Decision Trees vs. Ensembles in Regression-Based Offline RL

Interpretability–Performance Trade-offs and Return-to-Go Effects

Rareş Polenciuc
Supervisors: Anna Lukina, Daniël Vos
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Rareş Polenciuc
Final project course: CSE3000 Research Project
Thesis committee: Anna Lukina, Daniël Vos, Luciano Cavalcante Siebert

Abstract

Offline reinforcement learning (RL) trains policies from pre-collected data, valuable in scenarios where real-world interaction is costly or risky. This paper systematically investigates the interpretability-performance trade-off of decision tree policies in a framework that reframes offline RL as supervised regression. Through extensive empirical evaluation of single and decomposed decision trees against an XGBoost ensemble on diverse D4RL environments, we show that compact trees, though inherently interpretable, suffer significant performance loss. Conversely, achieving competitive returns with larger trees sacrifices practical human auditability. Critically, return-to-go (RTG) conditioning introduces significant behavioral fragility; policies, despite structural transparency, exhibit unpredictable responses to RTG shifts, complicating their practical interpretability in dynamic environments. Our findings demonstrate that structural simplicity alone is insufficient for practical transparency in goal-conditioned RL, underscoring the need for further research in robustly interpretable sequential decision-making systems.

1 Introduction

Offline reinforcement learning (RL) enables training decision-making policies exclusively from pre-collected datasets, without further online interactions [1, 2]. This approach is particularly valuable in domains where real-world experimentation involves significant risks, costs, or practical barriers. For example, offline RL can be used to derive insulin dosing strategies from healthcare records [1], develop autonomous driving policies based on previously collected sensor data [3], or operate robotic manipulators safely in hazardous industrial environments [4].

Despite these promising applications, current state-of-the-art offline RL methods—such as deep neural networks or large ensembles of decision trees—often produce policies that behave as opaque, "black-box" systems. Their internal reasoning is difficult for humans to interpret or verify [5, 6, 7]. In safety-critical scenarios, such opacity can severely limit user trust, regulatory acceptance, and ultimately, practical deployment. Thus, addressing this interpretability-performance trade-off remains a crucial research area.

Our work builds upon the approach introduced by Koirala and Fleming in an earlier version of their paper [8] (subsequently updated in [9]), where they reframed offline RL as a supervised regression problem, using gradient-boosted decision-tree (GBDT) ensembles to predict actions based on three inputs: the current state, the timestep, and the return-to-go (RTG), defined as the cumulative future reward the agent aims to achieve. While their approach achieves excellent performance across benchmark tasks, the resulting models are ensembles—collections of hundreds to thousands of individual trees working together. This inherent complexity poses significant challenges for practical interpretability.

In contrast, single decision trees such as Classification and Regression Trees (CART) [10] are inherently interpretable, presenting decision logic as a clear hierarchical structure where each prediction follows a traceable path of simple comparisons. This interpretability makes them attractive for safety-critical applications where understanding *why* a decision was made is as important as the decision itself. However, interpretable models face a fundamental constraint: they must remain simple enough for human comprehension, which traditionally limits their capacity to handle complex continuous-control tasks. Existing interpretable RL methods typically distill compact trees from pre-trained neural networks or employ other indirect approaches. Koirala's supervised regression framework, however, opens a different path: training decision trees directly from offline data using the same inputs (state, timestep,

return-to-go) that proved successful for ensembles. This direct training approach in Koirala's specific setup has not been systematically explored for interpretable offline RL policies.

This paper directly addresses this gap and the inherent interpretability-performance trade-off. Our main research question is: *What are the interpretability-performance trade-offs when using CART trees for offline reinforcement learning compared to GBDT ensembles?* To answer this, we explore how interpretability constraints affect performance by systematically varying the structure and complexity of CART-based policies, including both unified trees and decomposed forms that emphasize local decision logic. We also investigate how performance varies with the RTG value used during evaluation, as prior work indicates its significant influence on policy behavior, robustness, and interpretability in return-conditioned settings.

Our contributions include: (1) comprehensive empirical comparison of single trees versus ensembles under identical conditions; (2) systematic analysis of how tree complexity affects performance in continuous control tasks; and (3) investigation of return-to-go conditioning effects on both performance and interpretability. Taken together, these insights offer practical guidance for navigating the trade-off between interpretability and performance in offline RL systems.

The remainder of this paper is organized as follows: Section 2 reviews relevant literature and theoretical context. Section 3 formalizes the supervised regression approach and introduces our model families. Section 4 details the experimental setup, including evaluation protocols and our custom experimentation pipeline. Section 5 presents empirical findings. Section 6 discusses reproducibility and ethical considerations. Section 7 interprets the results, and Section 8 concludes with final insights and future directions.

2 Background and Related Work

Offline reinforcement learning (RL) trains decision-making policies solely from pre-collected data without further environment interaction [1, 2], proving beneficial in scenarios where direct exploration is hazardous, expensive, or impractical (e.g., healthcare, autonomous driving [11, 12]). Despite their effectiveness, standard offline RL methods, particularly those employing deep neural networks and large ensembles, often suffer from *opacity* due to their difficult-to-interpret decision-making, rendering them unsuitable for safety-critical or regulated environments [13, 5].

2.1 Levels and Definitions of Interpretability

Interpretability in machine learning (ML) refers to the degree to which humans can understand a model's prediction rationale [14, 15]. Doshi-Velez and Kim [15] differentiate between intrinsic interpretability (inherently transparent models) and post-hoc interpretability (explanations generated after training). Recent surveys highlight two complementary dimensions: *point-explainability* (explaining a specific action) and *global-understandability* (understanding overall model rules) [13, 16]. Achieving interpretability, especially globally, remains challenging in complex continuous-control tasks due to the sophisticated decision boundaries required to perform effectively [17, 18].

