UNIFYING THE SUBSET SELECTION PROBLEM IN MACHINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Many machine learning tasks, including feature and data selection, ensemble pruning, and circuit discovery, reduce to maximizing an expensive set function over a finite ground set. Existing methods exploit structural properties such as modularity, submodularity, or low degree; we develop certificates that quantify these properties and find that none holds exactly for any of the 321 set functions in our benchmark. Likelihood-free Bayesian optimization (LFBO) offers a natural alternative when such assumptions fail by adaptively learning improvement above a threshold, but it was developed primarily for continuous hyperparameter optimization rather than exponentially large Boolean search spaces. We adapt LFBO to subset selection using tree-ensemble surrogates, which can efficiently represent the sparse, low-degree, and hierarchical interactions we find in these set functions. Across all four domains, our method ranks first in three at equal query budget and second in circuit discovery, where it achieves quality comparable to the strongest Gaussian-process BO baseline at substantially lower computational cost.

1 INTRODUCTION

Selecting an optimal subset from a given set of items or components recurs throughout machine learning. Such a selection can lower computational costs, preserve or improve performance, and help explain how a system works. This goal takes different forms across machine learning: feature selection seeks a useful subset of input variables (Chen et al., 2018; Yamada et al., 2020); data selection seeks training examples that support effective learning (Sorscher et al., 2022; Ghorbani & Zou, 2019); ensemble pruning seeks a small set of models with strong combined predictions (Caruana et al., 2004; Martínez-Muñoz et al., 2009); and circuit discovery seeks a subset of transformer edges that preserves and helps explain a behavior of interest (Conmy et al., 2023; Syed et al., 2024).

These four tasks, and more, can all be formulated as instances of a **subset selection problem**. Let $[d] = \{1, \dots, d\}$ denote a set of players and let $f : 2^{[d]} \to \mathbb{R}$ be a set function, which we call a **utility**, that assigns a value to each subset $S \subseteq [d]$, such as the validation accuracy of a model trained on the selected examples. The objective is to find an optimal subset:

$$S^\star \in \arg\max_{S \subseteq [d]} f(S). \tag{1}$$

We assume query access to f: each query specifies a subset S and returns its value $f(S)$. Because evaluating a subset can be computationally expensive in all of the aforementioned applications, our goal is to find a subset whose value is close to $f(S^\star)$ **using as few queries as possible**. The structure of f determines which approaches can solve or approximate the selection problem:

- If f is **monotone**, a solution to the unconstrained problem is trivially given by the full set when f is monotone increasing, and by the empty set when f is monotone decreasing.
- If f is **modular**, each element contributes additively, so an optimal subset is obtained by selecting all elements with positive contributions in a linear approximation, such as Shapley (Ghorbani & Zou, 2019) or Banzhaf (Wang & Jia, 2023) values (see Figure 1).
- If f is **submodular**, its marginal gains exhibit diminishing returns. For nonnegative f, the above problem admits a constant-factor approximation via a greedy algorithm (Buchbinder et al., 2015).
- If f is **quadratic**, it has $O(d^2)$ parameters that can be learned from function evaluations, as in BOCS (Baptista & Poloczek, 2018). Finding its optimum, however, is NP-hard in general.

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.

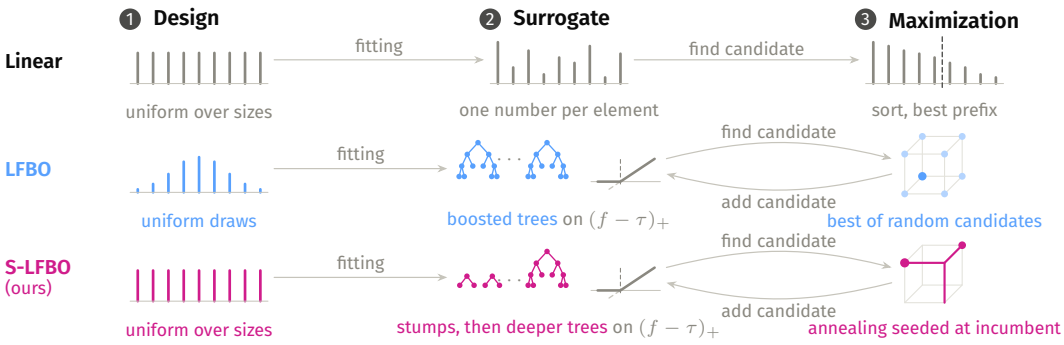

Figure 1: Subset selection as a three-step loop. Linear methods fit once and rank, LFBO iterates between surrogate fitting on $(f-\tau)_+$ and its maximizer. S-LFBO tailors the components to the selection problem, covers every subset size, stumps first, and anneals from the incumbent.

In practice, machine learning subset-selection objectives may **depart substantially from the structural assumptions underlying these methods** (Das & Kempe, 2008; Fryer et al., 2021; Wang et al., 2024). In fact, none of the objectives we study satisfies any of these structural assumptions exactly. Moreover, even when an objective is close to satisfying a structural property, a method designed to exploit that property need not recover near-optimal subsets. These findings (see Figure 2) motivate methods that learn and exploit alternative, more universal structures in the objective.

1.1 Our Contributions

Our starting point is to ask what alternative structure these objectives exhibit and how it can guide subset selection. Prior work suggests that the utilities in machine learning can be well approximated by sparse, low-degree polynomials (Stobbe & Krause, 2012; Gorji et al., 2025; Kang et al., 2025) and that low-depth tree ensembles can effectively capture their interactions (Butler et al., 2025). In the objectives we study, we find that, in the Fourier basis, **interactions are sparse, low-degree, and hierarchical**: a small fraction of low-degree coefficients accounts for nearly all of the Fourier spectral energy, and interactions over a set of players are accompanied by interactions over its subsets far more often than expected by chance.

Therefore, a natural idea is to learn a surrogate tree ensemble $\hat{f}$ and maximize it directly. In our experiments, this does not work well because we do not actually need the surrogate $\hat{f}$ to fit f well everywhere, **the surrogate only needs to be accurate for the largest values**. That is, fitting the surrogate everywhere spends too many queries on subsets with small values.

Although these selection tasks have largely been studied separately, they share the problem of optimizing an expensive function with limited evaluations. Bayesian optimization (BO) addresses this problem by using previous evaluations to guide the search, balancing exploration with exploitation (Shahriari et al., 2016; Baptista & Poloczek, 2018; Oh et al., 2019; Dreczkowski et al., 2023). We build on **likelihood-free Bayesian optimization** (LFBO; Song et al., 2022), which learns a surrogate for improvement above a threshold τ. The target is $a(S) = \max\{f(S) - \tau, 0\}$, measuring how far a subset's value exceeds τ and assigning zero below it. The surrogate, therefore, only needs to be accurate on the largest values. Additionally, LFBO iteratively searches for a subset that maximizes the learned surrogate, evaluates that subset, and updates the surrogate with the new observation. Section 2.2 provides the weighted classification objective and query rule.

LFBO offers a general framework, but applying it to subset selection requires choices suited to an **exponentially large search space** (Song et al., 2022; Boros & Hammer, 2002). We use **tree ensembles as surrogates**, motivated by the structure of both f and its improvement target a. If f is a single decision tree, applying the improvement transformation to its leaf values represents a exactly; if the tree approximates f, the transformed tree approximates a with no greater error (Proposition 3). This guarantee does not extend directly to ensembles, since thresholding a sum need not preserve its constituent trees. Empirically, however, we find sparse, hierarchical interactions in both f and a, supporting tree ensembles for learning the improvement target.

We also tailor the initial sampling and surrogate optimization to the selection problem. Uniform sampling over subsets concentrates their sizes near $d/2$, whereas the best subsets in our benchmark span a wide range of sizes. We therefore **sample uniformly over subset sizes**, then uniformly among subsets of the chosen size. To optimize the surrogate, we use **simulated annealing with warm starts** from previously promising subsets, since random candidate search in high dimensions tends to propose subsets far from those already evaluated. The resulting method, S-LFBO, **leads three of the four domains at equal query budget** and ranks second to GP-BO (Dreczkowski et al., 2023) in circuit discovery, where it reaches **similar quality much faster** (Table 2 and Figure 4).

Our contributions are four-fold:

1. **Unified formulation.** We formulate feature selection, data selection, ensemble pruning, and circuit discovery as instances of a common subset selection problem, bringing together methods from these domains and Bayesian optimization into one comparison.
2. **Extensive benchmark.** Our benchmark spans 321 utilities in four domains, across diverse datasets and models, and measures solution quality under both query and compute budgets.
3. **Structural analysis.** We quantify how closely these objectives satisfy properties underlying common optimization guarantees and find that none holds exactly in our benchmark. We instead identify sparse, hierarchical interactions in both the objectives and their improvement targets, motivating tree-ensemble surrogates.
4. **Effective selection method.** We develop S-LFBO, adapting LFBO with tree-ensemble surrogates, sampling across subset sizes, and warm-started acquisition optimization. It leads three domains at equal query budget and ranks second in circuit discovery, where it reaches quality comparable to GP-BO much faster.

1.2 Related Work

Attribution and NASH. Shapley and Banzhaf scores account for interactions, but selecting by a fixed ranking keeps one number per player (Ghorbani & Zou, 2019; Wang & Jia, 2023). NASH (Tian et al., 2026) combines several such scores nonlinearly. We instead learn a surrogate with interactions and refit it after every evaluation, so it also guides which subsets to query next.

ProxySPEX. ProxySPEX (Butler et al., 2025) fits boosted trees to exploit hierarchical interactions and extracts attributions from the fitted model. We build on its motivation for trees. Our focus is on learning improvement above a threshold and repeatedly choosing the next promising subset under a small query budget. The structure of the improvement target is crucial, since a surrogate that fits the original function well need not be equally suited to the target used for selection.

Bayesian optimization. Combinatorial Bayesian optimization uses expressive surrogates on the Boolean cube, such as sparse quadratic models (Baptista & Poloczek, 2018) or Gaussian processes (Oh et al., 2019; Dreczkowski et al., 2023), which either assume low degree or become costly to fit in our high-dimensional problems. Trees are cheap to refit and already appear in Bayesian optimization as random-forest surrogates in SMAC (Lindauer et al., 2022) and as classifiers in BORE (Tiao et al., 2021) and LFBO (Song et al., 2022). We inherit LFBO's weighted classification objective and focus on which classifier, initial samples, and search procedure make it work for subset selection, connecting these choices to the structure we measure in the utilities and their improvement targets.

2 Background

This section reviews the methods that are optimal under structural properties of f (Section 2.1) and likelihood-free Bayesian optimization, which assumes no such properties (Section 2.2).

2.1 Prior Work on the Subset Selection Problem

Table 1 defines the properties of f and lists the methods that are optimal or approximately optimal under each. Monotonicity is the simplest, where taking all players is optimal. An extended section can be found in Appendix B, and in Appendix D, we describe the implemented methods in detail.

Table 1: Properties of f, the methods they justify, and whether the improvement target $a = [f - \tau]_+$ inherits them. Proofs and counterexamples are in Appendix A.1.

property on f	**definition** (all $S \subseteq T \subseteq [d], i \notin T$)	**best method**	**property on** a
monotone	$f(S \cup \{i\}) \geq f(S)$	take all	preserved
modular	$f(S) = w_0 + \sum_{i \in S} w_i$	semivalue ranking	weakens to MTM
MTM	$f(S) = g\big(\sum_{i \in S} w_i\big)$, g nondecreasing	semivalue ranking	preserved, same weights
submodular	$f(S \cup \{i\}) - f(S) \geq f(T \cup \{i\}) - f(T)$	double greedy	can be lost
supermodular	$f(S \cup \{i\}) - f(S) \leq f(T \cup \{i\}) - f(T)$	min-cut (degree 2)	preserved, if f monotone
degree $\leq k$	$f(S) = \sum_{R \subseteq S,\, \lvert R \rvert \leq k} m(R)$	BOCS ($k = 2$)	can rise to d

Search. Search methods use the evaluations of f directly, without fitting a surrogate. Hill climbing starts from the empty set and flips one player at a time, only accepting improvements (Lei et al., 2021), whilst simulated annealing accepts worse subsets with a probability that decays over time (Kirkpatrick et al., 1983). For nonnegative submodular f and exact evaluations, randomized double greedy reaches half the optimum in expectation (Buchbinder et al., 2015), the best factor possible with polynomially many queries (Feige et al., 2011). Greedy forward selection evaluates every single-player addition and keeps the best (Nemhauser et al., 1978; Caruana et al., 2004), and double greedy adds or removes players to handle non-monotone utilities (Buchbinder et al., 2015).

Ranking. Semivalues such as Data Shapley (Ghorbani & Zou, 2019) and Data Banzhaf (Wang & Jia, 2023) average a player's marginal contributions and equal the coefficients of the best weighted least-squares additive fit to f, estimated once from sampled subsets (Lundberg & Lee, 2017; Faigle & Grabisch, 2020). Ranking by them is exact for modular f, where every semivalue equals w_i, and for monotonically transformed modular (MTM) f, where g preserves the order of the additive scores. The positive semivalues form $S^\star$, and the top k the best subset of size k (Wang et al., 2024). NASH (Tian et al., 2026) aggregates per-validation-point semivalues concavely into a monotone submodular surrogate, so the greedy guarantee holds for the surrogate, not for f.

Degree two. BOCS (Baptista & Poloczek, 2018) learns the $1 + d + \binom{d}{2}$ coefficients of a degree-two f from few queries via a sparse prior. If f is supermodular (nonnegative pairwise coefficients), a minimum cut maximizes it exactly; otherwise this is NP-hard (Boros & Hammer, 2002).

2.2 Likelihood-free Bayesian Optimization (LFBO)

Bayesian optimization assumes no property of f. It learns a surrogate from the evaluations, refits it after every query, and uses it to choose the next one (Shahriari et al., 2016). LFBO additionally fits the surrogate only to improvement above a threshold, learning it directly through weighted classification. This concentrates the fit on the best subsets observed so far (Song et al., 2022). Let $\mathcal{D}_t = \{(S_i, y_i)\}_{i=1}^t$ contain the evaluated subsets, with $y_i = f(S_i)$ for deterministic evaluations, and choose a threshold τ, typically a quantile of the observed values. The learning target is

$$a(S) = [f(S) - \tau]_+ , \qquad [u]_+ = \max\{u, 0\} ,$$

and a classifier $C_\theta : 2^{[d]} \to (0, 1)$ is fit by minimizing a weighted log-loss:

$$\hat{\theta}_t \in \arg\min_\theta -\frac{1}{t} \sum_{i=1}^{t} \big[[y_i - \tau]_+ \log C_\theta(S_i) + \log\big(1 - C_\theta(S_i)\big) \big] . \tag{2}$$

In Equation (2), every observation contributes a negative example with weight 1 and a positive example with weight $[y_i - \tau]_+$; zero-weight positive examples can be omitted. For a fixed deterministic f, the unrestricted pointwise optimum, allowing the boundary value $C^\star(S) = 0$, satisfies

$$C^\star(S) = \frac{a(S)}{1 + a(S)}, \qquad \frac{C^\star(S)}{1 - C^\star(S)} = a(S). \tag{3}$$

Thus, the classifier odds recover improvement in Equation (3), whereas a fit on a finite number of observations only approximates this relationship. With noisy evaluations Y, the population version instead recovers $\mathbb{E}[[Y - \tau]_+ \mid S]$, averaging over the observation noise at S. Since the odds increase

	additive	pairwise	MTM	monotone	submodular	supermodular
feature selection	.79	.97	.88	.72	.63	.48
66/90 utilities	.28–.98	.67–.99	.76–.96	.55–.98	.42–.92	.21–.92
data selection	.67	.87	.85	.53	.52	.49
141/173 utilities	.13–.91	.33–.98	.73–.94	.46–.99	.47–.76	.24–.55
ensemble pruning	.86	.96	.92	.62	.59	.43
51 utilities	.32–.93	.65–.99	.85–.95	.35–.95	.40–.94	.15–.93
circuit discovery	.52	.82	.79	.21	.73	.74
7 utilities	.19–.76	.38–.92	.70–.85	.02–.27	.70–.80	.71–.80

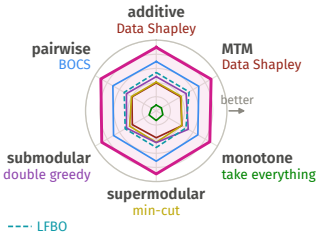

Figure 2: No property holds, and S-LFBO leads near each of them. *Left:* how much of f each property's model captures, over the utilities of Table 2, median and range; 1 means the utility has the property. *Right:* share of the other plotted methods beaten at 100 queries, each utility weighted by its score on the left; markers sit on the property a method is built for. Definitions, coverage and weighting in Appendix C.3.

with the classifier output, LFBO searches for

$$S_{t+1} \in \arg\max_{S \subseteq [d]} C_{\hat{\theta}_t}(S) = \arg\max_{S \subseteq [d]} \frac{C_{\hat{\theta}_t}(S)}{1 - C_{\hat{\theta}_t}(S)}. \tag{4}$$

The search in Equation (4) can be approximate. LFBO evaluates the proposed subset, appends the observation to $\mathcal{D}_t$, updates τ, and refits until the query budget is spent.

3 The Selection Problem in Machine Learning

We consider four applications of the selection problem in machine learning. In feature and data selection, $f(S)$ is the validation score of a model refit on the selected features or training points; in ensemble pruning, that of the selected members' combined predictions; and in circuit discovery, the faithfulness of the model with all unselected components patched (Zhang & Nanda, 2024). Refitting makes evaluations noisy in the first two applications, while the other two are exact. The optimal methods from Section 2.1 hold only under the mentioned properties, which we first investigate.

Every property fails in every domain. We test each property of Table 1 on sampled subsets, since a single counterexample rejects it and no enumeration of all 2^d subsets is needed. The counterexample is a violated inequality for monotonicity, sub- and supermodularity, a nonzero least-squares residual for modularity and degree two, and an infeasible linear program for MTM (Appendix C.3). Assuming exact evaluations of f, every property is rejected on every utility.

Closeness to a property does not make its method the best. To measure how close each domain comes, we assign each property a score of 1 exactly when it holds. For the inequalities (monotone, sub- and supermodular), it is the fraction of sampled subsets and players that satisfy them. Unlike the worst-case submodularity ratio (Das & Kempe, 2018), our submodularity score carries no greedy guarantee, though that guarantee requires monotone f anyway. Modularity and degree two are quantified by the share of the variance of f explained by the best least-squares fit of that degree, and MTM by the fraction of held-out pairs of subsets whose order is reproduced by a single fitted weight vector. Figure 2 (left) reports these scores, and no domain reaches 1 on any property. Figure 2 (right) compares the methods on each property, weighting every utility by its score, and marks the method optimal under that property. Even for utilities closest to a property, that method is not the strongest, yet the gap to S-LFBO gets narrower, as shown in Figure 13.

3.1 S-LFBO for Subset Selection

LFBO learns f from evaluations of this functions without assuming any property. However, this methods has been designed and mostly used for hyperparameter search over continuous spaces, whereas our search space consists of the 2^d subsets of $[d]$, with d up to several thousands, of which at most 100 can be evaluated. One run of LFBO consists of four steps:

1. **Design:** choose the initial subsets and evaluate them.

2. **Surrogate:** set τ to a quantile of the stored values and fit a classifier to the improvement target a by minimizing Equation (2).
3. **Maximization:** find a subset that maximizes the classifier as in Equation (4), evaluate it, and add it to the stored subsets.
4. **Repeat** steps 2 and 3 until the budget is spent.

We introduce **Selection LFBO (S-LFBO)**, which adapts LFBO to subset selection, outlined in Figure 1; further details can be found in Appendix E.2.

S-LFBO vs. LFBO. We make three modifications to LFBO (Figure 1). First, LFBO draws subsets uniformly, so their sizes concentrate around $d/2$, while the best subsets in our utilities vary widely in size. S-LFBO instead draws a size uniformly, then a uniform subset of that size. Second, LFBO's surrogate is typically a neural network or off-the-shelf gradient-boosted trees (Song et al., 2022). S-LFBO boosts stumps for the additive part, then deeper trees on the residual, a common strategy that models interactions only where the additive part falls short (Lou et al., 2013). Third, LFBO maximizes by scoring random candidates that, for large d, lie far from every evaluated subset, in regions where the classifier has seen no data. S-LFBO instead runs simulated annealing over neighboring subsets (one player added or removed), starting chains from the best subset so far and the previous proposal, so the search stays near evaluated subsets. Figure 9 ablates each change.

3.2 Learning the Improvement Target

LFBO learns the improvement target a, so a surrogate that works well for f must also work well for a. This raises the question of whether a useful structure of f carries over to a. For a fixed threshold τ, the target is

$$a(S) = [f(S) - \tau]_+. \tag{5}$$

Subsets below the threshold all receive 0, while subsets above it retain their ordering and differences in value. Section 2.2 explains how weighted classification learns this target. We first check the usual properties, then the structure of a single tree and tree ensembles, with proofs in Appendix A.2.

Properties and Fourier Spectrum. Table 1 summarizes what happens to each property. Monotonicity and MTM survive, but modularity, submodularity, and a low degree can all be lost, while supermodularity needs monotonicity as well. Any utility, such as f or a, can be written as $a(S) = \sum_{T \subseteq [d]} \widehat{a}(T)(-1)^{|S \cap T|}$, with one Fourier coefficient per interaction T (O'Donnell, 2014). Surrogates such as additive fits, or BOCS, approximate a utility well only when its Fourier spectrum is concentrated on low-degree interactions, i.e., when most of the mass $\sum_T \widehat{a}(T)^2$ lies on small $|T|$. Below, we provide four propositions that characterize how thresholding affects the structure a surrogate must learn.

Proposition 1 (Fourier density)**.** *Let $q_\tau > 0$ be the fraction of subsets with $f(S) > \tau$. Then a has at least $1/q_\tau$ nonzero Fourier coefficients. In particular, if f exceeds τ on a single subset, all 2^d coefficients are nonzero, even if f is modular.*

Raising τ decreases q_τ and thus increases the bound, so the more the target concentrates on the best subsets, the more coefficients such surrogates need.

Proposition 2 (Low-degree Concentration)**.** *Let $a_{\leq k}$ denote the Fourier truncation of a to degree at most k. Define the population coefficient of determination $R_k^2 = 1 - \frac{\|a - a_{\leq k}\|_2^2}{\mathrm{Var}(a)}$, assuming $\mathrm{Var}(a) > 0$. Then*

$$R_k^2 \geq 1 - \frac{\sum_{T \subseteq [d]} |T| \, \widehat{f}(T)^2}{(k+1)\,\mathrm{Var}(a)}.$$

This bound helps us understand when a low-degree approximation of $a(S)$ is effective. First, $f(S)$ should itself be concentrated on low-degree interactions, a property observed across a range of machine learning applications (Gorji et al., 2025; Kang et al., 2025). Second, the threshold τ should not be so large that thresholding removes most of the variation in $f(S)$, since $\mathrm{Var}([f(S) - \tau]_+)$ decreases as τ increases.

We now consider the simple case of a single tree. Unlike Fourier coefficients, the structure of a tree is inherited by a directly. Thus, it keeps its splits exactly, and changing only its leaf values turns an approximation of f into an approximation of a with no greater error.

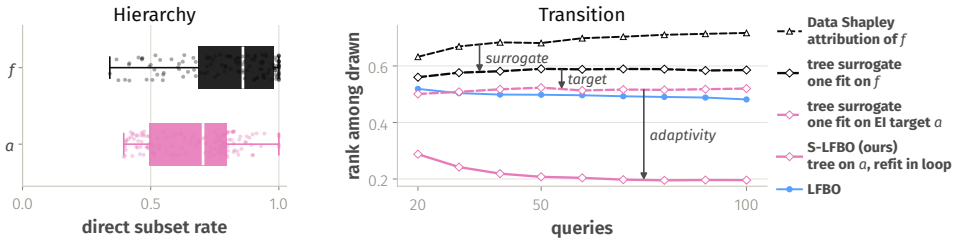

Figure 3: Each choice in S-LFBO adds to the previous one. Hierarchy: direct subset rate of f and a. Target: fitting a instead of f. Surrogate: tree ensemble against the additive Data Shapley fit. Adaptivity: static against adaptive refitting. The adaptive tree loop on a, S-LFBO, performs best.

Proposition 3 (Tree structure)**.** *If f is represented by a decision tree, then $a = [f - \tau]_+$ is represented by the same tree, with each leaf value v replaced by $[v - \tau]_+$ (and subtrees whose leaves all become zero pruned into a single leaf). Moreover, for any function g and every subset S,*

$$\left|a(S) - [g(S) - \tau]_+\right| \leq |f(S) - g(S)|.$$

Consequently, if a tree approximates f, transforming its leaf values gives a tree of the same size that approximates a with no greater pointwise error.

The guarantee in Proposition 3 does not carry over from individual trees to an ensemble, because clipping a sum is different from summing the clipped values. Instead, it applies to the *cells* on which the ensemble is constant, i.e., the subsets reaching the same leaf in every tree.

Proposition 4 (Tree ensembles)**.** *Let f be a tree ensemble, and let N cells satisfy $f > \tau$, where cell i fixes whether each of k_i players is in S. Then $a = f - \tau$ on these cells and $a = 0$ on all others, and N is non-increasing in τ. Moreover, a is represented by a tree with at most $1 + \sum_{i=1}^{N} k_i$ leaves.*

A single cell fixing k players thus costs a tree $k + 1$ leaves, but at least 2^k Fourier coefficients by Proposition 1. These results do not guarantee that f has such a structure or that a tree learns a from a few queries, so we investigate both empirically.

Empirically, f is hierarchical and a keeps much of this structure. Tree ensembles are well suited to functions whose interactions are sparse and hierarchical, and Butler et al. (2025) measure hierarchy by the direct subset rate, which quantifies how often removing one player from a dominant Fourier interaction leaves a dominant interaction: $\mathrm{DSR}(g) = \frac{\sum_{T \in \mathcal{F}(g),\, |T| \geq 2} \sum_{i \in T} \mathbf{1}\{T \setminus \{i\} \in \mathcal{F}(g)\}}{\sum_{T \in \mathcal{F}(g),\, |T| \geq 2} |T|}$, where the dominant interactions $\mathcal{F}(g)$ form the smallest set of non-empty interactions carrying at least 95% of the Fourier energy. Figure 3 (left) shows that f is strongly hierarchical and a somewhat less, on all our benchmarks with $d \leq 16$, where Fourier coefficients can be computed exactly. As highlighted in the introduction, fitting f makes the tree learn irrelevant information. We validate this empirically (Figure 3, right), where a tree fitted to a finds better subsets than one fitted to f, yet both outperform Data Shapley. The largest gain comes from refitting adaptively, which yields S-LFBO.

4 Experiments

We compare S-LFBO with domain methods and Bayesian optimization on 321 utilities from the four domains, at both equal query budget (Section 4.1) and wall clock (Section 4.2). Additionally we showcase the optimization mechanism of S-LFBO in Section 4.3.

Utilities. Feature and data selection use the TabArena datasets (Erickson et al., 2025), with LightGBM (Ke et al., 2017) and TabPFNv3 (Grinsztajn et al., 2026) as the refitted model, and reach up to $d = 1{,}776$ features and $d = 75{,}000$ training points. Ensemble pruning selects among 16 members of a LightGBM ensemble, and circuit discovery consists of $d = 53$ or $d = 117$ edges coming from 7 InterpBench transformers (Gupta et al., 2024). Each utility is run with up to 10 seeds at budgets of 20 to 100 queries, and refer to Appendix C.2 for the complete overview.

