Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 34(9): 6702–6733.
DOI: 10.3934/era.2026293
Received: 26 April 2026
Revised: 16 July 2026
Accepted: 21 July 2026
Published: 10 August 2026

Research article

An evaluation framework for integrated learning and optimization by revisiting parameters and hyperparameters

Bo Jiang, Xuecheng Tian* and Shuaian Wang

Faculty of Business, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong 999077, China

* **Correspondence:** Email: xuecheng-simon.tian@connect.polyu.hk.

Abstract: Integrated learning and optimization (ILO), an emerging paradigm for contextual optimization also known as "smart predict-then-optimize" or "end-to-end optimization", typically presents major challenges for practical applications in the community of operations research and management science (OR/MS). This study developed a general evaluation framework that systematically assesses the difficulty of implementing ILO for contextual optimization problems. It distinguishes between computationally tractable and challenging problems, offering the corresponding exact and approximate solution strategies to obtain the final prescriptions, respectively. Our main contributions are threefold. First, we identified conceptual ambiguities in traditional definitions of parameters and hyperparameters in machine learning (ML) models, proposing new complexity-based definitions that strengthen the theoretical foundations. Second, building on this, we developed optimization frameworks to identify the optimal ML models and prescriptions under sequential learning and optimization (SLO) and ILO, respectively. The ILO evaluation framework was then established, which is structured around several key questions regarding the specific characteristics of the ML model and the downstream optimization problem. Third, we illustrated the applicability of the proposed evaluation framework through a numerical case study on a stochastic binary decision-making problem and provided a cautious discussion of the observed numerical comparison. Moreover, we analyzed and compared the efficacy and efficiency of SLO and ILO. This study provides valuable insights and practical guidance for implementing ILO in solving real-world problems in OR/MS. Furthermore, it enriches the literature on the comparative effectiveness of SLO and ILO.

Keywords: machine learning; contextual optimization; sequential learning and optimization; integrated learning and optimization; evaluation framework

1. Introduction

In the fields of operations research and management science (OR/MS), the combination of machine learning (ML) and optimization techniques is commonly used to solve decision-making problems under uncertainty [1, 2]. Although the underlying probability distribution of uncertain parameters is unknown, historical data on these parameters and their associated covariates (features or contextual information) are typically available. In this situation, the contextual stochastic optimization (CSO) problem is introduced [3], where ML techniques are often used, based on available historical data, to predict/estimate relevant statistics of uncertain parameters, while the downstream optimization problem aims to prescribe actions/decisions for new feature observations.

In CSO problems, there are typically two primary paradigms for prescribing decisions, namely, sequential learning and optimization (SLO) and integrated learning and optimization (ILO). The SLO paradigm is also referred to as the "predict-then-optimize" framework [4], while the ILO paradigm is also described as the "smart predict-then-optimize" framework [5] or "end-to-end optimization". Both paradigms include a predictive component (where an ML model is first trained) and an optimization component (where a prescription is then made). However, SLO and ILO differ significantly in terms of how the ML model is trained on historical data. Specifically, the loss functions used in the training process for these two paradigms are distinct.

In SLO, the "predictive loss", often represented by the mean squared error (MSE), is used as the loss function, which is completely independent of the downstream optimization model. As a result, SLO is relatively straightforward to implement for real-world CSO problems. Despite its computational efficiency and widespread applications [6], SLO fails to capture the potential relationship between predictions and decisions. That is, ML models trained under SLO do not adequately consider the impact that predictions will have on the downstream optimization problem. Consequently, for a new observation, even a small predictive error can lead to significant deviations in the final prescription.

In contrast, in ILO, the "decision loss" serves as the loss function, which incorporates the downstream optimization model and associated decision costs, thereby addressing some of the limitations of SLO. This allows ILO to consider the underlying relationship between predictions and decisions, with the goal of maximizing the final prescriptive performance. However, the implementation of ILO faces several potential challenges, such as the discontinuity and non-convexity of the loss function [7] and the need to repeatedly solve the downstream optimization model during the ML model training process. These challenges limit the practical application of ILO. Existing ILO studies mainly focus on how to train a selected predictive model with a decision-aware loss, for example, by constructing differentiable surrogates or differentiating through an optimizer. Before such training is attempted, however, a practitioner still needs to know whether the candidate ML method and the downstream optimization model make the resulting ILO problem computationally tractable. Therefore, we primarily focus on how to evaluate the difficulty of implementing ILO for a specific CSO problem. This framework allows us to analyze in which scenarios resolving CSO problems with ILO is relatively straightforward and in which scenarios it becomes more difficult, while identifying the specific challenges. Furthermore, we present an outline scheme for handling both straightforward and difficult scenarios.

A fundamental aspect of our paper is to revisit and redefine the concepts of parameters and hyperparameters in ML models, particularly in the context of supervised learning. As mentioned

above, both SLO and ILO involve a predictive component, where an appropriate ML model must be determined. For a given ML method (e.g., decision trees), the process of determining a specific ML model is essentially the process of selecting appropriate values for the parameters and hyperparameters. Different values for parameters and hyperparameters result in different ML models. Therefore, from an anatomical perspective, for a specific CSO problem and dataset, the differences between ML models trained under SLO and ILO for a certain ML method are mainly the differences in parameters and hyperparameters.

Recognizing the critical role of parameters and hyperparameters, it is essential to first clarify their definitions in order to thoroughly analyze the optimality of ML models under SLO and ILO. Specifically, this analysis involves determining whether the parameters and hyperparameters achieve their optimal values under a specified loss function. Traditional perspectives suggest that hyperparameter values must be set prior to training, while parameter values are then learned from the training data (see Section 3.1 for details). However, we find that this definition, which seems to distinguish between parameters and hyperparameters based on the order of value assignment, is somewhat ambiguous, leading to an unclear and imprecise theoretical boundary between the two concepts. For ILO, this ambiguity is not merely terminological: it affects whether a configuration element should be treated as part of a complexity-realization search, a parameter-calibration search, or a learner-specific implementation choice. Consequently, it affects the structure of the optimization model used to identify the decision-aware ML model and the assessment of whether exact enumeration is feasible. As a result, the optimality of ML models under SLO and ILO is not supported by a clear and rigorous theoretical basis.

Therefore, we revisit parameters and hyperparameters, proposing more precise and rigorous definitions of these concepts from the perspective of model complexity. Broadly, model complexity refers to the extent to which an ML model can represent complex problems (see Section 3.2 for details). We propose that hyperparameters are the variables that govern the complexity of an ML model, while the remaining variables are parameters (see Section 3.3 for details). These refined definitions clarify the essential distinctions between parameters and hyperparameters. On this basis, we then establish optimization frameworks to identify the optimal ML models and prescriptions under SLO and ILO, respectively. Moreover, we establish our evaluation framework for ILO based on these new definitions.

1.1. Contributions

Our main contributions can be summarized as follows. First, we propose that the traditional definitions of parameters and hyperparameters (Definition 2) in ML models are somewhat ambiguous. To address this issue, we introduce novel definitions of these terms (Definition 4), which are based on the concept of model complexity (Definition 3). Building on these new definitions, we present optimization frameworks to identify the optimal ML models under SLO (Section 4.1) and ILO (Section 4.2).

Second, we propose a novel general evaluation framework (see Figure 1 for the flowchart) that systematically assesses the difficulty of implementing ILO in solving CSO problems. Unlike hyperparameter optimization methods that search for good configurations within a chosen training routine, our framework diagnoses the computational role of configuration elements before solving the ILO problem and indicates whether exact enumeration, restricted enumeration, or approximate search

is appropriate. This framework establishes an outline scheme to distinguish between computationally tractable and challenging problems under ILO, offering the corresponding exact and approximate solution approaches, respectively. Moreover, we demonstrate the use of the proposed framework through a numerical case study and confirm its applicability in the studied setting. This framework therefore offers valuable insights into the practical application of ILO to solve CSO problems.

Third, through numerical experiments in Section 6, we analyze and compare the prescriptive performance and solution efficiency of SLO and ILO in the context of a stochastic binary decision-making problem using synthetic data. Our findings indicate that the prescriptive effectiveness of SLO and ILO varies with different ML methods and training sample sizes, while the solution efficiency of SLO is superior to that of ILO in our experiments. The numerical study is intended to illustrate the framework and provide empirical evidence for the studied problem rather than to claim universal dominance of either paradigm. This contributes to the literature by enriching the discussion on the efficacy of SLO and ILO.

1.2. Related literature

This study is primarily related to the literature on two issues: (i) parameters and hyperparameters of ML models in supervised learning, and (ii) methodologies and applications of SLO and ILO to solve CSO problems.

1.2.1. Parameters and hyperparameters

As two essential concepts of supervised learning, parameters and hyperparameters not only form the basis of ML models in traditional predictive analytics but also influence prescriptive analytics research, which typically involves decision-making problems. As noted by Bischl et al. [8], the values of hyperparameters considerably affect the performance of ML models and must be carefully chosen. Most of the literature has focused on hyperparameter optimization and tuning [9–11]. This stream of literature provides powerful search strategies, such as grid search, random search [12], Bayesian optimization [13, 14], and multi-fidelity schemes [15], but usually takes the set of hyperparameters as given. A broader view of this stream shows several complementary research directions. For conventional regression and classification tasks, automated model selection and algorithm configuration frameworks, such as auto-WEKA and auto-sklearn, jointly search over learning algorithms and their hyperparameters [16, 17], and broad methodological reviews summarize the empirical behavior of hyperparameter optimization (HPO) methods across common ML algorithms [18]. Bayesian and surrogate-assisted HPO has been developed through sequential model-based optimization and related acquisition strategies [19–21], and it is often combined with bandit or multi-fidelity mechanisms for expensive learning pipelines [22]. Evolutionary and metaheuristic approaches have also been used to explore large or structured hyperparameter and architecture spaces [23]. For settings in which several criteria must be balanced simultaneously, multi-objective Bayesian optimization provides tools for recovering or targeting Pareto-efficient configurations [24, 25]. In deep learning, HPO is closely related to neural architecture search, including reinforcement-learning-based search and systematic surveys of architecture-search methods [26, 27]. Recent studies further show how rapidly this area is developing. Recent surveys and monographs provide updated views of AutoML and HPO foundations, systems, and open

challenges [28, 29]. Recent benchmarking work evaluates AutoML systems across many regression and classification tasks [30], while new algorithmic studies investigate simple yet competitive AutoML baselines [31], prior-guided HPO for deep learning [32], and in-context surrogate modeling for Bayesian optimization [33]. However, to the best of our knowledge, a clear and rigorous theoretical distinction between parameters and hyperparameters has rarely been discussed in the literature. Kuhn and Johnson [34] suggest that hyperparameter values must be set before training, while parameter values are learned from the training data. Similar viewpoints are presented in Probst et al. [35], Kumar and Garg [36], and Mishra and Silakari [37].

By examining these traditional definitions, we identify persistent ambiguities when distinguishing between parameters and hyperparameters. Therefore, we revisit and redefine parameters and hyperparameters from the perspective of model complexity. Model complexity concerns the capacity of an ML model to express complex problems [38]. Classical statistical learning theory uses complexity notions such as Vapnik-Chervonenkis (VC) dimension and Rademacher complexity to quantify the richness of a hypothesis class and to study generalization behavior [39, 40]. Our purpose is different: We use complexity as an operational criterion to classify configuration elements according to whether changing their values leads to ML models with different complexity levels. In this paper, we suggest that the concept of model complexity can enhance our understanding of the distinction between parameters and hyperparameters. The model complexity of several common ML methods has been extensively explored in the literature, including decision trees [41], logistic regression [39, 42], and deep learning models [43]. Notably, the definition and measurement of model complexity are typically contingent on the particular ML method of interest [44].

1.2.2. SLO and ILO

SLO has been widely investigated in the literature to solve CSO problems, primarily due to its computational efficiency [45]. Notable SLO methodologies include Kannan et al. [46], Bertsimas and Van Parys [47], and Ban and Rudin [48], among others. Furthermore, SLO has demonstrated significant real-world applications in diverse fields, such as logistics [49], power systems [50], healthcare [51], and production [52].

In contrast, the application of ILO faces limitations mainly due to the discontinuous and non-convex nature of the loss function [7], which poses major challenges in training ML models under ILO. The literature has proposed several approaches to address this issue, including training by unrolling [53], implicit differentiation [54], surrogate differentiable loss functions [5], and differentiable optimizers [55]. Nevertheless, real-world applications of ILO remain somewhat constrained to relatively straightforward problem contexts [56], with practical cases observed in domains such as last-mile delivery [57], maritime transportation [58], and ship inspection [59]. These studies typically start from a prespecified model class and then design a training mechanism for the decision-aware objective. Less attention has been paid to a prior diagnostic question [60]: For a given ML method, dataset, and downstream optimization model, which part of the ILO problem creates the computational bottleneck, and should the practitioner expect an exact or approximate prescription?

Although ILO mitigates certain limitations of SLO by incorporating the downstream optimization model, the efficacy of SLO versus ILO depends on the characteristics of both the studied problem and the available dataset [61, 62]. In other words, the final prescriptive performance (i.e., decision costs) of SLO and ILO should be analyzed and compared in the context of a specific CSO problem and a

specific dataset. Hu et al. [63] showed that SLO actually achieves faster regret convergence rates than ILO for contextual linear optimization. Elmachtoub et al. [64] demonstrated that ILO outperforms SLO when the ML model class is misspecified, while SLO tends to outperform ILO asymptotically when the ML model class is well-specified and data are sufficient. Vanderschueren et al. [65] compared SLO and ILO in the context of a cost-sensitive classification problem using real-world data, and the findings indicated that SLO is generally more effective than ILO. In this paper, we extend the literature by comparing SLO and ILO in the context of a stochastic binary decision-making problem (where, for ILO, we use our proposed evaluation framework). Our results reveal that their prescriptive effectiveness varies depending on the ML methods used and the size of the training samples.

