EvoFG: Decoupling Operator Evolution and Feature Binding for Efficient Feature Generation

Anonymous authors
Paper under double-blind review

Abstract

Combining large language models (LLMs) with evaluation feedback has emerged as an effective approach to feature generation for structured data. However, directly generating concrete features typically requires jointly searching over operator structures and input-feature bindings. Consequently, potentially useful structures may receive negative feedback due to mismatched bindings, obscuring the direction of subsequent evolution and limiting LLM utilization efficiency. Some methods also include dataset names and column semantics in their prompts, introducing potential data leakage risks. To address these challenges, we propose EvoFG, a feature generation paradigm that decouples operator structure evolution from feature binding optimization. EvoFG searches over reusable, parameterized operator families, allowing the LLM to generate code without specifying concrete column combinations. Instead, a program enumerates candidate bindings and identifies high-quality feature instances through coarse-to-fine evaluation. The best binding, peak gain, and supporting evidence from other high-performing bindings jointly provide feedback to guide subsequent structural mutations, mitigating the misjudgment of structures caused by binding mismatches and turning each proposal into an exploration of multiple concrete features. This design delegates concrete column selection to programmatic evaluation and does not rely on dataset-specific information, thereby minimizing potential data leakage risks associated with LLM use. The high-quality features generated by EvoFG significantly improve the performance of different downstream models. Evaluations across multiple datasets demonstrate that EvoFG outperforms existing AutoFE methods. With results averaged equally across datasets under the same LLM budget, EvoFG produces over 30 times as many positive-gain candidate features and approximately 2.5 times as many ultimately selected new features as direct LLM-based feature generation, while reducing token consumption per selected feature by approximately 50%. These results demonstrate that organizing search around operator families and optimizing bindings programmatically enables more effective LLM utilization, greater search efficiency, and higher token efficiency, providing an effective search paradigm for feedback-driven evolutionary feature generation.

1 Introduction

Structured data are central to scientific, medical, business, and engineering applications. Explicit nonlinear transformations and variable interactions can improve the representations available to a predictor, even when the predictor itself is expressive (Domingos, 2012). Classical automated feature engineering constructs and filters candidates from predefined operations (Kanter & Veeramachaneni, 2015; Katz et al., 2016; Horn et al., 2019; Zhang et al., 2023). Large language models (LLMs) extend this approach to a more open space of executable transformations, with evaluation feedback guiding subsequent proposals (Hollmann et al., 2023b; Abhyankar et al., 2025).

A concrete feature combines two decisions: its operator structure and its input columns. For example, multiplication $o(a, b) = ab$ with binding $(1, 2)$ produces x_1x_2. Poor performance may reflect either an unsuitable operator or an unsuitable pair of inputs; the same structure bound to $(3, 7)$ may be useful. If one binding is also used to judge structural potential, negative feedback can divert search away

[1]Code: https://anonymous.4open.science/r/EvoFG-D066/README.md

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.

from promising transformations. Direct program evolution can change both structures and bindings. Our question is how search and feedback should be organized so that limited LLM proposals are less often consumed by mismatched column choices.

Some generators additionally expose dataset descriptions, column names, or samples in their prompts (Hollmann et al., 2023b; Han et al., 2024; Nam et al., 2024). Such context can be useful, but may identify a known task and elicit memorized solutions. This is a potential source of contamination, not evidence that a particular baseline has leaked data. We therefore seek an interface that does not require dataset identity or original column semantics.

We propose EvoFG, which evolves *operator families*. The LLM proposes a reusable structure with unbound input slots. A local program enumerates feasible bindings, screens them cheaply, and evaluates a shortlist through training-only cross-validation. The best binding summarizes observed peak quality; other strong bindings describe usable alternatives. These statistics and the parent program guide structural mutation. Selected concrete features are frozen before outer-test evaluation. Figure 1 summarizes the separation between proposal, binding, and prediction.

Our contributions are threefold: (1) an operator-family search unit that separates open-ended structural proposals from discrete input selection; (2) coarse-to-fine binding evaluation and peak–support feedback that reuse one proposal across many instances; and (3) complementary evaluations of downstream prediction, matched-request discovery efficiency, binding mismatch, and semantic-context dependence. Prediction and search efficiency are assessed separately: more positive training candidates do not guarantee better generalization for every predictor.

2 Related Work

Automated feature engineering. Deep Feature Synthesis and ExploreKit established the value of programmatic construction and selection (Kanter & Veeramachaneni, 2015; Katz et al., 2016). AutoFeat composes predefined nonlinear transformations and selects a compact subset (Horn et al., 2019); OpenFE combines operator application with efficient incremental evaluation and pruning (Zhang et al., 2023). GRFG decomposes feature-group and operation selection (Wang et al., 2022), while SCOPE-FE controls feature pairs and operator utility before expansion (Park et al., 2026). Thus, enumeration, decomposed decisions, and screening are established ideas. EvoFG combines them with evolving code structures and feedback from multiple bindings.

LLM feature generation and program evolution. CAAFE, FeatLLM, and OCTree generate transformations using semantic context, rules, or decision-tree reasoning (Hollmann et al., 2023b; Han et al., 2024; Nam et al., 2024). LLM-FE and ELLM-FT evolve transformations using performance feedback (Abhyankar et al., 2025; Gong et al., 2025); EFE and TOPOFE study standardized feature programs and multi-island evolution (Taga et al., 2026; Li & Ramakrishnan, 2026). These methods already include feedback, memory, and sometimes multiple features per proposal. Our distinction is explicit binding search for a fixed unbound structure and feedback organized around that family. Anonymous prompting alone is not a novelty claim: OCTree also studies context-agnostic inputs.

Structure search with internal optimization. FunSearch combines LLM proposals with executable evaluation (Romera-Paredes et al., 2024). LLM-SR separates equation structure from continuous coefficient fitting (Shojaee et al., 2025). EvoFG instead optimizes discrete column bindings, where combinatorial coverage and noisy evaluation affect structural judgments. Its archive relates to MAP-Elites and quality-diversity search (Mouret & Clune, 2015; Pugh et al., 2016). Appendix A expands this comparison.

3 EvoFG: Operator-Family Evolution with Programmatic Binding

3.1 Feature Instances, Operator Families, and the Search Objective

For a training matrix $X \in \mathbb{R}^{n \times d}$, an operator o is a row-wise numerical program with $k(o)$ input slots. A binding $c = (i_1, \ldots, i_{k(o)}) \in \mathcal{C}(o)$ specifies its columns:

$$z_{o,c}(x) = o(x_{i_1}, \ldots, x_{i_{k(o)}}), \qquad \mathcal{F}(o) = \{z_{o,c} : c \in \mathcal{C}(o)\}. \tag{1}$$

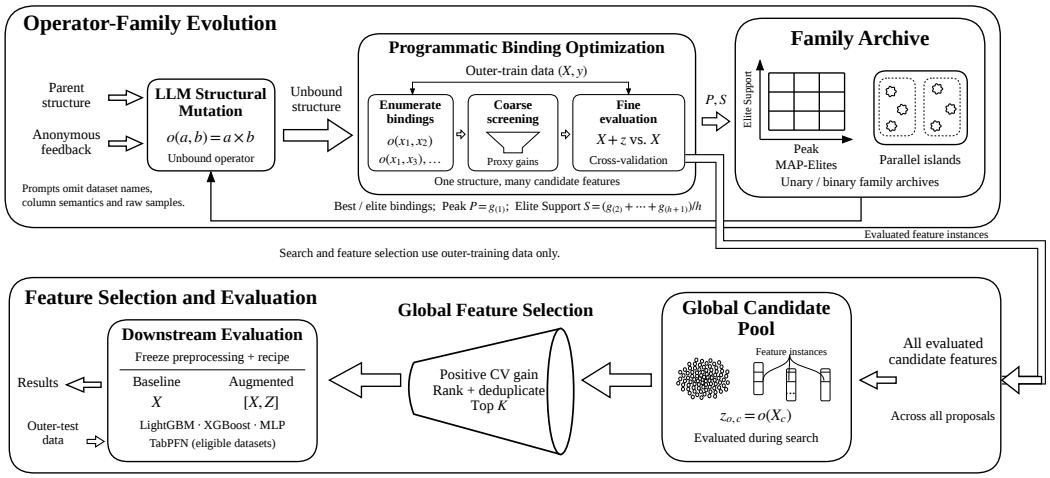


Figure 1: EvoFG separates LLM structural mutation from local binding optimization. Family feedback closes the evolutionary loop. Concrete features are selected and frozen before evaluation on outer-test data.

The implemented space contains unary and binary programs. Binary bindings are ordered pairs of distinct columns, giving $d(d-1)$ possibilities; output deduplication handles redundant instances, including symmetric operations. "Parameterized" refers to bindable input slots; constants are part of the proposed program, with no separate continuous optimizer.

Let $u(o,c)$ be signed utility relative to the original-feature baseline, with positive values indicating improvement, and let $V(o)=\max_{c\in\mathcal{C}(o)} u(o,c)$. For $V^\star=\sup_o V(o)$,

$$\sup_{o,c} u(o,c) = \sup_o V(o), \tag{2}$$

$$V^\star - u(o,c) = \underbrace{V^\star - V(o)}_{\text{structure loss}} + \underbrace{V(o) - u(o,c)}_{\text{binding loss}}. \tag{3}$$

A poor binding can obscure a useful structure. Decoupling changes how a finite budget is spent and what feedback is available; it does not increase the ideal optimum of the same feasible space. Appendix B gives definitions and proofs.

3.2 Coarse-to-Fine Optimization of Feature Bindings

The evaluator checks arity, legal indices, row-wise execution, and finite outputs. Cross-row statistics, row identifiers, and file access are prohibited. Candidates are materialized from outer-training data. For large families, a cheap proxy uses three training subsets: the first fits a shared baseline, the second estimates a binned residual or gradient correction, and the third measures proxy loss improvement. This stage uses log loss for classification and squared error for regression.

Families with at most 64 bindings are fully evaluated. Larger families receive 64 fine-evaluation slots, or 128 if the family exceeds 4,096 bindings. The shortlist combines proxy ranking, protected parent bindings, and approximately 20% random exploration. Invalid candidates, exact duplicates, and cache hits can reduce new evaluations. Fine evaluation uses fixed LightGBM with three-fold training-only cross-validation, comparing $[X,z]$ with X for every candidate. It does not measure conditional gain given previously selected features.

For the completed fine-evaluation set $\widetilde{\mathcal{C}}(o)$,

$$\widehat{V}(o) = \max_{c\in\widetilde{\mathcal{C}}(o)} \widehat{u}(o,c). \tag{4}$$

This is a partial, noisy estimate, not an exhaustive optimum. Appendix B.3 separates screening and evaluation error, and Appendix F.3 measures screening losses.

Algorithm 1 Operator-family evolution

Require: Outer-training data, initial structures $\mathcal{O}_0$, proposal budget Q, feature cap K
1: Evaluate $\mathcal{O}_0$; initialize family archives and candidate pool $\mathcal{Z}$
2: **for** $t = 1, \ldots, Q$ **do**
3: Select island, arity, and parent; construct anonymous family feedback
4: Ask the LLM for one unbound operator; validate the program
5: Enumerate bindings, screen candidates, and fine-evaluate the shortlist
6: Update family descriptors, archives, and evaluated candidates $\mathcal{Z}$
7: **end for**
8: Deduplicate positive-gain outputs; retain at most K by signed fine gain
9: Freeze transformations and preprocessing; evaluate downstream predictors

3.3 Peak Gain and Elite Support for Structural Evolution

For an increasing classification score q and baseline q_0, normalized gain is $(q - q_0)/\max(10^{-6}, 1 - q_0)$; for RMSE it is $(q_0 - q)/\max(10^{-6}, |q_0|)$. Scores are first aggregated over the same folds. Family descriptors clip this gain to $[0, 1]$, while positive-candidate decisions retain signed gains. For output-deduplicated gains $g_{(1)} \geq g_{(2)} \geq \cdots$, define

$$P(o) = g_{(1)}, \qquad S(o) = \frac{g_{(2)} + g_{(3)}}{2}, \tag{5}$$

padding absent alternatives with zeros. Peak P captures the best observed instance; elite support S describes strong alternatives, with $0 \leq S \leq P \leq 1$. It is neither independent evidence count nor generalization probability. One useful binding can suffice to retain a structure.

Two islands maintain separate unary and binary archives. Cells are indexed by peak and support bins; competition within a cell retains higher-peak structures. The method does not maximize a weighted sum of the descriptors. Parent code, best and alternative bindings, quality summaries, and evaluation coverage condition mutations. This organization relates to quality-diversity search (Mouret & Clune, 2015; Pugh et al., 2016). The equations define the intended family summary; Section 4.5 evaluates the executed component controls and their predictive effects.

3.4 Evolutionary Search and Final Feature Selection

Initial structures and proposals share the evaluation interface. Each child is rebound rather than forced to use its parent's columns. At budget exhaustion, let $\mathcal{Z}_Q$ be the unique evaluated outputs. The final representation is

$$\mathcal{Z}_K = \mathrm{TopK}_{z \in \mathcal{Z}_Q : \widehat{u}(z) > 0} \, \widehat{u}(z), \qquad \Phi_{\mathcal{Z}_K}(X) = [X, Z_K]. \tag{6}$$

Individual gains allocate a fixed capacity; joint utility need not be additive. We use $K = 20$ without selecting dataset-specific caps from test performance. Initialization, archive bins, and evaluator settings appear in Appendix C.2.

