Policy Trees for Prediction: Interpretable and Adaptive Model Selection for Machine Learning

Matthew Peroni
Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02142
mperoni1@mit.edu

Dimitris Bertsimas
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142
dbertsim@mit.edu

Abstract

As a multitude of capable machine learning (ML) models become widely available in forms such as open-source software and public APIs, central questions remain regarding their use in real-world applications, especially in high-stakes decision-making. Is there always one best model that should be used? When are the models likely to be error-prone? Should a black-box or interpretable model be used? In this work, we develop a prescriptive methodology to address these key questions, introducing a tree-based approach, **Optimal Predictive-Policy Trees ($\mathbf{OP^2T}$)**, that yields interpretable policies for adaptively selecting a predictive model or ensemble, along with a parameterized option to reject making a prediction. We base our methods on learning globally optimized prescriptive trees. Our approach enables interpretable and adaptive model selection and rejection while only assuming access to model outputs. Our approach works with structured and unstructured datasets by learning policies over different feature spaces, including the model outputs. We evaluate our approach on real-world datasets, including regression and classification tasks with structured and unstructured data. We demonstrate that our approach performs strongly against baseline methods while yielding insights that help answer critical questions about which models to use, and when.

1 Introduction

As increasingly advanced machine learning (ML) algorithms and pre-trained models become democratized and readily available across hundreds of open-source software packages and APIs, such as Scikit-learn [Pedregosa et al., 2011], PyTorch [Paszke et al., 2019], and HuggingFace [Wolf et al., 2020], central questions remain regarding their use in high-stakes decision-making.

- Given a collection of models, which should we use for our application, and when?
- Should we use a model ensemble?
- When are the models likely to be error-prone?
- Should we use a black-box model or an interpretable model?

These questions are, to some degree, open-ended and philosophical. Considerations may include computational resources, inference time, and the qualitative importance of interpretability. To make them more concrete, we aim to address these questions with respect to model performance. For example, when answering whether a model ensemble should be used, we are only concerned with whether the ensemble improves out-of-sample performance. While there is extensive literature on the topics of model selection [Ding et al., 2018], ensembling [Dong et al., 2020], and rejection learning

2nd Workshop on Regulatable ML at NeurIPS 2024.

[Hendrickx et al., 2021], there lacks a comprehensive, holistic modeling framework that can provide clear, interpretable answers to the questions we have posed.

In this work, we develop a prescriptive methodology for adaptively selecting the best model (or ensemble) from a collection of models to make a prediction for a given input or to reject predicting entirely if it is likely that all models will perform poorly. Our approach, which we refer to as Optimal Predictive-Policy Trees (**OP^{2}Ts**), is a tree-based method that partitions the feature space based on the relative performance of each constituent model. By constructing an interpretable policy model, we uncover sub-populations of the data where different models perform the best, and consequently, sub-populations where some (or all) models are likely to fail. We can also consider a fixed set of model ensembles, providing an interpretable and adaptive scheme for model ensembling. We also learn optimal policies for rejection, parameterized by the tolerance for error in predictions. In this work, by **learning with rejection**, we are referring to the class of learning algorithms that, in addition to making predictions, have the option of not making a prediction. This includes algorithms that learn to reject based on the scores of an existing model [Chow, 1970, Bartlett and Wegkamp, 2008], as well as approaches that learn a predictor and rejector jointly [Cortes et al., 2016].

Our work is motivated by overlapping gaps in the Mixture of Experts (ME) and Rejection Learning literature. The first is that model ensemble and ME approaches are generally not interpretable, with most recent developments focusing on deep neural network (DNN) models and assignment modules [Shazeer et al., 2017, Riquelme et al., 2021, Masoudnia and Ebrahimpour, 2014]. They cannot tell the practitioner *why* a certain set of weights was assigned to each of the constituent models, or why a certain model was selected. Recently, a framework referred to as the Interpretable Mixture of Experts (IME) was introduced in part to address this gap [Ismail et al., 2023]. However, the IME approach requires that all constituent models be differentiable (e.g. logistic regression, soft decision trees, DNNs) and that all models have accessible weights that can be tuned. In contrast, this work is motivated by the observation that in many real-world applications, practitioners already have a collection of models under consideration, some of which may not be differentiable. Further, there may be models that are differentiable but for privacy, cost, or licensing reasons, cannot be further trained. In these situations, such ME approaches **do not integrate with existing models**, and therefore become their own, separate modeling approach. The rise of publicly available APIs for large, multi-modal models (e.g. GPT-4, PaLM) suggests that there will be an increasing desire to incorporate models that either cannot practically, or in some cases legally, be fine-tuned.

Similar to the ME literature, there is also the assumption in recent rejection learning literature that the predictor can be optimized jointly with the rejector [Cortes et al., 2016, Charoenphakdee et al., 2021]. Earlier literature that focused on learning a rejector for a pre-existing predictor was either primarily concerned with learning a single predictor-rejector pair [Chow, 1970, Bartlett and Wegkamp, 2008], or did not learn a policy over the feature space [Provost and Fawcett, 2001]. In this work, we incorporate rejection learning with many constituent models, assuming only query access to these models. Further, since our approach yields interpretable policies, we can provide faithful descriptions of the contexts in which it is better to reject, and not use any of the available models. As we discuss in Section 3, such interpretability can be practically useful when creating real-world policies for high-stakes decision-making with machine learning systems.

Motivated by these observations, the main contributions of our work are as follows:

1. We develop a methodology to create interpretable policies for adaptive model selection and rejection. We do so while assuming only query access to the constituent models.
2. We develop our method for regression and classification tasks, and demonstrate empirically that our method is comparable with or improves over baseline approaches while yielding interpretable policies. We demonstrate how one can use the resulting policies to perform subgroup analysis, finding subsets of the data on which different models perform well.
3. We incorporate parameterized rejection, which, within our interpretable framework, can identify settings where all available models are likely to be error-prone.
4. We offer theoretical insights on the conditions under which the proposed approach can learn a policy that is stronger than the best single model in hindsight, as well as alternative approaches that do not consider the relative rewards of each model.

In Section 2, we develop our approach for adaptive model selection in the context of classification, before introducing the addition of a parameterized rejection option. In Section 3 we evaluate

our methodology on a variety of real-world datasets and benchmark our approach against related, predictive techniques. We provide additional experiments in Appendix B and our theoretical insights as supplementary material in Appendix C.

2 Adaptive Model Selection for Classification and Regression

In this section, we develop our Optimal Predictive-Policy Tree methodology for classification and regression tasks. For both settings, we introduce the option for parameterized rejection and provide an analysis of how this parameterization impacts the resulting policy model.

2.1 Adaptive Model Selection for Classification

In this section, we formalize our approach. We first consider classification tasks, where we have some input data $\{x_i\}_{i=1}^n$ from a feature space $\mathcal{X}$ and label data $\{y_i\}_{i=1}^n \in \mathcal{C}^n$, where $C = \{1, \ldots, K\}$. We assume access to a collection of m constituent models $\{h_1, \ldots, h_m\}$, $h_i : \mathcal{X} \to \Delta^K$, where Δ^K is the unit simplex, and all models are fit on some training set. We denote by $\mathbf{h} = [h_1, \ldots, h_m]$ the vectorized form of the constituent models such that $\mathbf{h} : \mathcal{X} \to \mathbb{R}^{m \times K}$. Let $\hat{y}_{ijk} = h_i(x_j)_k$, such that $\hat{y}_{ijk}$ is the output probability of class k for sample x_j from model h_i. Given a fixed, finite set of weights $\mathbf{W} \subset \Delta^m$ over the unit simplex, we want to learn a function $f : \mathcal{X} \to \mathbf{W}$ such that for any $x \in \mathcal{X}$, f selects the best set of weights $\mathbf{w} \in \mathbf{W}$ such that the resulting model (or ensemble) $\mathbf{w}^T\mathbf{h}$ makes the best prediction on x. In the case that $\mathcal{X}$ is not some real space (e.g. unstructured language data), we instead want to learn our policy model f over some real-valued side-information $\{z_i\}_{i=1}^n \in \mathcal{Z}^n$, where $\mathcal{Z}$ could be a subset of the features space $\mathcal{X}$ or separate data entirely, including the model outputs $\mathbf{h}(x)$ themselves. We can measure the quality of a model's prediction on a given sample, or the reward, in a few ways. For classification, we consider misclassification error and cross-entropy, defined as follows:

$$R_{CE}(x_j, y_j, \mathbf{w}, \mathbf{h}) = \sum_{k=1}^{K} \mathbb{1}\{y_j = k\} \log((\mathbf{w}^T\mathbf{h}(x_j))_k), \tag{1}$$

$$R_{MIS}(x_j, y_j, \mathbf{w}, \mathbf{h}) = \mathbb{1}\{y_j = \arg\max_k (\mathbf{w}^T\mathbf{h}(x_j))_k\}. \tag{2}$$

Note that $\mathbf{w} = \mathbf{e}_i$ corresponds to selecting a single model. We introduce these reward functions because they present inherent trade-offs. The cross-entropy reward is more informative than the misclassification reward. Making the reasonable assumption that the model predictions are fairly smooth over the feature space, R_{CE} will also form a smoother reward surface than R_{MIS}. One risk with using the cross-entropy reward is that models that are over-confident, as is often the case with neural networks [Wang et al., 2021], may be favored despite their ability to separate the data being similar to, or worse than, other models available. Since our policy prescribes models, rather than making class predictions directly, neither reward function requires committing to a threshold for class prediction a-priori. However, using R_{MIS} requires selecting a prediction threshold for generating the rewards, which will impact the resulting tree structure and model prescriptions. However, R_{MIS} benefits from a simpler interpretation, as model prescriptions are based directly on minimizing misclassification error.

With these definitions, we can construct our reward matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$ corresponding to the reward for using each model h_i to predict the label for each sample x_j. We define our action space as the set of weights $\mathbf{W}$ defined above. Then, prescribing model h_i corresponds to the action $\mathbf{e}_i = [0, \ldots, 1, \ldots, 0] \in \mathbf{W}$. Throughout this work, we assume $\mathbf{e}_i \in \mathbf{W}$ for all $i \in [m]$. That is, we can always take the action of prescribing an individual constituent model. As defined, any fixed ensemble of constituent models can be included in the action space $\mathbf{W}$. In Section 3, we demonstrate this flexibility and show empirically that our approach uncovers partitions of the feature space where different ensemble weights are better suited.

We are now able to formulate our objective as a policy-learning problem, where the set of actions $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_q\}$ is the set of weights over the constituent models that we can prescribe, our state space is $\mathcal{X}$ (or $\mathcal{Z}$), and the rewards for prescribing each model is given by our reward matrix $\mathbf{R}$. We propose learning an interpretable and adaptive policy model by using the Optimal Trees algorithm, introduced by Amram et al. [2022], to produce a policy tree T_O that learns to prescribe constituent

models (or ensembles) given the input data by maximizing the reward R over the class of decision trees. We fit T_O using a validation dataset that is separate from the data used to train the constituent models. Throughout this work, we refer to this general approach as Optimal Predictive-Policy Trees (OP^{2}Ts). We provide an overview of optimal policy trees in Appendix A.1.

We now introduce a simple extension that allows us to incorporate rejection into this framework. In addition to the m constituent models $h_1, \ldots, h_m$, we introduce a dummy rejection model h_r that we add to the action space of our formulation. We must assign reward values $R(\cdot, h_r)$ so that our policy learning algorithm can compare the action of rejection against the action of prescribing the other models. One approach is to fix some constants $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K) \in [0, 1)^K$ and suppose the output of the rejection model is always exactly within α_k of predicting the true label k. That is, for some input x_j with label $y_j = k$, we define $\hat{y}_{rjk} = 1 - \alpha_k$. For example, in the binary classification setting, we define the output of the rejection model as

$$h_r(x, y) = \mathbb{1}\{y = 1\}(1 - \alpha_1) + \mathbb{1}\{y = 0\}\alpha_0. \tag{3}$$

Therefore, if we were to set $\boldsymbol{\alpha} = (0.3, 0.2)$, then $h_r(x, y = 1) = 0.7$ and $h_r(x, y = 0) = 0.2$. In our experiments, we set $\boldsymbol{\alpha}$ as a constant vector such that $\alpha_i = \alpha$ for all $i \in [K]$ and generate OP^{2}Ts for a range of values for α. In practice, one may wish to select a non-constant $\boldsymbol{\alpha}$ to characterize a model that has varying performance on different classes. Notice that, if we assume $\alpha_k < \frac{1}{K}$ for all $k \in [K]$, h_r is a consistent classifier. That is, for any sample (x, y), we have $y = \arg\max_i h_r(x)_i$. Therefore, to construct rewards for rejection, we assume that h_r is **perfectly calibrated**, such that $P(y = k | h_r(x)_k = 1 - \alpha_k) = 1 - \alpha_k$. We can then define the reward as

$$R_{CE}(x_j, y_j, h_r) = \log(1 - \alpha_{y_j}), \tag{4}$$

$$R_{MIS}(x_j, y_j, h_r) = 1 - \alpha_{y_j}. \tag{5}$$

The misclassification reward for the rejection model can be interpreted as the expected reward. With the rewards defined for the rejection model, we can proceed with our approach as in Section 2, with the action space extended to $\bar{\mathbf{W}} = \mathbf{W} \cup \{h_r\}$. Since our decision tree formulation prescribes the action that maximizes (or minimizes) the total reward in each leaf, using R_{MIS}, we can immediately conclude that rejection is prescribed in a leaf if and only if the accuracy of the constituent models is less than the accuracy of the rejection model parameterized by $\boldsymbol{\alpha}$. In contrast, due to the non-linear transformation of the logarithm in the reward function R_{CE}, which more severely punishes predictions that are far from the true label, the decision to reject does not have a clear interpretation in terms of a performance metric.