1.3. Organization of the paper

The remainder of this paper is organized as follows. Section 2 outlines the basic concepts of supervised learning and provides a detailed description of SLO and ILO. Section 3 revisits and redefines parameters and hyperparameters. Section 4 establishes optimization frameworks to identify the optimal ML models under SLO and ILO. Section 5 proposes a novel general evaluation framework for ILO. Section 6 presents a numerical case study to illustrate the proposed evaluation framework. Finally, Section 7 concludes the paper.

2. Preliminaries

In this section, we first introduce several fundamental concepts of supervised learning in Section 2.1. We then formulate the CSO problem, along with two solution paradigms (i.e., SLO and ILO) in Section 2.2.

2.1. Supervised learning

Supervised learning primarily focuses on two essential components: (i) the target vector $Y \in \mathcal{Y} \subset \mathbb{R}^{d_Y}$, where $d_Y \in \mathbb{Z}_{++}$ denotes the dimension of Y, and the historical observed data $\{y_1, ..., y_N\}$, where $N \in \mathbb{Z}_{++}$; and (ii) the associated feature vector $X \in \mathcal{X} \subset \mathbb{R}^{d_X}$, where d_X denotes the dimension of X, along with the corresponding data $\{x_1, ..., x_N\}$, where x_i is observed simultaneously with y_i ($i \in \{1, ..., N\}$). Let $s_N = \{(x_1, y_1), ..., (x_N, y_N)\}$ be a set of N independent and identically distributed (i.i.d.) samples of the random pair (X, Y), and let $\mathcal{S}_N$ be the set of all such datasets of size N, so that $s_N \in \mathcal{S}_N$. Given a new observation x_0, the aim of supervised learning is to estimate the expected value of Y and provide a point prediction $\hat{y}_0$ based on the dataset s_N. To ensure clarity in subsequent discussions, we formally define several fundamental concepts of supervised learning as follows.

Definition 1 (ML method, ML model, and learner)**.** *Let $\mathcal{M}$ be the set of all ML methods, i.e., $\mathcal{M} = \{$linear regression (LR), k-nearest neighbor (kNN), decision tree (DT), neural network (NN), ...$\}$. For a given ML method $M \in \mathcal{M}$, let f_M be a specific ML model (e.g., a function), and let $\mathcal{F}_M$ be the hypothesis class of all possible ML models, such that $f_M \in \mathcal{F}_M$. An ML model f_M, also called a hypothesis, is a function that maps a feature observation x_0 to a point prediction $\hat{y}_0$, formally expressed as $f_M : \mathcal{X} \to \mathcal{Y}$, $x_0 \mapsto \hat{y}_0$. For a given ML method $M \in \mathcal{M}$, let L_M be a specific learner (e.g., a learning/model-fitting algorithm), and let $\mathcal{L}_M$ be the set of all potential learners, such that $L_M \in \mathcal{L}_M$. A learner L_M is an algorithm that maps a dataset s_N to a specific ML model f_M in the hypothesis class $\mathcal{F}_M$, formally*

written as $L_M : \mathcal{S}_N \to \mathcal{F}_M,\ s_N \mapsto f_M$.

For the definitions presented above, we provide the following additional clarifications. First, several ML methods serve as general terms that encompass a wide range of more specific methods. For example, DTs can be further divided into classification trees and regression trees, while NNs can be categorized into various types such as feedforward NNs, recurrent NNs, and convolutional NNs. Second, the term "optimal ML model" under method $M \in \mathcal{M}$ refers to the ML model that satisfies a predefined optimality criterion on a given dataset s_N, such as the lowest MSE on s_N. The term "optimal learner" under method $M \in \mathcal{M}$ refers to the learner that maps an arbitrary dataset s_N to a corresponding optimal ML model. In this paper, the optimal ML model is defined with respect to a given dataset s_N. Third, the concepts of "ML model" and "learner" presented in Definition 1 are further illustrated in Example 1.

Example 1 (LR)**.** For LR, the hypothesis class of all ML models is denoted by $\mathcal{F}_{\text{LR}} = \{\hat{y} = a^\top x + b \mid (a, b) \in \mathbb{R}^{d+1}\}$, where a and b denote the regression coefficients, and d ($d \in \mathbb{Z}_{++}$) denotes the dimension of the feature vector x (here, we consider a one-dimensional target for convenience). The set of all potential learners is denoted by $\mathcal{L}_{\text{LR}} = \{\text{least squares}, ...\}$. A learner in $\mathcal{L}_{\text{LR}}$ maps a dataset s_N in $\mathcal{S}_N$ to a specific ML model in $\mathcal{F}_{\text{LR}}$. For example, the least squares algorithm maps s_N to a specific ML model $f_{\text{LR}}^{\text{least-squares}} : \hat{y} = (a^*)^\top x + b^*$, where a^* and b^* denote the optimal regression coefficients determined by the least squares algorithm. As the $f_{\text{LR}}^{\text{least-squares}}$ model achieves the lowest MSE on s_N, it is referred to as the optimal ML model under the MSE criterion, and the least squares algorithm is referred to as the optimal learner.

2.2. CSO

In this study, we focus on CSO problems with a linear objective, where uncertainty (i.e., the target Y) appears linearly in the objective function. Let z be the decision variable constrained in $\mathcal{Z} \subset \mathbb{R}^{d_z}$, where d_z denotes the dimension of z, with the objective of minimizing the expected cost $\mathbb{E}[Y^\top z]$ (here, $d_Y = d_z$). We assume that the feasible region $\mathcal{Z}$ is a non-empty and compact (i.e., closed and bounded) set and that $\mathbb{E}[Y^\top z]$ is well-defined and finite for all $z \in \mathcal{Z}$. Let $\mathcal{D}_{x_0}$ be the underlying true conditional distribution of Y given $X = x_0$. The CSO problem of interest is then formulated as

$$\min_{z \in \mathcal{Z}} \mathbb{E}_{Y \sim \mathcal{D}_{x_0}}[Y^\top z] = \min_{z \in \mathcal{Z}} (\mathbb{E}_{Y \sim \mathcal{D}_{x_0}}[Y])^\top z, \tag{2.1}$$

where the optimal objective value represents the full-information optimum with full knowledge of $\mathcal{D}_{x_0}$ [4]. However, the underlying true conditional distribution $\mathcal{D}_{x_0}$ is generally unknown, and only the historical dataset s_N is available.

Based on s_N, both SLO and ILO aim to prescribe decisions for optimization problem (2.1), each comprising a predictive component and an optimization component. The predictive component typically consists of an ML model f trained on s_N, which maps a new feature observation x_0 to a point prediction $f(x_0)$. $f(x_0)$, which approximates $\mathbb{E}_{Y \sim \mathcal{D}_{x_0}}[Y]$, serves as input to the optimization component. The optimization component typically involves a downstream optimization model $[\mathbf{M}_{\text{ds}}]$:

$$v^*(y) = \min_{z \in \mathcal{Z}} y^\top z, \tag{2.2}$$

where y denotes the vector of target values. The prediction $f(x_0)$, derived from the predictive component, serves as input to solve the corresponding instance of model $\mathbf{M}_{\text{ds}}$ (i.e., optimization problem (2.2)), resulting in a final output prescription/decision $z^*(f(x_0))$ for observation x_0:

$$z^*(f(x_0)) \in \mathcal{Z}^*(f(x_0)) = \arg\min_{z\in\mathcal{Z}} [f(x_0)]^\top z. \tag{2.3}$$

In fact, if the ML model f could accurately predict the conditional expected value of Y, i.e., $f(x_0) = \mathbb{E}_{Y\sim\mathcal{D}_{x_0}}[Y]$, then the decision cost of the final data-driven prescription would be equal to the full-information optimum, i.e., $(\mathbb{E}_{Y\sim\mathcal{D}_{x_0}}[Y])^\top z^*(f(x_0)) = \min_{z\in\mathcal{Z}}(\mathbb{E}_{Y\sim\mathcal{D}_{x_0}}[Y])^\top z$.

Let $f(x_i)$ be the point prediction for x_i in s_N ($i \in \{1, ..., N\}$), and let $l(\cdot,\cdot)$ be a loss function that maps the pair $(f(x_i), y_i)$ to a non-negative real number, formally expressed as $l(\cdot,\cdot) : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_+$. Then, the loss functions and final prescriptions under SLO and ILO are formulated in Sections 2.2.1 and 2.2.2, respectively.

2.2.1. SLO

In SLO, the "predictive loss" is used as the loss function for an ML model f, denoted by $l_{\text{SLO}}(\cdot,\cdot)$, and is typically formulated as

$$l_{\text{SLO}}(f(x_i), y_i) = \frac{1}{2}\|f(x_i) - y_i\|_2^2, \quad i \in \{1, ..., N\}. \tag{2.4}$$

Notably, $l_{\text{SLO}}(\cdot,\cdot)$ is completely independent of the downstream optimization model. That is, the structure of model $\mathbf{M}_{\text{ds}}$ does not influence the loss function and thus has no impact on the training of the ML model [66]. Based on $l_{\text{SLO}}(\cdot,\cdot)$ in Formula (2.4), an optimal ML model f^*_{SLO} under method $M \in \mathcal{M}$ can be identified by solving the following optimization problem (recall that $\mathcal{F}_M$ denotes a hypothesis class of ML models under method $M \in \mathcal{M}$):

$$f^*_{\text{SLO}} \in \arg\min_{f\in\mathcal{F}_M} \frac{1}{N}\sum_{i=1}^{N} l_{\text{SLO}}(f(x_i), y_i). \tag{2.5}$$

Note that unless specified otherwise, f^*_{SLO} corresponds to a given method $M \in \mathcal{M}$. Given an optimal model f^*_{SLO} and a new observation x_0, SLO provides a final prescription $z^*(f^*_{\text{SLO}}(x_0))$ by solving the optimization problem as follows:

$$z^*(f^*_{\text{SLO}}(x_0)) \in \mathcal{Z}^*(f^*_{\text{SLO}}(x_0)) = \arg\min_{z\in\mathcal{Z}} [f^*_{\text{SLO}}(x_0)]^\top z. \tag{2.6}$$

In this study, $z^*(f^*_{\text{SLO}}(x_0))$ is referred to as an optimal prescription under SLO.

2.2.2. ILO

In ILO, the "decision loss" is used as the loss function for an ML model f, denoted by $l_{\text{ILO}}(\cdot,\cdot)$, and is typically formulated as

$$l_{\text{ILO}}(f(x_i), y_i) = \max_{z\in\mathcal{Z}^*(f(x_i))} \{y_i^\top z\} - v^*(y_i), \quad i \in \{1, ..., N\}, \tag{2.7}$$

where $\mathcal{Z}^*(f(x_i)) = \arg\min_{z\in\mathcal{Z}}[f(x_i)]^\top z$ and $v^*(y_i) = \min_{z\in\mathcal{Z}} y_i^\top z$ [5]. As multiple optimal solutions can exist in $\mathcal{Z}^*(f(x_i))$, the term "$\max_z\{\cdot\}$" indicates the selection of the solution that exhibits the worst-case performance, corresponding to the highest decision cost with respect to y_i. Notably, $l_{\text{ILO}}(\cdot,\cdot)$ incorporates the downstream decision cost. Based on Formula (2.7), an optimal ML model f^*_{ILO} under method $M \in \mathcal{M}$ can be identified by solving the following optimization problem:

$$f^*_{\text{ILO}} \in \arg\min_{f\in\mathcal{F}_M} \frac{1}{N}\sum_{i=1}^{N} l_{\text{ILO}}(f(x_i), y_i). \tag{2.8}$$

Note that unless specified otherwise, f^*_{ILO} corresponds to a given method $M \in \mathcal{M}$. Given an optimal model f^*_{ILO} and a new observation x_0, ILO provides a final prescription $z^*(f^*_{\text{ILO}}(x_0))$ by solving the optimization problem as follows:

$$z^*(f^*_{\text{ILO}}(x_0)) \in \mathcal{Z}^*(f^*_{\text{ILO}}(x_0)) = \arg\min_{z\in\mathcal{Z}}[f^*_{\text{ILO}}(x_0)]^\top z. \tag{2.9}$$

Remark 1 (An ILO problem and the optimal prescription)**.** In this study, Formulas (2.7)–(2.9) are collectively referred to as "an ILO problem". An ILO problem is considered to be exactly solved only when the optimal solutions to optimization problems (2.8) and (2.9) are obtained, with $z^*(f^*_{\text{ILO}}(x_0))$ being called an optimal prescription under ILO. Otherwise, the ILO problem is considered to be approximately solved, leading to a suboptimal prescription.

Although ILO considers the underlying relationship between prediction and decision, solving an ILO problem primarily involves two potential challenges: (i) the loss function $l_{\text{ILO}}(\cdot,\cdot)$ in Formula (2.7) is generally discontinuous, which makes it indifferentiable and often non-convex in $f(x_i)$ [7], thus preventing the derivation of the optimal model f^*_{ILO}; and (ii) the training process under ILO may be computationally challenging, as the downstream optimization model $\mathbf{M}_{\text{ds}}$ needs to be solved multiple times for each data sample in s_N. Therefore, the practical application of ILO is relatively limited.

3. Revisiting parameters and hyperparameters

In this section, we first introduce the traditional definitions of parameters and hyperparameters in Section 3.1. Next, we present the concept of model complexity in Section 3.2 and redefine parameters and hyperparameters in Section 3.3.

3.1. Traditional definitions of parameters and hyperparameters

For ML models, two key concepts are parameters and hyperparameters, with different parameters or hyperparameters leading to different ML models. Specifically, for a given ML method $M \in \mathcal{M}$, ML model f^1_M in the hypothesis class $\mathcal{F}_M$ differs from ML model f^2_M in $\mathcal{F}_M$ in that the values of the parameters and hyperparameters associated with the two ML models are not identical. An (optimal) learner L_M (where $L_M \in \mathcal{L}_M$) maps a dataset s_N to a set of (optimal) parameter and hyperparameter values, thereby determining a specific (optimal) ML model in $\mathcal{F}_M$.