3.5 Semantic-Context-Free LLM Interaction

The LLM sees arity, anonymous column identifiers, task type, metric, parent code, and training feedback. It does not require dataset names, target names, original column names, category text, or raw samples. Local binding evaluation selects concrete inputs. Statistics remain data-dependent, so this interface reduces semantic exposure rather than providing formal privacy or proving absence of pretraining contamination. Outer-test isolation addresses evaluation bias. Appendix G gives a prompt example and interface checks.

4 Experiments

4.1 Experimental Setup

The predictive benchmark contains 27 datasets: 18 classification tasks evaluated by accuracy and nine regression tasks evaluated by RMSE. Five outer experimental units establish training and

test boundaries, respecting temporal constraints where applicable. Search and selection use only outer-training data. Three internal model seeds are averaged within each outer unit; these are not independent feature-search restarts.

EvoFG, Direct, and Direct-Batch use DeepSeek V4.1 Flash and at most 50 proposals. Direct evolves already-bound features; Direct-Batch permits up to ten candidates per request. Comparators include AutoFeat, OpenFE, OCTree, CAAFE, and FeatLLM. Frozen features are evaluated with LightGBM, XGBoost, MLP, and TabPFN-2.5 (Chen & Guestrin, 2016; Ke et al., 2017; Gorishniy et al., 2021; Hollmann et al., 2023a; 2025; Grinsztajn et al., 2025). Appendix C specifies datasets and settings.

4.2 Predictive Performance with Conventional Downstream Models

Table 1: Predictive performance averaged equally over LightGBM, XGBoost, and MLP. Results average available outer experiments; model means use identical outer units. Bold and underlining identify the best and second-best unrounded means. —: unavailable comparable result; ×: evaluation unavailable.

Classification: accuracy ↑

Dataset	n	p	Base	AutoFeat	OpenFE	OCTree	CAAFE	FeatLLM	Direct	Direct-Batch	EvoFG
rl	4970	12	0.746	—	×	0.735	0.732	0.742	0.745	0.732	**0.789**
compass	16644	17	0.745	—	×	0.721	0.715	0.721	0.748	0.755	**0.778**
balance-scale	625	4	0.898	0.897	0.900	0.896	0.930	0.894	0.910	0.915	**0.954**
jungle_chess	44819	6	0.876	0.879	0.883	0.877	0.856	0.855	0.875	0.873	**0.904**
vehicle	846	18	0.786	0.790	0.790	0.740	0.748	0.738	0.788	0.783	**0.805**
MagicTelescope	13376	10	0.856	×	×	0.847	0.845	0.848	0.858	0.857	**0.863**
kdd_ipums_la	5188	20	0.865	×	×	0.865	0.861	0.863	0.866	0.862	**0.870**
Higgs	940160	24	0.741	0.742	0.742	0.724	0.725	0.725	0.742	0.742	**0.743**
Academic	4424	36	0.753	0.753	0.753	0.743	0.744	0.743	0.754	0.753	**0.755**
car	1728	6	0.994	0.992	0.994	0.969	0.970	0.971	0.995	0.993	**0.996**
airlines	539383	7	0.660	0.661	0.661	0.648	0.648	0.648	0.660	0.660	**0.661**
kddCup09	5032	45	0.767	×	×	0.762	0.762	0.763	0.767	0.764	**0.768**
adult	32561	14	0.857	×	×	0.855	0.854	0.855	0.857	0.854	**0.858**
pc1	1109	21	0.929	×	×	0.921	0.921	0.921	**0.930**	0.930	0.929
covtype	581012	54	**0.873**	—	—	—	—	—	0.872	0.844	0.863
road-safety	111762	32	**0.784**	—	—	—	—	—	0.760	0.760	0.768
pol	10082	26	0.986	×	×	0.975	0.975	0.976	0.986	**0.986**	0.985
pharyngitis	676	18	0.638	0.640	0.633	0.639	0.636	**0.640**	0.636	0.637	0.638

Regression: RMSE ↓

Dataset	n	p	Base	AutoFeat	OpenFE	Direct	Direct-Batch	EvoFG
diamonds	53940	9	559.668	559.040	553.189	555.920	554.980	**539.323**
airfoil_self_noise	1503	5	1.640	1.599	1.683	1.647	1.689	**1.565**
Enefit	2017824	11	442.214	—	—	420.324	434.037	**415.796**
forest-fires	517	12	**79.614**	84.552	83.925	81.227	81.365	79.963
insurance	1338	6	**5114.023**	5180.940	5174.549	5168.662	5273.773	5141.858
housing	20640	9	48479.332	48614.878	48795.709	47559.159	**46090.816**	47166.856
Tesla_Stock	3471	14	**31.828**	—	—	374.456	67.628	37.387
bike	17379	12	95.335	**32.863**	102.457	95.799	99.160	94.684
communities	1994	122	**0.148**	**0.148**	0.149	0.150	0.148	0.149

Table 1 averages three conventional predictors equally; Appendix D.1 gives separate means and standard deviations. The common outer-unit intersection is used for all three model means, making aggregation exactly reproducible before rounding. Missing evaluations are not replaced by zeros or baseline scores. Comparisons are descriptive at dataset level (Demšar, 2006); the tables do not establish universal superiority or statistical significance.

The averaged comparison identifies operator-family search as the more reliable way to improve conventional predictors. EvoFG most often records the best classification accuracy. Where results

are comparable, it improves on raw features more regularly than fixed operator libraries and more regularly than LLM generators that emit an already-bound transformation. Direct and Direct-Batch share the proposal budget, but each of their candidates is a single column assignment, so the feedback mixes operator quality with one input choice. EvoFG holds the structure fixed, scores its legal bindings, and returns the peak together with alternative bindings, so later mutations are guided by the family. Semantic generators that use context, rules, or tree feedback less often occupy the leading rank. On regression, RMSE reductions are smaller and more learner-dependent, yet EvoFG remains among the strongest methods. The next experiment keeps these features frozen and changes only the predictor.

4.3 Feature Augmentation for TabPFN

Table 2: TabPFN-2.5 performance on the 22 eligible datasets, using the same frozen features. All entries are measured TabPFN results, averaged over available outer experiments. The dataset order follows Table 1. Notation follows Table 1.

Classification: accuracy ↑

Dataset	n	p	Base	AutoFeat	OpenFE	OCTree	CAAFE	FeatLLM	Direct	Direct-Batch	EvoFG
rl	4970	12	0.861	—	×	0.850	0.849	0.849	0.862	0.856	**0.876**
compass	16644	17	0.842	—	×	0.832	0.817	0.837	0.836	0.831	**0.870**
balance-scale	625	4	0.978	0.993	0.978	0.970	0.988	0.974	**0.998**	0.996	0.997
jungle_chess	44819	6	0.877	0.877	0.880	**0.909**	0.871	0.865	0.877	0.884	0.907
vehicle	846	18	**0.878**	0.874	0.876	0.866	0.870	0.868	0.878	0.876	0.865
MagicTelescope	13376	10	0.881	×	×	0.871	0.871	0.871	0.883	0.882	**0.883**
kdd_ipums_la	5188	20	0.887	×	×	0.876	0.876	0.876	0.887	0.886	**0.887**
Academic	4424	36	**0.794**	0.793	0.793	0.783	0.782	0.783	0.792	0.793	0.790
car	1728	6	0.996	0.995	0.997	0.985	0.988	0.987	0.996	0.999	**1.000**
kddCup09	5032	45	0.806	×	×	—	—	—	**0.808**	0.807	0.804
adult	32561	14	0.867	×	×	0.856	0.857	0.857	0.867	0.867	**0.868**
pc1	1109	21	0.935	×	×	0.926	0.924	0.928	0.935	0.936	**0.940**
pol	10082	26	0.989	×	×	—	—	—	**0.990**	0.989	0.989
pharyngitis	676	18	0.681	**0.694**	0.689	—	—	—	0.681	0.682	0.680

Regression: RMSE ↓

Dataset	n	p	Base	AutoFeat	OpenFE	Direct	Direct-Batch	EvoFG
diamonds	53940	9	507.877	508.001	507.798	507.843	504.095	**503.807**
airfoil_self_noise	1503	5	1.043	1.054	1.072	1.037	1.032	**0.982**
forest-fires	517	12	53.809	**53.728**	53.929	53.795	53.773	53.833
insurance	1338	6	4484.977	4477.361	4472.311	4471.705	**4459.987**	4537.638
housing	20640	9	42153.210	42231.484	42109.365	41280.844	**40398.475**	40802.716
Tesla_Stock	3471	14	41.241	—	—	41.521	39.923	**37.957**
bike	17379	12	78.718	**21.725**	80.076	78.556	80.863	81.913
communities	1994	122	0.130	0.130	**0.130**	0.130	0.130	0.130

TabPFN-2.5 is evaluated on 22 eligible datasets (14 classification and eight regression tasks). Higgs, airlines, covtype, road-safety, and Enefit exceed the configured training-sample limit. No new feature search is performed. OCTree, CAAFE, and FeatLLM reuse generated features with fresh TabPFN evaluation on 11 classification tasks each. Appendix D.2 reports exactly the same means with standard deviations.

The same frozen features transfer to TabPFN-2.5 without a new search. On classification, EvoFG again sits at or beside the best result more often than direct concrete generation and more often than the semantic baselines. The selected bindings therefore complement a pretrained tabular prior, including under an evaluator that differs from the tree and MLP models used during search. Family organization supplies that transfer: each proposal is retained according to its legal inputs, and the downstream model receives concrete columns chosen by evaluation. When the original representation

is already strong, additional features move the score only slightly. Across the eligible tasks, evolving unbound operator families remains the more consistent augmentation.

4.4 Feature Yield and Token Efficiency under Matched LLM Budgets

An independent 15-dataset pilot compares EvoFG and Direct with shared initialization, three search repeats per method, and 50 request opportunities per repeat. Its 80/20 split and internal evaluation differ from the main five-unit protocol. Counts deduplicate exact training outputs and exclude initialization; "positive" means positive training-internal fine gain. Let $N^{+}_{d,m}$, $K_{d,m}$, and $T_{d,m}$ denote positive outputs, newly selected features, and recorded tokens, pooled within dataset d. With E and D denoting the methods,

$$R_{\text{pos}} = \frac{1}{|\mathcal{D}|}\sum_{d}\frac{N^{+}_{d,E}}{N^{+}_{d,D}}, \quad R_{\text{sel}} = \frac{1}{|\mathcal{D}|}\sum_{d}\frac{K_{d,E}}{K_{d,D}}, \tag{7}$$

$$\eta_{\text{tok}} = 1 - \frac{1}{|\mathcal{D}|}\sum_{d}\frac{T_{d,E}/K_{d,E}}{T_{d,D}/K_{d,D}}. \tag{8}$$

Table 3: Feature yield under matched request budgets. The last column weights datasets equally; token quantities refer to recorded usage.

Quantity	EvoFG	Direct	Dataset-equal comparison
Positive new candidates	25,003	616	40.47×
Newly selected features	874	371	2.60×
Request opportunities	2,250	2,250	1.00×
Recorded tokens	3,040,038	2,572,990	48.73% less per selected

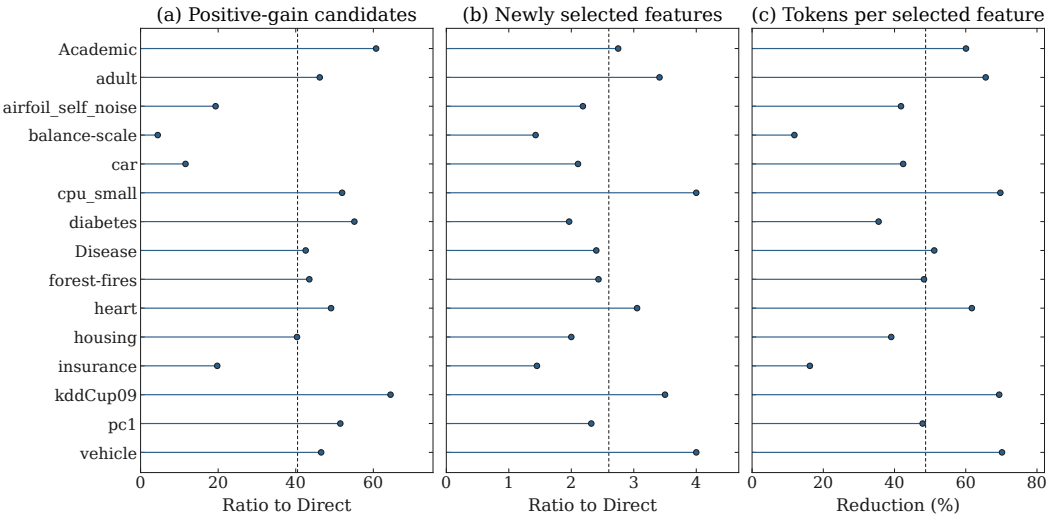


Figure 2: Dataset-level feature yield and recorded token efficiency. Dashed lines indicate dataset-equal means. Positive outputs are training-evaluation candidates, not independently verified generalization gains.

Table 3 supports the abstract's efficiency ratios. The 40.47× candidate ratio becomes 2.60× at selection because of the cap, ranking, and duplicates. EvoFG uses more recorded tokens overall but fewer per newly selected feature, reflecting reuse of each structure across bindings. Token records are incomplete for 14 searches; the saving describes observed usage and does not estimate missing consumption. Appendix H gives counts and accounting boundaries.

4.5 Ablation Studies

We consolidate component interventions, feature-cap sensitivity, and controlled mechanism checks here. These experiments distinguish predictive utility from the discovery advantages measured in Section 4.4.

