Submitted to *INFORMS Journal on Optimization*
manuscript MS-0001-1922.65

Holistic Prescriptive Analytics for Continuous and Constrained Optimization Problems

Dimitris Bertsimas
MIT Sloan School of Management, Cambridge, MA 02139, dbertsim@mit.edu

Omar Skali Lami
MIT Operations Research Center, Cambridge, MA 02139, oskali@mit.edu

We present a holistic framework for prescriptive analytics. Given side data $\boldsymbol{x}$, decisions $\boldsymbol{z}$, and uncertain quantities y, that are functions of $\boldsymbol{x}$ and $\boldsymbol{z}$, we propose a framework that simultaneously predicts y and prescribes the "should be" optimal decisions $\bar{\boldsymbol{z}}$. The algorithm can accommodate a large number of predictive machine learning models, as well as continuous and discrete decisions of high cardinality. It also allows for constraints on these decision variables. We show wide applicability and strong computational performances on synthetic experiments and on two real-world case studies.

Key words: prescriptive, analytics, decision making, regression, classification, optimization, machine learning

1. Introduction

A key objective in Operations Research (OR) is to obtain decisions based on data. The traditional approach in OR is to build models from which we can derive decisions. By and large, data in OR models has only played a supporting role. In contrast, data in Machine Learning (ML) models has played a protagonistic role.

We are given training data $\{\boldsymbol{x}_i, y_i, \boldsymbol{z}_i\}$ and a cost function $c(y_i, \boldsymbol{z}_i)$ for $i \in \{1, \ldots, n\} = [n]$, $n \in \mathbb{N}$, where $\boldsymbol{x}_i \in \mathbb{R}^m$ is the vector of covariates for observation i, $y_i \in \mathbb{R}$ is the outcome, and $\boldsymbol{z}_i \in \mathcal{Z}$ the decision set. Our goal is to prescribe the optimal treatment that minimizes the cost function c for a new observation $\boldsymbol{x}_0$. We introduce the decision variable $\bar{\boldsymbol{z}}_i \in \mathcal{Z}$ for $i \in \{0, \ldots, n\}$ which is the treatment that we prescribe for observation i, in contrast with $\boldsymbol{z}_i \in \mathcal{Z}$ which is the observed treatment, and which is available only for $i \in [n]$, i.e., training data.

Since the outcome of the new observation $\boldsymbol{x}_0$ is unknown, we need to predict it in order to be able to compute its cost, and ultimately prescribe the optimal $\bar{\boldsymbol{z}}_0$. If we are given the function $f(\boldsymbol{x}_0, \bar{\boldsymbol{z}}_0)$,

$\forall \bar{\boldsymbol{z}}_0 \in \mathcal{Z}$, to predict the outcome for $\boldsymbol{x}_0$ under any treatment $\bar{\boldsymbol{z}}_0$, the problem we are trying to solve can be written as in Problem (1).

$$\begin{aligned} \min_{\bar{\boldsymbol{z}}_0} \; & c(f(\boldsymbol{x}_0, \bar{\boldsymbol{z}}_0), \bar{\boldsymbol{z}}_0) \\ \text{s.t. } & \bar{\boldsymbol{z}}_0 \in \mathcal{Z}. \end{aligned} \tag{1}$$

The standard paradigm for this type of prescriptive problems has been to learn f separately within a class of ML functions $\mathcal{F}$ such that it minimizes the mean-squared error on the predictions over the training data as described in Problem (2), then to use it in order to solve Problem (1) in a predict-then-optimize fashion.

$$\begin{aligned} \min_{f} \; & \sum_{i=1}^{n} (y_i - f(x_i, z_i))^2 \\ \text{s.t. } & f \in \mathcal{F}. \end{aligned} \tag{2}$$

Note that what we mean by minimizing over f in Problem (2) is minimizing over a finite number of parameters that fully define the function f, for example the weights $\boldsymbol{W}$ of the neurons if f is a neural network, the design T of the tree if f is a decision tree, or the coefficients $\boldsymbol{\beta}$ of the regression if f is a linear, polynomial or convex regression.

However, going beyond the standard predict-then-optimize paradigm, recent efforts have tried to go directly from data to decisions using ML methods. Bertsimas and Kallus (2020) propose a framework that can accommodate a large number of ML algorithms, works for both continuous and discrete variables, and can also accommodate constraints, but still utilizes the framework of first predicting, and then prescribing in a sequential fashion. Bertsimas et al. (2019) propose an extension of Optimal Classification and Regression Trees (Bertsimas and Dunn (2017)) called Optimal Prescriptive Trees (OPTs), that simultaneously predicts and prescribes. While the method is powerful and interpretable, it is limited by that the decisions need to be discrete and small in cardinality, and cannot be constrained.

In this paper, we propose a generalization of OPTs to arbitrary ML methods, that simultaneously predicts and prescribes, can accommodate continuous decisions as well as discrete decisions of high cardinality, and also allow constraints on the decision variables.

To prescribe the optimal treatment $\bar{\boldsymbol{z}}_0$, we propose a framework for prescriptive analytics that jointly regroups observations into clusters with similar behaviors, learns a predictive model over each of these clusters and prescribes the optimal decisions under constraints. The intuition behind this framework is that clustering allows to achieve higher accuracy by aggregating data into clusters

and to divide the training process into smaller subproblems, while performing all the tasks (clustering, prediction and prescription) jointly allows us to find the right trade-off between accurate predictions and optimal decisions, in a tractable way that can account for continuous and constrained decision spaces. This framework can be used with a wide range of predictive ML methods and cost functions for the prescriptions, with both constrained and unconstrained problems, as well as with a decision space that can be infinite or even continuous. We show that the proposed method, which we will refer to as **Holistic Prescriptive Analytics method** or **HPA**, is scalable, and that it significantly improves state-of-the-art performance.

Like OPTs, our framework also provides interpretability when the ML methods used are interpretable, which can be vital in some applications (See Section 6 for a Case Study).

