0.21304**	**0.013**
20	0.25033	0.017
30	0.25192	0.021

This process is repeated 30 times for each model. MSE results are visualized by box plots in Fig. 7.

The box plots in Fig. 7 present the median value, interquartile range, and range of 30 outcomes obtained by each model. It can

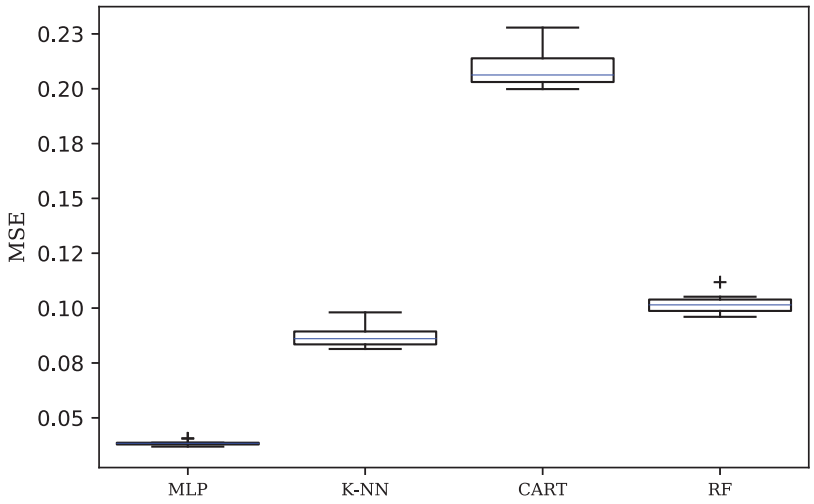

Fig. 7. MSE results of test data obtained during 10-fold cross-validation with 30 repetitions.

Table 6
Grid-search cross-validation MSE results of the RF model.

Max depth	Estimators	Mean of MSE
15	100	0.1062
20	100	0.1039
15	200	0.1056
20	**200**	**0.1034**
25	200	0.1035
20	250	0.1034

be seen that the MLP model surpasses the other competitor models, while the CART model results in the highest MSE value. This is not surprising as CART models generally do not perform as well as neural networks for nonlinear data (Curram and Mingers, 1994). Note that we did not need to run a statistical test as the results distributions of the different models do not overlap and their relative ranks are therefore clear.

4.2. Performance of models in practice

To evaluate the models in practice, the best MLP, k-NN, RF, and CART models obtained through the evaluation process are embedded in the blood supply chain process as depicted in Fig. 4. Table 7 summarizes the performance results in terms of the average and standard deviation of daily cost value of our models as well as the results obtained by three other models (Dehghani et al., 2019). The cost values are obtained by running the inventory system and using any of the models as a decision support tool for a period of 18,500 days. In the MLP model development, a 10-fold cross-validation technique is used to ensure generalization ability and to avoid model over-fitting. At the stage of building the models and applying the cross-validation technique, the models are evaluated and compared with regard to their MSE. To facilitate assessment of the models' generalization ability and procurement of unbiased results on the performance of the models in terms of the cost, the input data used to test the models are not the same as those used in development of the models. Although the same initial inventories (Day 1) and demand data have been used to obtain the average daily cost of all models (see Table 7), each model generated different order and transshipment quantities, which affect the inventory of the next day (fed as inputs of the models in the following day). The average cost reported in Table **6** is the average cost of the system over 18,500 days. For evaluation of the models, the mean and standard deviation of the total cost of the current empirical policy is taken as the baseline to be improved. Fig. B.2–B.4 visualize the key statistics of the inventories of the optimization model as well as those generated by each model. The following points can be deduced from these results:

- The No Transshipment policy leads to the poorest result and gives the highest total cost value. In other words, it is better to transship some units, even on the basis of intuitive policy as is the current policy, rather than totally ignore this opportunity for cost savings.
- The TS model (assuming access to the results of the optimization model) outperforms all other approaches in terms of average total cost. As mentioned earlier, this method involves accessing costly commercial software and requires the presence of professional staff to use it.
- The machine learning models could decrease the cost value of the process by around 29%, which is a significant reduction of cost.
- Although the age of the units in the inventory is not part of the objective function in the optimization model, because of the importance of issuing fresh units to hospitals (Abbasi and Hosseinifard, 2014), we compared the age profile of the inventory when the optimization model is used with that when the machine learning solution is applied. The results are presented in the appendix (B.2–B.4). They indicate that the machine learning models were successful in maintaining higher values of younger blood units compared with the optimization model's results, despite this not being a specific goal of either method.
- Finally, once an MLP model has been finalized and is being used to generate order and transshipment values, it obviously needs to be monitored carefully to verify the expected performance. Fundamentally, things can change.

Since the above results are obtained from a single case study, with a set of assumptions about Day 0 inventory levels and a certain distribution of demand patterns, we have also explored the generalization of the learned models in the face of variations to the demand set (randomly generated from the same distribution) and initial conditions (inventory on Day 0 at the hospitals). The mean and variance of the daily cost were the same as those reported in Table 7, showing that the model conclusions remain steadfast to the change in starting points and random demand generation. This implies that once these models are trained using an example set, they can perform well in new situations as long as the supply chain network and the demand distributions remain

Table 7
Cost results obtained by machine learning models, commercial package models (TS model and No Transshipment model), and the current policy.