4.5.1 Component contributions to predictive performance

The completed study compares six search variants using results averaged over five outer experiments. Frozen feature sets are evaluated with the same LightGBM, XGBoost, and MLP configurations. A search-quality filter excludes diamonds from the primary aggregate because independent proposals received only 23/50 successful replies. The seven retained tasks include four classification and three regression datasets; Appendix E.1 reports standard deviations across model-seed averages, every dataset, and the all-eight sensitivity analysis.

Table 4: Component ablations averaged over five outer experiments. Positive values favor full EvoFG over the indicated control. Dataset-equal means use four classification and three regression tasks after the search-quality filter; all signs are retained.

	Accuracy gain (pp)			RMSE reduction (%)		
Control	LGBM	XGB	MLP	LGBM	XGB	MLP
No explicit peak	**+0.39**	**+0.99**	-1.64	-1.67	**+0.25**	-1.12
Peak only	-0.01	**+0.93**	**+0.13**	-4.26	-1.79	-0.54
Single binding	-1.78	-1.20	-1.37	-0.20	-3.31	**+2.27**
Fixed library	-0.84	**+0.23**	-1.87	-2.10	-3.69	**+0.11**
Proxy only	**+1.22**	**+1.72**	-0.31	**+0.14**	-4.28	**+4.26**
Independent proposals	-1.23	-1.36	-0.92	-0.94	-0.26	**+3.12**

Grounded evaluation is the clearest component benefit. Compared with proxy-only evaluation, full EvoFG improves mean classification accuracy by 1.22 and 1.72 percentage points for LightGBM and XGBoost, and reduces MLP regression RMSE by 4.26%. It wins on five of seven datasets for both XGBoost and MLP. These results support spending local computation on fine evaluation to turn broad candidate generation into useful features. The benefit is task-dependent: XGBoost regression and MLP classification do not improve.

Family feedback provides complementary, conditional gains. Full feedback improves XGBoost classification by 0.99 points over removing the explicit peak and by 0.93 points over peak-only feedback. However, single binding, a fixed library, and independent proposals are competitive or stronger in several settings (Table 4). The executed controls also change some archive, initialization, or prompt details. Thus, the evidence supports useful component effects in particular settings, not universal gains from every component. Multiple bindings primarily improve discovery coverage; the final feature set remains learner-dependent.

4.5.2 Feature capacity and binding diagnostics

With search and ranking fixed, increasing K from 1 to 20 improves LightGBM on all eight sensitivity tasks; $K = 20$ wins on seven of eight against $K = 5$, and five of eight against $K = 10$. XGBoost and MLP respond less consistently, illustrating the role of redundancy and joint learning (Küken et al., 2024). Appendix E.2 gives the full capacity table; no test-based cap selection is used.

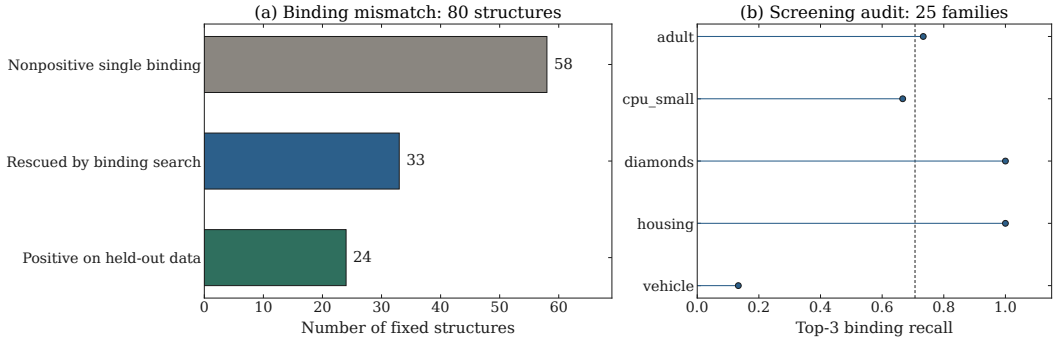


Figure 3: Controlled binding diagnostics. Left: nested counts among 80 fixed structures. Right: top-three recall in 25 exhaustively audited families, averaged by dataset; the dashed line is the mean.

Fixing operator structures isolates input mismatch: 58 of 80 structures have nonpositive single-binding gain; multi-binding search rescues 33, of which 24 remain positive on independent holdout. Matched-quota random search rescues the same cases, supporting coverage more directly than proxy ordering. Exhaustive audits show task-dependent screening recall (Figure 3); both methods recover the correct input set in all 12 synthetic conditions. These findings explain the opportunity created by family search without implying uniformly better final prediction.

4.5.3 Reliability and Semantic-Context Controls

Label permutation still yields positive training candidates, while independent rechecks retain positive gain for 122 of 160 selected features. Search signals therefore require held-out validation (Cawley & Talbot, 2010). Additional initializations preserve high candidate yield despite low feature-identity overlap (Appendix F). Removing semantic context from CAAFE also leaves usable generated features, but the matched balance-scale comparison has lower anonymous accuracy for all three predictors (Appendix G.3). This control separates reduced contextual requirements from predictive superiority.

4.6 Semantic Independence and Computational Costs

Renaming dataset and column semantics leaves anonymous prompts byte-identical on eight tasks; complete fixed-family evaluations satisfy column-permutation invariance. These checks concern the interface and fixed families, not identical stochastic trajectories or formal privacy (Appendix G).

Binding search moves work from the LLM to local evaluation. Across the 52-dataset timing study under shared scheduling, median search time is 28.32 minutes for EvoFG, 1.75 for Direct, and 6.37 for Direct-Batch. The timing study covers more datasets than the predictive table. Resource contention and model loading prevent interpretation as controlled exclusive-resource speedups. EvoFG improves yield per LLM opportunity while investing more conventional computation (Appendix H.3).

5 Discussion and Conclusion

EvoFG separates open-ended operator evolution from discrete feature binding. Binding search evaluates more instances per proposal, while family summaries give evolution more specific feedback. The matched-request pilot supports higher candidate and selected-feature yield and lower recorded token cost per selected feature; completed ablations identify fine-evaluation benefits for tree-model classification and MLP regression.

Discovery efficiency does not guarantee uniform predictive gains. Screening omissions, adaptive selection bias, redundancy, and predictor-specific inductive biases constrain the final representation. Family feedback is a search heuristic, not a generalization certificate. Our conclusions apply to the studied unary/binary space and LLM configuration. This paradigm makes structural proposals reusable, while independent evaluation determines their practical predictive utility.

AI USE STATEMENT

Generative AI assistance was used for retrieval and discovery, including identifying related work and locating relevant methodological descriptions. Retrieved claims and bibliographic details were checked against the cited source publications. AI tools also assisted with translating the existing Chinese manuscript into English and preparing the LaTeX text, tables, and English figure labels. This manuscript-preparation assistance is distinct from the LLM-based feature-generation algorithms evaluated in the paper, whose role and experimental settings are described in Sections 3 and 4.1. The authors are responsible for the final text, citations, claims, and experimental artifacts, including material prepared with AI assistance.

REPRODUCIBILITY STATEMENT

The anonymous code link appears on the first page. Appendix B gives formal definitions, Appendix C specifies datasets, aggregation and model parameters, and Appendix D reports model-specific results. Appendices E–H describe the sensitivity, mechanism, semantic-context and efficiency analyses. The main and model-specific means follow the same aggregation rules.

REFERENCES

Nikhil Abhyankar, Parshin Shojaee, and Chandan K. Reddy. LLM-FE: Automated Feature Engineering for Tabular Data with LLMs as Evolutionary Optimizers, 2025. URL https://arxiv.org/abs/2503.14434. arXiv:2503.14434, 2025.

Jacob Austin et al. Program Synthesis with Large Language Models, 2021. URL https://arxiv.org/abs/2108.07732. arXiv:2108.07732, 2021.

James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization, 2012. URL https://www.jmlr.org/papers/v13/bergstra12a.html. JMLR 13, 281–305, 2012.

Gavin C. Cawley and Nicola L. C. Talbot. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation, 2010. URL https://www.jmlr.org/papers/v11/cawley10a.html. JMLR 11, 2079–2107, 2010.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System, 2016. URL https://arxiv.org/abs/1603.02754. KDD, 785–794, 2016.

Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets, 2006. URL https://www.jmlr.org/papers/v7/demsar06a.html. JMLR 7, 1–30, 2006.

Pedro Domingos. A Few Useful Things to Know about Machine Learning, 2012. Communications of the ACM 55(10), 78–87, 2012.

Nanxu Gong, Chandan K. Reddy, Wangyang Ying, Haifeng Chen, and Yanjie Fu. Evolutionary Large Language Model for Automated Feature Transformation, 2025. URL https://arxiv.org/abs/2405.16203. AAAI, 2025.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting Deep Learning Models for Tabular Data, 2021. URL https://arxiv.org/abs/2106.11959. NeurIPS, 2021.

Gabriel Grand, Lionel Wong, Maddy Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, and Jacob Andreas. LILO: Learning Interpretable Libraries by Compressing and Documenting Code, 2024. URL https://arxiv.org/abs/2310.19791. ICLR, 2024.

Léo Grinsztajn et al. TabPFN-2.5: Advancing the State of the Art in Tabular Foundation Models, 2025. URL https://arxiv.org/abs/2511.08667. arXiv:2511.08667, 2025; revised 2026.

Sungwon Han, Jinsung Yoon, Sercan O. Arik, and Tomas Pfister. Large Language Models Can Automatically Engineer Features for Few-Shot Tabular Learning, 2024. URL https://arxiv.org/abs/2404.09491. ICML, 2024.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second, 2023a. URL https://arxiv.org/abs/2207.01848. ICLR, 2023.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large Language Models for Automated Data Science: Introducing CAAFE for Context-Aware Automated Feature Engineering, 2023b. URL https://arxiv.org/abs/2305.03403. NeurIPS, 2023.

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, 2025. URL https://doi.org/10.1038/s41586-024-08328-6. Nature, 2025.

Franziska Horn, Robert Pack, and Michael Rieger. The autofeat Python Library for Automated Feature Engineering and Selection, 2019. URL https://arxiv.org/abs/1901.07329. arXiv:1901.07329, 2019; revised 2020.

James Max Kanter and Kalyan Veeramachaneni. Deep Feature Synthesis: Towards Automating Data Science Endeavors, 2015. IEEE DSAA, 2015.

Gilad Katz, Eui Chul Richard Shin, and Dawn Song. ExploreKit: Automatic Feature Generation and Selection, 2016. IEEE ICDM, 979–984, 2016.

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, 2017. NeurIPS, 2017.

Jaris Küken, Lennart Purucker, and Frank Hutter. Large Language Models Engineer Too Many Simple Features for Tabular Data, 2024. URL https://arxiv.org/abs/2410.17787. Table Representation Learning Workshop at NeurIPS, 2024.

Sha Li and Naren Ramakrishnan. Adaptive Graph-of-Islands Evolution for Automatic Feature Engineering with LLMs, 2026. URL https://arxiv.org/abs/2607.23286. arXiv:2607.23286, 2026.

Aman Madaan et al. Self-Refine: Iterative Refinement with Self-Feedback, 2023. URL https://arxiv.org/abs/2303.17651. NeurIPS, 2023.

Jean-Baptiste Mouret and Jeff Clune. Illuminating Search Spaces by Mapping Elites, 2015. URL https://arxiv.org/abs/1504.04909. arXiv:1504.04909, 2015.

Jaehyun Nam, Kyuyoung Kim, Seunghyuk Oh, Jihoon Tack, Jaehyung Kim, and Jinwoo Shin. Optimized Feature Generation for Tabular Data via LLMs with Decision Tree Reasoning, 2024. URL https://arxiv.org/abs/2406.08527. NeurIPS, 2024.

Minhee Park, Seongyeon Son, Yonghyun Lee, and Eunchan Kim. SCOPE-FE: Structured Control of Operator and Pairwise Exploration for Feature Engineering via Quality-Aware Candidate-Space Reduction, 2026. URL https://arxiv.org/abs/2604.27025. arXiv:2604.27025, 2026.

Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality Diversity: A New Frontier for Evolutionary Computation, 2016. URL https://doi.org/10.3389/frobt.2016.00040. Frontiers in Robotics and AI 3:40, 2016.

Bernardino Romera-Paredes et al. Mathematical Discoveries from Program Search with Large Language Models, 2024. URL https://doi.org/10.1038/s41586-023-06924-6. Nature 625, 468–475, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366. NeurIPS, 2023.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K. Reddy. LLM-SR: Scientific Equation Discovery via Programming with Large Language Models, 2025. URL https://arxiv.org/abs/2404.18400. ICLR, 2025.

Ege Onur Taga, Yilin Zhuang, M. Emrullah Ildiz, Petros Mol, Abhimanyu Das, Karthik Duraisamy, and Samet Oymak. Evolutionary Feature Engineering for Structured Data, 2026. URL https://arxiv.org/abs/2607.01548. arXiv:2607.01548, 2026.

Dongjie Wang, Yanjie Fu, Kunpeng Liu, Xiaolin Li, and Yan Solihin. Group-wise Reinforcement Feature Generation for Optimal and Explainable Representation Space Reconstruction, 2022. URL https://arxiv.org/abs/2205.14526. arXiv:2205.14526, 2022.

Tianping Zhang, Zheyu Zhang, Zhiyuan Fan, Haoyan Luo, Fengyuan Liu, Qian Liu, Wei Cao, and Jian Li. OpenFE: Automated Feature Generation with Expert-level Performance, 2023. URL https://arxiv.org/abs/2211.12507. ICML, 2023.

