BEYOND FEATURE IMPORTANCE: HIERARCHICAL EXPLANATIONS OF FEATURE SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

An emerging family of explainable AI methods aims to identify, for each input, a minimal subset of observed feature values sufficient for predicting the output of a black box. These instance-wise feature-selection methods can reveal which features are important for a prediction. In this work, we go one step further and ask: why were these features selected rather than others? We introduce a recursive framework for hierarchical explanations of feature selection. At the first order, a sufficient-subset explainer produces an instance-wise binary feature-selection mask for explaining the original black-box output. Each subsequent order explains the selection mask produced at the previous order. Repeating this process yields an ordered hierarchy in which each level explains why the features at the previous level were selected. We develop a training procedure for learning these hierarchies. On nonlinear mechanisms generated by decision trees, our method recovers the underlying decision-tree structure substantially more accurately than conventional decision-tree baselines, while maintaining high predictive fidelity at every order. Our framework moves beyond identifying what features are selected toward explaining why they are selected.

1 INTRODUCTION

Feature selection is an important route to interpretable machine learning. By identifying the input variables relevant to an output, it can provide a reason for a prediction, simplify a model, and support scientific discovery. This is increasingly important as black-box models are used in settings such as medicine, finance, and public services, where users must be able to assess whether a model relies on the right information.

Classical feature-selection methods seek one subset of variables that is useful across an entire population, through filter, wrapper, embedded, or regularization-based procedures (Liu & Setiono, 1996; Kohavi & John, 1997; Tibshirani, 1996; Guyon & Elisseeff, 2003; Louppe et al., 2013). This global view is appropriate when the same variables matter for every observation. It is inadequate when the predictive mechanism changes from one input to another.

Instance-wise feature selection (IWFS) addresses this problem. Given an input $\mathbf{x}$, an IWFS method seeks a smallest subset of its observed feature values that is sufficient for predicting the black-box output at $\mathbf{x}$. The selected set may therefore vary across inputs. This idea appears in perturbation-based explanations (Fong & Vedaldi, 2017), local rules (Ribeiro et al., 2018), sufficient input subsets (Carter et al., 2019), L2X (Chen et al., 2018), INVASE (Yoon et al., 2018), REAL-X (Jethani et al., 2021), and the removal-based framework of Covert et al. (2021).

Example 1 (a switching black box). Consider the black box

$$y = \begin{cases} x_2, & x_1 < 0, \\ x_3, & x_1 \geq 0. \end{cases}$$

To produce its output, the function first checks whether $x_1 < 0$. If so, it needs x_2; otherwise, it needs x_3. Thus, for every input with $x_1 < 0$, the necessary and sufficient feature set is $\{X_1, X_2\}$, whereas for every input with $x_1 \geq 0$, it is $\{X_1, X_3\}$. A global selector would retain $\{X_1, X_2, X_3\}$. An ideal instance-wise selector instead returns the smaller set needed for the particular input.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Many IWFS methods learn a selector and a predictor together. The selector chooses features to retain, while the predictor is trained to reproduce the black-box output from the retained values. Usually, unselected features are replaced with a fixed ablation value, such as zero, and the training objective trades predictive fidelity against the number of retained features. This setup can suffer from information leakage: the predictor can infer the selection mask from the pattern of ablated values and use that pattern as an unintended communication channel. Jethani et al. (2021) identified instances of this failure in selector–predictor models. For example, in classification setting where the number of classes is less than the number of input features. Hide&Seek showed that it arises more generally, including in regression settings and the problem is especially serious for *switch variables* (or decision variables), whose role is to determine which local rule applies. (Ellinson et al., 2026).

In Example 1, X_1 is a switch variable. Once the branch has been identified, its exact value is irrelevant; only whether it lies below zero matters. A selector can therefore encode the branch by choosing whether to ablate X_1. The predictor may then predict accurately even though the resulting mask incorrectly omits X_1.

To address this problem, rather than using a recognizable fixed ablation value, Hide&Seek replaces hidden values with plausible random replacements. The predictor can therefore not simply read the ablation pattern as a substitute for the hidden information.

Reliable recovery of decision variables makes a further question possible: why was a particular feature set selected? For Example 1, the first-order instance-wise selection mask is

$$\mathbf{m}_1(\mathbf{x}) = [1, 1, 0] \quad \text{when } x_1 < 0, \qquad \mathbf{m}_1(\mathbf{x}) = [1, 0, 1] \quad \text{when } x_1 \geq 0.$$

The two masks encode $\{X_1, X_2\}$ and $\{X_1, X_3\}$, respectively. The mask is itself an input-dependent output. An IWFS explainer applied to this output identifies X_1, because the value of X_1 determines which first-order mask is selected. The first-order explainer says that X_1 and X_2 are needed for a particular prediction; the second-order explainer says that the value of X_1 is why this first-order explanation was selected.

We develop this observation into *Hierarchical EXplanations* (HEX). HEX repeatedly applies an IWFS explainer to the mask produced by the preceding explainer. The first-order explainer selects features needed for the original black-box output. The second-order explainer selects features needed to predict the first-order mask. The process continues until the mask is constant. Each order therefore explains a selection decision made at the order below it.

The resulting ordered masks form an explanation path for each input. Under suitable conditions, higher-order masks are nested within lower-order masks. By combining paths across inputs, HEX can in principle recover the decision process of the black box from the bottom up. This differs from conventional decision-tree learning, which constructs a global tree from the top down and typically relies on simple axis-aligned splits. In practice, an error in an early mask can affect later orders; our experiments study when the resulting hierarchy is recovered accurately.

HEX is distinct from higher-order feature attribution. Attribution methods such as LIME (Ribeiro et al., 2016), Integrated Gradients (Sundararajan et al., 2017), DeepLIFT (Shrikumar et al., 2017), and SHAP (Lundberg & Lee, 2017) assign numerical contributions to individual features. Their higher-order extensions assign numerical interaction values to pairs, triples, or larger feature groups. For example, Janizek et al. (2021) study axiomatic feature interactions in deep networks; related work develops interaction-attribution methods more broadly (Lundström & Razaviyayn, 2023; Baniecki et al., 2026).

HEX belongs instead to the sufficient-subset family of explanations. Its first-order output is a set of observed feature values that jointly suffice for a prediction, not a vector of numerical contributions. Its second-order output explains why that set was selected. Thus, higher-order attribution asks how features jointly contribute to a prediction, whereas HEX asks what determines the choice of an input-dependent sufficient subset. These are different explanatory objects and answer different questions. HEX provides a recursive way to move from identifying what is important for a prediction toward explaining why it is important.

2 INSTANCE-WISE FEATURE SELECTION AND HIDE&SEEK

This section states the instance-wise feature-selection problem and the Hide&Seek explainer that we use as the inner component of HEX.

2.1 LOCAL SUFFICIENT FEATURE SETS

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the input space, let $[d] = \{1, \dots, d\}$, and let $\mathcal{B}$ denote the black box to be explained. The black box may be probabilistic; write $p_{\mathcal{B}}(Y \mid \mathbf{x})$ for its output distribution at $\mathbf{x}$. For a deterministic black box, this distribution is a point mass at $\mathcal{B}(\mathbf{x})$.

For an input $\mathbf{x}$, instance-wise feature selection seeks a smallest set of observed feature values that suffices for the black-box output:

$$S(\mathbf{x}) \in \arg\min_{\mathcal{A} \subseteq [d]} |\mathcal{A}| \quad \text{such that} \quad p_{\mathcal{B}}(Y \mid \mathbf{X}_{\mathcal{A}} = \mathbf{x}_{\mathcal{A}}) = p_{\mathcal{B}}(Y \mid \mathbf{X} = \mathbf{x}).$$

Thus, the observed values indexed by $S(\mathbf{x})$ suffice to reproduce the black-box output distribution at $\mathbf{x}$, and no smaller set does. For a deterministic black box, this means that, after observing $\mathbf{x}_{S(\mathbf{x})}$, the remaining features cannot change the black-box output except on a probability-zero set. We call the retained indices necessary for the local prediction.

The set $S(\mathbf{x})$ is often represented by a hard binary *selection mask* $\mathbf{m}(\mathbf{x}) := (m_1(\mathbf{x}), \dots, m_d(\mathbf{x}))$ where $m_j(\mathbf{x}) = \mathbb{1}\{j \in S(\mathbf{x})\}$, for all $j \in [d]$.