1.1. Literature Review

The standard paradigm in real-world analytics problems involving prediction and optimization is predict-then-optimize, where ML tools are used to predict a point estimate of an uncertain quantity, that is then plugged in a nominal optimization problem to solve for the optimal decisions.

Due to the poor performances of this paradigm, researchers have tried to account for the variability of the prediction by solving the optimization problem over the expected value of the prescription cost, and by approximating this expected value by the empirical sample average (Sample Average Approximation SAA), see for example Birge and Louveaux (2011). On the other hand, researchers proposed to use Robust Optimization instead of Stochastic Optimization to solve for the optimal decisions once the prediction is made, see for example Ben-Tal et al. (2009) and Bertsimas et al. (2018). In this case, the uncertainty of the prediction is accounted for by considering the worst-case scenario within an uncertainty set.

Kallus et al. (2018) combines both approaches by proposing a more robust estimation of SAA using propensity scoring, while other approaches have tried to account for prescriptive cost in some way during the prediction. Tulabandhula and Rudin (2013) for example minimize a loss function that combines the prediction error with the operational cost. In Kao et al. (2009) and Ban and Rudin (2018), the ML models are trained to directly predict the optimal solution of the nominal optimization problem. Elmachtoub and Grigas (2017) extends the predict-then-optimize framework into a Smart Predict-then-Optimize (SPO) by introducing an SPO loss function which measures the decision error induced by a prediction as the new objective function for the prediction phase. This prediction is then used as a point estimate in the optimization phase in the same way as the standard predict-then-optimize.

Bertsimas and Kallus (2020) move away from this paradigm and use a framework that leverages previously trained ML models in a prescriptive optimization problem without actually using the

point estimate by optimizing over a weighted combination of training samples instead with the weights computed based on the ML methods trained for the prediction. Bertsimas et al. (2019) introduce OPTs, which change the objective function of the Optimal Classification and Regression Trees Bertsimas and Dunn (2017, 2019) to account for the prescription cost, and hence making the prediction and the optimal decisions at the same time instead of sequentially. However, the decisions need to be discrete and small in cardinality and cannot be constrained, due to the complexity of the formulation of the prescriptive tree model.

1.2. Contributions

Motivated by these recent efforts in advancing the field of prescriptive analytics and the limitations of some of the existing algorithms, we develop a prescriptive framework that

1. Can accommodate an arbitrary ML method for prediction.
2. Simultaneously predicts and prescribes, moving away from the predict-then-optimize paradigm.
3. Allows continuous and constrained decisions.
4. Allows the uncertain outcome y to be also a function of the decisions $\boldsymbol{z}$.
5. Regroups the observations into clusters of similar behavior which enables us to achieve higher accuracy by aggregating data into clusters and to divide the training process into smaller subproblems for tractability purposes.
6. Has strong computational performance and wide applicability for both synthetic and real world datasets.

We test different versions of this framework and perform extensive numerical experiments to benchmark the predictive accuracy, prescriptive cost, scalability and interpretability of the framework (see Section 6). We perform these experiments on synthetic data, i.e., generated data where the underlying behavior, and hence the counterfactuals, are known, and on two real-world case studies: a diabetes case study where we prescribe an optimal treatment in order to minimize the expected average blood sugar levels of patients, and an assortment optimization case study where we optimize display for a food retailer in order to maximize revenue. In the former case study we use an interpretable approach due to the medical nature of the application. In these experiments, we show that the HPA framework is scalable and provides a performance edge over alternative methods. Across the 3 computational experiments that we detail in this paper, with datasets up to 100,000 observations and about 100 features, we observe between 14% and 30% improvement in terms of out-of-sample prescriptive cost compared to the predict-then-optimize baseline, and between 5% and 17% improvement compared to the state-of-the-art predictive-prescriptive method (Bertsimas and Kallus (2020)).

2. Proposed Approach

The first idea of our approach combines Problem (1) and Problem (2) by jointly learning the predictive function and prescribing the optimal treatment. This is done by minimizing over f and $\bar{\boldsymbol{z}}$ the weighted average of the prediction error on the training data and the prescriptive cost on both the training data and the new data point $\boldsymbol{x}_0$. We denote $\lambda \in [0,1]$ the weight of the prediction error and $(1-\lambda)$ the weight of the prescriptive cost. We obtain Problem (3).

$$\begin{aligned} \min_{f,\bar{\boldsymbol{z}}} \ & \lambda \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} c(f(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\ \text{s.t} \ & \bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n \in \mathcal{Z}, \end{aligned} \tag{3}$$

Note that the first part of the objective function of Problem (3) is exactly the objective function of Problem (2), weighted by λ. The second part of the objective function of Problem (3) is just the cost function used in Problem (1), weighted by $1-\lambda$, and applied to both the new data point $i=0$ and the training data $i \in [n]$ in order to account for the entire prescriptive cost, even though the final objective is just to get the optimal treatment $\bar{\boldsymbol{z}}_0$ only.

Our overall approach extends Problem (3) by clustering each observation $i \in \{0, \ldots, n\}$ into a cluster $j \in [k]$, where k is a predefined number of clusters. Then, within each cluster j, we train a different predictive model f_j to predict outcomes based on covariates and treatments. Since we do not have any outcome for observation $\boldsymbol{x}_0$, we assign it to the cluster that minimizes the average distance in terms of features of its points to $\boldsymbol{x}_0$. Here, we take the ℓ_2-norm, but any other tractable distance can be considered. The key of our approach is that all three steps (clustering, prediction and prescription) are done jointly.

We also define the following additional notation:

(a) $u_{i,j}$: the binary outcome of whether observation i is assigned to cluster j.

(b) f_j: a function that learns the prediction of the outcome y based on the covariates $\boldsymbol{x}$ and the observed prescriptions $\boldsymbol{z}$.

(c) N_{min}: hyper-parameter defining the minimum number of training observations in each cluster.