Model	Mean of total daily costs	Standard deviation of total daily costs
MLP	84.95	27.74
CART	89.82	25.99
k-NN	85.95	21.90
RF	86.30	26.61
TS Model	74.51	22.05
Current Policy	119.49	25.31
No Transshipment	124.58	29.60

similar. Moreover, while training machine learning models might take time, once they are trained, they generate decisions for new situations almost instantly. In addition, using a trained machine learning model in new situations does not need special expertise.

5. Conclusion

This work showed the promising performance of machine learning models as decision-making agents in a blood supply chain network. We selected the four most-cited machine learning models to examine our idea that these models can be trained to be used in day-to-day operational decision-making where it is not possible for industries to run the large optimization models that provide the optimal decisions. All four tested models were able to successfully learn from a limited number of optimized solutions of a stochastic programming approach, and perform independently later. Particularly, the achieved results showed that the MLP model, once it has been primarily trained on an instance set of optimal solutions, can provide efficient decisions on order and transshipment quantities in a network of hospitals. The hospitals can use such models as a click-and-go software for their everyday usage of course, bearing in mind that they may not provide solutions as optimal as exact mathematical solutions.

One of the limitations of this study was testing the machine learning models on a limited set of data in the presented case study. Another limitation of our methodology is that the machine learning models need to be retrained if the network of hospitals or the demand distributions change. Future research should further examine the robustness of the results for suggesting the retraining time of the implemented machine learning model. This could be accomplished through integrating our methodology with a drift detection (or change point) methodology. Another suggestion for future research is to approach the problem from a meta-heuristics evolutionary-based optimization point of view and use the decisions obtained from machine learning models as the initial solutions of meta-heuristics or matheuristics algorithms. Meta-heuristics algorithms have been employed in several studies; however, we have not encountered any research applying meta-heuristics or matheuristics optimization in the supply chain of perishable products with the possibility of transshipment between supply chain's entities during our literature review. Finally, the framework proposed in this study can be applied to other applications, in particular, industrial operations management areas where decision support models need to be used in daily operations.

CRediT authorship contribution statement

Babak Abbasi: Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Toktam Babaei:** Methodology, Software, Writing - original draft, Writing - review & editing. **Zahra Hosseinifard:** Methodology, Software, Writing - original draft, Writing - review & editing. **Kate Smith-Miles:** Methodology, Validation, Writing - original draft, Writing - review & editing. **Maryam Dehghani:** Formal analysis, Software, Project administration, Writing - original draft, Writing - review & editing.

Acknowledgment

The authors are grateful to the review team and the area editor for providing constructive suggestions and comments.