Methods. Our benchmark contains 24 methods, though not all can be run across all domains, e.g., ACDC (Conmy et al., 2023) runs only in circuit discovery. They comprise search methods, attri-

Table 2: Mean rank percentile (lower is better). Share of other methods that find a better subset, ties broken by runtime (Appendix C.4), shadings mark the best methods. Subscripts give the standard error against S-LFBO, stars a Wilcoxon $p < .05$ (*) or $< .01$ (**), parentheses a partial utility set, "–" not applicable, and *others* the mean over 13 further methods.

	feature selection			data selection			ensemble pruning			circuit discovery		
	90 utilities			173 utilities			51 utilities			7 utilities		
query budget b	20	50	100	20	50	100	20	50	100	20	50	100
Ours												
S-LFBO	$\mathbf{.196}_{\pm.012}$	$\mathbf{.191}_{\pm.014}$	$\mathbf{.171}_{\pm.015}$	$\mathbf{.160}_{\pm.009}$	$\mathbf{.146}_{\pm.008}$	$\mathbf{.144}_{\pm.009}$	$.271_{\pm.016}$	$\mathbf{.137}_{\pm.012}$	$\mathbf{.178}_{\pm.010}$	$\mathbf{.222}_{\pm.025}$	$.109_{\pm.024}$	$.113_{\pm.025}$
Domain methods												
LightGBM Gini	$(.465^{**}_{\pm.042})$	$(.666^{**}_{\pm.042})$	$(.760^{**}_{\pm.040})$	–	–	–	–	–	–	–	–	–
Data Shapley	$.376^{**}_{\pm.021}$	$.451^{**}_{\pm.024}$	$.515^{**}_{\pm.027}$	$.313^{**}_{\pm.015}$	$.401^{**}_{\pm.019}$	$.445^{**}_{\pm.021}$	$.699^{**}_{\pm.025}$	$.778^{**}_{\pm.029}$	$.810^{**}_{\pm.029}$	$.398^{*}_{\pm.055}$	$.503^{*}_{\pm.048}$	$.575^{*}_{\pm.043}$
NASH	$.412^{**}_{\pm.025}$	$.574^{**}_{\pm.029}$	$.637^{**}_{\pm.032}$	$(.466^{**}_{\pm.028})$	$(.598^{**}_{\pm.034})$	$(.617^{**}_{\pm.035})$	$.708^{**}_{\pm.035}$	$.803^{**}_{\pm.028}$	$.843^{**}_{\pm.024}$	$.400^{*}_{\pm.053}$	$.518^{*}_{\pm.052}$	$.579^{*}_{\pm.036}$
greedy forward	$.802^{**}_{\pm.027}$	$.697^{**}_{\pm.036}$	$.657^{**}_{\pm.038}$	$.834^{**}_{\pm.015}$	$.762^{**}_{\pm.019}$	$.778^{**}_{\pm.020}$	$.544^{**}_{\pm.057}$	$.253_{\pm.041}$	$.246_{\pm.034}$	$.773^{*}_{\pm.034}$	$.822^{*}_{\pm.027}$	$.838^{*}_{\pm.024}$
ACDC	–	–	–	–	–	–	–	–	–	$.396^{*}_{\pm.056}$	$.488^{*}_{\pm.056}$	$.375^{*}_{\pm.110}$
PEP	$.616^{**}_{\pm.024}$	$.474^{**}_{\pm.031}$	$.402^{**}_{\pm.034}$	$.590^{**}_{\pm.015}$	$.510^{**}_{\pm.015}$	$.415^{**}_{\pm.021}$	$.681^{**}_{\pm.028}$	$.582^{**}_{\pm.026}$	$.239^{**}_{\pm.017}$	$.727^{*}_{\pm.059}$	$.740^{*}_{\pm.055}$	$.761^{*}_{\pm.055}$
Black-box optimisers												
GP-BO	$.235^{*}_{\pm.017}$	$.215^{**}_{\pm.011}$	$.206^{**}_{\pm.008}$	$.222^{**}_{\pm.013}$	$.165^{*}_{\pm.010}$	$.150_{\pm.009}$	$\mathbf{.253}_{\pm.020}$	$.145_{\pm.011}$	$.220^{**}_{\pm.009}$	$.269_{\pm.103}$	$\mathbf{.076}_{\pm.038}$	$\mathbf{.074}_{\pm.016}$
SMAC	$.414^{**}_{\pm.020}$	$.422^{**}_{\pm.016}$	$.384^{**}_{\pm.012}$	$(.387^{**}_{\pm.020})$	$(.462^{**}_{\pm.018})$	$(.415^{**}_{\pm.015})$	$.427^{**}_{\pm.025}$	$.471^{**}_{\pm.016}$	$.457^{**}_{\pm.014}$	$.260_{\pm.062}$	$.232_{\pm.065}$	$.201_{\pm.038}$
BOCS	$(.451^{**}_{\pm.022})$	$(.387^{**}_{\pm.016})$	$(.368^{**}_{\pm.014})$	$(.471^{**}_{\pm.020})$	$(.313^{**}_{\pm.014})$	$(.310^{**}_{\pm.012})$	$.556^{**}_{\pm.016}$	$.327^{**}_{\pm.015}$	$.318^{**}_{\pm.010}$	$.315_{\pm.093}$	$.313^{*}_{\pm.054}$	$.293^{*}_{\pm.026}$
LFBO	$.394^{**}_{\pm.024}$	$.412^{**}_{\pm.024}$	$.416^{**}_{\pm.021}$	$.358^{**}_{\pm.013}$	$.387^{**}_{\pm.013}$	$.389^{**}_{\pm.013}$	$.315^{*}_{\pm.021}$	$.303^{**}_{\pm.016}$	$.384^{**}_{\pm.012}$	$.349_{\pm.055}$	$.146_{\pm.036}$	$.232^{*}_{\pm.014}$
BORE	$.363^{**}_{\pm.022}$	$.410^{**}_{\pm.025}$	$.414^{**}_{\pm.021}$	$.340^{**}_{\pm.014}$	$.389^{**}_{\pm.015}$	$.388^{**}_{\pm.015}$	$.369^{**}_{\pm.020}$	$.395^{**}_{\pm.016}$	$.484^{**}_{\pm.015}$	$.315_{\pm.070}$	$.257^{*}_{\pm.049}$	$.280^{*}_{\pm.043}$
others (13)	$.519^{(13)}_{\pm.003}$	$.526^{(13)}_{\pm.003}$	$.530^{(13)}_{\pm.004}$	$.535^{(13)}_{\pm.002}$	$.533^{(13)}_{\pm.003}$	$.537^{(13)}_{\pm.002}$	$.496^{(13)}_{\pm.003}$	$.534^{(13)}_{\pm.003}$	$.538^{(13)}_{\pm.002}$	$.551^{(12)}_{\pm.009}$	$.560^{(12)}_{\pm.009}$	$.549^{(12)}_{\pm.008}$

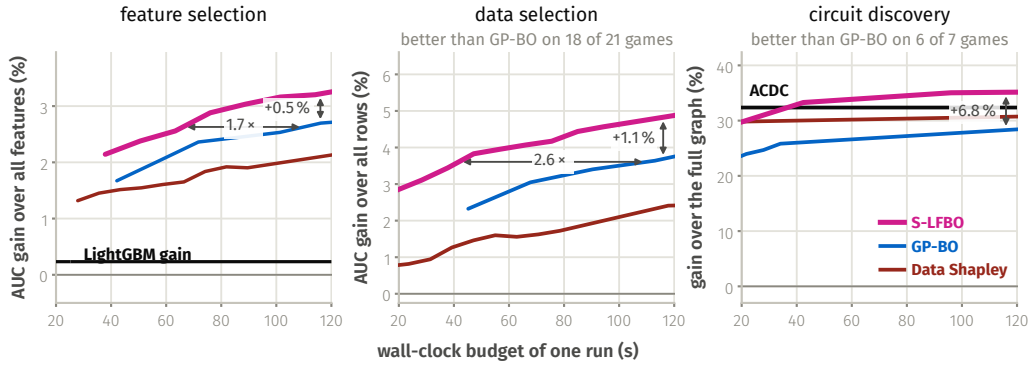

Figure 4: Where two minutes of wall clock get each method. Relative gain over using all features/datapoints/edges, having 20 to 120 seconds of compute time. Black: domain's one-shot method. Further details in Appendix C.4.

bution methods such as Data Shapley (Ghorbani & Zou, 2019) and NASH (Tian et al., 2026), and Bayesian optimization: BOCS (Baptista & Poloczek, 2018), BORE (Tiao et al., 2021), LFBO (Song et al., 2022), SMAC (Lindauer et al., 2022), COMBO (Oh et al., 2019), and GP-BO, a powerful Gaussian process configuration found by Dreczkowski et al. (2023). Every Bayesian optimization method, including S-LFBO, starts with 10 initial subsets. Details can be found in Appendix D.

Metrics. On each utility, a method's *rank percentile* is its rank among the methods run on that utility, with lower values indicating better performance, and ties in value are decided by runtime. We also measure the relative improvement $(f(S) - f([d]))/|f([d])|$ of the selected subsets S.

4.1 Performance across domains

S-LFBO ranks first in 9 of 12 columns of Table 2, beating GP-BO ($p < .01$) at every budget in feature and data selection. GP-BO leads ensemble pruning at 20 queries and circuit discovery at 50 and 100, substantially only at 100. S-LFBO, GP-BO, and all other Bayesian optimization methods outrank every attribution method, at least at 100 queries. Where S-LFBO and GP-BO are level, S-LFBO is much faster at equal wall clock (Figure 4). The gains over the full set at 100 queries range from 2–3% for both S-LFBO and GP-BO, versus 1–1.6% for Data Shapley (Appendix F.2).

4.2 Performance at equal wall clock

We now consider the runtime costs: TabPFN's data selection incurs higher runtime than a circuit forward pass, so each method's overhead matters more when evaluations are cheap. Figure 4 therefore gives every method the same wall-clock budget, 20 to 120 s. S-LFBO outperforms GP-BO at every time in all three domains, including circuit discovery, where GP-BO leads at equal query budgets. After 120 s, S-LFBO's subsets improve on the full set by 2.9% in feature and 4.4% in data selection, versus 2.5% and 3.4% for GP-BO and 1.9% and 2.2% for Data Shapley, and reach GP-BO's results 1.7× and 2.6× faster. The runtime difference stems from fitting the Gaussian process, which scales cubically in the number of evaluations, whilst a tree ensemble remains nearly linear.

4.3 S-LFBO Optimization Search Path

Figure 5 shows the subsets that the two adaptive methods evaluate. GP-BO's genetic optimizer starts at the incumbent, and its expected improvement peaks near evaluated subsets, so consecutive queries differ in 21 to 24% of features at every $d > 16$. Instead, S-LFBO anneals a tree ensemble, i.e., a flip changes the score only for the features the trees split on, whilst all others are unaffected. Up to $d = 60$, the booster splits on 94% of the features and S-LFBO settles sooner than GP-BO. From $d = 61$ to 300 it splits on 83%, and consecutive queries differ in 33% of the features against GP-BO's 24%. Above $d = 300$, the booster splits on 9% of the features, five chains of 256 flips reach only a tenth of the cube, and S-LFBO stays near its design. Further examples can be found in Figure 14.

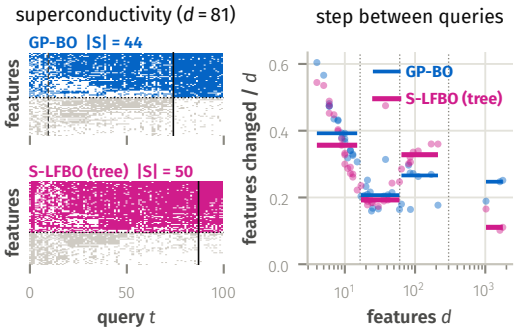

Figure 5: S-LFBO's Optimization Path. Left: queries (columns) by features (rows), filled where selected, with final pick in color. Removed features in grey, the final pick as a vertical line. Right: share of features changed per query.

5 Conclusion

Machine learning tasks, including feature selection, data selection, ensemble pruning, and circuit discovery, all aim to solve a subset selection problem, yet their standard methods rely on structural properties that, as we show, none of our 321 utilities satisfy. We propose measures to quantify the violation of properties and find that methods specifically designed for them fall behind adaptive methods as the number of queries increases. Driven by this insight, we find that the underlying structure is often hierarchical and sparse, hence favorable for tree surrogates in an adaptive loop. We propose S-LFBO, a tailored likelihood-free Bayesian optimization method (Song et al., 2022) built on utility analysis. Based on an extensive benchmark across the four domains, covering up to 23 baselines, we find S-LFBO to outperform in 3 of the 4 domains and be second in the other. Our experiments shows that S-LFBO yields better subsets than state-of-the-art Bayesian optimization.

Limitations and Future Work. Our analysis has primarily focused on tabular datasets, using two oracle models: LightGBM and TabPFNv3. As such, the interaction structure may not hold for other models. But as shown in Figure 2, S-LFBO performs well across the range of structures in our benchmark, and the properties and spectra we provide can diagnose a new domain before any method is run. Like all Bayesian optimization approaches, S-LFBO is driven by the choice of the surrogate, but as our analysis has found, tree ensembles seem a reasonable choice for the selection problem in the four domains. An investigation into whether this would hold across additional domains offers an interesting avenue for future work. Finally, we treat f as a black box, although the oracle is usually known. Gradients of a network, the importances of a fitted model, or per-example losses are available at little extra cost, and circuit discovery already uses them (Syed et al., 2024). While we consider such grey-box signals promising for shaping the initial design or the surrogate, the underlying problem of finding the best subset remains a selection problem of our framework.

Statement on AI use. In this work, we used LLM assistants for six purposes: polishing the text (writing); searching related work, where each cited reference was read and verified; exploring candidate methods and experiments; writing experiment code, which the authors reviewed and tested; turning our notes into draft sections of the paper; and drafting, checking proof, and verifying every proof in the appendix. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Andisheh Amrollahi, Amir Zandieh, Michael Kapralov, and Andreas Krause. Efficiently learning fourier sparse set functions. In *NeurIPS*, pp. 15094–15103, 2019.

Wenruo Bai and Jeffrey A. Bilmes. Greed is still good: Maximizing monotone submodular+supermodular (BP) functions. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 314–323, 2018.

Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 471–480, 2018.

Shahaf Bassan, Xuanxiang Huang, and Guy Katz. Unifying formal explanations: A complexity-theoretic perspective. In *ICLR*, 2026.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. In *NeurIPS*, 2024.

Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008.

Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discret. Appl. Math.*, 123(1-3): 155–225, 2002.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM J. Comput.*, 44(5):1384–1402, 2015.

Landon Butler, Abhineet Agarwal, Justin Singh Kang, Yigit Efe Erginbas, Bin Yu, and Kannan Ramchandran. Proxyspex: Inference-efficient interpretability via sparse feature interactions in llms. In *NeurIPS*, 2025.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *ICML*, volume 69 of *ACM International Conference Proceeding Series*, 2004.

Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 882–891, 2018.

Hongliang Chi, Qiong Wu, Zhengyi Zhou, Jonathan Light, Emily Dodwell, and Yao Ma. Unifying and optimizing data values for selection via sequential decision-making. In *ICML*, 2026.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. Contextcite: Attributing model generation to context. In *NeurIPS*, 2024.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *NeurIPS*, 2023.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation using linear regression. In *AISTATS*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3457–3465, 2021.

Abhimanyu Das and David Kempe. Algorithms for subset selection in linear regression. In *STOC*, pp. 45–54, 2008.

Abhimanyu Das and David Kempe. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. *J. Mach. Learn. Res.*, 19:3:1–3:34, 2018.

A. Philip Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Scalable combinatorial bayesian optimization with tractable statistical models. *CoRR*, abs/2008.08177, 2020.

David L Donoho and Philip B Stark. Uncertainty principles and signal recovery. *SIAM J. Appl. Math.*, 49:906–931, 1989.

Kamil Dreczkowski, Antoine Grosnit, and Haitham Bou-Ammar. Framework and benchmarks for combinatorial and mixed-variable Bayesian optimization. In *NeurIPS*, 2023.

Pradeep Dubey, Abraham Neyman, and Robert James Weber. Value theory without efficiency. *Math. Oper. Res.*, 6(1):122–128, 1981.

Ethan R. Elenberg, Rajiv Khanna, Alexandros G. Dimakis, and Sahand N. Negahban. Restricted strong convexity implies weak submodularity. *CoRR*, abs/1612.00804, 2016.

Yigit Efe Erginbas, Justin Singh Kang, Elizabeth Polito, and Kannan Ramchandran. Adaptive sparse möbius transforms for learning polynomials. *CoRR*, abs/2602.06246, 2026.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. In *NeurIPS*, 2025.

Ulrich Faigle and Michel Grabisch. Least square approximations and linear values of cooperative games. In *Algebraic Techniques and Their Use in Describing and Processing Uncertainty*, volume 878 of *Studies in Computational Intelligence*, pp. 21–32. Springer, 2020.

Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153, 2011.

Daniel Vidali Fryer, Inga Strümke, and Hien D. Nguyen. Shapley values for feature selection: The good, the bad, and the axioms. *IEEE Access*, 9:144352–144360, 2021.

Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games Econ. Behav.*, 55(1):72–99, 2006.

Amirata Ghorbani and James Y. Zou. Data shapley: Equitable valuation of data for machine learning. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251, 2019.

Ali Gorji, Andisheh Amrollahi, and Andreas Krause. SHAP values via sparse fourier representation. In *NeurIPS*, 2025.

Michel Grabisch and Marc Roubens. An axiomatic approach to the concept of interaction among players in cooperative games. *Int. J. Game Theory*, 28(4):547–565, 1999.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-3: Technical report. *CoRR*, abs/2605.13986, 2026.

Rohan Gupta, Iván Arcuschin Moreno, Thomas Kwa, and Adrià Garriga-Alonso. Interpbench: Semi-synthetic transformers for evaluating mechanistic interpretability techniques. In *NeurIPS*, 2024.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling*, 2024.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nat.*, 637(8044):319–326, 2025.

Yuzheng Hu, Pingbang Hu, Han Zhao, and Jiaqi W. Ma. Most influential subset selection: Challenges, promises, and beyond. In *NeurIPS*, 2024.

Jenny Y. Huang, David R. Burt, Yunyi Shen, Tin D. Nguyen, and Tamara Broderick. Approximations to worst-case data dropping: unmasking failure modes. *Trans. Mach. Learn. Res.*, 2025.

Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pp. 1511–1519, 2019.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9525–9587, 2022.

Hiroshi Ishikawa. Transformation of general binary MRF minimization to the first-order case. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(6):1234–1249, 2011.

Justin Singh Kang, Yigit Efe Erginbas, Landon Butler, Ramtin Pedarsani, and Kannan Ramchandran. Learning to understand: Identifying interactions via the möbius transform. In *NeurIPS*, 2024.

Justin Singh Kang, Landon Butler, Abhineet Agarwal, Yigit Efe Erginbas, Ramtin Pedarsani, Bin Yu, and Kannan Ramchandran. SPEX: scaling feature interaction explanations for llms. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

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 *NeurIPS*, pp. 3146–3154, 2017.

Scott Kirkpatrick, D. Gelatt Jr., and Mario P. Vecchi. Optimization by simulated annealing. *Sci.*, 220(4598):671–680, 1983.

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

Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *AISTATS*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8780–8802, 2022.

Zhendong Lei, Shaowei Cai, Chuan Luo, and Holger H. Hoos. Efficient local search for pseudo boolean optimization. In *SAT*, volume 12831 of *Lecture Notes in Computer Science*, pp. 332–348, 2021.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne M. VanBriesen, and Natalie S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pp. 420–429, 2007.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *J. Mach. Learn. Res.*, 23:54:1–54:9, 2022.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *KDD*, pp. 623–631, 2013.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pp. 4765–4774, 2017.

Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2): 245–259, 2009.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, pp. 234–243. Springer, 1978.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, pp. 1812–1818, 2015.

Velibor V. Misic. Optimization of tree ensembles. *Oper. Res.*, 68(5):1605–1624, 2020.

Maximilian Muschalik, Hubert Baniecki, Fabian Fumagalli, Patrick Kolpaczki, Barbara Hammer, and Eyke Hüllermeier. shapiq: Shapley interactions for machine learning. In *NeurIPS*, 2024.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

ChangYong Oh, Jakub M. Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian optimization using the graph cartesian product. In *NeurIPS*, pp. 2910–2920, 2019.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: attributing model behavior at scale. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27074–27113, 2023.

Pavel Pudil, Jana Novovicová, and Josef Kittler. Floating search methods in feature selection. *Pattern Recognit. Lett.*, 15(10):1119–1125, 1994.

Chao Qian, Yang Yu, and Zhi-Hua Zhou. Pareto ensemble pruning. In *AAAI*, pp. 2935–2941, 2015a.

Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. In *NeurIPS*, pp. 1774–1782, 2015b.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, pp. 1527–1535, 2018.

Ivo G. Rosenberg. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'Études de Recherche Opérationnelle*, 17(1):71–74, 1975.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proc. IEEE*, 104(1):148–175, 2016.

Jiaming Song, Lantao Yu, Willie Neiswanger, and Stefano Ermon. A general recipe for likelihood-free Bayesian optimization. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20384–20404, 2022.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In *NeurIPS*, 2022.

Peter Stobbe and Andreas Krause. Learning fourier sparse set functions. In *AISTATS*, volume 22 of *JMLR Proceedings*, pp. 1125–1133, 2012.

Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. The shapley taylor interaction index. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9259–9268, 2020.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *The 7th BlackboxNLP Workshop*, 2024.

Xiao Tian, Jue Fan, Rachael Hwee Ling Sim, Zixuan Wang, Nancy F. Chen, and Bryan Kian Hsiang Low. Is data shapley not better than random in data selection? ask NASH. In *ICML*, 2026.

Louis C. Tiao, Aaron Klein, Matthias W. Seeger, Edwin V. Bonilla, Cédric Archambeau, and Fabio Ramos. BORE: Bayesian optimization by density-ratio estimation. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10289–10300, 2021.

Che-Ping Tsai, Chih-Kuan Yeh, and Pradeep Ravikumar. Faith-shap: The faithful shapley interaction index. *J. Mach. Learn. Res.*, 24:94:1–94:42, 2023.

Jiachen T. Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pp. 6388–6421, 2023.

Jiachen T. Wang, Tianji Yang, James Zou, Yongchan Kwon, and Ruoxi Jia. Rethinking data shapley for data selection tasks: Misleads and merits. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 52033–52063, 2024.

Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. In *NeurIPS*, 2020.

Robert J. Weber. Probabilistic values for games. In *The Shapley Value: Essays in Honor of Lloyd S. Shapley*, pp. 101–119. Cambridge University Press, 1988. doi: 10.1017/CBO9780511528446.008.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10648–10659, 2020.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *ICLR*, 2024.

APPENDIX CONTENTS

A Proofs **16**

A.1 Properties under the target 16

A.2 Fourier density and trees 17

B Extended Prior Work **20**

C Experimental Details **21**

C.1 Domains and utilities 21

C.2 Datasets, splits and preprocessing 22

C.3 Measuring the properties 22

C.4 Ranking, ties and wall clocks 24

C.5 Hardware, software and compute 24

D Baseline Details **24**

D.1 Direct search on the oracle 25

D.2 Attribution and additive surrogates 26

D.3 Combinatorial Bayesian optimization 26

D.4 Domain-specific baselines 27

E S-LFBO Ablations **27**

E.1 The spectra of the utilities and of the target 27

E.2 Design, surrogate and maximizer 28

F Further Results **31**

F.1 Circuit discovery at other size prices 31

F.2 How large are the gains? 31

F.3 Does the lead depend on the size of the utility? 32

F.4 Does a property favour its own method? 32

F.5 Every query on eleven utilities 33

F.6 Per-utility results 33

A PROOFS

Notation. $f : 2^{[d]} \to \mathbb{R}$ is a set function on the ground set $[d]$, evaluated exactly; $w(S) = \sum_{i \in S} w_i$ for $w \in \mathbb{R}^d$, and $x_+ = \max(x, 0)$. The structural properties are those of Table 1. The Möbius coefficients $m(R) = \sum_{Q \subseteq R} (-1)^{|R|-|Q|} f(Q)$ are the unique ones with $f(S) = \sum_{R \subseteq S} m(R)$, so f has degree $\leq k$ exactly when $m(R) = 0$ for every $|R| > k$. In the Fourier basis $\chi_T(S) = (-1)^{|S \cap T|}$ and $\hat{p}(T) = \mathbb{E}_S\big[p(S)\chi_T(S)\big] = 2^{-d} \sum_S p(S)\chi_T(S)$ with S uniform on $2^{[d]}$, so that $p = \sum_T \hat{p}(T)\chi_T$ and $\mathbb{E}_S[p^2] = \sum_T \hat{p}(T)^2$, which we call the energy of p (O'Donnell, 2014); characters multiply by symmetric difference, $\chi_A \chi_B = \chi_{A \triangle B}$. The target of Equation (5) is $a = (f - \tau)_+$ with label $z = \mathbb{1}[f > \tau]$; we write $U = \{S : f(S) > \tau\}$ and $q_\tau = |U|/2^d$.

A.1 PROPERTIES UNDER THE TARGET

We prove the claims in Table 1, one property at a time.

Write $\varphi(x) = (x - \tau)_+$, so that $a = \varphi \circ f$, it holds $\varphi(x) \geq x - \tau$ and φ is non-decreasing.

Monotonicity and modularity. If f is monotone then $f(S \cup \{i\}) \geq f(S)$, and applying the nondecreasing φ to both sides gives $a(S \cup \{i\}) \geq a(S)$.

To show that φ does not preserve modularity, we construct an counterexample. Let $d = 2$, $f(S) = |S|$ and $\tau = \frac{1}{2}$, so that f is modular (and submodular), while a takes the values $0, \frac{1}{2}, \frac{1}{2}, \frac{3}{2}$ on $\emptyset, \{1\}, \{2\}, \{1, 2\}$. Its order-2 Möbius coefficient is $\frac{3}{2} - \frac{1}{2} - \frac{1}{2} + 0 = \frac{1}{2} \neq 0$, so a is not modular. □

MTM. If $f(S) = g\big(w(S)\big)$ with g nondecreasing, then $a(S) = \varphi\big(g(w(S))\big)$, and $\varphi \circ g$ is nondecreasing as a composition of nondecreasing maps. □

Submodularity. Use the modularity counterexample above, where f is also submodular. In particular for the triple $(\emptyset, 1, 2)$ the condition of submodularity fails, since $a(\{1\}) - a(\emptyset) = \frac{1}{2}$ is below $a(\{1, 2\}) - a(\{2\}) = 1$, so a is not submodular. □

Supermodularity. Let $S \subseteq T$ and $i \notin T$, and abbreviate $u = f(S)$, $u' = f(S \cup \{i\})$, $v = f(T)$ and $v' = f(T \cup \{i\})$. Supermodularity is equivalent to $v' - v \geq u' - u$ and monotonicity additionally has $v \geq u$, $u' \geq u$ and $v' \geq v$. Firstly observe that for any $y \geq x$ it holds

$$\varphi(y) - \varphi(x) = \big(y - \max(x, \tau)\big)_+,$$

which can easily verified for the three cases $y \leq \tau$, $x \geq \tau$ and $x < \tau < y$.

Supermodularity and monotonicity give

$$\begin{aligned} v' - \max(v, \tau) &\geq v + (u' - u) - \max(v, \tau) \\ &= u' - \max\big(u, \tau - (v - u)\big) \\ &\geq u' - \max(u, \tau). \end{aligned}$$

Taking positive parts preserves the inequality, so

$$\begin{aligned} \varphi(v') - \varphi(v) &= [v' - \max(v, \tau)]_+ \\ &\geq [u' - \max(u, \tau)]_+ \\ &= \varphi(u') - \varphi(u). \end{aligned}$$

These are the marginal gains of a at T and S, proving supermodularity.

Monotonicity cannot be dropped. For $d = 2$ take the modular, hence supermodular, f with $f(\emptyset) = 0$, $f(\{1\}) = -10$, $f(\{2\}) = 10$, $f(\{1, 2\}) = 0$ and $\tau = -1$, so that a takes the values $1, 0, 11, 1$ in the same order. With $S = \emptyset$, $T = \{2\}$ and $i = 1$ the required $a(\{1\}) - a(\emptyset) \leq a(\{1, 2\}) - a(\{2\})$ yields $-1 \leq -10$, violating supermodularity.

□

Degree. Fix $1 \le k < d$, take $f(S) = |S|$, of degree $1 \le k$, and put $\tau = d - \frac{1}{2}$. Then $f(S) < \tau$ for every $S \neq [d]$ and $f([d]) - \tau = \frac{1}{2}$, so $a = \frac{1}{2}\,\mathbb{1}[S = [d]]$, whose only non-zero Möbius coefficient is $m([d]) = \frac{1}{2}$. So a has degree $d > k$. □

A.2 Fourier density and trees

Write $\chi_T(S) = (-1)^{|S \cap T|}$, so that every set function satisfies $g = \sum_T \widehat{g}(T)\chi_T$ with $\widehat{g}(T) = 2^{-d} \sum_S g(S)\chi_T(S)$, and let $\operatorname{supp} g = \{S : g(S) \neq 0\}$ and $\operatorname{supp} \widehat{g} = \{T : \widehat{g}(T) \neq 0\}$.