2.2 Decision Trees as an Interpretable Policy Class

Decision trees, especially Classification and Regression Trees (CART), are widely regarded as a cornerstone of interpretable machine learning due to their transparent decision-making process [10]. Unlike opaque neural networks or ensemble methods, decision trees construct explicit hierarchical pathways where predictions follow a traceable sequence of conditional statements. This fundamental transparency has made them attractive for applications requiring algorithmic accountability and human oversight.

While inherently transparent, practical interpretability exists on a spectrum: shallow trees (depth $\leq$5, $\leq$32 leaves) are cognitively tractable and verifiable by humans [5, 6]. This has led to approaches like VIPER [19] and DTPO [20] that distill compact trees from neural networks for interpretable RL.

However, this interpretability incurs a steep cost in representational capacity, as trees constrained to human-auditable sizes (depth < 10) often underfit complex continuous control tasks [21, 22]. As trees grow deeper, their decision logic, while still visible (transparent), quickly becomes too complex for human auditing (losing practical interpretability).

This interpretability-performance tension remains underexplored for axis-aligned decision trees directly trained as policies in RL. While very large trees (e.g., depth > 20) lose human auditability, their structural transparency (deterministic paths from observation to output) fundamentally differs from neural network opacity, motivating our systematic exploration of their performance at scale.

2.3 Offline RL as Supervised Regression

Framing offline RL as supervised regression simplifies sequential decision-making into a familiar supervised learning framework [8, 23]. Koirala and Fleming showed that Gradient-Boosted Decision Tree (GBDT) ensembles like XGBoost (collections of many shallow trees) achieve competitive performance and efficient training [8, 24]. However, these ensembles, often comprising $\sim$1000 trees, severely limit global interpretability [18].

2.4 Challenges with RTG Conditioning

Return-to-Go (RTG) conditioned models, like Decision Transformer (DT), explicitly condition actions on a scalar return target (the sum of future rewards the agent aims to achieve), reshaping RL as conditional sequence modeling [23, 25]. While powerful, RTG conditioning introduces significant sensitivity and opacity. Small variations in the initial RTG value can significantly impact the resulting policy behavior, raising questions about reliability and robustness [26, 27]. Brantley et al. [26] specifically noted that return-conditioned policies require careful tuning and near-deterministic environmental assumptions, which can be difficult to achieve in practice.

2.5 Actuator-Level Interpretability in Robotics[1]

Beyond global interpretability, robotics often requires transparency at the individual actuator or joint level [28, 11]. Modular approaches, decomposing the overall control policy into individual actuator policies, enable engineers to reason more straightforwardly about local actions and simplify diagnostics. Our Multi-output CART (M-CART) approach extends

[1] An actuator is a component of a machine that is responsible for moving or controlling a mechanism or system, for example, a robot's joint or a motor that controls a specific movement.

this philosophy by training a separate decision tree for each actuator dimension, providing granular, actuator-specific interpretability at the expense of potentially reduced global coordination [28]. Evaluating this trade-off offers insights into practical interpretability for complex continuous-control scenarios.

2.6 Knowledge Gap and Contribution

No prior work systematically evaluates axis-aligned CART trees trained from scratch as interpretable offline RL policies in continuous-control settings. Existing interpretable RL approaches rely heavily on distillation from complex neural teacher networks, oblique splits, or hybrid policy forms, and generally lack thorough analyses of depth, leaf count, pruning, and RTG sensitivity for simpler regression-based policy classes.

To address this gap and the our main research question, *What are the interpretability-performance trade-offs when using CART trees for offline reinforcement learning compared to GBDT ensembles?*, our research aims to answer the following sub-questions: SQ1: How do CART policies perform, in absolute and relative terms, when benchmarked against GBDT ensembles on standard offline RL tasks? SQ2: How do different aspects of tree complexity affect both policy performance and human-level interpretability? SQ3: How does the return-to-go (RTG) conditioning mechanism influence the robustness and real-world auditability of these tree-based policies?

3 Methodology

To systematically address the sub-questions posed in Subsection 2.6, this section details our approach. We first describe the adopted regression-based offline RL framework, then introduce the policy model classes evaluated, and finally outline their learning objectives and how their performance and interpretability are assessed.

3.1 Offline RL as Supervised Regression

We adopt the offline reinforcement learning (RL) formulation originally proposed by Koirala and Fleming [8] (subsequently updated in [9]). This approach reframes policy learning as supervised regression, avoiding environment interaction and complex dynamic programming, this method treats offline RL as a straightforward supervised task: predicting the action given the current **state**, the desired **return-to-go (RTG)**, and the **timestep**. This framing bypasses the need for bootstrapping targets, policy gradients, or Q-value estimation, significantly simplifying the training process.

Inputs and Outputs

Each training data point is constructed from pre-collected trajectories. For each timestep t in a trajectory, the model receives a set of input features:

- **State** (s_t): Represents the continuous observation of the environment at timestep t.
- **Normalized Return-to-Go** ($\hat{R}_t$): This scalar value represents the sum of future rewards the agent aims to achieve from timestep t until the end of the episode. It is derived from the unnormalized return, $R_t = \sum_{k=t}^{T} r_k$, where r_k is the reward at timestep k and

T is the final timestep of the trajectory. This R_t is then normalized using environment-specific reference values ($R_{\min}$ and $R_{\max}$) from the dataset, ensuring consistency across tasks:

$$\hat{R}_t = \frac{R_t - R_{\min}}{R_{\max} - R_{\min}}$$

- **Timestep** (t): The current discrete time index within the episode.