Problem (3), with the additional layer of clustering becomes Problem (4).

$$
\begin{aligned}
\min_{\boldsymbol{u},\boldsymbol{f},\bar{\boldsymbol{z}}} \ & \lambda \sum_{i=1}^{n}\sum_{j=1}^{k} u_{i,j}(y_i - f_j(\boldsymbol{x}_i,\boldsymbol{z}_i))^2 + (1-\lambda)\sum_{i=0}^{n}\sum_{j=1}^{k} u_{i,j}c(f_j(\boldsymbol{x}_i,\bar{\boldsymbol{z}}_i),\bar{\boldsymbol{z}}_i) \\
\text{s.t} \ & \bar{\boldsymbol{z}}_0,\ldots,\bar{\boldsymbol{z}}_n \in \mathcal{Z} \ \text{(constraints on decisions)}, \\
& \sum_{i=1}^{n} u_{i,j} \geq N_{min}, \ \forall j \in [k] \ \text{(minimum number of training observations per cluster)}, \\
& \sum_{j=1}^{k} u_{i,j} = 1, \ \forall i \in \{0,\ldots,n\} \ \text{(one observation is assigned to exactly one cluster)}, \\
& u_{0,j} \leq \mathbb{1}\left(j = \underset{j' \in [k]}{\arg\min} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k] \ (\boldsymbol{x}_0 \text{ is assigned to the closest cluster}).
\end{aligned}
\tag{4}
$$

Note that the last constraint of Problem (4) enforcing that $\boldsymbol{x}_0$ is assigned to the closest cluster is an inequality instead of an equality to ensure that the problem remains feasible if several clusters satisfy the closeness condition.

The clustering has two main advantages: increasing the predictive accuracy of simpler tractable models (for example training a Linear Regression model on the entire dataset yields worse results than training a different Linear Regression model on each subgroup with similar underlying behaviors), and reducing the size of the training set for the different models by training them into smaller batches.

To highlight the fact that $\mathcal{Z}$ can be constrained, we replace, without loss of generality, $\mathcal{Z}$ by $\{(\boldsymbol{z}_0,\ldots,\boldsymbol{z}_n) \mid h_r(\boldsymbol{z}_0,\ldots,\boldsymbol{z}_n) \leq b_r, \ r \in [s]\}$ for some $s \in \mathbb{N}$. These constraints on the decisions can for example be budget constraints. Our proposed approach can then be summarized in Problem (5).

$$
\begin{aligned}
\min_{\boldsymbol{u},\boldsymbol{f},\bar{\boldsymbol{z}}} \ & \lambda \sum_{i=1}^{n}\sum_{j=1}^{k} u_{i,j}(y_i - f_j(\boldsymbol{x}_i,\boldsymbol{z}_i))^2 + (1-\lambda)\sum_{i=0}^{n}\sum_{j=1}^{k} u_{i,j}c(f_j(\boldsymbol{x}_i,\bar{\boldsymbol{z}}_i),\bar{\boldsymbol{z}}_i) \\
\text{s.t} \ & h_r(\bar{\boldsymbol{z}}_0,\ldots,\bar{\boldsymbol{z}}_n) \leq b_r, \ \forall r \in [s] \ \text{(constraints on decisions)}, \\
& \sum_{i=1}^{n} u_{i,j} \geq N_{min}, \ \forall j \in [k] \ \text{(minimum number of training observations per cluster)}, \\
& \sum_{j=1}^{k} u_{i,j} = 1, \ \forall i \in \{0,\ldots,n\} \ \text{(one observation is assigned to exactly one cluster)}, \\
& u_{0,j} \leq \mathbb{1}\left(j = \underset{j' \in [k]}{\arg\min} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k] \ (\boldsymbol{x}_0 \text{ is assigned to the closest cluster}).
\end{aligned}
\tag{5}
$$

2.1. Solving the General Problem

In this section, we outline Algorithm 1 for HPA. We assume that optimizing over the predictive function $\boldsymbol{f}$ involves a collection of parameters. We cover in Algorithm 1 the case of learning models that are fully defined by a set of parameters of fixed size which can be globally optimized over a custom loss function. This definition includes, but is not limited to neural networks, where the parameters are the weights of the neurons $\boldsymbol{W}_j$, optimal trees, where the parameters are the design of the tree T_j and linear, polynomial, and convex regression, where the parameters are the coefficients $\boldsymbol{\beta}_j$ of the regression for each cluster j. When we say optimize over the function f_j, we mean optimize over the finite parameters that define f_j.

We propose an alternative algorithm for non-parametric and black-box models that minimize the mean-squared error loss in Section 3 and expand extensively on the cases of decision trees and linear regression in Section 4 and Section 5, respectively.

We use a local search iterative approach with multiple restarts to solve Problem (5). We start with a warm start $\boldsymbol{u}^{(1)}$ and $\bar{\boldsymbol{z}}^{(1)}$ for $\boldsymbol{u}$ and $\bar{\boldsymbol{z}}$. Then, for $t \in [N_{max}-1]$, we fix $\boldsymbol{u}^{(t)}$ and $\bar{\boldsymbol{z}}^{(t)}$ to find $\boldsymbol{f}^{(t+1)}$ as in Equation (6), then we fix $\boldsymbol{u}^{(t)}$ and $\boldsymbol{f}^{(t+1)}$ to find $\bar{\boldsymbol{z}}^{(t+1)}$ as in Equation (7), and finally we fix $\bar{\boldsymbol{z}}^{(t+1)}$ and $\boldsymbol{f}^{(t+1)}$ to find $\boldsymbol{u}^{(t+1)}$ as in Equation (8). We describe the Algorithm in Algorithm 1.