Proposition 1 (Fourier density, restated). *Let $q_\tau > 0$ be the fraction of subsets with $f(S) > \tau$. Then a has at least $1/q_\tau$ nonzero Fourier coefficients. In particular, if f exceeds τ on a single subset, all 2^d coefficients are nonzero, even if f is modular.*

Proof. For any nonzero g with $M = \max_S |g(S)|$, attained at S^*, we have $|\widehat{g}(T)| \le 2^{-d} |\operatorname{supp} g|\, M$ and hence

$$M = \Big| \sum_T \widehat{g}(T)\chi_T(S^*) \Big| \le |\operatorname{supp} \widehat{g}|\, 2^{-d}\, |\operatorname{supp} g|\, M,$$

so $|\operatorname{supp} g| \cdot |\operatorname{supp} \widehat{g}| \ge 2^d$, following the argument of Donoho & Stark (1989) for the discrete Fourier transform. Since a is nonzero exactly on the $q_\tau 2^d$ subsets with $f(S) > \tau$, it follows that $|\operatorname{supp} \widehat{a}| \ge 1/q_\tau$. If f exceeds τ only on a single subset S°, then $\widehat{a}(T) = 2^{-d} a(S^\circ)\chi_T(S^\circ) \neq 0$ for all T. This case occurs for modular $f(S) = \sum_{i \in S} w_i$ with $w_i > 0$, which has at most $d + 1$ nonzero coefficients, when $f([d]) - \min_i w_i < \tau < f([d])$. □

Proposition 2 (Low-degree Concentration, restated). *Let $a_{\le k}$ denote the Fourier truncation of a to degree at most k. Define the population coefficient of determination $R_k^2 = 1 - \frac{\|a - a_{\le k}\|_2^2}{\operatorname{Var}(a)}$, assuming $\operatorname{Var}(a) > 0$. Then*

$$R_k^2 \ge 1 - \frac{\sum_{T \subseteq [d]} |T|\, \widehat{f}(T)^2}{(k+1)\operatorname{Var}(a)}.$$

Proof. Since $a_{\le k}$ is the Fourier truncation of a to degrees at most k, Parseval's identity gives

$$\|a - a_{\le k}\|_2^2 = \sum_{|T| > k} \widehat{a}(T)^2.$$

For every T with $|T| > k$, we have $|T| \ge k + 1$, and hence

$$\widehat{a}(T)^2 \le \frac{|T|}{k+1} \widehat{a}(T)^2.$$

Therefore,

$$\|a - a_{\le k}\|_2^2 \le \frac{1}{k+1} \sum_{T \subseteq [d]} |T|\, \widehat{a}(T)^2.$$

Now write

$$a = \phi_\tau \circ f, \qquad \phi_\tau(x) = [x - \tau]_+.$$

Since ϕ_τ is 1-Lipschitz, for every coordinate i,

$$|a(x) - a(x^{\oplus i})| \le |f(x) - f(x^{\oplus i})|.$$

Thus the Boolean Dirichlet energy cannot increase under the thresholding map, so

$$\sum_{T \subseteq [d]} |T|\, \widehat{a}(T)^2 \le \sum_{T \subseteq [d]} |T|\, \widehat{f}(T)^2.$$

Combining the two inequalities yields

$$\|a - a_{\leq k}\|_2^2 \leq \frac{1}{k+1} \sum_{T \subseteq [d]} |T| \, \widehat{f}(T)^2.$$

Finally, by the definition of R_k^2,

$$1 - R_k^2 = \frac{\|a - a_{\leq k}\|_2^2}{\mathrm{Var}(a)}.$$

Hence

$$R_k^2 \geq 1 - \frac{\sum_{T \subseteq [d]} |T| \, \widehat{f}(T)^2}{(k+1)\,\mathrm{Var}(a)}.$$

□

Proposition 3 (Tree structure, restated). *If f is represented by a decision tree, then $a = [f - \tau]_+$ is represented by the same tree, with each leaf value v replaced by $[v - \tau]_+$. Subtrees whose leaves all become zero can be pruned into a single leaf. Moreover, for any function g and every subset S,*

$$\big|a(S) - [g(S) - \tau]_+\big| \leq |f(S) - g(S)|.$$

Consequently, if a tree approximates f, transforming its leaf values gives a tree of at most the same size that approximates a with no greater pointwise error.

Proof. A subset descends through a tree by tests that do not depend on the leaf values. If S reaches a leaf of value v in the tree for f, then $f(S) = v$, and in the transformed tree it reaches the same leaf and reads $[v - \tau]_+ = a(S)$. A subtree whose leaves all read 0 represents the constant 0 and can be replaced by a single leaf. For the bound, $\varphi(x) = [x - \tau]_+$ is 1-Lipschitz: if $x, y \geq \tau$ then $|\varphi(x) - \varphi(y)| = |x - y|$; if $x, y \leq \tau$ it is 0; and if $y < \tau \leq x$ it is $x - \tau \leq x - y$. Setting $x = f(S)$ and $y = g(S)$ gives the claim. If g is represented by a tree, the first claim shows that the transformed tree represents $[g - \tau]_+$, and the bound shows that its error against a at every S is at most that of g against f. □

Cells of an ensemble. A tree ensemble $f = \sum_{k=1}^{K} T_k$ is constant on each *cell*, the set of subsets reaching the same leaf in every tree. A cell fixes, for each player on any of its K paths, whether that player is in S, and consists of the 2^{d-k} subsets agreeing with it if it fixes k players. Since the cells partition $2^{[d]}$, the N cells with $f > \tau$, fixing $k_1, \ldots, k_N$ players, give $q_\tau = \sum_{i=1}^{N} 2^{-k_i}$.

Lemma 1 (Stacked tree)**.** *A tree ensemble f is represented by a single decision tree whose leaves are exactly the cells of f, the leaf of a cell fixing k players lying at depth k.*

Proof. Insert a copy of T_2 below every leaf of T_1, adding that leaf's value to the copy's leaf values, and continue in the same way through T_K, so that a subset collects exactly the K leaf values summing to $f(S)$. Since every test asks whether a player is in S, a test of a player already tested higher on the same path has only one reachable branch; replacing such a node by its reachable child leaves the function unchanged. Afterwards, no path tests a player twice, and each remaining leaf is reached exactly by the subsets agreeing with all K paths above it. Hence the leaves are the cells of f, and the path to a cell tests each of its k fixed players exactly once. □

Proposition 4 (Tree ensembles, restated). *Let f be a tree ensemble, and let N cells satisfy $f > \tau$, where cell i fixes whether each of k_i players is in S. Then $a = f - \tau$ on these cells and $a = 0$ on all others, and N is non-increasing in τ. Moreover, a is represented by a tree with at most $1 + \sum_{i=1}^{N} k_i$ leaves.*

Proof. Since f is constant on every cell, $a = [f - \tau]_+$ equals $f - \tau$ on cells with $f > \tau$ and 0 on all others. Raising τ can only remove cells from those with $f > \tau$, so N is non-increasing. For the tree, transform the leaf values of the stacked tree of Lemma 1 by Proposition 3, and collapse every maximal subtree without a leaf of a cell above τ into a single leaf of value 0. Every remaining internal node lies on the path to a cell above τ, and the path to cell i has k_i internal nodes, so there are at most $\sum_i k_i$ internal nodes. A full binary tree has one more leaf than internal nodes, which gives the bound; if $N = 0$, the tree is a single leaf of value 0. □

Remark 1. *For a single cell fixing k players both bounds are tight. Then a is a multiple of the cell's indicator $\prod_i \frac{1}{2}(1 \pm \chi_{\{i\}})$ over its k fixed players, which expands into exactly 2^k nonzero Fourier coefficients. Any tree for a needs a leaf contained in the cell, since a is nonzero on the cell and zero outside, hence at depth at least k, so it has at least $k + 1$ leaves.*

Table 3: Baselines measured in the benchmark. value oracle does not expose.

Group	Method	Reference
Direct search on the oracle	Random search	Bergstra & Bengio (2012)
	Hill climbing	Kohavi & John (1997)
	Forward–backward selection	Pudil et al. (1994)
	Simulated annealing	Kirkpatrick et al. (1983)
	Greedy forward	Nemhauser et al. (1978)
	Lazy greedy	Minoux (1978); Leskovec et al. (2007)
	Stochastic greedy	Mirzasoleiman et al. (2015)
	Static / adaptive greedy	Hu et al. (2024)
	Double greedy	Buchbinder et al. (2015)
	GreedMax	Bai & Bilmes (2018)
	POSS	Qian et al. (2015b)
	PEP	Qian et al. (2015a)
	Anchors	Ribeiro et al. (2018)
Attribution and additive surrogates	Data Shapley	Ghorbani & Zou (2019)
	Data Banzhaf	Wang & Jia (2023)
	NASH	Tian et al. (2026)
Combinatorial Bayesian optimization	BOCS	Baptista & Poloczek (2018)
	Min-cut surrogate†	Boros & Hammer (2002)
	LFBO	Song et al. (2022)
	BORE	Tiao et al. (2021)
	COMBO	Oh et al. (2019)
	GP-BO	Dreczkowski et al. (2023)
	SMAC	Lindauer et al. (2022)
	LA-MCTS	Wang et al. (2020)
Domain-specific baselines	ACDC	Conmy et al. (2023)
	LightGBM gain importance	Ke et al. (2017)

B Extended Prior Work

We place the compared methods in their literature, ordered by how expressive a surrogate of f they commit to, see Table 3 for an overview. Table 1 defines the structural properties and Appendix D describes each method.

Search without a surrogate. Greedy forward selection adds the best element at each step, and lazy evaluation cuts its query count (Minoux, 1978; Leskovec et al., 2007), and double greedy is its unconstrained counterpart for submodular f (Buchbinder et al., 2015). POSS replaces the greedy chain by a bi-objective evolutionary search over value and size (Qian et al., 2015b), and simulated annealing (Kirkpatrick et al., 1983) drops the structural assumption altogether. The submodularity ratio extends greedy's guarantee to weakly submodular f (Das & Kempe, 2018; Elenberg et al., 2016), but its constant is unknown for a given utility. In data dropping, scoring points by influence fails once they interact, and greedy re-scoring after every pick is the approximation that holds up (Hu et al., 2024; Huang et al., 2025).

Attribution and additive surrogates. Semivalues score an element by a weighted average of its marginal contributions (Dubey et al., 1981; Weber, 1988); Data Shapley (Ghorbani & Zou, 2019), Data Banzhaf (Wang & Jia, 2023) and Beta Shapley (Kwon & Zou, 2022) differ only in the weights, and much of the literature estimates them from few queries (Lundberg & Lee, 2017; Covert & Lee, 2021). Datamodels, TRAK, and ContextCite fit the same degree-one object by regression or gradients (Ilyas et al., 2022; Park et al., 2023; Cohen-Wang et al., 2024), and every linear value solves a weighted least-squares problem (Faigle & Grabisch, 2020), so the whole family is one surrogate class. Using such scores for selection has been questioned, both axiomatically (Fryer et al., 2021) and as a myopic approximation of a sequential decision (Chi et al., 2026).

Polynomial surrogates and sparse spectra. Interaction indices extend attribution from elements to subsets (Grabisch & Roubens, 1999; Fujimoto et al., 2006; Sundararajan et al., 2020; Tsai et al., 2023; Muschalik et al., 2024). In the Fourier or Möbius basis, estimating them is sparse recovery,

with query counts that scale with the number of coefficients instead of 2^d (Stobbe & Krause, 2012; Amrollahi et al., 2019; Kang et al., 2024; Erginbas et al., 2026; Kang et al., 2025; Butler et al., 2025; Gorji et al., 2025). A recovered polynomial is maximized by pseudo-Boolean optimization (Boros & Hammer, 2002): quadratization to a QUBO (Rosenberg, 1975; Ishikawa, 2011), integer programming (Butler et al., 2025), or variable elimination on a junction tree (Dawid, 1992; Bodlaender & Koster, 2008); tree ensembles are maximized exactly by mixed-integer programming (Misic, 2020). These query counts grow with the sparsity and with d, beyond the budgets of Section 4.

Combinatorial Bayesian optimization. BOCS fixes a degree-two surrogate and maximizes a Thompson draw (Baptista & Poloczek, 2018); COMBO replaces it by a Gaussian process with a diffusion kernel on the combinatorial graph (Oh et al., 2019); and Deshwal et al. (2020) relax the acquisition to a submodular problem solved by a minimum cut. Dreczkowski et al. (2023) unify these methods in one framework, from which we take the Hamming-kernel Gaussian process run as GP-BO. The family targets tens of variables and a few hundred evaluations, and its cubic refitting cost is what Section 4.2 prices.

Likelihood-free Bayesian optimization. BORE replaces the posterior by a classifier of good against bad observations (Tiao et al., 2021), and LFBO weights that classifier so that it recovers expected improvement (Song et al., 2022), the target we adopt. Both were developed for continuous hyperparameter spaces with generic classifiers and random-candidate maximizers; on the Boolean cube S-LFBO chooses the classifier and the maximizer to match the structure of f (Section 3.2).

Selection problems in other fields. In circuit discovery, ACDC prunes edges against a faithfulness threshold (Conmy et al., 2023), attribution patching replaces that pass by gradients (Syed et al., 2024), and edge pruning learns a mask (Bhaskar et al., 2024). The right faithfulness metric is itself debated (Zhang & Nanda, 2024; Hanna et al., 2024), and planted circuits allow evaluation against a ground truth (Gupta et al., 2024). Ensemble pruning uses greedy selection and ordered aggregation (Caruana et al., 2004; Martínez-Muñoz et al., 2009). Anchors and abductive explanations select feature sets that preserve a prediction (Ribeiro et al., 2018; Ignatiev et al., 2019; Bassan et al., 2026).

C Experimental Details

C.1 Domains and utilities

A utility is one domain, one dataset and one oracle model, an overview of the utilities in each domain is given in Table 4. A utility is *enumerated* when the whole 2^d table was computed, so that the true $\arg\max$ is known, and *wide* when it was not, in which case the reference optimum is the best value any method reached on that utility. Every utility is compared at budgets of up to 100 queries, so it needs $2^d > 100$ subsets. On the three tabular domains f is validation ROC-AUC for classification and validation R^2 for regression, and the empty subset scores a model with no input, which predicts the training mean resp. the majority-class frequencies.

Table 4: The utilities. One row per domain. A domain with two oracles holds each of its datasets once per oracle, the same dataset under two oracles being two set functions. d is the number of players.

			enumerated		wide	
domain	players	oracles	utilities	d	utilities	d
feature selection	features	LightGBM, TabPFN	38	7–15	52	17–1 776
data selection	training points	LightGBM, TabPFN	71	12	102	374–75 000
ensemble pruning	bagged members	LightGBM	51	16	—	
circuit discovery	graph edges	InterpBench transformers	—		7	53, 117
total			160		161	

Feature selection. A player is a column and $f(S)$ is the validation performance of the oracle model retrained on the columns in S, the 25 pool datasets with at most 16 features are enumerated and the other 26 are run budgeted, each under both oracles.

Data selection. Players are training examples and $f(S)$ is the validation performance of the model retrained on the rows in S, all features kept; the two granularities are different utilities. Enumerated, the training split is partitioned into 12 seeded class-stratified groups, and a player is a group. Wide, a player is one training row, all 51 datasets run under LightGBM and the 28 with the smallest splits ($d \leq 3\,746$) also under TabPFNv3.

Ensemble pruning. The library is 16 LightGBM members, each fit on its own bootstrap resample of the training split, and $f(S)$ is the validation performance of their average over S (soft voting for classification).

Circuit discovery. A player is an edge of the computational graph of a frozen InterpBench SIIT transformer (Gupta et al., 2024), and one query is one hooked forward pass on a fixed batch of 96 examples in which every edge outside S is ablated by resampling. With $\mathrm{KL}(S)$ the mean Kullback–Leibler divergence of the ablated model from the full one,

$$f(S) \;=\; 1 - \frac{\mathrm{KL}(S)}{\mathrm{KL}(\emptyset)} \;-\; \lambda \frac{|S|}{d},$$

so the full graph scores $1 - \lambda$ and the fully ablated one 0. A different size price λ is a different set function, so results are never averaged over it, the main results use $\lambda = 0.3$ on the seven released transformers and Appendix F.1 reports $\lambda \in \{0.1, 1\}$.

Oracle configurations. *Linear* is logistic regression for classification and ordinary least squares for regression, *LightGBM* (Ke et al., 2017) is 100 trees at library defaults, and *TabPFN* (Hollmann et al., 2025; Grinsztajn et al., 2026) is v3 with four estimators.

C.2 Datasets, splits and preprocessing

Every tabular utility is built on one pool: the 51 datasets of the TabArena suite (Erickson et al., 2025), read from OpenML, 38 classification and 13 regression tasks from 748 to 150 000 rows (Table 6). Thirty of them carry an official OpenML task and take its repeat-0/fold-0 test split, while for the other 21 we fall back to a seeded random split at a test fraction of $1/3$. A validation set is then carved out of the training portion at 25%, and each split is capped at 2 000 rows so that one evaluation of f costs about the same across the pool, with data selection over individual training rows being the one exception. Categorical columns are ordinal-encoded, so that a subset is a set of column indices, and the remaining missing values are imputed by the column mean. Licenses are those of the OpenML records.

Rank percentile. In a cell c (one utility, one budget, one seed), each method m reaches a best value v_m after t_m seconds. A method m' *beats* m if it finds a better value, $v_{m'} > v_m$, or the same value faster, $v_{m'} = v_m$ and $t_{m'} < t_m$; the two *tie* if both value and time are equal. With M_c the other methods run in the cell, the rank percentile of m is the share of them that beat it, counting ties as one half:

$$\mathrm{pct}(m) \;=\; \frac{1}{|M_c|} \sum_{m' \in M_c} \Big(\mathbb{1}[\, m' \text{ beats } m \,] + \tfrac{1}{2}\, \mathbb{1}[\, m' \text{ ties } m \,] \Big).$$

A percentile of 0 means no method did better, and 1 means every method did.

C.3 Measuring the properties

Table 1 defines the six properties and Figure 2 scores each. In the following section, we describe how the scores are computed and on which utilities. Every score equals 1 exactly when the utility has the property. On an enumerated utility, probabilities and expectations run over all 2^d subsets, and the score is exact. For all the others, they are estimated from random subsets that include each player independently with probability $1/2$. Table 5 summarizes the coverage of the metrics across our utilities.

Monotonicity, sub- and supermodularity. For a uniform subset S and uniform players $i \neq j$ outside it,

$$\textsc{Mon}(f) = \Pr\big[\, f(S \cup \{i\}) \geq f(S) \,\big], \tag{6}$$

$$\textsc{Sub}(f) = \Pr\big[\, f(S \cup \{i\}) - f(S) \;\geq\; f(S \cup \{i,j\}) - f(S \cup \{j\}) \,\big], \tag{7}$$

and $\textsc{Sup}(f)$ is $\textsc{Sub}(f)$ with the inequality reversed. On an enumerated utility we count every (S, i) and (S, i, j); otherwise we draw 2 000 triples and evaluate f on S, $S \cup \{i\}$, $S \cup \{j\}$ and $S \cup \{i, j\}$. Equalities count as holding, so a zero-second order difference counts for both Sub and Sup, and differences below 10^{-12} times the largest $|f|$ involved are treated as zero to absorb rounding.

Modularity and degree two. The score is the share of the variance of f explained by the best polynomial of degree at most k,

$$\textsc{Deg}_k(f) = 1 - \min_{m} \frac{\mathbb{E}_S\big[\big(f(S) - \sum_{R \subseteq S,\, |R| \le k} m(R)\big)^2\big]}{\operatorname{Var}_S f(S)}, \tag{8}$$

with $1 + d$ coefficients for $k = 1$ and $1 + d + \binom{d}{2}$ for $k = 2$. S ranges over non-empty subsets: $f(\emptyset)$ is a convention, since no model is fitted on it, and in ensemble pruning and data selection, it lies 89 and 27 standard deviations from the median, enough to decide the score alone. On an enumerated utility, the minimum is a least-squares fit to all $2^d - 1$ non-empty subsets. Otherwise, we fit on 80% of the sampled subsets and report R^2 on the other 20%, provided the fit has at least five subsets per coefficient. We omit $\textsc{Deg}_2$ for $d < 8$, where the coefficients are a large share of all subsets and the score is close to 1 by counting alone.

MTM. Detecting MTM holds then for f with weights w exactly when $f(S) > f(T)$ implies $w(S) > w(T)$, so we score the largest share of ordered pairs a single w reproduces,

$$\textsc{Mtm}(f) = \max_{w \in \mathbb{R}^d} \Pr\big[\, w(S) > w(T) \;\big|\; f(S) > f(T) \,\big], \tag{9}$$

over independent uniform subsets S and T. The maximum is a maximum-feasible-subsystem problem, so we report the held-out agreement for one fitted w, an estimate of a lower bound. Where Wang et al. (2024) learns the monotone transform with a neural network, we fit w by least squares to the normal scores $z(S) = \Phi^{-1}\big((r(S) - \tfrac{1}{2})/n\big)$, with $r(S)$ the rank of S among the n training subsets. In contrast to Wang et al. (2024), we can thus compare across domains. We train on half of all subsets for an enumerated utility and on 80% of the sample otherwise, with at least five subsets per weight, and count how often w orders 200 000 held-out pairs as f does, dropping pairs tied in f and counting ties in w as one half.

Deciding MTM with a linear program. A score below 1 does not rule out MTM, since a better w may exist. Therefore, we use a linear program to determine whether one exists. Let $v_1 > \cdots > v_K$ be the distinct values of f on the evaluated subsets and B_k the subsets with value v_k. The program asks for $w \in \mathbb{R}^d$ and thresholds $t \in \mathbb{R}^K$ with

$$w(S) \;\ge\; t_k \quad \text{and} \quad w(S) \;\le\; t_{k-1} - \varepsilon \;\; (k \ge 2) \qquad \text{for } S \in B_k, \tag{10}$$

which separates consecutive value classes by a margin ε. Rescaling (w, t) rescales the margin, so we fix $\varepsilon = 1$ without loss of generality, and solve with HiGHS through `scipy.optimize.linprog` and a zero objective. Infeasibility in the evaluated subsets rules out MTM for f, while feasibility confirms it only for an enumerated utility. Replacing $w(S)$ by $\sum_{i \in S} w_i + \sum_{\{i,j\} \subseteq S} q_{ij}$ gives the same program for monotone transforms of degree-two functions; this class contains every degree-two function, so infeasibility also rejects degree two. Without the 2^d table, we solve both programs on slices. We draw a set K of 10 players and a base set B that contains each other player with probability $1/2$, and evaluate $f_B(T) = f(B \cup T)$ on all 1 024 subsets $T \subseteq K$. A slice of an MTM utility is MTM, since $f_B(T) = g\big(w(B) + w(T)\big)$ is nondecreasing in $w(T)$, and a slice of a monotone transform of a degree-two function is again one, since cross terms between B and Tbecome linear in T. An infeasible slice, therefore, rejects the property; we draw three slices per utility.

Weighting in Figure 2. On the radar, a utility counts on a spoke with weight $(P(f) - P_{\min})/(P_{\max} - P_{\min})$, where $P(f)$ is its score for that property and $P_{\min}$ and $P_{\max}$ are the smallest and largest score for that property over the utilities; a utility without a score gets weight zero there. This min-max scaling puts every property on the same range, so each spoke weights the utilities by how close they come to the property relative to the other utilities, with no tuning constant. In each cell of one utility and one seed at 100 queries, a method scores the share of the other plotted methods it beats, with ties in value going to the faster method as in Table 2. A spoke shows the weighted mean of this share over utilities and seeds. A method that did not run on a utility, or cannot finish within 100 queries, is averaged over the remaining utilities.

Table 5: Utilities on which each score was measured and each linear program was infeasible. The scores are exact on the enumerated utilities and sampled on the others; the programs run on the full table resp. on three slices. Enumerated utilities are feature selection, data selection over groups and ensemble pruning with $d \leq 16$; a wide utility has no table.

			scored			LP infeasible	
utilities	**count**	d	MON, SUB, SUP	DEG_1, MTM	DEG_2	MTM	deg. 2
enumerated	160	7–16	160	160	160	160	160
wide feature sel. (LightGBM)	26	17–1 776	26	23	2	26	25
data sel., rows	51	374–75 000	51	6	0	51	51
circuit discovery	7	53–117	7	7	7	7	7

C.4 RANKING, TIES AND WALL CLOCKS

Rank percentile. For one utility, budget and seed, a method's rank percentile is the share of the domain's reference field that scores better in the same cell. The field is the published black-box methods run on every utility and budget of the domain, 11 to 22 depending on the domain and oracle; neither S-LFBO nor the gain-importance fit is a member. A cell of Table 2 averages the percentile over seeds and then over the domain's utilities, enumerated and wide alike.

Ties. A tie in f goes to the faster run and is split when the two clocks differ by less than $0.5\,\text{s}$: methods run against one memoized oracle per utility, so a method that reuses evaluations another has paid for records a clock near zero. Splitting every tie instead would move 7 medals.

Marks. A method run on fewer than 80% of a column's utilities is shown in parentheses and not medalled. $*$ and $**$ mark a paired Wilcoxon signed-rank test against S-LFBO over the column's utilities, each method's percentile first averaged over the utilities seeds, at $p < .05$ and $p < .01$. The subscript is the standard error of that paired difference, and of the mean itself for S-LFBO and *others*. A '–' marks a method that does not apply to the domain. The gain-importance fit reads the training split, so it is placed against the field but never medalled. *others* averages the benchmark's remaining 13 published methods; the small number in a cell is how many of them that column measured. On ensemble pruning at $b = 100$, hill climbing from *others* ranks ahead of the gold.

Wall clocks in Figure 4. In feature selection and data selection, every method was timed again at 100 queries on one isolated NVIDIA L40S, one method per job; a run's clock is a fixed start-up of $12.8\,\text{s}$, the TabPFN checkpoint load and utility build, plus the rest of that measurement in proportion to the queries it makes. The gradient-boosted default makes one query and is drawn at its own measured clock. In circuit discovery, a run's clock is the method's own time plus its queries times the utility's forward-pass time, $6\,\text{ms}$ at $d = 53$ and $10\,\text{ms}$ at $d = 117$. A method is drawn over the clocks at which it was run and continued as a dotted level: ACDC because more budget does not change its answer, GP-BO because no run beyond 500 queries finished.

C.5 HARDWARE, SOFTWARE AND COMPUTE

CPU work ran as job arrays on shared dual-socket x86 nodes, two cores, and $8\,\text{GB}$ of memory per cell. The only work that needs a GPU is TabPFN: those cells ran on 20 GB A100 slices, and the wall clocks reported for TabPFN data selection in Figure 4a were measured with one dedicated L40S and 16 cores per run. The methods used about 1 900 hours of CPU time and about 640 hours of GPU time, and computing the enumerated tables beforehand requires an additional 3.6 million subset evaluations. The implementation can be found here `https://anonymous.4open.science/r/UnifyingSelectionProblem/README.md`.

D BASELINE DETAILS

We describe the core idea of each method. BOCS, COMBO, GP-BO and LFBO/BORE are transcribed from the authors' reference implementations, the others follow the published pseudocode, and hyperparameters are the originals' defaults.