Xinhao Zhang, Jinghan Zhang, Banafsheh Rekabdar, Yuanchun Zhou, Pengfei Wang, and Kunpeng Liu. Dynamic and Adaptive Feature Generation with LLM, 2025. URL https://arxiv.org/abs/2406.03505. IJCAI, 2025.

APPENDIX CONTENTS

Detailed methods, experimental protocols, and supplementary results.

A Extended Comparison with Related Methods 14
A.1 Structure and Input Selection in Feature Generation . . . 14
A.2 Reusable Structures and Internal Optimization . . . 16
A.3 Program Generation and Evaluation Feedback . . . 16
B Formalizing Search and Approximate Evaluation 16
B.1 Feasible Space and Utility . . . 16
B.2 Nested Objective and Loss Decomposition . . . 16
B.3 Screening Omissions and Evaluation Error . . . 17
B.4 Family Feedback and the Final Representation . . . 17
C Datasets and Experimental Protocol 17
C.1 Datasets and Task Metrics . . . 17
C.2 Evaluation Units and Implementation Settings . . . 19
D Predictive Results by Downstream Model 20
D.1 Conventional Predictors: Means and Dispersion . . . 20
D.2 TabPFN: Means and Dispersion . . . 24
E Supplementary Ablation Results 25
E.1 Component Protocol and Complete Results . . . 25
E.2 Detailed Feature-Cap Sensitivity . . . 25
F Supplementary Diagnostic Protocols 27
F.1 Binding Mismatch at Fixed Structures . . . 27
F.2 Synthetic Tasks with Known Relevant Inputs . . . 27
F.3 Exhaustive-Family Reference for Screening . . . 28
F.4 Label-Permutation Negative Control . . . 28
F.5 Sensitivity to Search Initialization . . . 28
F.6 Independent Recheck of Selected Features . . . 29
F.7 Rebinding after Structural Mutation . . . 29
G Dependence on Semantic Information 30
G.1 Anonymous Interaction Interface . . . 30
G.2 Renaming and Column-Permutation Checks . . . 30
G.3 Anonymous Inputs for an External Generator . . . 30
H Discovery Efficiency and Computational Cost 31
H.1 Candidate and Selection Counts . . . 31
H.2 Per-Dataset Yield and Token Records . . . 31
H.3 Search Wall-Clock and Downstream Evaluation Costs . . . 32
H.4 Interpreting Computational Cost . . . 34

A Extended Comparison with Related Methods

A.1 Structure and Input Selection in Feature Generation

The relevant distinction is not merely whether a method uses an LLM. A generator must define a transformation space, select inputs, evaluate candidates, and decide how feedback affects future proposals. The following descriptions expand the comparison using the original publications. Table 5 summarizes the resulting search units.

AutoFeat. AutoFeat constructs nonlinear candidates through multiple rounds of predefined transformations and combinations, then selects a small feature set for an interpretable linear prediction model (Horn et al., 2019). Its expansion can represent relationships that a linear model cannot recover from raw inputs alone. Selection limits the large intermediate space and removes unhelpful features. Thus, AutoFeat already separates broad construction from compact final use. The difference from EvoFG is that transformations come from predefined operations rather than continuing LLM proposals of new program structures.

OpenFE. OpenFE follows expansion and reduction: operators generate candidate features, and FeatureBoost estimates their incremental value on top of baseline predictions (Zhang et al., 2023). Its first pruning stage progressively removes weak candidates while increasing the evaluation sample; a second stage uses feature importance to retain a useful subset. This avoids retraining a complete downstream model for every candidate during initial screening. Operator application to different inputs and coarse-to-fine reduction are therefore direct precedents. EvoFG uses an evolving program space and explicitly returns multiple evaluated bindings to the structural proposal process; it does not claim to introduce enumeration or screening.

GRFG. Group-wise Reinforcement Feature Generation organizes generation through three cascading decisions: a feature group, an operation, and another feature group (Wang et al., 2022). Feature crossing and selection interact with downstream performance rewards. Grouping reduces the granularity of individual decisions and gives the reinforcement learner a more structured search process. This is a precedent for decomposing feature and operation selection. EvoFG differs by asking an LLM to produce new executable structures with a fixed slot interface, then optimizing their discrete bindings locally.

SCOPE-FE. SCOPE-FE controls the candidate space before expansion (Park et al., 2026). FeatureClustering gates input pairs according to mixed-type feature associations, while OperatorProbing estimates dataset-specific operator utility. The resulting candidate pool concentrates computation on selected operator–input regions. This supports the broader point that operator quality and pair selection can be treated separately. Its principal objective is to reduce a fixed, OpenFE-compatible candidate space; EvoFG instead evolves the structures themselves and uses family-level evidence to condition later proposals.

CAAFE. Context-Aware Automated Feature Engineering prompts an LLM with a dataset description to generate Python transformations and textual explanations of why the new features may help (Hollmann et al., 2023b). An evaluation loop provides feedback for further generation. The program operates on concrete dataset fields, and semantic context is an intentional source of domain knowledge. EvoFG moves concrete column selection into a local binding optimizer, so semantic field descriptions are unnecessary for its proposal interface. This difference concerns information requirements and search organization; it does not establish that CAAFE's use of context causes contamination.

FeatLLM. FeatLLM uses the LLM as a feature engineer for few-shot tabular learning (Han et al., 2024). It generates predictive rules, converts their outputs into features, and trains a simple downstream model to estimate class likelihoods. The LLM is not queried for every test example: the learned rule features and downstream predictor handle inference. Multiple generated rules can already provide multiple features per generation process. The setting and optimization target differ from evolving an unbound numerical operator and systematically evaluating its legal column bindings.

OCTree. OCTree iteratively optimizes feature-generation rules using validation scores and decision-tree reasoning (Nam et al., 2024). A proposed rule creates a new column; a predictor is trained, and tree-based descriptions of its behavior are returned to the LLM as feedback alongside past rules and scores. The best rule is retained after a fixed search budget. Its feedback is richer than a scalar reward, and its experiments include both context-aware and context-agnostic inputs. Thus, neither rich feedback nor anonymous inputs alone distinguish EvoFG. The latter explicitly holds a numerical structure fixed while comparing its binding family.

LLM-FE. LLM-FE treats feature-transformation programs as evolutionary individuals (Abhyankar et al., 2025). The LLM proposes modifications conditioned on evaluated programs, and island-based program memory preserves useful solutions for subsequent search. Program execution and downstream evaluation ground the evolutionary feedback. A complete program can contain several transformations; the distinction is not that only EvoFG can yield multiple features. Instead, EvoFG exposes the binding dimension of a reusable operator as a separate local optimization problem and reports evidence across its instances.

ELLM-FT. ELLM-FT combines an evolutionary multi-population database with LLM generation of feature-transformation sequences (Gong et al., 2025). An RL data collector supplies transformation experiences; evolutionary maintenance prioritizes stronger populations, and few-shot prompts use their sequences to guide generation. In these sequences, operations and feature identities jointly specify a concrete transformation. EvoFG places column identities outside the structural proposal and reoptimizes them for each child program.

EFE and TOPOFE. EFE represents preprocessing transformations with a standardized Python fit/transform interface and evolves programs using dataset context, summary statistics, and validation feedback (Taga et al., 2026). A standard interface enables pipeline reuse, but does not by itself separate a program from its input bindings. TOPOFE partitions the transformation space into semantically coherent islands, adapts prompts from acceptance feedback, and learns a directed graph for useful cross-island transfer (Li & Ramakrishnan, 2026). Its transformation families group related program categories. An EvoFG family instead consists of different column bindings of one fixed program. These are different organizational levels and can be complementary.

Table 5: Search objects and feedback organization. The comparison concerns mechanisms described in the cited methods, not a claim of matched experimental protocols.

Method	Structure source	Input selection / feedback	Difference from EvoFG
AutoFeat	Fixed compositions	Program construction and selection	No open code evolution
OpenFE	Fixed operator library	Enumeration and pruning	Structures do not evolve
GRFG	Predefined operations	Cascaded group/operation policies	Joint feature crossing decisions
SCOPE-FE	Fixed operators	Pair gating and operator probing	Pre-generation space reduction
CAAFE	Context-based code	Concrete fields and performance	No separate binding optimizer
FeatLLM	Generated rules	Rule features and simple predictor	Few-shot rule learning
OCTree	Generated rules	Tree reasoning and validation	Concrete rule programs
LLM-FE	Evolving programs	Island memory and scores	Whole-program individuals
ELLM-FT	Transformation sequences	Population-based sequence feedback	Inputs included in sequences
EFE	Fit/transform programs	Context and validation feedback	Bound preprocessing programs
TOPOFE	Multi-island programs	Adaptive memory and transfer	Category-level families
EvoFG	Unbound unary/binary code	Binding search and family summaries	Structural evolution over reusable operators

A.2 Reusable Structures and Internal Optimization

LLM-SR proposes equation programs in an outer loop and fits continuous coefficients in an inner numerical optimization loop (Shojaee et al., 2025). This is a direct precedent for assigning different variables to different optimizers. In EvoFG, the inner variables are discrete columns rather than coefficients. Ordered pair enumeration, finite fine-evaluation quotas, and binding mismatch therefore replace continuous fitting as the central concerns. LILO studies reusable program libraries through code compression and documentation (Grand et al., 2024). Reusability is shared motivation, but our evaluation unit is specifically a structure's family of tabular input bindings.

A.3 Program Generation and Evaluation Feedback

LLM program synthesis (Austin et al., 2021), Self-Refine (Madaan et al., 2023), and Reflexion (Shinn et al., 2023) provide precedents for program proposals, iterative correction, and memory. Self-generated language feedback differs from the training-data evaluation used here. Dynamic and Adaptive Feature Generation with LLM also studies multi-agent search organization (Zhang et al., 2025). These connections delimit our claim: the contribution is the combination of operator-family search, programmatic rebinding, and family feedback. Direct-Batch further tests whether simply permitting multiple candidates per request reproduces the same search organization.

B Formalizing Search and Approximate Evaluation

B.1 Feasible Space and Utility

Let $\mathcal{O}$ be the set of structures satisfying the interface and validity constraints. Each $o \in \mathcal{O}$ has a finite nonempty binding set $\mathcal{C}(o)$, and

$$\mathcal{A} = \{(o, c) : o \in \mathcal{O},\ c \in \mathcal{C}(o)\}. \tag{9}$$

A binding maps input slots to data columns and yields $z_{o,c}$. If two instances have identical output vectors on training matrix X, write $z_{o,c} \sim_X z_{o',c'}$ and treat them as one output in evaluation caching and candidate counting. This is equivalence on a given matrix, not equivalence on every possible input.

Fix a learner, a reference feature set, and the metric direction. Let $u(o, c)$ denote the target utility of adding an instance, and $\widehat{u}(o, c)$ its training-validation estimate. Positive utility denotes improvement. Define

$$V(o) = \max_{c \in \mathcal{C}(o)} u(o, c), \qquad V^\star = \sup_{o \in \mathcal{O}} V(o). \tag{10}$$

Assume $V^\star$ is finite. Target utility and empirical gain share the same reference and sign convention, but empirical gain is affected by finite samples, the evaluator, and adaptive selection. Final selection permits no augmentation when no positive candidate is found.

B.2 Nested Objective and Loss Decomposition

Proposition 1 (Equivalence of nested and joint objectives)**.** *For the feasible space above,*

$$\sup_{(o,c) \in \mathcal{A}} u(o, c) = \sup_{o \in \mathcal{O}} \max_{c \in \mathcal{C}(o)} u(o, c) = V^\star. \tag{11}$$

Proof. Every feasible instance satisfies $u(o, c) \le V(o) \le V^\star$, so the joint supremum is at most $V^\star$. Conversely, for each $\delta > 0$ there is a structure o_δ with $V(o_\delta) > V^\star - \delta$. Its finite binding set contains a maximizer, so the joint supremum is at least $V^\star - \delta$. Letting $\delta \to 0$ gives the result. □

Adding and subtracting $V(o)$ gives

$$V^\star - u(o, c) = [V^\star - V(o)] + [V(o) - u(o, c)]. \tag{12}$$

Both terms are nonnegative. Even with low structural loss, binding loss may be large, so one poor instance cannot invalidate its structure. Decoupled search seeks a better estimate of $V(o)$ under finite computation. This identity is not a convergence or speed guarantee.

B.3 Screening Omissions and Evaluation Error

For a nonempty completed shortlist $\widetilde{\mathcal{C}}(o)$, define

$$\widetilde{V}(o) = \max_{c \in \widetilde{\mathcal{C}}(o)} u(o,c), \qquad \widehat{V}(o) = \max_{c \in \widetilde{\mathcal{C}}(o)} \widehat{u}(o,c), \tag{13}$$

$$R_{\mathrm{scr}}(o) = V(o) - \widetilde{V}(o) \geq 0. \tag{14}$$

If the evaluated set satisfies

$$\max_{c \in \widetilde{\mathcal{C}}(o)} |\widehat{u}(o,c) - u(o,c)| \leq \epsilon, \tag{15}$$

then $|\widehat{V}(o) - \widetilde{V}(o)| \leq \epsilon$. For $\widehat{c} \in \arg\max_{c \in \widetilde{\mathcal{C}}(o)} \widehat{u}(o,c)$,

$$0 \leq V(o) - u(o,\widehat{c}) \leq R_{\mathrm{scr}}(o) + 2\epsilon. \tag{16}$$

To see this, let $\widetilde{c}$ maximize true utility within the shortlist. Then

$$u(o,\widetilde{c}) \leq \widehat{u}(o,\widetilde{c}) + \epsilon \leq \widehat{u}(o,\widehat{c}) + \epsilon \leq u(o,\widehat{c}) + 2\epsilon. \tag{17}$$

Adding the screening loss proves the bound. It distinguishes coverage from estimation accuracy, but does not establish a uniform error bound for adaptively reused cross-validation. Appendix F.3 measures omissions under the same empirical evaluator; Appendices F.4 and F.6 examine label-permutation and independent-holdout reliability.