The **target output** is the action a_t observed in the dataset at timestep t.

Training Objective

The objective is to train a policy model π mapping $(s_t, \hat{R}_t, t)$ to the corresponding action a_t. This is achieved by minimizing the **mean squared error (MSE)** between predicted actions $\hat{a}_t$, and true recorded actions a_t over the offline dataset:

$$\theta^* = \arg\min_{\theta} \sum_{t} \left\| a_t - \pi(s_t, \hat{R}_t, t; \theta) \right\|^2$$

Here, θ represents the parameters of the policy model π. In Koirala and Fleming's original work, π is instantiated as a gradient-boosted ensemble, specifically XGBoost.[8]

Simulation and Evaluation

Once trained, the policy model π functions as the decision-making component during environmental simulation. Evaluation involves deterministic roll-outs, where the policy interacts with the environment to assess its ability to achieve high returns across full episodes.

The simulation of the trained policy begins with an initial state (s_0) observed from the environment and a predefined target return ($\hat{R}_0$) at timestep $t = 0$. At each subsequent timestep, the model predicts an action (a_t) by taking the current state (s_t), the dynamically evolving return-to-go ($\hat{R}_t$), and the current timestep (t) as its inputs. This predicted action is applied to the environment, which provides a new state (s_{t+1}) and a reward (r_t). Critically, the return-to-go for the next step is updated by decrementing it with the received reward, ensuring that the policy's target return aligns with the actual rewards accumulated. This iterative process of observation, action prediction, environment stepping, and RTG update continues until the episode concludes, either by reaching a terminal state or a maximum episode length.

This evaluation scheme distinguishes itself from traditional RL, where policies are typically purely Markovian (dependent only on the current state). Here, the policy additionally conditions on a dynamically evolving RTG target, rendering the observed behavior sequence-dependent. This formulation enables faster training, but introduces new dynamics and potential sensitivities during policy evaluation. The primary performance metric is the mean normalized return across multiple evaluation episodes.

3.2 Policy Model Classes and Experimental Design

To address our research questions on the interpretability-performance trade-off in offline RL, we systematically evaluate three policy model classes within the supervised regression framework. This allows direct comparison of inherently interpretable models against a high-performing, opaque benchmark, and investigation of interpretability constraints.

3.2.1 Performance Benchmark: Gradient-Boosted Decision Trees (XGBoost)

As a high-performance baseline (for **SQ1**), we utilize **XGBoost** [24]. GBDTs are ensemble methods that sequentially build many shallow decision trees. While effective, their ensemble nature, typically comprising hundreds to thousands of trees, renders them globally opaque. To ensure a fair and reproducible comparison, we reproduced the XGBoost baseline results using the official XGBoost library, as original authors' code was unavailable.

3.2.2 Interpretable Policy Variants (CART and M-CART)

To investigate the interpretability-performance trade-off and the impact of tree complexity (**SQ2**), we employ two inherently interpretable decision tree architectures:

Single Decision Tree (CART) The **Single Decision Tree (CART)** serves as our main interpretable policy. Implemented using the `scikit-learn` library for Python [29], a CART model partitions the input space through hierarchical, axis-aligned splits. This yields transparent decision logic where each prediction follows a traceable path to a leaf [10]. This transparency, especially for compact trees, makes them practically interpretable. We systematically vary parameters like tree depth and maximum leaves to explore how these constraints affect performance and auditability.

Multi-output CART (M-CART) To further enhance interpretability, particularly in high-dimensional action spaces, we introduce **Multi-output CART (M-CART)**. This variant trains a separate decision tree for each individual action dimension. For an N-dimensional action space, this results in N independent CART trees, each predicting one component. This decomposition is motivated by the hypothesis that specialized, smaller trees per actuator can yield more localized and practically auditable logic, even if the overall policy is complex. This design promotes localized interpretability at the actuator level, facilitating granular understanding and diagnostics.

3.2.3 Assessing RTG Conditioning Robustness

To address the influence of RTG conditioning on policy behavior and practical interpretability (**SQ3**), we perform comprehensive RTG sweeps during policy evaluation. Policies are tested by prompting them with a wide range of RTG values. This allows us to assess policy robustness to varying return targets, identify optimal conditioning strategies, and understand how the structural transparency of decision trees interacts with the dynamic nature of RTG conditioning, vital for real-world auditability in goal-conditioned RL.

This systematic experimental design enables rigorous evaluation of the interpretability-performance trade-off across different tree complexities and RTG conditions, providing insights for deploying interpretable offline RL systems. Our implementation and experimental pipeline are publicly available[2] to ensure reproducibility.

4 Experimental Setup

This section details the experimental setup for evaluating decision tree policies in offline reinforcement learning (RL). We describe the benchmark environments, the hyperparameter

[2] `https://github.com/rae-ra/Decision-Trees-vs.-Ensembles-in-Regression-Based-Offline-RL`

search space for our models, and the evaluation protocol used to assess performance and interpretability.

All models—CART, M-CART, and the XGBoost baseline—are tested on nine continuous-control tasks from the D4RL benchmark suite [2]. These tasks span three environments (HalfCheetah, Hopper, Walker2d) and three dataset types: **Medium** (partially trained), **Medium-Replay** (transitions logged during training), and **Medium-Expert** (mixed medium and expert trajectories).

4.1 Model Families and Search Space

Our experiments use the following hyperparameter search design for the policy models.