Table 6: The dataset pool. The 51 OpenML datasets behind the feature- and data-selection utilities, split at the feature count that separates enumerated ($d \leq 16$) from wide ($d > 16$) feature-selection utilities. "id": OpenML dataset id; "task": official OpenML task id, or – when none exists and a seeded random split is used; d: number of players in feature selection.

enumerated: at most 16 features, full 2^d table						*wide*: 17 features or more, no table					
slug	id	task	type	rows	d	slug	id	task	type	rows	d
blood_transfusion	46913	363621	clf	748	4	online_shoppers	46947	363691	clf	12 330	17
airfoil_self_noise	46904	363612	reg	1 503	5	churn	46915	—	clf	5 000	19
fitness_club	46927	363671	clf	1 500	6	credit_g	46918	363626	clf	1 000	20
health_insurance	46931	363675	reg	1 338	6	airline_satisfaction	46920	—	clf	129 880	21
maternal_health	46941	363685	clf	1 014	6	jm1	46979	—	clf	10 885	21
qsar_fish_toxicity	46954	363698	reg	907	6	credit_card_default	46919	363627	clf	30 000	23
fiat_500	46907	363615	reg	1 538	7	heloc	46932	—	clf	10 459	23
concrete_strength	46917	363625	reg	1 030	8	coupon_recommendation	46937	—	clf	12 684	24
diabetes	46921	363629	clf	768	8	marketing_campaign	46940	363684	clf	2 240	25
houses	46934	363678	reg	20 640	8	hazelnut	46930	363674	clf	2 400	30
amazon_employee_access	46905	363613	clf	32 769	9	students_dropout	46960	—	clf	4 424	36
diamonds	46923	363631	reg	53 940	9	anneal	46906	—	clf	898	38
food_delivery	46928	363672	reg	45 451	9	qsar_biodeg	46952	—	clf	1 054	41
protein	46949	363693	reg	45 730	9	diabetes130us	46922	—	clf	71 518	47
website_phishing	46963	363707	clf	1 353	9	splice	46958	—	clf	3 190	60
bank_customer_churn	46911	363619	clf	10 000	10	bankruptcy	46950	—	clf	5 910	64
ecommerce_shipping	46924	363632	clf	10 999	10	superconductivity	46961	—	reg	21 263	81
give_me_credit	46929	363673	clf	150 000	10	coil2000	46916	—	clf	9 822	85
sdss17	46955	363699	clf	78 053	11	naticusdroid	46969	—	clf	7 491	86
hr_analytics	46935	363679	clf	19 158	12	taiwanese_bankruptcy	46962	—	clf	6 819	94
wine_quality	46964	363708	reg	6 497	12	mic	46980	—	clf	1 699	111
bank_marketing	46910	363618	clf	45 211	13	aps_failure	46908	—	clf	76 000	170
good_customer	46938	363682	clf	1 723	13	kddcup09	46939	—	clf	50 000	212
miami_housing	46942	363686	reg	13 776	15	qsar_tid11	46953	—	reg	5 742	1 024
seismic_bumps	46956	363700	clf	2 584	15	hiva_agnostic	46933	—	clf	3 845	1 617
						bioresponse	46912	—	clf	3 751	1 776

Common setup. Every method reaches f only through an oracle that counts queries, runs until the budget is spent, and returns the best subset it queried. Repeated queries and queries to a fitted surrogate are free. A method built for a fixed cardinality is made unconstrained by evaluating the prefixes of its own ranking and keeping the best one. Methods whose cost rules out large utilities are run only up to a size and budget cap, stated with each method; the results mark the remaining cells as not applicable.

D.1 Direct search on the oracle

Random search, hill climbing, forward–backward selection and simulated annealing (Kirkpatrick et al., 1983). Reference points that query f directly. Random search draws subsets uniformly; hill climbing flips single elements while that improves f, restarting from random subsets; forward–backward selection takes the best single addition or removal until none improves. Simulated annealing accepts worsening flips with a probability that decreases over the run, so it can leave local optima.

Greedy selection (Nemhauser et al., 1978; Minoux, 1978; Leskovec et al., 2007; Mirzasoleiman et al., 2015; Hu et al., 2024). Forward greedy adds the element with the largest marginal gain until no addition improves f, which is near-optimal when f is monotone submodular. Lazy evaluation keeps the same trajectory with fewer queries, and stochastic greedy scores only a random pool of candidates per step. Static greedy ranks the elements once by their singleton gains, while adaptive greedy re-scores every remaining element after each pick; the pair separates one-shot scoring from re-scoring.

Double greedy (Buchbinder et al., 2015) and GreedMax (Bai & Bilmes, 2018). Double greedy grows a set from $\emptyset$ and shrinks one from $[d]$ in parallel, deciding each element at random in proportion to its gain in either direction. It is the $\frac{1}{2}$-approximation for unconstrained non-monotone submodular maximization, at $2d$ queries. GreedMax targets the sum of a submodular and a supermodular function; with a single value oracle, it follows the forward-greedy trajectory.

POSS (Qian et al., 2015b) and PEP (Qian et al., 2015a). Both treat selection as a bi-objective problem, maximizing f while minimizing $|S|$, and evolve an archive of non-dominated subsets by random bit flips. POSS is the subset-selection variant with an approximation guarantee matching greedy's; PEP, the published ensemble-pruning method, adds a local search that may accept worsening steps.

Anchors (Ribeiro et al., 2018). A beam search over rules that fix some elements in or out of S, where each rule is scored by bandit sampling of its random completions. The rule is accepted when its completions exceed a value threshold with high probability, and the method returns the best completion of the final rule.

D.2 Attribution and additive surrogates

Data Shapley (Ghorbani & Zou, 2019) and Data Banzhaf (Wang & Jia, 2023). Both score each element by a semivalue, a weighted average of its marginal contributions, and keep the best prefix of the resulting ranking. Data Shapley estimates the Shapley value by walking random permutations and truncating a walk once the value is close to $f([d])$. Data Banzhaf uses uniform weights over coalitions, which makes it the most robust semivalue to noisy utilities; its estimator reuses every sampled subset for all d scores.

NASH (Tian et al., 2026). NASH decomposes f into per-validation-point utilities and computes a Shapley value for each of them. It then greedily maximizes a concave function of these additive utilities, which captures diminishing returns across validation points. It therefore applies only where f averages over a validation set. We run it up to $d = 512$, since its greedy pass is quadratic in d with one exponential per validation point.

D.3 Combinatorial Bayesian optimization

BOCS (Baptista & Poloczek, 2018). BOCS models f as a degree-two polynomial with a sparsity-inducing horseshoe prior. Each round draws one sample from the posterior by Gibbs sampling and queries its maximizer, found by simulated annealing; the Thompson draw provides the exploration. We run it up to $d = 128$ and 256 queries: each step Gibbs-samples all $1 + d + \binom{d}{2}$ coefficients, which already takes minutes per query at $d \approx 100$.

Min-cut on a degree-two surrogate. Each round fits a degree-two model by ridge regression, removes its negative pairwise terms so that it is supermodular, and maximizes the result exactly by one minimum cut (Boros & Hammer, 2002). We run it up to $d = 4096$, where the $d \times d$ pairwise matrix and the max-flow graph of $d^2/2$ edges reaches its memory limit.

LFBO and BORE (Song et al., 2022; Tiao et al., 2021). The recipe our method instantiates: a classifier of the event $f(S) > \tau$ replaces the model of f, and its odds serve as the acquisition. BORE's unweighted classifier recovers the probability of improvement $\mathbb{1}_{f>\tau}$, and LFBO's weighting by $(f(S) - \tau)_+$ recovers expected improvement. By default, both use a random forest and maximize the acquisition by scoring random candidates.

COMBO (Oh et al., 2019). COMBO places a Gaussian process on the graph product of the elements, with a diffusion kernel whose hyperparameters are sampled from their posterior. It maximizes expected improvement by local search from the best of many candidate subsets. It is the slowest method here, its posterior sampling and Gaussian-process refits being cubic in the number of observations, so we run it only up to $d = 16$ and 100 queries.

GP-BO (Dreczkowski et al., 2023). The best combination found by this benchmark study: a Gaussian process with a Hamming-type kernel, expected improvement, and a genetic algorithm restricted to a trust region around the incumbent. The trust region grows after successes and shrinks after failures. We stop it at 500 queries: the Gaussian process is refitted after every query at a cost cubic in the archive, about an hour per run at 500 queries, and no longer runs are finished.

SMAC (Lindauer et al., 2022). The AutoML standard, run as the released package: a random forest whose spread across trees gives the uncertainty for expected improvement, maximized by local and random search. We run it up to $d = 512$, since the refit and the acquisition search take about ten seconds per query at $d = 2000$.

LA-MCTS (Wang et al., 2020). LA-MCTS recursively partitions the search space into good and bad regions using learned classifiers and selects a region via Monte Carlo tree search. A Gaussian process then proposes the next subset inside that region. Rebuilding the tree and refitting the Gaussian process at every query is cubic in the archive size, as with COMBO, so we stop at 100 queries.

D.4 Domain-specific baselines

ACDC (Conmy et al., 2023). ACDC walks the components in reverse topological order and removes each one whose removal changes the output by less than a threshold. We run it over a range of thresholds, so each pass yields one candidate circuit.

LightGBM gain importance (Ke et al., 2017). A often-adopted feature-selection variant, using the importance scores of a fitted gradient-boosted model from the training data. It never queries f, so it is shown as a single level rather than a curve, and it runs only when the players are used as input features.

E S-LFBO Ablations

Each ablation below changes one component of S-LFBO and keeps the rest at the default of Section 3.2: a design of 10 points, the threshold τ at the $\frac{2}{3}$-quantile of the evaluations so far, five annealing chains of 256 flips, and one proposal per fit. Comparisons are paired over utilities, on each utility's mean over three seeds, with a Wilcoxon signed-rank test. An enumerated utility leaves a comparison once the budget equals 2^d.

E.1 The spectra of the utilities and of the target

We compute the exact Walsh spectrum of f, of the label $z = \mathbf{1}\{f > \tau\}$ and of the target a on the 135 enumerated utilities of the Hierarchy panel in Figure 3: feature and data selection (groups) under LightGBM and TabPFN-v3 and ensemble pruning, $4 \le d \le 16$, the tables the ladder panels run on; those panels add the wide LightGBM feature- and data-selection (rows) utilities. The support of a function is the smallest set of interactions carrying 95% of its energy, as in Section 3.2. Figure 6 adds the label z, which the main panel leaves out.

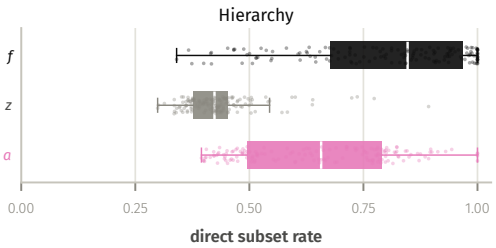

Figure 6: Hierarchy of f, z and a. Direct subset rate of the exact Walsh spectra on 155 of the 160 enumerated games of Table 2, the games of the Hierarchy panel of Figure 3. Boxes: quartiles over games, whiskers: range.

The utilities are hierarchical. The direct subset rate of f exceeds a degree-matched null, which permutes the coefficients within each order, and so keeps every magnitude and the degree profile, on 97% of the utilities (median 0.86 against 0.64).

Thresholding costs structure; the improvement weight returns most of it. Thresholding moves the energy to high orders: a third of the energy of z sits at order three and above, against a sixth for f, and its direct subset rate falls to 0.42, barely above its null. Weighting by the improvement undoes most of this: a is more hierarchical than z on 138 of the 155 games (median 0.66, $p = 10^{-23}$), though it stays below f.

The tree selects best on a. Figure 7 extends the Transition panel of Figure 3 by the tree fitted once on z. On the enumerated utilities, the order follows the spectra: the tree on a ranks ahead of the tree on z (71 utilities against 25 at 100 queries, $p = 4 \cdot 10^{-7}$), which ranks ahead of the tree on f (67 against 25, $p = 4 \cdot 10^{-6}$), so thresholding and the improvement weight each contribute about half of the target's gain.

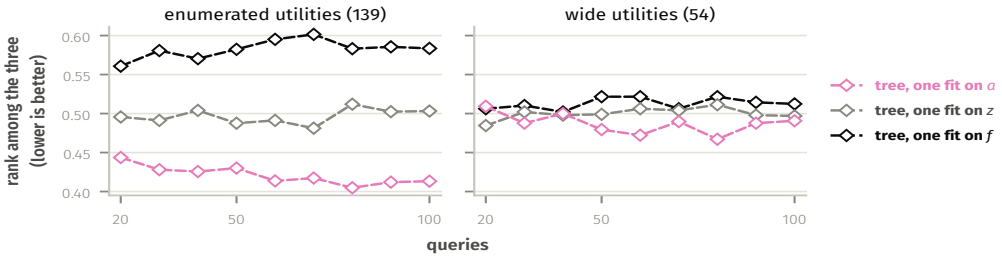

Figure 7: The target for the tree. The tree surrogate of S-LFBO fitted once on f, on the label z and on the target a, with design and maximizer unchanged; rank among the three per utility and budget, mean over utilities, three seeds.

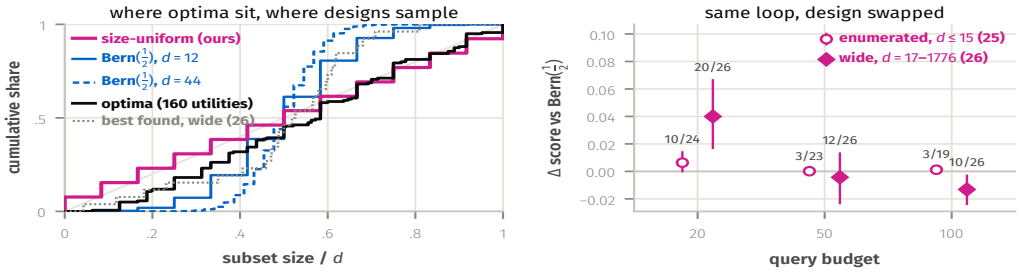

Figure 8: The initial design. Left: cumulative share of $|S^\star|/d$ over 179 enumerated utilities and of the best subset found on 26 wide utilities, against the sizes the two designs draw at $d = 12$ and $d = 44$. Right: the same loop with only the ten design points exchanged; mean difference in normalized score, 95% bootstrap interval, labels: utilities where size-uniform is ahead.

E.2 Design, surrogate and maximizer

From LFBO to S-LFBO. Figure 9 ranks LFBO, LFBO with gradient-boosted trees, and S-LFBO with one component reverted to LFBO's choice against the reference field of Table 2. Replacing LFBO's random forest by 200 depth-five boosted trees recovers only part of the gap: at 100 queries it reaches .29, .32, .25 and .22 in the four domains, against .38, .37, .36 and .23 for LFBO and .16, .14, .16 and .11 for S-LFBO ($p < .01$ in the first three domains, $p < .05$ in circuit discovery). Most of the remaining gap is the maximizer. Scanning 5,000 uniform random subsets instead of annealing from warm starts drops S-LFBO to .27, .26, .24 and .22, behind in every domain ($p < .01$; $p < .05$ in circuit discovery). The uniform design costs in data selection at every budget ($p < .01$) and in circuit discovery at 20 queries ($p < .05$), is ahead in ensemble pruning at 20 queries ($p < .05$), and is level elsewhere. Depth-five trees alone stay within .03 of the stumps-then-deeper schedule in the first three domains and lead in circuit discovery at 100 queries (.07 against .11, $p < .05$), so the schedule is never significantly behind either pure depth. The schedule is cheaper, since 150 of its 200 trees are stumps: depth-five trees alone take 1.25 times as long per 100-query run (median, 1.04 to 1.56 across domains).

Sampling design. The size of the optimum is not known in advance and varies widely: $|S^\star|/d$ ranges from 0.06 to 1 over the enumerated utilities, and a third of the optima lie outside $[0.2, 0.8]$ (Figure 8, left). Uniformly random subsets, the baselines' design, almost never have such sizes once d is moderate, since their size concentrates at $d/2$; drawing the size uniformly first keeps about 40% of the design there at every d. Exchanging only the ten design points (Figure 8, right), the

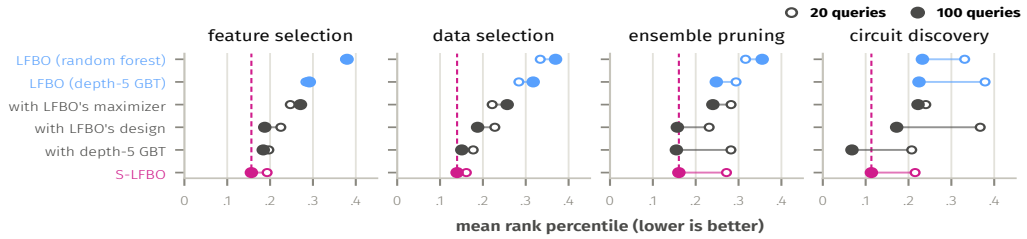

Figure 9: Step-by-step from LFBO to S-LFBO. Mean rank percentile against the reference field of Table 2, on its utilities and three seeds (lower is better); filled: 100 queries, hollow: 20 queries. Shaded rows: S-LFBO with one component returned to LFBO's choice (maximizer: best of 5,000 uniformly random subsets; design: uniformly random subsets; surrogate: 200 trees of depth five). Dashed: S-LFBO at 100 queries.

size-uniform design is ahead at the smallest budgets, on both the wide and the enumerated utilities ($p = 0.004$ and $p = 0.012$), and level from 50 queries on. It matters while the design is a large share of the budget.

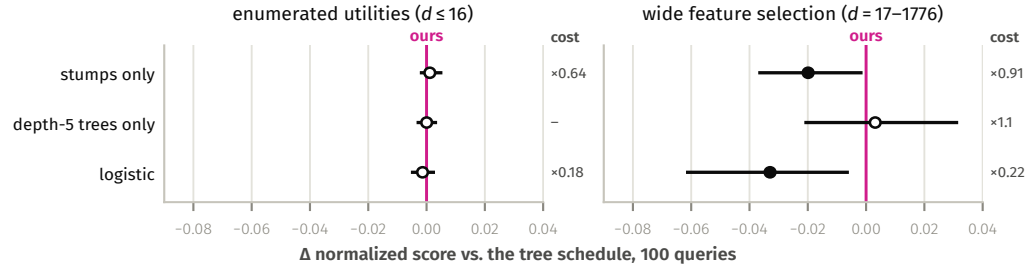

Figure 10: The surrogate. Paired difference in the normalized score to the tree schedule at 100 queries, mean over utilities, 95% bootstrap interval; filled: $p < 0.05$. Cost: run time relative to the tree schedule on the same machine.

Surrogate. We swap the classifier inside the loop and pair each swap with the tree schedule on the same utility and seed (Figure 10). On the enumerated utilities every swap is level with the schedule ($p \geq 0.4$). On the wide feature-selection utilities both cheaper surrogates lose, stumps alone ($p = 0.024$) and the logistic model ($p = 0.02$), while depth-five trees tie the schedule. Interaction capacity therefore pays only once d is large, and the schedule provides it without choosing a depth: it is never significantly behind either pure depth.

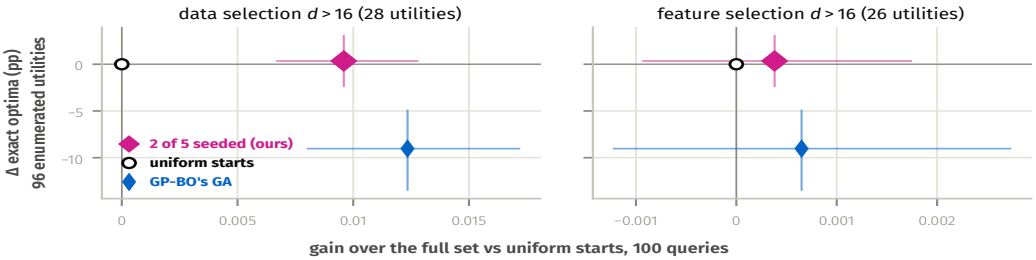

Figure 11: The maximizer. Each point is one maximizer in the same loop at 100 queries, relative to uniform starts: gain over the full set on the wide data-selection (28, left) and feature-selection (26, right) utilities against exact-optimum rate on 96 enumerated utilities. Whiskers: 95% bootstrap over utilities.

Maximizer. On a wide cube, a chain from a uniform start ends as far from every evaluated subset as a random draw and never refines what the loop has learned; on a small cube, uniform starts are

what find the optimum. Starting two of the five chains at the incumbent and the previous proposal serves both (Figure 11). It adds gain on wide data selection at every budget ($p \leq 0.002$) and raises the exact-optimum rate on enumerated utilities ($p = 0.011$). Seeding all five chains, or using the genetic algorithm of GP-BO, adds no gain and loses optima; on wide feature selection, no maximizer separates. We anneal the booster's forward pass rather than its Walsh spectrum: the two are the same function on $\{0, 1\}^d$, so the gain is unchanged, and the forward pass is cheaper on wide utilities.

Table 7: Circuit discovery by size price: mean rank percentile (lower is better). The 7 InterpBench models at three size prices λ, ranked against one field of 16 black-box methods over all 21 utilities. Mean over utilities of seed means (10 seeds), subscript: spread across utilities. Gold, silver and bronze mark each row's three best values; the main results use $\lambda = 0.3$.

λ	b	utilities	S-LFBO	GP-BO	SMAC	LFBO	BORE	Data Shapley	greedy forward	ACDC
$\lambda = 0.1$	20	7	$.26_{\pm.12}$	$\mathbf{.22}_{\pm.10}$	$.36_{\pm.20}$	$.32_{\pm.17}$	$.36_{\pm.08}$	$.34_{\pm.18}$	$.82_{\pm.05}$	$.34_{\pm.18}$
	50	7	$.14_{\pm.06}$	$\mathbf{.13}_{\pm.11}$	$.27_{\pm.07}$	$.21_{\pm.15}$	$.27_{\pm.09}$	$.38_{\pm.18}$	$.85_{\pm.02}$	$.40_{\pm.20}$
	100	7	$\mathbf{.09}_{\pm.05}$	$.10_{\pm.09}$	$.22_{\pm.08}$	$.21_{\pm.11}$	$.27_{\pm.15}$	$.50_{\pm.19}$	$.86_{\pm.02}$	$.32_{\pm.24}$
$\lambda = 0.3$	20	7	$.24_{\pm.09}$	$\mathbf{.20}_{\pm.09}$	$.27_{\pm.13}$	$.36_{\pm.08}$	$.31_{\pm.11}$	$.39_{\pm.18}$	$.81_{\pm.06}$	$.39_{\pm.18}$
	50	7	$.12_{\pm.07}$	$\mathbf{.07}_{\pm.08}$	$.23_{\pm.08}$	$.14_{\pm.03}$	$.27_{\pm.12}$	$.48_{\pm.14}$	$.84_{\pm.03}$	$.50_{\pm.15}$
	100	7	$.10_{\pm.05}$	$\mathbf{.09}_{\pm.10}$	$.19_{\pm.07}$	$.18_{\pm.05}$	$.26_{\pm.14}$	$.54_{\pm.14}$	$.85_{\pm.02}$	$.39_{\pm.25}$
$\lambda = 1$	20	7	$.26_{\pm.12}$	$\mathbf{.18}_{\pm.09}$	$.31_{\pm.15}$	$.33_{\pm.09}$	$.27_{\pm.15}$	$.64_{\pm.12}$	$.64_{\pm.12}$	$.64_{\pm.12}$
	50	7	$.12_{\pm.07}$	$\mathbf{.08}_{\pm.05}$	$.16_{\pm.07}$	$.27_{\pm.09}$	$.21_{\pm.13}$	$.68_{\pm.09}$	$.72_{\pm.07}$	$.72_{\pm.07}$
	100	7	$.12_{\pm.09}$	$\mathbf{.08}_{\pm.08}$	$.18_{\pm.06}$	$.24_{\pm.11}$	$.24_{\pm.12}$	$.61_{\pm.04}$	$.78_{\pm.07}$	$.56_{\pm.35}$

F Further Results

This appendix presents the circuit utilities at the two size prices that the main text does not use, and every utility in Table 2 on its own.

F.1 Circuit discovery at other size prices

The circuit objective charges the size of the selected graph at a price λ (Appendix C.1), and a different price is a different set function. The main results use $\lambda = 0.3$; Table 7 repeats the comparison at 0.1 and 1.

GP-BO has the best mean rank in seven of the nine rows and S-LFBO in the other two, $\lambda = 0.3$ at 20 queries and $\lambda = 1$ at 50; wherever GP-BO leads, S-LFBO is second. Averaged over the three budgets, the margin of GP-BO over S-LFBO is 0.05 at $\lambda = 0.1$, 0.02 at 0.3 and below 0.01 at $\lambda = 1$. A higher price moves the optimum to smaller subsets: the best subset any method found within 100 queries keeps 27% of the edges on average at $\lambda = 0.1$ and 12% at $\lambda = 1$. Both methods stay above that size, and after 100 queries S-LFBO keeps 32% of the edges at $\lambda = 1$ against GP-BO's 30%. ACDC loses most from the price, from 0.33 at $\lambda = 0.1$ to 0.63 at $\lambda = 1$ at 20 queries: below its pass, it returns the full graph, which a higher λ charges more.

F.2 How large are the gains?

Rank percentile in Table 2 indicates which methods wins, but does not report the relative improvement the subset yields, which we showcase in Table 8, averaged over seeds and utilities. At 100 queries S-LFBO improves on the full set by 3.0%, 2.4% and 1.8% in feature selection, data selection and ensemble pruning, on 94%, 89% and 98% of the utilities. This is three-quarters of the gain available from the best-known subset in the first two domains and all of it in ensemble pruning. The gains are skewed: the top tenth of the utilities accounts for over half of the total, led by `hiva_agnostic`, where the full feature set scores near chance, and a subset improves on it by up to 55%. Data Shapley gains 1.6%, 0.9% and 1.1%, and S-LFBO beats it on 82 of 90, 153 of 173 and 50 of 51 utilities. GP-BO reaches nearly the same value: at 100 queries, the two differ by at most 0.11 percentage points. Taking into account the wall-time, on utilities with $d \leq 16$, a 100-query run of S-LFBO takes a median of 4 to 8 s against about 17 s for GP-BO, and with the oracle's cost included, 120 s buy S-LFBO +2.9% and +4.4% in feature and data selection, against +2.5% and +3.4% for GP-BO (Figure 4). In circuit discovery, the full graph pays the whole size price, $f([d]) = 0.7$, so every pruning method gains tens of percent (23.7% for S-LFBO, 25.8% for GP-BO, which leads on 6 of 7 utilities), mostly from the size price saved.