B.4 Family Feedback and the Final Representation

Monotone normalization and clipping give $g \in [0,1]$. Sort output-deduplicated family gains and pad missing entries with zero. For $h \in \mathbb{N}_{>0}$,

$$P(o) = g_{(1)}, \qquad S(o) = \frac{1}{h} \sum_{j=2}^{h+1} g_{(j)}. \tag{18}$$

Ordering gives $0 \leq S \leq P \leq 1$. Holding existing estimates fixed, adding a new output cannot decrease any order statistic, and hence cannot decrease peak or support. This is a property of observed summaries; bindings need not be independent, and larger search does not necessarily lower generalization error.

The archive uses (P, S) to organize families and peak to choose within cells. Global selection pools instances from all structures, deduplicates outputs, and keeps at most K with positive empirical gain. If $U(\mathcal{Z})$ is joint augmentation utility, generally

$$U(\mathcal{Z}) \neq \sum_{z \in \mathcal{Z}} u(z). \tag{19}$$

Family summaries organize search and individual scores allocate capacity; the final representation requires separate downstream evaluation.

C Datasets and Experimental Protocol

C.1 Datasets and Task Metrics

The dataset overview lists sample count, input dimension, task type, source collection, and split. It includes exactly the union of Tables 1 and 2: 27 datasets, with 22 eligible for TabPFN. Dimensions refer to the experimental inputs. IID denotes independent and identically distributed splitting; TIME denotes temporal splitting. Main tables report accuracy for classification and RMSE for regression. Search retains its registered metric: accuracy for Academic, balance-scale, car, covtype, jungle_chess, and vehicle; AUC for the other classification tasks; RMSE for regression.

Table 6: Classification dataset overview. Sample and feature counts refer to the registered experimental inputs.

Dataset	n	p	Task type	Source	Split
Academic	4,424	36	Multi-class (3)	Kaggle	IID
adult	32,561	14	Binary	Kaggle	IID
airlines	539,383	7	Binary	OpenML	IID
balance-scale	625	4	Multi-class (3)	Kaggle	IID
car	1,728	6	Multi-class (4)	Kaggle	IID
compass	16,644	17	Binary	OpenML	IID
covtype	581,012	54	Multi-class (7)	OpenML	IID
Higgs	940,160	24	Binary	OpenML	IID
jungle_chess	44,819	6	Multi-class (3)	OpenML	IID
kdd_ipums_la	5,188	20	Binary	OpenML	IID
kddCup09	5,032	45	Binary	OpenML	IID
MagicTelescope	13,376	10	Binary	OpenML	IID
pc1	1,109	21	Binary	OpenML	IID
pharyngitis	676	18	Binary	Kaggle	IID
pol	10,082	26	Binary	OpenML	IID
rl	4,970	12	Binary	OpenML	IID
road-safety	111,762	32	Binary	OpenML	IID
vehicle	846	18	Multi-class (4)	Kaggle	IID

Table 7: Regression dataset overview. Sample and feature counts refer to the registered experimental inputs.

Dataset	n	p	Task type	Source	Split
airfoil_self_noise	1,503	5	Regression	Kaggle	IID
bike	17,379	12	Regression	UCI	TIME
communities	1,994	122	Regression	UCI	IID
diamonds	53,940	9	Regression	Kaggle	IID
Enefit	2,017,824	11	Regression	Kaggle	TIME
forest-fires	517	12	Regression	Kaggle	IID
housing	20,640	9	Regression	Kaggle	IID
insurance	1,338	6	Regression	Kaggle	IID
Tesla_Stock	3,471	14	Regression	Kaggle	TIME

C.2 Evaluation Units and Implementation Settings

C.2.1 Five outer experiments and aggregation

Each dataset has five planned outer experimental units with fixed training/test boundaries. Feature search sees training data only, and transformations are frozen before downstream testing. Each outer unit uses model seeds $s \in \{42, 52, 62\}$; their average constitutes one outer score rather than three independent experiments. If $q_{d,m,b,r,s}$ is the score for dataset d, method m, predictor b, outer unit r, and seed s, then

$$\widehat{q}^{(r)}_{d,m,b} = \frac{1}{3} \sum_{s} q_{d,m,b,r,s}. \tag{20}$$

Let $\mathcal{R}_{d,m}$ contain the outer units with all three seeds available for all three conventional predictors. Table 1 is

$$\overline{q}_{d,m} = \frac{1}{3} \sum_{b \in \{\text{LGBM,XGB,MLP}\}} \frac{1}{|\mathcal{R}_{d,m}|} \sum_{r \in \mathcal{R}_{d,m}} \widehat{q}^{(r)}_{d,m,b}. \tag{21}$$

The model-specific tables use this same intersection, so their unrounded means average exactly to the main table. TabPFN uses its own available outer units. Partial coverage uses the actual denominator; failed or missing scores are never filled with zero.

C.2.2 LLM, search, and binding evaluation

Table 8: LLM and feature-search settings.

Item	Setting
LLM backbone	DeepSeek V4.1 Flash; `deepseek-v4.1-flash`
Proposal ceiling	$Q_{\max} = 50$ for EvoFG / Direct / Direct-Batch
Generation	Temperature 1.0; top-p 0.95; max tokens 8,192
Reasoning / fallback	Thinking disabled; no model fallback
Predictor reuse	Identical frozen features for Tables 1 and 2
Initialization	Unary $\log(1 + \lvert a \rvert)$ and binary ab
Arity allocation	25 unary and 25 binary proposal opportunities
Fine evaluator	Three-fold LightGBM; 128 trees; learning rate 0.05; 31 leaves; max bin 63
Binding quotas	All if $\lvert\mathcal{C}\rvert \leq 64$; otherwise 64, or 128 if $\lvert\mathcal{C}\rvert > 4096$
Family support	Two non-peak elite alternatives ($h = 2$)
Archive	Two islands, separately grouped by arity
Descriptor bins	0, 0.001, 0.005, 0.02, 0.1, 1.0
Within-cell selection	Higher peak; deterministic identity tie-break
Final cap	At most 20 unique outputs with positive signed fine gain

Invalid proposals consume budget. Each request produces one operator in EvoFG, one concrete feature proposal in Direct, or up to ten candidates in Direct-Batch. Native settings are 50 steps for OCTree, ten rounds for CAAFE, and five ensemble members for FeatLLM; these units do not imply identical actual calls. Base, AutoFeat, and OpenFE do not use the LLM backbone in the controlled search. No extra search is run for TabPFN.

Fine-evaluation gains follow the normalization in Section 3. Descriptors are clipped to $[0, 1]$; signed gains determine candidate positivity and final ranking. Common internal scaling is removed in the reported descriptors. The proxy's nominal 12,000-row cap applies to its IID path; temporal/group paths are not assumed to perform the same truncation. Binary enumeration is $O(d^2)$. Evaluating b new unique candidates through three folds entails approximately $3b$ additional model fits, plus proxy work and reusable baseline evaluation. Caching, duplicates, and invalid outputs alter the realized count.

C.2.3 Downstream predictors and preprocessing

Numerical imputation uses training statistics. Categorical mappings are fitted on training categories and include unknown-value handling. MLP input scaling and regression-target scaling are fitted on training data. TabPFN uses local classification and regression checkpoints, with configured limits of

Table 9: Downstream predictor configurations.

Predictor	Configuration
LightGBM	500 trees; learning rate 0.05; 31 leaves; max bin 63
XGBoost	500 trees; depth 6; learning rate 0.05; histogram method; max bin 256
MLP	scikit-learn; hidden layers 64/64; ReLU; Adam; learning rate 0.001; alpha 0.0001; batch 256; 500 iterations; no early stopping
TabPFN-2.5	Package 6.4.1; eight estimators; FP32; fit preprocessors; memory saving enabled

50,000 training samples, 2,000 augmented features, and ten classes. The selected features remain frozen when the predictor changes.

D Predictive Results by Downstream Model

D.1 Conventional Predictors: Means and Dispersion

Tables 10–12 report LightGBM, XGBoost, and MLP on the same 27 datasets, in the same order as Table 1. All three use identical available outer units for each dataset–method pair. Their unrounded means therefore satisfy

$$\mu_{\mathrm{LGBM}} + \mu_{\mathrm{XGB}} + \mu_{\mathrm{MLP}} = 3\,\overline{q}_{\text{Table 1}}. \tag{22}$$

For seed-aggregated results, first average each internal seed over the common outer set $\mathcal{F}$, then compute

$$a_s = \frac{1}{|\mathcal{F}|}\sum_{f\in\mathcal{F}} q_{f,s}, \qquad \mu = \frac{1}{3}\sum_{s} a_s, \qquad \mathrm{SD}_{\mathrm{seed}} = \sqrt{\frac{1}{2}\sum_{s}(a_s-\mu)^2}. \tag{23}$$

Dispersion entries report standard deviations. For seed-aggregated scores, the formula above measures variation among the three seed means; for per-fit summaries, it measures variation among the individual evaluations. Neither is a confidence interval. A displayed 0.000 can represent a small nonzero value. Rounding can prevent exact arithmetic on displayed means; the unrounded means agree by construction.

Table 10: LightGBM results (mean ± standard deviation) on exactly the 27 datasets in Table 1. The three model means reconstruct Table 1 by equal averaging before rounding. See Appendix D.1 for dispersion definitions.

Classification: accuracy ↑

Dataset	Base	AutoFeat	OpenFE	OCTree	CAAFE	FeatLLM	Direct	Direct-Batch	EvoFG
rl	0.785 ± 0.000	—	×	0.777 ± 0.011	0.778 ± 0.012	0.779 ± 0.010	0.782 ± 0.000	0.775 ± 0.000	**0.820** ± 0.000
compass	0.762 ± 0.000	—	×	0.726 ± 0.008	0.724 ± 0.009	0.726 ± 0.009	0.765 ± 0.000	0.769 ± 0.000	**0.792** ± 0.000
balance-scale	0.869 ± 0.000	0.862 ± 0.000	0.862 ± 0.000	0.870 ± 0.020	0.916 ± 0.023	0.878 ± 0.023	0.877 ± 0.000	0.888 ± 0.000	**0.934** ± 0.000
jungle_chess	0.872 ± 0.000	0.877 ± 0.000	0.884 ± 0.000	0.880 ± 0.003	0.858 ± 0.003	0.858 ± 0.003	0.873 ± 0.000	0.873 ± 0.000	**0.916** ± 0.000
vehicle	0.771 ± 0.000	0.780 ± 0.000	0.769 ± 0.000	0.754 ± 0.026	0.755 ± 0.018	0.747 ± 0.023	0.769 ± 0.000	0.771 ± 0.000	**0.806** ± 0.000
MagicTelescope	0.861 ± 0.000	×	×	0.850 ± 0.006	0.849 ± 0.006	0.850 ± 0.006	0.862 ± 0.000	0.865 ± 0.000	**0.870** ± 0.000
kdd_ipums_la	0.882 ± 0.000	×	×	0.873 ± 0.009	0.874 ± 0.011	0.876 ± 0.008	**0.883** ± 0.000	0.881 ± 0.000	0.881 ± 0.000
Higgs	0.736 ± 0.000	0.736 ± 0.000	0.736 ± 0.000	0.720 ± 0.001	0.720 ± 0.001	0.720 ± 0.001	0.736 ± 0.000	0.736 ± 0.000	**0.739** ± 0.000
Academic	0.773 ± 0.000	0.772 ± 0.000	0.774 ± 0.000	0.759 ± 0.006	0.761 ± 0.007	0.761 ± 0.009	**0.775** ± 0.000	0.775 ± 0.000	0.771 ± 0.000
car	0.997 ± 0.000	0.994 ± 0.000	0.997 ± 0.000	0.984 ± 0.005	0.984 ± 0.004	0.983 ± 0.005	0.999 ± 0.000	**1.000** ± 0.000	0.998 ± 0.000
airlines	0.663 ± 0.000	0.664 ± 0.000	0.664 ± 0.000	0.650 ± 0.002	0.650 ± 0.001	0.649 ± 0.002	0.662 ± 0.003	0.663 ± 0.000	**0.665** ± 0.000
kddCup09	0.791 ± 0.000	×	×	0.783 ± 0.012	0.781 ± 0.012	0.781 ± 0.009	0.791 ± 0.000	0.788 ± 0.000	**0.791** ± 0.000
adult	0.871 ± 0.000	×	×	0.864 ± 0.005	0.863 ± 0.005	0.864 ± 0.005	0.872 ± 0.000	0.868 ± 0.000	**0.872** ± 0.000
pc1	0.933 ± 0.000	×	×	0.921 ± 0.010	0.921 ± 0.011	0.921 ± 0.009	0.932 ± 0.000	**0.935** ± 0.000	0.933 ± 0.000
covtype	**0.855** ± 0.014	—	—	—	—	—	0.851 ± 0.014	0.768 ± 0.074	0.812 ± 0.041
road-safety	**0.790** ± 0.000	—	—	—	—	—	0.719 ± 0.063	0.719 ± 0.025	0.730 ± 0.046
pol	0.985 ± 0.000	×	×	0.975 ± 0.003	0.975 ± 0.003	0.976 ± 0.003	0.985 ± 0.000	**0.988** ± 0.000	0.986 ± 0.000
pharyngitis	**0.663** ± 0.000	0.661 ± 0.000	0.638 ± 0.000	0.652 ± 0.044	0.643 ± 0.031	0.646 ± 0.038	0.647 ± 0.000	0.645 ± 0.000	0.652 ± 0.000