2.2 Adaptive Model Selection for Regression

In the context of regression, we assume we have some input data $\{x_i\}_{i=1}^n \in \mathcal{X}^n$ and output data $\{y_i\}_{i=1}^n \in \mathbb{R}^n$. We assume access to a collection of m constituent regression models $\mathbf{h} = [h_1, \ldots, h_m]$, such that $h_i : \mathcal{X} \to \mathbb{R}$. Similar to classification, given a fixed, finite set of weights $\mathbf{W} \subset \mathbb{R}^m$ we want to learn a function $f : \mathcal{X} \to \mathbf{W}$ such that for any $x \in \mathcal{X}$ (or $z \in \mathcal{Z}$), f selects the best ensemble weights $\mathbf{w} \in \mathbf{W}$ such that $\mathbf{w}^T\mathbf{h}$ makes the best prediction on x. For this work, we measure the reward (or loss) using the squared error:

$$R_{SE}(x_j, y_j, \mathbf{w}, \mathbf{h}) = (\mathbf{w}^T\mathbf{h}(x_j) - y_j)^2. \tag{6}$$

Again, for this work, we assume $\mathbf{e}_i \in \mathbf{W}$ for all $i \in [m]$. In this case, it is natural to frame the problem in terms of minimizing the reward. To our benefit, the reward function we define for regression is directly connected to the metrics used to evaluate regression models, namely mean squared error (MSE). That is, since our formulation uses total reward to select a model prescription in each leaf, the model that is prescribed in a given leaf is the one that has the minimum squared error over the samples that fall into that leaf. Minimizing the total reward also means that the resulting tree is directly optimizing to minimize the MSE. While we focus on squared error for this work, note that we could also easily measure reward in terms of absolute error if that was our metric of interest.

Incorporating rejection learning for regression is simpler than in the case of classification. In this setting, we benefit from the reward function being directly connected to the evaluation metric of interest. Further, since the output of the models is one-dimensional, we only have to focus on a one-dimensional parameterization, as opposed to a K-dimensional parameterization for the multi-class classification setting with K classes. We can introduce parameterized rejection by adding a dummy

rejection model h_r that is always a squared (or absolute) distance of $\alpha \in \mathbb{R}_{\geq 0}$ from any $x \in \mathcal{X}$ by fixing a constant reward

$$R(x_j, y_j, h_r) = \alpha.$$

We treat α as our rejection threshold parameter. We then extend our action space to $\bar{\mathbf{W}} = \mathbf{W} \cup \{h_r\}$. With the rejection model added, we can again learn a policy tree T_O that minimizes the reward. Then, for any leaf $l \in T_O$, the model prescribed in l must have an MSE less than α over the data that falls into l. Otherwise, the leaf prescription will be to reject making a prediction. Therefore, rejection learning can be viewed as enforcing a maximum MSE over the training dataset. Similar to classification, for regression, as $\alpha \to 0$, we expect any policy model, including OP^{2}T, to converge to prescribing rejection always. We formalize this notion in Proposition 6.

2.3 Alternative Predictive Approaches

Having introduced our OP^{2}T approach, we now describe two alternative approaches, which we evaluate and compare against our method in Section 3. Central to our work is the idea of using a prescriptive framework, measuring the relative reward of different models and ensembles, for predictive ML. It is natural then to ask what the benefit is of taking a prescriptive approach. A simpler alternative would be to instead treat the model prescription problem as a multi-class classification problem. In this setting, we have M classes corresponding to each constituent model and ensemble, and each sample is labeled by the model or ensemble with the closest prediction to the true target. Formally, for models $[h_1, \ldots, h_M]$, and data $\{(x_i, y_i)\}_{i=1}^n$, we assign sample x_i a label $q_i \in [M]$ as

$$q_i = \underset{i \in [M]}{\arg\min} |y_i - h_i(x_i)|, \tag{7}$$

where we break ties arbitrarily (e.g. smallest index). Then the equivalent of our approach in this setting would be a decision tree for classification, fit on the dataset $\{(x_i, q_i)\}_{i=1}^n$, or $\{(z_i, q_i)\}_{i=1}^n$ in case we are learning a policy over a modified feature space $\mathcal{Z}$. We refer to such a classification tree as a **Meta-Tree**. The Meta-Tree approach is a close comparison to our OP^{2}T method, as both learn over the same hypothesis space. That is, both approaches learn decision trees over the same feature space, with an equivalence between the M classes and the prescriptions. The difference lies in how the tree is learned, and we demonstrate both theoretically in Appendix C and empirically in Section 3 that this difference can lead to the resulting OP^{2}Ts outperforming the Meta-Trees.

In addition to introducing a prescriptive framework to the problem of adaptive model selection, our approach yields interpretable policies that can be used to gain insights regarding existing predictive models. We therefore want to compare our approach against a related, black-box method. As we are focused on learning policies over structured data, it is natural to consider boosted trees as an alternative, black-box approach. Specifically, in the same vein as the Meta-Tree approach, we train boosted tree models, using XGBoost [Chen and Guestrin, 2016], on the multi-class data $\{(x_i, q_i)\}_{i=1}^n$, or $\{(z_i, q_i)\}_{i=1}^n$. We refer to this approach as **Meta-XGB**. Since boosted tree models are more expressive than individual decision trees, this allows us to evaluate further the power of incorporating the relative rewards of each model and ensemble in a prescriptive framework.

3 Experiments

We evaluate our methodology on a variety of real-world datasets with both structured and unstructured data, including regression and classification tasks. For all experiments, we perform a hyperparameter search over the tree complexity penalty λ, max depth D_{max}, and minimum number of samples per leaf c_{min}. To preserve interpretability, we limit the max depth to at most $D_{max} = 10$. We then fit all of the meta-learning models via k-fold cross validation (for $k \in \{3, 5\}$). All experiments were performed on a private cluster of 48 Intel Xeon-P8 CPUs with 4GB of RAM per core. We benchmark against two other predictive, tree-based methods, introduced in Section 2.3. For both approaches, we perform hyperparameter tuning over the max depth and the minimum number of samples per leaf. In addition, for the Meta-XGB approach, we tune the total number of estimators.

3.1 Recidivism Prediction

To investigate the benefits of our methodology in highly regulated, high-stakes decision-making, we evaluate our framework on a real-world recidivism dataset, made publicly available through the

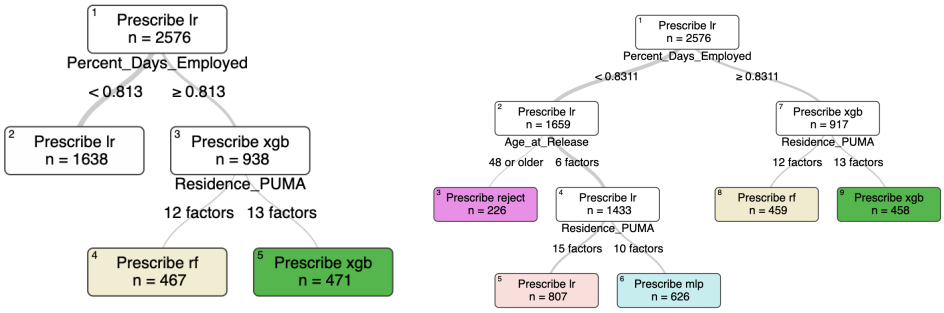
National Institute of Justice (NIJ) [Hunt, 2021]. The dataset consists of covariates and recidivism outcomes for residents in the state of Georgia. The dataset consists of $n = 26761$ samples, each corresponding to a separate individual, and $p = 48$ features, including binary, nominal and ordinal categorical, and continuous variables. The data was already partitioned into training and testing sets and we use this same split, reserving 15% of the training data as a validation set. The features contain demographic information (gender, age, residence location, education), drug-use information, arrest and imprisonment history (offenses, sentence length, prior arrests), previous violations, and employment history. For this experiment, we focus on predicting three-year recidivism, a binary classification task. The dataset was included as part of a competition hosted by the NIJ, during which the evaluation metric was the Brier score. The Brier score corresponds to the squared distance between the true label and the predicted probability output, $\text{Brier}(X, \mathbf{y}, h) = \frac{1}{n}\sum_i (y_i - h(\mathbf{x}_i))^2$, where $y_i \in \{0, 1\}$ and $h(\mathbf{x}_i) \in [0, 1]$ is the model output. Accordingly, we fit our OP^{2}Ts using the reward function $R_{SE}(\mathbf{x}_i, y_i, h_j) = (y_i - h_j(\mathbf{x}_i))^2$, such that minimizing the reward over the training set is equivalent to minimizing the Brier score. For the constituent models, we fit a logistic regression model, a CART decision tree model, an MLP model, an XGBoost model, and a Random Forest model. We summarize our performance results in Figure 1. The best single model in hindsight on the test dataset is the Random Forest model, which achieves a Brier score of 0.1882. The OP^{2}T approach improves the out-of-sample performance over selecting an individual model and outperforms the alternative predictive meta-learning approaches. For higher levels of rejection, the Meta-XGB approach has an advantage. This suggests that setting the rejection level to a lower Brier score induces many regions with nonlinear boundaries over the feature space where rejection is the best option, and the Meta-XGB approach is more capable of capturing these regions.

Model	No Rejection		Rejection w/ $\alpha = 0.22$		Rejection w/ $\alpha = 0.21$	
	Brier	Reject (%)	Brier	Reject (%)	Brier	Reject (%)
Meta-XGB	0.1918	0	0.1879	8.20	0.1713	29.06
Meta-Tree	0.1896	0	0.1866	9.01	0.1776	29.95
OP^{2}T	0.1862	0	0.1852	8.08	0.1772	28.81

Table 1: Out-of-sample Brier Score on the NIJ recidivism dataset across different rejection thresholds.

In Figures 1 and 2, we provide visualizations of the resulting OP^{2}Ts. The leaf nodes correspond to selecting (or prescribing) the logistic regression (lr), XGBoost (xgb), random forest (rf), multi-layer perceptron (mlp) models, or rejecting the sample (reject) for prediction. We can see that the trees have discovered partitions of the feature space over which different models perform the best. Tree (a) in Figure 1 prescribes an interpretable model, logistic regression, for the majority of samples in the dataset. Therefore, we have discovered a potential balance between interpretability and performance, as Tree (a) outperforms the best constituent model in hindsight with respect to the Brier score. Looking at the right subtree of Tree (a), we observe a split on the geographic location of the person, given by the Public Use Microdata Area (PUMA). There are 25 PUMA zones included in the dataset, and this split roughly divides the zones into the Metro Atlanta region in the right leaf and the suburban and rural zones in the left leaf. Surprisingly, the Brier score for the XGBoost model over the data in the right leaf is 0.16 ± 0.01, while the score for the XGBoost model over the data in the left leaf is 0.22 ± 0.01, highlighting a significant difference in the performance of the XGBoost model depending on the region where the person under consideration resides. Further, the Brier score for logistic regression is 0.20 ± 0.01 over the data that falls into the leaf where XGBoost is prescribed. Had conventional, static model selection methods been used, one might select logistic regression for its performance over the entire validation dataset, missing this subgroup for which another model exhibits significantly better performance. Due to the interpretable nature of our meta-learning algorithm, we can also understand and explain what these subgroups are.