Table 8: Relative gain over the full set at three query budgets (higher is better). Relative Improvement as 100 $\big(f(S) - f([d])\big)/|f([d])|$, averaged over seeds and then utilities; utilities, methods and marks as in Table 2. *Best subset*: the most there is to gain, from the true optimum on enumerated utilities and the best value found on wide ones. Subscripts and stars as in Table 2. Circuit gains are on a different scale, since the full graph pays the whole size price $f([d]) = 1 - \lambda$ (Appendix F.2).

	feature selection			data selection			ensemble pruning			circuit discovery		
	90 utilities			173 utilities			51 utilities			7 utilities		
b	20	50	100	20	50	100	20	50	100	20	50	100
best subset	3.96	3.96	3.96	3.16	3.16	3.16	1.83	1.83	1.83	28.73	28.73	28.73
Ours												
S-LFBO	**1.83**$_{\pm 0.52}$	**2.55**$_{\pm 0.60}$	**2.97**$_{\pm 0.67}$	**1.38**$_{\pm 0.29}$	**2.04**$_{\pm 0.38}$	2.38$_{\pm 0.42}$	**1.21**$_{\pm 0.30}$	**1.76**$_{\pm 0.41}$	**1.81**$_{\pm 0.42}$	**9.17**$_{\pm 4.08}$	18.74$_{\pm 3.04}$	23.70$_{\pm 3.01}$
Domain methods												
LightGBM Gini	$(-0.09^{**}_{\pm 0.57})$	$(-0.09^{**}_{\pm 0.69})$	$(-0.09^{**}_{\pm 0.78})$	–	–	–	–	–	–	–	–	–
Data Shapley	$0.78^{**}_{\pm 0.23}$	$1.41^{**}_{\pm 0.24}$	$1.61^{**}_{\pm 0.29}$	$0.44^{***}_{\pm 0.21}$	$0.63^{**}_{\pm 0.27}$	$0.91^{**}_{\pm 0.25}$	$0.66^{**}_{\pm 0.12}$	$0.99^{**}_{\pm 0.13}$	$1.12^{**}_{\pm 0.10}$	$0.00_{\pm 4.08}$	$1.12^{*}_{\pm 2.48}$	$4.10^{*}_{\pm 2.34}$
NASH	$0.24^{**}_{\pm 0.19}$	$0.39^{**}_{\pm 0.26}$	$0.55^{**}_{\pm 0.31}$	$(0.39^{*}_{\pm 0.23})$	$(0.63^{**}_{\pm 0.30})$	$(0.79^{**}_{\pm 0.34})$	$0.46^{**}_{\pm 0.29}$	$0.59^{**}_{\pm 0.38}$	$0.59^{**}_{\pm 0.38}$	$0.00_{\pm 4.08}$	$0.00^{*}_{\pm 3.04}$	$3.31^{*}_{\pm 1.83}$
Greedy forward	$-11.53^{**}_{\pm 1.76}$	$-5.75^{**}_{\pm 1.67}$	$-2.76^{**}_{\pm 1.52}$	$-34.69^{***}_{\pm 2.48}$	$-33.08^{***}_{\pm 2.57}$	$-33.06^{***}_{\pm 2.58}$	$0.76^{*}_{\pm 0.22}$	$1.71_{\pm 0.04}$	$1.79^{*}_{\pm 0.02}$	$-100.00^{*}_{\pm 4.08}$	$-100.00^{*}_{\pm 3.04}$	$-100.00^{*}_{\pm 3.01}$
ACDC	–	–	–	–	–	–	–	–	–	$0.00_{\pm 4.08}$	$0.00^{*}_{\pm 3.04}$	$10.84^{*}_{\pm 4.92}$
PEP	$-1.52^{**}_{\pm 0.39}$	$0.79^{**}_{\pm 0.36}$	$1.46^{**}_{\pm 0.41}$	$-1.18^{**}_{\pm 0.32}$	$-0.24^{**}_{\pm 0.36}$	$0.04^{**}_{\pm 0.40}$	$0.51^{**}_{\pm 0.15}$	$1.10^{**}_{\pm 0.14}$	$1.72^{**}_{\pm 0.02}$	$-78.30^{*}_{\pm 6.85}$	$-65.91^{*}_{\pm 8.57}$	$-59.03^{*}_{\pm 10.44}$
Black-box optimisers												
GP-BO	$1.63^{*}_{\pm 0.15}$	$2.47_{\pm 0.13}$	$2.87_{\pm 0.14}$	$0.95^{**}_{\pm 0.14}$	$1.84^{*}_{\pm 0.11}$	**2.49**$_{\pm 0.11}$	$1.13_{\pm 0.06}$	**1.76**$_{\pm 0.01}$	$1.79^{**}_{\pm 0.01}$	$-5.29_{\pm 13.76}$	**20.12**$_{\pm 1.73}$	**25.82**$^{*}_{\pm 0.67}$
SMAC	$0.32^{**}_{\pm 0.16}$	$1.32^{**}_{\pm 0.09}$	$1.82^{**}_{\pm 0.07}$	$(0.98^{**}_{\pm 0.11})$	$(1.61^{**}_{\pm 0.11})$	$(2.14^{**}_{\pm 0.09})$	$0.85^{**}_{\pm 0.14}$	$1.33^{**}_{\pm 0.09}$	$1.58^{**}_{\pm 0.06}$	$-0.02_{\pm 7.46}$	$8.91_{\pm 5.41}$	$20.62^{*}_{\pm 1.04}$
BOCS	$(0.29^{**}_{\pm 0.14})$	$(1.48^{**}_{\pm 0.09})$	$(1.96^{**}_{\pm 0.07})$	$(0.58^{**}_{\pm 0.12})$	$(1.83^{**}_{\pm 0.04})$	$(2.08^{**}_{\pm 0.03})$	$0.79^{**}_{\pm 0.11}$	$1.59^{**}_{\pm 0.04}$	$1.77^{**}_{\pm 0.02}$	$-9.37_{\pm 13.39}$	$1.58^{*}_{\pm 10.05}$	$18.04^{*}_{\pm 0.83}$
LFBO	$0.87^{**}_{\pm 0.19}$	$1.84^{**}_{\pm 0.21}$	$2.33^{**}_{\pm 0.21}$	$0.44^{**}_{\pm 0.16}$	$1.05^{**}_{\pm 0.14}$	$1.37^{**}_{\pm 0.14}$	$1.05^{**}_{\pm 0.07}$	$1.53^{**}_{\pm 0.06}$	$1.65^{**}_{\pm 0.05}$	$-9.50^{*}_{\pm 7.72}$	$17.43_{\pm 1.52}$	$20.32^{*}_{\pm 0.84}$
BORE	$1.13^{**}_{\pm 0.20}$	$1.96^{**}_{\pm 0.16}$	$2.34^{**}_{\pm 0.17}$	$0.44^{**}_{\pm 0.14}$	$1.07^{**}_{\pm 0.12}$	$1.41^{**}_{\pm 0.11}$	$1.01^{**}_{\pm 0.09}$	$1.40^{**}_{\pm 0.10}$	$1.54^{**}_{\pm 0.06}$	$-4.47_{\pm 6.76}$	$5.43^{*}_{\pm 6.95}$	$9.67^{*}_{\pm 7.46}$
others (13)	$-3.01^{(13)}_{\pm 0.74}$	$-0.70^{(13)}_{\pm 0.72}$	$0.45^{(13)}_{\pm 0.71}$	$-14.47^{(13)}_{\pm 1.21}$	$-13.13^{(13)}_{\pm 1.21}$	$-12.53^{(13)}_{\pm 1.19}$	$0.75^{(13)}_{\pm 0.23}$	$1.20^{(13)}_{\pm 0.28}$	$1.36^{(13)}_{\pm 0.31}$	$-50.20^{(12)}_{\pm 5.43}$	$-38.56^{(12)}_{\pm 5.09}$	$-29.94^{(12)}_{\pm 4.71}$

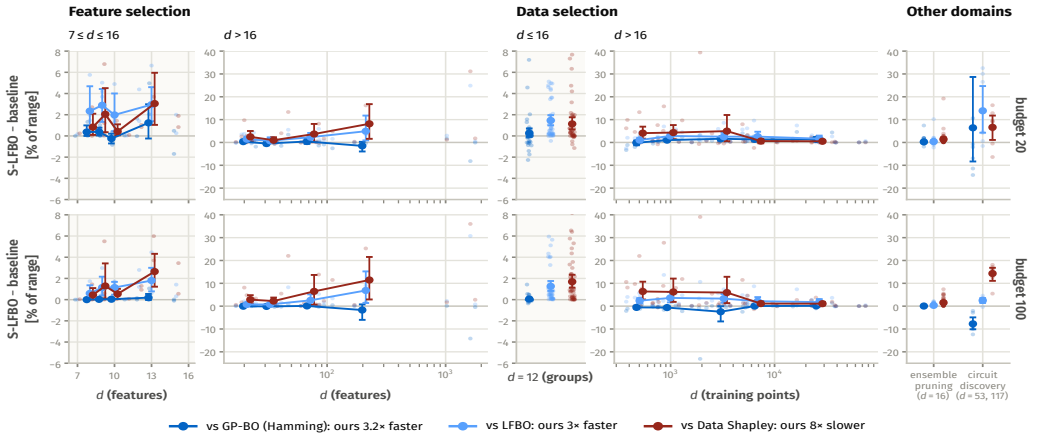

Figure 12: The lead of S-LFBO against the number of players d. S-LFBO minus GP-BO, LFBO and Data Shapley at 20 (top) and 100 queries (bottom), in % of the range between the empty set and the best subset (the optimum for $d \leq 16$, the best value any method found above). Faint points are datasets, with the LightGBM and TabPFN utilities of a dataset averaged; strong points are equal-count bins in d with 95% bootstrap intervals over datasets. The tinted panels ($d \leq 16$) have their own scale. The legend gives each baseline's runtime relative to S-LFBO, as the geometric mean over the utilities with $d > 16$ at both budgets.

F.3 Does the lead depend on the size of the utility?

Many of our utilities have $d \leq 16$, where 100 queries cover 39% of the subsets at $d = 8$ and still 0.15% at $d = 16$, so Figure 12 splits every domain at $d = 16$. In comparison to LFBO and Data Shapley, the lead of S-LFBO grows above $d = 16$: at 100 queries it is 2.8 and 5.9 points of the range in feature selection, against 1.2 and 1.4 below, and 2.7 and 4.2 in data selection over training points, against 1.3 and 1.7 over groups. Against GP-BO the gap stays within about one point at every size in feature and data selection. Above $d = 16$ it is significant only in data selection, where S-LFBO leads by 1.0 point at 20 queries ($p < .001$) and trails by 0.7 at 100 ($p = .01$). In circuit discovery, GP-BO leads by 7.7 points after 100 queries across all seven utilities. Where the two are level, S-LFBO takes a third of GP-BO's time, while Data Shapley runs about eight times faster than S-LFBO but trails it in every domain.

F.4 Does a property favour its own method?

Figure 13 asks whether the method built for a property closes on S-LFBO as a utility gains that property. For each utility, it divides the normalized performance $(f(S) - f(\emptyset))/(f^\star - f(\emptyset))$ of S-LFBO by that of the property's method. Here $f^\star$ is the optimum, or the best value found on a wide utility. Property scores are normalized within each domain, so that the bins do not simply sort utilities by domain. The gap closes for MTM, monotonicity and submodularity: Data Shapley, taking-everything, and double greedy approach S-LFBO as their property takes hold (Spearman $\rho = -.46$, $-.44$ and $-.53$ over utilities). BOCS matches S-LFBO at every pairwise score, and the additive curve shows no trend. The supermodular gap widens, but for all methods alike: min-cut, Data Shapley and double greedy fall behind at the same rate ($\rho = +.23$, $+.23$ and $+.36$). Within a domain the supermodular score mostly tracks the absence of submodularity ($\rho = -.88$), and no utility in the pool is supermodular.

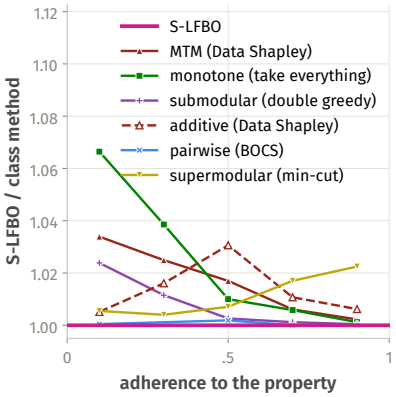

Figure 13: Three properties favor their own method. Median utility per bin, 100 queries.

F.5 Every query on eleven utilities

Figure 14 repeats the rasters of Figure 5 for selected utilities ranging from $d = 8$ to 1776. We describe the overall numbers below from all 44 LightGBM feature-selection utilities over three seeds. GP-BO changes 21 to 24% of the features per query at every $d > 16$. The step of S-LFBO follows the features the booster splits on:

- for $d \leq 60$, it splits on 94% and the step matches GP-BO's (22%);
- for $61 \leq d \leq 300$, it splits on 83%, the step grows to 33%, and S-LFBO is ahead on all 8 utilities ($p = 0.008$);
- above $d = 300$, it splits on 9% and the step falls to 13%.

Unsplit features change more often than split ones below $d = 300$ (0.55 and 0.46 against 0.18 and 0.29), and the chains find a strictly better leaf set in 82% of the middle-band rounds. Above $d = 300$, where five chains of 256 flips reach a tenth of the cube, half the rounds end where they started. On the enumerated utilities, both methods step across 37 to 41% of the cube, because never re-evaluating a subset pushes each proposal away from those already seen.

F.6 Per-utility results

Tables 9 to 12 break Table 2 down to its 321 utilities, one table per domain and one block per oracle, the utilities in order of size. The columns are the methods of Table 2 in the same order, and each cell is the same quantity: the method's rank percentile within the domain's field for that utility and budget, with ties broken by wall clock time, averaged over three seeds. *others* averages the methods Table 2 pools, utility by utility. The mean over a table's utilities is therefore the column of Table 2, and each block's mean row gives it per oracle.

Feature selection. S-LFBO has the lowest mean under both oracles at every budget, with GP-BO second: at 100 queries .181 against .209 under LightGBM and .160 against .203 under TabPFN.

Data selection. Under LightGBM S-LFBO leads at every budget (.130 against .153 for GP-BO at 100 queries). Under TabPFN it leads at 20 queries (.153 against .200), and from 50 queries GP-BO is slightly ahead (.147 against .154 at 100). GP-BO did not run on the five largest utilities under TabPFN, with 35,760 to 75,000 training points.

Ensemble pruning. GP-BO leads at 20 queries and S-LFBO at 50 and 100 among the methods of the table. At 100 queries, most methods reach the optimum of these 16-member ensembles, and hill climbing, pooled in *others*, ranks ahead of S-LFBO (.152 against .178) because it reaches it faster.

Circuit discovery. On the 7 utilities at $\lambda = 0.3$, S-LFBO leads at 20 queries and GP-BO at 50 and 100 (.074 against .113).

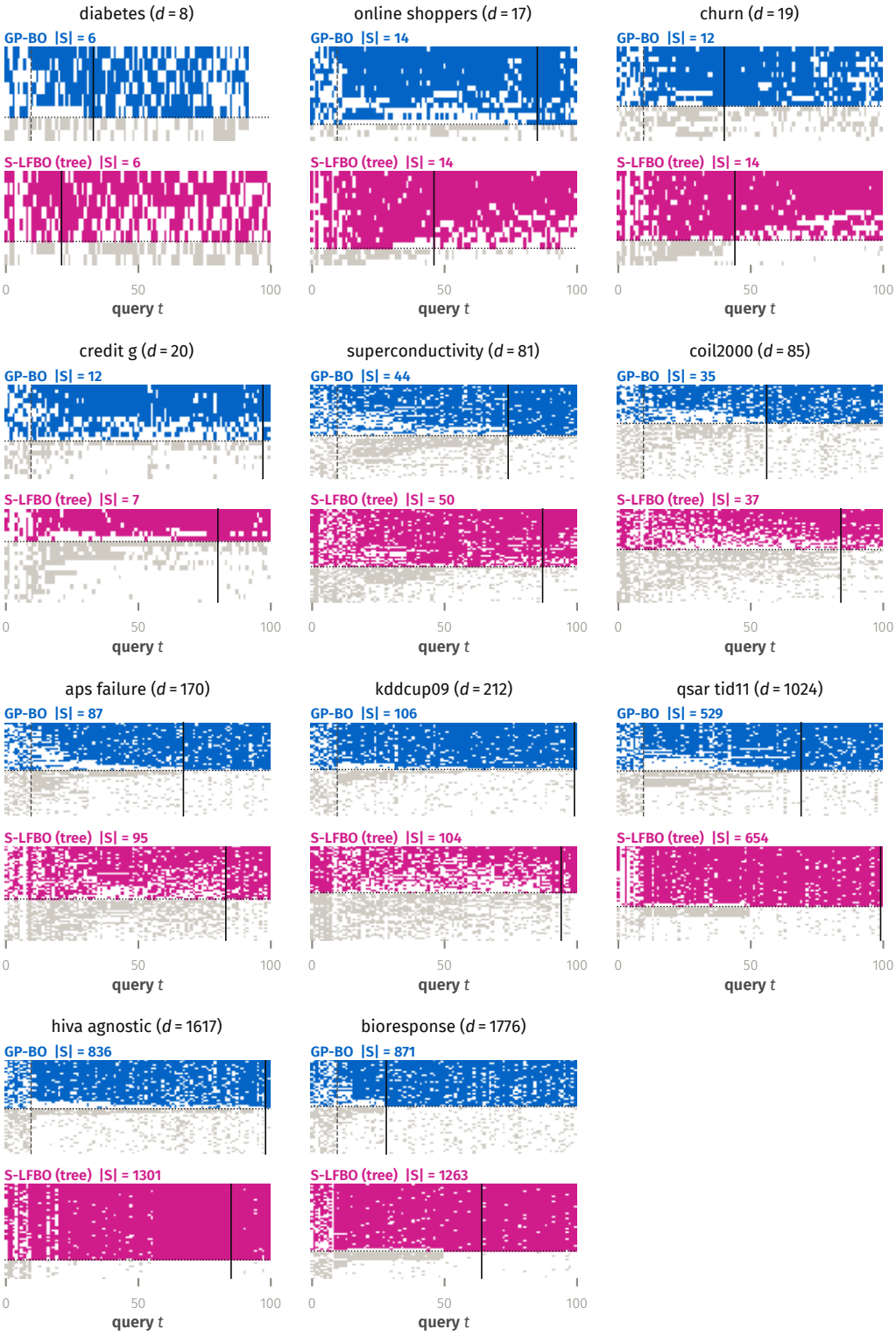

Figure 14: Every subset evaluated, eleven utilities. As Figure 5: GP-BO above, S-LFBO below, seed 0, 100 queries, LightGBM feature selection.

Table 9: Feature selection per utility at 20, 50 and 100 queries: rank percentile (lower is better) as in Table 2, averaged over ten seeds; *others* as there. Medals: each row's three lowest. Mean rows average the utilities of a block and of the table; the table's mean row is the column of Table 2. Parentheses, unmedalled: a method ran on fewer than 80% of the utilities. '–': no run.

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*	LightGBM Gini
mean, all utilities		20	**.196**	.376	.412	.802	.616	.235	.414	(.451)	.394	.363	.519	(.465)
		50	**.191**	.451	.574	.697	.474	.215	.422	(.387)	.412	.410	.526	(.666)
		100	**.171**	.515	.637	.657	.402	.206	.384	(.368)	.416	.414	.530	(.760)
LightGBM oracle (45 utilities)														
mean		20	**.197**	.385	.422	.793	.619	.226	.416	.443	.392	.360	.520	.465
		50	**.203**	.453	.600	.722	.463	.219	.414	.378	.400	.374	.529	.666
		100	**.181**	.514	.651	.706	.390	.209	.389	.346	.405	.393	.530	.760
fiat_500	7	20	**.083**	.788	.720	.462	**.083**	.197	.508	.348	.591	.712	.507	.864
		50	.379	.424	.659	.705	**.174**	.402	.508	.583	.667	.667	.478	.939
		100	.409	.409	.409	.833	**.205**	.477	.614	.553	.644	.598	.481	.970
concrete_str	8	20	**.174**	**.174**	**.174**	.818	.818	.371	.583	.644	.765	.629	.463	.174
		50	.583	**.273**	**.273**	**.273**	**.273**	.614	.750	.765	.871	.826	.468	.273
		100	.606	**.303**	**.303**	**.303**	**.303**	.674	.720	.765	.902	.856	.452	.303
diabetes	8	20	**.083**	.379	.250	.856	.735	.295	.417	.462	.689	.341	.506	.750
		50	.182	.508	.545	.924	**.083**	.220	.629	.265	.795	.523	.501	.894
		100	.227	.470	.697	.955	**.114**	.311	.326	.386	.705	.508	.502	.924
houses	8	20	**.068**	.470	.530	.364	.902	.167	.705	.455	.667	.364	.491	.530
		50	.205	.561	.561	.750	**.091**	.235	.629	.402	.780	.568	.494	.886
		100	.288	.591	.371	.795	**.144**	.356	.523	.432	.583	.689	.501	.871
amazon_access	9	20	**.159**	.674	.902	.409	.545	.174	.553	.659	.621	.348	.470	.985
		50	.220	.614	.977	.485	**.106**	.280	.402	.447	.773	.515	.492	1.000
		100	.288	.788	.962	.576	**.144**	.356	.492	.432	.689	.508	.466	1.000
diamonds	9	20	.333	.439	.455	.864	.250	.598	.705	.492	**.227**	**.227**	.519	.500
		50	.227	.583	.697	.932	**.197**	.477	.477	.523	.311	.326	.498	.773
		100	.167	.712	.674	.902	**.083**	.417	.538	.492	.371	.417	.492	.795
food_delivery	9	20	**.076**	.326	.326	.530	.811	.424	.311	.538	.455	.424	.527	.432
		50	.182	.432	.515	.818	**.083**	.280	.432	.265	.720	.659	.523	.742
		100	.212	.523	.576	.894	**.106**	.280	.508	.341	.644	.644	.499	.841
protein	9	20	**.045**	.348	.379	.311	.856	.265	.735	.523	.742	.712	.471	.523
		50	.205	.333	.500	.621	**.098**	.311	.826	.523	.727	.621	.495	.833
		100	.258	.455	.621	.803	**.129**	.326	.553	.386	.591	.765	.490	.932
website_phish	9	20	**.083**	.227	.318	.864	.780	.189	.689	.644	.538	.538	.478	.538
		50	.273	.318	.795	.780	**.129**	.311	.523	.371	.826	.826	.471	.864
		100	.379	**.189**	.795	.780	**.189**	.455	.561	.523	.811	.765	.456	.864
bank_churn	10	20	**.129**	.159	.561	.629	.750	.189	.508	.371	.712	.591	.502	.561
		50	.242	.242	.848	**.121**	**.121**	.311	.689	.417	.689	.750	.524	.902
		100	.348	**.174**	.659	**.174**	**.174**	.371	.606	.477	.811	.720	.526	.932
ecomm_shipping	10	20	.318	.182	.591	.477	.652	**.144**	.205	.235	.644	.598	.559	.614
		50	**.152**	.417	.833	.720	.409	.303	.508	.417	.795	.689	.455	.833
		100	**.227**	.348	.917	.841	.341	.250	.538	.341	.826	.750	.450	.917
give_me_credit	10	20	.348	.333	**.167**	.902	.705	.174	.583	.629	.568	.598	.488	.333
		50	.197	.348	.348	.508	**.023**	.114	.462	.250	.735	.856	.566	.705
		100	.136	.530	.530	.652	**.068**	.182	.477	.311	.811	.795	.511	.818
sdss17	11	20	.273	.523	.795	.924	.432	**.189**	.295	.326	.242	.303	.536	.795
		50	**.106**	.765	.856	.492	.545	.144	.402	.326	.371	.295	.523	.977
		100	.121	.720	.871	.735	.250	**.098**	.523	.174	.568	.462	.508	1.000
hr_analytics	12	20	**.212**	.545	.598	.826	.917	.250	.295	.402	.561	.424	.476	.712
		50	.197	.538	.811	**.129**	.402	.144	.523	.220	.644	.553	.541	.909
		100	.061	.697	.902	.242	**.030**	.205	.432	.447	.780	.386	.529	.970
wine_quality	12	20	**.091**	.364	.500	.917	.750	.326	.568	.235	.538	.417	.491	.500
		50	**.091**	.250	.485	.962	.265	.205	.402	.280	.705	.705	.519	.780
		100	**.091**	.424	.508	.977	.167	.114	.432	.341	.674	.735	.510	.856
bank_marketing	13	20	**.152**	.348	.424	.735	.856	.174	.462	.568	.583	.523	.487	.879
		50	.136	.508	.811	**.038**	.606	.341	.356	.341	.477	.583	.534	.985
		100	.197	.553	.856	**.068**	.523	.295	.371	.220	.341	.659	.547	1.000
good_customer	13	20	**.106**	.326	.356	.780	.644	.417	.492	.598	.417	.189	.522	.356
		50	.106	.417	.735	.909	.598	.220	.371	**.068**	.462	.386	.526	.735
		100	.091	.629	.795	.924	.424	**.068**	.326	.174	.614	.432	.509	.795
miami_housing	15	20	**.030**	.417	.432	.955	.735	.129	.205	.614	.098	.311	.547	.432
		50	**.061**	.811	.470	.780	.629	.068	.205	.311	.432	.462	.526	.894
		100	.136	.697	.515	.856	.114	**.098**	.311	.311	.553	.598	.534	.977
seismic_bumps	15	20	.258	.462	.689	.492	.417	**.174**	.311	.462	.341	.583	.544	.985
		50	.273	.076	.917	.795	.432	**.053**	.447	.311	.189	.417	.566	.985
		100	.136	.258	.947	.826	.212	**.068**	.553	.144	.205	.477	.562	.985
online_shoppers	17	20	.389	.380	.361	.806	.583	.231	.648	.444	**.176**	.324	.542	.315
		50	.278	.435	.611	.806	.657	**.157**	.241	.278	**.157**	**.157**	.552	.574

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*	LightGBM Gini
		100	**.037**	.472	.778	.917	.269	.120	.278	.148	.102	.157	.580	.778
churn	19	20	**.130**	.454	.454	.944	.639	.176	.426	.241	.380	.157	.521	.426
		50	.222	.546	.593	.657	.343	.231	.370	.167	**.102**	.176	.579	.759
		100	.352	.435	.426	.880	.269	.250	.333	.370	.250	**.176**	.562	.944
credit_g	20	20	.259	.713	.630	.731	.787	.213	.204	.352	**.194**	.269	.508	.824
		50	.333	.583	.926	**.083**	.639	.324	.519	.685	.241	.111	.585	.981
		100	.352	.676	.889	.120	.528	.398	.500	.370	.250	**.102**	.577	.981
airline_sat	21	20	**.130**	.194	.185	.944	.528	.157	.259	.481	.250	.398	.544	.148
		50	.111	.324	.537	.935	.620	**.083**	.648	.352	.176	.361	.542	.500
		100	**.056**	.361	.333	.787	.491	.102	.426	.352	.213	.269	.565	.593
jm1	21	20	**.148**	.398	.324	.833	.694	.157	.259	.463	.333	.389	.516	.315
		50	.204	.602	.731	**.083**	.565	.213	.333	.333	.213	.435	.574	.731
		100	**.056**	.750	.880	.213	.306	.231	.222	.278	.231	.176	.591	.880
credit_card	23	20	**.148**	.352	.278	.907	.398	.231	.185	.444	.287	.343	.540	.296
		50	**.111**	.417	.398	.824	.583	.194	.278	.296	.120	.157	.559	.519
		100	.111	.639	.759	.917	.287	**.065**	.278	.111	.102	.083	.576	.852
heloc	23	20	**.148**	.241	.241	.769	.565	.269	.519	.537	.157	.306	.558	.167
		50	.222	.222	.444	.898	.565	.194	.370	.333	**.046**	.139	.578	.389
		100	.093	.398	.676	.926	.269	**.065**	.333	.222	.139	.157	.587	.630
coupon_rec	24	20	**.130**	.213	.213	.944	.676	.176	.537	.463	.306	.361	.527	.185
		50	.296	.407	.481	.843	.750	**.194**	.370	.556	.213	.287	.525	.537
		100	.278	.472	.722	.898	.491	.194	.259	**.111**	.120	.306	.543	.759
marketing_cmp	25	20	.315	.315	.315	.907	.583	**.083**	.481	.370	.324	.231	.546	.278
		50	.130	.417	.620	.898	.546	**.046**	.370	.519	.287	.102	.559	.611
		100	.167	.472	.685	.954	.361	.120	.426	**.093**	.176	.102	.568	.778
hazelnut	30	20	**.148**	.213	.213	.926	.694	.324	.463	.444	.269	.176	.533	.167
		50	.241	.454	.556	.917	.806	.250	.574	.352	.176	**.065**	.528	.500
		100	.185	.602	.722	.935	.750	**.083**	.426	.333	.306	.102	.519	.722
students_drop	36	20	.259	.269	.259	.944	.491	.269	.241	.444	**.231**	.287	.542	.204
		50	**.111**	.454	.491	.843	.546	.139	.204	.296	.194	.120	.559	.426
		100	.056	.472	.537	**.046**	.491	.102	.315	.278	.213	.213	.622	.648
anneal	38	20	.333	**.130**	**.130**	.907	.676	.407	.389	.519	.444	.546	.491	.009
		50	.444	**.259**	.269	.907	.491	.472	.519	.667	.583	.639	.471	.009
		100	.491	.287	.241	**.185**	**.185**	.639	.620	.833	.694	.769	.520	.056
qsar_biodeg	41	20	.204	.676	.685	.944	.657	.306	.278	.352	**.120**	.157	.512	.667
		50	**.111**	.769	.806	.917	.750	.139	.148	.204	.157	.139	.511	.833
		100	**.037**	.787	.741	.574	.759	.102	.315	.241	.120	.176	.540	.926
diabetes130us	47	20	.593	.611	.741	.315	.398	**.120**	.667	.556	.435	.287	.569	.889
		50	.500	.769	.741	.528	.509	.120	.333	.296	.176	**.065**	.569	.926
		100	.222	.787	.833	.361	.639	**.083**	.278	.148	.176	.120	.569	.944
splice	60	20	.167	.176	**.157**	.944	.546	.176	.407	.537	.343	.417	.533	.148
		50	**.093**	.306	.278	.944	.676	.194	.315	.463	.269	.194	.535	.278
		100	.056	.231	.481	.880	.713	**.028**	.241	.407	.306	.287	.546	.481
bankruptcy	64	20	.278	.269	.222	.944	.750	**.028**	.296	.370	.306	.250	.538	.222
		50	.278	.472	.454	.907	.713	**.139**	.222	.481	.194	.250	.527	.444
		100	.111	.472	.481	.935	.694	**.046**	.130	.389	.213	.176	.539	.556
supercond	81	20	**.130**	.583	.565	.907	.509	.398	.315	.370	.204	.241	.513	.556
		50	**.000**	.602	.611	.944	.546	.194	.352	.315	.157	.102	.538	.574
		100	**.000**	.639	.648	.917	.565	.046	.148	.333	.139	.194	.542	.648
coil2000	85	20	.296	.426	.417	.870	.676	**.139**	.333	.389	.324	.176	.532	.389
		50	.111	.509	.602	.759	.787	.176	.241	.315	.417	**.046**	.525	.593
		100	.185	.676	.806	**.028**	.750	.287	.352	.481	.250	.324	.557	.796
naticusdroid	86	20	.167	.102	**.093**	.944	.602	.250	.426	.389	.380	.380	.529	.046
		50	.111	.176	.148	.944	.602	**.102**	.407	.463	.269	.361	.546	.074
		100	**.019**	.250	.241	.861	.639	.028	.204	.389	.231	.306	.557	.167
taiwanese_bnk	94	20	.148	.565	.500	.870	.694	.065	**.037**	.148	.176	.194	.536	.500
		50	**.093**	.398	.463	.889	.694	.176	.241	.389	.194	.231	.535	.648
		100	**.074**	.343	.611	.880	.750	.139	.148	.407	.250	.250	.532	.759
mic	111	20	.315	.509	.481	.889	.509	**.028**	.333	.185	.231	.176	.555	.481
		50	.111	.546	.519	.907	.528	**.102**	.259	.278	.213	.176	.544	.519
		100	**.074**	.657	.630	.935	.639	.083	.185	.278	.176	.231	.523	.630
aps_failure	170	20	.222	.472	.444	.926	.472	**.139**	.296	–	.324	.343	.527	.444
		50	**.111**	.602	.593	.944	.454	.120	.222	–	.324	.176	.532	.574
		100	**.056**	.639	.611	.954	.565	.065	.222	–	.269	.139	.531	.611
kddcup09	212	20	.259	.657	.639	.907	.694	.231	.352	–	.231	**.176**	.508	.630
		50	.278	.694	.704	.907	.731	**.176**	.333	–	.287	.213	.499	.704
		100	**.093**	.713	.722	.880	.824	.120	.296	–	.269	.213	.498	.722
qsar_tid11	1 024	20	.167	**.120**	–	.907	.583	.231	–	–	.380	.287	.541	.093
		50	.185	**.120**	–	.889	.620	.231	–	–	.324	.324	.541	.093
		100	.185	.176	–	.935	.639	**.157**	–	–	.324	.250	.543	.148
hiva_agnostic	1 617	20	.296	.676	–	.907	.398	**.176**	–	–	.306	**.176**	.530	.963
		50	.407	.528	–	.833	.472	.306	–	–	.287	**.213**	.530	.944