For the switching black box in Example 1, $S(-3, 5, 1) = \{1, 2\}$ and $S(3, 5, 1) = \{1, 3\}$, represented by the masks $[1, 1, 0]$ and $[1, 0, 1]$, respectively.

2.2 THE IWFS-EXPLAINER INTERFACE

Given black-box training pairs, $\mathcal{D}_{\text{train}} = \big((\mathbf{x}^{(n)}, y^{(n)})\big)_{n=1}^{N}$, where $y^{(n)}$ is obtained by querying the black box at $\mathbf{x}^{(n)}$, an IWFS explainer returns a selection mask, $\mathbf{m}(\mathbf{x})$, and an output prediction, $\widehat{y}(\mathbf{x})$:

$$\mathcal{E}(\mathbf{x}) = \big(\mathbf{m}(\mathbf{x}), \widehat{y}(\mathbf{x})\big).$$

Its mask should be both sparse and sufficient for predicting the black-box output. Accordingly, IWFS training balances a task-appropriate prediction loss—mean-squared error for regression and binary cross-entropy for a binary mask target—against a penalty on the number of selected features.

2.3 HIDE&SEEK

Many selector–predictor methods replace hidden features with a fixed ablation value, often zero. Such a value can make the pattern of replacement informative about the selector's decision: a predictor may exploit that pattern rather than the selected feature values. This is the information-leakage problem discussed in Section 1.

Hide&Seek replaces the fixed ablation operation with a continuous gate and random replacement values. Let, $\mathbf{g}_\theta(\mathbf{x}) \in [0, 1]^d$ denote the soft gate produced by a selector network. Given internally generated replacement values $\widetilde{\mathbf{x}}$, the predictor, $h_\phi(\cdot)$, receives

$$\mathbf{z}_\theta(\mathbf{x}, \widetilde{\mathbf{x}}) = \mathbf{g}_\theta(\mathbf{x}) \odot \mathbf{x} + \big(\mathbf{1} - \mathbf{g}_\theta(\mathbf{x})\big) \odot \widetilde{\mathbf{x}}, \qquad \widehat{y}_{\theta,\phi} = h_\phi(\mathbf{z}_\theta)\,.$$

The selector and predictor are trained jointly with an objective of the form

$$\frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\widetilde{\mathbf{X}} \sim q_{\mathcal{D}}} \left[\ell\Big(y^{(n)}, \widehat{y}_{\theta,\phi}(\mathbf{x}^{(n)}, \widetilde{\mathbf{X}})\Big) + \frac{\lambda}{d} \left\| \mathbf{g}_\theta(\mathbf{x}^{(n)}) \right\|_1 \right],$$

where ℓ is the task loss and λ controls parsimony. In our implementation, each coordinate of $\widetilde{\mathbf{x}}$ is sampled independently from its empirical marginal distribution in the training data:

$$q_{\mathcal{D}}(\widetilde{\mathbf{x}}) = \prod_{j=1}^{d} \mathrm{Unif}\Big(\{x_j^{(1)}, \dots, x_j^{(N)}\}\Big)\,.$$

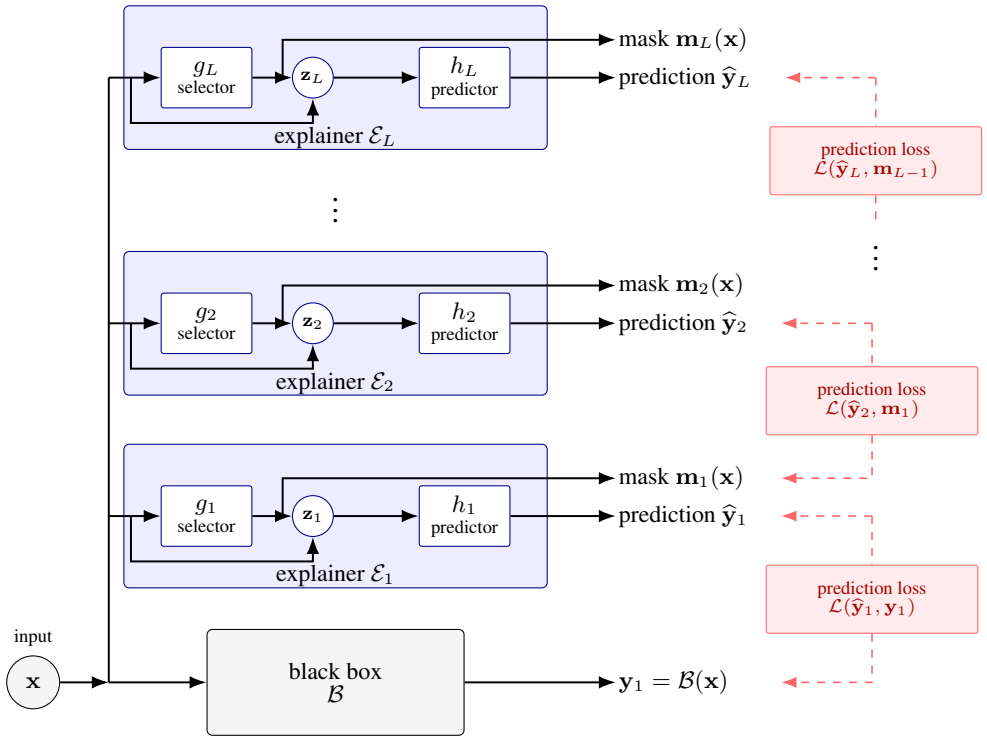


Figure 1: Hierarchical explanation by HEX. A single input $\mathbf{x}$ is provided to the original black box $\mathcal{B}$ and to every explainer. Explainer $\mathcal{E}_i$ produces a binary selection mask $\mathbf{m}_i(\mathbf{x})$ and predicts its order-specific target. The original output $\mathbf{y}_1$ trains $\mathcal{E}_1$; thereafter, the preceding selection mask is the next target: $\mathbf{y}_{i+1} = \mathbf{m}_i(\mathbf{x})$. Thus every higher order explains why the preceding order selected its features. Black arrows denote forward computation; dashed pale-red arrows and boxes denote the prediction losses used to train the explainers.

The motivation for replacement is simple. If the joint distribution of the masked-and-replaced signal were the same as that of the original signal, the predictor could not tell which coordinates had been replaced and hence could not infer the selection mask. A fixed ablation value instead makes replacement directly visible in the input. Conditional replacement would reproduce the exact distribution of unobserved features given the observed ones. Marginal replacement is a simple practical approximation: by removing the recognizable fixed ablation pattern, it is sufficient in practice to prevent the predictor from recovering the mask.

After training, the hard selection mask returned by the explainer is $\mathbf{m}(\mathbf{x}) = \mathbb{1}\{\mathbf{g}_\theta(\mathbf{x}) \geq \tau\}$, where the indicator and inequality are applied componentwise, and $\tau \in (0, 1)$ is the mask threshold. The distinction is deliberate: $\mathbf{g}_\theta$ is an internal soft training gate, whereas $\mathbf{m}$ is the hard mask returned by the explainer.

3 Hierarchical Explanations of Feature Selection (HEX)

An IWFS explainer maps each input $\mathbf{x}$ to a selection mask $\mathbf{m}_1(\mathbf{x})$. It may select $\{X_1, X_2\}$ for one input and $\{X_1, X_3\}$ for another. The first-order mask tells us which feature values were necessary for each prediction. What it does not explain is why the selection map returned different masks.

HEX explains that map. It treats $\mathbf{x} \mapsto \mathbf{m}_1(\mathbf{x})$ as a new black box and fits another instance-wise explainer to it. The resulting second-order mask identifies the feature values needed to determine the first-order mask. Its own selection map can be explained in the same way. Each level therefore explains the selection made by the level below it, rather than re-explaining the original prediction.