Looking at Tree (b) in Figure 1, we observe a particular subgroup, people over the age of 48 who have been employed for less than 80% of days while on parole, has been identified for rejection. Indeed, the Brier score for logistic regression over the general population is 0.19 ± 0.01, while the score over this subgroup is 0.25 ± 0.01, which is also the best score for this subgroup among all the available models. While this subgroup comprises only 8.7% of the population in the dataset, we know due to the interpretable nature of our meta-learning approach that our models are particularly unreliable for a certain group of older people. In a real-world application, one could use this result to inform further model training and deployment, potentially avoiding systematic algorithmic bias.



(a) OP^{2}T (b) OP^{2}T ($\alpha = 0.22$)

Figure 1: OP^{2}Ts for the NIJ recidivism dataset.

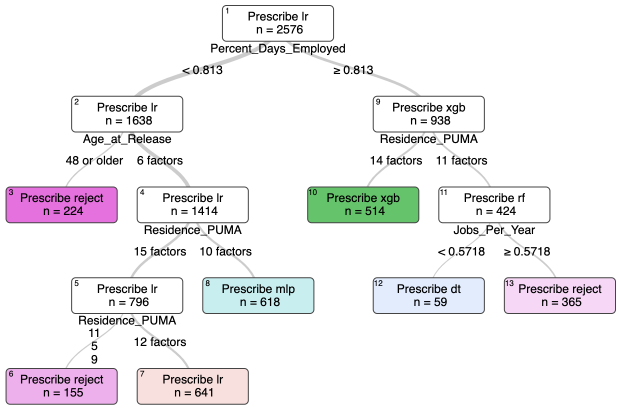


Figure 2: OP^{2}T for the NIJ recidivism dataset with $\alpha = 0.21$, where α corresponds to the Brier score.

In Figure 2 we provide a visualization of the OP^{2}T with $\alpha = 0.21$. Consistent with our performance results, we see that improving the Brier score for the rejection option to this level induces multiple, separate new regions in the feature space over which rejection attains the best reward. Since the resulting tree is shallow and relatively sparse, we are still able to clearly interpret the subgroups for which rejection is prescribed. It is clear that model selection and rejection depend primarily on the person's age, location, and employment history. Again we see significant variation in performance at the splits where rejection is prescribed in one of the leaf nodes. In the left subtree of Figure 2, we see a split on residence zone, rejecting prediction for people living in certain counties in Metro Atlanta and West Georgia. For the corresponding leaf node that prescribe logistic regression, the model score is 0.15 ± 0.01, while the score for logistic regression over the population that falls into the "reject" leaf is 0.24 ± 0.02. In the right subtree, we see a split on jobs per year while on parole that is used to prescribe rejection. In this case, the Brier score for the decision tree over the data in the corresponding leaf node is 0.10 ± 0.02, while the score for the decision tree over the samples falling into the "reject" node is 0.24 ± 0.01. Therefore, the score for the decision tree doubles depending on the employment history over this subset of the population. These results illustrate the potential of our approach to not only increase performance, but to gain insight into the relative performance of our models, conditioned on the data itself. To summarize, we can connect the results of this experiment directly back to the motivating questions we posed at the beginning of the paper.

Which model should we use for our application, and when? When the person has been employed for less than 80% of days while on parole, it is best to use logistic regression. Otherwise, if the person is from metro Atlanta, the XGBoost model performs the best. For people outside of Atlanta who have been employed for more than 80% of days while on parole, use the Random Forest model.

When are the models likely to be error-prone? There are a few cases. The first and most significant is when the person has been employed for less than 80% of days and is age 48 or older. The next is when the person has been employed less than 80% of days, is under the age of 48, and resides in

certain counties in Atlanta or west Georgia. Another case is when the person has been employed more than 80% of days, lives outside of Atlanta, and has held more than about 0.6 jobs per year.

Should we use an interpretable model? Yes, for the population that has been employed less than 80% of days while on parole, logistic regression is the best model with respect to the Brier score. Otherwise, the black-box models have a performance advantage. When incorporating rejection, there are also significant portions of the population for which an interpretable model is best.

3.2 Concrete Compressive Strength

As a baseline for evaluating our OP^{2}Ts on regression tasks, we apply our approach to the Concrete Compressive Strength dataset (n=1080, p=8), publicly available through the UCI ML Repository [Yeh, 2007]. We fit a boosted tree model (xgb), a random forest model (rf), a linear regression (lr) model, and an MLP model on the training data. We then create two ensembles: a mean ensemble and a ridge ensemble. The weights for the ridge ensemble are found by solving the optimization problem

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}\in\mathbb{R}^m} \frac{1}{n}||\mathbf{w}\mathbf{h}(\mathbf{X})^T - \mathbf{y}||_2^2 + \lambda||\mathbf{w}||_2^2,$$

where $\mathbf{h}(\mathbf{X}) \in \mathbb{R}^{n\times m}$ are the predictions from the m constituent models on some input data $\mathbf{X} \in \mathbb{R}^{n\times m}$ with n samples and $\mathbf{y} \in \mathbb{R}^n$. This formulation has a closed-form solution,

$$\mathbf{w}^* = ((\mathbf{h}(\mathbf{X})\mathbf{h}(\mathbf{X})^T + \lambda\mathbf{I})^{-1}\mathbf{h}(\mathbf{X}))^T\mathbf{y}.$$

We fix $\lambda = 1$ for our experiments. Then, for our experiments, our actions are the set of weights $\mathbf{W} = (\mathbf{e}_1, \ldots, \mathbf{e}_m, (\frac{1}{m}, \ldots, \frac{1}{m}), \mathbf{w}^*)$. For rejection, we evaluate the threshold parameter α over the set $[100, 40, 30]$. The results of this experiment are shown given in Table 2. Without rejection, the best constituent model (including the ensembles) achieved an out-of-sample MSE of 39.2, while the OP^{2}T without rejection achieved an out-of-sample MSE of 36.13, demonstrating that this approach can learn a simple, interpretable policy that generalizes well. We also observe that both the mean and ridge ensemble weights are prescribed in the trees, yielding an interpretable, adaptive model ensembling policy. Further, we gain some insight from the learned policy. In Figure 3, we provide a visualization of three OP^{2}T models fit with a max depth of $D_{\max} = 3$, for ease of interpretability. We see that by partitioning the data based on the density of concrete in the cement mixture, splitting at 357.5 kg/m^3, the XGBoost model outperforms all other constituent models when the density is less than 357.5 kg/m^3, while the random forest model performs the best when the density is greater than or equal to 357.5 kg/m^3. Looking at the statistics for the dataset, we found that the 75th percentile for concrete density is 350 kg/m^3. Therefore, we can conclude that the learned policy is to use the random forest model when the cement density is high (above the 75th percentile), otherwise, it is best to use the XGBoost model.

Model	No Rejection		Rejection w/ $\alpha = 40$		Rejection w/ $\alpha = 30$	
	MSE	Reject (%)	MSE	Reject (%)	MSE	Reject (%)
Meta-XGB	39.69	0	37.32	18.93	33.58	43.20
Meta-Tree	39.57	0	42.82	16.02	37.35	37.86
OP^{2}T	36.13	0	27.30	17.01	25.22	37.43

Table 2: Out-of-sample MSE on the concrete compressive strength dataset across different rejection thresholds.

With the rejection option, we see that our OP^{2}T approach learns policies that generalize well. In each case, we can also interpret the policies that are learned, and we observe consistency across the policy trees. For example, in tree (b) we see that the left subtree learns a split similar to tree (a), prescribing the random forest model when cement density is high, and prescribing the XGBoost model otherwise. Something similar is learned in tree (c), in the left-most subtree, except when the cement density is high, the tree opts to reject rather than prescribing the random forest model. This is due to the reward for rejection being considerably higher in tree (c) than in trees (a) and (b). We can also observe that Trees (b) and (c) both learn to split on Blast Furnace Slag and Water density at very similar values. Further, both reject when Blast Furnace Slag density is high (above the 60th percentile) and Water density is low (below the 50th percentile). Tree (d) also learns a similar split on Cement density, as well as on Blast Furnace Slag density, except it rejects any time the Blast Furnace Slag density is high. From these tree-based policies, we can identify clear and consistent partitions of the data within

which different models perform well and partitions under which no models perform particularly well. To summarize, we can connect the results of this experiment directly back to the motivating questions we posed at the beginning of the paper.

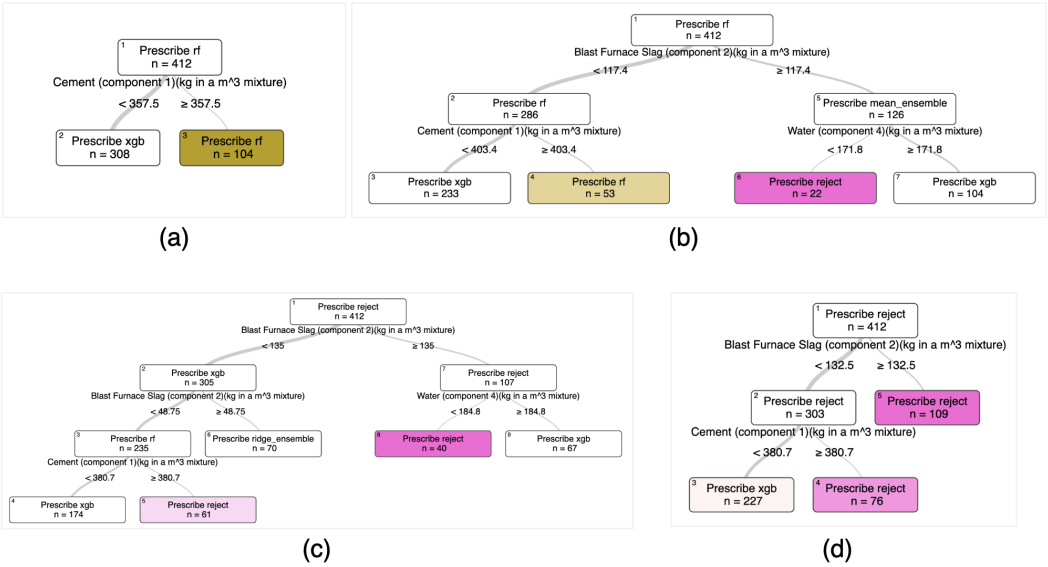


Figure 3: OP^{2}T models fit on the Concrete Compression dataset with a max depth of $d = 3$ (for interpretability) and varying rewards for rejection. Tree (a) is fit without a rejection option, while trees (b), (c), and (d) are fit with rejection parameters $\alpha = [100, 40, 25]$ respectively.

Should we use a model ensemble? If so, when? Yes, in some settings, the mean ensemble is the best, while in other cases the ridge ensemble is better. For example, when the blast furnace slag component is in the range $[49, 135]$, tree (c) in Figure 3 identifies the ridge ensemble to be the best model.

When are the models likely to be error-prone? In short, when there is a high concentration of the blast furnace slag or the cement component in the concrete mixture.

Should we use an interpretable model? In this case, when the interpretable model is linear regression, the answer is no. Across all rejection thresholds, the OP^{2}Ts do not identify any partition of the feature space where the linear regression model performs the best.

4 Conclusion

In this work, we introduce a prescriptive framework for creating interpretable and adaptive model selection and ensembling policies, along with a parameterized rejection option. We demonstrate on a variety of tasks, model classes, and datasets, that our approach yields strong performance while offering interpretable policies that aid in the understanding of the efficacy of the constituent models. In particular, our OP^{2}T approach directly provides answers to the questions we identified in the introduction, such as under what conditions different models should be used, and when all of the models are likely to be error-prone. Our approach also makes minimal assumptions regarding the constituent models, only requiring access to model outputs, increasing the applicability of our method. In addition to empirical results, we introduce a theoretical framework for prescriptive, adaptive model selection. We show both theoretically and empirically that our prescriptive approach can outperform alternative, predictive methods. There are many exciting directions for future work, such as investigating different reward functions, dynamically adapting ensemble weights, and exploring the theoretical properties of such expert selection systems. This work takes a first step toward using prescriptive analytics to enhance predictive models, and we hope it encourages further research in this direction.

Acknowledgments and Disclosure of Funding

The authors acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing high-performance computing resources that have contributed to the research results reported in this paper.

References

Maxime Amram, Jack Dunn, and Ying Daisy Zhuo. Optimal policy trees. *Machine Learning*, 111(7):2741–2768, July 2022. ISSN 1573-0565. doi: 10.1007/s10994-022-06128-5. URL https://doi.org/10.1007/s10994-022-06128-5.

Peter L. Bartlett and Marten H. Wegkamp. Classification with a Reject Option using a Hinge Loss. *Journal of Machine Learning Research*, 9(59):1823–1840, 2008. ISSN 1533-7928. URL http://jmlr.org/papers/v9/bartlett08a.html.

Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, July 2017. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-017-5633-9. URL http://link.springer.com/10.1007/s10994-017-5633-9.