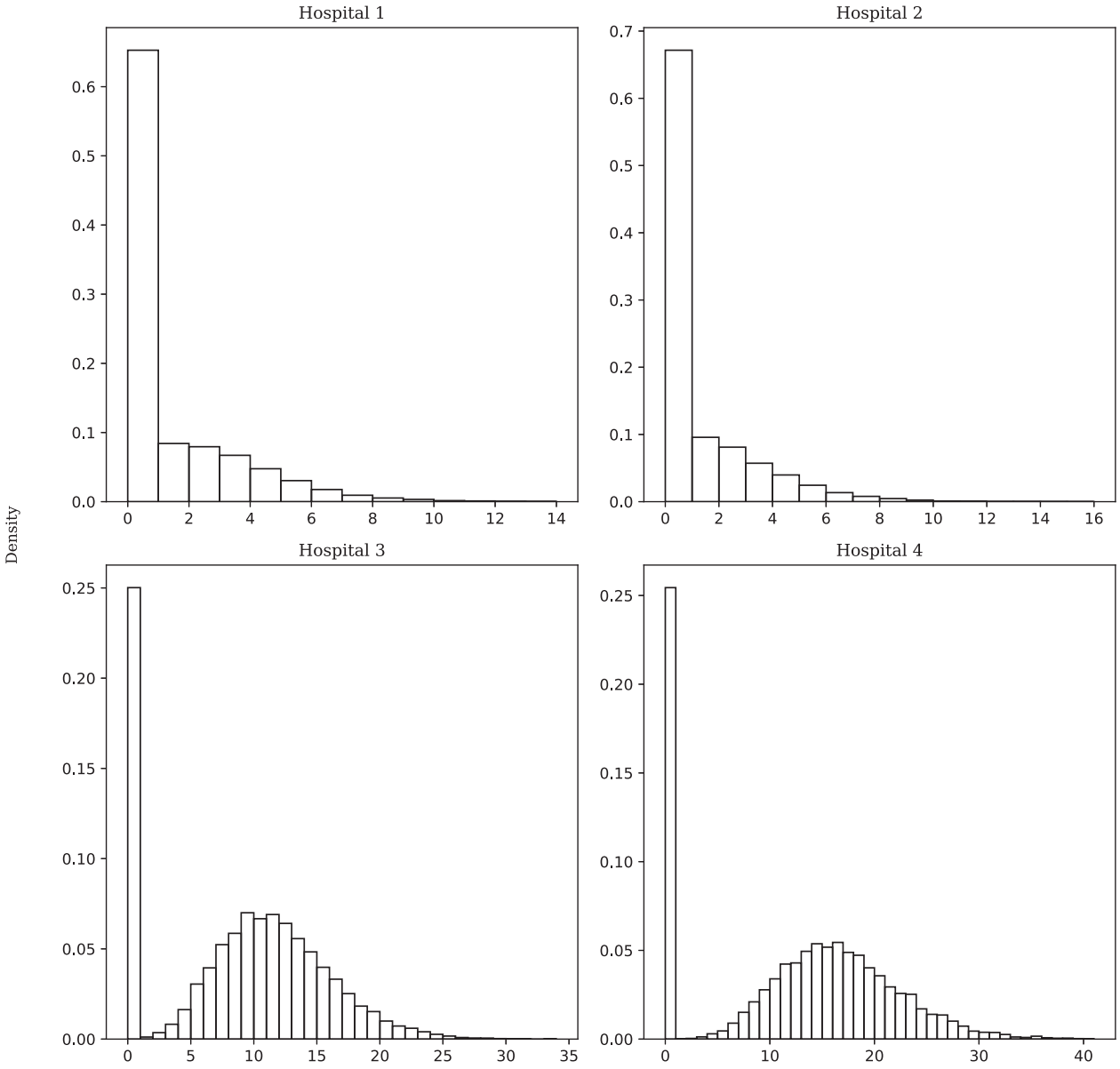

Fig. A1. Blood demand distribution over 18,500 days of four hospitals considered in this study.

Appendix A. The histograms of Blood demand data used to develop the machine learning models.

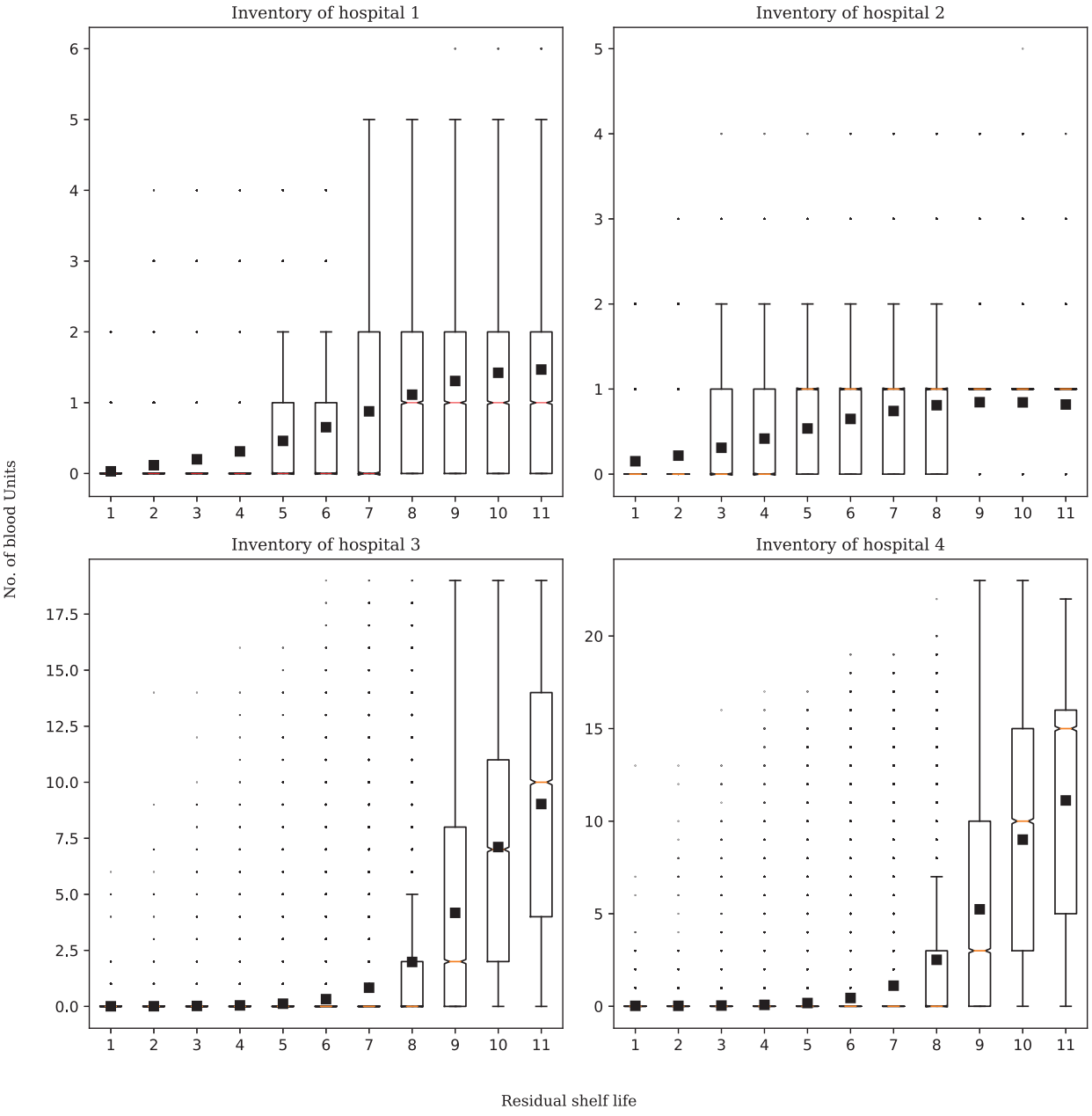

Fig. B2. Age distribution of the inventories of hospitals generated by optimized mathematical model.