Algorithm 1 IterativeSolve($\boldsymbol{X}$,$\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{u}^{(1)}$,$\bar{\boldsymbol{z}}^{(1)}$, N_{min}, N_{max}, k)

for $t \in [N_{max}-1]$ **do**

for $j \in [k]$ **do**

$$f_j^{(t+1)} = \arg\min_{f_j} \lambda \sum_{i=1}^{n} u_{i,j}^{(t)} (y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} u_{i,j}^{(t)} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i^{(t)}), \bar{\boldsymbol{z}}_i^{(t)}), \quad (6)$$

end for

$$\bar{\boldsymbol{z}}^{(t+1)} = \arg\min_{\bar{\boldsymbol{z}}} \sum_{i=0}^{n} \sum_{j=1}^{k} u_{i,j}^{(t)} c(f_j^{(t+1)}(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \text{ s.t } h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \forall r \in [s], \quad (7)$$

$$\boldsymbol{u}^{(t+1)} = \arg\min_{\boldsymbol{u}} \lambda \sum_{i=1}^{n} \sum_{j=1}^{k} u_{i,j} (y_i - f_j^{(t+1)}(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} \sum_{j=1}^{k} u_{i,j} c\left(f_j^{(t+1)}(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i^{(t+1)}), \bar{\boldsymbol{z}}_i^{(t+1)}\right) \quad (8)$$

$$\text{s.t } \sum_{i=1}^{n} u_{i,j} \geq N_{min} \quad \forall j \in [k]$$

$$\sum_{j=1}^{k} u_{i,j} = 1, \quad \forall i \in \{0, \ldots, n\}$$

$$u_{0,j} \leq \mathbb{1}\left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k],$$

end for

Return $f_1^{(N_{max})}, f_2^{(N_{max})}, \ldots, f_k^{(N_{max})}$, $\bar{\boldsymbol{z}}^{(N_{max})}$, $\boldsymbol{u}^{(N_{max})}$.

Algorithm 1 converges to a local minimum of Problem (5), which is why we use multiple restarts. We can get warm starts $\bar{\boldsymbol{z}}^{(1)}$ and $\boldsymbol{u}^{(1)}$ by setting $\bar{\boldsymbol{z}}_i^{(1)}$ to $\boldsymbol{z}_i$ for $i \in [n]$ and $\bar{\boldsymbol{z}}_0^{(1)}$ at random, and getting $\boldsymbol{u}^{(1)}$ from a clustering method such as k-means on the $\boldsymbol{x}_i$'s. Note that Problem (8) can be

solved optimally similarly to Bertsimas et al. (2020), or solved greedily by first optimizing over the values of $\boldsymbol{u}_{i,j}^{(t+1)}$ for $i \in [n]$ and $j \in [k]$ then assigning $\boldsymbol{x}_0$ to the closest cluster in average.

3. Quadratic Cost Function

When the cost function can be written $c(y, \bar{\boldsymbol{z}}) = y^2$, we can solve Problem (6) more efficiently. In this case, we outline Algorithm 2 for HPA, which generalizes to any ML method that minimizes the mean-squared error loss, regardless of whether or not it can formulated as an explicit optimization problem. This includes for example greedy, non-parametric models such as CART (Section 5) and Random Forests, and black-box models, in addition to the class of models covered by Algorithm 1. In this section, optimizing over function f_j means selecting the "best" f_j with regards to the mean-squared error on a dataset that we define, within a selected class of functions. We further discuss conditions for tractability in the complexity analysis.

In Algorithm 2, instead of changing the objective function of the ML method f_j for each cluster j to get Problem (6), which is hard to do for some ML methods such as CART (Breiman et al. (1984)) or even some complex neural network structures (LeCun et al. (2015)), we change the dataset over which we minimize regular mean-squared error loss. If we assume λ is rational, i.e. can be written $\frac{p}{p+q}$, $p, q \in \mathbb{Z}_+$, with p and q preferably small, we create p copies of every training observation i in cluster j to account for the predictive part of the objective function as in Problem (9). Then we create q modified copies of every observation i in cluster j where we set the target variable to 0 to get exactly the prescriptive part of the objective function as in Problem (10).

Algorithm 2 PredictiveFit($\boldsymbol{X}$,$\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{l}$, $\bar{\boldsymbol{z}}$, k, p, q)

Let $\lambda = \frac{p}{p+q}$, $p, q \in \mathbb{Z}_+$.

for $j \in [k]$ **do**

Create p copies of every training observation i in cluster j: $\boldsymbol{x}_i$, $\boldsymbol{z}_i$, y_i, (9)

Create q modified copies of every observation i in cluster j: $\boldsymbol{x}_i$, $\bar{\boldsymbol{z}}_i$, 0, (10)

Solve the regular mean-squared error model fitting for f_j on this new dataset,

i.e. $f_j = \arg\min_f \sum_{i'=1}^{p\bar{n}_j + qn_j} (y'_i - f(\boldsymbol{x}'_i, \boldsymbol{z}'_i))^2$ with the notations that variables with a " $'$ " superscript indicate variables in the new created dataset, n_j the number of observations in cluster j and $\bar{n}_j$ the number of training observations in cluster j.

end for

Return $f_1, f_2, \ldots, f_k$.

Proposition: The output of Algorithm 2, and that of Problem (6) are equal.

Proof: Let $j \in [k]$ a cluster with n_j observations and $\bar{n}_j$ training observations. The minimization problem in Algorithm 2 is equivalent to Equation (11).

$$\begin{aligned}
&\min_{f_j} \sum_{i'=1}^{p\bar{n}_j+qn_j} (y'_i - f_j(\boldsymbol{x}'_i, \boldsymbol{z}'_i))^2 \\
\iff &\min_{f_j} \sum_{i=1}^{\bar{n}_j} p(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + \sum_{i=1}^{n_j} q(0 - f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i))^2 \\
\iff &\min_{f_j} \sum_{i=1}^{\bar{n}_j} \frac{p}{p+q}(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + \sum_{i=1}^{n_j} \frac{q}{p+q}(0 - f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i))^2 \\
\iff &\min_{f_j} \sum_{i=1}^{\bar{n}_j} \lambda(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + \sum_{i=1}^{n_j} (1-\lambda)(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i))^2 \\
\iff &\min_{f_j} \sum_{i=1}^{\bar{n}_j} \lambda(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + \sum_{i=1}^{n_j} (1-\lambda) c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\
\iff &\min_{f_j} \lambda \sum_{i=1}^{n} u_{i,j}(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} u_{i,j} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i)
\end{aligned} \tag{11}$$