XGBoost hyperparameters. We replicate the ensemble baseline of Koirala and Fleming [9], using 1,000 gradient-boosted trees with depth 6 and learning rate 0.1.

Shared hyperparameter grid. For the interpretable models (CART and M-CART), we sweep over the following hyperparameters: **Tree depth** ($\{2, 3, \ldots, 10, 15, 20, 25\}$), **Max leaves** ($\{8, 32, 64, 128, 1024, \text{unlimited}\}$), and `min_samples_leaf` ($\{1, 5, 10, 50\}$). We vary `min_samples_leaf` to encourage structural diversity, but we do not report its independent effects, as our analysis is structured around broader capacity tiers.

Interpretability tiers. We categorize model complexity using four tiers:

Small Depth ≤ 3 or ≤ 8 leaves (fully human-auditable)

Medium Depth ≤ 5 or ≤ 32 leaves (cognitively tractable)

Large Depth ≤ 20 and ≤ 1024 leaves (transparent but hard to audit)

Unbounded Depth up to 30, no leaf constraint (maximal capacity)

Model-specific structure and tier coverage. CART models are trained as single multi-output regressors and evaluated across the entire interpretability spectrum, from Small to Unbounded tiers.

M-CART models, by contrast, predict each action dimension with a separate tree (e.g., 3 trees for Hopper and 6 for HalfCheetah and Walker2d). To manage the resulting growth in total model size, M-CART evaluation is restricted to the Small, Medium, and Large tiers only (depth ≤ 20, leaves ≤ 1024). This decomposition enables more localized decision logic while preserving comparability within shared capacity limits.

4.2 Evaluation Protocol

Each model is trained on the full offline dataset. Evaluation uses deterministic rollouts with five fixed random seeds (0–4) and ten episodes per seed. For each seed, we sweep over RTG prompts $\{0.10, 0.15, \ldots, 1.00, \hat{R}_{\max}\}$—where $\hat{R}$max denotes the normalized return corresponding to the maximum trajectory return in the dataset—and report the highest mean normalized return:

$$\text{Score} = \frac{R_{\text{achieved}} - R_{\text{ref,min}}}{R_{\text{ref,max}} - R_{\text{ref,min}}} \times 100$$

5 Results

This section presents the empirical evaluation of interpretable decision tree policies (CART and M-CART) against an XGBoost ensemble baseline across nine D4RL continuous-control environments. We examine the impact of interpretability constraints on policy performance and analyze key behavioral characteristics.

Training and evaluation followed the standardized protocol (Section 4.2), including RTG sweeps, fixed-seed rollouts, and performance normalization. Model hyperparameters are detailed in Section 4.1. Unless otherwise stated, we report mean normalized return over five seeds, with error bars indicating standard deviation.

The section is structured as follows: §5.1 validates our XGBoost replication; §5.2 compares overall model family performance; §5.3 examines capacity-dependent trade-offs; §5.4 analyzes RTG sensitivity; and §5.5 discusses tree complexity and generalization, including M-CART's depth scaling (§5.5.2).

5.1 XGBoost Baseline Verification

Table 1 compares our reproduced XGBoost scores with the original results reported by Koirala and Fleming [9]. Our reproduction aligns within a few normalized return points in most environments, confirming our evaluation pipeline's validity. However, discrepancies occur, notably in *hopper-medium-replay* and *walker2d-medium-expert* (deviations of 8 and 24 points, respectively). These gaps likely stem from implementation details or evaluation subtleties not fully specified in the original paper. Nonetheless, our reproduction captures the general performance profile and provides a stable reference point for analyzing interpretability–performance trade-offs

Table 1: Normalized return comparison between our XGBoost reproduction (**Reproduced**), with `Std. Dev.` the standard deviation, and results from Koirala and Fleming [8] (**Paper**). Std. Dev. While most results are closely aligned, larger deviations appear (e.g. *walker2d-medium-expert*).

Environment	Paper	Reproduced	Std. Dev.
halfcheetah-medium	43.19	**43.7**	0.80
halfcheetah-medium-replay	40.91	**41.5**	0.74
halfcheetah-medium-expert	**90.34**	87.2	5.24
hopper-medium	**72.91**	66.0	5.70
hopper-medium-replay	**91.66**	83.3	5.46
hopper-medium-expert	109.85	**112.3**	2.74
walker2d-medium	82.73	**85.6**	1.89
walker2d-medium-replay	**87.86**	86.2	11.50
walker2d-medium-expert	**108.96**	85.1	4.21

5.2 Model Family Performance Comparison

Table 2 summarizes the best-performing configurations for CART, M-CART, and XGBoost across all nine D4RL tasks, selected by highest average per-seed return (each seed's score reflects its best RTG value). CART's best configurations derive exclusively from the Unbounded tier, while M-CART's optimal performance comes invariably from the Large tier. XGBoost generally achieves the highest returns, though CART performs competitively in several tasks, even outperforming XGBoost in *hopper-medium*. Despite size constraints, M-CART performs moderately well in certain tasks (e.g., *hopper-medium-expert*). These re-

sults benchmark the maximal performance of tree-based policies against GBDT ensembles, as observed in our experiments.

Table 2: Best normalized return per environment and model family. Bold values indicate the highest performance in each row. XGBoost generally achieves the highest performance, but CART and M-CART can be competitive in specific environments (e.g. *hopper-medium*).