Appendix B. Inventory profiles of the system at various hospitals for the cases that different models are used as the decision maker

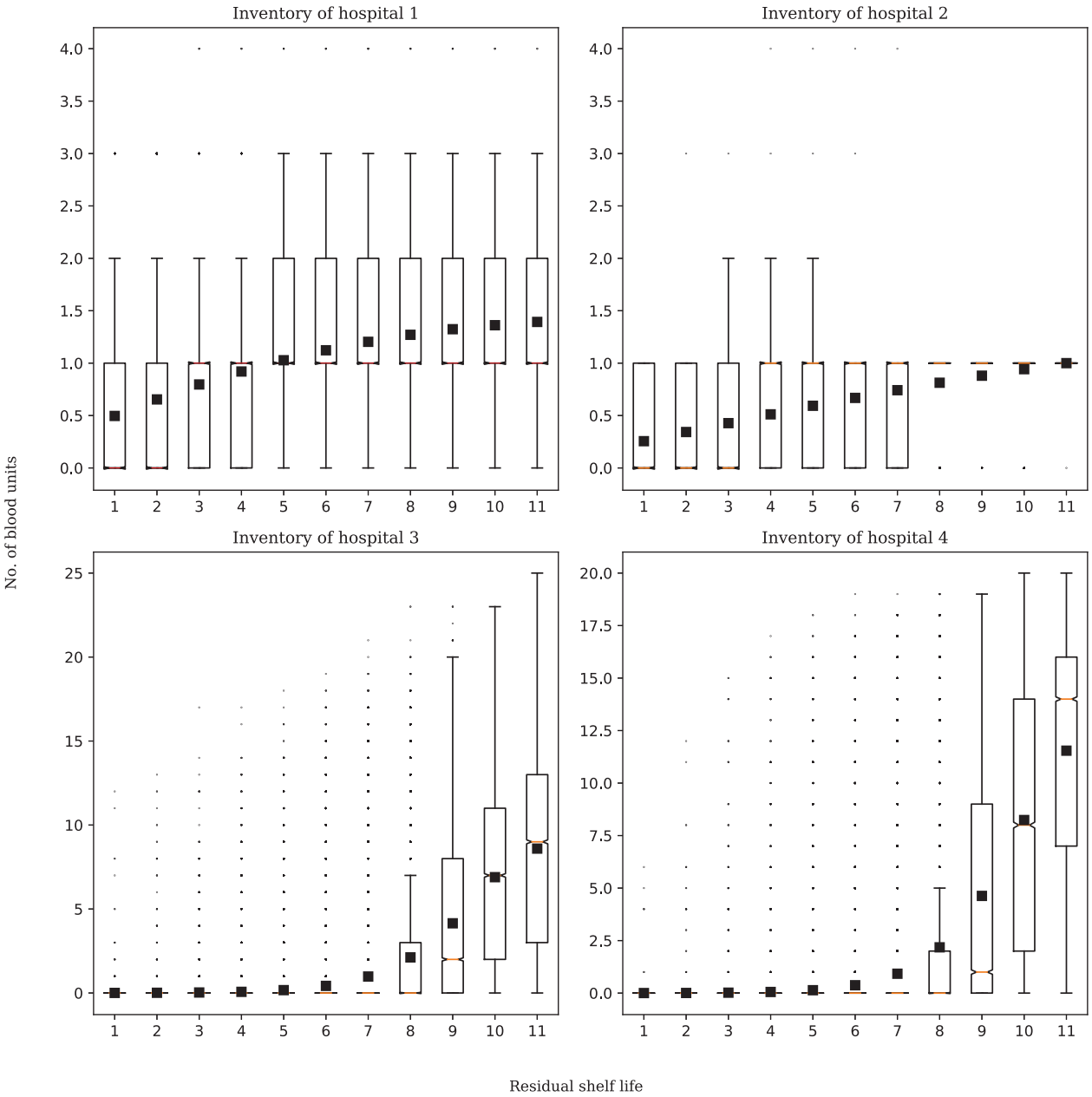

Fig. B3. Age distribution of the inventories of hospitals obtained by using MLP model as supply chain's decision maker.