Which proves the equivalence. ■

Complexity of the algorithm

In order to bound the complexity of Algorithm 1 in the quadratic case, we assume that Problem (12), which minimizes the mean-squared error loss for each ML method f_j over a dataset of size n can be solved with complexity $\mathcal{O}(q_1(n))$.

$$\min_{f_j} \sum_{i=1}^{n} (y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2, \tag{12}$$

We also assume that Problem (13) which minimizes, given f_j and x, the prescriptive cost for decisions z over a dataset of size $n+1$ can be solved with complexity $\mathcal{O}(q_2(n))$.

$$\begin{aligned}
&\min_{\bar{\boldsymbol{z}}} \sum_{i=0}^{n} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\
&\text{s.t. } h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \ \forall r \in [s],
\end{aligned} \tag{13}$$

We have that complexity of solving Problem (6) at each time t which consists of fitting $f_1, \ldots, f_k$ using Algorithm 2 is $\mathcal{O}(\sum_{j=1}^{k} q_1((p+q) \times n_j))$, where n_j is the number of observations in cluster j.

Complexity of solving Problem (7) at each time t is $\mathcal{O}(q_2(n))$, while Problem (8) is an assignment problem, completely independent of the complexity of the ML model or that of the constraints on the decisions, its complexity is negligible compared to the 2 previous steps.

So we have a total complexity of q_T bounded by $O(N_{max}(kq_1((p+q)\times n)+q_2(n))$. On average, for evenly-distributed clusters, this complexity becomes $q_T = O(N_{max}(kq_1((p+q)\times \frac{n}{k})+q_2(n))$, which is tractable for reasonably small p and q.

4. Linear Parametrization

We can further improve the efficiency of solving Problem (5) for particular ML methods. In this section, we discuss the case where f_j is linear for each cluster $j \in [k]$, i.e. there exist coefficients $\boldsymbol{\beta}_j$ and γ_j such that $f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i) = \boldsymbol{\beta}_j^T \boldsymbol{x}_i - \gamma_j \bar{\boldsymbol{z}}_i \ \forall i \in [n]$. Problem (5) can be then rewritten to get Problem (14).

$$
\begin{aligned}
\min_{\boldsymbol{u},\boldsymbol{\beta},\boldsymbol{\gamma},\bar{\boldsymbol{z}}} \ & \lambda \sum_{i=1}^{n}\sum_{j=1}^{k} u_{i,j}(y_i - \boldsymbol{\beta}_j^T \boldsymbol{x}_i - \gamma_j \boldsymbol{z}_i)^2 + (1-\lambda)\sum_{i=0}^{n}\sum_{j=1}^{k} u_{i,j}(\boldsymbol{\beta}_j^T \boldsymbol{x}_i + \gamma_j \bar{\boldsymbol{z}}_i)^2 \\
\text{s.t} \ & h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \forall r \in [s], \\
& \sum_{i=1}^{n} u_{i,j} \geq N_{min} \ \ \forall j \in [k], \\
& \sum_{j=1}^{k} u_{i,j} = 1, \ \ \forall i \in \{0, \ldots, n\} \ , \\
& u_{0,j} \leq \mathbb{1}\left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k]
\end{aligned}
\tag{14}
$$

For the linear parametrization, Problem (6) can easily be formulated as a tracatable optimization problem, to obtain Problem (15), which can be solved directly. The corresponding iterative procedure to solve Problem (14) is described in Algorithm 3, which is similar, but more efficient than Algorithm 1.

Algorithm 3 IterativeSolveLinear($\boldsymbol{X}$,$\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{u}^{(1)}$,$\bar{\boldsymbol{z}}^{(1)}$, N_{min}, N_{max}, k)

for $t \in [N_{max}-1]$ **do**

for $j \in [k]$ **do**

$$\boldsymbol{\beta}_j^{(t+1)}, \gamma_j^{(t+1)} = \arg\min_{\boldsymbol{\beta}_j, \gamma_j} \lambda \sum_{i=1}^{n} u_{i,j}^{(t)} (y_i - \boldsymbol{\beta}_j^T \boldsymbol{x}_i - \gamma_j \boldsymbol{z}_i)^2 + (1-\lambda) \sum_{i=0}^{n} u_{i,j}^{(t)} (\boldsymbol{\beta}_j^T \boldsymbol{x}_i + \gamma_j \bar{\boldsymbol{z}}_i^{(t)})^2, \quad (15)$$

end for

$$\bar{\boldsymbol{z}}^{(t+1)} = \arg\min_{\bar{\boldsymbol{z}}} \sum_{i=0}^{n} \sum_{j=1}^{k} u_{i,j}^{(t)} ((\boldsymbol{\beta}_j^{(t+1)})^T \boldsymbol{x}_i + \gamma_j^{(t+1)} \bar{\boldsymbol{z}}_i)^2 \text{ s.t } h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \forall r \in [s],$$

$$\boldsymbol{u}^{(t+1)} = \arg\min_{\boldsymbol{u}} \sum_{j=1}^{k} \left[\sum_{i=1}^{n} u_{i,j} \lambda (y_i - (\boldsymbol{\beta}_j^{(t+1)})^T \boldsymbol{x}_i - \gamma_j^{(t+1)} \boldsymbol{z}_i)^2 + \sum_{i=0}^{n} u_{i,j} (1-\lambda) ((\boldsymbol{\beta}_j^{(t+1)})^T \boldsymbol{x}_i + \gamma_j^{(t+1)} \bar{\boldsymbol{z}}_i^{(t+1)})^2 \right]$$

$$\text{s.t } \sum_{i=1}^{n} u_{i,j} \geq N_{min} \quad \forall j \in [k]$$

$$\sum_{j=1}^{k} u_{i,j} = 1, \quad \forall i \in \{0, \ldots, n\}$$

$$u_{0,j} \leq \mathbb{1} \left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k],$$

end for

Return $\boldsymbol{\beta}_1^{(N_{max})}, \boldsymbol{\beta}_2^{(N_{max})}, \ldots, \boldsymbol{\beta}_k^{(N_{max})}$, $\gamma_1^{(N_{max})}, \gamma_2^{(N_{max})}, \ldots, \gamma_k^{(N_{max})}$, $\bar{\boldsymbol{z}}^{(N_{max})}$, $\boldsymbol{u}^{(N_{max})}$.