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*	LightGBM Gini
		100	.407	.731	–	.861	.509	**.306**	–	–	**.306**	.324	.497	.944
bioresponse	1 776	20	**.056**	.139	–	.741	.417	.324	–	–	.528	.343	.542	.111
		50	**.037**	.343	–	.741	.361	.157	–	–	.231	.361	.567	.315
		100	**.037**	.528	–	.843	.454	.120	–	–	.287	.306	.539	.500
TabPFN oracle (45 utilities)														
mean		20	**.195**	.366	.401	.810	.613	.244	.411	(.465)	.396	.366	.519	–
		50	**.179**	.450	.547	.673	.485	.211	.430	(.401)	.425	.445	.523	–
		100	**.160**	.515	.622	.609	.414	.203	.378	(.403)	.426	.436	.530	–
fiat_500	7	20	.341	.432	.432	**.098**	**.098**	.220	.705	.674	.606	.576	.551	–
		50	.424	.553	.553	**.189**	**.189**	.447	.720	.644	.720	.674	.485	–
		100	.485	.447	.447	**.242**	**.242**	.553	.795	.614	.780	.659	.478	–
concrete_str	8	20	.295	**.159**	**.159**	.841	.826	.356	.629	.614	.735	.598	.468	–
		50	.614	**.288**	**.288**	**.288**	**.288**	.644	.780	.705	.871	.856	.461	–
		100	.591	**.295**	**.295**	**.295**	**.295**	.659	.705	.811	.917	.841	.453	–
diabetes	8	20	**.114**	.417	.644	.447	.636	.280	.386	.492	.818	.606	.483	–
		50	.295	.606	.917	**.136**	.280	.341	.462	.417	.689	.841	.485	–
		100	.333	.553	.939	**.167**	**.167**	.402	.523	.477	.614	.750	.493	–
houses	8	20	**.068**	.235	.235	.856	.856	.295	.447	.614	.780	.644	.464	–
		50	.439	.371	**.205**	.962	**.205**	.477	.644	.553	.765	.886	.456	–
		100	.409	.371	**.205**	.962	**.205**	.477	.523	.659	.705	.765	.471	–
amazon_access	9	20	**.152**	.636	.750	.871	.674	.311	.447	.417	.492	.235	.474	–
		50	.280	.636	.932	.636	.152	**.114**	.477	.174	.568	.462	.527	–
		100	**.121**	.598	.947	.758	.189	.250	.417	.220	.553	.462	.508	–
diamonds	9	20	**.212**	.280	.280	.871	.273	.250	.462	.402	.591	.545	.542	–
		50	.114	.326	.250	.886	**.053**	.386	.644	.311	.553	.705	.530	–
		100	.197	.326	.326	.917	**.098**	.235	.674	.432	.598	.750	.511	–
food_delivery	9	20	**.053**	.295	.492	.189	.871	.205	.447	.538	.523	.462	.537	–
		50	.273	.273	.318	.576	**.129**	.311	.386	.386	.750	.689	.552	–
		100	.288	.326	.386	.674	**.144**	.356	.462	.432	.841	.765	.509	–
protein	9	20	**.098**	.333	.348	.432	.818	.250	.644	.598	.682	.652	.480	–
		50	.326	**.152**	.326	.780	**.152**	.371	.598	.447	.871	.795	.501	–
		100	.379	.341	**.189**	.841	**.189**	.455	.485	.538	.795	.780	.491	–
website_phish	9	20	**.068**	.197	.295	.886	.841	.250	.568	.523	.652	.652	.472	–
		50	.348	.280	.568	.568	**.167**	.386	.674	.447	.811	.871	.479	–
		100	.379	.333	.667	.667	**.189**	.447	.614	.523	.886	.886	.445	–
bank_churn	10	20	.144	**.091**	.515	.636	.811	.235	.629	.326	.689	.553	.501	–
		50	**.121**	.424	.803	.356	.258	.174	.614	.356	.644	.811	.505	–
		100	**.167**	.561	.917	.470	.212	.189	.258	.326	.674	.856	.503	–
ecomm_shipping	10	20	.477	.621	.629	**.121**	.311	.295	.432	.432	.583	.508	.544	–
		50	.258	.720	.576	.250	.250	.311	.583	**.174**	.795	.629	.516	–
		100	**.121**	.598	.939	.402	.189	.295	.295	.235	.598	.705	.519	–
give_me_credit	10	20	**.159**	.811	.924	.538	.659	**.159**	.598	.356	.439	.455	.466	–
		50	.182	.583	.977	**.091**	.197	.235	.553	.598	.674	.689	.492	–
		100	.288	.659	.977	**.144**	**.144**	.311	.500	.523	.644	.765	.487	–
sdss17	11	20	**.053**	.364	.591	.939	.598	.447	.417	.477	.394	.318	.497	–
		50	.152	.614	.917	**.076**	.288	.311	.598	.614	.795	.568	.478	–
		100	.242	.697	.879	**.121**	**.121**	.341	.477	.492	.826	.644	.492	–
hr_analytics	12	20	.144	**.061**	.333	.917	.826	.295	.508	.583	.379	.530	.505	–
		50	.197	**.076**	.621	.720	.341	.227	.553	.402	.659	.629	.521	–
		100	.182	**.076**	.780	.826	.447	.220	.523	.250	.538	.598	.519	–
wine_quality	12	20	.288	**.114**	**.114**	.902	.795	.447	.614	.417	.659	.614	.487	–
		50	.197	.303	.303	.659	**.182**	.371	.629	.568	.735	.856	.492	–
		100	.136	.477	.477	.803	**.068**	.159	.553	.280	.826	.795	.505	–
bank_marketing	13	20	**.167**	.417	.485	.924	.795	.250	.508	.356	.553	.553	.474	–
		50	.091	.485	.833	**.045**	.523	.114	.386	.174	.462	.765	.555	–
		100	.227	.629	.909	**.114**	**.114**	.250	.477	.326	.538	.674	.536	–
good_customer	13	20	.288	.758	.803	.962	.280	.311	.311	.508	**.174**	**.174**	.517	–
		50	.076	.735	.939	.386	.280	**.023**	.568	.250	.417	.311	.545	–
		100	**.076**	.712	.947	.591	.114	.098	.265	.295	.644	.477	.527	–
miami_housing	15	20	.136	.250	.265	.917	.629	.326	.341	.462	**.129**	.432	.558	–
		50	**.030**	.667	.515	.341	.606	.129	.386	.250	.098	.765	.557	–
		100	**.045**	.659	.636	.553	.144	.083	.341	.311	.235	.689	.565	–
seismic_bumps	15	20	.212	.288	.492	**.189**	.614	.280	.492	.432	.523	.250	.572	–
		50	**.182**	.439	.856	.197	.652	.280	.432	.189	.735	.326	.530	–
		100	**.182**	.826	.917	.356	.424	.356	.280	.205	.447	.341	.527	–
online_shoppers	17	20	.259	.509	.500	.954	.657	.231	.667	–	.287	**.213**	.512	–
		50	.204	.454	.778	.898	.694	**.157**	.500	–	**.157**	.231	.534	–
		100	**.037**	.546	.944	.398	.343	.157	.352	–	.167	.296	.591	–
churn	19	20	**.000**	.491	.519	.954	.583	.157	.296	–	.324	.380	.509	–
		50	**.019**	.287	.704	.269	.509	.139	.204	–	.306	.472	.585	–
		100	**.019**	.657	.574	.472	.102	.139	.167	–	.231	.287	.593	–

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*	LightGBM Gini
credit_g	20	20	**.037**	.380	.444	.861	.694	.120	.296	–	.361	.343	.520	–
		50	.056	.583	.741	.361	.787	**.028**	.296	–	.231	.194	.568	–
		100	.074	.657	.852	.435	.361	.083	.093	–	.250	**.065**	.596	–
airline_sat	21	20	.241	**.157**	.185	.954	.657	.306	.407	–	.306	.306	.526	–
		50	**.037**	.213	.250	.972	.639	.139	.519	–	.287	.454	.525	–
		100	**.056**	.417	.500	.806	.639	.120	.426	–	.120	.139	.563	–
jm1	21	20	.222	.454	.500	.935	.657	**.102**	.222	–	.287	.250	.526	–
		50	.111	.509	.333	.861	.546	**.046**	.352	–	.065	.250	.560	–
		100	**.019**	.472	.741	.935	.407	.028	.185	–	.083	.306	.564	–
credit_card	23	20	.167	.194	.463	.954	.528	**.028**	.315	–	.231	.435	.552	–
		50	**.019**	.380	.704	.880	.528	.102	.185	–	.120	.306	.557	–
		100	**.000**	.565	.944	.120	.306	.157	.241	–	.176	.157	.627	–
heloc	23	20	.148	.287	.315	.917	.602	.343	.519	–	**.102**	.213	.545	–
		50	**.019**	.231	.435	.843	.657	.231	.556	–	.120	.102	.568	–
		100	**.037**	.620	.593	.269	.565	.065	.315	–	.157	.139	.599	–
coupon_rec	24	20	.278	.213	**.185**	.954	.694	.213	.481	–	.417	.361	.512	–
		50	.111	.491	.537	.861	.787	**.028**	.519	–	.269	.324	.520	–
		100	.037	.546	.685	.491	.824	**.028**	.241	–	.176	.139	.566	–
marketing_cmp	25	20	.222	.435	.426	.954	.583	**.083**	.296	–	.194	.176	.548	–
		50	.056	.639	.759	.917	.620	**.028**	.315	–	.343	.361	.508	–
		100	**.019**	.602	.833	.324	.454	.028	.296	–	.287	.120	.599	–
hazelnut	30	20	.167	**.139**	.167	.935	.676	.287	.444	–	.269	.269	.535	–
		50	**.148**	.157	.185	.954	.787	.306	.389	–	.380	.343	.506	–
		100	**.037**	.361	.370	.954	.787	.065	.278	–	.509	.528	.483	–
students_drop	36	20	.241	**.139**	.167	.935	.491	.361	.167	.463	.250	.287	.545	–
		50	.148	.343	.333	.917	.620	**.083**	.444	.463	.194	.306	.545	–
		100	**.056**	.250	.556	.731	.565	.065	.352	.296	.157	.176	.588	–
anneal	38	20	.593	.472	.500	.917	.639	**.213**	.583	.472	.519	.444	.483	–
		50	.463	.444	.630	.954	.361	**.343**	.444	.722	.602	.556	.478	–
		100	.537	.509	.722	.194	**.185**	.565	.630	.889	.713	.713	.510	–
qsar_biodeg	41	20	.278	.620	.704	.935	.472	**.120**	.222	.370	.269	.139	.537	–
		50	.185	.657	.722	.972	.676	.343	.315	.315	.213	**.157**	.498	–
		100	**.037**	.731	.759	.861	.546	.102	.407	.167	.157	.250	.529	–
diabetes130us	47	20	**.074**	.287	.222	.843	.435	.380	.241	.093	.083	.222	.563	–
		50	**.093**	.435	.519	.806	.583	.102	.185	.167	.102	.194	.565	–
		100	.093	.509	.778	.546	.602	**.028**	.111	.296	.102	.157	.588	–
splice	60	20	.352	.139	**.130**	.954	.528	.287	.407	.444	.435	.398	.522	–
		50	.185	**.157**	.185	.954	.657	.213	.185	.481	.231	.269	.543	–
		100	.130	.176	.241	.917	.694	**.028**	.222	.370	.231	.324	.552	–
bankruptcy	64	20	.148	**.120**	.148	.954	.657	.157	.241	.574	.361	.269	.540	–
		50	**.093**	.250	.389	.954	.676	.120	.204	.222	.102	.139	.563	–
		100	.111	.324	.481	.917	.676	**.046**	.204	.111	.139	.157	.562	–
supercond	81	20	**.093**	.620	.593	.954	.546	.194	.259	–	.102	.287	.525	–
		50	**.074**	.657	.648	.954	.602	.139	.111	–	.083	.102	.539	–
		100	**.074**	.546	.722	.917	.602	.120	.222	–	.176	.120	.543	–
coil2000	85	20	**.056**	.602	.556	.917	.620	.157	.148	–	.324	.157	.519	–
		50	**.037**	.713	.722	.880	.731	.065	.204	–	.213	.157	.520	–
		100	.093	.731	.407	.731	.676	**.065**	.259	–	.269	.231	.525	–
naticusdroid	86	20	.167	.139	**.056**	.954	.657	.213	.463	–	.398	.324	.526	–
		50	.204	.157	.148	.954	.620	**.139**	.315	–	.343	.306	.540	–
		100	.056	.194	.222	.880	.731	**.028**	.278	–	.324	.324	.543	–
taiwanese_bnk	94	20	.333	.657	.611	.954	.731	**.231**	.259	–	.343	.269	.485	–
		50	.259	.676	.611	.954	.824	**.194**	.389	–	.306	.306	.478	–
		100	**.056**	.694	.370	.713	.898	.250	.444	–	.213	.250	.498	–
mic	111	20	**.074**	.435	.426	.917	.380	.083	.167	–	.306	.139	.562	–
		50	.056	.602	.574	.954	.528	**.046**	.333	–	.231	.139	.542	–
		100	.037	.676	.574	.954	.620	**.028**	.389	–	.213	.250	.522	–
aps_failure	170	20	.278	.583	**.037**	.917	.417	.046	.241	–	.287	.231	.543	–
		50	.093	.620	**.000**	.917	.528	.046	.185	–	.231	.157	.542	–
		100	.130	.750	**.019**	.935	.639	.139	.259	–	.287	.231	.502	–
kddcup09	212	20	.259	.343	.222	.954	.731	.343	.333	–	.157	**.148**	.527	–
		50	.148	.491	.370	.917	.769	**.120**	.241	–	.213	.287	.517	–
		100	**.037**	.583	.463	.880	.824	.065	.352	–	.250	.250	.512	–
qsar_tid11	1 024	20	**.148**	.324	–	.898	.639	.269	–	–	.324	.250	.525	–
		50	**.074**	.417	–	.898	.639	.194	–	–	.380	.213	.522	–
		100	**.093**	.491	–	.880	.639	.102	–	–	.306	.213	.531	–
hiva_agnostic	1 617	20	**.111**	.343	–	.769	.565	.380	–	–	.333	.463	.512	–
		50	**.167**	.509	–	.528	.713	.380	–	–	.602	.343	.494	–
		100	**.204**	.435	–	.343	.676	.472	–	–	.639	.472	.497	–
bioresponse	1 776	20	.352	.639	–	.769	.509	.231	–	–	**.157**	**.157**	.545	–
		50	.389	.639	–	.806	.565	**.120**	–	–	.176	.231	.539	–
		100	.259	.639	–	.898	.583	**.046**	–	–	.176	.120	.545	–

Table 10: Data selection per utility at 20, 50 and 100 queries: rank percentile (lower is better) as in Table 2, averaged over ten seeds; *others* as there. Medals: each row's three lowest. Mean rows average the utilities of a block and of the table; the table's mean row is the column of Table 2. Parentheses, unmedalled: a method ran on fewer than 80% of the utilities. '–': no run.

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
mean, all utilities		20	**.160**	.313	(.466)	.834	.590	.222	(.387)	(.471)	.358	.340	.535
		50	**.146**	.401	(.598)	.762	.510	.165	(.462)	(.313)	.387	.389	.533
		100	**.144**	.445	(.617)	.778	.415	.150	(.415)	(.310)	.389	.388	.537
LightGBM oracle (76 utilities)													
mean		20	**.168**	.332	(.506)	.817	.560	.248	(.351)	(.450)	.348	.341	.536
		50	**.138**	.395	(.589)	.762	.509	.188	(.435)	(.280)	.380	.374	.538
		100	**.130**	.421	(.609)	.758	.444	.153	(.365)	(.260)	.383	.376	.546
airfoil_noise	12	20	**.091**	**.091**	**.091**	.955	.780	.250	.583	.523	.447	.583	.515
		50	.439	**.220**	**.220**	.932	.735	.470	.705	.568	.674	.720	.443
		100	.530	**.265**	**.265**	**.265**	**.265**	.553	.735	.644	.750	.811	.496
amazon_access	12	20	**.136**	.144	.394	.902	.765	.508	.311	.371	.409	.439	.520
		50	.106	**.098**	.652	.939	.356	.197	.598	.114	.644	.508	.530
		100	.045	.280	.720	.962	**.023**	.083	.402	.129	.492	.553	.566
bank_churn	12	20	**.121**	.492	.841	.917	.402	.432	.326	.402	.205	.356	.510
		50	.121	.621	.902	.212	**.098**	.311	.341	.326	.523	.568	.546
		100	.091	.705	.932	.333	**.045**	.189	.341	.235	.598	.492	.548
bank_marketing	12	20	.348	.508	.886	.879	.644	.341	.371	.553	.447	**.326**	.465
		50	.197	.311	.811	.955	.492	**.129**	.614	.159	.553	.583	.492
		100	**.136**	.818	.780	.955	.197	.189	.538	.189	.371	.417	.503
blood_transf	12	20	.136	.598	.765	.932	.629	**.083**	.417	.356	.356	.538	.487
		50	**.061**	.508	.750	.955	.379	.159	.295	.280	.341	.371	.536
		100	**.167**	.500	.886	.955	.379	.250	.311	.189	.386	.432	.516
concrete_str	12	20	**.030**	.136	.174	.932	.780	.220	.432	.553	.462	.598	.516
		50	.242	.371	.530	**.121**	.780	.265	.598	.341	.636	.871	.499
		100	.318	.674	.515	**.159**	**.159**	.341	.523	.462	.508	.856	.523
diabetes	12	20	**.159**	.750	.765	.932	.606	.220	.280	.371	.386	.189	.500
		50	**.106**	.765	.795	.955	.326	.114	.371	.295	.432	.311	.510
		100	.136	.705	.652	.955	.205	**.114**	.235	.189	.553	.386	.539
diamonds	12	20	.174	**.152**	**.152**	.955	.720	.341	.417	.523	.614	.553	.506
		50	**.212**	.424	.523	.826	.780	.242	.553	.523	.583	.614	.456
		100	.288	.409	.652	**.144**	.242	.311	.477	.402	.689	.629	.542
ecomm_shipping	12	20	**.030**	.280	.485	.765	.917	.311	.538	.311	.462	.417	.501
		50	**.061**	.477	.735	.886	.697	.083	.265	.174	.462	.598	.509
		100	.152	.727	.674	.902	.159	**.053**	.265	.159	.545	.455	.543
fiat_500	12	20	.288	.735	.659	.864	.553	**.189**	.280	.356	.341	.341	.514
		50	.258	.841	.417	.417	.295	**.159**	.492	.189	.750	.568	.529
		100	.136	.902	.417	.455	.280	**.129**	.523	.159	.568	.538	.541
fitness_club	12	20	.273	.659	.917	.235	.583	**.038**	.492	.553	.462	.614	.496
		50	.121	.485	.629	.553	**.061**	.174	.841	.235	.295	.568	.551
		100	.182	.508	.583	.689	**.091**	.205	.644	.295	.402	.735	.527
food_delivery	12	20	.136	**.091**	**.091**	.917	.629	.295	.447	.417	.598	.402	.547
		50	.197	**.152**	.295	.962	.356	.220	.614	.447	.629	.614	.516
		100	**.061**	.326	.462	.962	.227	.083	.477	.265	.629	.705	.528
give_me_credit	12	20	.364	.462	**.038**	.667	.644	.220	.265	.568	.598	.432	.547
		50	.318	.902	.152	.477	.530	.098	.432	**.068**	.689	.720	.533
		100	.273	.682	.235	.591	.288	**.068**	.311	.144	.780	.795	.547
good_customer	12	20	.462	.644	.932	.636	.280	**.205**	.523	.553	.470	.470	.484
		50	.242	.750	.886	.606	**.114**	.356	.538	.402	.568	.174	.508
		100	.485	.553	.727	.727	.265	.508	.492	**.235**	.598	**.235**	.512
health_ins	12	20	**.250**	.606	.682	.902	.477	.356	.295	.280	.424	.288	.515
		50	.242	.848	.773	.977	.492	**.159**	.280	.250	.523	.477	.478
		100	.106	.667	.803	.977	.144	**.083**	.326	.189	.508	.538	.520
houses	12	20	.220	.341	.341	.932	.841	**.053**	.311	.553	.523	.568	.503
		50	**.136**	.705	.780	.227	.689	.159	.583	.250	.341	.765	.500
		100	.152	.803	.538	.386	**.076**	.174	.386	.356	.523	.659	.546
hr_analytics	12	20	.303	.591	.758	.894	.356	**.098**	.477	.356	.341	.447	.514
		50	**.000**	.826	.795	.970	.212	.129	.477	.295	.545	.545	.477
		100	**.015**	.826	.674	.970	.227	.083	.159	.205	.705	.311	.526
maternal_health	12	20	.295	.333	.386	.932	.720	**.174**	.235	.326	.386	.477	.541
		50	.227	.462	.364	**.068**	.447	.341	.598	.417	.636	.735	.533
		100	.167	.621	.530	**.083**	.318	.341	.462	.356	.614	.689	.537
miami_housing	12	20	.364	.538	.689	.932	.598	**.265**	.447	**.265**	.295	.341	.510
		50	.303	.432	.477	.917	.212	**.129**	.432	.205	.432	.447	.563
		100	.121	.144	.538	.947	**.053**	.174	.432	.174	.462	.432	.588
protein	12	20	.333	**.159**	.326	.932	.826	.235	.371	.568	.492	.659	.495
		50	.152	.265	.614	**.076**	.826	.174	.250	.295	.727	.742	.541

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
		100	.258	.318	.750	**.129**	**.129**	.280	.326	.447	.644	.705	.559
qsar_fish	12	20	.439	.530	.553	.886	.159	**.053**	.417	.492	.250	.280	.568
		50	.182	.614	.727	.947	**.121**	.144	.295	.144	.568	.492	.534
		100	**.030**	.697	.735	.947	.152	.129	.341	.205	.432	.568	.523
sdss17	12	20	.265	.629	.720	.924	.523	.159	.371	.523	.250	**.144**	.520
		50	.439	.917	.902	.735	**.220**	.265	.477	.295	.303	.394	.499
		100	**.121**	.939	.856	.788	.144	.295	.341	.159	.417	.250	.524
seismic_bumps	12	20	**.152**	.689	.780	.864	.583	.280	.159	.402	.409	.333	.500
		50	**.045**	.583	.886	.886	.470	.159	.386	.220	.356	.402	.512
		100	**.045**	.561	.871	.902	.409	.159	.265	.220	.432	.462	.517
website_phish	12	20	**.068**	.515	.500	.932	.538	.220	.250	.583	.318	.318	.525
		50	**.106**	.659	.629	.705	.295	.220	.417	.205	.508	.477	.530
		100	.182	.833	.886	.811	**.091**	.205	.326	.326	.568	.553	.492
wine_quality	12	20	.197	.295	.394	.902	.780	**.083**	.311	.492	.583	.568	.507
		50	**.212**	.508	.477	.970	.598	.235	.386	.295	.402	.750	.491
		100	.242	.530	.591	.970	.273	.295	.432	**.174**	.492	.780	.497
blood_transf	374	20	.275	.578	.578	.824	.500	.127	.118	–	.245	**.069**	.560
		50	.137	.608	.608	.824	.500	.108	.118	–	.147	**.049**	.570
		100	.098	.520	.431	.824	.618	**.029**	.176	–	.147	.108	.569
diabetes	384	20	.176	.618	.618	.824	.480	**.049**	.235	–	.147	.147	.567
		50	.078	.618	.618	.824	.480	**.029**	.157	–	.225	.108	.565
		100	.059	.324	.647	.824	.539	**.029**	.176	–	.167	.088	.594
anneal	450	20	.137	**.029**	.049	.804	.618	.186	.255	–	.402	.422	.549
		50	.176	**.039**	.059	.794	.618	.382	.392	–	.225	.245	.563
		100	.255	**.147**	.157	.794	.618	.363	.353	–	.382	.284	.537
qsar_fish	453	20	.176	.402	.402	.824	.422	**.108**	.294	–	.265	.284	.563
		50	.078	.539	.539	.814	.441	**.029**	.392	–	.206	.304	.561
		100	.078	.578	.578	.755	.461	**.049**	.216	–	.225	.108	.575
credit_g	500	20	.216	.618	.618	.824	.500	**.147**	.373	–	.167	.284	.542
		50	.137	.618	.618	.824	.500	**.108**	.137	–	.245	.167	.549
		100	**.020**	.618	.637	.824	.520	.049	.059	–	.147	.108	.567
maternal_health	507	20	.118	**.088**	**.088**	.804	.618	.245	.275	–	.206	.324	.565
		50	**.078**	.088	.088	.794	.618	.245	.451	–	.206	.343	.564
		100	.118	.147	.147	.794	.618	**.029**	.275	–	.245	.304	.578
concrete_str	515	20	**.078**	.088	–	.814	.480	.304	–	–	.206	.480	.557
		50	**.059**	.088	–	.824	.500	.206	–	–	.225	.304	.578
		100	**.039**	.088	–	.814	.500	.167	–	–	.225	.245	.587
qsar_biodeg	528	20	.294	.578	–	.824	.461	**.147**	–	–	.167	.186	.558
		50	.176	.598	–	.814	.500	**.029**	–	–	.206	.265	.553
		100	.098	.598	–	.804	.520	**.049**	–	–	.108	.324	.554
health_ins	669	20	**.176**	.500	–	.853	.618	.186	–	–	.304	.363	.516
		50	.098	.500	–	.784	.618	**.069**	–	–	.265	.225	.549
		100	.157	.578	–	.824	.441	**.049**	–	–	.147	.108	.578
website_phish	677	20	**.020**	.206	–	.824	.618	.127	–	–	.343	.245	.558
		50	**.000**	.402	–	.814	.618	.167	–	–	.343	.147	.546
		100	**.000**	.500	–	.794	.618	.069	–	–	.284	.088	.559
fitness_club	750	20	.196	.618	–	.824	.441	.265	–	–	.147	**.127**	.553
		50	.118	.618	–	.814	.500	**.108**	–	–	.265	.127	.552
		100	.118	.265	–	.814	.578	**.049**	–	–	.265	.324	.564
airfoil_noise	752	20	.098	**.088**	–	.765	.422	.206	–	–	.343	.382	.572
		50	**.078**	.088	–	.804	.480	.147	–	–	.363	.402	.565
		100	**.078**	.088	–	.745	.618	.186	–	–	.324	.284	.569
fiat_500	769	20	**.118**	.598	–	.824	.461	.245	–	–	.255	.196	.538
		50	.157	.618	–	.824	.500	**.069**	–	–	**.069**	.343	.553
		100	.118	.618	–	.824	.520	**.049**	–	–	.147	.225	.556
mic	850	20	.255	.578	–	.794	.598	.304	–	–	.225	**.206**	.527
		50	.157	.578	–	.794	.598	.186	–	–	**.147**	.167	.548
		100	.137	.578	–	.814	.598	**.108**	–	–	.186	.225	.545
good_customer	861	20	.275	.520	–	.824	.461	.147	–	–	**.108**	.147	.572
		50	.118	.578	–	.814	.461	**.108**	–	–	.127	.167	.568
		100	.118	.578	–	.794	.520	**.029**	–	–	.127	.167	.571
marketing_cmp	1 120	20	**.078**	.167	–	.804	.598	.147	–	–	.490	.275	.547
		50	**.059**	.225	–	.794	.618	.088	–	–	.324	.225	.566
		100	**.098**	.343	–	.794	.618	.127	–	–	.225	.304	.553
hazelnut	1 200	20	.098	**.088**	–	.794	.520	.127	–	–	.343	.520	.555
		50	**.039**	.127	–	.794	.559	.108	–	–	.225	.343	.577
		100	**.000**	.167	–	.794	.598	.088	–	–	.265	.304	.571
seismic_bumps	1 292	20	**.059**	.539	–	.814	.618	.225	–	–	.324	.167	.529
		50	.137	.578	–	.794	.598	**.088**	–	–	.324	.245	.534
		100	**.098**	.559	–	.794	.618	.108	–	–	.324	.127	.543
splice	1 596	20	**.039**	.480	–	.794	.598	.069	–	–	.324	.245	.545
		50	.078	.578	–	.794	.598	.245	–	–	.245	**.069**	.543