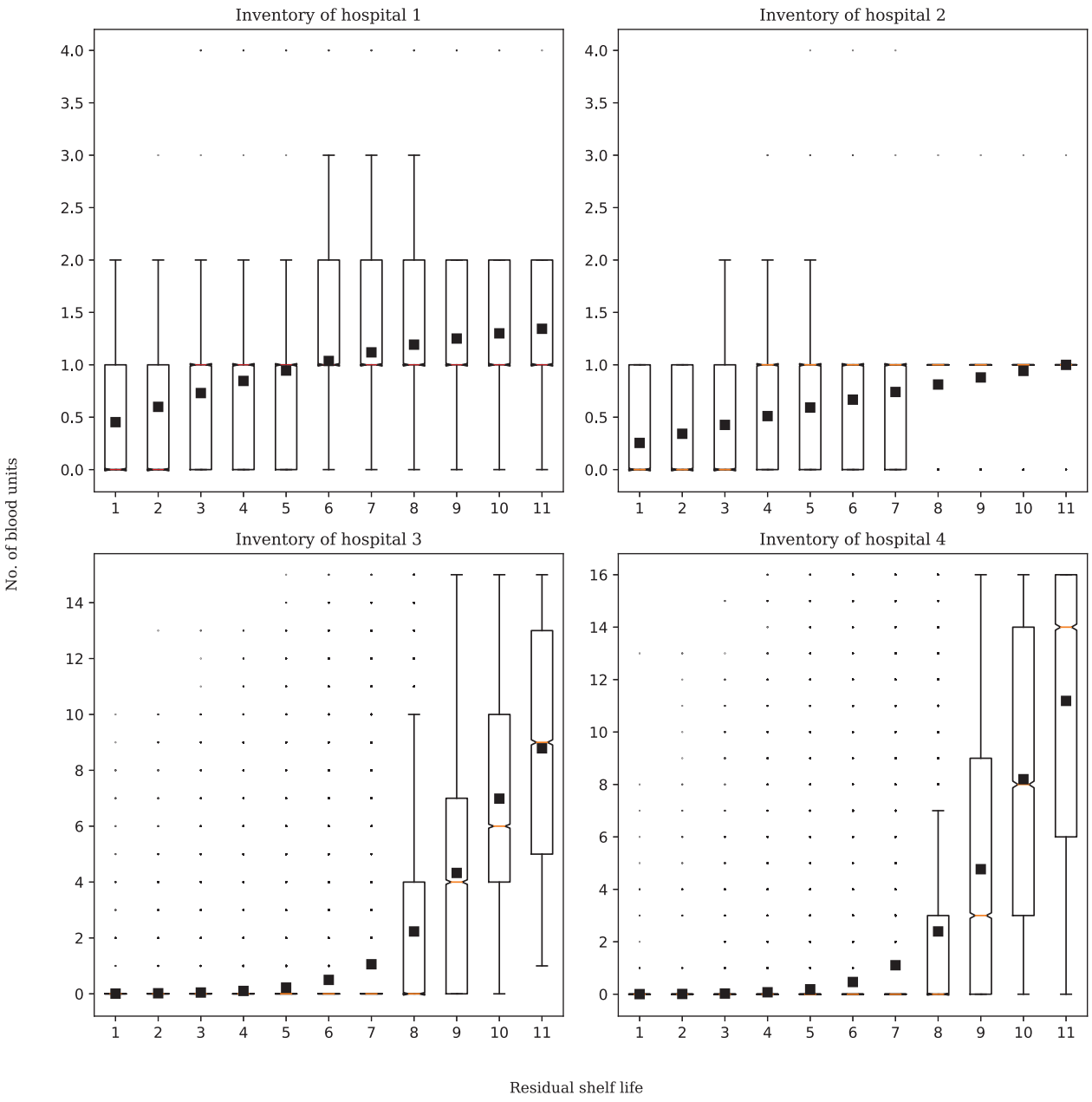

Fig. B4. Age distribution of the inventories of hospitals obtained by using CART model as supply chain's decision maker.