Note that this approach can also be applied to regularized linear regression, such as Ridge Regression or Elastic Net.

5. Tree-Based Parametrization

In this section we set f_j for $j \in [k]$ to be equal to a tree ML method, such as CART (Breiman et al. (1984)) or Optimal Trees (Bertsimas and Dunn (2017)). We apply Algorithm 1 and Algorithm 2 to this parametrization.

In order to solve Problem (6), i.e. learning the predictive function $f_j^{t+1} = \arg\min_{f_j} \lambda \sum_{i=1}^{n} u_{i,j}^{(t)} (y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} u_{i,j}^{(t)} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i^{(t)}), \bar{\boldsymbol{z}}_i^{(t)})$ for each cluster j at iteration t, we use Algorithm 2 as described in Section 3. Since Algorithm 2 only requires to minimize the mean-squared error loss over a new dataset, it can be used with both the globally optimal trees, or with the iterative CART algorithm.

For Problem (7), there are two approaches, depending on whether the problem is constrained or not:

1. If the problem is unconstrained, then we can find an optimum just by iterating over each observation $i \in [1, n]$ independently. For each i, we find all the possible leaves of the trained tree to which this observation can belong, and pick one treatment that minimizes the corresponding cost. We note that even though this number of leaves is theoretically bounded by 2^d where d is the depth of the tree, it is usually extremely small, as it corresponds to the number of splits on the treatment $\bar{\boldsymbol{z}}_i$ for the region of space where X_i, i.e. the covariates of observations i, belongs. Even in the case where the treatment space is large or continuous, the fact that the cost function is a

function of the outcome y_i only, which is itself a tree-based function of X_i and $\bar{\boldsymbol{z}}_i$, the structure of the tree makes the minimization computationally tractable: any $\bar{\boldsymbol{z}}_i$ that verifies the constraints that would lead to the optimal leaf can be chosen as the optimal solution.

2. If the problem is constrained, then it is necessary to use heuristics, or formulate Problem (7) as an Mixed-Integer Program. The exact formulation is given by Bertsimas et al. (2019). In our case however, the problem is simpler because it does not require to retrain the tree, hence reducing the number of variables of the problem by several orders of magnitude.

Problem (8) is again an assignment problem that does not depend on the chosen ML method.
We also note that once this model is trained, the decision-making process that outputs an optimal decision $\bar{\boldsymbol{z}}$ given covariates $\boldsymbol{x}$, can be described with a simple decision tree, which makes it interpretable. In Section 6, we show computational results for the tree-based formulation applied in a medical setting where interpretability is particularly relevant.

6. Experimental Results

We perform extensive experiments for both the linear and the tree-based parametrizations (applied with Classification and Regression Trees - Breiman et al. (1984)). In the following tests, we use multiple restarts for the local-search algorithm.
For the real-world data, since the counterfactuals are unknown, we evaluate the results with a combination of XGBoost (Chen and Guestrin (2016)) and Feed-Forward Neural Networks (LeCun et al. (2015)), that will be considered ground-truth for the underlying behavior.

6.1. Linear Parametrization

We test the linear parametrization on synthetic data and on an assortment problem in a real-world case study, against the predict-then-optimize framework (with linear regression), sample average approximation (SAA Birge and Louveaux (2011)) and linear predictive-prescriptive (Bertsimas and Kallus (2020)).

Synthetic Data We simulate a group of $n \in \mathbb{N}$ patients. Each patient $i \in [n]$ is characterized by $p \in \mathbb{N}$ features and reacts differently to a binary treatment $z_i \in \{0,1\}$ that affects a target variable $y_i \in \mathbb{R}$ that we need to predict and that we want to minimize. We note $\boldsymbol{x}_i \in [0,1]^p$ the vector of features for patient i.
We have $r \in \mathbb{N}$ different reactions to the treatment, for each type of reaction $j \in [r]$, we set $y_i = \boldsymbol{\beta_j}^T \boldsymbol{x}_i + \gamma_j z_i$ if and only if i belongs to group j, where $\boldsymbol{\beta_j} \in \mathbb{R}^p$ and $\gamma_j = 10 \cdot ((-1)^j) \cdot (j+1)$. Note that the direction and the magnitude of the effect of z_i on y_i differs in each group.
We set $r = 5$ and $p = 4$. Then, we create the groups from the following tree structure:

- i belongs to group 1 $\iff x_{i,1} \leq 0.5$ and $x_{i,2} \leq 0.5$.
- i belongs to group 2 $\iff x_{i,1} \leq 0.5$ and $x_{i,2} > 0.5$
- i belongs to group 3 $\iff x_{i,1} > 0.5$ and $x_{i,3} \leq 0.33$
- i belongs to group 4 $\iff x_{i,1} > 0.5$, $x_{i,3} > 0.33$ and $x_{i,4} \leq 0.66$
- i belongs to group 5 $\iff x_{i,1} > 0.5$, $x_{i,3} > 0.33$ and $x_{i,4} > 0.66$

We draw patients uniformly from to 5 groups, and then we draw the features of the patient uniformly and independently within the bounds that define each group. We minimize Mean Squared Error (MSE) for the predictive cost, and $c(y, z) = y^2$ for the prescriptive cost, similarly to Section 3.

We take the predict-then-optimize approach as a baseline, and test it against our HPA approach, while varying the number of clusters k and selecting the best one in a validation set. We obtain the results in Table 1 for training data and Table 2 for unseen, testing data, for this best value of k.