Regression: RMSE ↓

Dataset	Base	AutoFeat	OpenFE	Direct	Direct-Batch	EvoFG
diamonds	546.212 ± 0.000	545.329 ± 0.000	545.816 ± 0.000	540.278 ± 0.000	532.403 ± 0.000	**526.843** ± 0.000
airfoil_self_noise	1.674 ± 0.000	1.634 ± 0.000	1.674 ± 0.000	1.683 ± 0.000	1.720 ± 0.000	**1.569** ± 0.000
Enefit	**328.258** ± 1.088	—	—	340.427 ± 4.175	353.491 ± 7.735	334.900 ± 8.108
forest-fires	67.801 ± 0.000	69.411 ± 0.000	66.575 ± 0.000	68.215 ± 0.000	70.392 ± 0.000	**66.042** ± 0.000
insurance	**5079.181** ± 0.000	5126.125 ± 0.000	5126.125 ± 0.000	5103.805 ± 0.000	5129.639 ± 0.000	5100.697 ± 0.000
housing	46882.588 ± 0.000	46948.423 ± 0.000	46943.163 ± 0.000	45560.529 ± 0.000	**44167.469** ± 0.000	44561.353 ± 0.000
Tesla_Stock	44.313 ± 0.000	—	—	44.300 ± 0.000	44.075 ± 0.000	**42.963** ± 0.000
bike	73.676 ± 0.000	**30.598** ± 0.000	73.850 ± 0.000	73.693 ± 0.000	73.656 ± 0.000	73.003 ± 0.000
communities	**0.138** ± 0.000	**0.138** ± 0.000	0.140 ± 0.000	0.139 ± 0.000	0.138 ± 0.000	0.138 ± 0.000

Table 11: XGBoost results (mean ± standard deviation) on exactly the 27 datasets in Table 1. The three model means reconstruct Table 1 by equal averaging before rounding. See Appendix D.1 for dispersion definitions.

Classification: accuracy ↑

Dataset	Base	AutoFeat	OpenFE	OCTree	CAAFE	FeatLLM	Direct	Direct-Batch	EvoFG
rl	0.790 ± 0.000	—	×	0.779 ± 0.011	0.774 ± 0.012	0.778 ± 0.011	0.791 ± 0.000	0.776 ± 0.000	**0.823** ± 0.000
compass	0.749 ± 0.000	—	×	0.743 ± 0.007	0.733 ± 0.008	0.744 ± 0.008	0.754 ± 0.000	0.755 ± 0.000	**0.779** ± 0.000
balance-scale	0.853 ± 0.000	0.854 ± 0.000	0.864 ± 0.000	0.877 ± 0.019	0.913 ± 0.022	0.869 ± 0.027	0.878 ± 0.000	0.882 ± 0.000	**0.944** ± 0.000
jungle_chess	0.868 ± 0.000	0.871 ± 0.000	0.875 ± 0.000	**0.907** ± 0.002	0.860 ± 0.003	0.859 ± 0.002	0.867 ± 0.000	0.868 ± 0.000	0.901 ± 0.000
vehicle	0.762 ± 0.000	0.762 ± 0.000	0.775 ± 0.000	0.743 ± 0.023	0.751 ± 0.019	0.744 ± 0.024	0.772 ± 0.000	0.761 ± 0.000	**0.798** ± 0.000
MagicTelescope	0.862 ± 0.000	×	×	0.850 ± 0.007	0.847 ± 0.005	0.851 ± 0.007	0.866 ± 0.000	0.863 ± 0.000	**0.872** ± 0.000
kdd_ipums_la	0.883 ± 0.000	×	×	0.871 ± 0.009	0.870 ± 0.008	0.872 ± 0.010	**0.885** ± 0.000	0.885 ± 0.000	0.884 ± 0.000
Higgs	0.738 ± 0.000	0.738 ± 0.000	0.738 ± 0.000	0.727 ± 0.001	0.727 ± 0.001	0.727 ± 0.001	0.738 ± 0.000	0.738 ± 0.000	**0.740** ± 0.000
Academic	0.775 ± 0.000	**0.776** ± 0.000	0.773 ± 0.000	0.756 ± 0.008	0.756 ± 0.009	0.755 ± 0.014	0.774 ± 0.000	0.775 ± 0.000	0.775 ± 0.000
car	0.994 ± 0.000	0.994 ± 0.000	0.995 ± 0.000	0.981 ± 0.006	0.981 ± 0.006	0.981 ± 0.006	0.996 ± 0.000	0.997 ± 0.000	**0.997** ± 0.000
airlines	0.666 ± 0.000	0.666 ± 0.000	0.666 ± 0.000	0.657 ± 0.002	0.657 ± 0.001	0.657 ± 0.001	0.666 ± 0.000	0.666 ± 0.000	**0.667** ± 0.000
kddCup09	0.790 ± 0.000	×	×	0.776 ± 0.009	0.779 ± 0.014	0.779 ± 0.013	0.793 ± 0.000	0.787 ± 0.000	**0.793** ± 0.000
adult	0.872 ± 0.000	×	×	0.861 ± 0.006	0.860 ± 0.004	0.861 ± 0.006	**0.873** ± 0.000	0.870 ± 0.000	0.872 ± 0.000
pc1	0.931 ± 0.000	×	×	0.922 ± 0.011	0.922 ± 0.013	0.922 ± 0.012	**0.932** ± 0.000	0.931 ± 0.000	0.931 ± 0.000
covtype	0.852 ± 0.000	—	—	—	—	—	0.853 ± 0.000	0.854 ± 0.000	**0.863** ± 0.000
road-safety	0.788 ± 0.000	—	—	—	—	—	0.789 ± 0.000	0.790 ± 0.000	**0.803** ± 0.000
pol	0.985 ± 0.000	×	×	0.976 ± 0.003	0.975 ± 0.003	0.976 ± 0.003	0.985 ± 0.000	0.985 ± 0.000	**0.986** ± 0.000
pharyngitis	0.641 ± 0.000	0.651 ± 0.000	0.649 ± 0.000	0.635 ± 0.030	0.643 ± 0.042	0.637 ± 0.024	0.648 ± 0.000	**0.654** ± 0.000	0.648 ± 0.000

Regression: RMSE ↓

Dataset	Base	AutoFeat	OpenFE	Direct	Direct-Batch	EvoFG
diamonds	534.719 ± 0.000	532.620 ± 0.000	534.267 ± 0.000	532.561 ± 0.000	531.093 ± 0.000	**521.889** ± 0.000
airfoil_self_noise	1.566 ± 0.000	**1.500** ± 0.000	1.676 ± 0.000	1.609 ± 0.000	1.643 ± 0.000	1.586 ± 0.000
Enefit	331.970 ± 0.000	—	—	349.820 ± 0.000	337.537 ± 0.000	**325.721** ± 0.000
forest-fires	82.097 ± 0.000	88.957 ± 0.000	95.567 ± 0.000	**81.028** ± 0.000	83.132 ± 0.000	90.047 ± 0.000
insurance	5136.594 ± 0.000	5121.071 ± 0.000	5121.071 ± 0.000	5153.060 ± 0.000	5178.566 ± 0.000	**5084.987** ± 0.000
housing	46305.569 ± 0.000	46218.168 ± 0.000	46306.117 ± 0.000	45494.981 ± 0.000	**44102.051** ± 0.000	44232.201 ± 0.000
Tesla_Stock	39.870 ± 0.000	—	—	39.886 ± 0.000	39.987 ± 0.000	**39.149** ± 0.000
bike	76.412 ± 0.000	**27.227** ± 0.000	76.576 ± 0.000	76.813 ± 0.000	77.327 ± 0.000	74.232 ± 0.000
communities	0.139 ± 0.000	0.139 ± 0.000	0.140 ± 0.000	0.140 ± 0.000	0.140 ± 0.000	**0.139** ± 0.000

Table 12: MLP results (mean ± standard deviation) on exactly the 27 datasets in Table 1. The three model means reconstruct Table 1 by equal averaging before rounding. See Appendix D.1 for dispersion definitions.

Classification: accuracy ↑

Dataset	Base	AutoFeat	OpenFE	OCTree	CAAFE	FeatLLM	Direct	Direct-Batch	EvoFG
rl	0.664 ± 0.002	—	×	0.648 ± 0.016	0.645 ± 0.019	0.668 ± 0.014	0.661 ± 0.001	0.646 ± 0.001	**0.725** ± 0.005
compass	0.725 ± 0.003	—	×	0.694 ± 0.006	0.688 ± 0.006	0.694 ± 0.005	0.726 ± 0.004	0.742 ± 0.003	**0.764** ± 0.004
balance-scale	0.973 ± 0.002	0.973 ± 0.002	0.974 ± 0.000	0.940 ± 0.014	0.962 ± 0.016	0.936 ± 0.018	0.974 ± 0.003	0.976 ± 0.002	**0.983** ± 0.002
jungle_chess	0.889 ± 0.005	0.888 ± 0.003	0.890 ± 0.003	0.845 ± 0.004	0.849 ± 0.003	0.849 ± 0.004	0.886 ± 0.003	0.878 ± 0.001	**0.895** ± 0.002
vehicle	0.826 ± 0.002	**0.827** ± 0.008	0.824 ± 0.004	0.722 ± 0.026	0.739 ± 0.027	0.724 ± 0.028	0.822 ± 0.005	0.816 ± 0.015	0.811 ± 0.013
MagicTelescope	0.844 ± 0.006	×	×	0.841 ± 0.006	0.839 ± 0.006	0.842 ± 0.007	0.845 ± 0.001	0.842 ± 0.001	**0.846** ± 0.002
kdd_ipums_la	0.830 ± 0.002	×	×	**0.851** ± 0.006	0.840 ± 0.007	0.840 ± 0.008	0.830 ± 0.002	0.818 ± 0.002	0.845 ± 0.005
Higgs	0.751 ± 0.001	**0.752** ± 0.000	0.751 ± 0.000	0.727 ± 0.001	0.727 ± 0.001	0.727 ± 0.001	0.750 ± 0.000	0.751 ± 0.000	0.751 ± 0.001
Academic	0.711 ± 0.003	0.711 ± 0.009	0.713 ± 0.006	0.714 ± 0.012	0.716 ± 0.013	0.713 ± 0.013	0.712 ± 0.005	0.708 ± 0.005	**0.719** ± 0.002
car	0.990 ± 0.002	0.989 ± 0.001	0.991 ± 0.001	0.943 ± 0.011	0.944 ± 0.011	0.949 ± 0.009	0.989 ± 0.002	0.982 ± 0.002	**0.994** ± 0.000
airlines	0.651 ± 0.000	0.652 ± 0.001	0.652 ± 0.000	0.636 ± 0.002	0.636 ± 0.002	0.639 ± 0.001	0.652 ± 0.000	0.652 ± 0.000	**0.652** ± 0.001
kddCup09	0.719 ± 0.005	×	×	0.728 ± 0.014	0.726 ± 0.015	**0.728** ± 0.016	0.718 ± 0.006	0.718 ± 0.007	0.720 ± 0.004
adult	0.828 ± 0.002	×	×	0.839 ± 0.005	0.839 ± 0.005	**0.841** ± 0.005	0.826 ± 0.001	0.825 ± 0.001	0.829 ± 0.001
pc1	0.922 ± 0.002	×	×	0.920 ± 0.007	0.919 ± 0.010	0.920 ± 0.009	**0.925** ± 0.001	0.924 ± 0.001	0.924 ± 0.005
covtype	0.912 ± 0.001	—	—	—	—	—	0.912 ± 0.001	0.909 ± 0.002	**0.914** ± 0.001
road-safety	**0.773** ± 0.001	—	—	—	—	—	0.772 ± 0.002	0.771 ± 0.002	0.770 ± 0.001
pol	**0.987** ± 0.000	×	×	0.975 ± 0.003	0.974 ± 0.003	0.976 ± 0.003	0.986 ± 0.000	0.986 ± 0.002	0.985 ± 0.001
pharyngitis	0.610 ± 0.014	0.607 ± 0.003	0.611 ± 0.008	0.629 ± 0.031	0.622 ± 0.034	**0.635** ± 0.032	0.614 ± 0.009	0.614 ± 0.007	0.615 ± 0.002

Regression: RMSE ↓

Dataset	Base	AutoFeat	OpenFE	Direct	Direct-Batch	EvoFG
diamonds	598.075 ± 25.132	599.171 ± 19.675	579.485 ± 16.605	594.923 ± 20.316	601.445 ± 21.801	**569.238** ± 9.335
airfoil_self_noise	1.681 ± 0.069	1.663 ± 0.033	1.700 ± 0.012	1.648 ± 0.061	1.703 ± 0.101	**1.540** ± 0.021
Enefit	666.414 ± 36.296	—	—	**570.725** ± 25.857	611.083 ± 80.087	586.767 ± 103.575
forest-fires	88.943 ± 4.418	95.290 ± 7.910	89.633 ± 4.191	94.439 ± 6.058	90.571 ± 1.168	**83.799** ± 1.773
insurance	**5126.294** ± 57.950	5295.624 ± 13.383	5276.450 ± 35.486	5249.121 ± 61.574	5513.115 ± 48.554	5239.890 ± 21.098
housing	52249.840 ± 327.834	52678.044 ± 307.898	53137.848 ± 426.477	51621.966 ± 135.903	**50002.927** ± 142.750	52707.013 ± 404.790
Tesla_Stock	**11.300** ± 6.113	—	—	1039.182 ± 1291.668	118.824 ± 39.662	30.048 ± 0.609
bike	135.916 ± 12.518	**40.766** ± 1.934	156.945 ± 15.015	136.890 ± 16.190	146.496 ± 8.524	136.818 ± 1.312
communities	0.167 ± 0.006	0.167 ± 0.006	0.168 ± 0.003	0.170 ± 0.004	**0.167** ± 0.004	0.169 ± 0.002

D.2 TabPFN: Means and Dispersion

Table 13 contains exactly the 22 datasets and corresponding means in Table 2. Removing the ± lines reproduces that main table. All methods use actual TabPFN evaluations; OCTree, CAAFE, and FeatLLM each cover 11 classification datasets. The mean and sample standard deviation are calculated across the three seed means after averaging common available outer units, following Appendix D.1.