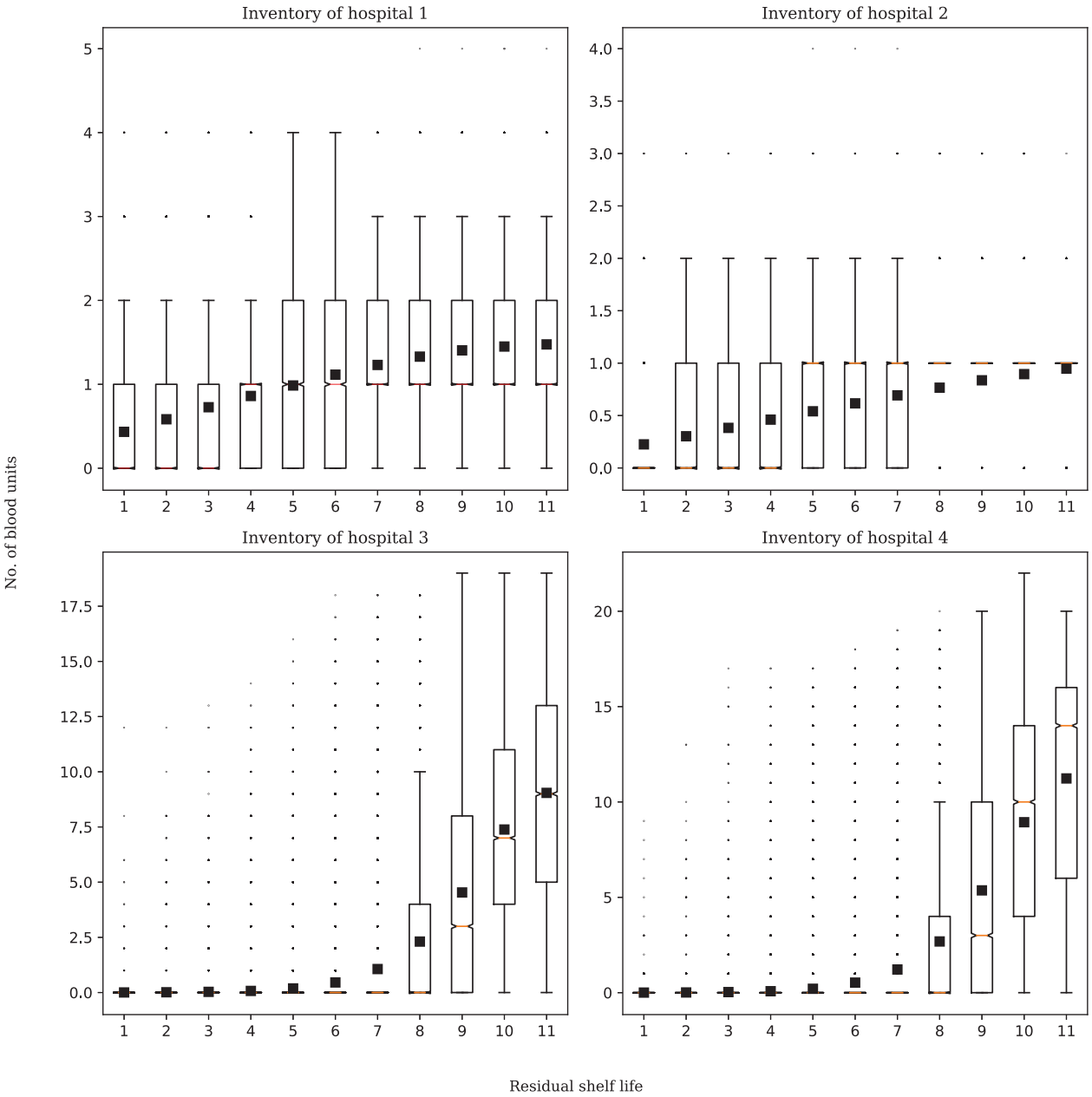

Fig. B5. Age distribution of the inventories of hospitals obtained by using k-NN model as supplu chain's decision maker.