Dimitris Bertsimas and Jack Dunn. *Machine learning under a modern optimization lens*. Dynamic Ideas LLC Charlestown, MA, 2019. URL https://www.lnmb.nl/conferences/2018/programlnmbconference/Bertsimas-1.pdf.

Max Biggs, Wei Sun, and Markus Ettl. Model Distillation for Revenue Optimization: Interpretable Personalized Pricing, June 2021. URL http://arxiv.org/abs/2007.01903. arXiv:2007.01903 [cs, stat].

Léonard Boussioux, Cynthia Zeng, Théo Guénais, and Dimitris Bertsimas. Hurricane Forecasting: A Novel Multimodal Machine Learning Framework. *Weather and Forecasting*, 37(6):817–831, June 2022. ISSN 1520-0434, 0882-8156. doi: 10.1175/WAF-D-21-0091.1. URL https://journals.ametsoc.org/view/journals/wefo/37/6/WAF-D-21-0091.1.xml. Publisher: American Meteorological Society Section: Weather and Forecasting.

Nontawat Charoenphakdee, Zhenghang Cui, Yivan Zhang, and Masashi Sugiyama. Classification with Rejection Based on Cost-sensitive Classification, September 2021. URL http://arxiv.org/abs/2010.11748. arXiv:2010.11748 [cs, stat].

Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145%2F2939672.2939785.

C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, January 1970. ISSN 1557-9654. doi: 10.1109/TIT.1970.1054406. URL https://ieeexplore.ieee.org/document/1054406. Conference Name: IEEE Transactions on Information Theory.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with Rejection. In Ronald Ortner, Hans Ulrich Simon, and Sandra Zilles, editors, *Algorithmic Learning Theory*, volume 9925, pages 67–82. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46378-0 978-3-319-46379-7. doi: 10.1007/978-3-319-46379-7_5. URL http://link.springer.com/10.1007/978-3-319-46379-7_5. Series Title: Lecture Notes in Computer Science.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv:1810.04805 [cs].

Jie Ding, Vahid Tarokh, and Yuhong Yang. Model selection techniques: An overview. *IEEE Signal Processing Magazine*, 35(6):16–34, 2018.

Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, April 2020. ISSN 2095-2236. doi: 10.1007/s11704-019-8208-z. URL https://doi.org/10.1007/s11704-019-8208-z.

Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis. Machine learning with a reject option: A survey. *arXiv preprint arXiv:2107.11277*, 2021.

Joel Hunt. NIJ's recidivism challenge full dataset, 2021. URL https://data.ojp.usdoj.gov/Courts/NIJ-s-Recidivism-Challenge-Full-Dataset/ynf5-u8nk/about_data.

Aya Abdelsalam Ismail, Sercan O. Arik, Jinsung Yoon, Ankur Taly, Soheil Feizi, and Tomas Pfister. Interpretable Mixture of Experts. *Transactions on Machine Learning Research*, March 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=DdZoPUPm0a.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, August 2014. ISSN 1573-7462. doi: 10.1007/s10462-012-9338-y. URL https://doi.org/10.1007/s10462-012-9338-y.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Foster Provost and Tom Fawcett. Robust Classification for Imprecise Environments. *Machine Learning*, 42(3):203–231, March 2001. ISSN 1573-0565. doi: 10.1023/A:1007601015854. URL https://doi.org/10.1023/A:1007601015854.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable Machine Learning: Fundamental Principles and 10 Grand Challenges, July 2021. URL http://arxiv.org/abs/2103.11251. arXiv:2103.11251 [cs, stat].

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Chandan Singh, Armin Askari, Rich Caruana, and Jianfeng Gao. Augmenting interpretable models with large language models during training. *Nature Communications*, 14(1):7913, November 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-43713-1. URL https://www.nature.com/articles/s41467-023-43713-1. Number: 1 Publisher: Nature Publishing Group.

Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. Rethinking Calibration of Deep Neural Networks: Do Not Be Afraid of Overconfidence. In *Advances in Neural Information Processing Systems*, volume 34, pages 11809–11820. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/61f3a6dbc9120ea78ef75544826c814e-Abstract.html.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

I-Cheng Yeh. Concrete Compressive Strength. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C5PK67.

Zhengyuan Zhou, Susan Athey, and Stefan Wager. Offline Multi-Action Policy Learning: Generalization and Optimization, November 2018. URL http://arxiv.org/abs/1810.04778. arXiv:1810.04778 [cs, econ, stat].

A Appendix

A.1 Optimal Policy Trees

At the center of our methodology is learning interpretable, tree-based policies that are capable of capturing nonlinear interactions and partitioning the feature space. To accomplish this, we base our approach on the Optimal Policy Tree (OPT) formulation, introduced in Amram et al. [2022]. The setup is as follows. Suppose we have some dataset $\{x_i\}_{i=1}^n$ of size n, where $x_i \in \mathbb{R}^d$ along with m possible treatments $T = \{t_1, \ldots, t_m\}$. Suppose we are given a reward for each observation i and treatment t, denoted by R_{it}. In many cases, such as observational data in medicine, the counterfactuals, the reward for sample i under each treatment, may not be known. In these settings, an additional reward estimation step is required. However, as we demonstrated in Section 2, in our approach, no reward estimation is necessary. We would like to learn a function $f : \mathbb{R}^d \to T$ that prescribes a treatment for any input sample. Specifically, we will learn this function from the class of decision trees. Our objective is to maximize (or minimize) the total reward over all samples using the treatments prescribed by the decision tree. Assuming we are interested in maximizing the reward, our objective becomes

$$\max_{\tau(\cdot)} \sum_{i=1}^{n} \sum_{t=1}^{m} \mathbb{1}\{\tau(x_i) = t\} R_{it}, \tag{8}$$

where we maximize over the class of decision trees τ up to some depth D and $\tau(x_i)$ is the prescription made by the decision tree for sample x_i. To avoid overfitting, we add a penalty to the objective for tree complexity, measured by the number of splits in the tree τ. The formulation then becomes

$$\max_{\tau(\cdot)} \sum_{i=1}^{n} \sum_{t=1}^{m} \mathbb{1}\{\tau(x_i) = t\} R_{it} + \lambda \cdot \text{numsplits}(\tau), \tag{9}$$

where $\lambda \in \mathbb{R}_+$ is a parameter to control the complexity penalty. Focusing on the prescription in the leaf nodes, we introduce the leaf assignment function $v : \mathbb{R}^d \to L$ for a given tree τ with the set of leaf nodes L, where $|L| = q$. Denoting the prescriptions for each leaf by $\mathbf{z} = (z_1, \ldots, z_q)$, we can rewrite our formulation as

$$\max_{v,\mathbf{z}} \sum_{i=1}^{n} \sum_{l=1}^{q} \mathbb{1}\{v(x_i) = l\} R_{iz_l} + \lambda \cdot \text{numsplits}(v). \tag{10}$$

Notice that this problem is separable in the leaves, such that we can rewrite the formulation as

$$\max_{v,\mathbf{z}} \sum_{l=1}^{q} \sum_{i:v(x_i)=l} R_{iz_l} + \lambda \cdot \text{numsplits}(v). \tag{11}$$

Then, for any given tree structure τ and corresponding leaf assignment function v, the optimal prescription in each leaf is given by

$$z_l = \arg\min_{t \in T} \sum_{i:v(x_i)=l} R_{it}, \tag{12}$$

which can be solved by enumerating the possible treatments. We add constraints for the maximum depth, D, of the tree and the minimum number of samples for each leaf, c_{min}. Together, the parameters λ, c_{min} and D help control overfitting. Putting this together, our final formulation can be

written as

$$\begin{aligned} \max_{v,\mathbf{z}} \quad & \sum_{l=1}^{q} \sum_{i:v(x_i)=l} R_{iz_l} + \lambda \cdot \text{numsplits}(v) \\ \text{s.t.} \quad & \text{depth}(v) \leq D, \\ & \sum_{i=1}^{n} \mathbb{1}\{v(x_i) = l\} \geq c_{min} \quad \forall l. \end{aligned} \tag{13}$$

The tree τ is then optimized using the Optimal Trees framework given in Section 8.4 of Bertsimas and Dunn [2019]. Specifically, rather than using a greedy heuristic, we use a global optimization approach, based on coordinate descent, to optimize the tree structure and the prescriptions jointly. The hyperparameters D and c_{min} are optimized using a traditional grid search over a discrete set of values, while λ is optimized through a pruning approach that generates a sequence of trees and identifies the value of λ that minimizes the validation loss. The problem posed in Eq. (A.1) was also formulated by Zhou et al. [2018] and Biggs et al. [2021], however, they both used greedy heuristics to fit their policy trees. In Amram et al. [2022], the authors demonstrate that training policy trees using a global optimization methodology can yield significant performance and interpretability advantages over greedy heuristic approaches.

For this work, we focus on the class of parallel-split decision trees, but our formulation can be easily extended to hyperplane splits, as in Bertsimas and Dunn [2017]. We do this to maintain the focus of this work, and preserve interpretability. In addition, we found in preliminary experiments that hyperplane splits did not yield a significant change in performance for the datasets we evaluated.

A.2 A Toy 1-D Example

To validate our approach and demonstrate learning tree-based policies and the effects of varying rejection thresholds, we provide a simple synthetic experiment with 1-dimensional data. Specifically, we assume $\mathcal{X} = \mathbb{R}$, and there are two models, M_1 and M_2 with reward functions $R(M_1, x, y) = \exp(-\frac{1}{2}(x-4)^2)$ and $R(M_2, x, y) = \exp(-\frac{1}{2}(x-8)^2)$. We then draw $n = 500$ samples uniformly over the interval $[0, 12]$ and fit our OP^{2}T model to this data. In addition, we add rejection models with $\alpha = 0.1$ and $\alpha = 0.3$. A visualization of the reward surfaces is given in Figure 4, and a visualization of the policy trees generated is given in Figure 5. The resulting trees match the splits we would expect which are highlighted in the plot.

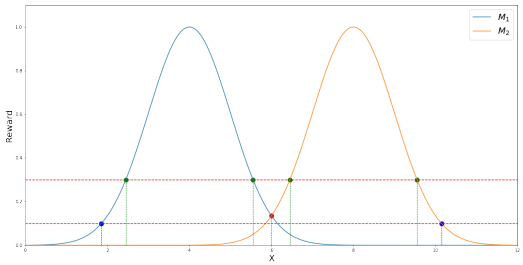


Figure 4: A simple 1-D example of synthetic model rewards with different rejection thresholds, denoted by the red dashed horizontal lines. In this case, the feature space $\mathcal{X} = [0, 12]$ and the rejection parameters are $\alpha = 0.1$ and $\alpha = 0.3$.

By making a small change to the previous example, we can also construct a setting that demonstrates the difference between learning an OP^{2}T and a Meta-Tree. Consider now two models, M_1 and M_2, over the feature space $\mathcal{X} = [0, 18]$ with reward functions $R(M_1, x, y) = \exp(-\frac{1}{2}(x-4)^2) + 0.01$ and $R(M_2, x, y) = \exp(-\frac{1}{2}(x-8)^2)$. This modification creates an interval, $[11, 18]$, over which model M_1 achieves a marginally higher reward than M_2. A visualization of the reward surfaces is given in Figure 6, shaded corresponding to the dominant model in each region. We sample $n = 500$ points uniformly over $[0, 18]$ and generate a policy tree using our OP^{2}T approach, given in Figure

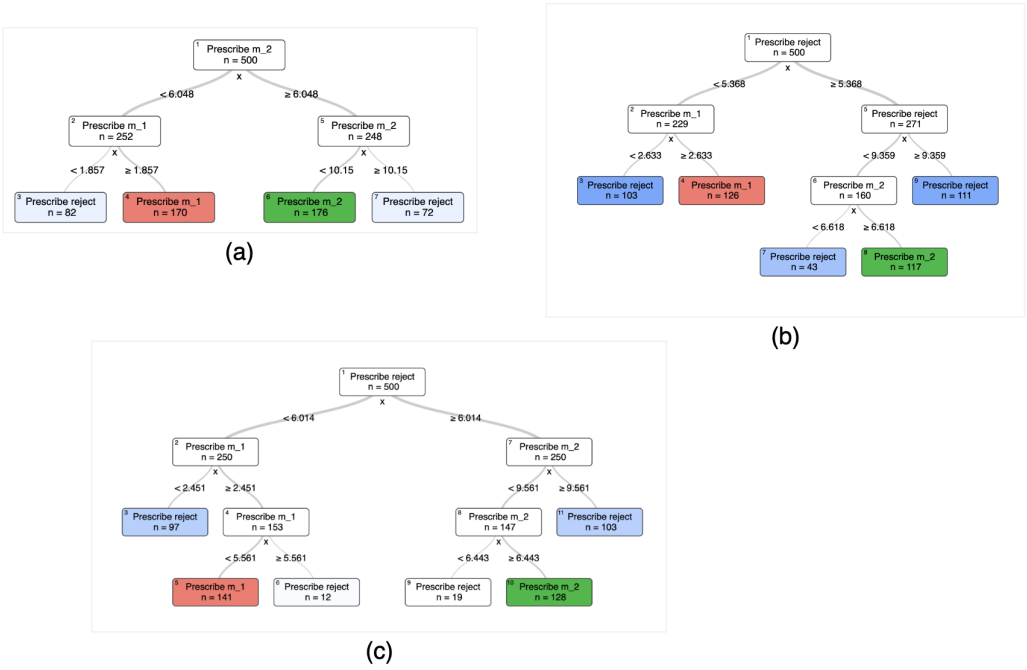


Figure 5: OP^{2}T models fit on the 1-D synthetic data described in Section A.2. Tree (a) corresponds to no rejection, (b) to rejection with $\alpha = 0.1$, and (c) to rejection with $\alpha = 0.3$.