Environment	CART	M-CART	XGBoost	Paper
HalfCheetah				
Medium	42.3	36.7	**43.7**	43.19
Medium-Replay	37.2	33.8	**41.5**	40.91
Medium-Expert	47.4	38.2	87.2	**90.34**
Hopper				
Medium	**79.9**	57.5	66.0	72.91
Medium-Replay	67.7	16.4	83.3	**91.66**
Medium-Expert	105.6	80.9	**112.3**	109.85
Walker2d				
Medium	80.0	42.3	**85.6**	82.73
Medium-Replay	32.9	14.7	86.2	**87.86**
Medium-Expert	79.6	22.8	85.1	**108.96**
Average	63.6	38.1	76.8	**80.93**

5.3 Capacity–Performance Trade-offs

We partition all CART and M-CART models into four capacity tiers: Small, Medium, Large, and Unbounded (defined in Section 4.1). Figure 1 and Table 3 summarize that performance generally scales with model capacity. Interpretable models (Small and Medium tiers) recover only 13–14% of XGBoost's performance (average 76.8), while Unbounded CART approaches ensemble-level scores (average 63.6) at the cost of practical auditability. Notably, M-CART (average 38.1) surpasses CART (average 29.3) in the Large tier, suggesting modular decomposition improves efficiency under shared complexity constraints.

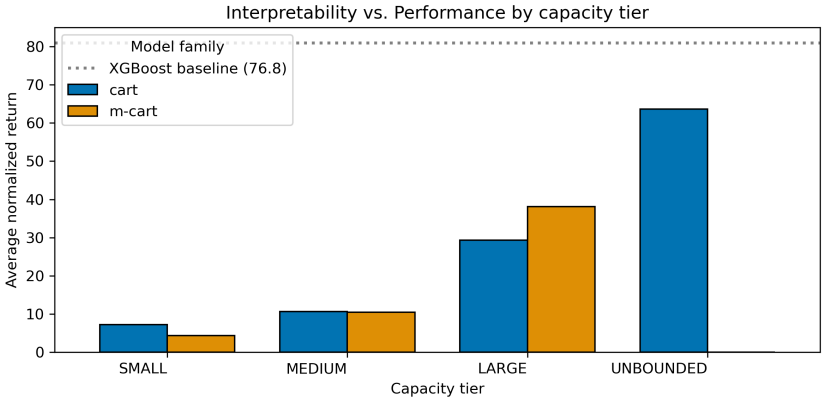


Figure 1: Average normalized return by capacity tier. Performance increases with model complexity. M-CART outperforms monolithic CART in the Large tier, highlighting modular gains.

Table 3: Numerical breakdown of average performance per capacity tier. Only UNBOUNDED CART approaches ensemble-level scores.

Capacity Tier	CART	M-CART	XGBoost Ceiling
SMALL	8.0	4.3	76.8
MEDIUM	10.7	10.5	
LARGE	29.3	38.1	
UNBOUNDED	63.6	—	

5.4 RTG Sensitivity Analysis

We analyze the impact of return-to-go (RTG) input values on the performance of tree-based policies. Let ρ denote the RTG input value. Our evaluation, primarily focusing on an UNBOUNDED CART model (depth 30, unbounded leaves, minimum 5 samples per leaf) on *hopper-medium-expert*, reveals that these policies exhibit high sensitivity to ρ.

Figure 2 and Table 4 vividly illustrate this volatility. For instance, at a fixed prompt $\rho = 0.85$, the model's return ranges from 13.59 (seeds 0–1) to **97.78** (seed 3)—an 84-point spread. At $\rho = 0.90$, seed 4 peaks at **113.52**, yet drops to 58.55 when ρ is reset to 0.85. The model remains unchanged in all cases; only the RTG input differs.

This observed sensitivity manifests in two key forms: (1) distinct optimal RTG values across different training seeds, and (2) abrupt performance swings from small RTG shifts at test time. Such fragility highlights how return conditioning can significantly dominate policy behavior. While XGBoost shows similar sharp RTG-local optima and regular trends, CART models exhibit erratic fluctuations across both seeds and prompts, complicating practical interpretability despite their transparent structure.

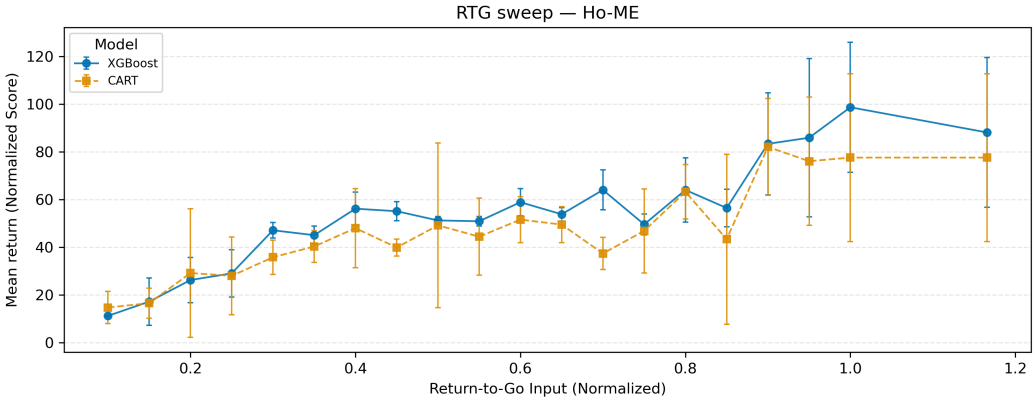


Figure 2: *hopper-medium-expert*: RTG sweep for CART and XGBoost. The plots display how performance varies across RTG prompts (x-axis) and training seeds standard deviation (error bars). The CART tree exhibits abrupt and highly erratic fluctuations in performance, which complicates its practical interpretability despite its transparent structure.

5.5 Tree Complexity and Generalization Patterns

We now examine how model structure—particularly tree depth—impacts performance across environments, focusing on both monolithic (CART) and modular (M-CART) trees.