Furthermore, determining the values of parameters and hyperparameters in ML models involves different procedures. In the classical ML literature, parameter values are typically estimated using a learning algorithm (i.e., a learner), which performs an efficient search over possible parameter

values [34]. These algorithms can be exact (e.g., least squares for LR) or heuristic (e.g., greedy top-down construction for DTs). In contrast, determining hyperparameter values often entails applying rules of thumb, adopting values used in similar problems, or tuning hyperparameters by trial and error. The tuning process typically involves splitting the dataset s_N into a training subset s_N^{train} and a validation subset s_N^{valid} according to a specified ratio, ensuring that $s_N = s_N^{\text{train}} \cup s_N^{\text{valid}}$. In general, the optimal hyperparameter values for an ML model on a given dataset remain unknown, because it is not practical to exhaustively search all possible combinations of hyperparameters. Consequently, we can typically only identify a relatively good (suboptimal) set of hyperparameters.

Despite their importance, the precise definitions of parameters and hyperparameters, as well as their distinctions, remain ambiguous. The traditional definitions are provided in Definition 2.

Definition 2 (Traditional definitions of parameters and hyperparameters [34, 35])**.** *A model parameter is a variable that is internal to the model and whose value is estimated/learned from the given data through the training process (i.e., fitting the model to the given data). A model hyperparameter is a variable that is external to the model and whose value is predetermined manually and empirically (i.e., independent of the given data) before the training/model-fitting process.*

However, the definitions presented above lack the necessary clarity and rigor, failing to theoretically emphasize the fundamental distinctions between parameters and hyperparameters. This therefore leads to several common misunderstandings and confusions, as listed below. The first confusion is how to distinguish between internal and external variables of an ML model. Parallels could be drawn with concepts from econometrics or statistics [67], such as endogenous variables (those influenced by other variables within the model) and exogenous variables (those determined by external factors). However, these concepts become considerably ambiguous when applied to the context of supervised learning. For the second confusion, Definition 2 seems to imply that the variable of an ML model whose value is learned from the data is definitively a parameter, and that the variable whose value is empirically determined before the training process is definitively a hyperparameter. However, this distinction is not always true, as illustrated in Example 2.

Example 2 (LR)**.** As mentioned in Example 1, for LR, we have $\mathcal{F}_{\text{LR}} = \{\hat{y} = a^\top x + b \mid (a, b) \in \mathbb{R}^{d+1}\}$. If the least squares algorithm is used to calibrate the values of a and b based on s_N, it is clear that, according to Definition 2, a and b are model parameters, with no hyperparameters involved. However, in a scenario where both a and b are constrained to integer values, determining their optimal values on s_N requires solving a complex integer programming problem, which can be difficult. To address this, one could search for the best values of (a, b) in the set $\{a_1, a_2, ..., a_{10}\} \times \{b_1, b_2, ..., b_{10}\}$ (Cartesian product) using s_N, where the integer values a_i and b_i $(i = 1, 2, ..., 10)$ are determined empirically based on rules of thumb or domain knowledge. In this case, a and b appear to be hyperparameters according to Definition 2, because their values are manually and empirically predetermined before the training/model-fitting process. Therefore, Definition 2 is somewhat confusing.

To clarify the essential distinctions between parameters and hyperparameters, we propose novel definitions of parameters and hyperparameters from the perspective of model complexity.

3.2. Model complexity

Model complexity is a fundamental concept in ML. A thorough understanding of the complexity of ML models is essential to comprehensively evaluate their capabilities and limitations. A general

definition of model complexity in the context of ML is provided in Definition 3.

Definition 3 (Model complexity in ML [44])**.** *In the context of ML, model complexity is a measure of an ML model's ability to capture underlying patterns in the given data, which is often associated with its ability to both fit training data and generalize to unseen data.*

Specifically, simple models, which exhibit low complexity, often struggle to capture underlying patterns in the given data, leading to underfitting. In this case, models tend to perform poorly on both training data and unseen data. In contrast, complex models, characterized by high complexity, can capture more complex relationships in the given data. Although complex models can perform well on training data, they are prone to overfitting and typically generalize poorly. Therefore, achieving a balance between simplicity and complexity is essential to ensure robust generalization of an ML model. To ensure clarity in subsequent discussions of factors affecting model complexity, we introduce the term "configuration elements" for supervised learning in Remark 2.

Remark 2 (Configuration elements)**.** In this study, the term "configuration elements" is used to denote all elements that need to be determined when a conceptualized model/hypothesis under an ML method is instantiated into a predictive function. For a given ML method $M \in \mathcal{M}$, let $\mathcal{E}_M$ be the set of all associated configuration elements. For example, for NNs, we have $\mathcal{E}_{\text{NN}} = \{\text{Weights}, \text{Biases}, \text{Activation Function}, \text{Number of Hidden Layers}, ...\}$. Configuration elements encompass a wide range of entities, including functions (e.g., activation functions for NNs, kernel functions for support vector machines) and coefficients (e.g., slope a and intercept b for LR, regularization coefficient λ for the least absolute shrinkage and selection operator (LASSO)). Therefore, when we refer to "the value of a configuration element", we may indicate a specific function, a specific numeric value, or something else.

Model complexity in supervised learning is mainly governed by three key factors. The first is the type of ML method. The variety of ML methods influences, to some extent, the types and quantities of associated configuration elements. Consequently, different methods can result in models of varying complexity, with more sophisticated methods typically yielding more complex models. For example, models generated by NNs tend to exhibit higher complexity than those produced by LR. The second is the type of learner. Different learners map the given dataset to ML models of varying complexity. More complex learners typically generate models with higher complexity. For example, when considering classification trees, models derived from the C4.5 algorithm (which uses the gain ratio for splitting and performs post-pruning) are generally more complex than those derived from the ID3 algorithm (which uses the information gain for splitting and does not perform pruning) [68]. The third is the value of configuration elements. The specific values assigned to configuration elements can significantly affect model complexity. For example, in NNs, the selection of activation functions and the number of hidden layers can influence model complexity, while in DTs, the maximum depth of the tree and the minimum number of samples in a leaf node are essential to determine model complexity.

Therefore, in supervised learning, the complexity of ML models is typically model-specific, with both definitions and measures of complexity closely related to model-specific characteristics. Consequently, complexity measures between different ML models are generally not directly comparable [44].

Remark 3. In this study, we discuss model complexity with respect to a given dataset s_N, as specified in Definition 3. Under identical conditions, model complexity is of course influenced by data

characteristics. Key factors that affect model complexity include data dimensionality, data distribution, and information volume as measured by Kolmogorov complexity [69], among others. For example, the complexity of an ML model typically increases with the number of features it considers.

3.3. New definitions of parameters and hyperparameters

Based on the concept of model complexity, we propose the following new definitions of parameters and hyperparameters for ML models.

Definition 4 (Parameters and hyperparameters based on model complexity)**.** *Given an ML method $M \in \mathcal{M}$, an associated learner $L_M \in \mathcal{L}_M$, and the hypothesis class $\mathcal{F}_M$:*

(1) *If all ML models in $\mathcal{F}_M$ exhibit the same complexity, then all associated configuration elements in $\mathcal{E}_M$ are defined as the parameters of the models.*
(2) *If the ML models in $\mathcal{F}_M$ differ in complexity, then the associated configuration elements that govern the complexity of an ML model are defined as the hyperparameters of the models, while the remaining configuration elements are defined as the parameters of the models.*

Let $\mathcal{P}_M$ and $\mathcal{H}_M$ be the sets of all parameters and hyperparameters of ML models for method M, respectively, so that $\mathcal{E}_M = \mathcal{P}_M \cup \mathcal{H}_M$.

Definition 4 is not intended to replace quantitative complexity measures such as VC dimension, Rademacher complexity, norm-based complexity, or architecture-specific capacity measures. Instead, it uses the effect of configuration elements on model complexity to determine their computational roles in the SLO/ILO optimization models. If a configuration element changes the admissible complexity level or capacity pattern of the induced hypothesis class, it is treated as a hyperparameter in our framework; if it only selects a model within a fixed complexity realization, it is treated as a parameter. This distinction is particularly useful for ILO because it determines whether the element appears in the uppermost complexity-realization search or in the middle/upper parameter-calibration search.

Based on Definition 4, Example 3 demonstrates the differences between parameters and hyperparameters for LASSO. Examples for other common ML methods are given in Appendix A.

Example 3 (LASSO)**.** For LASSO, the optimal regression coefficients (a^*, b^*) on s_N are determined by solving the following optimization problem (here, we consider a one-dimensional target for convenience):

$$(a^*, b^*) \in \underset{(a,b)\in\mathbb{R}^{d+1}}{\arg\min} \left\{ \frac{1}{2N} \sum_{i=1}^{N} (a^\top x_i + b - y_i)^2 + \lambda \|a\|_1 \right\}, \tag{3.1}$$

where $\lambda \in \mathbb{R}_+$ denotes the regularization coefficient. Therefore, the hypothesis class of all ML models is denoted by $\mathcal{F}_{\text{LASSO}} = \{\hat{y} = a^\top x + b \mid (a, b) \in \mathbb{R}^{d+1}\}$. The set of all configuration elements is denoted by $\mathcal{E}_{\text{LASSO}} = \{\lambda, a, b\}$, where λ governs the model complexity. When λ is close to zero, LASSO behaves similarly to LR, where the coefficients of a can take larger values and the model is more complex, fitting the training data closely. This can lead to overfitting, especially when there are many features or noise in the given data. As λ increases, the regularization becomes stronger, forcing more coefficients of a to shrink toward zero. This results in a simpler model, with fewer features contributing to the

prediction. In the extreme case, large values of λ can force a to be a zero vector, leading to a model that predicts just the mean (or intercept) of the given data, which is a very simple model. Therefore, λ should be considered as a hyperparameter, while the remaining elements should be considered as parameters, i.e., $\mathcal{H}_{\text{LASSO}} = \{\lambda\}$ and $\mathcal{P}_{\text{LASSO}} = \{a, b\}$.

Therefore, the concept of model complexity discussed in Section 3.2 and Definition 4 provide a theoretical clarification of the essential distinctions between parameters and hyperparameters. The practical implication is that, once $\mathcal{P}_M$ and $\mathcal{H}_M$ are identified, the implementation difficulty of ILO can be traced to three concrete sources: the cost of computing the decision loss, the size of the parameter-induced model set for each fixed complexity realization, and the size of the complexity-realization set. These three sources provide the basis for formulating the SLO/ILO optimization models in Section 4 and, through those formulations, for developing the evaluation framework in Section 5.

4. Optimal ML models under SLO and ILO

Based on our proposed definitions of parameters and hyperparameters in Section 3.3, we establish optimization frameworks to identify the optimal ML models under SLO and ILO in Sections 4.1 and 4.2, respectively. The predictive and decision losses have been introduced in Section 2.2; therefore, this section does not aim to redefine SLO or ILO, but instead shows how the previously defined losses lead to different optimization structures once parameters and hyperparameters are distinguished by model complexity.

4.1. A framework for identifying the optimal ML model under SLO

Traditional predictive analytics is primarily concerned with predictive accuracy using "predictive loss" as described in Section 2.2.1. In this section, we propose a general optimization framework, aiming to identify the optimal ML model under SLO, along with the optimal values of both parameters and hyperparameters.

4.1.1. ML models with both parameters and hyperparameters

In this case, the dataset s_N is typically divided into s_N^{train} (containing U data samples, where $U \in \{1, ..., N-1\}$) and s_N^{valid} (containing $N-U$ data samples). Meanwhile, the "optimal ML model" depends on both a specific subset s_N^{train} and a specific subset s_N^{valid}. To ensure clarity in subsequent discussions, we introduce the term "complexity realization" in Remark 4.

Remark 4 (Complexity realization)**.** In Definition 4, we define the configuration elements that govern the complexity of an ML model as hyperparameters. This implies that each combination of hyperparameter values corresponds to a specific level of model complexity. In this study, we introduce the term "complexity realization", denoted as c_M, to refer to a specific combination of hyperparameter values for models under method $M \in \mathcal{M}$. Moreover, let $\mathcal{C}_M$ be the set of all possible complexity realizations for models under method $M \in \mathcal{M}$, so that $c_M \in \mathcal{C}_M$. Moreover, for $c_M \in \mathcal{C}_M$, let f_{c_M} be a corresponding ML model, and let $\mathcal{F}_{c_M}$ be the set of all possible ML models, such that $f_{c_M} \in \mathcal{F}_{c_M}$ and $\mathcal{F}_M = \bigcup_{c_M \in \mathcal{C}_M} \mathcal{F}_{c_M}$. Consider LASSO as an example where $\mathcal{H}_{\text{LASSO}} = \{\lambda\}$ and $\mathcal{P}_{\text{LASSO}} = \{a, b\}$. Each value of λ corresponds to a specific level of model complexity. For instance, a

complexity realization could be $c_{\text{LASSO}} = \lambda = 0.1$, where $\lambda \in \mathcal{C}_{\text{LASSO}} = \mathbb{R}_+$, and $f_{0.1} \in \mathcal{F}_{0.1}$ denotes a corresponding ML model.