Table 13: TabPFN-2.5 mean ± standard deviation on exactly the 22 datasets in Table 2. Removing the standard-deviation line gives the same means and ranking marks as Table 2. Standard deviations follow Appendix D.1.

Classification: accuracy ↑

Dataset	Base	AutoFeat	OpenFE	OCTree	CAAFE	FeatLLM	Direct	Direct-Batch	EvoFG
rl	0.861 ± 0.002	—	×	0.850 ± 0.002	0.849 ± 0.002	0.849 ± 0.002	0.862 ± 0.003	0.856 ± 0.001	**0.876** ± 0.002
compass	0.842 ± 0.007	—	×	0.832 ± 0.007	0.817 ± 0.018	0.837 ± 0.009	0.836 ± 0.003	0.831 ± 0.007	**0.870** ± 0.002
balance-scale	0.978 ± 0.000	0.993 ± 0.003	0.978 ± 0.002	0.970 ± 0.001	0.988 ± 0.000	0.974 ± 0.002	**0.998** ± 0.002	0.996 ± 0.001	0.997 ± 0.000
jungle_chess	0.877 ± 0.005	0.877 ± 0.001	0.880 ± 0.001	**0.909** ± 0.012	0.871 ± 0.004	0.865 ± 0.002	0.877 ± 0.004	0.884 ± 0.003	0.907 ± 0.003
vehicle	**0.878** ± 0.004	0.874 ± 0.001	0.876 ± 0.004	0.866 ± 0.001	0.870 ± 0.001	0.868 ± 0.001	0.878 ± 0.006	0.876 ± 0.001	0.865 ± 0.002
MagicTelescope	0.881 ± 0.001	×	×	0.871 ± 0.000	0.871 ± 0.001	0.871 ± 0.000	0.883 ± 0.001	0.882 ± 0.001	**0.883** ± 0.001
kdd_ipums_la	0.887 ± 0.001	×	×	0.876 ± 0.001	0.876 ± 0.001	0.876 ± 0.002	0.887 ± 0.000	0.886 ± 0.001	**0.887** ± 0.001
Academic	**0.794** ± 0.002	0.793 ± 0.001	0.793 ± 0.003	0.783 ± 0.003	0.782 ± 0.001	0.783 ± 0.002	0.792 ± 0.001	0.793 ± 0.002	0.790 ± 0.002
car	0.996 ± 0.003	0.995 ± 0.001	0.997 ± 0.001	0.985 ± 0.001	0.988 ± 0.000	0.987 ± 0.001	0.996 ± 0.001	0.999 ± 0.000	**1.000** ± 0.001
kddCup09	0.806 ± 0.001	×	×	—	—	—	**0.808** ± 0.001	0.807 ± 0.001	0.804 ± 0.002
adult	0.867 ± 0.001	×	×	0.856 ± 0.001	0.857 ± 0.000	0.857 ± 0.000	0.867 ± 0.000	0.867 ± 0.000	**0.868** ± 0.000
pc1	0.935 ± 0.001	×	×	0.926 ± 0.001	0.924 ± 0.002	0.928 ± 0.002	0.935 ± 0.002	0.936 ± 0.001	**0.940** ± 0.001
pol	0.989 ± 0.000	×	×	—	—	—	**0.990** ± 0.000	0.989 ± 0.000	0.989 ± 0.000
pharyngitis	0.681 ± 0.001	**0.694** ± 0.000	0.689 ± 0.004	—	—	—	0.681 ± 0.002	0.682 ± 0.002	0.680 ± 0.001

Regression: RMSE ↓

Dataset	Base	AutoFeat	OpenFE	Direct	Direct-Batch	EvoFG
diamonds	507.877 ± 1.649	508.001 ± 0.598	507.798 ± 0.905	507.843 ± 0.530	504.095 ± 0.466	**503.807** ± 0.837
airfoil_self_noise	1.043 ± 0.017	1.054 ± 0.015	1.072 ± 0.011	1.037 ± 0.017	1.032 ± 0.011	**0.982** ± 0.005
forest-fires	53.809 ± 0.003	**53.728** ± 0.025	53.929 ± 0.007	53.795 ± 0.014	53.773 ± 0.009	53.833 ± 0.016
insurance	4484.977 ± 19.722	4477.361 ± 8.852	4472.311 ± 15.855	4471.705 ± 12.071	**4459.987** ± 7.720	4537.638 ± 8.108
housing	42153.210 ± 331.711	42231.484 ± 157.075	42109.365 ± 224.648	41280.844 ± 12.151	**40398.475** ± 79.554	40802.716 ± 93.817
Tesla_Stock	41.241 ± 1.280	—	—	41.521 ± 1.978	39.923 ± 1.990	**37.957** ± 0.031
bike	78.718 ± 0.808	**21.725** ± 0.399	80.076 ± 0.503	78.556 ± 1.056	80.863 ± 1.256	81.913 ± 0.964
communities	0.130 ± 0.000	0.130 ± 0.000	**0.130** ± 0.000	0.130 ± 0.000	0.130 ± 0.000	0.130 ± 0.000

E SUPPLEMENTARY ABLATION RESULTS

E.1 COMPONENT PROTOCOL AND COMPLETE RESULTS

Section 4.5 presents the component-level conclusions. Here we provide the executed controls, aggregation protocol, and complete numerical results. Frozen feature sets are evaluated with model seeds 42, 52, and 62 using the main LightGBM, XGBoost, and MLP settings, with common test samples across methods in each outer experiment. Each seed-level value averages five outer experiments. For each dataset, method, and predictor, the reported mean averages these three seed-level values, and the sample standard deviation is computed across them ($\mathrm{ddof} = 1$), following Appendix D.1. The analysis contains 576 seed-level values across eight datasets, eight methods, and three predictors. Model seeds are not independent LLM-search restarts. Tree predictions can be identical across seeds, giving measured rather than imputed zero deviations.

The primary comparison uses the existing search-quality rule: exclude an entire dataset if any variant has fewer than 50% successful replies or a failed downstream evaluation. Diamonds has 23/50 replies for independent proposals and is excluded from the seven-dataset aggregate. Its full scores and the all-eight sensitivity aggregate are retained below. The filter is based on execution quality, not predictive outcomes. Model–dataset cells are not counted as independent datasets, and no significance test is inferred from the three model seeds.

Table 14: Executed component controls. All frozen feature sets have completed downstream evaluation.

ID	Control	Actual intervention
A1	No explicit peak	Remove explicit peak field and alter archive; retain per-binding gains
A2	Peak only	Remove support axis and its feedback; retain only one binding in the prompt
A3	Single binding	One binding; peak archive; changed initialization binding scope and feedback
A4	Fixed library	First 50 structures from a fixed operator library
A5	Proxy only	Proxy-based screening; no actual candidate fine fits
A6	Independent proposals	New structures without evolutionary feedback

A1 does not remove all quality information. A2 and A3 share a peak archive but still differ in feedback. Single binding retains 2–20 features, while the other settings retain 20; capacity therefore also contributes to predictive differences. These comparisons identify effects of the executed settings, rather than perfectly isolated single-factor causal effects.

For classification, the effect is $100(\mu_{\mathrm{full}} - \mu_{\mathrm{control}})$ percentage points. For regression, it is $100(\mu_{\mathrm{control}} - \mu_{\mathrm{full}})/\mu_{\mathrm{control}}$ percent. Effects are averaged equally within each task type. Positive values favor full EvoFG.

Table 15: Sensitivity analysis including all eight datasets, adding diamonds to the regression average. Positive values favor full EvoFG. This complements the seven-dataset comparison in Table 4.

	Accuracy gain (pp)			RMSE reduction (%)		
Control	LGBM	XGB	MLP	LGBM	XGB	MLP
No explicit peak	**+0.39**	**+0.99**	-1.64	-1.66	-0.03	-1.43
Peak only	-0.01	**+0.93**	**+0.13**	-2.96	-1.19	-0.91
Single binding	-1.78	-1.20	-1.37	**+0.29**	-2.05	**+1.26**
Fixed library	-0.84	**+0.23**	-1.87	-1.81	-2.98	-0.94
Proxy only	**+1.22**	**+1.72**	-0.31	**+0.10**	-3.09	**+2.54**
Independent proposals	-1.23	-1.36	-0.92	-0.79	-0.43	**+1.71**

E.2 DETAILED FEATURE-CAP SENSITIVITY

The separate capacity experiment freezes search and training-gain ordering, varying only $K \in \{1, 5, 10, 20\}$. Eight representative datasets use five outer experiments, three predictors, and three

Table 16: LightGBM component results (mean ± sample standard deviation across three seed-level averages of five outer experiments). A1–A6 are defined in Table 14.

Dataset	Base	Full	A1	A2	A3	A4	A5	A6
balance-scale	0.840 ± 0.000	0.896 ± 0.000	0.896 ± 0.000	0.896 ± 0.000	0.968 ± 0.000	0.920 ± 0.000	0.880 ± 0.000	0.936 ± 0.000
vehicle	0.771 ± 0.000	0.806 ± 0.000	0.788 ± 0.000	0.812 ± 0.000	0.806 ± 0.000	0.818 ± 0.000	0.771 ± 0.000	0.824 ± 0.000
Academic	0.782 ± 0.000	0.782 ± 0.000	0.783 ± 0.000	0.777 ± 0.000	0.783 ± 0.000	0.781 ± 0.000	0.786 ± 0.000	0.775 ± 0.000
adult	0.867 ± 0.000	0.869 ± 0.000	0.870 ± 0.000	0.868 ± 0.000	0.867 ± 0.000	0.868 ± 0.000	0.867 ± 0.000	0.867 ± 0.000
airfoil_self_noise	1.676 ± 0.000	1.583 ± 0.000	1.494 ± 0.000	1.413 ± 0.000	1.474 ± 0.000	1.525 ± 0.000	1.607 ± 0.000	1.567 ± 0.000
cpu_small	2.712 ± 0.000	2.688 ± 0.000	2.699 ± 0.000	2.711 ± 0.000	2.686 ± 0.000	2.668 ± 0.000	2.664 ± 0.000	2.688 ± 0.000
housing	47133.371 ± 0.000	44236.369 ± 0.000	44488.155 ± 0.000	43534.220 ± 0.000	47487.472 ± 0.000	43472.863 ± 0.000	44164.515 ± 0.000	43441.098 ± 0.000
diamonds	544.161 ± 0.000	523.014 ± 0.000	514.653 ± 0.000	527.995 ± 0.000	532.387 ± 0.000	518.174 ± 0.000	522.986 ± 0.000	521.225 ± 0.000

Table 17: XGBoost component results (mean ± sample standard deviation across three seed-level averages of five outer experiments). A1–A6 are defined in Table 14.

Dataset	Base	Full	A1	A2	A3	A4	A5	A6
balance-scale	0.848 ± 0.000	0.912 ± 0.000	0.912 ± 0.000	0.904 ± 0.000	0.984 ± 0.000	0.912 ± 0.000	0.888 ± 0.000	0.944 ± 0.000
vehicle	0.794 ± 0.000	0.818 ± 0.000	0.776 ± 0.000	0.794 ± 0.000	0.800 ± 0.000	0.812 ± 0.000	0.776 ± 0.000	0.841 ± 0.000
Academic	0.779 ± 0.000	0.777 ± 0.000	0.782 ± 0.000	0.775 ± 0.000	0.776 ± 0.000	0.776 ± 0.000	0.776 ± 0.000	0.777 ± 0.000
adult	0.868 ± 0.000	0.871 ± 0.000	0.868 ± 0.000	0.868 ± 0.000	0.866 ± 0.000	0.869 ± 0.000	0.869 ± 0.000	0.870 ± 0.000
airfoil_self_noise	1.481 ± 0.000	1.633 ± 0.000	1.630 ± 0.000	1.578 ± 0.000	1.387 ± 0.000	1.491 ± 0.000	1.457 ± 0.000	1.643 ± 0.000
cpu_small	2.708 ± 0.000	2.717 ± 0.000	2.665 ± 0.000	2.662 ± 0.000	2.724 ± 0.000	2.679 ± 0.000	2.691 ± 0.000	2.670 ± 0.000
housing	46723.108 ± 0.000	43815.880 ± 0.000	45124.999 ± 0.000	43895.758 ± 0.000	47429.682 ± 0.000	43769.328 ± 0.000	43916.847 ± 0.000	43976.311 ± 0.000
diamonds	529.224 ± 0.000	521.306 ± 0.000	516.808 ± 0.000	524.566 ± 0.000	530.374 ± 0.000	516.813 ± 0.000	523.741 ± 0.000	516.495 ± 0.000

Table 18: MLP component results (mean ± sample standard deviation across three seed-level averages of five outer experiments). A1–A6 are defined in Table 14.