Figure 1 shows this process. Every explainer receives the same original input $\mathbf{x}$: it must know the input in order to determine which selection applies. What changes from one order to the next is the

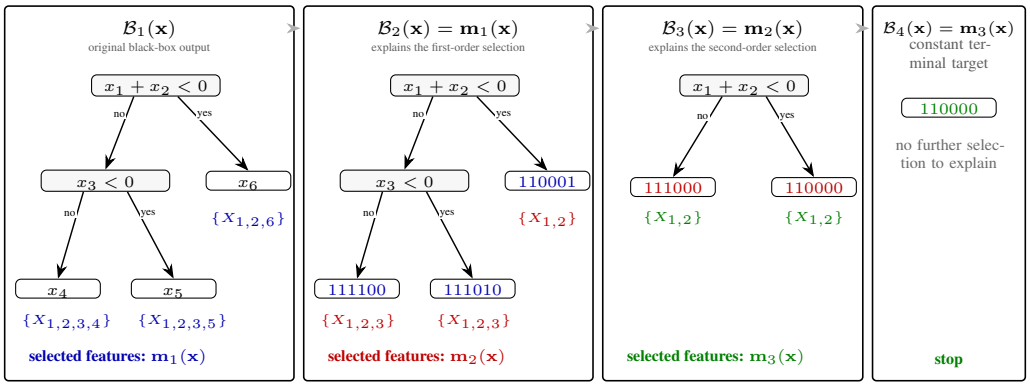


Figure 2: An idealized HEX hierarchy for a decision-tree mechanism. In the first panel, the blue sets are the first-order selections $\mathbf{m}_1(\mathbf{x})$ on the corresponding black-box paths. The leaves of $\mathcal{B}_2$ output those blue masks; the red sets are the features selected to explain which blue mask occurs. The same recursion produces the green third-order masks. The process stops once the next target is constant.

target. The first explainer predicts the original black-box output, $\mathcal{B}_1(\mathbf{x})$. The second predicts the first selection mask. The third predicts the second selection mask. The solid arrows show these forward computations; the pale-red arrows show the corresponding training losses.

More formally, a Hide&Seek explainer $\mathcal{E}_i$ is fitted to the frozen target $\mathcal{B}_i$ and returns a hard selection mask $\mathbf{m}_i(\mathbf{x})$. The following order explains that mask:

$$\mathcal{B}_{i+1}(\mathbf{x}) = \mathbf{m}_i(\mathbf{x}).$$

Thus $\mathcal{E}_1$ explains the original black box, $\mathcal{E}_2$ explains the selection made by $\mathcal{E}_1$, and so on. The first-order loss is chosen for the original target—for example, mean-squared error for regression, binary cross-entropy for binary or multilabel targets, and categorical cross-entropy for a multiclass target. Every higher-order target is a binary mask, so its prediction loss is coordinatewise binary cross-entropy. In every case, Hide&Seek also penalizes mask mass to encourage parsimonious selection. Algorithm 1 gives the HEX recursion, and Algorithm 2 gives its Hide&Seek inner fitter.

For an input $\mathbf{x}$, HEX returns the ordered explanation path

$$\big(S_1(\mathbf{x}), S_2(\mathbf{x}), \ldots, S_L(\mathbf{x})\big), \qquad S_i(\mathbf{x}) = \{j : m_{i,j}(\mathbf{x}) = 1\}.$$

The process stops once a newly produced mask is constant on the probe population, or once a prescribed maximum order is reached. A terminal all-zero mask is not appended to the path.

The decision-tree example in Figure 2 makes the idea concrete. Consider $\mathbf{x}^* = (10, 20, -30, 1.5, -6, 17)$. To obtain the original output, the black box evaluates its root condition using X_1 and X_2, then evaluates the X_3 condition, and finally returns the value of X_5. The first-order mask is therefore $(1, 1, 1, 0, 1, 0)$: it records $\{X_1, X_2, X_3, X_5\}$.

Now ignore the black-box value and ask instead which first-order mask will be selected. For $\mathbf{x}^*$, this requires only the root condition and the X_3 condition, so the second-order mask is $(1, 1, 1, 0, 0, 0)$. One level higher, the root condition alone determines which second-order mask occurs; the third-order mask is $(1, 1, 0, 0, 0, 0)$. In this example the third-order mask is constant, so the recursion stops. The path has separated the variables used to produce a leaf value from the variables that decide the local branch and, finally, from those that decide the broadest branch.

This picture is not merely intuitive. Under three ideal conditions, the explanation path must be nested: every higher-order set is a subset of the set below it. The conditions are that each order has a unique necessary feature set, the inner explainer returns that set exactly, and the next-order target is the preceding hard mask. A necessary feature set is a sufficient set from which no selected feature can be removed. When it is unique, it is also the unique minimum-size sufficient set. The following proposition states the result.

Proposition 3.1 (Nested masks)**.** *Suppose that, for every applicable order i and every input $\mathbf{x}$, $S_i(\mathbf{x})$ is the unique necessary feature set for the order-i target $\mathcal{B}_i(\mathbf{x})$, and that the HEX recursion is*

$$\mathcal{B}_{i+1}(\mathbf{x}) = \mathbf{m}_i(\mathbf{x}).$$

Algorithm 1 HIERARCHICAL EXPLANATIONS (HEX)

Input: black-box training data $\mathcal{D} := \big((\mathbf{x}^{(n)}, y^{(n)})\big)_{n=1}^{N}$, where $\mathbf{x}^{(n)} \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y^{(n)} \in \mathcal{Y}$
Requirements: a binary-mask explainer-fitting procedure FITEXPLAINER; maximum order $L_{\max} \in \mathbb{N}$
Output: a sequence of frozen explainers $(\mathcal{E}_i)_{i=1}^{L}$

$\mathcal{D}_1 \leftarrow \mathcal{D}$
for $i = 1, \dots, L_{\max}$ **do**
  $\mathcal{E}_i \leftarrow \text{FITEXPLAINER}(\mathcal{D}_i)$
  Write $\mathcal{E}_i(\mathbf{x}) = (\mathbf{m}_i(\mathbf{x}), -)$, where $\mathbf{m}_i : \mathcal{X} \to \{0,1\}^d$
  $\mathcal{D}_{i+1} \leftarrow \big((\mathbf{x}^{(n)}, \mathbf{m}_i(\mathbf{x}^{(n)}))\big)_{n=1}^{N}$
  if $\big|\{\mathbf{m}_i(\mathbf{x}^{(n)}) : n = 1, \dots, N\}\big| = 1$ **then**
    return $(\mathcal{E}_k)_{k=1}^{i}$
  end if
end for
return $(\mathcal{E}_k)_{k=1}^{L_{\max}}$

Algorithm 2 FITEXPLAINER with Hide&Seek

Input: $\mathcal{D} := ((\mathbf{x}^{(n)}, y^{(n)}))_{n=1}^{N} \in (\mathcal{X} \times \mathcal{Y})^N$, where $N, d \in \mathbb{N}$, $\mathcal{X} \subseteq \mathbb{R}^d$, and $\mathcal{Y}$ is the output space
Requirements: a task loss $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$; hyperparameter $\lambda > 0$; and threshold $\tau \in (0,1)$
Output: a frozen explainer $\mathcal{E}$

Define neural networks $g_\theta : \mathcal{X} \to [0,1]^d$ and $h_\phi : \mathbb{R}^d \to \mathcal{Y}$
Define $q_{\mathcal{D}}(\tilde{\mathbf{x}}) := \prod_{j=1}^{d} \text{Unif}\left(\left\{x_j^{(1)}, \dots, x_j^{(N)}\right\}\right)$
Define $\mathbf{z}_\theta(\mathbf{x}, \tilde{\mathbf{x}}) := g_\theta(\mathbf{x}) \odot \mathbf{x} + (\mathbf{1} - g_\theta(\mathbf{x})) \odot \tilde{\mathbf{x}}$
Define $\hat{y}_{\theta,\phi}(\mathbf{x}, \tilde{\mathbf{x}}) := h_\phi(\mathbf{z}_\theta(\mathbf{x}, \tilde{\mathbf{x}}))$
$(\theta^\star, \phi^\star) \leftarrow \arg\min_{\theta,\phi} \left\{ \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\mathcal{D}}} \left[\ell\Big(y^{(n)}, \hat{y}_{\theta,\phi}(\mathbf{x}^{(n)}, \tilde{\mathbf{x}})\Big) \right] + \frac{\lambda}{Nd} \sum_{n=1}^{N} \Big\| g_\theta(\mathbf{x}^{(n)}) \Big\|_1 \right\}$
Define $\mathbf{m}(\mathbf{x}) := \mathbb{1}\{g_{\theta^\star}(\mathbf{x}) \geq \tau\}$ componentwise
Define $\mathcal{E}(\mathbf{x}) := \big(\mathbf{m}(\mathbf{x}), \hat{y}_{\theta^\star,\phi^\star}(\mathbf{x}, \tilde{\mathbf{x}})\big)$, where $\tilde{\mathbf{x}} \sim q_{\mathcal{D}}$ is generated internally
return frozen $\mathcal{E}$

Then, for every applicable i and $\mathbf{x}$*,*

$$S_{i+1}(\mathbf{x}) \subseteq S_i(\mathbf{x}).$$

The proof has the following interpretation. Fix an input and suppose that its order-i necessary feature set is S. Knowing the observed values indexed by S fixes the order-i output. Because this set is unique, it also fixes which order-i mask is selected: every input that agrees on these values receives the same mask. Therefore, S is sufficient to predict the next-order target. The unique necessary feature set for that target cannot contain a feature outside S. The formal proof is given in Appendix A.