Model	**Training MSE**	**In-Sample R^2**	**Prescriptive Cost**	**Difference vs Baseline**
Linear Regress-then-Prescribe	172546	78.4%	12490	0%
Sample Average Approximation	172546	78.4%	12146	-3%
Linear Predictive-Prescriptive	177045	77.9%	11912	-5%
Linear HPA	190831	76.1%	9315	-25%
Linear HPA w/ Penalization	196943	75.4%	9306	-25%
Linear HPA w/ Multiple Restarts	**171094**	**78.6%**	8706	-30%
Linear HPA w/ Pen. & Multiple Restarts	180244	77.5%	**8166**	**-35%**

Table 1 Results of the Synthetic Data simulation (Minimization) - Training.

We notice that the HPA method substantially outperforms the other linear prescriptive methods both in terms of prediction accuracy and most importantly in terms of prescription cost with over 30% improvement compared to baseline in both training and testing data.

In addition to that, Figure 1 (train) and Figure 2 (test) summarize the results for all tested number of clusters $k \in [1, 10]$.

Model	**Testing MSE**	**Out-of-Sample R^2**	**Prescriptive Cost**	**Difference vs Baseline**
Linear Regress-then-Prescribe	184771	76.9%	11896	0%
Sample Average Approximation	184771	76.9%	11189	-6%
Linear Predictive-Prescriptive	208937	73.9%	10350	-13%
Linear HPA	200743	74.9%	9380	-21%
Linear HPA w/ Penalization	195601	75.5%	9250	-22%
Linear HPA w/ Multiple Restarts	183144	77.1%	8622	-28%
Linear HPA w/ Pen. & Multiple Restarts	**178244**	**77.7%**	**8351**	**-30%**

Table 2 Results of the Synthetic Data simulation (Minimization) - Testing

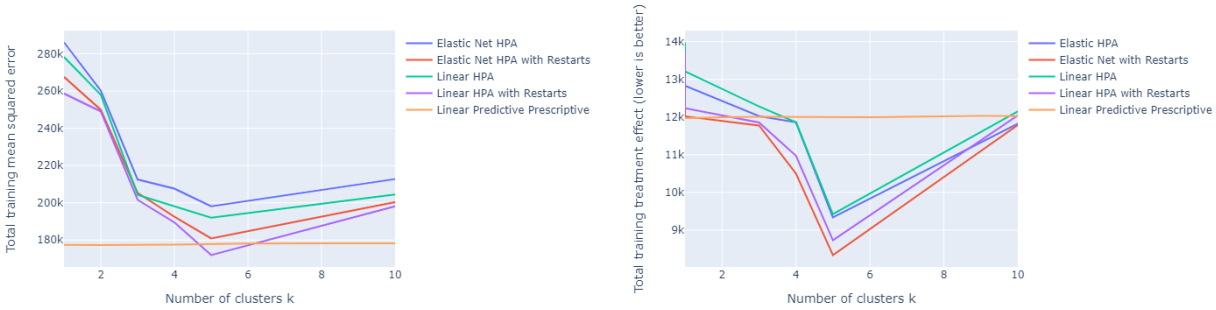

Figure 1 **For top models: in-sample MSE on Synthetic Simulation (on left), and in-sample Treatment on Synthetic Simulation (Lower is better on right.**

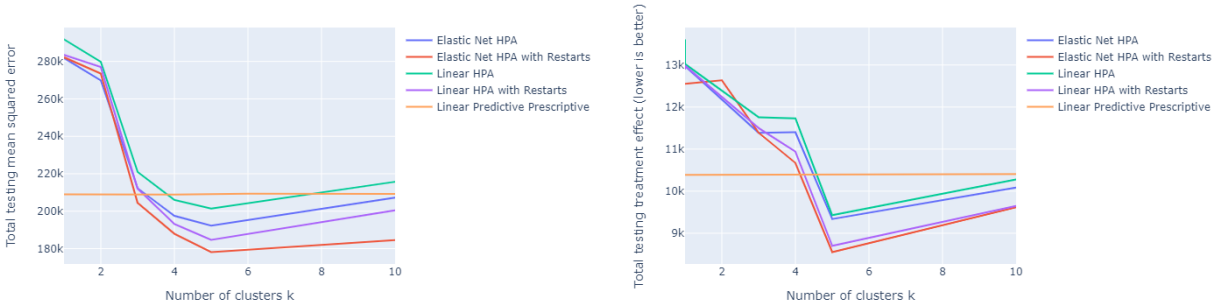

Figure 2 **For top models: out-of-sample MSE on Synthetic Simulation (on left), and out-of-sample Treatment on Synthetic Simulation (Lower is better) on right. Our framework outperforms other benchmarks and recovers the correct number of clusters.**

Figure 1 and Figure 2 outline that our framework was able to recover the correct number of clusters $k = 5$ from the data samples. By looking at the clusters with the majority of observations from the initial groups and matching them accordingly, we also observe that more than 91% of the observations a classified correctly (i.e. in the correct group) with the HPA method, and that without imposing any tree structure.

Assortment Case Study The dataset consists of 25,000 observations, each one corresponding to a product sales at a particular date with:

- **Products Features**: ex. type of product, brand, whether it is perishable or not.
- **Pricing Decisions**: price, whether it is promotion or not.
- **Time-series Data**: historical sales for the product.
- **External Indicators**: market features, competitors' prices.
- **Decision Variable**: whether to display the product or not.
- **Target Variable**: number of units sold for this product SKU at this time t.

The goal is to select products for the display of the retailer to maximize revenue. We assume that the target variable is representative of the demand as long as the product is displayed (no inventory limitation) and we ignore the cannibalisation effect of the products (i.e. the product demand only depends on the features and not on the selected portfolio). Note that this is a constrained optimization problem, since the shelf space is limited. We summarize the results in Table 3.