7. Notice that in the right sub-tree, the prescription at the inner node is M_2, despite model M_1 yielding a greater reward than M_2 for the majority of the samples falling into this sub-tree. Were we to take a predictive, classification approach, and label each sample by the model that maximizes reward, the prediction at this node would be M_1. This would ignore the fact that the average reward for M_2 over the interval $[6, 18]$ is significantly greater than M_1 (in this case, the difference in total reward is 2.27, and in expectation is 0.1856). We can see this difference in Figure 7(a), where the Meta-Tree approach prescribes model M_1 at the root, left, and right subtrees. Therefore, while both trees perfectly separate the data, the depth-1 subtree of the OP^{2}T, starting from the root, achieves a total reward of 4.959, while the depth-1 subtree of the Meta-Tree achieves a total reward of 2.686. The result of this example is consistent with our analysis in Section C.2. We also observe another, more dramatic example of this phenomenon in Section B.1.

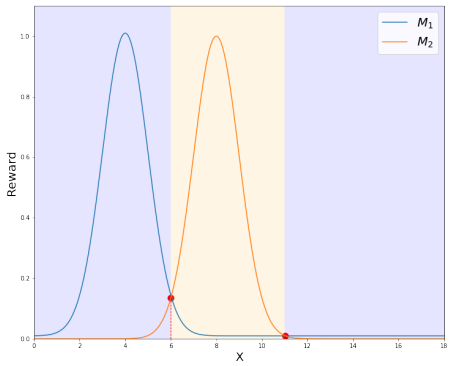
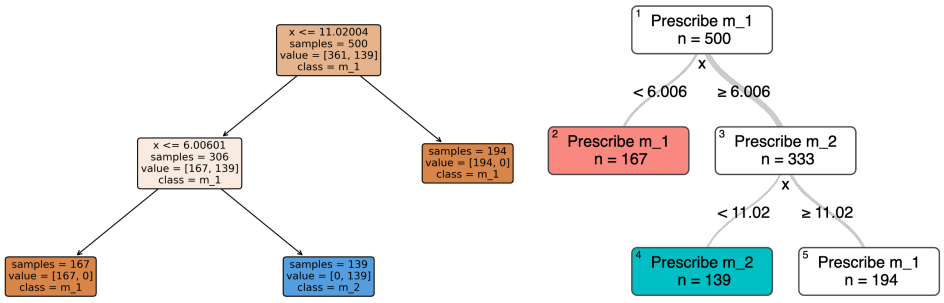


Figure 6: A simple 1-D example of synthetic model rewards demonstrating the potential difference between taking a prescriptive versus a predictive approach to model selection.



(a) Meta-Tree (b) OP^{2}T

Figure 7: OP^{2}T and Meta-Tree models fit on the 1-D synthetic data corresponding to the rewards shown in Figure 6.

B Additional Experiments

In this section, we provide two additional experiments, one for regression and one for classification. Through the classification experiment, we demonstrate how our approach can be used for tasks with unstructured data by using the model outputs as our feature space for learning the model selection policy.

B.1 Projectile Motion with Drag

Motivated by recent work that has successfully combined physics-based and ML models for predicting the dynamics of physical systems [Boussioux et al., 2022], we develop a synthetic experiment to demonstrate how OP^{2}Ts can adaptively prescribe these models based on the conditions of the physical system, significantly reducing the overall error. The setup is as follows. Suppose we are launching a projectile in 2-D space with a fixed mass $m = 1$ from ground level, at the origin $(0, 0)$, with a launch angle θ and initial speed v_0. Further, we are going to assume that there is some air resistance parameterized by a scalar c which we will soon define. We are interested in estimating the total ground distance covered by the projectile as a function of θ, v_0 and c. Let x_f denote the final position of the projectile when it hits the ground. In the absence of air resistance or other effects, this can be found to be

$$\hat{x}_f = \frac{v_0 \sin(2\theta)}{g}. \tag{14}$$

Now, we incorporate a notion of drag by assuming that air resistance occurs in the opposite direction of the projectile's velocity, and its magnitude is directly proportional to the current speed. The equation of motion for the projectile is then given by

$$\mathbf{F} = m\mathbf{a} = m\frac{d\mathbf{v}}{dt} = m\mathbf{g} - c\mathbf{v}, \tag{15}$$

where $\mathbf{g} = (0, -g)$ is the gravitational force, and $\mathbf{v}$ is the velocity of the projectile. With these dynamics defined, the vertical position of the projectile at time t is given by

$$y(t) = \frac{v_t}{g}(v_0 \sin(\theta) + v_t)(1 - \exp(-\frac{gt}{v_t})) - v_t t, \tag{16}$$

where $v_t = mg/c$ is the terminal velocity. The horizontal position is then given by

$$x(t) = \frac{v_0 v_t \cos(\theta)}{g}(1 - \exp(-\frac{gt}{v_t})). \tag{17}$$

Then, by finding $t^* > 0$ such that $y(t^*) = 0$, we can find $x_f = x(t^*)$. We also notice that in the limit, as $t \to \infty$, we have

$$\bar{x}_f = \frac{v_0 v_t \cos(\theta)}{g}, \tag{18}$$

which gives us an upper bound on the ground distance traveled by the projectile in the presence of this drag force. From this analysis, we now have two closed-formed estimates of x_f as a function of θ, v_0, and c, namely Eqs. (14) and (18). For brevity, we refer to the estimate from (14) as $\hat{x}_f$ and (18) as $\bar{x}_f$. We want to create a pattern-based ML model that learns to predict x_f using "real-world" samples. Assuming that, perhaps due to physical limitations, the maximum initial speed at which we can launch the projectile is $100\ m/s$, we can define our feature space as $\mathcal{X} = [0, 100] \times [0, \pi/2] \times [0, 1]$, since $v_0 \in [0, 100]$, $\theta \in [0, \pi/2]$, and $c \in [0, 1]$. We then discretize the feature space and randomly sample $n = 20k$ data points from $\mathcal{X}$. For each sample, we find the "ground-truth" value for x_f using Eqs. (17) and (16), and use these values as our target. Then, splitting the data into 50% training, 25% validation, and 25% test sets, we fit an MLP model h with hidden sizes $[16, 32, 16]$ on the training data. We chose the MLP model class because of its ability to model smooth, nonlinear functions. Our constituent models are then $\hat{x}_f$, $\bar{x}_f$, and h. For this experiment, we do not consider model ensembles, so we set $\mathbf{W} = \text{diag}(1, 1, 1)$. We fit our OP^{2}T model with the predictions from these constituent models on the validation set. The results, without the rejection option, are given in Table 4.

Model	MSE
Physics (no drag)	59098.38
Physics (drag)	188948.88
MLP	241.16

Table 3: Out-of-sample MSE for the constituent models of the projectile motion dataset. Notice how the closed-form physics equations are extremely poor estimators on average over the entire feature space.

Model	No Rejection		Rejection w/ $\alpha = 250$		Rejection w/ $\alpha = 50$	
	MSE	Reject (%)	MSE	Reject (%)	MSE	Reject (%)
Meta-XGB	140.85	0	28.65	9.96	5.56	31.32
Meta-Tree	150.56	0	36.19	12.14	7.67	34.20
OP^{2}T	141.74	0	31.53	11.54	6.63	32.8

Table 4: Out-of-sample MSE on the projectile motion dataset across different thresholds for rejection.

What is most surprising from these results is that despite the closed-form physics-based equations for x_f performing terribly on average, as shown in Table B.1, our OP^{2}T approach successfully uncovers the relatively small regions of $\mathcal{X}$ within which these estimates are very close to the ground-truth, and uses this to drastically reduce the MSE compared to the MLP model alone. Further, the OP^{2}T significantly outperforms the Meta-Tree, especially for shallow tree depths. It is also not obvious a priori that such regions would exist. Since the MLP model has been trained on data sampled across the entire feature space, it is possible for this model to fit all of the data quite well. In practice, however, in minimizing the average error, the MLP has made some trade-offs and cannot compete with the physics-based equations in some regions of $\mathcal{X}$. We also observe that for this example, the Meta-XGB approach yields the best performance in terms of MSE and percent of samples rejected. This agrees with our theoretical analysis in Sections C.2 and C.3, as the boosted tree model is more capable of fitting complex, nonlinear boundaries, such that the relative rewards of the models in each partition is not significant. A primary conclusion of Proposition 4 is that the OP^{2}T has an advantage in the non-realizable setting when we cannot perfectly separate the feature space by the best-performing model. In this case, the Meta-XGB approach is successful in learning more complex partitions of the feature space. However, the Meta-XGB approach lacks interpretability, so we cannot gain insights from the resulting boosted tree model. Standard explainability methods for boosted trees, such as plotting relative feature importance, are not particularly useful in this setting either. We aim to answer questions regarding which model to use, and when, not which features are generally important.

We observe that the OP^{2}T approach significantly outperforms Meta-Tree, demonstrating the impact of taking a prescriptive approach. Further, by restricting the depth of the trees, we can observe an even more drastic disparity between the OP^{2}Ts and Meta-Trees. For a depth-2 tree with a minimum leaf size of 50, our OP^{2}T model achieved an MSE of 183.26, while the Meta-Tree achieved an MSE of 79344.09. A visualization of these trees is given in Figure 8. This result is due to the extremely poor accuracy of the closed-form physics equations when the conditions of the system are far from the assumptions made for them, and the Meta-Tree only being able to consider which model is best for each sample, rather than the relative reward for each model. The Meta-Tree finds partitions of

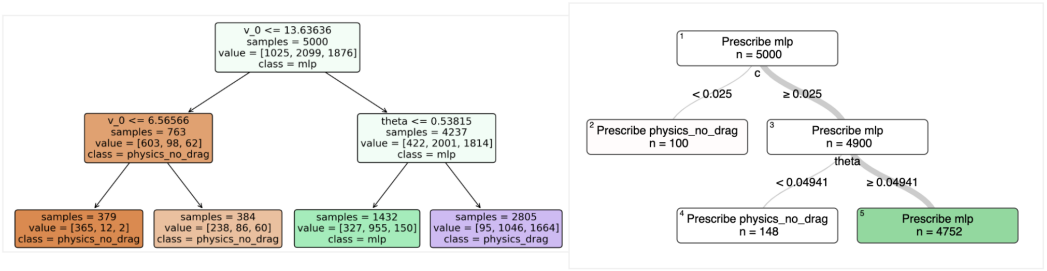


Figure 8: Left: A depth-2 Meta-Tree for the projectile motion problem. Right: A depth-2 OP^{2}T for the same problem. The Meta-Tree achieves an out-of-sample MSE of 79344.09, while the OP^{2}T achieves an MSE of 183.26. The best single model in hindsight, the MLP, achieves an MSE of 241.16. This demonstrates the benefit of prescription over prediction in this setting.