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
		100	.118	.578	–	.794	.598	**.029**	–	–	.245	.167	.553
bioresponse	1 876	20	.098	**.069**	–	.794	.618	.422	–	–	.343	.382	.534
		50	**.078**	.088	–	.794	.618	.284	–	–	.382	.324	.546
		100	**.059**	.088	–	.794	.618	.167	–	–	.363	.402	.552
hiva_agnostic	1 923	20	.294	.578	–	.755	.696	.206	–	–	**.167**	.265	.530
		50	.098	.578	–	.755	.716	**.049**	–	–	.147	.127	.557
		100	.078	.559	–	.755	.735	**.029**	–	–	.186	.127	.555
students_drop	2 212	20	**.078**	.206	–	.794	.598	.422	–	–	.324	.265	.536
		50	**.098**	.422	–	.794	.618	.147	–	–	.324	.265	.539
		100	.039	.520	–	.794	.618	**.029**	–	–	.324	.324	.536
churn	2 500	20	.235	.500	–	.794	.559	**.069**	–	–	.265	.382	.539
		50	.137	.578	–	.794	.598	**.088**	–	–	.324	.402	.520
		100	**.078**	.578	–	.794	.598	.088	–	–	.284	.402	.523
qsar_tid11	2 871	20	**.059**	.088	–	.794	.618	.382	–	–	.363	.265	.545
		50	**.059**	.088	–	.794	.618	.186	–	–	.402	.284	.557
		100	**.020**	.088	–	.794	.618	.147	–	–	.382	.245	.566
bankruptcy	2 955	20	.333	.363	–	.794	.598	.402	–	–	**.275**	.333	.521
		50	**.059**	.539	–	.794	.618	.245	–	–	.343	.127	.530
		100	.039	.578	–	.794	.598	**.029**	–	–	.304	.245	.541
wine_quality	3 249	20	**.039**	.088	–	.804	.461	.343	–	–	.441	.363	.545
		50	**.039**	.088	–	.824	.461	.343	–	–	.363	.284	.558
		100	**.000**	.088	–	.824	.500	.284	–	–	.382	.324	.554
taiwanese_bnk	3 410	20	.196	.598	–	.794	.539	.245	–	–	.206	**.108**	.546
		50	.314	.598	–	.794	.539	**.088**	–	–	.108	.265	.555
		100	.275	.598	–	.794	.559	**.049**	–	–	.147	.265	.553
naticusdroid	3 746	20	**.000**	.206	–	.794	.618	.304	–	–	.363	.245	.543
		50	**.039**	.304	–	.794	.618	.108	–	–	.422	.363	.536
		100	**.039**	.324	–	.814	.618	.088	–	–	.363	.343	.541
coil2000	4 911	20	**.196**	.578	–	.794	.441	.245	–	–	.245	.245	.541
		50	.275	.578	–	.794	.441	**.049**	–	–	.225	.245	.561
		100	.216	.598	–	.794	.461	**.029**	–	–	.343	.245	.548
bank_churn	5 000	20	.137	**.088**	–	.794	.618	.265	–	–	.422	.284	.548
		50	.118	.137	–	.794	.618	**.049**	–	–	.500	.265	.558
		100	.098	.147	–	.794	.618	**.029**	–	–	.441	.324	.559
heloc	5 230	20	.176	**.088**	–	.794	.461	.284	–	–	.441	.284	.559
		50	.196	**.167**	–	.794	.520	.245	–	–	.343	.284	.559
		100	**.157**	.245	–	.794	.598	.245	–	–	.363	.363	.536
jm1	5 443	20	**.039**	.108	–	.794	.618	.127	–	–	.382	.422	.550
		50	**.039**	.127	–	.794	.618	.108	–	–	.461	.363	.548
		100	**.059**	.127	–	.794	.618	.088	–	–	.343	.382	.559
ecomm_shipping	5 499	20	.255	.578	–	.794	.539	**.127**	–	–	.382	.324	.523
		50	.235	.578	–	.794	.500	**.069**	–	–	.225	.186	.559
		100	.255	.598	–	.794	.500	**.088**	–	–	.245	.245	.548
online_shoppers	6 165	20	.294	**.206**	–	.794	.598	.245	–	–	.363	.363	.539
		50	.275	.265	–	.794	.598	**.147**	–	–	.402	.402	.536
		100	.353	.402	–	.794	.598	**.147**	–	–	.441	.324	.527
coupon_rec	6 342	20	**.078**	.088	–	.794	.559	.363	–	–	.402	.324	.543
		50	**.078**	.088	–	.794	.598	.284	–	–	.245	.324	.561
		100	**.078**	.088	–	.794	.598	.167	–	–	.422	.324	.555
miami_housing	6 888	20	.098	**.088**	–	.706	.559	.343	–	–	.363	.314	.557
		50	**.078**	.088	–	.716	.578	.382	–	–	.422	.284	.548
		100	**.078**	.127	–	.725	.598	.225	–	–	.422	.363	.549
hr_analytics	9 579	20	**.039**	.225	–	.794	.618	.382	–	–	.363	.206	.537
		50	**.039**	.304	–	.794	.618	.245	–	–	.461	.265	.529
		100	**.059**	.343	–	.794	.618	.245	–	–	.304	.324	.534
houses	10 320	20	**.039**	.088	–	.745	.461	.382	–	–	.265	.382	.561
		50	**.039**	.088	–	.804	.480	.284	–	–	.265	.265	.574
		100	**.000**	.147	–	.843	.520	.127	–	–	.265	.382	.565
supercond	10 632	20	**.078**	.088	–	.794	.578	.284	–	–	.343	.363	.550
		50	**.059**	.088	–	.794	.618	.245	–	–	.441	.402	.537
		100	**.059**	.088	–	.824	.618	.167	–	–	.441	.382	.544
credit_card	15 000	20	.098	**.088**	–	.794	.500	.382	–	–	.500	.382	.532
		50	.157	**.147**	–	.794	.539	.324	–	–	.402	.500	.527
		100	.196	.186	–	.814	.539	**.069**	–	–	.363	.422	.555
amazon_access	16 385	20	**.059**	.088	–	.794	.578	.422	–	–	.402	.304	.537
		50	**.059**	.088	–	.794	.539	.284	–	–	.539	.382	.534
		100	**.059**	.108	–	.794	.578	.225	–	–	.363	.304	.557
bank_marketing	22 605	20	**.039**	.108	–	.794	.461	.480	–	–	.422	.363	.534
		50	**.059**	.186	–	.794	.480	.402	–	–	.441	.422	.525
		100	**.059**	.284	–	.794	.461	.127	–	–	.520	.500	.529
food_delivery	22 725	20	.118	**.088**	–	.794	.480	.343	–	–	.461	.265	.552
		50	**.078**	.088	–	.804	.520	.245	–	–	.520	.343	.544

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
		100	**.078**	.088	–	.814	.578	.147	–	–	.343	.382	.559
protein	22 865	20	**.039**	.088	–	.794	.324	.324	–	–	.422	.363	.562
		50	**.039**	.108	–	.814	.402	.186	–	–	.363	.500	.557
		100	**.039**	.108	–	.814	.441	.147	–	–	.402	.441	.559
kddcup09	25 000	20	.255	.304	–	.794	.441	.402	–	–	**.108**	.304	.559
		50	.275	.480	–	.794	.480	.324	–	–	**.245**	.284	.536
		100	.333	.539	–	.794	.539	**.284**	–	–	**.284**	.382	.516
diamonds	26 970	20	.216	**.088**	–	.706	.598	.284	–	–	.225	.363	.567
		50	**.059**	.088	–	.706	.618	.225	–	–	.265	.402	.563
		100	**.039**	.088	–	.716	.618	.186	–	–	.343	.422	.557
diabetes130us	35 760	20	**.000**	.088	–	.794	.500	.480	–	–	.245	.304	.553
		50	**.000**	.108	–	.794	.559	.225	–	–	.382	.441	.545
		100	**.000**	.206	–	.814	.578	.167	–	–	.461	.245	.548
aps_failure	38 001	20	.235	.559	–	.794	.578	.304	–	–	.324	**.206**	.521
		50	.275	.578	–	.794	.500	**.206**	–	–	.284	.382	.523
		100	.294	.578	–	.794	.500	**.225**	–	–	.324	.324	.523
sdss17	39 027	20	.353	.284	–	.794	.343	**.167**	–	–	.461	.402	.550
		50	.353	.382	–	.794	.461	**.088**	–	–	.480	.225	.552
		100	.431	.441	–	.794	.441	**.088**	–	–	.480	.186	.552
airline_sat	64 941	20	**.039**	.088	–	.794	.539	.284	–	–	.186	.382	.566
		50	**.020**	.088	–	.794	.598	.265	–	–	.225	.363	.561
		100	**.020**	.088	–	.794	.618	.284	–	–	.167	.441	.555
give_me_credit	75 000	20	**.059**	.088	–	.794	.500	.167	–	–	.441	.422	.553
		50	**.078**	.088	–	.794	.520	.225	–	–	.402	.186	.571
		100	**.078**	.088	–	.794	.500	.225	–	–	.422	.284	.562
TabPFN oracle (97 utilities)													
mean		20	**.153**	.297	(.443)	.846	.613	.200	(.408)	(.482)	.366	.340	.535
		50	.153	.406	(.604)	.762	.510	**.147**	(.478)	(.331)	.392	.401	.529
		100	.154	.464	(.622)	.793	.392	**.147**	(.444)	(.337)	.395	.398	.531
airfoil_noise	12	20	**.083**	**.083**	**.083**	.955	.750	.280	.477	.614	.371	.568	.524
		50	.379	**.189**	**.189**	.689	.720	.402	.735	.523	.674	.735	.473
		100	.455	**.227**	**.227**	.833	**.227**	.477	.644	.583	.765	.629	.491
airline_sat	12	20	.273	**.098**	**.098**	.932	.856	.250	.477	.538	.447	.674	.510
		50	.455	**.227**	**.227**	**.227**	.652	.477	.735	.629	.720	.841	.482
		100	.515	**.258**	**.258**	**.258**	**.258**	.538	.674	.598	.765	.886	.501
amazon_access	12	20	.083	.159	**.038**	.894	.750	.235	.250	.629	.371	.341	.564
		50	.333	.439	**.114**	.917	.689	.265	.644	.311	.614	.674	.487
		100	.318	.288	**.159**	.932	**.159**	.341	.644	.417	.689	.705	.513
anneal	12	20	**.311**	**.311**	**.311**	.939	**.311**	.742	.795	.742	.697	.727	.417
		50	**.303**	**.303**	**.303**	.947	**.303**	.712	.811	.856	.682	.712	.413
		100	**.311**	**.311**	**.311**	.962	**.311**	.720	.826	.871	.682	.667	.411
bank_churn	12	20	**.023**	.447	.591	.932	.538	.114	.159	.553	.508	.492	.513
		50	**.121**	.629	.818	.462	.197	.144	.705	.235	.583	.583	.511
		100	.182	.674	.932	.583	**.091**	.205	.386	.265	.659	.598	.508
bank_marketing	12	20	.242	.318	.402	.939	.735	**.159**	.250	.432	.462	.477	.525
		50	.227	.629	.720	.962	.371	**.205**	.341	.235	.508	.583	.496
		100	**.167**	.795	.795	.962	.220	.189	.341	.265	.386	.644	.492
bankruptcy	12	20	.258	**.091**	**.091**	.932	.841	.265	.432	.492	.417	.462	.537
		50	.288	**.144**	**.144**	.947	.750	.341	.629	.447	.500	.621	.498
		100	.364	**.182**	**.182**	.977	**.182**	.447	.659	.447	.659	.644	.509
blood_transf	12	20	.220	.644	.780	.689	.614	**.189**	.220	.477	.462	.341	.506
		50	.273	.871	.833	.712	.273	**.205**	.250	.341	.394	.258	.528
		100	.167	.841	.765	.788	.182	.265	.280	**.083**	.462	.326	.539
churn	12	20	.303	.750	.962	.841	.629	**.174**	.189	.583	.439	.439	.461
		50	.242	.841	.977	.765	.356	**.114**	.371	.159	.515	.485	.494
		100	**.030**	.841	.886	.871	.129	.189	.402	.205	.614	.568	.484
coil2000	12	20	**.121**	**.121**	.689	.879	.568	.235	.477	.265	.576	.500	.515
		50	**.106**	.500	.811	.947	.220	.220	.220	.205	.644	.614	.509
		100	**.167**	.636	.856	.947	.205	.189	.235	.356	.720	.720	.472
concrete_str	12	20	**.030**	.152	.152	.932	.780	.280	.538	.583	.364	.591	.510
		50	**.197**	.485	.333	.977	.735	.242	.462	.326	.606	.667	.474
		100	.303	.598	.273	.977	**.152**	.326	.477	.432	.720	.765	.483
coupon_rec	12	20	.227	**.098**	**.098**	.932	.765	.295	.477	.644	.538	.462	.515
		50	.409	**.205**	**.205**	.917	.735	.462	.614	.523	.735	.735	.452
		100	.470	**.227**	**.227**	.977	**.227**	.492	.636	.598	.750	.705	.474
credit_card	12	20	**.008**	.295	.273	.932	.614	.098	.265	.598	.583	.311	.541
		50	**.015**	.720	.515	.871	.295	.068	.674	.341	.538	.402	.506
		100	.106	.788	.583	.932	**.053**	.129	.492	.326	.508	.432	.520
credit_g	12	20	**.091**	.311	.538	.886	.841	.159	.402	.447	.629	.280	.501
		50	.121	.614	.735	.917	.477	**.114**	.508	.220	.629	.447	.488
		100	**.061**	.598	.705	.947	.402	.083	.295	.159	.629	.523	.512

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
diabetes	12	20	.523	.598	.841	**.068**	.788	.447	.356	.462	.530	.424	.499
		50	.212	.962	.848	.212	.356	**.053**	.326	.462	.523	.417	.526
		100	**.030**	.886	.962	.356	.326	.068	.356	.220	.644	.447	.518
diabetes130us	12	20	**.159**	.386	.568	.833	.841	.538	.205	.598	.227	.424	.491
		50	.182	.508	.871	**.068**	.545	.280	.583	.326	.561	.636	.509
		100	.303	.750	.932	**.091**	.371	.311	.583	.371	.674	.674	.480
diamonds	12	20	**.008**	.242	.242	.955	.795	.205	.508	.689	.455	.515	.492
		50	**.182**	.402	.561	.561	.591	.205	.553	.523	.583	.765	.481
		100	**.227**	.712	.568	.712	.288	.250	.303	.402	.508	.644	.509
ecomm_shipping	12	20	**.136**	.879	.856	.545	.735	.189	.311	.295	.311	.447	.495
		50	**.045**	.644	.811	.735	.326	.250	.326	.265	.371	.598	.513
		100	**.061**	.811	.947	.735	.371	.356	.280	.265	.417	.659	.474
fiat_500	12	20	**.129**	.311	.439	.939	.720	**.129**	.144	.417	.508	.508	.530
		50	**.121**	.386	.568	.750	.598	.144	.462	.356	.629	.447	.512
		100	.227	.735	.455	.826	**.114**	.250	.417	.356	.583	.402	.528
fitness_club	12	20	.424	.750	.583	**.053**	.220	.235	.598	.629	.652	.561	.517
		50	.227	.871	.667	**.076**	.235	.280	.568	.326	.720	.689	.505
		100	.273	.765	.402	**.106**	.174	.295	.583	.402	.750	.462	.543
food_delivery	12	20	.182	.280	.742	.955	.750	**.129**	.598	.311	.462	.447	.487
		50	.182	.636	.742	.121	.417	**.038**	.553	.371	.432	.538	.550
		100	**.061**	.917	.697	.220	.295	.083	.598	.265	.447	.538	.534
give_me_credit	12	20	.303	.886	.841	.417	.598	**.144**	.235	.386	.265	.583	.511
		50	.091	.674	.917	.674	.205	.205	.523	**.083**	.462	.417	.526
		100	.106	.773	.917	.773	**.053**	.129	.341	.311	.386	.492	.525
good_customer	12	20	.242	.644	.962	.212	.417	**.205**	.689	.432	.447	.477	.501
		50	**.061**	.765	.962	.568	.235	.098	.598	.174	.356	.720	.502
		100	**.091**	.689	.962	.682	.189	.114	.356	.220	.447	.735	.508
hazelnut	12	20	**.167**	.250	.250	.955	.871	.235	.371	.629	.553	.598	.484
		50	**.182**	.348	.265	.477	.841	.205	.538	.341	.780	.886	.486
		100	**.182**	.682	.553	.591	.220	.205	.386	.295	.841	.795	.495
health_ins	12	20	**.045**	.530	.674	.962	.705	.144	.235	.341	.136	.288	.537
		50	.152	.902	.886	.326	.265	**.083**	.295	.341	.356	.371	.552
		100	**.136**	.955	.864	.568	.197	.174	.447	.220	.492	.417	.513
heloc	12	20	.341	.371	.530	.917	.386	**.068**	.538	.568	.614	.553	.497
		50	.227	.780	.826	.917	.167	**.129**	.386	.235	.508	.417	.510
		100	.091	.902	.659	.962	**.045**	.114	.417	.280	.553	.417	.512
houses	12	20	**.045**	.121	.121	.932	.780	.265	.568	.447	.538	.598	.510
		50	**.182**	.386	.295	.780	.735	.205	.614	.371	.720	.780	.470
		100	.242	.553	.402	.553	**.121**	.265	.720	.326	.826	.674	.505
hr_analytics	12	20	.205	.492	.705	.962	.841	**.053**	.295	.326	.371	.386	.505
		50	**.015**	.939	.833	.871	.583	.038	.386	.098	.386	.462	.492
		100	**.045**	.886	.750	.932	.205	.068	.462	.129	.477	.265	.525
jm1	12	20	**.129**	.197	.326	.924	.720	.220	.326	.538	.462	.447	.526
		50	**.197**	.273	.614	.856	.303	.205	.644	.311	.523	.689	.506
		100	**.106**	.508	.606	.856	.152	.129	.644	.189	.538	.720	.512
marketing_cmp	12	20	**.167**	.614	.727	.939	.689	.235	.477	.402	.447	.295	.475
		50	**.030**	.432	.932	.917	.492	.083	.356	.235	.477	.462	.509
		100	.152	.758	.962	.917	.220	**.098**	.356	.189	.553	.659	.484
maternal_health	12	20	.235	**.121**	.159	.955	.735	.189	.538	.265	.583	.508	.534
		50	**.242**	.371	.348	.636	.742	.280	.765	.326	.629	.841	.466
		100	.333	.326	.477	.780	**.318**	.356	.705	.447	.614	.841	.472
miami_housing	12	20	.106	.235	.341	.939	.538	**.053**	.492	.644	.341	.705	.516
		50	**.106**	.636	.644	.258	.129	.129	.447	.265	.659	.553	.560
		100	.212	.871	.697	.455	**.106**	.235	.280	.341	.674	.553	.522
mic	12	20	**.045**	.280	.583	.977	.750	.235	.705	.523	.508	.417	.463
		50	**.061**	.689	.864	.962	.265	.144	.235	.205	.462	.538	.510
		100	.212	.795	.864	.962	**.106**	.235	.371	.326	.477	.508	.489
naticusdroid	12	20	.121	**.091**	.212	.962	.780	.250	.386	.553	.439	.530	.523
		50	**.106**	.492	.455	.780	.705	.129	.568	.265	.568	.614	.494
		100	.152	.508	.591	.841	**.076**	.174	.311	.250	.629	.689	.533
online_shoppers	12	20	**.167**	.386	.402	.932	.705	.189	.568	.417	.341	.189	.529
		50	**.106**	.758	.568	.977	.333	.280	.326	.280	.417	.417	.511
		100	**.045**	.697	.636	.977	.098	.174	.462	.447	.508	.492	.501
protein	12	20	**.091**	.159	.197	.932	.871	.295	.432	.386	.538	.614	.506
		50	.227	.311	.652	**.114**	.871	.250	.598	.432	.598	.826	.488
		100	.379	**.189**	.341	**.189**	**.189**	.402	.614	.492	.598	.811	.552
qsar_biodeg	12	20	.288	.492	.659	.917	.523	**.068**	.492	.629	.341	.371	.501
		50	.136	.629	.811	.818	.311	**.068**	.341	.311	.538	.538	.510
		100	**.045**	.492	.780	.924	.258	.068	.402	.295	.598	.614	.505
qsar_fish	12	20	.333	.508	.386	.864	.750	.280	.280	**.250**	.356	.538	.522
		50	.318	.386	.917	.917	**.121**	.280	.492	.432	.417	.583	.497
		100	.303	.689	.894	.902	.242	**.144**	.523	.295	.447	.689	.475

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
sdss17	12	20	**.152**	.326	.545	.955	.780	.280	.295	.417	.432	.311	.512
		50	**.045**	.742	.773	.720	.394	.114	.583	.341	.629	.462	.480
		100	**.136**	.909	.909	.765	.288	.159	.432	.280	.432	.553	.483
seismic_bumps	12	20	**.167**	.652	.947	.523	.568	.295	.674	.568	.568	.386	.448
		50	**.152**	.417	.977	.220	.333	.174	.705	.220	.735	.659	.505
		100	**.152**	.500	.977	.326	.242	.174	.674	.311	.432	.720	.511
splice	12	20	**.220**	.280	.235	.955	.614	**.220**	.265	.523	.553	.417	.534
		50	**.076**	.644	.455	.735	.341	.098	.371	.235	.371	.432	.563
		100	**.136**	.750	.598	.841	.152	.159	.462	.235	.447	.432	.533
students_drop	12	20	.129	.182	.265	.932	.750	**.114**	.447	.356	.417	.614	.533
		50	**.121**	.674	.674	.409	.470	.144	.508	.205	.553	.508	.527
		100	.182	.712	.765	.515	**.091**	.205	.583	.280	.583	.614	.512
supercond	12	20	**.023**	.212	.280	.955	.811	.174	.508	.477	.545	.424	.509
		50	.167	.659	.682	**.083**	.447	.189	.614	.341	.765	.568	.512
		100	.242	.636	.583	**.121**	**.121**	.265	.371	.356	.765	.477	.562
taiwanese_bnk	12	20	.348	.538	.477	.705	.932	**.250**	.538	.356	.432	.417	.489
		50	.152	.697	.765	.492	.644	**.114**	.492	.295	.265	.614	.509
		100	.227	.818	.826	.674	.205	**.189**	.462	**.189**	.356	.629	.512
website_phish	12	20	**.038**	.470	.553	.932	.470	.220	.326	.250	.523	.523	.518
		50	**.076**	.803	.652	.803	.280	.144	.295	.326	.386	.356	.535
		100	.121	.879	.879	.803	**.114**	.144	.311	.295	.462	.417	.515
wine_quality	12	20	**.091**	**.091**	**.091**	.955	.856	.326	.538	.492	.447	.508	.515
		50	.455	**.227**	**.227**	.871	.689	.492	.689	.568	.750	.750	.441
		100	.500	**.250**	**.250**	.902	**.250**	.523	.674	.629	.765	.750	.462
blood_transf	374	20	**.242**	.500	.500	.864	.591	.333	.424	–	.258	.288	.500
		50	**.091**	.500	.485	.864	.621	.212	**.091**	–	.227	.136	.525
		100	**.030**	.409	.515	.864	.621	.152	.152	–	.288	.106	.535
diabetes	384	20	**.152**	.530	.576	.864	.561	**.152**	.182	–	.288	.167	.515
		50	.061	.591	.576	.864	.561	**.030**	.121	–	.106	.136	.540
		100	.152	.409	.606	.864	.561	.121	**.091**	–	.167	.227	.545
anneal	450	20	.152	.530	.212	.864	.258	**.061**	.394	–	.333	.227	.548
		50	**.212**	.515	.258	.864	.227	.303	.500	–	.318	.288	.548
		100	.227	.561	.242	.864	**.091**	.576	.606	–	.409	.318	.543
qsar_fish	453	20	.152	.530	.515	.864	.561	**.121**	.242	–	.136	.258	.525
		50	.121	.530	.515	.864	.530	**.030**	.212	–	.136	.136	.551
		100	**.000**	.530	.515	.848	.530	**.000**	.182	–	.136	.136	.553
credit_g	500	20	**.030**	.409	.394	.864	.591	**.030**	.333	–	.167	.227	.540
		50	.030	.470	.485	.864	.591	**.000**	.121	–	.106	.197	.545
		100	.030	.500	.515	.864	.591	**.000**	.121	–	.136	.106	.551
maternal_health	507	20	.182	.045	**.000**	.864	.470	.273	.303	–	.439	.288	.566
		50	.242	**.076**	.091	.864	.530	.182	.364	–	.348	.197	.581
		100	.061	.136	.091	.864	.591	**.000**	.061	–	.409	.167	.556
concrete_str	515	20	**.030**	.045	–	.864	.561	.242	–	–	.348	.258	.571
		50	**.000**	.045	–	.864	.561	.212	–	–	.288	.227	.586
		100	**.000**	.045	–	.864	.591	.061	–	–	.258	.167	.596
qsar_biodeg	528	20	.273	.500	–	.864	.409	.091	–	–	**.076**	.318	.556
		50	.182	.530	–	.864	.439	**.000**	–	–	.167	.167	.556
		100	.061	.530	–	.864	.530	**.000**	–	–	.045	.167	.561
health_ins	669	20	**.091**	.288	–	.894	.530	.212	–	–	.136	.197	.576
		50	.091	.470	–	.879	.591	**.000**	–	–	.136	.106	.553
		100	.061	.500	–	.909	.591	**.000**	–	–	.076	.167	.543
website_phish	677	20	.061	.318	–	.864	.591	**.030**	–	–	.076	.227	.571
		50	.091	.439	–	.864	.591	**.000**	–	–	.167	.167	.545
		100	.030	.439	–	.864	.591	**.000**	–	–	.136	.076	.566
fitness_club	750	20	**.121**	.591	–	.864	.288	**.121**	–	–	.167	.197	.566
		50	.212	**.076**	–	.864	.348	.182	–	–	.167	.394	.609
		100	.182	.258	–	.864	.379	.182	–	–	**.167**	.379	.576
airfoil_noise	752	20	.061	**.045**	–	.848	.591	.091	–	–	.227	.318	.578
		50	.061	**.045**	–	.879	.591	.121	–	–	.288	.348	.558
		100	.061	**.045**	–	.879	.591	.091	–	–	.258	.348	.563
fiat_500	769	20	**.000**	.045	–	.909	.470	.061	–	–	.258	.227	.598
		50	**.000**	.076	–	.909	.530	**.000**	–	–	.227	.106	.609
		100	**.000**	.227	–	.909	.530	**.000**	–	–	.136	.106	.598
mic	850	20	.030	.136	–	.864	.500	**.000**	–	–	.227	.288	.581
		50	**.000**	.288	–	.864	.561	**.000**	–	–	.197	.318	.545
		100	.030	.379	–	.864	.591	**.000**	–	–	.106	.076	.581
good_customer	861	20	.273	.561	–	.864	.318	**.182**	–	–	.409	.197	.525
		50	.152	.439	–	.864	.439	**.121**	–	–	.227	.167	.561
		100	.091	.530	–	.864	.470	**.000**	–	–	.348	.167	.520
marketing_cmp	1 120	20	.152	.470	–	.864	.591	.121	–	–	.318	**.106**	.525
		50	**.000**	.500	–	.864	.591	**.000**	–	–	.136	.288	.520
		100	**.000**	.500	–	.864	.591	**.000**	–	–	.227	.076	.540