The theorem concerns ideal explainers, not finite-sample estimates. In practice, a learned mask can contain errors, and those errors can affect later orders. Nevertheless, the result explains why recursive selection is the right operation: it follows the decision process from the variables used at a local leaf toward the variables that determine the branch. Combining such paths across inputs can, in principle, reconstruct the decision process of the original black box from the bottom up. This is different from conventional decision-tree induction, which greedily fits one global tree from the top down through typically axis-aligned splits (Breiman et al., 1984; Quinlan, 1993).

4 EXPERIMENTS

We evaluate whether HEX recovers the ordered feature blocks that generate a prediction, rather than merely fitting the black-box output. Anonymous source code is available online.[1] We assess

[1] https://github.com/anonymous123anonymous2027-boop/hex_code_anon

instance-wise decision-path recovery, first-order predictive fidelity, and reconstruction of the black box's internal decision-tree structure.

4.1 CONTROLLED MECHANISMS

We use five nonlinear, known decision-tree mechanisms, denoted MODEL 1–MODEL 5; their complete decision rules and leaf functions are given in Appendix C. Models 1–3 are established synthetic mechanisms used to evaluate feature-selection methods by Yoon et al. (2018); Chen et al. (2018); Arik & Pfister (2021). Models 4–5 are more demanding mechanisms with deeper decision structures, multivariate conditions, unbalanced branches, and repeated features along a path. For every input, the generating mechanism provides its exact ordered path and the feature block used at each visited node. Methods are trained only on black-box input–output pairs; paths are used only for evaluation.

4.2 METHODS AND EVALUATION

For a test input, we compare the predicted ordered feature blocks with the ground-truth blocks after the best order-preserving alignment. We report the occurrence-aware ordered block-alignment (OBA) loss, for which zero means an identical ordered path, and aligned F1, which measures feature recovery after that alignment. We additionally report exact-path accuracy: the fraction of inputs for which every ordered block is correct. First-order test R^2 measures fidelity of the learned predictor to the original black box. Formal definitions of these metrics are given in Appendix F.

We compare with CART, Extra Trees (ET), Random Forests (RF), AdaBoost, Gradient Boosting (GB), XGBoost, LightGBM, and CatBoost (Breiman et al., 1984; Geurts et al., 2006; Breiman, 2001; Freund & Schapire, 1997; Friedman, 2001; Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018). For an ensemble, structural scores are averaged across its constituent trees;

HEX-S and HEX-M use the same HEX recursion and differ only in their inner Hide&Seek fitter. HEX-S directly optimizes the usual joint objective

$$\mathcal{L}_{\mathrm{S}} = \mathcal{L}_{\mathrm{fid}} + \lambda M,$$

where $\mathcal{L}_{\mathrm{fid}}$ is the task loss and M is the mean selected-mask mass. HEX-M uses a more sophisticated four-phase training process. First, the predictor is trained alone on the full input to predict the black-box output. Second, this predictor is frozen and the selector is trained for a short period to reproduce binary pseudo-masks constructed from the frozen predictor. Third, the selector and predictor are trained jointly with $\mathcal{L}_{\mathrm{S}}$. At the selected Phase-3 checkpoint, we record the expected development fidelity loss L_0 and mask mass M_0. The final regime-refinement phase minimizes

$$\mathcal{L}_{\mathrm{R}} = \mathcal{L}_{\mathrm{S}} + G_L(\mathcal{L}_{\mathrm{fid}}; L_0) + G_M(M; M_0) + \gamma R.$$

Here G_L and G_M are soft barriers against fidelity-loss and mask-mass increases relative to their Phase-3 reference values. A mask regime is the group of training inputs assigned the same hard mask $\mathbf{m}(\mathbf{x})$, and R penalizes regimes containing few inputs. Controllers increase λ when reference mass exceeds its budget and γ when the number of regimes stops falling. Complete training details are in Appendix F.

4.3 ORDERED-PATH RECOVERY ON SYNTHETIC INPUTS

Table 1 reports ordered feature-path recovery for independently sampled synthetic inputs. HEX-M has the lowest OBA loss and the highest aligned F1 on all five mechanisms; its advantage over the conventional baselines is substantial. HEX-S shows that recursion alone is useful, but the multi-phase fitter is substantially more reliable at recovering the intended masks.

First-order predictive fidelity results are reported in Appendix D: HEX-M also has the highest test R^2 on every synthetic-input mechanism. Exact-path recovery is reported in Appendix E.

4.4 RECOVERING THE BLACK BOX'S INTERNAL DECISION-TREE STRUCTURE

For each fitted HEX chain, we construct a tree directly from its Bottom-to-Top-surviving training paths. We first remove paths with no nonempty hard mask. We then choose as the root the topmost

Table 1: Structural recovery on synthetic mechanisms. Entries are mean ± 95% Student-t confidence intervals across $n = 10$ seeds. Lower OBA loss and higher aligned F1 are better.

	OBA loss (↓)					Aligned F1 (↑)				
Method	Model 1	Model 2	Model 3	Model 4	Model 5	Model 1	Model 2	Model 3	Model 4	Model 5
CART	0.551 ± 0.011	0.526 ± 0.012	0.483 ± 0.013	0.558 ± 0.004	0.576 ± 0.003	0.598 ± 0.013	0.626 ± 0.012	0.663 ± 0.013	0.498 ± 0.004	0.511 ± 0.004
ET	0.623 ± 0.002	0.608 ± 0.003	0.610 ± 0.002	0.597 ± 0.002	0.612 ± 0.003	0.505 ± 0.002	0.528 ± 0.003	0.517 ± 0.002	0.458 ± 0.002	0.475 ± 0.003
RF	0.555 ± 0.004	0.529 ± 0.007	0.489 ± 0.005	0.564 ± 0.002	0.582 ± 0.002	0.595 ± 0.004	0.622 ± 0.007	0.657 ± 0.005	0.493 ± 0.002	0.504 ± 0.002
AdaBoost	0.856 ± 0.005	0.849 ± 0.002	0.847 ± 0.003	0.935 ± 0.008	0.791 ± 0.004	0.189 ± 0.005	0.207 ± 0.003	0.226 ± 0.005	0.080 ± 0.008	0.176 ± 0.003
GB	0.551 ± 0.011	0.526 ± 0.013	0.482 ± 0.013	0.559 ± 0.003	0.576 ± 0.003	0.598 ± 0.013	0.626 ± 0.013	0.664 ± 0.013	0.498 ± 0.004	0.511 ± 0.004
XGBoost	0.535 ± 0.012	0.547 ± 0.010	0.583 ± 0.006	0.664 ± 0.004	0.726 ± 0.003	0.588 ± 0.012	0.546 ± 0.011	0.559 ± 0.007	0.364 ± 0.004	0.297 ± 0.003
LightGBM	0.665 ± 0.012	0.664 ± 0.007	0.588 ± 0.007	0.707 ± 0.003	0.735 ± 0.003	0.440 ± 0.014	0.423 ± 0.008	0.540 ± 0.008	0.323 ± 0.003	0.269 ± 0.003
CatBoost	0.647 ± 0.006	0.621 ± 0.008	0.667 ± 0.004	0.682 ± 0.006	0.727 ± 0.006	0.472 ± 0.007	0.492 ± 0.008	0.480 ± 0.005	0.346 ± 0.005	0.307 ± 0.005
HEX-S	0.268 ± 0.188	0.138 ± 0.018	0.192 ± 0.044	0.724 ± 0.097	0.437 ± 0.039	0.787 ± 0.178	0.912 ± 0.011	0.877 ± 0.030	0.369 ± 0.119	0.534 ± 0.029
HEX-M	**0.003 ± 0.001**	**0.007 ± 0.003**	**0.006 ± 0.003**	**0.086 ± 0.012**	**0.023 ± 0.002**	**0.998 ± 0.001**	**0.995 ± 0.002**	**0.996 ± 0.002**	**0.936 ± 0.011**	**0.977 ± 0.002**

Table 2: Tree structural recovery on synthetic mechanisms. Entries are normalized feature-labelled unordered tree-edit distance (TD), reported as mean ± 95% Student-t confidence intervals across $n = 10$ seeds; lower is better and 0 denotes an identical recovered tree. For HEX, a global tree is reconstructed from the learned paths by retaining at most four children per node until their cumulative path mass reaches 0.80. For ensembles, TD is the weighted expectation over constituent trees.