Table 4: Numerical Evidence of RTG Fragility for CART on *hopper-medium-expert*. Each entry shows the normalized return for a fixed RTG prompt ρ. Seed 4 drops from **113.52** at $\rho = 0.90$ to 58.55 at $\rho = 0.85$, despite no change to the model itself

Seed	$\rho = 0.85$	$\rho = 0.90$	$\rho = 0.95$	$\rho = 1.00$
0	13.59	73.68	79.15	**110.28**
1	13.59	73.68	79.15	**110.28**
2	33.18	60.35	**89.30**	60.57
3	**97.78**	88.93	30.96	27.48
4	58.55	**113.52**	101.62	79.14

5.5.1 Depth Scaling in CART

Figure 3 illustrates the distinct generalization behaviors of CART models across different environments with unbounded leaf settings. These plots display the best-per-seed mean return, revealing how model complexity impacts performance and potential overfitting.

The analysis of these performance curves reveals several key insights. For instance, ***Hopper-Medium-Expert*** (Figure 3a) shows continuous performance improvement, exceeding 105% return, suggesting no saturation or overfitting within tested depths (20-30), though such depths are practically uninterpretable. In contrast, ***Walker2d-Medium*** (Figure 3b) clearly shows overfitting, with performance declining and error bars widening at depth 30. ***HalfCheetah-Medium*** (Figure 3c) exhibits a saturation plateau around 42 return from depth 15, indicating no further gains with increased complexity. Finally, ***Hopper-Medium*** (Figure 3d) presents mixed behavior, peaking around depth 20 before becoming unstable. These results illustrate that environments respond differently to increased model complexity, and that strong performance often requires depths well beyond typical interpretability limits (e.g. depth > 10).

5.5.2 M-CART Performance Across Depth

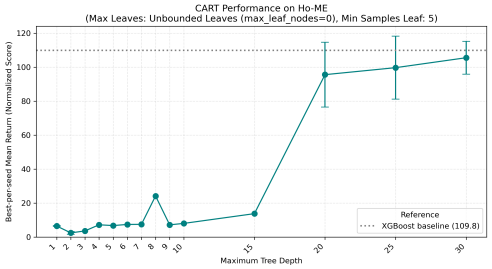
This section examines how decomposed M-CART policies scale with depth under shared capacity constraints. Figure 4 shows performance curves for two environments, using models capped at 1024 leaves and depth ≤ 20 (within the LARGE tier).

Performance improves gradually up to depth 10, then it rises sharply—indicating that even decomposed policies benefit significantly from added capacity. In *hopper-medium-expert*, M-CART reaches over 80 normalized return points ($\sim$72% of XGBoost score), highlighting the potential of modular decomposition to recover strong performance while retaining localized interpretability.

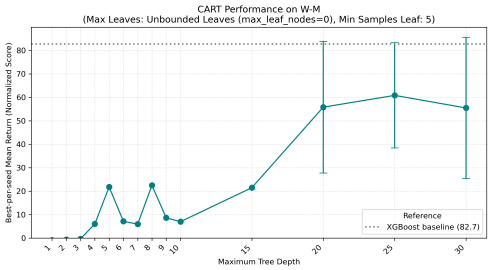
6 Responsible Research

This work supports safer reinforcement learning by investigating interpretable offline policies—an essential concern in high-stakes domains [15, 5]. While black-box models may perform well, their opacity hinders auditability and trust. Our analysis of single-tree and modular variants highlights key trade-offs between performance and transparency.

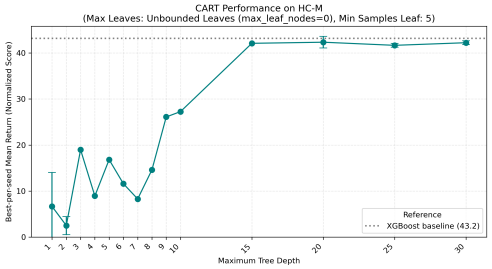
We ensure reproducibility through fixed seeds, deterministic rollouts, exhaustive hyperparameter and RTG sweeps, and reporting of per-seed and per-prompt results. To support replication, all training, configuration, and evaluation code will be publicly released.



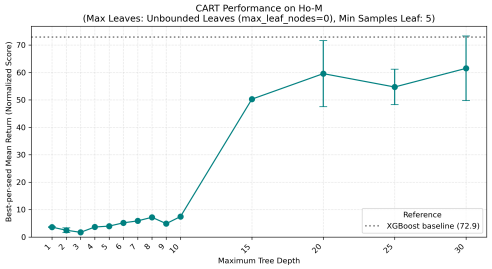
(a) *Hopper-Medium-Expert*: Continuous improvement



(b) *Walker2d-Medium*: Overfitting

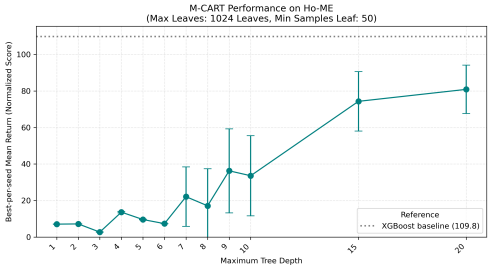


(c) *HalfCheetah-Medium*: Saturation plateau

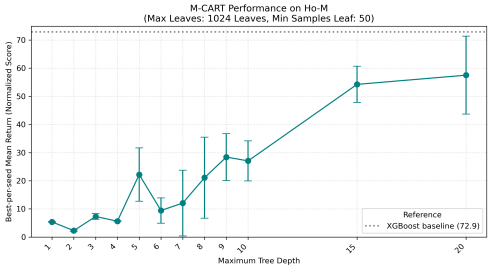


(d) *Hopper-Medium*: Mixed behavior

Figure 3: CART Performance across Environments (Unbounded Leaves). Error bars: Standard deviation. These plots show how increasing maximum tree depth impacts the best-per-seed mean return for CART models across four distinct environments. Responses to increasing model complexity vary by environment, highlighting differences in generalization behavior. Competitive performance necessitates depths beyond typical interpretability limits.