Model	**Out-of-Sample R^2**	**Prescriptive Cost**	**Improvement From Baseline**
Linear Regress-then-Prescribe	74 %	1404.93	0%
Sample Average Approximation	74 %	1545.79	10%
Linear Predictive-Prescriptive	73 %	1580.13	12%
Linear HPA	71 %	1508.7	7%
Linear HPA w/ Penalization	75 %	1597.22	14%
Linear HPA w/ Multiple Restarts	79 %	1641.99	17%
Linear HPA w/ Pen. & Multiple Restarts	**83 %**	**1643.42**	**17%**

Table 3 Results for Invetory Dataset (Maximization)

On this real-world experiment, the linear HPA outperforms its competitors in both out-of-sample accuracy and prescriptive cost. We might reasonably assume that the improvement in predictive accuracy comes mainly from the fact that the clustering allows the model to capture non-linear relationships (that are linear within each cluster but not linear over all the dataset), but that the improvement in the predictive cost comes from performing the clustering, the prediction and the prescription jointly instead of sequentially.

6.2. Tree-based Formulation

In this section, we move beyond linear models and use Classification and Regression Trees within the HPA method, using Algorithm 2.

We benchmark the tree-based HPA on a diabetes case-study, against predict-then-optimize based on point estimate (with CART). sample average approximation (SAA), tree-based predictive-prescriptive, and prescriptive trees (Bertsimas et al. (2019)).

Diabetes Case Study The dataset consist of 100,000 observations, with patient-level data with:

- **Patient Characteristics**: age, sexe, medical history, BMI, vitals.
- **Previous Treatments**: drugs previously taken, number of visits to care centers, treatment history.
- **Decision Variable**: treatment given at this time t (11 possible treatments).
- **Target Variable**: average blood sugar levels.

The goal is to prescribe the optimal treatment at time t given patient's characteristics and the previous sequence of treatments in order to minimize his average blood sugar levels. Note that interpretability is key is this experiment since doctors need to understand the decision-making process to prescribe the treatment to real patients. We summarize the results in Table 4.

Method	**Out-of-Sample R2**	**Prescriptive Cost**	**Improvement From Baseline**
CART Regress-then-Prescribe	75%	13510.1	0%
Sample Average Approximation	75%	12980.95	4%
CART Predictive-Prescriptive	73%	12624.8	7%
Prescriptive Trees	**85%**	11682.5	14%
CART HPA	71%	13058.97	3%
CART HPA w/ Multiple Restarts	82%	**11623.18**	**14%**

Table 4 Results for Diabetes Dataset (Mimimization).

The HPA Framework significantly outperforms all other prescriptive methods in terms of prescriptive cost, with a 14% improvement from the standard paradigm predict-then-optimize and 7% improvement from the state-of-the-art predictive-prescriptive approach. Only Prescriptive Trees are marginally behind HPA in terms of prescriptive cost, and outperform it in terms of out-of-sample accuracy. However, the HPA Framework can accommodate constraints and continuous treatments, which is not the case for Prescriptive Trees.

In this example, the prediction and the prescription can both be represented with a simple decision tree, explaining how the patients are clustered into groups, and how decisions are made based on their characteristics. This ensures the model is interpretable and easily explainable to the medical doctors, which are in fact the decision makers in this case study.
By delving into these clusters, we observe that the grouping happens mainly over 4 features: patient's age, sexe, BMI and number of previous visits. The splits separates the BMI into 3 categories: Low, Average, High, the age groups into less than 30 years old, between 30 and 69 years old, and above 69 years old. The split on age only happens for patients in one of the two first age groups ($\leq$ 69 years old), while the cutoff for the previous visits is 5 visits. Which results in a total of 30 intepretable groups. Within each cluster, a different predictive tree model is trained, which splits on drug count and previous treatments to predict sugar level and prescribe the optimal next treatment. This example illustrates the claim that the HPA framework is interpretable when used with interpretable ML methods.

7. Conclusions

In this paper, we introduced a novel method for prescription we refer to as Holistic Prescriptive Analytics (HPA) framework. This framework allows for the use for a large variety of predictive ML

methods and prescriptive cost functions. Moreover, it accommodates constrained and continuous problems, while being scalable and providing a performance edge over the state-of-the-art prescriptive methods. It also preserves interpretability of the ML models that are being used within the framework.

The complexity analysis of the algorithm as well as the computation experiments, on synthetic data and on two real-world case studies provide strong evidence to these claims. Its fundamental holistic structure allows us to combine strong predictive accuracy within a powerful prescriptive framework thanks to the clustering and the fact the three tasks (clustering, prediction and prescription) are optimized jointly instead of sequentially.

References

Ban GY, Rudin C (2018) The big data newsvendor: Practical insights from machine learning. *Operations Research* 67:90–108.

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) *Robust optimization*, volume 28 (Princeton University Press).

Bertsimas D, Dunn J (2017) Optimal classification trees. *Machine Learning* 106(7):1039–1082.

Bertsimas D, Dunn J (2019) *Machine Learning under a Modern Optimization Lens* (Dynamic Ideas).

Bertsimas D, Dunn J, Mundru N (2019) Optimal prescriptive trees. *INFORMS Journal on Optimization* 1(2):164–183.

Bertsimas D, Gupta V, Kallus N (2018) Data-driven robust optimization. *Mathematical Programming* 167(2):235–292.

Bertsimas D, Kallus N (2020) From predictive to prescriptive analytics. *Management Science* 66(3):1025–1044.

Bertsimas D, Orfanoudaki A, Wiberg H (2020) Interpretable clustering: An optimization approach. *Machine Learning* (to appear).

Birge JR, Louveaux F (2011) *Introduction to stochastic programming* (Springer Science & Business Media).

Breiman L, Friedman J, Stone C, Olshen R (1984) *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series (Taylor & Francis).

Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Elmachtoub AN, Grigas P (2017) Smart "predict, then optimize". *ArXiv* 1710.08005.

Kallus N, Pennicooke B, Santacatterina M (2018) More robust estimation of sample average treatment effects using kernel optimal matching in an observational study of spine surgical interventions. *ArXiv* 1811.04274.

Kao Yh, Roy BV, Yan X (2009) Directed regression. *Advances in Neural Information Processing Systems*, 889–897.

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436.

Tulabandhula T, Rudin C (2013) Machine learning with operational costs. *The Journal of Machine Learning Research* 14(1):1989–2028.