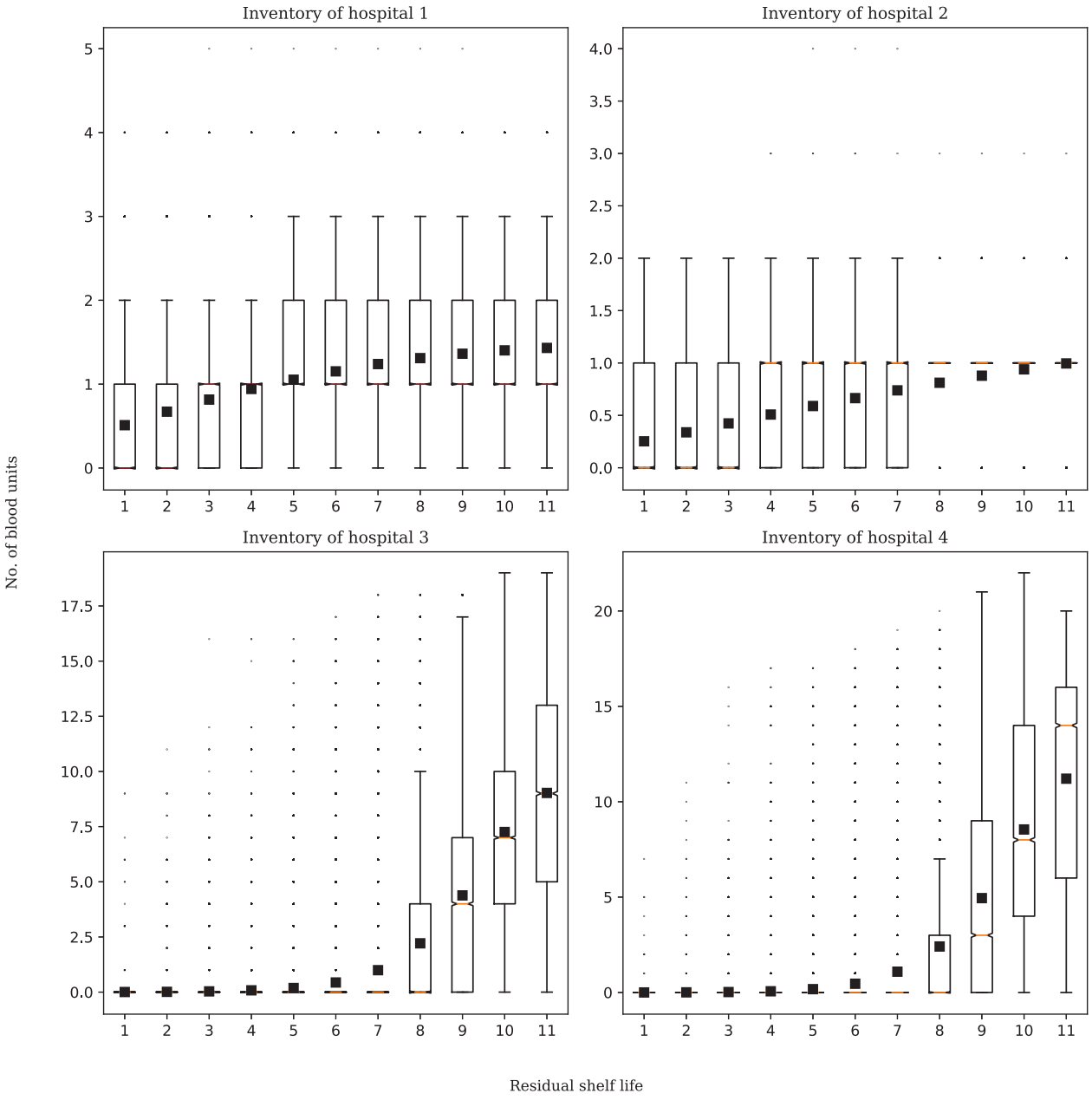

Fig. B6. Age distribution of the inventories of hospitals obtained by using RF model as supply chain's decision maker.

Algorithm 1: The pseudo code of developing machine learning models for large-scale optimization problems.

Initialization

Step 1. Formulate the optimization problem usingmathematical models (often the model will be a two-stage or multistage mixed-integer linear program (MILP) or a dynamic program).

Step 2. Solve the model with an initial setting using commercial solvers (and high-performance computing facilities if required).

Simulation and Data Preparation

Step 3. Build a simulation model that uses the solutions obtained from the mathematical model to evaluate the decisions in a rolling horizon approach.**Step 4**. Run the simulation model for a large number of iterations where in each run the required decisions are obtained from the mathematical model (developed in Step 2).**Step 5**. Save the parameters and the obtained solutions of the mathematical model (obtained in Step 4) in a data set and name it the reference data set.

Building Machine Learning Models

Step 6. Use the reference data set to build machine learning models. The parameters of the mathematical model in the references data set are inputs and the solution (the values of decision variables) is the output (target) of the machine learning models.
Note:In building the machine learning model, a portion of data is used for training and a portion is used for testing.

Model Selection

Step 7. Run the simulation model (built in Step 3) for a large number of iterations where in each run the required decisions are obtained from the developed machine learning models (in Step 6).

Step 8. Select the machine learning model that gives the best results in terms of the objective function of the problem. The average and standard deviation of the objective function value obtained in Step 4 can be compared with the corresponding values obtained in Step 7.

The Output

Step 9. Use the selected machine learning model in practice (e.g., in day-to-day operational decisions).

References

Abbasi, B., Hosseinifard, S.Z., 2014. On the issuing policies for perishable items such as red blood cells and platelets in blood service. Decis. Sci. 45 (5), 995–1020.

Abbasi, B., Hosseinifard, S.Z., Coit, D.W., 2010. A neural network applied to estimate burr xii distribution parameters. Reliab. Eng. Syst. Saf. 95 (6), 647–654.

Abbasi, B., Rabelo, L., Hosseinkouchack, M., 2008. Estimating parameters of the three-parameter Weibull distribution using a neural network. Eur. J. Ind. Eng. 2 (4), 428–445.

Abbasi, B., Vakili, G., Chesneau, S., 2017. Impacts of reducing the shelf life of red blood cells: a view from down under. Interfaces 47 (4), 336–351.

Bakker, M., Riezebos, J., Teunter, R.H., 2012. Review of inventory systems with deterioration since 2001. Eur. J. Oper. Res. 221 (2), 275–284.

Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S., 2016. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

Bengio, Y., Grandvalet, Y., 2004. No unbiased estimator of the variance of k-fold cross-validation. J. Mach. Learn. Res. 5 (Sep), 1089–1105.

Bengio, Y., Lodi, A., Prouvost, A., 2018. Machine learning for combinatorial optimization: a methodological tour d'horizon. arXiv preprint arXiv:1811.06128.

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford university press.

Burden, F.R., Brereton, R.G., Walsh, P.T., 1997. Cross-validatory selection of test and validation sets in multivariate calibration and neural networks as applied to spectroscopy. Analyst 122 (10), 1015–1022.

Burke, L.I., Ignizio, J.P., 1992. Neural networks and operations research: an overview. Comput. Oper. Res. 19 (3–4), 179–189.

Curram, S.P., Mingers, J., 1994. Neural networks, decision tree induction and discriminant analysis: an empirical comparison. J. Oper. Res. Soc. 45 (4), 440–450.

Dehghani, M., Abbasi, B., 2018. An age-based lateral-transshipment policy for perishable items. Int. J. Prod. Econ. 198, 93–103.

Dehghani, M., Abbasi, B., Oliveira, F., 2019. Proactive transshipment in the blood supply chain: a stochastic programming approach. Omega Online-published, 1–16.