(a) *Hopper-Medium-Expert*



(b) *Hopper-Medium*

Figure 4: M-CART depth scaling. **Error bars:** Standard deviation. Both configurations operate with ≤ 1024 leaves. Performance increases with depth, approaching ensemble scores in *Ho-ME*.

This work is not deployed in real-world safety-critical systems. However, our findings show that even structurally transparent models can exhibit erratic behavior under small RTG changes—a fragility that undermines practical interpretability. If such models were misapplied in deployment, this sensitivity could lead to inconsistent or unsafe outcomes. We thus emphasize that transparency alone does not guarantee robustness. Further evaluation in stochastic or human-interactive settings remains essential.

7 Discussion

This section interprets our empirical findings, addressing the research sub-questions outlined in Section 2.6 (SQ1–SQ3). We discuss the observed interpretability-performance trade-offs and the impact of tree complexity, followed by an analysis of RTG conditioning effects, before outlining the study's limitations and broader implications.

Interpretability-Performance Trade-offs and the Impact of Tree Complexity. Our findings demonstrate a fundamental interpretability-performance trade-off for tree-based policies (SQ1). Compact decision trees (SMALL and MEDIUM tiers) with constrained depth and leaf counts, while cognitively interpretable, consistently yield low normalized D4RL scores. Conversely, larger UNBOUNDED CART configurations approach or match XGBoost performance, showcasing their capacity to learn complex policies. However, this high performance often necessitates depths (e.g., > 10) and leaf counts that exceed practical human auditability [5, 6], effectively transforming them into black-box models from a cognitive standpoint.

The performance profiles across different environments further illuminate these trade-offs (SQ2). In tasks like *hopper-medium*, CART demonstrates competitive performance, even surpassing our XGBoost reproduction. This suggests that simpler environment dynamics or policy requirements may be more amenable to interpretable single-tree solutions. Conversely, environments such as *halfcheetah-medium-expert* or replay variants present more complex optimal policies or diverse, sometimes suboptimal, data distributions [2]. Learning effective policies from such data appears to require greater model capacity, explaining why CART and M-CART struggle significantly compared to XGBoost. While M-CART's overall performance is generally lower than monolithic CART, its relative strength in the LARGE tier, particularly in environments like *hopper-medium-expert*, hints at the potential of modular decomposition for achieving strong results under strict complexity budgets [28].

Behavioral Fragility and RTG Sensitivity. Another central finding concerns the behavioral fragility of return-conditioned tree policies (SQ3). Despite their white-box structural transparency, CART policies exhibit strong sensitivity to the RTG prompt (ρ). As observed in tasks like *hopper-medium-expert*, small shifts in ρ at test time can yield erratic performance changes across different seeds. This phenomenon poses a significant challenge to practical interpretability, likely caused by the policy's tendency to learn highly specialized responses to specific RTG values present in the offline training data [26].

Return conditioning fundamentally alters how these models are understood. In classical decision trees, the same state deterministically maps to the same action. Here, the same state can yield different actions depending on ρ. Since RTG evolves dynamically during test-time execution ($\rho_{t+1} = \rho_t - r_t$), an auditor must reconstruct the full reward trajectory to explain a given decision [27]. This introduces a latent, time-evolving control signal into

an otherwise transparent model. In essence, a model may be white-box in structure yet black-box in its runtime behavior.

Consequently, structural simplicity alone is insufficient for interpretability [16]. Even shallow or sparse trees can become opaque if behavior is significantly influenced by ρ. Interpretability must account not only for model structure, but also for the hidden influence of time-varying inputs.

Limitations and Threats to Validity. Our study's findings are subject to several limitations. First, the evaluation was conducted on nine D4RL continuous-control environments [2], which may not represent the full spectrum of RL tasks. Second, the offline nature means our policies are not subjected to real-time interaction. Third, while we performed comprehensive hyperparameter sweeps within predefined tiers, the search space for decision trees is vast [10], and other configurations could yield different trade-offs. Finally, our interpretability analysis primarily focuses on structural transparency; other facets such as feature importance warrant further investigation.

In summary, this work systematically demonstrated the complex trade-off between interpretability and performance for tree-based policies in offline return-conditioned RL (SQ1–SQ3). While structural simplicity often entails performance costs, the observed RTG sensitivity crucially undermines the practical auditability of even transparent models. This highlights a critical area for future research in responsible and trustworthy sequential decision-making.

8 Conclusions and Future Work

This thesis investigated the viability and behavioral characteristics of interpretable decision tree policies in offline reinforcement learning. Our findings reveal inherent trade-offs between model transparency and performance: small, cognitively interpretable decision trees yield modest performance, while achieving high returns often necessitates models of such complexity that their practical auditability is severely diminished, effectively transforming them into black boxes from a human comprehension standpoint [5, 6]. Crucially, we identified return-to-go (RTG) sensitivity as a significant factor undermining these policies' practical interpretability, where despite a tree's transparent structure, its behavior becomes opaque when critical decisions are heavily influenced by dynamically evolving, unobserved RTG input, leading to unpredictable performance shifts [26].