Method	Model 1	Model 2	Model 3	Model 4	Model 5
CART	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
ET	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
RF	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
AdaBoost	0.747 ± 0.000	0.742 ± 0.004	0.756 ± 0.006	0.494 ± 0.005	0.379 ± 0.000
GB	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
XGBoost	0.963 ± 0.000	0.964 ± 0.000	0.967 ± 0.000	0.911 ± 0.000	0.891 ± 0.000
LightGBM	0.929 ± 0.000	0.929 ± 0.000	0.935 ± 0.000	0.835 ± 0.002	0.795 ± 0.001
CatBoost	0.965 ± 0.000	0.966 ± 0.000	0.968 ± 0.000	0.920 ± 0.000	0.896 ± 0.000
HEX-S	0.222 ± 0.209	0.117 ± 0.056	0.281 ± 0.024	0.697 ± 0.133	0.603 ± 0.057
HEX-M	**0.000 ± 0.000**	**0.000 ± 0.000**	**0.000 ± 0.000**	**0.157 ± 0.182**	**0.000 ± 0.000**

mask supported by the largest number of remaining paths and discard paths with a different topmost mask. At every remaining node, we group paths by their next mask, rank the child masks by support, and retain the most supported children until they account for 80% of the node's path mass (with at most four children). Paths assigned to the other, unlikely regimes are pruned. Finally, an internal node that adds no new selected feature is contracted with its children. Thus this is a transparent post-processing of the learned paths, rather than another fitted tree learner.

We compare the resulting recovered structure with the generating black-box decision tree using normalized feature-labelled unordered tree-edit distance (TD) (Zhang & Shasha, 1989); zero denotes identical structures. Table 2 shows that the near-perfect HEX-M paths reconstruct the complete internal decision-tree structure of the black box on four of the five mechanisms, and remain substantially closer on Model 4 than the conventional alternatives. Theorem B.1 predicts this outcome for a perfect inner fitter: its ordered masks recover the node blocks needed to reconstruct the tree.

4.5 Recovery with real-world features: CoverType

Feature selection can become unstable when the covariates are highly correlated: several variables may carry overlapping predictive information, so small changes in fitting can lead to different selected sets. We study this effect using standardized covariates from the UCI Forest CoverType data set (Blackard & Dean, 1999), which records cartographic measurements of forest cells, including elevation, slope, distances, and hillshade. We use its ten continuous covariates.

We apply the same five controlled decision mechanisms to these real covariates. This keeps the input distribution, marginal scales, and feature correlations of a real data set, while retaining a precisely known decision path for each input. We use the same training protocol and structural metrics as in the synthetic-input study. For ordered-path recovery (Table 3), HEX-S is better, in terms of both OBA loss and aligned F1, on Models 1–3, whereas multi-phase HEX-M is better on the remaining two. Thus correlations make the preferred inner fitter mechanism-dependent, but do not remove the advantage of the recursive HEX construction. Exact-path recovery results for the CoverType setting are reported in Appendix E.

Table 3: Structural recovery with CoverType covariates. Entries are mean ± 95% Student-t confidence intervals. Lower OBA loss and higher aligned F1 are better.

	OBA loss (↓)					Aligned F1 (↑)				
Method	Model 1	Model 2	Model 3	Model 4	Model 5	Model 1	Model 2	Model 3	Model 4	Model 5
CART	0.575 ± 0.001	0.552 ± 0.001	0.588 ± 0.001	0.383 ± 0.002	0.522 ± 0.002	0.564 ± 0.001	0.598 ± 0.001	0.546 ± 0.001	0.714 ± 0.002	0.562 ± 0.002
ET	0.556 ± 0.003	0.420 ± 0.006	0.579 ± 0.001	0.485 ± 0.004	0.581 ± 0.003	0.583 ± 0.003	0.704 ± 0.005	0.555 ± 0.001	0.607 ± 0.004	0.502 ± 0.003
RF	0.581 ± 0.000	0.556 ± 0.000	0.593 ± 0.000	0.392 ± 0.000	0.543 ± 0.001	0.558 ± 0.001	0.593 ± 0.000	0.542 ± 0.000	0.704 ± 0.000	0.545 ± 0.001
AdaBoost	0.829 ± 0.001	0.617 ± 0.001	0.840 ± 0.001	0.838 ± 0.002	0.855 ± 0.000	0.209 ± 0.001	0.522 ± 0.001	0.206 ± 0.001	0.205 ± 0.002	0.173 ± 0.000
GB	0.574 ± 0.000	0.552 ± 0.000	0.589 ± 0.000	0.383 ± 0.000	0.524 ± 0.000	0.564 ± 0.000	0.598 ± 0.000	0.545 ± 0.000	0.713 ± 0.000	0.562 ± 0.000
XGBoost	0.561 ± 0.000	0.506 ± 0.000	0.697 ± 0.000	0.623 ± 0.000	0.711 ± 0.000	0.547 ± 0.000	0.608 ± 0.000	0.417 ± 0.000	0.437 ± 0.000	0.344 ± 0.000
LightGBM	0.630 ± 0.000	0.571 ± 0.000	0.691 ± 0.000	0.660 ± 0.000	0.741 ± 0.000	0.466 ± 0.000	0.540 ± 0.000	0.422 ± 0.000	0.395 ± 0.000	0.309 ± 0.000
CatBoost	0.696 ± 0.004	0.595 ± 0.004	0.746 ± 0.005	0.679 ± 0.003	0.740 ± 0.004	0.412 ± 0.005	0.526 ± 0.003	0.364 ± 0.005	0.374 ± 0.004	0.329 ± 0.005
HEX-S	**0.029 ± 0.041**	**0.092 ± 0.116**	**0.178 ± 0.122**	0.666 ± 0.040	0.526 ± 0.045	**0.980 ± 0.030**	**0.936 ± 0.086**	**0.879 ± 0.089**	0.454 ± 0.045	0.498 ± 0.030
HEX-M	0.156 ± 0.131	0.550 ± 0.082	0.269 ± 0.170	**0.240 ± 0.031**	**0.346 ± 0.049**	0.873 ± 0.110	0.553 ± 0.082	0.777 ± 0.141	**0.828 ± 0.022**	**0.620 ± 0.061**

Table 4: Tree structural recovery with CoverType covariates. Entries are mean ± 95% Student-t confidence intervals. Lower normalized feature-labelled unordered tree-edit distance (TD) is better, and 0 denotes an identical recovered tree. For HEX, a global tree is constructed from its learned paths as in the synthetic-input study. For ensembles, TD is the weighted expectation over constituent trees.

Method	Model 1	Model 2	Model 3	Model 4	Model 5
CART	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
ET	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
RF	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
AdaBoost	0.747 ± 0.000	0.762 ± 0.000	0.767 ± 0.000	0.516 ± 0.001	**0.377 ± 0.000**
GB	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
XGBoost	0.962 ± 0.000	0.965 ± 0.000	0.966 ± 0.000	0.915 ± 0.000	0.888 ± 0.000
LightGBM	0.929 ± 0.000	0.930 ± 0.000	0.934 ± 0.000	0.845 ± 0.000	0.794 ± 0.000
CatBoost	0.966 ± 0.000	0.967 ± 0.000	0.968 ± 0.000	0.923 ± 0.000	0.896 ± 0.000
HEX-S	**0.025 ± 0.059**	**0.194 ± 0.215**	0.279 ± 0.206	0.816 ± 0.089	0.732 ± 0.154
HEX-M	0.104 ± 0.086	0.352 ± 0.092	**0.187 ± 0.157**	**0.279 ± 0.138**	0.406 ± 0.149

We further apply the same path-to-tree construction used in the synthetic-input study and measure normalized feature-labelled unordered tree-edit distance. Table 4 shows that a HEX variant has the lowest distance on Models 1–4; AdaBoost is best on Model 5.