Max Depth	MSE	
	Meta-Tree	OP^{2}T
2	79344.09	183.26
3	317.13	165.95
4	216.21	162.82
5	171.307	156.21

Table 5: Out-of-sample MSE for the Meta-Tree and OP^{2}T approaches for shallow tree depths on the projectile motion dataset. Such shallow structures are necessary in settings where interpretability is crucial.

the space where the physics equations perform the best for the majority of samples (minimizing entropy), but the predicted model performs orders of magnitude worse on the remaining samples in the partition. For example, the Meta-Tree predicts $\bar{x}_f$, the physics equation with drag, when $v_0 \geq 13.63$ and $\theta \geq 0.538$. From the data used to train the tree, 2805 samples fall into this partition, of which a majority ($N = 1664$) are best predicted by $\bar{x}_f$. However, the average reward (MSE) for $\bar{x}_f$ over the samples in this leaf node is 135197.52, while the average reward for the MLP model is 220.38, orders of magnitude lower. The depth-3 Meta-Tree eliminates most of this error by replacing this leaf with a split on the drag coefficient, c, only predicting $\bar{x}_f$ when $c \geq 0.295$. In contrast, we see in Figure 8 that the depth-2 OP^{2}T first splits on $c \leq 0.025$, essentially determining if the drag force is negligible, matching the true physical assumption made to derive Eq. (14). We provide a comparison of the Meta-Tree and OP^{2}T for shallow depths in Table 5. Since trees become less interpretable with increasing depth, the performance of shallow trees is crucial for applications that require interpretability. For this dataset, we can see that the OP^{2}T approach yields shallow trees that are significantly more accurate than the Meta-Tree approach, providing a concise, interpretable policy for selecting whether to use a physics-based model or a pattern-based deep learning model.

B.2 IMDb Sentiment Analysis: Glass-box vs. Black-box Models

The IMDb Movie Review Dataset, introduced by Maas et al. [2011], is a collection of $n = 50k$ movie reviews from the IMDb website, along with binary sentiment labels, indicating positive or negative sentiment. When Maas et al. [2011] introduced this dataset, they developed and trained a Word2Vec model for sentiment prediction on this dataset. This approach preserved interpretability, allowing for both faithful explanations of model predictions as well as the ability to investigate the learned word embedding space. However, with the development of deep contextual encoders, such as BERT, introduced by Devlin et al. [2019], along with large public model repositories like HuggingFace, it is now possible to download a BERT model fine-tuned on this dataset in a matter of minutes that significantly outperforms the original Word2Vec model on this dataset across every standard metric. In the context of high-stakes decision-making problems, however, it is often the case that interpretable models are preferred. This naturally leads to the problem of how to best balance interpretability and performance, which has been well-studied and debated in the literature [Rudin et al., 2021]. In Singh et al. [2023], the authors develop an interpretable language model for sentiment analysis and explore

Model	No Rejection		Rejection ($\alpha = 0.15$)		Rejection ($\alpha = 0.10$)		Rejection ($\alpha = 0.02$)	
	Acc	Reject	Acc	Reject	Acc	Reject	Acc	Reject
Meta-XGB	93.44	0	97.27	16.44	98.04	24.43	98.53	49.17
Meta-Tree	92.78	0	97.22	16.66	98.01	25.02	98.50	49.30
OP^{2}T	93.53	0	97.44	15.34	98.14	23.03	99.13	48.63

Table 6: Out-of-sample accuracy on the IMDb Movie Review dataset across different thresholds for rejection.

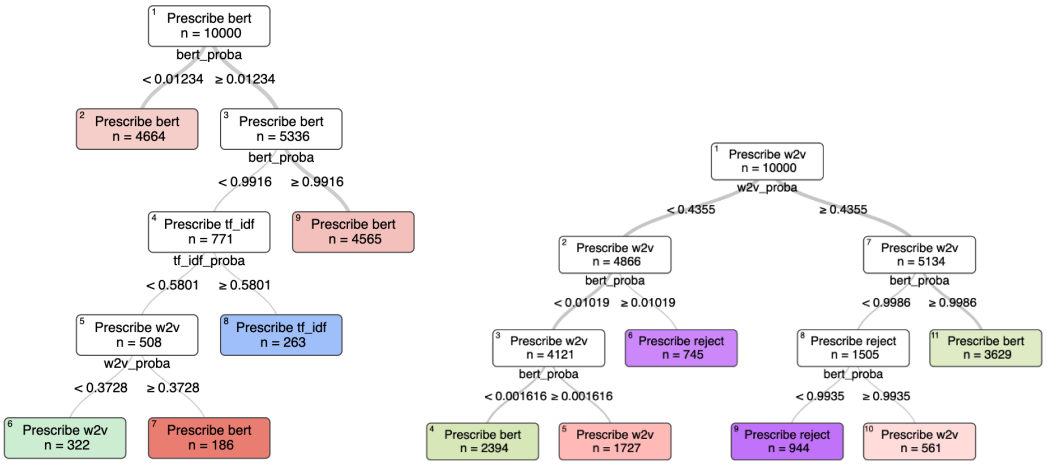
the idea of adaptively selecting either their interpretable model or a black-box, BERT model. They do this by manually fixing confidence thresholds for deferring to a BERT model based on the output of the interpretable model. In their experiments, they demonstrate that performance can be preserved while using the interpretable model for a significant portion of samples across a variety of prediction tasks. However, such a process is inefficient, as it requires manually testing various thresholds, and it only considers the confidence of the interpretable model, rather than considering the confidence of the interpretable model and BERT model jointly. The problem is further complicated when there are more than two models under consideration.

In the context of natural language processing (NLP) and sentiment analysis, there are a variety of interpretable modeling approaches. In this experiment, we consider three models: a BERT model, a Word2Vec model, and a logistic regression model using term-frequency inverse-document-frequency (TF-IDF) features. The latter two models are fairly interpretable, as they are linear functions over the words contained in the reviews. All models were fit on the IMDb training dataset ($n = 25k$ samples). To fit the policy models, we sampled $n = 10k$ samples from the IMDb test set, and used the remaining $n = 15k$ samples as our final test set. For this application, since the dataset is balanced and the primary metric is accuracy, we use the misclassification reward R_{MIS}, to fit our OP^{2}T models. Since we are interested in answering the question of interpretability, we only consider selecting the individual constituent models, not ensembles, so we set $\mathbf{W} = \text{diag}(1, 1, 1)$. For the feature space $\mathcal{Z}$, we consider the space of constituent model outputs, which we will also refer to as the model confidence scores. We originally considered extending this feature space with additional, structured meta-data from the reviews (e.g. review length, keywords, negation). However, we found that for this setup, the resulting OP^{2}Ts never used this meta-data, so we focus instead on the feature space being the model confidence scores alone.

The main results of our experiments are given in Table 6. The best constituent model in hindsight was the BERT model, achieving an out-of-sample accuracy of 92.8. We can then observe that our OP^{2}T approach outperforms the best model in hindsight as well as the Meta-XGB and Meta-Tree approaches across all levels of rejection. Further, the policies learned by our method find partitions of the data where the interpretable models empirically maximize the expected reward. Therefore, with access to model outputs alone, we can construct an interpretable policy for adaptive model selection that outperforms the black-box baseline model and makes use of the interpretable models.

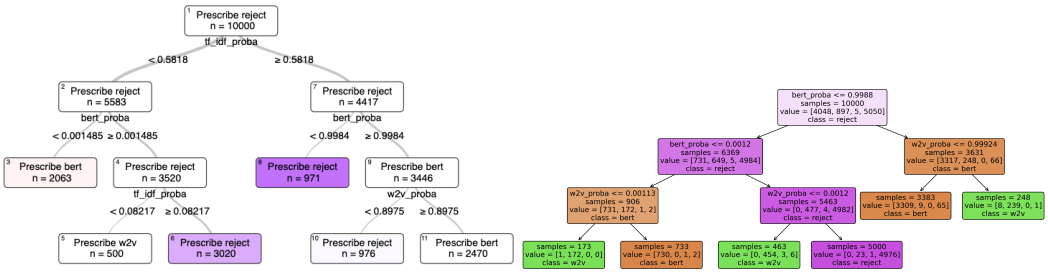
A visualization of the learned OP^{2}Ts is given in Figure 9. We see that over the space of model confidence scores, our OP^{2}T approach finds intersections of confidence intervals over all three of our models to provide prescriptions that outperform the Meta-Tree and Meta-XGB approaches. For example, in tree (a) in Figure 9, we see that the learned policy is to use the BERT model if its output score is outside the interval $(0.012, 0.992)$, otherwise, query the interpretable models. Notice that rather than simply deferring to one of the interpretable models, the policy looks at the confidence scores of both interpretable models, and will still prescribe the BERT model if the TF-IDF model is not confident the sample is positive while the Word2Vec model is not confident the sample is negative. Similarly, tree (b) contains intersections of confidence intervals between the Word2Vec and BERT models, prescribing rejection when the BERT model is not confident and in agreement with the Word2Vec model. Since many of the resulting OP^{2}Ts are composed of intersections of all three constituent models' confidence scores, the policies generated for model selection would not feasibly be discovered by manually searching over confidence intervals.

In Figure 10, we give a visualization of the OP^{2}T and corresponding Meta-Tree generated with $\alpha = 0.02$. The OP^{2}T achieves higher out-of-sample accuracy while rejecting fewer samples. We provide this visualization to highlight the structural differences that can occur between Meta-Trees and OP^{2}Ts. By optimizing tree structure globally and considering relative rewards, the OP^{2}T makes



(a) No Rejection (b) Rejection $\alpha = 0.15$

Figure 9: Visualizations of the OP^{2}Ts fit on the IMDb dataset using the original feature space extended with the constituent model scores. We observe that the best performing OP^{2}Ts rely exclusively on the model scores.



(a) OP^{2}T with $\alpha = 0.02$ (b) Meta-Tree with $\alpha = 0.02$

Figure 10: Visualization of an OP^{2}T and a Meta-Tree fit on the IMDb dataset with a high rejection reward ($\alpha = 0.02$).

use of the confidence scores of all three constituent models and identifies three separate regions within which rejection is prescribed. In contrast, the Meta-Tree, grown using a greedy heuristic and unable to consider the relative rewards of each model, finds simpler conditions for rejection, resulting in lower accuracy and more samples being rejected. Our results are also consistent with our theoretical analysis in Section C.2, as we see a smaller gap between OP^{2}T and the alternative approaches due to the small difference on average in relative reward between the constituent models. To summarize, we can connect our results back to our motivating questions as follows.

Should an interpretable model be used? Yes, in many cases the interpretable models can outperform the black-box BERT model. Specifically, when the BERT model disagrees with the predictions of the interpretable models, and the BERT model is not very confident, using one of the interpretable models improves performance. In Figure 9, tree (b) prescribes the Word2Vec model for 23% of samples.

When are the models likely to be error-prone? In short, when the models disagree, and none of the models are very confident, all of the models are likely to be error-prone. The resulting trees give specific conditions for these cases, such as in Figure 10, OP^{2}T (a) prescribes rejection when the output of the TF-IDF model is in the interval $[0.082, 0.582]$ and the BERT model output is in the interval $[0.0015, 1]$.

C Theoretical Insights

In this section, we formalize some of the concepts introduced in this work and provide theoretical insights for our approach. We discuss sufficient conditions under which OP^{2}T models can learn policies that improve over the best model in hindsight, the setting where OP^{2}Ts can outperform the Meta-Tree and Meta-XGB approaches, and the impact of rejection learning on the resulting policies. Throughout this section, we consider a fixed set of m constituent models $\mathcal{H} = \{h_1, \ldots, h_m\}$. We do not consider model ensembles separately, as any model ensemble can be represented as another constituent model, and this simplifies our notation. Our setup consists of an original feature space $\mathcal{X}$ over which the constituent models $\mathcal{H}$ are defined and a secondary feature space $\mathcal{Z}$ over which some policy model π is defined. We first introduce the following definition:

Definition 1. ***(Informative Policy)*** *For some policy function $\pi : \mathcal{Z} \to \mathcal{H}$, function $g : \mathcal{X} \to \mathcal{Z}$, data distribution $\mathcal{D}$ over $\mathcal{X}$, true target function $f^* : \mathcal{X} \to \mathcal{Y}$, and set of constituent models $H = [h_1, \ldots, h_m]$, we refer to the policy as **informative** if*

$$E_{\mathcal{D}}[R(x, f^*(x), \pi(g(x)))] > \max_{h \in \mathcal{H}} E_{\mathcal{D}}[R(x, f^*(x), h)]. \tag{19}$$

A policy is **informative** if it finds a meaningful partition of the space $\mathcal{Z}$ such that the expected reward under π is greater than the expected reward for the best single constituent model. The function g defines the relationship between samples in the original features space $\mathcal{X}$ and samples in the feature space $\mathcal{Z}$. This definition is helpful for reasoning about the potential effectiveness of adaptive model selection, regardless of the specific approach.