These findings contribute to the growing understanding of interpretable machine learning in sequential decision-making. We empirically demonstrated that directly applying traditional interpretable models in return-conditioned RL faces significant hurdles—not only in bridging the performance gap with ensemble methods like XGBoost [24], but also in maintaining true behavioral transparency under dynamic input conditioning. Our work highlights that structural interpretability alone is insufficient for guaranteeing auditability in complex, state-dependent, and goal-conditioned systems.

Looking forward, addressing RTG fragility is a critical direction for developing more trustworthy interpretable RL policies. One promising avenue is to replace explicit RTG prompts with more semantically meaningful, interpretable conditioning signals, such as hierarchical subgoal conditioning. Approaches similar to Hierarchical Decision Transformers [25] could guide policies through learned, interpretable waypoints rather than relying on raw return values difficult to set, interpret, and maintain consistently. Furthermore, future research should explore the generalizability of these findings across a wider array of

reinforcement learning domains, including continuous-control environments, robotic simulation tasks, and diverse offline dataset distributions beyond D4RL [2]. Investigation into alternative interpretable model architectures, or the integration of complementary explainability techniques, could also yield valuable insights into mitigating the observed performance–interpretability trade-off. Ultimately, the goal is to develop robust, high-performing, and auditable RL agents suitable for real-world high-stakes applications where trust and traceability are paramount.

References

[1] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[2] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[3] Dmitry Kalashnikov, Alexander Irpan, Julian Ibarz, Sergey Pastor, Julian Young, Fangchen Liu, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6226–6233, 2018.

[4] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Szymon Debiak, Piotr Włodarczyk, Wojciech Zaremba, and Ilya Sutskever. Learning dexterous in-hand manipulation. In *International Conference on Robotics and Automation*, 2018.

[5] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[6] Christoph Molnar. *Interpretable Machine Learning*. Christoph Molnar, 2019. Available at `https://christophm.github.io/interpretable-ml-book/`.

[7] Ankit Gohel, Akshit Gohel, Hardik Kothari, and Shivam Shah. Explainable artificial intelligence (xai): A comprehensive review of concepts, methods, and applications. *Machine Learning with Applications*, 12:100473, 2023.

[8] Prajwal Koirala and Cody Fleming. Reframing offline reinforcement learning as a regression problem. *arXiv preprint arXiv:2401.11630*, pages 1–11, January 2024. arXiv:2401.11630v1 [cs.LG] — Iowa State University, 21 Jan 2024.

[9] Prajwal Koirala and Cody Fleming. Solving offline reinforcement learning with decision tree regression. *arXiv preprint arXiv:2401.11630*, October 2024. arXiv:2401.11630v2 [cs.LG] — 14 Oct 2024.

[10] Leo Breiman, Jerome Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1st edition, 1984.

[11] Ergys Likmeta, Mykel J. Kochenderfer, and Jonathan Ballson. Combining reinforcement learning with rule-based controllers for transparent decision-making in autonomous driving. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3812–3818, 2020.

[12] David Waibel, Ben Scheuermann, Henrik Bieg, Frank Althoff, and Marc Toussaint. Causal explanations for robot failures. *arXiv preprint arXiv:2309.11718*, 2023.

[13] Tom Glanois, Timothée Lesort, Alessandro Gritti, Benjamin Effenberger, Hervé Mounier, and Arno Starke. A survey on interpretable reinforcement learning. *arXiv preprint arXiv:2401.07767*, 2024.

[14] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[15] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[16] Farnood Poursabzi-Sangdeh, Dan G. Goldstein, Jake M. Hofman, Jennifer Wortman Vaughan, and Hanna Wallach. Manipulating and measuring model interpretability. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[17] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2280–2288, 2016.

[18] Ellis Brown, Akshat Singh, Yuan Ma, Abhinav Verma, and Scott Niekum. Scaling interpretable reinforcement learning via decision tree policies. *arXiv preprint arXiv:2106.01429*, 2021.

[19] Osbert Bastani, Alexandros Ioannou, Lamya Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Verifiable reinforcement learning via policy extraction. In *Advances in Neural Information Processing Systems*, pages 2494–2504, 2018.

[20] Daniël Vos and Sicco Verwer. Decision-tree policy optimisation. *arXiv preprint arXiv:2408.11632*, 2024.

[21] Yiming Chen, Fangyu Li, Rohit Singh, and Jing He. Robust greedy model distillation for decision trees in offline rl. In *Advances in Neural Information Processing Systems*, 2024.

[22] Giulia-Helene Pace, Edward Liu, Hanchuan Yang, and Quoc V. Le. Poetree: Interpretable and efficient policy trees via differentiable dendrites. In *International Conference on Learning Representations*, 2022.

[23] Lili Chen, Marcin Andrychowicz, Marcin Raichuk, Bowen Baker, Maciek Chociej, Szymon Debiak, Piotr Włodarczyk, Wojciech Zaremba, and Ilya Sutskever. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, 2021.

[24] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[25] Marcelo Correia and Luís A. Alexandre. Hierarchical decision transformer. *arXiv preprint arXiv:2307.03952*, 2023.

[26] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning? *arXiv preprint arXiv:2204.05618*, 2022.

[27] Anand Thambi, Tom Glanois, Benjamin Effenberger, and Arno Starke. Interpreting decision transformer: Insights from continuous control tasks. *arXiv preprint arXiv:2403.07270*, 2024.

[28] Milad Ghazvininejad, Marc Toussaint, and Sebastian Herzog. Learning modular policies for robotics. *Frontiers in Computational Neuroscience*, 8(62), 2014.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.