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
hazelnut	1 200	20	.061	**.045**	–	.864	.591	.121	–	–	.348	.197	.576
		50	**.030**	.045	–	.864	.591	.091	–	–	.470	.227	.551
		100	**.000**	.045	–	.864	.591	.030	–	–	.197	.258	.591
seismic_bumps	1 292	20	.182	.591	–	.848	.591	.152	–	–	.379	**.106**	.497
		50	.242	.409	–	.864	.682	**.182**	–	–	.227	.288	.505
		100	**.121**	.409	–	.864	.682	**.121**	–	–	.197	.258	.515
splice	1 596	20	.303	**.076**	–	.864	.591	.333	–	–	.258	.167	.591
		50	**.061**	.227	–	.864	.591	.121	–	–	.197	.167	.576
		100	**.030**	.348	–	.864	.591	.061	–	–	.106	.167	.571
bioresponse	1 876	20	.091	**.045**	–	.864	.591	.333	–	–	.258	.288	.576
		50	**.000**	.045	–	.864	.591	.121	–	–	.348	.409	.540
		100	**.000**	.045	–	.864	.591	.030	–	–	.409	.348	.540
hiva_agnostic	1 923	20	**.121**	.652	–	.652	.833	.212	–	–	.258	.318	.465
		50	.182	.652	–	.758	.833	**.121**	–	–	.288	.227	.457
		100	.152	.500	–	.773	.803	**.000**	–	–	.288	.167	.495
students_drop	2 212	20	**.030**	.076	–	.864	.455	.273	–	–	.288	.318	.583
		50	**.061**	.106	–	.864	.591	.182	–	–	.318	.288	.556
		100	.061	.106	–	.864	.591	**.000**	–	–	.348	.318	.545
churn	2 500	20	.273	.530	–	.864	.439	**.091**	–	–	.258	.106	.551
		50	.273	.561	–	.864	.470	**.000**	–	–	.167	.076	.561
		100	.273	.561	–	.864	.530	**.000**	–	–	.197	.106	.540
qsar_tid11	2 871	20	.061	**.045**	–	.803	.591	.303	–	–	.348	.136	.596
		50	**.030**	.045	–	.864	.591	.212	–	–	.227	.348	.571
		100	**.030**	.045	–	.864	.591	.121	–	–	.167	.227	.601
bankruptcy	2 955	20	**.000**	.045	–	.864	.591	.030	–	–	.318	.227	.576
		50	**.000**	.197	–	.864	.591	**.000**	–	–	.288	.258	.551
		100	.030	.439	–	.864	.591	**.000**	–	–	.288	.045	.545
wine_quality	3 249	20	.061	**.045**	–	.864	.591	.121	–	–	.379	.303	.553
		50	**.030**	.045	–	.864	.591	.091	–	–	.288	.409	.551
		100	**.030**	.045	–	.864	.591	.091	–	–	.227	.318	.576
taiwanese_bnk	3 410	20	.424	.470	–	.864	.591	.212	–	–	.318	**.167**	.515
		50	.303	.500	–	.864	.591	**.000**	–	–	.318	.197	.505
		100	.303	.500	–	.864	.591	**.000**	–	–	.348	.242	.492
naticusdroid	3 746	20	**.000**	.167	–	.864	.530	.182	–	–	.288	.136	.586
		50	**.000**	.167	–	.864	.561	**.000**	–	–	.167	.379	.561
		100	.030	.197	–	.864	.591	**.000**	–	–	.076	.439	.556
coil2000	4 911	20	**.030**	.045	–	.833	.591	**.030**	–	–	.379	.318	.556
		50	.091	.167	–	.864	.591	**.030**	–	–	.288	.258	.556
		100	.061	.288	–	.864	.591	**.000**	–	–	.136	.258	.561
bank_churn	5 000	20	.333	.409	–	.848	.591	**.030**	–	–	.227	.288	.523
		50	.333	.470	–	.864	.591	**.000**	–	–	.258	.167	.525
		100	**.000**	.500	–	.864	.591	**.000**	–	–	.136	.106	.551
heloc	5 230	20	.152	.348	–	.864	.591	.152	–	–	**.106**	.258	.556
		50	.273	.470	–	.864	.591	**.091**	–	–	.227	.167	.530
		100	.152	.500	–	.864	.591	**.000**	–	–	.167	.197	.530
jm1	5 443	20	**.030**	.045	–	.864	.591	.273	–	–	.318	.288	.566
		50	**.030**	.045	–	.864	.591	.364	–	–	.227	.318	.576
		100	**.030**	.045	–	.864	.591	**.030**	–	–	.227	.288	.581
ecomm_shipping	5 499	20	.273	.561	–	.864	.530	**.000**	–	–	.197	.136	.535
		50	.303	.561	–	.864	.561	**.000**	–	–	.167	.106	.540
		100	.273	.561	–	.864	.561	**.000**	–	–	.106	.136	.545
online_shoppers	6 165	20	.182	**.076**	–	.864	.439	.182	–	–	.379	.288	.576
		50	.212	.167	–	.864	.561	**.030**	–	–	.288	.288	.556
		100	.242	.227	–	.864	.591	**.000**	–	–	.288	.348	.530
coupon_rec	6 342	20	.091	**.045**	–	.864	.470	.152	–	–	.258	.348	.586
		50	.091	**.045**	–	.864	.561	.121	–	–	.318	.258	.576
		100	.091	**.045**	–	.864	.561	.091	–	–	.258	.167	.601
miami_housing	6 888	20	.212	.227	–	.788	.591	.212	–	–	.258	**.197**	.573
		50	.091	.258	–	.848	.591	**.000**	–	–	.197	.227	.563
		100	.061	.409	–	.879	.591	**.000**	–	–	.106	.106	.568
hr_analytics	9 579	20	.152	.348	–	.864	.591	.485	–	–	.348	**.106**	.540
		50	.182	.409	–	.864	.591	**.000**	–	–	.136	.258	.540
		100	.061	.470	–	.864	.591	**.000**	–	–	.136	.288	.525
houses	10 320	20	**.030**	.045	–	.848	.591	.121	–	–	.258	.258	.583
		50	**.000**	.045	–	.879	.591	.091	–	–	.227	.227	.588
		100	**.000**	.045	–	.909	.591	.091	–	–	.167	.348	.573
supercond	10 632	20	.061	**.045**	–	.864	.591	.212	–	–	.197	.227	.596
		50	.061	**.045**	–	.864	.591	.091	–	–	.318	.258	.571
		100	**.030**	.045	–	.879	.591	.091	–	–	.318	.288	.563
credit_card	15 000	20	.091	**.045**	–	.864	.591	.212	–	–	.409	.136	.576
		50	.242	.106	–	.864	.591	**.000**	–	–	.409	.167	.561
		100	.364	.379	–	.864	.591	**.000**	–	–	.197	.076	.566

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
amazon_access	16 385	20	.061	**.045**	–	.848	.591	.394	–	–	.288	.288	.573
		50	.061	**.045**	–	.864	.591	.091	–	–	.167	.197	.606
		100	**.030**	.045	–	.864	.591	**.030**	–	–	.197	.227	.596
bank_marketing	22 605	20	.182	.227	–	.864	.591	**.030**	–	–	.258	.136	.571
		50	.212	.379	–	.864	.591	**.061**	–	–	.227	.288	.525
		100	.303	.470	–	.864	.591	**.000**	–	–	.258	.348	.495
food_delivery	22 725	20	.091	**.045**	–	.864	.591	.364	–	–	.439	.227	.556
		50	.212	**.136**	–	.864	.591	.182	–	–	.379	.227	.551
		100	.212	**.136**	–	.864	.591	.242	–	–	.318	.197	.566
protein	22 865	20	**.030**	.045	–	.833	.561	.212	–	–	.258	.227	.596
		50	**.030**	.045	–	.864	.591	.091	–	–	.227	.227	.591
		100	**.030**	.045	–	.864	.591	.091	–	–	.227	.197	.596
kddcup09	25 000	20	.576	.561	–	.833	.439	.303	–	–	.318	**.076**	.545
		50	.515	.561	–	.848	.530	.121	–	–	.167	**.106**	.548
		100	.485	.561	–	.864	.530	**.000**	–	–	.167	.136	.540
diamonds	26 970	20	.212	**.045**	–	.818	.591	.212	–	–	.318	.303	.571
		50	.212	**.076**	–	.818	.591	.152	–	–	.288	.288	.573
		100	.242	**.106**	–	.848	.591	.121	–	–	.318	.288	.558
diabetes130us	35 760	20	**.030**	.045	–	.848	.591	–	–	–	.379	.167	.578
		50	.061	**.045**	–	.864	.591	–	–	–	.379	.288	.556
		100	**.000**	.076	–	.864	.591	–	–	–	.348	.318	.551
aps_failure	38 001	20	.121	.530	–	.833	.439	–	–	–	.197	**.045**	.576
		50	.242	.530	–	.848	.530	–	–	–	.288	**.136**	.528
		100	**.182**	.530	–	.864	.561	–	–	–	.227	.227	.515
sdss17	39 027	20	.182	**.136**	–	.864	.409	–	–	–	.409	.227	.576
		50	.242	**.167**	–	.864	.409	–	–	–	.409	.258	.566
		100	.394	**.288**	–	.864	.409	–	–	–	.348	**.288**	.551
airline_sat	64 941	20	**.030**	.045	–	.864	.591	–	–	–	.227	.227	.591
		50	**.000**	.045	–	.864	.591	–	–	–	.136	.258	.601
		100	**.000**	.045	–	.864	.591	–	–	–	.197	.288	.586
give_me_credit	75 000	20	.091	**.076**	–	.864	.591	–	–	–	.348	.227	.566
		50	**.091**	.136	–	.864	.591	–	–	–	.318	.348	.540
		100	.273	.348	–	.864	.591	–	–	–	.348	**.167**	.530

Table 11: Ensemble pruning per utility at 20, 50 and 100 queries: rank percentile (lower is better) as in Table 2, averaged over ten seeds; *others* as there. Medals: each row's three lowest. Mean rows average the utilities of a block and of the table; the table's mean row is the column of Table 2. Parentheses, unmedalled: a method ran on fewer than 80% of the utilities. '–': no run.

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
mean, all utilities		20	.271	.699	.708	.544	.681	**.253**	.427	.556	.315	.369	.496
		50	**.137**	.778	.803	.253	.582	.145	.471	.327	.303	.395	.534
		100	**.178**	.810	.843	.246	.239	.220	.457	.318	.384	.484	.538
LightGBM oracle (51 utilities)													
airfoil_noise	16	20	.227	.864	.712	.750	.826	**.159**	.326	.568	.205	.280	.485
		50	.106	.750	.902	**.053**	.750	.129	.477	.326	.295	.265	.543
		100	.197	.902	.841	**.098**	.341	.220	.386	.341	.356	.341	.552
airline_sat	16	20	.197	.773	.894	.826	.462	**.174**	.492	.720	**.174**	.250	.480
		50	**.076**	.902	.871	.265	.356	.083	.447	.386	.098	.311	.560
		100	**.136**	.955	.879	.250	.189	.159	.371	.326	.235	.462	.552
amazon_access	16	20	.227	.545	.886	.780	.902	**.205**	.386	.659	.280	.265	.469
		50	.167	.591	.962	.182	.886	**.159**	.508	.280	.189	.402	.526
		100	.136	.742	.962	.432	.402	**.129**	.326	.205	.326	.508	.536
anneal	16	20	**.371**	**.371**	**.371**	**.371**	**.371**	.856	.841	.886	**.371**	**.371**	.476
		50	**.371**	**.371**	**.371**	**.371**	**.371**	.788	.856	.902	**.371**	**.371**	.479
		100	**.371**	**.371**	**.371**	**.371**	**.371**	.765	.811	.902	**.371**	**.371**	.484
aps_failure	16	20	.242	.886	.326	.727	.750	**.189**	.447	.553	.508	.356	.481
		50	.182	.962	.644	.523	.644	**.114**	.295	.265	.394	.379	.522
		100	**.106**	.962	.515	.689	.189	.129	.220	.235	.326	.538	.554
bank_churn	16	20	**.030**	.795	.811	.886	.841	.144	.386	.326	.341	.402	.467
		50	**.045**	.841	.947	.167	.523	.068	.553	.280	.189	.402	.541
		100	**.121**	.917	.977	.356	.197	.144	.371	.220	.280	.492	.542
bank_marketing	16	20	.167	.371	.917	.568	.689	**.159**	.583	.629	.447	.356	.483
		50	.091	.848	.977	**.045**	.523	.114	.583	.326	.250	.417	.532
		100	.273	.750	.977	**.136**	**.136**	.295	.508	.447	.432	.508	.524

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
bankruptcy	16	20	.197	.856	.598	.856	.644	**.114**	.220	.447	.295	.447	.502
		50	.152	.902	.902	.705	.439	**.053**	.386	.189	.250	.432	.519
		100	**.076**	.932	.947	.795	.212	.144	.189	.114	.447	.553	.513
bioresponse	16	20	.258	.795	.644	.508	.689	**.114**	.235	.538	.341	.402	.518
		50	**.091**	.811	.636	.326	.462	.114	.523	.295	.205	.326	.562
		100	**.152**	.932	.583	.629	.167	.189	.371	.280	.356	.417	.544
blood_transf	16	20	.152	.508	.735	**.030**	.765	.402	.477	.447	.508	.492	.510
		50	.136	.568	.818	**.030**	.720	.189	.553	.402	.568	.432	.517
		100	.167	.674	.932	**.068**	.523	.250	.371	.371	.492	.462	.527
churn	16	20	.273	.886	.886	.364	.765	**.144**	.250	.644	.227	.318	.501
		50	.121	.871	.886	**.068**	.470	.144	.402	.311	.295	.356	.554
		100	.197	.902	.962	**.114**	.205	.326	.538	.371	.417	.508	.512
coil2000	16	20	.167	.689	.902	**.023**	.220	.583	.386	.508	.402	.508	.522
		50	**.106**	.765	.962	.167	.311	.144	.492	.341	.432	.508	.529
		100	**.121**	.780	.962	.333	.129	.144	.402	.205	.477	.644	.533
concrete_str	16	20	.318	.841	.644	.386	.947	.402	**.220**	.629	**.220**	.311	.492
		50	.152	.962	.674	**.076**	.826	.174	.523	.280	.250	.386	.527
		100	.242	.977	.826	**.121**	.402	.265	.432	.371	.341	.432	.526
coupon_rec	16	20	.167	.659	.705	.932	.402	**.068**	.189	.265	.265	.538	.537
		50	**.076**	.720	.902	.826	.265	.098	.523	.189	.159	.402	.532
		100	.273	.917	.962	**.136**	**.136**	.295	.492	.386	.371	.523	.522
credit_card	16	20	.167	.606	.833	.917	.598	**.144**	.386	.492	.250	.432	.488
		50	**.091**	.902	.932	.356	.538	.114	.417	.341	.280	.417	.516
		100	.152	.871	.977	.432	**.076**	.174	.447	.265	.356	.462	.534
credit_g	16	20	.197	.917	.758	.432	.462	**.144**	.447	.280	.417	.326	.524
		50	.061	.841	.629	**.030**	.409	.083	.477	.205	.402	.432	.576
		100	.182	.856	.689	**.091**	.295	.280	.598	.341	.523	.326	.538
diabetes	16	20	.318	.477	.379	**.098**	.689	.477	.689	.705	.553	.311	.509
		50	.273	.629	.409	**.235**	.538	**.235**	.583	.553	.432	.432	.535
		100	.197	.538	.523	.462	.371	**.114**	.614	.250	.394	.591	.550
diabetes130us	16	20	.485	.856	**.098**	**.098**	.902	.341	.644	.614	.432	.568	.496
		50	**.045**	.424	.197	.197	.674	.159	.674	.326	.417	.598	.564
		100	.136	.674	.371	.371	**.068**	.159	.538	.220	.447	.659	.576
diamonds	16	20	.288	.811	.773	.583	.780	.341	.341	.659	**.189**	.250	.483
		50	**.061**	.947	.871	.189	.674	.083	.371	.295	.371	.265	.533
		100	.182	.856	.962	**.091**	.182	.205	.402	.295	.477	.447	.545
ecomm_shipping	16	20	.288	.477	.902	.545	.644	**.250**	.492	.598	.311	.326	.497
		50	.197	.735	.932	**.061**	.538	.144	.462	.189	.250	.462	.556
		100	.242	.765	.962	**.121**	**.121**	.265	.508	.371	.371	.614	.531
fiat_500	16	20	.348	.750	.644	**.038**	.689	.417	.326	.689	.432	.508	.501
		50	.212	.902	.902	**.068**	.583	.174	.432	.432	.417	.644	.496
		100	.167	.977	.598	**.083**	.303	.205	.447	.341	.477	.735	.526
fitness_club	16	20	.500	.682	.455	**.083**	.811	.492	.326	.629	.477	.598	.496
		50	.242	.439	.795	**.076**	.674	.174	.568	.462	.402	.326	.545
		100	.273	.364	.765	**.136**	.341	.295	.462	.371	.538	.477	.558
food_delivery	16	20	.258	.697	.697	.955	.826	**.098**	.220	.447	.311	.371	.491
		50	**.045**	.667	.720	.674	.568	.068	.356	.311	.402	.311	.533
		100	.182	.841	.811	**.091**	.288	.205	.462	.326	.447	.508	.540
give_me_credit	16	20	.333	.720	.917	**.038**	.447	.402	.523	.674	.371	.311	.508
		50	.288	.576	.962	**.091**	.462	.250	.644	.492	.311	.508	.516
		100	.333	.568	.962	**.167**	**.167**	.356	.508	.477	.508	.568	.517
good_customer	16	20	.364	.515	.977	**.053**	.492	.386	.553	.720	.523	.674	.470
		50	.227	.432	.962	**.076**	.492	.386	.462	.417	.371	.568	.526
		100	.227	.492	.977	**.091**	.402	.250	.447	.326	.508	.492	.540
hazelnut	16	20	.106	.811	.841	**.061**	.629	.174	.583	.523	.341	.311	.517
		50	**.015**	.962	.932	.106	.682	.038	.447	.265	.235	.462	.529
		100	.167	.939	.962	.326	**.083**	.189	.583	.356	.386	.432	.519
health_ins	16	20	.212	.727	.864	.795	.886	**.083**	.447	.311	.144	.280	.497
		50	.242	.962	.879	**.061**	.674	.174	.386	.205	.356	.402	.531
		100	.227	.765	.947	**.114**	.235	.250	.386	.386	.402	.538	.537
heloc	16	20	.318	.697	.697	.932	.477	**.114**	.417	.477	.197	.242	.519
		50	.197	.788	.886	.189	.341	**.068**	.462	.447	.114	.265	.572
		100	**.030**	.947	.811	.318	.106	.144	.508	.205	.205	.235	.579
hiva_agnostic	16	20	**.136**	.598	.947	.591	.583	.220	.629	.553	.326	.386	.474
		50	.152	.508	.947	**.076**	.545	.174	.432	.220	.341	.477	.560
		100	.288	.402	.947	**.144**	**.144**	.311	.492	.462	.371	.568	.551
houses	16	20	.212	.576	.682	.955	.689	.220	.265	.659	**.174**	.280	.500
		50	**.030**	.917	.871	.644	.583	.038	.417	.159	.265	.311	.523
		100	.136	.947	.962	**.068**	.303	.159	.538	.205	.432	.492	.530
hr_analytics	16	20	.318	.902	.871	.720	.356	**.159**	.189	.598	.295	.220	.515
		50	**.061**	.886	.902	.167	.447	.083	.477	.311	.220	.371	.549
		100	.136	.902	.977	.348	**.068**	.159	.492	.250	.205	.568	.541

continued on the next page

continued from the previous page

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
jm1	16	20	.273	.614	.705	.932	.386	**.174**	.568	.462	.303	.424	.495
		50	.167	.568	.871	**.045**	.258	.129	.508	.250	.235	.508	.587
		100	.167	.765	.917	**.083**	.144	.250	.538	.303	.303	.583	.547
kddcup09	16	20	**.258**	.530	.447	.644	.826	.311	.659	.644	.280	.356	.485
		50	.121	.720	.856	**.061**	.780	.144	.598	.280	.303	.409	.527
		100	.212	.765	.902	**.106**	.341	.235	.447	.356	.447	.538	.528
marketing_cmp	16	20	.258	.871	.583	.811	.826	**.189**	.386	.614	.288	.288	.473
		50	.167	.932	.470	**.053**	.545	.189	.462	.326	.318	.379	.564
		100	.227	.856	.674	**.106**	.348	.265	.538	.356	.371	.477	.539
maternal_health	16	20	.167	.826	.917	.477	.962	**.129**	.250	.447	.311	.432	.481
		50	.091	.902	.788	.409	.947	**.053**	.553	.129	.250	.220	.519
		100	**.045**	.947	.765	.523	.341	.068	.508	.159	.250	.295	.550
miami_housing	16	20	.348	.659	.485	.288	.811	**.189**	.492	.674	.280	.614	.501
		50	.152	.856	.795	**.023**	.765	.068	.189	.295	.447	.492	.544
		100	.227	.902	.871	**.114**	.189	.250	.371	.341	.386	.614	.536
mic	16	20	.409	.742	.856	**.083**	.735	.371	.674	.689	.508	.553	.445
		50	.167	.553	.947	**.061**	.750	.189	.614	.644	.280	.508	.497
		100	.182	.538	.962	**.091**	.258	.265	.432	.432	.417	.629	.537
naticusdroid	16	20	.318	.818	.939	**.197**	.917	.220	.553	.598	.273	.258	.479
		50	.091	.841	.932	**.038**	.659	.129	.371	.371	.220	.402	.541
		100	.197	.932	.947	**.083**	.235	.265	.371	.341	.417	.447	.536
online_shoppers	16	20	.212	.795	.765	.894	.538	.341	.174	.508	**.159**	.174	.512
		50	**.045**	.871	.917	.076	.523	.129	.386	.371	.220	.341	.551
		100	.136	.932	.886	.235	**.129**	.205	.523	.250	.432	.462	.534
protein	16	20	**.076**	.523	.765	.932	.644	.098	.538	.371	.174	.462	.499
		50	**.061**	.894	.886	.674	.629	.083	.356	.250	.205	.356	.513
		100	.152	.848	.955	**.076**	.174	.174	.477	.311	.326	.477	.552
qsar_biodeg	16	20	.379	.523	.682	.265	.705	.235	.492	.583	**.144**	.326	.542
		50	.136	.697	.667	.212	.735	**.098**	.614	.250	.189	.280	.558
		100	**.167**	.705	.727	.417	.417	.205	.432	.295	.295	.311	.554
qsar_fish	16	20	.364	.727	.674	**.076**	.735	.417	.689	.598	.674	.508	.454
		50	.212	.932	.530	**.083**	.644	.189	.553	.417	.583	.583	.499
		100	.273	.826	.841	**.136**	.258	.295	.568	.356	.568	.553	.508
qsar_tid11	16	20	.182	.811	.553	.932	.508	.174	.311	.386	.159	**.114**	.543
		50	**.045**	.947	.659	.795	.417	.068	.341	.114	.311	.235	.547
		100	.182	.947	.765	**.091**	**.091**	.205	.341	.295	.432	.386	.573
sdss17	16	20	.273	.848	.727	.629	.720	**.144**	.341	.508	.295	.250	.503
		50	.348	.856	.826	**.045**	.636	.311	.614	.553	.280	.402	.498
		100	.167	.917	.947	**.076**	.242	.280	.568	.341	.326	.447	.527
seismic_bumps	16	20	.561	.583	.447	**.053**	.765	.735	.508	.598	.477	.402	.495
		50	.288	.795	.962	**.076**	.697	.220	.462	.386	.538	.508	.489
		100	.227	.811	.977	**.083**	.326	.220	.432	.311	.477	.568	.523
splice	16	20	.394	.856	.917	.288	.644	.250	.492	.598	**.129**	.356	.498
		50	**.076**	.818	.818	.242	.447	.083	.492	.402	.265	.508	.533
		100	**.061**	.871	.826	.432	.114	.083	.462	.250	.280	.523	.551
students_drop	16	20	.197	.553	.629	.955	.886	**.053**	.447	.492	.220	.280	.499
		50	.091	.795	.705	.811	.841	**.083**	.402	.220	.311	.311	.502
		100	**.106**	.917	.811	.538	.326	.129	.265	.220	.447	.280	.544
supercond	16	20	.394	.682	.773	.932	.750	**.098**	.129	.614	.144	.432	.496
		50	.136	.947	.917	.500	.629	**.053**	.265	.250	.250	.174	.540
		100	.212	.947	.917	**.106**	.348	.235	.432	.356	.348	.394	.532
taiwanese_bnk	16	20	.364	.780	.689	.879	.598	**.098**	.523	.492	.250	.280	.493
		50	.091	.750	.871	.606	.402	**.038**	.417	.129	.235	.250	.562
		100	**.091**	.750	.932	.720	.121	.114	.538	.235	.250	.326	.540
website_phish	16	20	.455	.667	.553	.667	.659	.417	.402	.538	**.303**	.333	.497
		50	.152	.841	.750	.227	.583	.265	.265	.508	**.144**	.265	.550
		100	**.106**	.750	.735	.462	.212	.159	.386	.280	.159	.462	.569
wine_quality	16	20	.106	.583	.636	.932	.886	**.068**	.250	.477	.386	.265	.501
		50	**.030**	.788	.629	.765	.811	.053	.371	.144	.326	.371	.519
		100	.136	.970	.727	**.068**	.409	.159	.477	.205	.379	.424	.552

Table 12: Circuit discovery per utility at 20, 50 and 100 queries: rank percentile (lower is better) as in Table 2, averaged over ten seeds; *others* as there. Medals: each row's three lowest. Mean rows average the utilities of a block and of the table; the table's mean row is the column of Table 2. Parentheses, unmedalled: a method ran on fewer than 80% of the utilities. '–': no run.

utility	d	b	S-LFBO	Data Shapley	NASH	greedy forward	ACDC	PEP	GP-BO	SMAC	BOCS	LFBO	BORE	*others*
mean, all utilities		20	**.222**	.398	.400	.773	.396	.727	.269	.260	.315	.349	.315	.551
		50	.109	.503	.518	.822	.488	.740	**.076**	.232	.313	.146	.257	.560
		100	.113	.575	.579	.838	.375	.761	**.074**	.201	.293	.232	.280	.549
$\lambda = 0.3$ **(7 utilities)**														
case63	53	20	.159	.516	.516	.722	.516	.738	**.071**	.373	.127	.452	.516	.507
		50	**.095**	.532	.635	.825	.635	.675	.167	.310	.143	.119	.437	.514
		100	.238	.698	.651	.857	**.048**	.738	.183	.341	.413	.357	.532	.508
case7	53	20	.206	.421	.421	.746	.421	.738	.230	**.087**	.159	.357	.294	.565
		50	**.048**	.516	.516	.802	.516	.659	.135	.167	.206	.103	.278	.567
		100	.127	.571	.508	.825	**.032**	.659	.167	.230	.381	.262	.373	.573
case103	117	20	.286	.484	.484	.817	.484	.595	.214	**.151**	.190	.333	.286	.554
		50	.175	.595	.595	.849	.548	.643	**.024**	.087	.413	.119	.246	.566
		100	.079	.635	.635	.873	.563	.659	**.024**	.151	.254	.183	.214	.547
case111	117	20	**.111**	.135	.135	.802	.135	.802	.246	.357	.540	.452	.437	.583
		50	**.016**	.286	.294	.802	.254	.881	.040	.452	.444	.230	.278	.582
		100	.048	.325	.468	.802	.405	.881	**.024**	.325	.254	.230	.341	.558
case124	117	20	.270	.214	.230	.659	.198	.944	.881	**.151**	.746	.246	.310	.556
		50	.190	.357	.357	.762	.310	.976	**.040**	.341	.540	.119	.214	.585
		100	.111	.476	.476	.794	.405	.976	**.024**	.167	.349	.183	.135	.572
case45	117	20	.270	.468	.468	.849	.468	.690	.214	.405	.206	.373	**.119**	.537
		50	.111	.595	.587	.865	.556	.722	.103	**.087**	.175	.183	.119	.557
		100	.143	.651	.651	.865	.579	.722	**.071**	.087	.190	.230	.183	.538
case93	117	20	.254	.548	.548	.817	.548	.579	**.024**	.294	.238	.230	.246	.556
		50	.127	.643	.643	.849	.595	.627	**.024**	.183	.270	.151	.230	.546
		100	.048	.667	.667	.849	.595	.690	**.024**	.103	.206	.183	.183	.545