Based on the concept of complexity realization, for a given ML method $M \in \mathcal{M}$, the optimal ML model f^*_{SLO} in $\mathcal{F}_M$, along with its corresponding parameter and hyperparameter values, is determined as follows: (i) For each complexity realization $c_M \in \mathcal{C}_M$, the parameter values of f_{c_M} are calibrated in terms of MSE and irregularity on s_N^{train}, yielding the optimal ML model $f^*_{c_M}$ in $\mathcal{F}_{c_M}$. (ii) To identify the global optimal model, all $f^*_{c_M}$ for $c_M \in \mathcal{C}_M$ are compared in terms of MSE on s_N^{valid}, and the model with the lowest MSE is selected as the global optimal ML model f^*_{SLO}. The above process can be formulated as a bi-level optimization model:
[$\mathbf{M}^{\text{bi}}_{\text{SLO}}$] Upper-level:

$$f^*_{\text{SLO}} \in \underset{f^*_{c_M}(c_M \in \mathcal{C}_M)}{\arg\min} \frac{1}{N-U} \sum_{i=U+1}^{N} \|f^*_{c_M}(x_i) - y_i\|_2^2. \tag{4.1}$$

Lower-level (for each complexity realization $c_M \in \mathcal{C}_M$):

$$f^*_{c_M} \in \underset{f_{c_M} \in \mathcal{F}_{c_M}}{\arg\min} \left\{ \frac{1}{U} \sum_{i=1}^{U} \|f_{c_M}(x_i) - y_i\|_2^2 + \Omega(f_{c_M}) \right\}, \tag{4.2}$$

where the regularization term $\Omega(f_{c_M})$ controls the complexity of model f_{c_M}. The lower-level optimization problem (4.2) aims to identify the optimal solution $f^*_{c_M}$, along with the optimal parameter values, for each $c_M \in \mathcal{C}_M$, while the upper-level optimization problem (4.1) aims to identify the global optimal solution f^*_{SLO}, along with the optimal hyperparameter values.

However, the global optimal ML model f^*_{SLO} is generally difficult to identify for two main reasons: (i) Model $\mathbf{M}^{\text{bi}}_{\text{SLO}}$, as a bi-level optimization model, is typically NP-hard [70]; and (ii) the ML models under different complexity realizations are independent of each other, without forming a cohesive and interconnected structure for optimization problem (4.1). Consequently, we are typically confined to enumerating all possible complexity realizations. However, the cardinality $|\mathcal{C}_M|$ is often large or even infinite (e.g., $|\mathcal{C}_{\text{LASSO}}| = |\mathbb{R}_+|$ is infinite). As a result, the typical and practical approach involves (i) empirically selecting a small subset $\mathcal{C}^{\text{sub}}_M \subset \mathcal{C}_M$, where $|\mathcal{C}^{\text{sub}}_M|$ is finite and relatively small, and (ii) approximately solving model $\mathbf{M}^{\text{bi}}_{\text{SLO}}$ based on $\mathcal{C}^{\text{sub}}_M$, which probably employs techniques such as grid search, random search, and cross-validation. This approach results in a suboptimal ML model $f^{\text{sub}}_{\text{SLO}}$, with corresponding suboptimal parameter and hyperparameter values. For example, we commonly select a small subset $\mathcal{C}^{\text{sub}}_{\text{LASSO}} = \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$ for LASSO. Note that a suboptimal ML model $f^{\text{sub}}_{\text{SLO}}$ also leads to a suboptimal prescription $z^*(f^{\text{sub}}_{\text{SLO}}(x_0))$ under SLO:

$$z^*(f^{\text{sub}}_{\text{SLO}}(x_0)) \in \mathcal{Z}^*(f^{\text{sub}}_{\text{SLO}}(x_0)) = \underset{z \in \mathcal{Z}}{\arg\min} [f^{\text{sub}}_{\text{SLO}}(x_0)]^\top z. \tag{4.3}$$

Additionally, the adoption of a small subset $\mathcal{C}^{\text{sub}}_M$ further explains why hyperparameter values are typically determined through empirical methods before the training process, as stated in Definition 2.

4.1.2. ML models with only parameters

In this case, there is no need for a validation subset, and all data in s_N are used for training to calibrate the parameter values. For a given ML method $M \in \mathcal{M}$, the optimal ML model f^*_{SLO} in

$\mathcal{F}_M$, along with its optimal parameter values, can be identified by solving the following single-level optimization model
$[\mathbf{M}_{\text{SLO}}^{\text{single}}]$:

$$f_{\text{SLO}}^* \in \arg\min_{f \in \mathcal{F}_M} \frac{1}{N} \sum_{i=1}^{N} \|f(x_i) - y_i\|_2^2. \tag{4.4}$$

As optimization problem (4.4) is generally convex, it is possible to identify effective and exact solution algorithms, such as the least squares algorithm for LR.

4.2. A framework for identifying the optimal ML model under ILO

As discussed in Section 2.2.2, two potential challenges hinder the identification of an optimal ML model under ILO, which makes it difficult to solve the ILO problem, thus limiting its practical applications. In this section, we propose a general optimization framework to identify the optimal ML model under ILO, along with the optimal values of both parameters and hyperparameters. Moreover, we further investigate the potential computational challenges in identifying the optimal ML model under ILO.

4.2.1. ML models with both parameters and hyperparameters

First, the dataset s_N is typically divided into s_N^{train} (containing U data samples) and s_N^{valid} (containing $N - U$ data samples). Then, for a given ML method $M \in \mathcal{M}$ and a downstream optimization model $\mathbf{M}_{\text{ds}}$, the optimal ML model f_{ILO}^* in $\mathcal{F}_M$, along with its corresponding parameter and hyperparameter values, is determined as follows: (i) Given an ML model f in $\mathcal{F}_M$ and a data sample (x_i, y_i) in s_N, a point prediction $f(x_i)$ is first provided, followed by solving model $\mathbf{M}_{\text{ds}}$ to obtain the "decision loss" $l_{\text{ILO}}(f(x_i), y_i)$ using Formula (2.7). (ii) For each complexity realization $c_M \in \mathcal{C}_M$, the parameter values of model f_{c_M} are calibrated in terms of the "decision loss" on s_N^{train}, yielding the optimal ML model $f_{c_M}^*$ in $\mathcal{F}_{c_M}$. (iii) To identify the global optimal model, all $f_{c_M}^*$ for $c_M \in \mathcal{C}_M$ are compared in terms of the "decision loss" on s_N^{valid}, and the model with the lowest loss is selected as the global optimal ML model f_{ILO}^*. The above process can be formulated as a tri-level optimization model:
$[\mathbf{M}_{\text{ILO}}^{\text{tri}}]$ Uppermost level:

$$f_{\text{ILO}}^* \in \arg\min_{f_{c_M}^*(c_M \in \mathcal{C}_M)} \frac{1}{N-U} \sum_{i=U+1}^{N} l_{\text{ILO}}(f_{c_M}^*(x_i), y_i). \tag{4.5}$$

Middle level (for each complexity realization $c_M \in \mathcal{C}_M$):

$$f_{c_M}^* \in \arg\min_{f_{c_M} \in \mathcal{F}_{c_M}} \frac{1}{U} \sum_{i=1}^{U} l_{\text{ILO}}(f_{c_M}(x_i), y_i). \tag{4.6}$$

Lowest level (for an ML model f and a downstream model $\mathbf{M}_{\text{ds}}$):

$$\mathcal{Z}^*(f(x_i)) = \arg\min_{z \in \mathcal{Z}} [f(x_i)]^\top z, \quad i \in \{1, ..., N\}, \tag{4.7}$$

$$l_{\text{ILO}}(f(x_i), y_i) = \max_{z \in \mathcal{Z}^*(f(x_i))} \{y_i^\top z\} - \min_{z \in \mathcal{Z}} \{y_i^\top z\}, \quad i \in \{1, ..., N\}. \tag{4.8}$$

The lowest-level optimization problems (4.7) and (4.8) aim to predict and then solve model $\mathbf{M}_{\text{ds}}$ to obtain $l_{\text{ILO}}(f(x_i), y_i)$, which is subsequently used by both the middle-level and uppermost-level models. The middle-level optimization problem (4.6) aims to identify the optimal solution $f^*_{c_M}$, along with the optimal parameter values, for each $c_M \in \mathcal{C}_M$. The uppermost-level optimization problem (4.5) aims to identify the global optimal solution f^*_{ILO}, along with the optimal hyperparameter values. From model $\mathbf{M}^{\text{tri}}_{\text{ILO}}$, we further investigate the potential computational challenges involved in identifying an optimal ML model under ILO, as detailed below.

In the lowest-level problem, it may be computationally challenging to obtain the "decision loss". The downstream optimization model must be solved multiple times for each data sample in s_N. Specifically, model $\mathbf{M}_{\text{ds}}$ must be solved $|\mathcal{C}_M| \times |\mathcal{F}_{c_M}|$ times for each sample in s^{train}_N and $|\mathcal{C}_M|$ times for each sample in s^{valid}_N. Only when the optimization model is relatively straightforward to solve, such as in the case of a simple linear programming model, can the computation of the "decision loss" be considered comparatively easy.

In the middle-level problem, it may be computationally challenging to solve when considering the set $\mathcal{F}_{c_M}$ that incorporates all possible ML models for $c_M \in \mathcal{C}_M$. optimization problem (4.6) is generally discontinuous, making it indifferentiable and often non-convex in $f_{c_M}(x_i)$. Consequently, we are typically confined to enumerating all possible ML models. However, for each complexity realization $c_M \in \mathcal{C}_M$, the cardinality of the set of all possible ML models, i.e., $|\mathcal{F}_{c_M}|$, is generally large, even infinite. Only when $|\mathcal{F}_{c_M}|$ is relatively small to allow enumeration, or when considering a small subset $\mathcal{F}^{\text{sub}}_{c_M} \subset \mathcal{F}_{c_M}$ (leading to a suboptimal ML model $f^{\text{sub}}_{\text{ILO}}$), can the solution of the middle-level problem be considered comparatively easy.

In the uppermost-level problem, it may be computationally challenging to solve when considering the set $\mathcal{C}_M$ that incorporates all possible complexity realizations. The ML models under different complexity realizations are independent of each other, without forming a cohesive and interconnected structure for optimization problem (4.5). Consequently, we are typically confined to enumerating all possible complexity realizations. However, the cardinality of the set of all possible complexity realizations, i.e., $|\mathcal{C}_M|$, is generally large, even infinite. Only when $|\mathcal{C}_M|$ is relatively small to allow enumeration, or when considering a small subset $\mathcal{C}^{\text{sub}}_M \subset \mathcal{C}_M$ (leading to a suboptimal ML model $f^{\text{sub}}_{\text{ILO}}$), can the solution of the uppermost-level problem be considered comparatively easy.

4.2.2. ML models with only parameters

In this case, there is no need for a validation subset, and all data in s_N are used for training to calibrate the parameter values. For a given ML method $M \in \mathcal{M}$ and a downstream optimization model $\mathbf{M}_{\text{ds}}$, the optimal ML model f^*_{ILO} in $\mathcal{F}_M$, along with its optimal parameter values, can be determined through a bi-level optimization model:

[$\mathbf{M}^{\text{bi}}_{\text{ILO}}$] Upper level:

$$f^*_{\text{ILO}} \in \arg\min_{f \in \mathcal{F}_M} \frac{1}{N} \sum_{i=1}^{N} l_{\text{ILO}}(f(x_i), y_i). \tag{4.9}$$

Lower level (for an ML model f and a downstream model $\mathbf{M}_{\text{ds}}$):

Formulas (4.7) and (4.8).

In this case, the first potential computational challenge discussed above persists. Moreover, in the upper-level problem, it may be computationally challenging when considering the set $\mathcal{F}_M$ that

incorporates all possible ML models, because optimization problem (4.9) is generally discontinuous and non-convex in $f(x_i)$. Only when $|\mathcal{F}_M|$ is relatively small to allow enumeration, or when considering a small subset $\mathcal{F}_M^{\text{sub}} \subset \mathcal{F}_M$ (leading to a suboptimal ML model $f_{\text{ILO}}^{\text{sub}}$), can the solution of the upper-level problem be considered comparatively easy.

5. A general evaluation framework for ILO

In this section, we propose a general evaluation framework for assessing the difficulty of solving ILO problems. These problems are classified into two categories: (i) computationally tractable problems, where identifying an optimal ML model f_{ILO}^* for an optimal prescription is generally easy, indicated by "it is EASY"; or (ii) computationally challenging problems, where identifying an optimal ML model is generally hard, and a suboptimal ML model $f_{\text{ILO}}^{\text{sub}}$ is often obtained, leading to a suboptimal prescription, indicated by "it is HARD". This framework is structured around several key questions, as detailed below.

To make the terms "easy" and "small" operational, we interpret them relative to a computational budget specified by the practitioner. Let B_{eval} denote the maximum number of model/loss evaluations that can be performed, and let T_{lim} denote the maximum acceptable running time. Let τ_{ds} be an estimated upper bound on the time required to solve one instance of the downstream optimization model $\mathbf{M}_{\text{ds}}$ to the required accuracy. For an ML method with only parameters, exact enumeration is regarded as computationally tractable only if $|\mathcal{F}_M|$ is finite and

$$N_{\text{eval}}^{\text{only-p}} := N|\mathcal{F}_M| \le B_{\text{eval}}, \quad \tau_{\text{ds}} N_{\text{eval}}^{\text{only-p}} \le T_{\text{lim}}. \tag{5.1}$$

For an ML method with both parameters and hyperparameters, exact enumeration is regarded as computationally tractable only if $|\mathcal{C}_M|$ and $|\mathcal{F}_{c_M}|$ are finite and

$$N_{\text{eval}}^{\text{p-h}} := U \sum_{c_M \in \mathcal{C}_M} |\mathcal{F}_{c_M}| + (N-U)|\mathcal{C}_M| \le B_{\text{eval}}, \quad \tau_{\text{ds}} N_{\text{eval}}^{\text{p-h}} \le T_{\text{lim}}. \tag{5.2}$$

Thus, "small" does not refer to a universal numerical threshold; it means that the relevant set is finite and enumerable under the problem-specific budget and required solver accuracy. If these inequalities are not satisfied, the framework classifies the ILO implementation as computationally challenging and recommends restricted enumeration or approximate search over subsets such as $\mathcal{F}_M^{\text{sub}}$, $\mathcal{C}_M^{\text{sub}}$, and $\mathcal{F}_{c_M}^{\text{sub}}$.

Given a dataset s_N, an ML method $M \in \mathcal{M}$, a downstream optimization model $\mathbf{M}_{\text{ds}}$, and a new observation x_0, the first step is to address *Question 1*: Is computing the "decision loss" easy? (i) If NO, "it is HARD". In this case, approximation algorithms, such as heuristics, can be considered to efficiently solve model $\mathbf{M}_{\text{ds}}$. Consequently, the ILO problem will also be solved approximately. (ii) If YES, proceed to *Question 2*: Does the ML model to be used include hyperparameters? This leads to the following two scenarios.