5 Conclusion

Instance-wise feature selection usually answers only which observed features suffice for one prediction. HEX treats that selection itself as an object to explain. Recursively explaining successive selection masks yields an ordered path of feature blocks: the first block supports the black-box output, and every higher block explains why the block below it was selected. Under unique necessary feature sets, we show that these blocks are nested, which makes the hierarchy both interpretable and structurally constrained.

On independently sampled synthetic inputs, HEX-M recovers ordered paths substantially more accurately than conventional tree and ensemble baselines. The recovered paths can in turn reconstruct the black box's internal decision-tree feature structure. With correlated real-world covariates, performance becomes more sensitive to the inner fitter, but a HEX variant achieves the lowest tree-edit distance on four of the five mechanisms.

HEX learns routing conditions and leaf behavior implicitly through its neural explainers, but it does not yet express them as explicit, human-readable symbolic conditions or functions. Extracting such forms from the recovered hierarchy, and extending HEX to settings with non-unique sufficient sets, are natural next steps. More broadly, HEX provides a way to move from identifying what was selected for a prediction to explaining why it was selected.

References

Sercan O. Arik and Tomas Pfister. TabNet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6679–6687, 2021.

Hubert Baniecki, Przemyslaw Biecek, and Fabian Fumagalli. Attributions all the way down? the metagame of interpretability. *arXiv preprint arXiv:2605.06295*, 2026.

Jock A. Blackard and Denis J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131–151, 1999. doi: 10.1016/S0168-1699(99)00046-0.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, California, 1984.

Brandon Carter, Jonas Mueller, Siddhartha Jain, and David K. Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pp. 567–576, 2019.

Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

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*, pp. 785–794, 2016. doi: 10.1145/2939672.2939785.

Ian Covert, Scott M. Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

Tal Ellinson, Hadi Mohasel Afshar, and Sally Cripps. Hide&seek: Learning to explain in an end-to-end differentiable network. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, 2017.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. doi: 10.1006/jcss.1997.1504.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. doi: 10.1214/aos/1013203451.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006. doi: 10.1007/s10994-006-6226-1.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

Joseph D. Janizek, Pascal Sturmfels, and Su-In Lee. Explaining explanations: Axiomatic feature interactions for deep networks. *Journal of Machine Learning Research*, 22(104):1–54, 2021.

Neil Jethani, Mukund Sudarshan, Yindalon Aphinyanaphongs, and Rajesh Ranganath. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pp. 1459–1467, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97 (1–2):273–324, 1997.

Huan Liu and Rudy Setiono. A probabilistic approach to feature selection: A filter solution. In *Proceedings of the 13th International Conference on Machine Learning*, pp. 319–327, 1996.

Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Daniel Lundström and Meisam Razaviyayn. A unifying framework to the analysis of interaction methods using synergy functions. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3145–3153, 2017.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3319–3328, 2017.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. INVASE: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.

Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989. doi: 10.1137/0218082.

A Nested Explanation Paths

This appendix gives the formal version of the nesting argument in Section 3. Recall that a necessary feature set is a sufficient set from which no selected feature can be removed; when it is unique, it is the unique minimum-size sufficient set.

Proposition A.1 (Nested masks)**.** *Suppose that, for every applicable order i and every input $\mathbf{x}$, $S_i(\mathbf{x})$ is the unique necessary feature set for the order-i target $\mathcal{B}_i(\mathbf{x})$, and that the HEX recursion is*

$$\mathcal{B}_{i+1}(\mathbf{x}) = \mathbf{m}_i(\mathbf{x}).$$

Then, for every applicable i and $\mathbf{x}$,

$$S_{i+1}(\mathbf{x}) \subseteq S_i(\mathbf{x}).$$

Proof. Fix i and $\mathbf{x}$, and let $U := S_i(\mathbf{x})$. Because U is sufficient for the order-i target, every feasible $\mathbf{x}'$ satisfying $\mathbf{x}'_U = \mathbf{x}_U$ has the same order-i target as $\mathbf{x}$.

Moreover, U remains sufficient at $\mathbf{x}'$, and every feature in U remains necessary: if some $j \in U$ could be removed at $\mathbf{x}'$, then $U \setminus \{j\}$ would also be sufficient at $\mathbf{x}$, contradicting the necessity of j in U. Thus, by uniqueness,

$$S_i(\mathbf{x}') = U = S_i(\mathbf{x}).$$

Consequently, fixing the features in U fixes the mask $\mathbf{m}_i(\mathbf{x})$. Hence U is sufficient for the next-order target $\mathcal{B}_{i+1}(\mathbf{x})$.

Starting from U, remove every feature that is not necessary for this next-order target. Since U is finite, this produces a necessary feature set $V \subseteq U$ for $\mathcal{B}_{i+1}(\mathbf{x})$. By uniqueness at order $i+1$, $V = S_{i+1}(\mathbf{x})$. Therefore,

$$S_{i+1}(\mathbf{x}) \subseteq U = S_i(\mathbf{x}).$$

□

B IDEAL RECOVERY OF DECISION-TREE FEATURE BLOCKS

This result describes an ideal setting in which HEX recovers the feature blocks along every decision path exactly. It is a perfect-explainer result: the path-faithfulness assumption below is not a guarantee of Hide&Seek or of any finite-sample estimator.

Consider a finite deterministic decision tree whose leaves all occur at depth D, with the root at depth 0. Each node v has a nonempty feature block $A_v \subseteq [d]$. An internal node routes an input using exactly the values indexed by A_v, and a leaf computes its output using exactly those indexed by A_v. For an input $\mathbf{x}$, write

$$v_0(\mathbf{x}), v_1(\mathbf{x}), \ldots, v_D(\mathbf{x})$$

for its executed root-to-leaf path, and define the cumulative feature sets and their masks by

$$P_q(\mathbf{x}) := \bigcup_{r=0}^{q} A_{v_r(\mathbf{x})}, \qquad q = 0, \ldots, D,$$

and

$$[\mathbf{p}_q(\mathbf{x})]_j := \mathbb{1}\{j \in P_q(\mathbf{x})\}, \qquad j \in [d].$$

Perfect explainer assumption. Feature blocks are disjoint along every root-to-leaf path. The inner explainer is *exactly path-faithful*: for the original tree target it returns $\mathbf{p}_D(\mathbf{x})$, and when fitted to the target $\mathbf{x} \mapsto \mathbf{p}_q(\mathbf{x})$, it returns $\mathbf{p}_{q-1}(\mathbf{x})$, for $q = 1, \ldots, D$. Finally, on the training population, each $\mathbf{p}_q$, $q = 1, \ldots, D$, takes at least two distinct values. The last condition prevents an earlier terminal mask.

Theorem B.1 (Ideal recovery by HEX)**.** *Under the perfect explainer assumptions above, HEX, run without a maximum-order cap, fits exactly $D+1$ explainers and then halts because its final mask is constant. For every input $\mathbf{x}$, its selection sets satisfy*

$$S_i(\mathbf{x}) = P_{D-i+1}(\mathbf{x}), \qquad i = 1, \ldots, D+1.$$

Consequently,

$$S_i(\mathbf{x}) \setminus S_{i+1}(\mathbf{x}) = A_{v_{D-i+1}(\mathbf{x})}, \qquad i = 1, \ldots, D,$$

and $S_{D+1}(\mathbf{x}) = A_{v_0(\mathbf{x})}$.