C.1 Sufficient Conditions for Learning an Informative Policy

Our first goal is to determine the conditions under which an informative policy exists. These conditions help us understand when an adaptive policy can outperform static model selection. Intuitively, an informative policy exists when there are separate regions of the feature space over which different models perform the best in terms of relative reward. To formalize this setting, we introduce the following definition:

Definition 2. ***(Dominated Subspace)*** *Let $\mathcal{Z} \subseteq \mathbb{R}^d$ be the feature space used to learn some policy function π, $\mathcal{D}$ some distribution over $\mathcal{X}$, $v : \mathcal{Z} \to 2^{\mathcal{X}}$ a function mapping elements of $\mathcal{Z}$ to disjoint subsets of the original feature space $\mathcal{X}$, and $f^* : \mathcal{X} \to \mathcal{Y}$ be the true target function. Suppose there are m constituent models $\mathcal{H} = \{h_1, \ldots, h_m\}$. Let us define the function $g : \mathcal{X} \to \mathcal{Z}$ as the unique function such that for all $x \in \mathcal{X}$, we have $x \in v(g(x))$. We refer to a connected subspace $A \subset \mathcal{Z}$ as a **dominated subspace** if $P_{\mathcal{D}}(\cup_{z \in A} v(z)) \in (0, 1)$ and there exists a model $h^*_A \in \mathcal{H}$ such that for all $h \in \mathcal{H} \setminus h^*_A$,*

$$E_{\mathcal{D}}[R(x, f^*(x), h^*_A) \cdot \mathbb{1}\{g(x) \in A\}] > E_{\mathcal{D}}[R(x, f^*(x), h) \cdot \mathbb{1}\{g(x) \in A\}].$$

A **dominated subspace** is any connected subspace over which there is a single, best-performing model in expectation. Since we are reasoning about multiple reward surfaces jointly, it is natural to consider partitioning the feature space into these dominated subspaces. The functions g and v define the relationship between the feature spaces $\mathcal{X}$ and $\mathcal{Z}$. Notice that we assume the function v maps to disjoint subsets of $\mathcal{X}$. That is, for all $z_1, z_2 \in \mathcal{Z}$ such that $z_1 \neq z_2$, $v(z_1) \cap v(z_2) = \emptyset$. This assumption gives g the property of uniqueness. In many settings, this many-to-one relationship between $\mathcal{X}$ and $\mathcal{Z}$ is a reasonable assumption. For example, suppose $\mathcal{X}$ is unstructured language data (sequences of characters up to some maximum length) and $\mathcal{Z}$ is structured meta-data about this language data, such as the length of the sequence or specific character counts. Then an element $z \in \mathcal{Z}$ can map to many sequences in $\mathcal{X}$, but each sequence $x \in \mathcal{X}$ has one specific element in $\mathcal{Z}$ corresponding to its meta-data. Further, the elements of $\mathcal{Z}$ map to disjoint subsets of $\mathcal{X}$. Another setting with this property is when $\mathcal{Z}$ is the outputs of the constituent models H. With these definitions, we can then formulate the following proposition.

Proposition 3. *Let $\mathcal{Z} \subseteq \mathbb{R}^d$ be the feature space used to learn some policy function π, $\mathcal{D}$ some distribution over $\mathcal{X}$, a function $v : \mathcal{Z} \to 2^{\mathcal{X}}$ mapping elements of $\mathcal{Z}$ to disjoint subsets of the original feature space $\mathcal{X}$, and the unique function $g : \mathcal{X} \to \mathcal{Z}$ such that for all $x \in \mathcal{X}$, we have $x \in v(g(x))$. There exists an **informative policy** if there exist two disjoint dominated subspaces $A, B \subset \mathcal{Z}$ with corresponding models h^*_A and h^*_B, such that $h^*_A \neq h^*_B$.*

The proof is given in Appendix C.4.1. This proposition gives us clear sufficient conditions for the existence of an informative policy. Namely, given some assumptions, if there exist at least two disjoint dominated subspaces in $\mathcal{Z}$ corresponding to different constituent models, then an informative policy exists. In addition to this result, it is clear that if there exists a model $h^* \in H$ such that $R(x, f^*(x), h^*) \geq \max_{h \in H} R(x, f^*(x), h)$ for all $x \in \mathcal{X}$, then an informative policy cannot be found, as one model strictly dominates the rest.

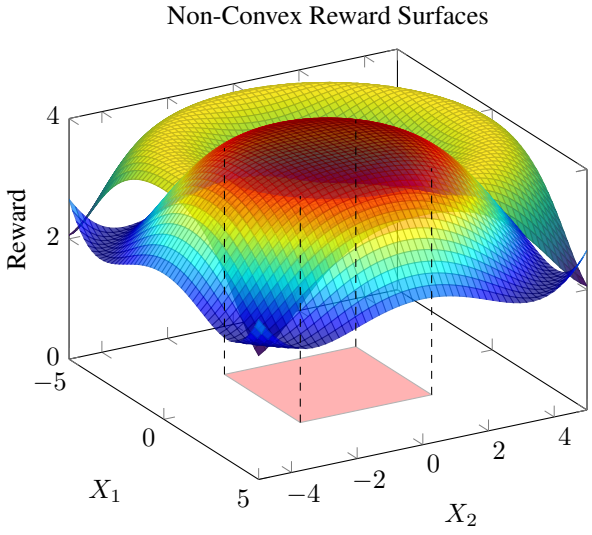


Figure 11: An example of two non-convex reward surfaces with a square dominated subspace.

We can apply this result to reason about the quality of OP^{2}T models. For example, when $\mathcal{Z} = \mathbb{R}$, connected subspaces are intervals (a, b). Any such interval can be expressed with a depth-2 tree using parallel splits. Further, any pair of disjoint intervals, (a, b) and (c, d), can be expressed with a depth-3 tree using parallel splits. Therefore, it follows from Proposition 3 that if there exist two disjoint dominated subspaces in $\mathcal{Z} = \mathbb{R}$, an informative policy can be learned by a depth-3 OP^{2}T. More generally, it follows that if there exist two disjoint dominated subspaces that can be formed by parallel splits of a depth D_{max} decision tree, then OP^{2}Ts can recover an informative policy. Note that the reward surfaces can be highly nonlinear as long as there exist dominated subspaces that can be defined by parallel splits generating an informative policy. For example, in Figure 11, we show two non-convex reward surfaces, $f_1(X_1, X_2) = \sin(5(X_1^2 + X_2^2)) + 3$ and $f_2(X_1, X_2) = \cos(5(X_1^2 + X_2^2)) + 3$. Despite the surfaces being non-convex, there exists a dominated subspace that can be constructed by a depth-4 decision tree with axis-aligned splits.

C.2 The Advantage of Prescription over Prediction

So far, we have reasoned about the combined reward surfaces for the constituent models and conditions for learning an informative policy with decision trees. The next question is then, what is the benefit of our prescriptive approach? In Section 2.3, we introduce two alternative approaches which we refer to as Meta-Tree and Meta-XGB. These approaches do not consider the relative rewards of the constituent models. Instead, the problem is framed as a multi-class classification task, where each class corresponds to a different model or ensemble. Specifically, for models and ensembles $[h_1, \ldots, h_M]$, and data $\{(x_i, y_i)\}_{i=1}^n$, we assign sample x_i a label $q_i \in [M]$ as

$$q_i = \arg\min_{i \in [M]} |y_i - h_i(x_i)|, \tag{20}$$

where we break ties arbitrarily (e.g. smallest index). These methods are simpler to implement, so it is important to quantify the advantage of using our prescriptive approach instead. To answer this question, we have the following result.

Proposition 4. *Under reward maximization, suppose we have a set of constituent models $\mathcal{H} = \{h_1, \ldots, h_m\}$, data $\{(x_i, y_i)\}_{i=1}^n \in \mathcal{X}^n$, function $g : \mathcal{X} \to \mathcal{Z}$, and a reward function $R : \mathcal{X} \times$*

*$\mathcal{Y} \times \mathcal{H} \to \mathbb{R}$ with either no finite lower bound or upper bound. Then the difference in total reward, $\sum_{i=1}^{n} R(x_i, y_i, T_O(g(x_i)) - \sum_{i=1}^{n} R(x_i, y_i, T_M(g(x_i))$, between the corresponding OP^{2}T, T_O, and a Meta-Tree, T_M, **can be arbitrarily large**. For a fixed dataset, if we define $R_{max} = \max_{i\in[n],(j,k)\in[m]} |R(x_i, y_i, h_j) - R(x_i, y_i, h_k)|$, we have the upper bound*

$$\frac{1}{n}\sum_{i=1}^{n} R(x_i, y_i, T_O(g(x_i)) - \frac{1}{n}\sum_{i=1}^{n} R(x_i, y_i, T_M(g(x_i)) \leq \frac{m-1}{m} R_{max}.$$

The same result holds for the corresponding Meta-XGB model.

The proof is given Appendix C.4.2. The key insight from the proof is that the OP^{2}T approach can significantly outperform the Meta-Tree approach when the data $\{(x_i, q_i)\}_{i=1}^{n}$ is not separable such that we are in the non-realizable setting with the decision tree hypothesis class. In these settings, a trade-off must be made between model prescriptions. The OP^{2}T approach considers more information regarding the relative reward of each constituent model, resulting in a prescription that can be arbitrarily better than the one made by the Meta-Tree approach. Note that the condition that R must have either no finite lower bound or upper bound is satisfied by both R_{CE} and R_{SE}. Importantly, and more practically, the difference in expected reward between the OP^{2}T approach and the alternative approaches is bounded by the magnitude of the relative difference in rewards between the constituent models. When the relative difference in reward between the models and ensembles is small, we can expect minimal differences between the approaches. However, in the non-realizable setting, for large differences in reward, we expect the OP^{2}T to outperform the alternative approaches. While the same results hold for Meta-XGB, we note that boosted trees are a more expressive model class, such that they can separate data better than decision trees. Therefore, we expect a smaller gap between OP^{2}Ts and the Meta-XGB approach in terms of total reward. We give an example highlighting this phenomenon in Section B.1, in addition to showing a similar result on a 1-D toy dataset in Section A.2.

C.3 The Quality of OP^{2}T Policies and the Impact of Rejection Learning

In this section, we investigate the impact of rejection learning on adaptive model selection. We do so through the lens of counting the number of dominated subspaces. Intuitively, as the number of regions with different dominant models increases, more expressive policy functions are required to partition the feature space accordingly. Conversely, rejection learning can serve as a mechanism to decrease the number of dominated subspaces, specifically those with low expected reward. To help reason about the number of dominated subspaces, we introduce the following definition.

Definition 5. ***(Maximal Dominated Subspace)*** *A dominated subspace A is a maximal dominated subspace if for all dominated subspaces B such that $A \subset B$, $h_A^* \neq h_B^*$.*

It is important to distinguish these dominated subspaces for the purpose of counting, as many dominated subspaces, such as an interval $(a, b) \subset \mathbb{R}$, $a < b$, may imply the existence of infinite, nested dominated subspaces, all corresponding to the same constituent model. Even with these conditions, there may also exist many, or even infinite, maximal dominated subspaces. In the example depicted in Figure 11, if we take the feature space to be all of $\mathbb{R}^2$, there are infinite maximal dominated subspaces. Practically, we may not expect such a case to occur in reality, but given the likely non-convexity of reward surfaces, it is possible that there are many maximal dominated subspaces. Given that an informative policy exists, the quality of our tree-based approach depends on the number of maximal dominated subspaces and the shape and dimension of these subspaces. In the case that the set of maximal dominated subspaces, S, is small (i.e., $|S| << 2^{D_{max}}$) and each subspace $s \in S$ is representable by D_{max} parallel splits, an OP^{2}T model of depth D_{max} could learn a policy that approximately recovers all of these subspaces. Conversely, if there are many maximal dominated subspaces that are high dimensional and nonlinear, there will be a gap in expected reward between the policy learned by an OP^{2}T and the true optimal policy function. We note, however, that extending our formulation to include hyperplane splits is simple, and we implemented this version of the model, although empirically we did not see a benefit. These hyperplanes splits would allow the OP^{2}Ts to represent more complex dominated subspaces with fewer splits.