Deist, T., Patti, A., Wang, Z., Krane, D., Sorenson, T., Craft, D., 2018. Simulation assisted machine learning. Working paper 1–15.

Dillon, M., Oliveira, F., Abbasi, B., 2017. A two-stage stochastic programming model for inventory management in the blood supply chain. Int. J. Prod. Econ. 187, 27–41.

Dudani, S.A., 1976. The distance-weighted k-nearest-neighbor rule. IEEE Trans. Syst. Man Cybern. 4, 325–327.

Fischetti, M., Fraccaro, M., 2019. Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. Comput. Oper. Res. 106, 289–297.

Freeman, N.K., Melouk, S.H., Mittenthal, J., 2015. A scenario-based approach for operating theater scheduling under uncertainty. Manuf. Serv. Oper. Manage. 18 (2), 245–261.

Goyal, S.K., Giri, B.C., 2001. Recent trends in modeling of deteriorating inventory. Eur. J. Oper. Res. 134 (1), 1–16.

Graves, A., Mohamed, A.-r., Hinton, G., 2013. Speech recognition with deep recurrent neural networks. In: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. IEEE, pp. 6645–6649.

Hahnloser, R.H., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S., 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature 405 (6789), 947.

Han, J., Pei, J., Kamber, M., 2011. Data Mining: Concepts and Techniques. Elsevier.

Hosseinifard, Z., Abbasi, B., 2018. The inventory centralization impacts on sustainability of the blood supply chain. Comput. Oper. Res. 89, 206–212.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al., 2003. A practical guide to support vector classification. Working paper.

Kingma, D.P., Ba, J., 2014. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kosorok, M.R., 2008. Introduction to Empirical Processes and Semiparametric Inference.. Springer.

Kruber, M., Lübbecke, M.E., Parmentier, A., 2017. Learning when to use a decomposition. In: International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer, pp. 202–210.

Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., Lodi, A., 2018. Predicting solution summaries to integer linear programs under imperfect information with machine learning. arXiv preprint arXiv:1807.11876.

Lodi, A., Mossina, L., Rachelson, E., 2019. Learning to handle parameter perturbations in combinatorial optimization: an application to facility location. arXiv preprint arXiv:1907.05765.

Lodi, A., Zarpellon, G., 2017. On learning and branching: a survey. TOP 25 (2), 207–236.

Londhe, S., Charhate, S., 2010. Comparison of data-driven modelling techniques for river flow forecasting. Hydrol. Sci. J.–Journal des Sciences Hydrologiques 55 (7), 1163–1174.

Maier, H.R., Dandy, G.C., 2000. Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environ. Model. & Softw. 15 (1), 101–124.

Mossina, L., Rachelson, E., Delahaye, D., 2019. Multi-label classification for the generation of sub-problems in time-constrained combinatorial optimization. https://hal-enac.archives-ouvertes.fr/hal-02120128/document.

Paliwal, M., Kumar, U.A., 2009. Neural networks and statistical techniques: a review of applications. Expert Syst. Appl. 36 (1), 2–17.

Pasiouras, F., Tanna, S., Zopounidis, C., 2005. Application of quantitative techniques for the prediction of bank acquisition targets, 5. World Scientific.

Powell, W.B., 2016. A unified framework for optimization under uncertainty tutorials in operations research. Tutor. Oper. Res. 45–83.

Ravi, V., Kurniawan, H., Thai, P.N.K., Kumar, P.R., 2008. Soft computing system for bank performance prediction. Appl. Soft Comput. 8 (1), 305–315.

Seera, M., Lim, C.P., Tan, S.C., Loo, C.K., 2015. A hybrid fam–cart model and its application to medical data classification. Neural Comput. Appl. 26 (8), 1799–1811.

Smith, K.A., 1999. Neural networks for combinatorial optimization: a review of more than a decade of research. INFORMS J. Comput. 11 (1), 15–34.

Smith, K.A., Gupta, J.N., 2000. Neural networks in business: techniques and applications for the operations researcher. Comput. Oper. Res. 27 (11–12), 1023–1044.

Smith-Miles, K.A., 2009. Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. (CSUR) 41 (1), 6.

Stone, M., 1974. Cross-validatory choice and assessment of statistical predictions. J. R. Stat. Soc. Series B (Methodological) 111–147.

Vaclavik, R., Novak, A., Scha, P., Hanzlek, Z., 2018. Accelerating the branch-and-price algorithm using machine learning. Eur. J. Oper. Res..

Vinyals, O., Fortunato, M., Jaitly, N., 2015. Pointer networks. In: Advances in Neural Information Processing Systems, pp. 2692–2700.

Xu, H., Caramanis, C., Mannor, S., 2016. Statistical optimization in high dimensions. Oper. Res. 64 (4), 958–979.

Zhang, Z., 2018. Artificial neural network. In: Multivariate Time Series Analysis in Climate and Environmental Research. Springer, pp. 1–35.

Zhu, S., Fan, W., Yang, S., Pei, J., Pardalos, P.M., 2018. Operating room planning and surgical case scheduling: a review of literature. J. Combin. Optim. 1–49.