Proof. Path faithfulness gives $\mathbf{m}_1(\mathbf{x}) = \mathbf{p}_D(\mathbf{x})$. Hence the second HEX target is $\mathbf{p}_D$, and path faithfulness gives $\mathbf{m}_2(\mathbf{x}) = \mathbf{p}_{D-1}(\mathbf{x})$. Repeating this argument yields

$$\mathbf{m}_i(\mathbf{x}) = \mathbf{p}_{D-i+1}(\mathbf{x}), \qquad i = 1, \ldots, D+1.$$

The masks $\mathbf{p}_D, \ldots, \mathbf{p}_1$ are nonconstant on the training population, so HEX cannot stop earlier. The final mask, $\mathbf{p}_0$, is the fixed root-node block and is constant; hence HEX halts. Taking supports gives the first claim. Path-disjointness gives

$$P_q(\mathbf{x}) \setminus P_{q-1}(\mathbf{x}) = A_{v_q(\mathbf{x})}, \qquad q = 1, \ldots, D,$$

which yields the remaining claims after substituting $q = D - i + 1$. □

Remark B.2 (Scope of recovery). The theorem identifies the ordered feature blocks on each observed path. It does not recover numerical routing conditions, leaf functions, or unvisited branches. Those require a separate procedure after the HEX paths have been obtained.

Table 5: First-order predictive fidelity on synthetic-input mechanisms. Entries are test R^2, reported as mean $\pm$ 95% Student-t confidence intervals across $n = 10$ seeds. Higher is better.

Method	Model 1	Model 2	Model 3	Model 4	Model 5
CART	0.923 ± 0.010	0.956 ± 0.006	0.916 ± 0.009	0.045 ± 0.111	0.554 ± 0.028
ET	0.931 ± 0.007	0.953 ± 0.005	0.922 ± 0.010	0.456 ± 0.045	0.774 ± 0.010
RF	0.970 ± 0.002	0.986 ± 0.003	0.965 ± 0.008	0.531 ± 0.050	0.789 ± 0.013
AdaBoost	0.139 ± 0.016	0.530 ± 0.047	0.209 ± 0.066	0.155 ± 0.057	0.450 ± 0.014
GB	0.931 ± 0.006	0.961 ± 0.005	0.926 ± 0.010	0.104 ± 0.122	0.591 ± 0.027
XGBoost	0.915 ± 0.011	0.914 ± 0.051	0.933 ± 0.015	0.396 ± 0.133	0.761 ± 0.009
LightGBM	0.861 ± 0.010	0.893 ± 0.011	0.932 ± 0.011	0.457 ± 0.039	0.755 ± 0.009
CatBoost	0.817 ± 0.011	0.870 ± 0.017	0.850 ± 0.014	0.413 ± 0.036	0.697 ± 0.012
HEX-S	0.932 ± 0.013	0.919 ± 0.016	0.891 ± 0.014	0.722 ± 0.025	0.761 ± 0.014
HEX-M	$\mathbf{0.994 \pm 0.001}$	$\mathbf{0.997 \pm 0.001}$	$\mathbf{0.995 \pm 0.001}$	$\mathbf{0.965 \pm 0.006}$	$\mathbf{0.971 \pm 0.005}$

C SYNTHETIC MECHANISMS

All synthetic-input experiments use independent standard-normal features $X_1, \ldots, X_{11}$. Define three shared leaf functions,

$$h_1(\mathbf{x}) = x_1 x_2,$$
$$h_2(\mathbf{x}) = x_3^2 + x_4^2 + x_5^2 + x_6^2 - 4,$$
$$h_3(\mathbf{x}) = -10 \sin(0.2 x_7) + |x_8| + x_9 + \exp(-x_{10}) - 2.4.$$

Models 1–3 share the root condition $x_{11} < 0$ and return, respectively, (h_1, h_2), (h_1, h_3), and (h_2, h_3) on its true and false branches. Thus these three mechanisms isolate different pairs of leaf-feature blocks under the same decision node.

Model 4 has root condition $x_1 x_2 < 0$. On its true branch it tests $x_4 < 0$ and returns $x_5 - x_6$ or h_3; on its false branch it tests $x_8 < 0$ and returns $x_5 + x_6$ or $x_1 x_3$. It therefore includes a repeated feature occurrence on one realized path.

Model 5 has root condition $\sin(x_1) + \sin(x_2) > 0$. Its true branch returns $x_3 + 2x_4$. Its false branch first tests $x_5^2 - x_6^2 > 0$: the true branch then tests $x_7 > 0$ and returns $1.2x_9 - 0.6x_{10}$ or $-0.9x_{11}$; the false branch tests $x_8^2 < \tau$ and returns $0.7x_9 + 1.1x_{10}$ or $1.4x_{11}$, where $\tau = [\Phi^{-1}(3/4)]^2$. This deliberately unbalanced tree combines nonlinear multivariate conditions with paths of different lengths.

D PREDICTIVE FIDELITY

Tables 5 and 6 report first-order predictive fidelity against the controlled black-box output for, respectively, independent synthetic inputs and real-world CoverType covariates. For HEX, this is the fidelity of its first-order predictor. Across the ten model–input-distribution settings, a HEX variant attains the highest mean R^2 in nine; both variants retain strong predictive fidelity in every setting. Thus, the structural gains reported in the main text are not obtained by sacrificing first-order predictive fidelity.

E EXACT ORDERED-PATH RECOVERY

Tables 7 and 8 report exact ordered-path recovery for independent synthetic inputs and CoverType covariates, respectively. This is the strictest path metric: an input is correct only when every recovered feature mask matches its complete ground-truth ordered path. On independent synthetic inputs (Table 7), none of the conventional tree and ensemble baselines recovers an exact path on any mechanism. HEX-M recovers nearly all paths on Models 1–3 and 5, and a clear majority on Model 4.

With correlated CoverType covariates (Table 8), exact recovery is more difficult, but a HEX variant remains the best method on every mechanism: HEX-S on Models 1–2 and HEX-M on Models 3–5.

Table 6: First-order predictive fidelity with CoverType covariates. Entries are test R^2, reported as mean $\pm$ 95% Student-t confidence intervals. Higher is better.

Method	Model 1	Model 2	Model 3	Model 4	Model 5
CART	0.976 ± 0.001	0.996 ± 0.000	0.990 ± 0.000	0.979 ± 0.000	0.873 ± 0.009
ET	0.995 ± 0.000	0.999 ± 0.000	0.998 ± 0.000	0.968 ± 0.001	0.946 ± 0.001
RF	0.992 ± 0.000	0.999 ± 0.000	0.996 ± 0.000	$\mathbf{0.992 \pm 0.000}$	0.938 ± 0.002
AdaBoost	0.329 ± 0.005	0.742 ± 0.003	0.695 ± 0.002	0.503 ± 0.003	0.561 ± 0.001
GB	0.980 ± 0.000	0.997 ± 0.000	0.991 ± 0.000	0.980 ± 0.000	0.887 ± 0.001
XGBoost	0.977 ± 0.000	0.995 ± 0.000	0.984 ± 0.000	0.959 ± 0.000	0.872 ± 0.000
LightGBM	0.973 ± 0.000	0.993 ± 0.000	0.989 ± 0.000	0.949 ± 0.000	0.873 ± 0.000
CatBoost	0.912 ± 0.002	0.975 ± 0.001	0.973 ± 0.000	0.904 ± 0.002	0.817 ± 0.001
HEX-S	0.967 ± 0.010	0.981 ± 0.002	0.980 ± 0.003	0.820 ± 0.016	0.846 ± 0.003
HEX-M	$\mathbf{0.999 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	0.979 ± 0.006	$\mathbf{0.987 \pm 0.003}$

Table 7: Exact ordered-path recovery on synthetic-input mechanisms. Entries are mean $\pm$ 95% Student-t confidence intervals across $n = 10$ seeds.

Method	Model 1	Model 2	Model 3	Model 4	Model 5
CART	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
ET	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
RF	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
AdaBoost	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
GB	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
XGBoost	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
LightGBM	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
CatBoost	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
HEX-S	0.247 ± 0.149	0.271 ± 0.103	0.111 ± 0.069	0.000 ± 0.000	0.080 ± 0.122
HEX-M	$\mathbf{0.991 \pm 0.006}$	$\mathbf{0.975 \pm 0.012}$	$\mathbf{0.978 \pm 0.012}$	$\mathbf{0.631 \pm 0.024}$	$\mathbf{0.927 \pm 0.008}$

In particular, both variants retain substantial exact-path recovery on several mechanisms despite the ambiguity induced by correlated features.