Scenario 1: ML models with only parameters. *Question 3*: Is the cardinality of the set of all possible ML models, i.e., $|\mathcal{F}_M|$, small?

- If YES, "it is EASY". In this case, model $\mathbf{M}_{\text{ILO}}^{\text{bi}}$ (comprising Formulas (4.9), (4.7), and (4.8)) can be exactly solved for an optimal ML model f_{ILO}^*, leading to an optimal prescription $z^*(f_{\text{ILO}}^*(x_0))$ (see Formula (2.9)).
- If NO, "it is HARD". In this case, we can consider a small subset $\mathcal{F}_M^{\text{sub}} \subset \mathcal{F}_M$, and approximately solve model $\mathbf{M}_{\text{ILO}}^{\text{bi}}$ for a suboptimal ML model $f_{\text{ILO}}^{\text{sub}}$, leading to a suboptimal prescription

$z^*(f_{\text{ILO}}^{\text{sub}}(x_0))$:

$$z^*(f_{\text{ILO}}^{\text{sub}}(x_0)) \in \mathcal{Z}^*(f_{\text{ILO}}^{\text{sub}}(x_0)) = \arg\min_{z\in\mathcal{Z}} [f_{\text{ILO}}^{\text{sub}}(x_0)]^\top z. \tag{5.3}$$

Scenario 2: ML models with both parameters and hyperparameters. *Question 4*: Is the cardinality of the set of all possible complexity realizations, i.e., $|\mathcal{C}_M|$, small?

- If NO, "it is HARD". In this case, we can consider a small subset $\mathcal{C}_M^{\text{sub}} \subset \mathcal{C}_M$, and approximately solve model $\mathbf{M}_{\text{ILO}}^{\text{tri}}$ (comprising Formulas (4.5)–(4.8)) for a suboptimal ML model $f_{\text{ILO}}^{\text{sub}}$, leading to a suboptimal prescription.
- If YES, proceed to *Question 5*: Is the cardinality of the set of all possible ML models, i.e., $|\mathcal{F}_{c_M}|$, small for each $c_M \in \mathcal{C}_M$?
  - If YES, "it is EASY". In this case, model $\mathbf{M}_{\text{ILO}}^{\text{tri}}$ can be exactly solved for an optimal ML model f_{ILO}^*, leading to an optimal prescription.
  - If NO, "it is HARD". In this case, we can consider a small subset $\mathcal{F}_{c_M}^{\text{sub}} \subset \mathcal{F}_{c_M}$, and approximately solve model $\mathbf{M}_{\text{ILO}}^{\text{tri}}$ for a suboptimal ML model $f_{\text{ILO}}^{\text{sub}}$, leading to a suboptimal prescription.

The flowchart of the above evaluation framework for ILO is shown in Figure 1.

Example 4 demonstrates the difficulty of solving an ILO problem with LASSO, assuming that the downstream optimization model is relatively easy to solve. More illustrative examples for other common ML methods are given in Appendix B.

The proposed framework should be interpreted as a diagnostic and implementation-guidance tool rather than as a guarantee that every approximate subset search will be successful. When the full set $\mathcal{F}_M$ or $\mathcal{C}_M$ is replaced by a subset, poor prescriptions may arise if the subset misses regions with low decision loss, if the validation subset is too small to distinguish competing models, if the decision loss has large flat or discontinuous regions, or if approximate solutions of the downstream problems introduce non-negligible errors. In such cases, the practitioner should enlarge the candidate subset, use multiple initializations, compare several subset-generation rules, or report the resulting prescription as an explicitly suboptimal one. These limitations are important because the workflow helps identify where computational difficulty arises and what type of exact or approximate solution strategy is reasonable before implementation.

Example 4 (LASSO)**.** As mentioned in Example 3, the hypothesis class of all ML models is $\mathcal{F}_{\text{LASSO}} = \{\hat{y} = a^\top x + b \mid (a, b) \in \mathbb{R}^{d+1}\}$, and we have $\mathcal{H}_{\text{LASSO}} = \{\lambda\}$ and $\mathcal{P}_{\text{LASSO}} = \{a, b\}$. It is clear that the cardinality of the set of all possible complexity realizations, i.e., $|\mathcal{C}_{\text{LASSO}}|$, is infinite as $\lambda \in \mathbb{R}_+$; hence "it is HARD". Additionally, for each $\lambda \in \mathcal{C}_{\text{LASSO}}$, the cardinality of the set of all possible ML models $|\mathcal{F}_\lambda|$ is also infinite. An approximate solution approach is described as follows. First, we select a small subset of $\mathcal{C}_{\text{LASSO}}$, e.g., $\mathcal{C}_{\text{LASSO}}^{\text{sub}} = \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, and for each $\lambda \in \mathcal{C}_{\text{LASSO}}^{\text{sub}}$, we predetermine a set of initial parameter values $(a_\lambda^{\text{init}}, b_\lambda^{\text{init}})$ under ILO. These initial values can be set empirically, or the optimal parameter values under SLO can be selected as the initial values. Then, for each $\lambda \in \mathcal{C}_{\text{LASSO}}^{\text{sub}}$, we conduct a finite number of searches in specific neighborhoods of a_λ^{init} and b_λ^{init}, which corresponds to a small subset $\mathcal{F}_\lambda^{\text{sub}} \subset \mathcal{F}_\lambda$, and identify the best parameter values a_λ and b_λ, and the corresponding suboptimal ML model f_λ^{sub} in $\mathcal{F}_\lambda^{\text{sub}}$ under ILO by approximately solving the middle-level optimization problem (4.6) in model $\mathbf{M}_{\text{ILO}}^{\text{tri}}$. Finally, based on $\mathcal{C}_{\text{LASSO}}^{\text{sub}}$, we approximately solve the uppermost-level optimization problem (4.5) in model $\mathbf{M}_{\text{ILO}}^{\text{tri}}$, and identify the best (suboptimal) ML

model $f^{\text{sub}}_{\text{LASSO}}$ under ILO among all f^{sub}_{λ} for $\lambda \in \mathcal{C}^{\text{sub}}_{\text{LASSO}}$, along with the corresponding hyperparameter value λ^{sub} and parameter values $a_{\lambda^{\text{sub}}}$ and $b_{\lambda^{\text{sub}}}$.

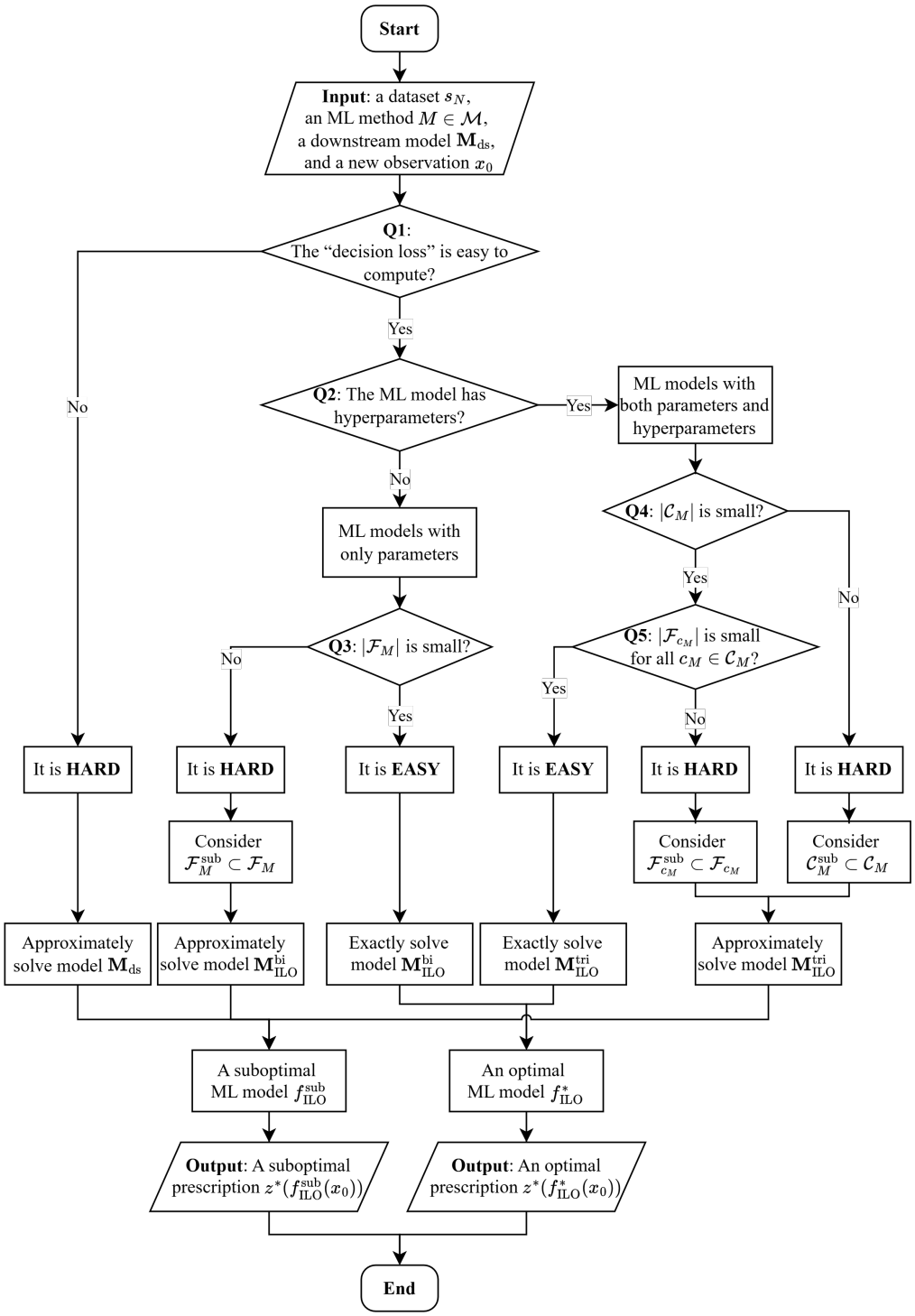


Figure 1. The flowchart of the evaluation framework for ILO.

6. Case study: A stochastic binary decision-making problem

In this section, we first describe a stochastic binary decision-making problem in Section 6.1, and then compare the prescriptive performance under SLO and ILO, demonstrating our proposed evaluation framework for ILO in Section 6.2.

6.1. Problem description

Consider an enterprise that needs to make a decision $z \in \{0, 1\}$ between two production plans for a product, with the objective of minimizing the expected production cost. Plan A ($z = 0$) incurs a deterministic per-unit cost of p_A, while Plan B ($z = 1$) incurs an uncertain per-unit cost $Y \in \mathcal{Y} \subset \mathbb{R}_+$. Two associated feature variables are considered: the fluctuating market prices of two raw materials, denoted by $X^{(1)}$ and $X^{(2)}$, respectively. Thus, the feature vector is $X = (X^{(1)}, X^{(2)}) \in \mathcal{X} \subset \mathbb{R}_+^2$.

Suppose that the enterprise can observe the market prices of the two raw materials at the beginning of each month. The enterprise must then decide on the production plan for the month, a decision that cannot be changed once made. If Plan B is adopted, a one-time procurement of the two raw materials must be made at the beginning of the month. Let $\mathcal{D}_{x_0}$ be the underlying true conditional distribution of Y given $X = x_0$, which is assumed to have a bounded first moment, thereby ensuring that the expected cost is well-defined. The CSO problem is then formulated as

$$\min_{z\in\{0,1\}} \mathbb{E}_{Y\sim\mathcal{D}_{x_0}}[p_A(1-z) + Yz] = \min_{z\in\{0,1\}} \left\{p_A(1-z) + \mathbb{E}_{Y\sim\mathcal{D}_{x_0}}[Y]z\right\}. \tag{6.1}$$

Suppose that the enterprise has a set of N i.i.d. historical data samples $s_N = \{(x_1, y_1), ..., (x_N, y_N)\}$, where y_i is the realized per-unit cost under Plan B for market prices x_i ($i \in \{1, ..., N\}$). Based on s_N, the enterprise aims to prescribe a decision that minimizes the corresponding decision cost as much as possible for a new observation x_0.

6.2. Numerical experiments

To solve the aforementioned stochastic binary decision-making problem, we conduct numerical experiments with synthetic data to prescribe decisions using three ML methods (i.e., kNN, LR, and LASSO) under the SLO and ILO paradigms, respectively. Moreover, we demonstrate our proposed evaluation framework for ILO.

6.2.1. Experimental settings

The settings for the cost parameters and distributions are detailed below. The per-unit cost of Plan A is $p_A = 100$. The synthetic data samples are independently drawn from the following assumed joint distribution of (X, Y): (i) the random market prices follow truncated normal distributions $X^{(1)} \sim \mathcal{N}(40, 2)$ and $X^{(2)} \sim \mathcal{N}(60, 2)$, with truncated intervals of $[0, 80]$ and $[0, 120]$, respectively, ensuring positive numbers for the prices; and (ii) the random per-unit cost of Plan B follows a truncated normal distribution $Y \sim \mathcal{N}(X^{(1)} + X^{(2)}, 5)$, with a truncated interval of $[0, 2(X^{(1)} + X^{(2)})]$, ensuring that the cost is positive.