The inclusion of a rejection model can simplify the task of learning informative policies while adding a form of regularization. Formally, we make the following statement:

Proposition 6. *Under reward maximization, given some data distribution $\mathcal{D}$ over $\mathcal{X}$ and reward function $R : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}$ with finite upper bound R_{ub} such that $R(\cdot) \leq R_{ub}$, and constituent models $H = \{h_1, \ldots, h_m\}$, $h_i : \mathcal{X} \to \mathcal{Y}$, and true target function $f^* : \mathcal{X} \to \mathcal{Y}$, suppose there exist finite maximal dominated subspaces. Further, let us define a rejection model h_r^n such that $R(x, y, h_r^n) = \alpha_n$. We denote the set of maximal dominated subspaces, not including the regions dominated by the rejection model h_r^n, by $S^n = \{s_1^n, \ldots, s_q^n\}$. Given an increasing sequence of rejection parameters $\{\alpha_n\}$ such that $\alpha_n \to R_{ub}$, we have the following properties:*

1. *If $P_D(R(x, f^*(x), h_i) = R_{ub}) = 0$ for all $i \in [m]$, then $|S^n| \to 0$ as $n \to \infty$.*
2. *For all dominated regions s_i^n, s_j^{n+1} such that $s_j^{n+1} \subseteq s_i^n$, we have $P_D(s_i^n) \geq P_D(s_j^{n+1})$ and $E_D[R(x, f^*(x), h^*_{s_j^{n+1}})|x \in s_j^{n+1}] \geq E_D[R(x, f^*(x), h^*_{s_j^n})|x \in s_j^{n+1}]$.*

The proof of these properties is brief and is given in Appendix C.4.3. These properties give us a sense of the impact of rejection, specifically that increasing the reward for rejection reduces the total number of maximal dominated subspaces in the limit while also making these subspaces smaller and increasing their expected reward. For example, consider a pair of models that generate reward surfaces such that there are many maximal dominated subspaces, far greater than can be partitioned by a tree of any reasonable depth. By increasing the reward for rejection, we will likely reduce the number of maximal dominated subspaces, allowing the tree to focus on identifying fewer, significant partitions with high expected reward.

C.4 Proofs

C.4.1 Proof of Proposition 3

Proof. We prove this proposition by construction. We first define the model that maximizes the expected reward over all of $\mathcal{X} \sim \mathcal{D}$:

$$h^* = \arg\max_{h \in H} E_{\mathcal{D}}[R(x, f^*(x), h)] \tag{21}$$

Let $C = \mathcal{Z} \backslash (A \cup B)$. There are two cases to consider. The first is when $C = \emptyset$ or $P_D(\cup_{z \in C} v(z)) = 0$. In this case, we can define the following policy function:

$$\pi(z) = \begin{cases} h_A^* & z \in A \\ h_B^* & z \in B \end{cases} \tag{22}$$

When $C \neq \emptyset$ and $P_D(\cup_{z \in C} v(z)) > 0$, we can define

$$h_C^* = \arg\max_{h \in H} E_{\mathcal{D}}[R(x, f^*(x), h) \cdot \mathbb{1}\{x \in g(C)\}], \tag{23}$$

such that

$$E_{\mathcal{D}}[R(x, f^*(x), h_C^*) \cdot \mathbb{1}\{g(x) \in C\}] \geq E_{\mathcal{D}}[R(x, f^*(x), h^*) \cdot \mathbb{1}\{g(x) \in C\}].$$

This model h_C^* does not need to be unique, and it can be that $h_C^* = h_A^*$ or $h_C^* = h_B^*$. Further, C may not be a dominated subspace, as it may not be connected. We can then define the following policy function:

$$\pi(z) = \begin{cases} h_A^* & z \in A \\ h_B^* & z \in B \\ h_C^* & z \in C \end{cases} \tag{24}$$

Since $h_A^* \neq h_B^*$, it must be that either $h^* \neq h_A^*$ or $h^* \neq h_B^*$, or both. This implies that we have either

$$E_{\mathcal{D}}[R(x, f^*(x), h_A^*) \cdot \mathbb{1}\{g(x) \in A\}] > E_{\mathcal{D}}[R(x, f^*(x), h^*) \cdot \mathbb{1}\{g(x) \in A\}]$$

or

$$E_{\mathcal{D}}[R(x, f^*(x), h_B^*) \cdot \mathbb{1}\{g(x) \in B\}] > E_{\mathcal{D}}[R(x, f^*(x), h^*) \cdot \mathbb{1}\{g(x) \in B\}]$$

or both. Putting these inequalities together, we get

$$\begin{aligned} E_{\mathcal{D}}[R(x, f^*(x), \pi(g(x))] = {} & E_{\mathcal{D}}[R(x, f^*(x), h_A^*) \cdot \mathbb{1}\{g(x) \in A\}] \\ & + E_{\mathcal{D}}[R(x, f^*(x), h_B^*) \cdot \mathbb{1}\{g(x) \in B\}] \\ & + E_{\mathcal{D}}[R(x, f^*(x), h_C^*) \cdot \mathbb{1}\{g(x) \in C\}] \\ > {} & E_{\mathcal{D}}[R(x, f^*(x), h^*)]. \end{aligned} \tag{25}$$

By construction, since we assume A and B are disjoint, we know A, B, C are disjoint sets. Let us define $A' = \cup_{z \in A} v(z)$, $B' = \cup_{z \in B} v(z)$, and $C' = \cup_{z \in C} v(z)$. Since we assume the function v maps to disjoint subsets of $\mathcal{X}$, we know A', B', C' are disjoint. It follows that the sets $\{x \in \mathcal{X} : g(x) \in A\}$, $\{x \in \mathcal{X} : g(x) \in B\}$, and $\{x \in \mathcal{X} : g(x) \in C\}$ are disjoint. Further, since g is defined over all $x \in \mathcal{X}$ and $A \cup B \cup C = \mathcal{Z}$, it follows that $P_D(\{x \in \mathcal{X} : g(x) \in A\} \cup \{x \in \mathcal{X} : g(x) \in B\} \cup \{x \in \mathcal{X} : g(x) \in C\}) = 1$. We can therefore split the reward function as in Eq. (25) and by linearity of expectation, we get our result. □

C.4.2 Proof of Proposition 4

Proof. For simplicity, let us consider the case with two constituent models $\mathcal{H} = \{h_1, h_2\}$, and assume $\mathcal{X} = \mathcal{Z}$. Suppose for this set of models $\mathcal{H}$ that there exists a subset of the data $A \subseteq \{(x_i, q_i)\}_{i=1}^n$ that is not separable, either at all, or by parallel split decision trees up to some maximum depth D_{max}. That is, for all decision trees up to D_{max}, the misclassification rate (or entropy) is greater than or equal to the misclassification rate or entropy of A itself. Since the data in A is not separable, the best the Meta-Tree approach can do is prescribe a single constituent model for the data in A. Let us define $A_1 = \{(x_i, q_i) \in A : q_i = h_1\}$ and similarly $A_2 = \{(x_i, q_i) \in A : q_i = h_2\}$. Suppose $|A_1| > |A_2|$, such that h_1 corresponds to the majority class. It follows that the best a Meta-Tree can do over the set A is prescribe the model h_1, as this minimizes the misclassification error. Now, let us define

$$\sum_{x_i \in A_1} R(x_i, y_i, h_1) = R_{11},$$

$$\sum_{x_i \in A_1} R(x_i, y_i, h_2) = R_{12},$$

$$\sum_{x_i \in A_2} R(x_i, y_i, h_1) = R_{21},$$

$$\sum_{x_i \in A_2} R(x_i, y_i, h_2) = R_{22}.$$

Then the total reward for prescribing model h_2 is $R_{12}+R_{22}$, and the total reward for prescribing model h_1 is $R_{11} + R_{21}$. Let $\Delta_1 = R_{11} - R_{12}$, and note that $\Delta_1 \geq 0$ since $R(x_i, y_i, h_1) \geq R(x_i, y_i, h_2)$ for all $i \in A_1$ by definition. Similarly, we can define $\Delta_2 = R_{22} - R_{21}$. We assume R has either no finite upper or lower bound, such that Δ_2 can be arbitrarily large, as we can either send $R_{22} \to \infty$ or $R_{21} \to -\infty$. Then the difference in total reward,

$$R_{12} + R_{22} - R_{11} - R_{21} = \Delta_2 - \Delta_1,$$

can be arbitrarily large as $\Delta_2 \to \infty$. Since the OP^{2}T approach comes from the same hypothesis class, the best an OP^{2}T can do is prescribe a single model over A. However, the model that maximizes total reward over A will be prescribed. Then in our example, since the total reward for h_2 will be greater than h_1 over A as $\Delta_2 \to \infty$, an OP^{2}T will prescribe model h_2. Therefore, the total reward for the OP^{2}T approach can be arbitrarily greater than the reward from the Meta-Tree approach. Next, let us consider a fixed dataset of size n with m constituent models, and define $R_{max} = \max_{i \in [n], (j,k) \in [m]} |R(x_i, y_i, h_j) - R(x_i, y_i, h_k)|$. Using our previous construction, we know for all $i \in A_2$ that $R(x_i, y_i, h_2) - R(x_i, y_i, h_1) \leq R_{max}$. We also observe that $|A_2| < \frac{n}{2}$ since we assume $|A_1| > |A_2|$. It follows that $\Delta_2 < \frac{n}{2} R_{max}$ and, noting that $\Delta_1 \geq 0$, we have

$$\Delta_2 - \Delta_1 < \frac{n}{2} R_{max}.$$

To prove this upper bound in general, suppose to the contrary that for the resulting decision trees T_O and T_M we have

$$\sum_{i=1}^{n} R(x_i, y_i, T_O(g(x_i)) - \sum_{i=1}^{n} R(x_i, y_i, T_M(g(x_i)) > \frac{n(m-1)}{m} R_{max}.$$

Since R_{max} is an upper bound on the difference in reward among all models for the given dataset, we know, for all $i \in [n]$,

$$R(x_i, y_i, T_O(g(x_i)) - R(x_i, y_i, T_M(g(x_i)) \leq R_{max}.$$

It follows that for at least $\frac{n(m-1)}{m}$ samples, $\max_{j\in[m]} R(x_i, y_i, h_j) > R(x_i, y_i, T_M(x_i))$. Since, in the Meta-Tree setup, the samples are labeled according to the constituent model achieving the maximum reward, this implies that T_M misclassifies more than $\frac{n(m-1)}{m}$ samples. However, there must be at least $\frac{n}{m}$ samples in the majority class, such that predicting the majority class alone achieves a lower misclassification error than T_M. This leads to a contradiction, as a depth-zero decision tree would achieve a lower misclassification error than T_M. This gives us our result and provides an intuitive upper bound on the potential gain of using relative rewards as a function of the maximum difference in rewards between constituent models. The same argument holds if we replace the Meta-Tree approach with Meta-XGB. In that case, we can consider a corresponding subset of the data $A \subseteq \{(x_i, q_i)\}_{i=1}^n$ that is not separable by a boosted tree model. □

C.4.3 Proof of Proposition 6

Proof. We begin with the first property. Suppose $P_D(R(x, f^*(x), h_i) = R_{ub}) = 0$. Then for any connected subspace $A \subseteq \mathcal{X}$ with positive measure, $P_D(A) > 0$, we know

$$E_{\mathcal{D}}[R(x, f^*(x), h_i) \cdot \mathbb{1}\{x \in A\}] < R_{ub} \quad \forall i \in [m].$$

It then follows immediately that since $\alpha_n \to R_{ub}$, there exists some N such that for all $n \geq N$,

$$E_{\mathcal{D}}[R(x, f^*(x), h_r^n) \cdot \mathbb{1}\{x \in A\}] > \max_{h\in H} E_{\mathcal{D}}[R(x, f^*(x), h) \cdot \mathbb{1}\{x \in A\}].$$

Since this reasoning holds for any maximal dominated subspace, we have our result.

The first part of the second statement holds by definition since we state $s_j^{n+1} \subseteq s_i^n$. The second part follows from the fact that $s_j^{n+1} \subseteq s_i^n$ and the expected reward for the corresponding dominant model $h^*_{s_j^{n+1}}$ must be greater than α_{n+1}, which is greater than α_n. Since we can always set $h^*_{s_j^{n+1}} = h^*_{s_j^n}$, the expected reward for $h^*_{s_j^{n+1}}$ over s_j^{n+1} must always be at least the same as the expected reward for $h^*_{s_j^n}$. Therefore, the inequality holds. □

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims in the abstract, including methodology and experimental results, are reflected in the paper itself.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: In addition to comparing against other approaches, we provide theoretical results, such as Proposition 4 which explicitly provide an upper bound on the performance gains that can be achieved by our method.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Definitions for new terminology are provided and proofs are given for each proposition.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: Section 2 provides the necessary details to implement our method, and Appendix A.1 provides all details and references necessary to implement the prescriptive tree approach used in this work.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Currently, this work relies on commercial optimization software that is not open source. We hope to soon provide an open-source version of our method, but that is not yet available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We discuss how we trained our models, the relevant hyperparameters, and how they were tuned in Section 3. We also reference all public datasets used and clearly define our setup for all synthetic experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Where relevant, we include the standard error of the rewards achieved by the models.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Compute resources are discussed in Section 3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The authors have read the Code of Ethics and affirm that our work conforms to these guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The societal impact of deploying learning algorithms is central to the motivation for our work, and we discuss these implications and how our approach can have a positive impact on these issues.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All relevant code, models, and data have been properly cited in our paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.