F Evaluation and training details

F.1 Ordered-path metrics

For a test input $\mathbf{x}$, let $\mathcal{O}^\star(\mathbf{x}) = (G_1^\star, \dots, G_L^\star)$ be its ground-truth sequence of local feature blocks and let $\widehat{\mathcal{O}}(\mathbf{x}) = (A_1, \dots, A_K)$ be a method's predicted sequence, both in root-to-leaf order. An alignment is a nondecreasing sequence

$$0 = t_0 \le t_1 \le \cdots \le t_L = K.$$

It assigns the union

$$\widehat{G}_\ell^{\mathbf{t}} = \bigcup_{r=t_{\ell-1}+1}^{t_\ell} A_r$$

to ground-truth level ℓ; an empty interval represents an omitted block. For sets U, V, write

$$d_J(U, V) = 1 - \frac{|U \cap V|}{|U \cup V|},$$

with $d_J(\varnothing, \varnothing) = 0$. The OBA loss is the minimum average Jaccard loss over all such alignments:

$$\mathrm{OBA}(\mathbf{x}) = \min_{\mathbf{t}} \frac{1}{L} \sum_{\ell=1}^{L} d_J\big(G_\ell^\star, \widehat{G}_\ell^{\mathbf{t}}\big).$$

Aligned precision, recall, and F1 are computed from the same minimizing alignment and then averaged across inputs. Exact-path accuracy is the fraction of test inputs with OBA equal to zero. For

Table 8: Exact ordered-path recovery with CoverType covariates. Entries are mean ± 95% Student-t confidence intervals.

Method	Model 1	Model 2	Model 3	Model 4	Model 5
CART	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
ET	0.000 ± 0.000	0.003 ± 0.001	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
RF	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
AdaBoost	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
GB	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
XGBoost	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
LightGBM	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
CatBoost	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
HEX-S	$\mathbf{0.915 \pm 0.128}$	$\mathbf{0.624 \pm 0.249}$	0.375 ± 0.234	0.000 ± 0.000	0.026 ± 0.042
HEX-M	0.713 ± 0.198	0.000 ± 0.000	$\mathbf{0.580 \pm 0.275}$	$\mathbf{0.052 \pm 0.080}$	$\mathbf{0.404 \pm 0.022}$

tree recovery, we use normalized feature-labelled unordered tree-edit distance; for an ensemble, its reported value is the estimator-weighted mean over constituent trees.

F.2 Multi-phase Hide&Seek fitting

HEX-S jointly trains Hide&Seek with fidelity loss plus mean mask mass. HEX-M uses the following four stages. Write B for a minibatch of b examples, F for the functional input expansion, and let h_ϕ and g_θ denote the predictor and selector, respectively.

Phase 1: predictor pretraining. The predictor is first trained on the complete input, without a mask, using the task loss. For regression, this is

$$\mathcal{L}_{\text{predictor}}(B) = \frac{1}{b} \sum_{n \in B} \left[h_\phi\Big(F(\mathbf{x}^{(n)})\Big) - y^{(n)} \right]^2 . \tag{1}$$

For a binary or multilabel higher-order target, the corresponding cross-entropy task loss replaces squared error. The best development checkpoint is frozen for the next two stages.

Pseudo-mask construction and Phase 2: selector pretraining. For each training input, we start with every allowable feature retained. Using the frozen predictor, greedy conditional deletion repeatedly tests a coordinate by replacing it with a value independently drawn from its empirical training marginal. A coordinate is deleted when this replacement does not increase the frozen predictor's loss (or, for the first regression order, its output shift) beyond a fixed tolerance. The remaining binary vector is the pseudo-mask $\widetilde{\mathbf{m}}^{(n)}$. The selector is then pretrained to reproduce these masks with coordinatewise binary cross-entropy:

$$\mathcal{L}_{\text{selector}}(B) = -\frac{1}{bd} \sum_{n \in B} \sum_{j=1}^{d} \left[\widetilde{m}_j^{(n)} \log g_{\theta,j}(\mathbf{x}^{(n)}) + (1 - \widetilde{m}_j^{(n)}) \log\Big(1 - g_{\theta,j}(\mathbf{x}^{(n)})\Big) \right] . \tag{2}$$

Phase 3: joint Hide&Seek training. The selector and predictor are unfrozen and trained jointly. Let L_B be the task loss on the randomly replaced, soft-masked inputs and let $M_B = (bd)^{-1} \sum_{n \in B} \sum_j g_{\theta,j}(\mathbf{x}^{(n)})$ be mean soft-mask mass. Before this phase, detached training estimates define the reference fidelity and mass L_0, M_0, and the tolerances define $L_{\max}^{(3)} = (1 + c_L)L_0$ and $M_{\max}^{(3)} = (1 + c_M)M_0$. With $\text{sp}_\kappa(u) = \kappa \log(1 + \exp(u/\kappa))$, the objective is

$$J_B^{(3)} = L_B + \lambda M_B + G_{L,B}^{(3)} + G_{M,B}^{(3)}, \tag{3}$$

$$G_{L,B}^{(3)} = \beta_L L_0 \ \text{sp}_\kappa\left(\frac{L_B}{L_{\max}^{(3)}} - 1 \right), \tag{4}$$

$$G_{M,B}^{(3)} = \beta_M \lambda M_0 \ \text{sp}_\kappa\left(\frac{M_B}{M_{\max}^{(3)}} - 1 \right). \tag{5}$$

Thus, the ordinary fidelity–sparsity trade-off remains active, while the two smooth barriers discourage avoidable deterioration beyond the pre-joint-training references. Development fidelity and deployed hard mask mass select the Phase-3 checkpoint.

Phase 4: regime-support refinement. Starting from that selected checkpoint, we reestimate detached training references $L_0^{(4)}, M_0^{(4)}$ and their slack budgets $L_{\max}^{(4)}, M_{\max}^{(4)}$. A *mask regime* is a distinct hard mask pattern. For a soft mask in a minibatch, let q_n denote the differentiable count of the other masks sufficiently similar to it, and let r_B be the required support count implied by the target minimum regime fraction. The support penalty is

$$R_B = \frac{1}{b} \sum_{n \in B} \mathrm{sp}_T \left(\frac{r_B - q_n}{T} \right). \tag{6}$$

The final objective is

$$J_B^{(4)} = L_B + \lambda M_B + G_{L,B}^{(4)} + G_{M,B}^{(4)} + \gamma R_B, \tag{7}$$

$$G_{L,B}^{(4)} = \beta_L L_0^{(4)} \, \mathrm{sp}_\kappa \left(\frac{L_B}{L_{\max}^{(4)}} - 1 \right), \tag{8}$$

$$G_{M,B}^{(4)} = \beta_M \lambda M_0^{(4)} \, \mathrm{sp}_\kappa \left(\frac{M_B}{M_{\max}^{(4)}} - 1 \right). \tag{9}$$

Here R_B makes undersupported mask patterns costly while the reanchored barriers retain the selected model's fidelity and sparsity. During refinement, λ grows only when a detached training mass reference exceeds its budget, and γ grows only after the hard regime count plateaus. Checkpoint selection uses held-out development fidelity, mask mass, and regime count; ground-truth masks and paths are never used for training or selection.

F.3 HEX HYPERPARAMETERS

This subsection records the main settings used for all reported experiments; run-specific artifacts retain the complete configurations. We did not conduct an exhaustive hyperparameter search: preliminary experiments indicated that performance was not highly sensitive to most of these settings. Both selector and predictor use width 32, a batch size of 256, and a mask threshold of 0.5. Functional input expansion is disabled.

HEX-S. The selector and predictor each have two hidden layers and are jointly trained for 500 epochs with Adam step size 10^{-3}. The sparsity coefficient is $\lambda = 0.3$, with the quadratic mass schedule used by the original single-phase Hide&Seek implementation.

HEX-M loss coefficients. The Phase-1–3 Adam step size is 10^{-3}; refinement uses 5×10^{-4}. During refinement, λ is multiplied by 1.1 only if detached mask mass violates its budget. $\beta_L = \beta_M = 1.5$. The initial regime-support weight is $\gamma_0 = 0.2 L_0 / R_0$, where R_0 is the initial support penalty; the default controller increases it linearly by γ_0 after a two-evaluation regime-count plateau. Refinement reanchors the barriers and uses $(c_L, c_M) = (0.05, 0)$.