To investigate the impact of different sizes of s_N on decision-making, the training sample size N ranges from 10 to 10^3, following a logarithmic scale, with specific values selected from the set $N_{\text{size}} = \{10, 22, 46, 100, 215, 464, 1000\}$. Let $\bar{s}_J = \{(\bar{x}_1, \bar{y}_1), ..., (\bar{x}_J, \bar{y}_J)\}$ be the test dataset. To mitigate the

impact of random variability, ten training datasets of s_N are randomly generated for each N in N_{size}, and ten test datasets of $\bar{s}_J$ are randomly generated, each containing $J = 100$ samples. Under the SLO and ILO paradigms, three ML methods (i.e., kNN, LR, and LASSO) are employed, each being trained on s_N. Decisions are subsequently prescribed for each $\bar{x}_j$ ($j \in \{1, ..., J\}$) in $\bar{s}_J$, and the corresponding decision losses are calculated. Note that our downstream optimization model is relatively simple to solve, so the decision loss is easy to compute under ILO (i.e., the answer to Q1 is YES for the proposed evaluation framework). For kNN and LASSO, the dataset s_N is randomly divided into a training subset s_N^{train} and a validation subset s_N^{valid} with a 4:1 ratio. The computational processes for the three ML methods are presented below.

kNN. First, the feature data of each s_N are normalized. Then, for each s_N, considering $\mathcal{C}_{k\text{NN}} = \{1, 2, ..., U\}$ (recall that U is the size of s_N^{train}, and here we consider all possible complexity realizations) and the Euclidean distance metric, the optimal ML models under SLO and ILO, i.e., $f^*_{k\text{NN,SLO}}$ and $f^*_{k\text{NN,ILO}}$, can be identified by exactly solving model $\mathbf{M}^{\text{bi}}_{\text{SLO}}$ (comprising Formulas (4.1) and (4.2)) and model $\mathbf{M}^{\text{tri}}_{\text{ILO}}$ (comprising Formulas (4.5)–(4.8)), respectively. Here, using the proposed evaluation framework for ILO, the answers to Q2, Q4, and Q5 are all YES, and therefore "it is EASY". Finally, the optimal prescriptions for $\bar{x}_j$ under SLO and ILO can be made using Formulas (2.6) and (2.9), respectively, with the corresponding decision losses calculated using Formula (2.7).

LR. First, for each s_N, the optimal ML model under SLO, i.e., $f^*_{\text{LR,SLO}}$, along with the corresponding optimal parameter values a_1^*, a_2^*, and b^* (where a_1^* and a_2^* denote the optimal regression coefficients for features $X^{(1)}$ and $X^{(2)}$, respectively), can be identified by exactly solving model $\mathbf{M}^{\text{single}}_{\text{SLO}}$ (i.e., Formula (4.4)). Then, the optimal prescription for $\bar{x}_j$ under SLO can be made using Formula (2.6). Next, we conduct a finite number of searches in specific neighborhoods of a_1^*, a_2^*, and b^*, and identify the best (suboptimal) parameter values a_1^{sub}, a_2^{sub}, and b^{sub} and the corresponding suboptimal ML model $f^{\text{sub}}_{\text{LR,ILO}}$ under ILO by approximately solving model $\mathbf{M}^{\text{bi}}_{\text{ILO}}$ (comprising Formulas (4.9), (4.7), and (4.8)). Specifically, $(a_1^{\text{sub}}, a_2^{\text{sub}}, b^{\text{sub}})$ can be identified in the set $\{0.85a_1^*, 0.90a_1^*, 0.95a_1^*, a_1^*, 1.05a_1^*, 1.10a_1^*, 1.15a_1^*\} \times \{0.85a_2^*, ..., 1.15a_2^*\} \times \{0.85b^*, ..., 1.15b^*\}$ (Cartesian product). Here, using the proposed evaluation framework for ILO, the answers to Q2 and Q3 are all NO, and therefore "it is HARD". Then, the suboptimal prescription for $\bar{x}_j$ under ILO can be made using Formula (5.3). Finally, the decision losses for these prescriptions under SLO and ILO can be calculated using Formula (2.7).

LASSO. First, for each s_N, considering $\mathcal{C}^{\text{sub}}_{\text{LASSO}} = \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, the suboptimal ML models under SLO and ILO, i.e., $f^{\text{sub}}_{\text{LASSO,SLO}}$ and $f^{\text{sub}}_{\text{LASSO,ILO}}$, can be identified by approximately solving model $\mathbf{M}^{\text{bi}}_{\text{SLO}}$ (comprising Formulas (4.1) and (4.2)) and model $\mathbf{M}^{\text{tri}}_{\text{ILO}}$ (comprising Formulas (4.5)–(4.8)), respectively. Here, using the proposed evaluation framework for ILO, the answers to Q2 and Q4 are YES and NO, respectively, and therefore "it is HARD". The determination of the suboptimal parameter values under ILO is similar to that of LR. Then, the suboptimal prescriptions for $\bar{x}_j$ under SLO and ILO can be made using Formulas (4.3) and (5.3), respectively, with the corresponding decision losses calculated using Formula (2.7).

6.2.2. Results

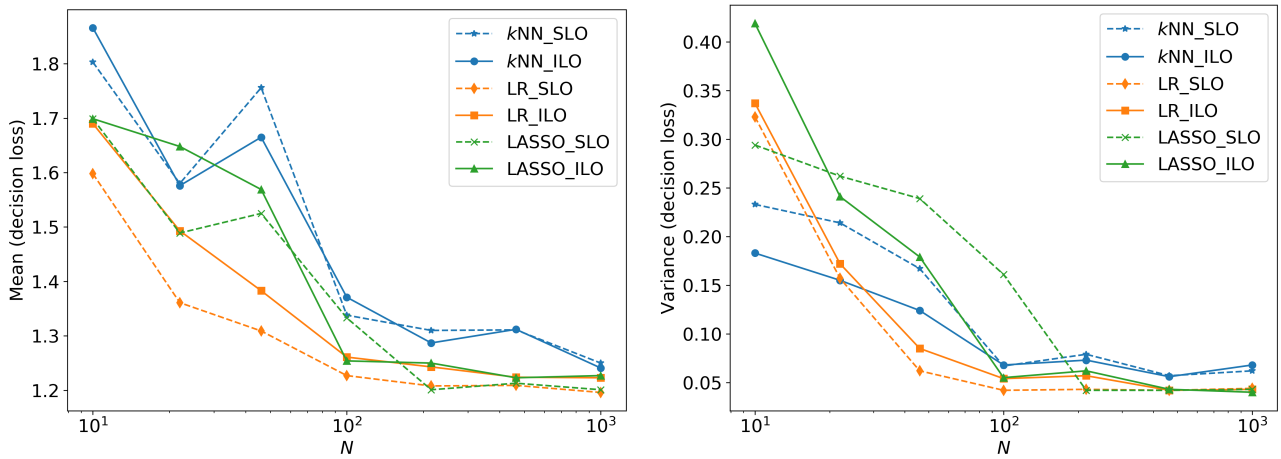
The means and variances of the decision losses of the test datasets across different training sample sizes are shown in Tables 1 and 2, with line charts shown in Figure 2(a),(b), respectively.

Table 1. The mean of decision losses for kNN, LR, and LASSO.

Method	Paradigm	N (training sample size)							Average
		10	22	46	100	215	464	1000	
kNN	SLO	1.803	1.580	1.756	1.338	1.310	1.311	1.250	1.478
	ILO	1.866	1.576	1.665	1.371	1.287	1.312	1.241	1.474
LR	SLO	1.598	1.361	1.309	1.227	1.208	1.209	1.196	1.301
	ILO	1.690	1.493	1.383	1.261	1.243	1.224	1.223	1.360
LASSO	SLO	1.699	1.489	1.525	1.333	1.201	1.213	1.201	1.380
	ILO	1.699	1.648	1.569	1.254	1.250	1.223	1.227	1.410

Table 2. The variance of decision losses for kNN, LR, and LASSO.

Method	Paradigm	N (training sample size)							Average
		10	22	46	100	215	464	1000	
kNN	SLO	0.233	0.214	0.167	0.067	0.079	0.057	0.062	0.126
	ILO	0.183	0.155	0.124	0.068	0.073	0.056	0.068	0.104
LR	SLO	0.323	0.157	0.062	0.042	0.043	0.042	0.044	0.102
	ILO	0.337	0.172	0.085	0.054	0.057	0.042	0.043	0.113
LASSO	SLO	0.294	0.262	0.239	0.161	0.042	0.042	0.043	0.155
	ILO	0.419	0.241	0.179	0.055	0.062	0.043	0.040	0.148



(a) Mean **(b)** Variance

Figure 2. The mean and variance of decision losses for kNN, LR, and LASSO.

The results show that, overall, as the training sample size N increases, the means and variances of decision losses for kNN, LR, and LASSO under both SLO and ILO tend to decrease. This pattern is

consistent with the intuition that larger training datasets generally improve the stability of the learned prescriptions.

For kNN, SLO has a lower average decision cost than ILO when $N = 10$ and $N = 100$, whereas ILO has a lower average decision cost when $N = 46, 215,$ and 1000; for $N = 22$ and $N = 464$, the average decision costs are nearly the same. The variance results are similarly mixed: SLO is more stable when $N = 1000$, ILO is more stable when $N = 10, 22, 46,$ and 215, and the two paradigms are nearly the same when $N = 100$ and $N = 464$. Across all training sample sizes, the average values of the mean and variance of decision losses under ILO are 0.27% and 17.46% lower, respectively, than those under SLO. Because the mean-loss difference is very small, we interpret this result as evidence that kNN under SLO and ILO has comparable average decision costs in this case study, while ILO shows somewhat lower average variability.

For LR, SLO consistently yields a lower average decision cost than ILO across all training sample sizes in N_{size}. SLO also has lower variance for $N = 10, 22, 46, 100,$ and 215, while the stability of SLO and ILO is nearly the same for $N = 464$ and $N = 1000$. Across the training sample sizes, the average values of the mean and variance of decision losses under SLO are 4.34% and 9.73% lower, respectively, than those under ILO. These observed results suggest that LR under SLO is more favorable than LR under ILO for the studied stochastic binary decision-making problem, while the conclusion should be understood as case-specific evidence illustrating the framework rather than as a general dominance claim.

For LASSO, SLO has a lower average decision cost than ILO when $N = 22, 46, 215, 464,$ and 1000, whereas ILO has a lower average decision cost when $N = 100$, and the two paradigms are nearly the same when $N = 10$. Regarding variance, SLO is more stable when $N = 10$ and $N = 215$, ILO is more stable when $N = 22, 46,$ and 100, and the two paradigms are nearly the same when $N = 464$ and $N = 1000$. Across all training sample sizes, the average mean decision loss under SLO is 2.13% lower than that under ILO, while the average variance under ILO is 4.52% lower than that under SLO. Thus, for LASSO, the observed comparison does not support a uniform dominance claim: SLO has a modest advantage in average decision cost, whereas ILO has a modest advantage in average stability.

Comparing ML methods within each paradigm, LR has the lowest average values of both mean and variance of decision losses under SLO. Specifically, LR's average mean decision loss is 11.98% and 5.72% lower than those of kNN and LASSO, respectively, while its average variance is 19.05% and 34.19% lower than those of kNN and LASSO, respectively. Under ILO, LR achieves the lowest average mean decision loss, which is 7.73% and 3.55% lower than those of kNN and LASSO, respectively. However, kNN has the lowest average variance under ILO, which is 7.96% and 29.73% lower than those of LR and LASSO, respectively. These findings reinforce that the relative effectiveness of SLO and ILO is method- and data-size-dependent, rather than universally favoring one paradigm.

Furthermore, to analyze the solution efficiency of SLO and ILO, we compute the means of running times to identify the optimal or suboptimal ML models for kNN, LR, and LASSO under SLO and ILO. The results are presented in Table 3 and Figure 3. Overall, as the training sample size N increases, the mean running times to identify the optimal or suboptimal ML models show an increasing trend. For kNN, the mean running times under SLO and ILO are nearly the same across all training sample sizes, with the average mean time under SLO being 2.61% lower than that under ILO. For LR and LASSO, the mean running times to identify the suboptimal ML models under SLO are nearly zero across all

training sample size s, and the average mean times under SLO are 99.63% and 99.57% lower than those under ILO, respectively. This efficiency comparison is consistent with the evaluation framework: LR and LASSO require approximate searches under ILO, whereas their SLO counterparts are much easier to solve in the present setting.

Table 3. The mean of running times for identifying the optimal or suboptimal ML models for *k*NN, LR, and LASSO (unit: seconds).

Method	Paradigm	*N* (training sample size)							Average
		10	22	46	100	215	464	1000	
*k*NN	SLO	0.009	0.016	0.035	0.080	0.208	0.833	7.177	1.194
	ILO	0.006	0.012	0.028	0.066	0.190	0.847	7.435	1.226
LR	SLO	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
	ILO	0.040	0.051	0.073	0.121	0.225	0.444	0.923	0.268
LASSO	SLO	0.005	0.003	0.004	0.006	0.005	0.005	0.004	0.005
	ILO	0.198	0.248	0.345	0.551	0.985	1.920	3.955	1.172

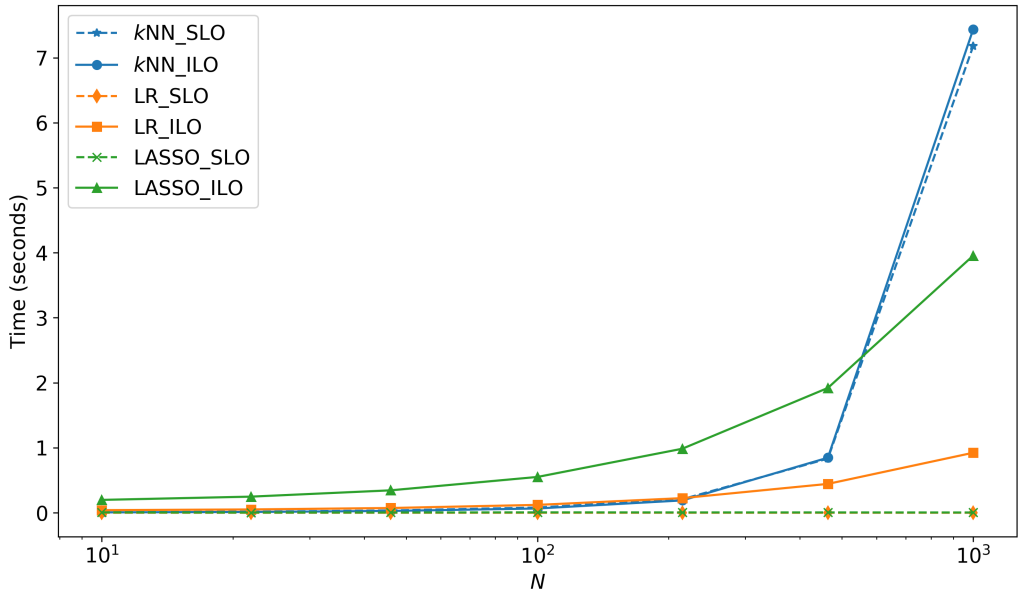


Figure 3. The mean of running times for identifying the optimal or suboptimal ML models for *k*NN, LR, and LASSO.

In summary, we conduct a numerical case study to demonstrate our proposed evaluation framework for ILO. Moreover, we analyze and compare the prescriptive performance and solution efficiency of SLO and ILO, concluding that although SLO achieves higher solution efficiency than ILO in most scenarios, their prescriptive effectiveness is dependent on the specific ML methods and training sample sizes. These numerical observations should be interpreted as case-specific evidence that illustrates the framework. This evaluation framework, therefore, offers valuable insights for the practical applications

of ILO in solving real-world CSO problems.

7. Conclusions

In this paper, we primarily propose a general evaluation framework for ILO in solving CSO problems. This framework allows us to classify ILO problems into those that are relatively easy to solve, enabling the exact determination of optimal prescriptions, and those that are more challenging, for which approximate approaches are required to obtain suboptimal prescriptions.

We first propose new definitions of parameters and hyperparameters based on the concept of model complexity, aiming to clarify existing ambiguities in traditional definitions. This leads to the development of optimization frameworks for identifying the optimal ML models under SLO and ILO, respectively. Subsequently, we introduce our evaluation framework that assesses the difficulty of implementing ILO for solving CSO problems, distinguishing between computationally tractable and challenging cases. Finally, we present a numerical case study that (i) illustrates how the proposed evaluation framework can be applied to an ILO implementation, and (ii) compares the efficacy and efficiency of SLO and ILO in solving a stochastic binary decision-making problem. Our results show that their prescriptive effectiveness varies across different ML methods and training sample sizes, and the solution efficiency of SLO exceeds that of ILO.

In summary, our study provides both valuable insights and practical guidance for implementing the ILO paradigm in solving real-world CSO problems in OR/MS. In addition, it contributes to the literature on the comparative effectiveness of SLO and ILO.

This study has several limitations. The numerical study focuses on one stochastic binary decision-making problem and three representative ML methods, so the empirical findings should not be interpreted as a universal ranking of SLO and ILO. Moreover, when the framework classifies a problem as computationally challenging, the quality of the resulting suboptimal prescription depends on how the restricted subsets $\mathcal{C}_M^{\text{sub}}$ and $\mathcal{F}_{c_M}^{\text{sub}}$ are constructed. Future research may extend the empirical evaluation to additional real-world datasets and downstream optimization problems, develop adaptive subset-generation strategies, and derive problem-specific bounds for the loss incurred by approximate ILO implementations.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

Shuaian Wang is an editorial board member for Electronic Research Archive and was not involved in the editorial review or the decision to publish this article. All authors declare that there are no competing interests.

References

1. Z. Hu, M. Xu, Q. Cheng, Multimodal large-language model empowering next-generation autonomous driving systems, *J. Intell. Connected Veh.*, **8** (2025), 9210059. https://doi.org/10.26599/jicv.2025.9210059
2. Z. Gao, B. Jia, D. Xie, W. Wang, J. Wu, A discussion on the complexity and transit mechanisms of urban traffic systems, *Engineering*, **44** (2025), 24–29. https://doi.org/10.1016/j.eng.2024.12.006
3. U. Sadana, A. Chenreddy, E. Delage, A. Forel, E. Frejinger, T. Vidal, A survey of contextual optimization methods for decision-making under uncertainty, *Eur. J. Oper. Res.*, **320** (2025), 271–289. https://doi.org/10.1016/j.ejor.2024.03.020
4. D. Bertsimas, N. Kallus, From predictive to prescriptive analytics, *Manage. Sci.*, **66** (2020), 1025–1044. https://doi.org/10.1287/mnsc.2018.3253
5. A. N. Elmachtoub, P. Grigas, Smart "predict, then optimize", *Manage. Sci.*, **68** (2022), 9–26. https://doi.org/10.1287/mnsc.2020.3922
6. Y. Deng, S. Sen, Predictive stochastic programming, *Comput. Manage. Sci.*, **19** (2022), 65–98. https://doi.org/10.1007/s10287-021-00400-0
7. N. Ho-Nguyen, F. Kılınç-Karzan, Risk guarantees for end-to-end prediction and optimization processes, *Manage. Sci.*, **68** (2022), 8680–8698. https://doi.org/10.1287/mnsc.2022.4321
8. B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, et al., Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges, *Wiley Interdiscip. Rev.: Data Min. Knowl. Discovery*, **13** (2023), e1484. https://doi.org/10.1002/widm.1484
9. T. Yu, H. Zhu, Hyper-parameter optimization: A review of algorithms and applications, preprint, arXiv:2003.05689.
10. M. Feurer, F. Hutter, *Hyperparameter Optimization*, Springer International Publishing, Cham, 2019. https://doi.org/10.1007/978-3-030-05318-5_1
11. J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, *Adv. Neural Inf. Process. Syst.*, **24** (2011).
12. J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, *J. Mach. Learn. Res.*, **13** (2012), 281–305.
13. J. Snoek, H. Larochelle, R. P. Adams, Practical Bayesian optimization of machine learning algorithms, *Adv. Inf. Process. Syst.*, **25** (2012).
14. B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, N. de Freitas, Taking the human out of the loop: A review of Bayesian optimization, *Proc. IEEE*, **104** (2015), 148–175. https://doi.org/10.1109/JPROC.2015.2494218
15. L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, *J. Mach. Learn. Res.*, **18** (2018), 1–52.
16. C. Thornton, F. Hutter, H. H. Hoos, K. Leyton-Brown, Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms, in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, (2013), 847–855. https://doi.org/10.1145/2487575.2487629

17. M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, *Adv. Neural Inf. Process. Syst.*, **28** (2015).

18. L. Yang, A. Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, *Neurocomputing*, **415** (2020), 295–316. https://doi.org/10.1016/j.neucom.2020.07.061

19. F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in *Proceedings of the International Conference on Learning and Intelligent Optimization*, Springer, (2011), 507–523. https://doi.org/10.1007/978-3-642-25566-3_40

20. A. Klein, S. Falkner, S. Bartels, P. Hennig, F. Hutter, Fast Bayesian optimization of machine learning hyperparameters on large datasets, in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, PMLR, (2017), 528–536.

21. M. Feurer, B. Letham, F. Hutter, E. Bakshy, Practical transfer learning for Bayesian optimization, preprint, arXiv:1802.02219.

22. S. Falkner, A. Klein, F. Hutter, BOHB: Robust and efficient hyperparameter optimization at scale, in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, (2018), 1437–1446.

23. E. Real, A. Aggarwal, Y. Huang, Q. V. Le, Regularized evolution for image classifier architecture search, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2019), 4780–4789. https://doi.org/10.1609/aaai.v33i01.33014780

24. D. Hernández-Lobato, J. M. Hernández-Lobato, A. Shah, R. P. Adams, Predictive entropy search for multi-objective Bayesian optimization, in *Proceedings of the 33rd International Conference on Machine Learning*, PMLR, (2016), 1492–1501.

25. B. Paria, K. Kandasamy, B. Póczos, A flexible framework for multi-objective Bayesian optimization using random scalarizations, in *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, PMLR, (2020), 766–776.

26. B. Zoph, Q. V. Le, Neural architecture search with reinforcement learning, in *Proceedings of the International Conference on Learning Representations*, 2017.

27. T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: A survey, *J. Mach. Learn. Res.*, **20** (2019), 1–21.

28. M. Baratchi, C. Wang, S. Limmer, J. N. van Rijn, H. Hoos, T. Bäck, et al., Automated machine learning: Past, present and future, *Artif. Intell. Rev.*, **57** (2024), 122. https://doi.org/10.1007/s10462-024-10726-1

29. L. Franceschi, M. Donini, V. Perrone, A. Klein, C. Archambeau, M. Seeger, et al., Hyperparameter optimization in machine learning, *Found. Trends Mach. Learn.*, **18** (2025), 975–1109. https://doi.org/10.1561/2200000088

30. P. Gijsbers, M. L. P. Bueno, S. Coors, E. LeDell, S. Poirier, J. Thomas, et al., AMLB: An AutoML benchmark, *J. Mach. Learn. Res.*, **25** (2024), 1–65.

31. F. Mohr, M. Wever, Naive automated machine learning, *Mach. Learn.*, **112** (2023), 1131–1170. https://doi.org/10.1007/s10994-022-06200-0

32. N. Mallik, E. Bergman, C. Hvarfner, D. Stoll, M. Janowski, M. Lindauer, et al., PriorBand: Practical hyperparameter optimization in the age of deep learning, *Adv. Neural Inf. Process. Syst.*, **36** (2023), 7377–7391. https://doi.org/10.52202/075280-0323
33. S. Müller, M. Feurer, N. Hollmann, F. Hutter, PFNs4BO: In-context learning for Bayesian optimization, preprint, arXiv:2305.17535.
34. M. Kuhn, K. Johnson, *Applied Predictive Modeling*, Springer, New York, 2013. https://doi.org/10.1007/978-1-4614-6849-3
35. P. Probst, A. L. Boulesteix, B. Bischl, Tunability: Importance of hyperparameters of machine learning algorithms, preprint, arXiv:1802.09596.
36. V. Kumar, M. L. Garg, Predictive analytics: A review of trends and techniques, *Int. J. Comput. Appl.*, **182** (2018), 31–37. https://doi.org/10.5120/ijca2018917434
37. N. Mishra, S. Silakari, Predictive analytics: A survey, trends, applications, oppurtunities & challenges, *Int. J. Comput. Sci. Inf. Technol.*, **3** (2012), 4434–4438.
38. M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, J. Sohl-Dickstein, On the expressive power of deep neural networks, preprint, arXiv:1606.05336.
39. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer Science & Business Media, Berlin, 2013. https://doi.org/10.1007/978-1-4757-3264-1
40. P. L. Bartlett, S. Mendelson, Rademacher and Gaussian complexities: Risk bounds and structural results, *J. Mach. Learn. Res.*, **3** (2002), 463–482.
41. H. Buhrman, R. De Wolf, Complexity measures and decision tree complexity: A survey, *Theor. Comput. Sci.*, **288** (2002), 21–43. https://doi.org/10.1016/S0304-3975(01)00144-X
42. N. Bulso, M. Marsili, Y. Roudi, On the complexity of logistic regression models, *Neural Comput.*, **31** (2019), 1592–1623. https://doi.org/10.1162/neco_a_01207
43. T. Liang, T. Poggio, A. Rakhlin, J. Stokes, Fisher-rao metric, geometry, and complexity of neural networks, preprint, arXiv:1711.01530.
44. X. Hu, L. Chu, J. Pei, W. Liu, J. Bian, Model complexity of deep learning: A survey, *Knowl. Inf. Syst.*, **63** (2021), 2585–2619. https://doi.org/10.1007/s10115-021-01605-0
45. S. Liu, S. Cong, L. Yang, Survey of research on autonomous driving testing with large models, *Commun. Transp. Res.*, **5** (2025), 100179. https://doi.org/10.1016/j.commtr.2025.100179
46. R. Kannan, G. Bayraksan, J. R. Luedtke, Residuals-based distributionally robust optimization with covariate information, preprint, arXiv:2012.01088.
47. D. Bertsimas, B. Van Parys, Bootstrap robust prescriptive analytics, *Math. Program.*, **195** (2022), 39–78. https://doi.org/10.1007/s10107-021-01679-2
48. G. Y. Ban, C. Rudin, The big data newsvendor: Practical insights from machine learning, *Oper. Res.*, **67** (2019), 90–108. https://doi.org/10.1287/opre.2018.1757
49. S. Liu, L. He, Z. Shen, On-time last-mile delivery: Order assignment with travel-time predictors, *Manage. Sci.*, **67** (2021), 4095–4119. https://doi.org/10.1287/mnsc.2020.3741
50. A. Esteban-Pérez, J. M. Morales, Distributionally robust optimal power flow with contextual information, *Eur. J. Oper. Res.*, **306** (2023), 1047–1058. https://doi.org/10.1016/j.ejor.2022.10.024

51. Z. Y. Chen, M. Sun, X. X. Han, Prediction-driven collaborative emergency medical resource allocation with deep learning and optimization, *J. Oper. Res. Soc.*, **74** (2023), 590–603. https://doi.org/10.1080/01605682.2022.2101953

52. G. Perakis, M. Sim, Q. Tang, P. Xiong, Robust pricing and production with information partitioning and adaptation, *Manage. Sci.*, **69** (2023), 1398–1419. https://doi.org/10.1287/mnsc.2022.4446

53. V. Monga, Y. Li, Y. C. Eldar, Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing, *IEEE Signal Process Mag.*, **38** (2021), 18–44. https://doi.org/10.1109/MSP.2020.3016905

54. M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, et al., Efficient and modular implicit differentiation, preprint, arXiv:2105.15183.

55. Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J. P. Vert, F. Bach, Learning with differentiable pertubed optimizers, preprint, arXiv:2002.08676.

56. C. Wang, D. Jia, W. Wang, D. Ngoduy, B. Peng, J. Wang, A knowledge-informed deep learning paradigm for generalizable and stability-optimized car-following models, *Commun. Transp. Res.*, **5** (2025), 100211. https://doi.org/10.1016/j.commtr.2025.100211

57. H. Chu, W. Zhang, P. Bai, Y. Chen, Data-driven optimization for last-mile delivery, *Complex Intell. Syst.*, **9** (2023), 2271–2284. https://doi.org/10.1007/s40747-021-00293-1

58. X. Tian, R. Yan, Y. Liu, S. Wang, A smart predict-then-optimize method for targeted and cost-effective maritime transportation, *Transp. Res. Part B Methodol.*, **172** (2023), 32–52. https://doi.org/10.1016/j.trb.2023.03.009

59. R. Yan, S. Wang, K. Fagerholt, A semi-"smart predict then optimize"(semi-SPO) method for efficient ship inspection, *Transp. Res. Part B Methodol.*, **142** (2020), 100–125. https://doi.org/10.1016/j.trb.2020.09.014

60. S. Cong, M. Zheng, W. Zhang, Y. Bie, E. Szczepański, Closed-loop coordination of connected and automated vehicles in structured road networks, *Commun. Transp. Res.*, **5** (2025), 100182. https://doi.org/10.1016/j.commtr.2025.100182

61. L. Han, X. Ma, Y. Wang, L. He, Y. Li, L. Zhang, et al., Safe and efficient DRL driving policies using fuzzy logic for urban lane changing scenarios, *J. Intell. Connected Veh.*, **8** (2025), 9210054. https://doi.org/10.26599/jicv.2024.9210054

62. C. Ma, X. Huang, Y. Zhao, T. Wang, B. Du, GRU-LSTM model based on the SSA for short-term traffic flow prediction, *J. Intell. Connected Veh.*, **8** (2025), 9210051. https://doi.org/10.26599/jicv.2024.9210051

63. Y. Hu, N. Kallus, X. Mao, Fast rates for contextual linear optimization, *Manage. Sci.*, **68** (2022), 4236–4245. https://doi.org/10.1287/mnsc.2022.4383

64. A. N. Elmachtoub, H. Lam, H. Zhang, Y. Zhao, Estimate-then-optimize versus integrated-estimation-optimization versus sample average approximation: A stochastic dominance perspective, preprint, arXiv:2304.06833.

65. T. Vanderschueren, T. Verdonck, B. Baesens, W. Verbeke, Predict-then-optimize or predict-and-optimize? An empirical evaluation of cost-sensitive learning strategies, *Inf. Sci.*, **594** (2022), 400–415. https://doi.org/10.1016/j.ins.2022.02.021

66. J. Mandi, J. Kotary, S. Berden, M. Mulamba, V. Bucarey, T. Guns, et al., Decision-focused learning: Foundations, state of the art, benchmark and future opportunities, *J. Artif. Intell. Res.*, **80** (2024), 1623–1701. https://doi.org/10.1613/jair.1.15320

67. S. Fujita, G. Ramey, Exogenous versus endogenous separation, *Am. Econ. J. Macroecon.*, **4** (2012), 68–93. https://doi.org/10.1257/mac.4.4.68

68. B. Hssina, A. Merbouha, H. Ezzikouri, M. Erritali, A comparative study of decision tree ID3 and C4.5, *Int. J. Adv. Comput. Sci. Appl.*, **4** (2014), 13–19. https://doi.org/10.14569/SpecialIssue.2014.040203

69. J. R. Cano, Analysis of data complexity measures for classification, *Expert Syst. Appl.*, **40** (2013), 4820–4831. https://doi.org/10.1016/j.eswa.2013.02.025

70. E. G. Talbi, A taxonomy of metaheuristics for bi-level optimization, in *Metaheuristics for Bi-level Optimization*, Springer, (2013), 1–39. https://doi.org/10.1007/978-3-642-37838-6_1

Appendix

A. Exemplifications of the differences between parameters and hyperparameters

LR. As mentioned in Example 1, the hypothesis class of all ML models is $\mathcal{F}_{\text{LR}} = \{\hat{y} = a^\top x + b \mid (a, b) \in \mathbb{R}^{d+1}\}$ and the set of all configuration elements is denoted by $\mathcal{E}_{\text{LR}} = \{a, b\}$. It is clear that all ML models in $\mathcal{F}_{\text{LR}}$ exhibit the same complexity. Therefore, a and b should be considered as parameters and there are no hyperparameters, i.e., $\mathcal{H}_{\text{LR}} = \emptyset$ and $\mathcal{P}_{\text{LR}} = \{a, b\}$.

kNN. The set of all configuration elements is denoted by $\mathcal{E}_{k\text{NN}} = \{k\}$, where $k \in \mathbb{Z}_{++}$ denotes the number of neighbors, governing the model complexity. When k is small (e.g., $k = 1$), the model tends to be more complex because it closely fits the training data. In this case, the model can be highly sensitive to noise and outliers, which leads to overfitting. Essentially, the model memorizes the training data rather than generalizing well to unseen data. When k is large, the model becomes simpler and tends to underfit. A larger k means that predictions are based on a broader neighborhood, which smooths out noise and outliers, but may also lead to poor model performance if the given data have complex patterns. Therefore, k should be considered as a hyperparameter, i.e., $\mathcal{H}_{k\text{NN}} = \{k\}$ and $\mathcal{P}_{k\text{NN}} = \emptyset$. kNN is a non-parametric ML method, which means that it does not explicitly learn parameters. Instead, the model "remembers" the training data and uses them when making predictions. As such, kNN does not have traditional model parameters.

DT. The set of all configuration elements is denoted by $\mathcal{E}_{\text{DT}} = \{\text{Feature Selections}, \text{Split Points}, \text{Max Depth}, \text{Min Samples Split}, \text{Min Samples Leaf}, ...\}$. "Feature Selections" denote which features to use at each node for splitting. "Split Points" denote the specific threshold values that define the splits for features. "Max Depth" denotes the maximum depth of the tree, where limiting the depth helps control overfitting by preventing the tree from growing too complex. "Min Samples Split" denotes the minimum number of samples required to split an internal node, where higher values prevent the model from creating small, deep branches that may overfit the

training data. "Min Samples Leaf" denotes the minimum number of samples required in a leaf node, where higher values can help smooth the model and avoid overfitting. Therefore, "Max Depth", "Min Samples Split", and "Min Samples Leaf", which govern the model complexity, should be considered as hyperparameters, i.e., $\mathcal{H}_{\text{DT}} = \{\text{Max Depth}, \text{Min Samples Split}, \text{Min Samples Leaf}, ...\}$ and $\mathcal{P}_{\text{DT}} = \{\text{Feature Selections}, \text{Split Points}\}$.

NN. The set of all configuration elements is denoted by $\mathcal{E}_{\text{NN}} = \{\text{Weights}, \text{Biases}, \text{Activation Function}, \text{Number of Hidden Layers}, ...\}$. "Activation Function" determines the output of a neuron for a given input, which introduces non-linearity into the model, allowing it to capture complex patterns in the given data. Different activation functions influence the model complexity. For example, ReLU is widely used in deep networks due to its simplicity and ability to mitigate the vanishing gradient problem, while Sigmoid and Tanh are commonly used in shallow networks, but they can suffer from the vanishing gradient problem. "Number of Hidden Layers" also influences the model complexity. With a greater number of hidden layers, the network typically becomes capable of learning more complex patterns, which also increases its likelihood of overfitting. Therefore, "Activation Function" and "Number of Hidden Layers" should be considered as hyperparameters, i.e., $\mathcal{H}_{\text{NN}} = \{\text{Activation Function}, \text{Number of Hidden Layers}, ...\}$ and $\mathcal{P}_{\text{NN}} = \{\text{Weights}, \text{Biases}\}$.

B. Exemplifications for the evaluation framework for ILO

LR. Consider LR with a one-dimensional target and feature variables for convenience. The hypothesis class of all ML models is denoted by $\mathcal{F}_{\text{LR}} = \{\hat{y} = ax + b \mid (a, b) \in \mathbb{R}^2\}$, and there are no hyperparameters, i.e., $\mathcal{H}_{\text{LR}} = \varnothing$ and $\mathcal{P}_{\text{LR}} = \{a, b\}$. It is clear that $|\mathcal{F}_{\text{LR}}|$ is infinite; hence "it is HARD". An approximate solution approach is described as follows. First, we predetermine two initial values $(a^{\text{init}}, b^{\text{init}})$ under ILO. For example, we can exactly and efficiently solve model $\mathbf{M}_{\text{SLO}}^{\text{single}}$ (i.e., Formula (4.4)) using the least squares algorithm and select the optimal parameter values under SLO as the initial values. Then, we conduct a finite number of searches in specific neighborhoods of a^{init} and b^{init}, which corresponds to a small subset $\mathcal{F}_{\text{LR}}^{\text{sub}} \subset \mathcal{F}_{\text{LR}}$, and identify the best parameter values a^{sub} and b^{sub}, and the corresponding suboptimal ML model $f_{\text{LR}}^{\text{sub}}$ in $\mathcal{F}_{\text{LR}}^{\text{sub}}$ under ILO by approximately solving model $\mathbf{M}_{\text{ILO}}^{\text{bi}}$ (comprising Formulas (4.9), (4.7), and (4.8)). Specifically, let $\epsilon_a \in \mathbb{R}_+$ and $\epsilon_b \in \mathbb{R}_+$ be small numbers with respect to a^{init} and b^{init}, respectively, let $\delta \in \mathbb{Z}_+$ be a counting coefficient, and let $n \in \mathbb{Z}_+$ be the maximum value of δ. Then, a small subset $\mathcal{F}_{\text{LR}}^{\text{sub}}$ to be considered may have the form

$$\mathcal{F}_{\text{LR}}^{\text{sub}} = \{\hat{y} = ax + b \mid a = a^{\text{init}} \pm \delta\epsilon_a,\ b = b^{\text{init}} \pm \delta\epsilon_b,\ \epsilon_a, \epsilon_b \in \mathbb{R}_+,\ \delta = 0, 1, 2, ..., n, \text{ and } n \in \mathbb{Z}_+\}.$$

For example, with $(a^{\text{init}}, b^{\text{init}}) = (1.5, 20)$, $\epsilon_a = 0.1$, $\epsilon_b = 1$, and $n = 3$, the corresponding small subset is given by $\hat{\mathcal{F}}_{\text{LR}}^{\text{sub}} = \{\hat{y} = ax + b \mid a \in \{1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8\},\ b \in \{17, 18, 19, 20, 21, 22, 23\}\}$.

kNN. There are no traditional model parameters for kNN, and thus we have $\mathcal{H}_{k\text{NN}} = \{k\}$ and $\mathcal{P}_{k\text{NN}} = \varnothing$. For a given dataset s_N, it is clear that the cardinality of the set of all possible complexity realizations, i.e., $|\mathcal{C}_{k\text{NN}}|$, which equals N, is finite and relatively small. Furthermore, for each $k \in \mathcal{C}_{k\text{NN}} = \{1, ..., N\}$, the cardinality of the set of all possible ML models is 1, i.e., $|\mathcal{F}_k| = 1$. Thus, we have $|\mathcal{C}_{k\text{NN}}| = |\mathcal{F}_{k\text{NN}}| = N$. Hence, "it is EASY". Therefore, based on $\mathcal{C}_{k\text{NN}}$, we can exactly solve model $\mathbf{M}_{\text{ILO}}^{\text{tri}}$ (comprising Formulas (4.5)–(4.8)), identifying the optimal hyperparameter value k^* and the corresponding optimal ML model $f_{k\text{NN}}^*$ in $\mathcal{F}_{k\text{NN}}$ under ILO. In this case, the middle-level optimization

problem (4.6) is unnecessary and does not need to be solved, as $|\mathcal{F}_k| = 1$ for all $k \in \mathcal{C}_{k\text{NN}}$.

DT. We consider a commonly used DT method with the set of all configuration elements denoted by $\mathcal{E}_{\text{DT}} = \{\text{Feature Selections, Split Points, Max Depth, Min Samples Split, Min Samples Leaf}\}$. Accordingly, we have $\mathcal{H}_{\text{DT}} = \{\text{Max Depth, Min Samples Split, Min Samples Leaf}\}$ and $\mathcal{P}_{\text{DT}} = \{\text{Feature Selections, Split Points}\}$. For a given dataset s_N, it is clear that the cardinality of the set of all possible complexity realizations, i.e., $|\mathcal{C}_{\text{DT}}|$, is finite and relatively small. Specifically, we have $|\mathcal{C}_{\text{DT}}| \leq N^3$, as the possible values for "Max Depth", "Min Samples Split", and "Min Samples Leaf" are all in the set $\{1, ..., N\}$. Thus, the complexity realizations in $\mathcal{C}_{\text{DT}}$ can be enumerated. However, for each complexity realization $c_{\text{DT}} \in \mathcal{C}_{\text{DT}}$, the cardinality of the set of all possible ML models, i.e., $|\mathcal{F}_{c_{\text{DT}}}|$, is generally infinite, making it impossible to enumerate all potential ML models. Hence, "it is HARD". An approximate solution approach is described as follows. First, for each $c_{\text{DT}} \in \mathcal{C}_{\text{DT}}$, we select a small subset $\mathcal{F}^{\text{sub}}_{c_{\text{DT}}} \subset \mathcal{F}_{c_{\text{DT}}}$, and then approximately solve the middle-level optimization problem (4.6) in model $\mathbf{M}^{\text{tri}}_{\text{ILO}}$, identifying a suboptimal ML model $f^{\text{sub}}_{c_{\text{DT}}}$ in $\mathcal{F}^{\text{sub}}_{c_{\text{DT}}}$ under ILO, along with the corresponding parameter values. Then, based on $\mathcal{C}_{\text{DT}}$, we exactly solve the uppermost-level optimization problem (4.5) in model $\mathbf{M}^{\text{tri}}_{\text{ILO}}$, identifying the best (suboptimal) ML model $f^{\text{sub}}_{\text{DT}}$ under ILO among all $f^{\text{sub}}_{c_{\text{DT}}}$ for $c_{\text{DT}} \in \mathcal{C}_{\text{DT}}$, along with the corresponding parameter values.

AIMS Press