Dataset	Base	Full	A1	A2	A3	A4	A5	A6
balance-scale	0.973 ± 0.005	0.973 ± 0.005	0.995 ± 0.005	0.979 ± 0.005	0.992 ± 0.000	0.984 ± 0.008	0.968 ± 0.000	0.979 ± 0.005
vehicle	0.820 ± 0.012	0.790 ± 0.009	0.837 ± 0.007	0.792 ± 0.012	0.814 ± 0.007	0.837 ± 0.024	0.822 ± 0.009	0.825 ± 0.009
Academic	0.720 ± 0.006	0.720 ± 0.009	0.720 ± 0.006	0.716 ± 0.011	0.733 ± 0.006	0.738 ± 0.004	0.713 ± 0.016	0.713 ± 0.002
adult	0.829 ± 0.003	0.834 ± 0.004	0.831 ± 0.004	0.825 ± 0.003	0.834 ± 0.002	0.833 ± 0.007	0.827 ± 0.005	0.837 ± 0.003
airfoil_self_noise	1.528 ± 0.135	1.378 ± 0.028	1.343 ± 0.063	1.285 ± 0.071	1.368 ± 0.035	1.331 ± 0.034	1.567 ± 0.107	1.428 ± 0.049
cpu_small	2.980 ± 0.028	2.991 ± 0.072	3.025 ± 0.030	2.962 ± 0.020	3.155 ± 0.065	2.977 ± 0.057	3.083 ± 0.040	2.963 ± 0.065
housing	52591.786 ± 826.370	52728.019 ± 508.407	51747.514 ± 696.030	56443.997 ± 777.160	53973.877 ± 233.003	55074.158 ± 5482.053	51545.094 ± 920.616	56570.413 ± 5094.424
diamonds	559.333 ± 4.850	563.778 ± 10.989	550.833 ± 11.320	552.719 ± 12.125	553.866 ± 14.800	541.671 ± 6.731	549.440 ± 12.824	549.874 ± 20.757

internal model seeds (1,440 scores), with compatible $K = 20$ results reused. It includes cpu_small and remains separate from both the main 27-dataset table and the component comparisons above. Test performance does not reorder features. Section 4.5 summarizes the capacity comparisons; Table 19 retains their underlying scores.

Table 19: Predictive performance at different fixed feature caps.

Dataset	Model	$K = 1$	$K = 5$	$K = 10$	$K = 20$
Academic	lightgbm	0.770	0.777	0.775	0.771
Academic	mlp	0.710	0.716	0.708	0.719
Academic	xgboost	0.776	0.776	0.776	0.775
adult	lightgbm	0.871	0.871	0.872	0.872
adult	mlp	0.828	0.830	0.829	0.829
adult	xgboost	0.873	0.872	0.872	0.872
airfoil_self_noise	lightgbm	1.610	1.600	1.572	1.569
airfoil_self_noise	mlp	1.635	1.553	1.563	1.540
airfoil_self_noise	xgboost	1.582	1.597	1.578	1.586
balance-scale	lightgbm	0.893	0.922	0.918	0.934
balance-scale	mlp	0.973	0.979	0.985	0.983
balance-scale	xgboost	0.899	0.926	0.920	0.944
cpu_small	lightgbm	2.583	2.587	2.577	2.583
cpu_small	mlp	2.931	2.912	3.465	3.491
cpu_small	xgboost	2.717	2.717	2.875	2.853
diamonds	lightgbm	533.999	528.920	529.528	526.843
diamonds	mlp	564.685	581.569	572.233	569.238
diamonds	xgboost	525.181	521.540	522.041	521.715
housing	lightgbm	45958.131	45049.697	44709.078	44561.353
housing	mlp	52509.141	52494.295	52107.354	52707.013
housing	xgboost	45200.302	44524.102	44214.700	44232.201
vehicle	lightgbm	0.794	0.784	0.791	0.806
vehicle	mlp	0.826	0.829	0.814	0.811
vehicle	xgboost	0.778	0.792	0.793	0.798

F Supplementary Diagnostic Protocols

The following analyses distinguish binding mismatch, input recovery, screening omissions, selection bias, and search variability. Candidate yield is not treated as generalization evidence. Label-permutation and initialization experiments use a common run-quality rule: if a dataset has a failed task or any search with successful replies below 50% of requests, all methods and repeats for that dataset are excluded from that experiment. The threshold was selected after inspecting execution failures, not performance. Label permutation excludes adult, housing, and vehicle; initialization sensitivity excludes adult, airfoil_self_noise, and housing. Exclusion is local to the affected experiment.

F.1 Binding Mismatch at Fixed Structures

We fix five unary and five binary operators on each of eight representative datasets and compare one random binding, matched-quota random binding search, and programmatic binding search (Bergstra & Bengio, 2012). The first outer-training partition is used, with an independent 20% holdout for rechecking; temporal or group boundaries are respected where applicable. A rescue means that the single binding is nonpositive while search finds a positive instance. Figure 3 reports the resulting counts, so the dataset-level count table is omitted here. Holdout validation concerns the rescued instance, not every binding in its family.

F.2 Synthetic Tasks with Known Relevant Inputs

Each condition has 2,000 standard-normal samples and target $y = x_0^2 + \epsilon$ or $y = x_0 x_1 + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $\sigma \in \{0, 0.25\}$. Adding 0, 8, or 32 unrelated columns to two base inputs yields 12 conditions; the second base input is also irrelevant in unary tasks. Both methods share the proposal

ceiling, use an 80/20 split, and evaluate candidates with internal three-fold validation. RMSE is averaged across the three conventional predictors. These are separate synthetic evaluations, not five-unit real-data results. Both methods hit the true input set in every condition. Prediction differences

Table 20: Recovery of the true input set and downstream RMSE on synthetic tasks.

Arity	Distractors	Noise SD	EvoFG hit	Direct hit	EvoFG RMSE	Direct RMSE
1	0	0.0	Yes	Yes	0.1218	0.1536
1	0	0.25	Yes	Yes	0.3165	0.3281
1	8	0.0	Yes	Yes	0.1146	0.2063
1	8	0.25	Yes	Yes	0.3509	0.3210
1	32	0.0	Yes	Yes	0.3124	0.3539
1	32	0.25	Yes	Yes	0.4110	0.4607
2	0	0.0	Yes	Yes	0.0505	0.0438
2	0	0.25	Yes	Yes	0.2900	0.2896
2	8	0.0	Yes	Yes	0.0852	0.0893
2	8	0.25	Yes	Yes	0.3086	0.3123
2	32	0.0	Yes	Yes	0.1411	0.1510
2	32	0.25	Yes	Yes	0.3487	0.3243

depend on noise and distractor dimensionality. A hit checks input identities and does not establish recovery of the true operator or symbolic expression.

F.3 Exhaustive-Family Reference for Screening

To measure omissions, we fully evaluate fixed families whose binding count exceeds the actual shortlist quota but is at most 512. Comparisons use exact output deduplication. Recall is computed for the top 10% and top three outputs of the complete family, together with peak and support losses. Of 80 planned families, 25 meet the audit conditions; the others require no screening or exceed the exhaustive-evaluation cap. Figure 3 summarizes top-three recall. On vehicle, mean normalized peak and support losses are 0.02775 and 0.02777; both are zero on diamonds and housing. Ties affect top-three identity sets, so recall and numerical loss answer different questions. This compares the same empirical evaluator, not population generalization error.

F.4 Label-Permutation Negative Control

Permuting targets breaks the input–label relationship and tests whether adaptive search still produces positive internal gains. Original and permuted labels use the same input matrix, search program, and request ceiling. Five datasets remain after the run-quality rule. Each side has at most 50 requests; counts deduplicate fine-evaluated outputs. Positive candidates remain after permutation, and increase

Table 21: Original-label and permuted-label searches under the run-quality filter.

Dataset	Replies: orig./perm.	Positive: orig./perm.	Selected: orig./perm.
Academic	44/50; 45/50	305/1081	19/18
airfoil_self_noise	50/50; 45/50	154/37	17/19
balance-scale	50/50; 50/50	175/75	18/20
cpu_small	50/50; 50/50	867/369	20/19
diamonds	50/50; 44/50	767/568	20/20

on Academic. Repeated selection and evaluation variation can therefore yield positive internal signals without the original predictive relationship. Successful replies also differ between conditions. The experiment exposes a limitation of the search statistic and does not quantify an amount of selection bias eliminated by the method.

F.5 Sensitivity to Search Initialization

For five retained datasets and each outer unit, EvoFG and Direct are searched again with seeds 43 and 44, giving 100 searches. Code-and-binding identities are compared with the main search. Within each

new seed, results are first averaged over five outer units; the tables give means and sample standard deviations across the two new seeds. Predictive scores additionally average the three internal model seeds. Jaccard measures identity overlap and does not test output equivalence. EvoFG maintains

Table 22: Feature yield and identity overlap across two new search initializations. Values average outer units within each initialization before computing mean and sample standard deviation.

Dataset	Method	Positive outputs	Newly selected	Jaccard	Replies / 50
Academic	EvoFG	582.0 ± 56.9	19.6 ± 0.0	0.005	50–50
Academic	Direct	3.4 ± 0.6	3.4 ± 0.6	0.099	50–50
balance-scale	EvoFG	122.1 ± 2.7	18.9 ± 0.4	0.018	45–50
balance-scale	Direct	4.7 ± 0.4	4.7 ± 0.4	0.186	44–50
cpu_small	EvoFG	751.1 ± 25.0	20.0 ± 0.0	0.000	37–50
cpu_small	Direct	4.2 ± 1.1	4.2 ± 1.1	0.086	42–50
diamonds	EvoFG	792.5 ± 18.8	20.0 ± 0.0	0.000	44–50
diamonds	Direct	6.8 ± 1.4	6.8 ± 1.4	0.083	31–50
vehicle	EvoFG	683.6 ± 105.8	19.6 ± 0.3	0.005	30–50
vehicle	Direct	2.8 ± 0.8	2.8 ± 0.8	0.203	50–50

Table 23: Predictive performance across two new search initializations (mean ± sample standard deviation).

Dataset	Method	LightGBM	XGBoost	MLP
Academic	EvoFG	0.775 ± 0.004	0.774 ± 0.002	0.714 ± 0.005
Academic	Direct	0.772 ± 0.001	0.775 ± 0.002	0.709 ± 0.001
balance-scale	EvoFG	0.932 ± 0.006	0.931 ± 0.007	0.981 ± 0.002
balance-scale	Direct	0.879 ± 0.010	0.885 ± 0.002	0.970 ± 0.003
cpu_small	EvoFG	2.575 ± 0.004	2.699 ± 0.026	2.913 ± 0.015
cpu_small	Direct	2.622 ± 0.010	2.726 ± 0.005	2.987 ± 0.022
diamonds	EvoFG	523.456 ± 1.116	520.538 ± 0.940	575.820 ± 17.664
diamonds	Direct	540.300 ± 0.075	532.622 ± 0.721	593.173 ± 19.795
vehicle	EvoFG	0.793 ± 0.011	0.790 ± 0.006	0.818 ± 0.010
vehicle	Direct	0.769 ± 0.001	0.767 ± 0.006	0.831 ± 0.000

high candidate yield but shares few feature identities with the main search. Low identity overlap can reflect either different transformations or equivalent implementations, and cannot alone imply predictive instability. The predictive table must be considered separately. Two new initializations provide a limited descriptive estimate of variation.

F.6 Independent Recheck of Selected Features

An additional search/holdout split is made inside outer-training data. Search does not access holdout labels. Each selected feature is individually added to the raw representation and evaluated with LightGBM at the same capacity as the fine evaluator. Gains use each task's registered search metric and normalization; classification may therefore use AUC rather than the accuracy displayed in the main tables. Of 160 positive search candidates, 122 remain positive on independent recheck. Mean recheck gains become negative for balance-scale and vehicle. Together with label permutation, this shows that programmatic binding addresses mismatched inputs without automatically removing adaptive selection bias (Cawley & Talbot, 2010).

F.7 Rebinding after Structural Mutation

For a fixed child structure, compare its reoptimized binding with the parent's best binding:

$$G_{\text{rebind}} = \widehat{u}(o_{\text{child}}, c^*_{\text{child}}) - \widehat{u}(o_{\text{child}}, c^*_{\text{parent}}). \tag{24}$$

In the available trajectories, 304 of 477 positive peak improvements depend on rebinding. Thus, changing inputs after mutation can contribute to observed search improvement. This is a training-trajectory diagnostic, not an outer-test ablation of binding search.

Table 24: Independent recheck of individually selected features. Gains use each task's registered search metric.

Dataset	Features	Search positive	Holdout positive	Search gain	Holdout gain	Gap
Academic	20	20	16	0.0434	0.0172	0.0263
adult	20	20	15	0.0104	0.0077	0.0027
airfoil_self_noise	20	20	19	0.0923	0.0456	0.0467
balance-scale	20	20	4	0.0321	-0.0583	0.0904
cpu_small	20	20	20	0.0695	0.0253	0.0441
diamonds	20	20	20	0.0141	0.0178	-0.0037
housing	20	20	20	0.0135	0.0213	-0.0078
vehicle	20	20	8	0.0915	-0.0129	0.1045

G Dependence on Semantic Information

G.1 Anonymous Interaction Interface

Requests contain arity, anonymous feature count, task and metric, parent code, family quality, binding identifiers, and evaluation coverage. They do not require dataset or target names, original column names, category text, or raw samples. The following illustrative payload explains the fields rather than reproducing a measured search event:

```
{
  "arity": 2,
  "feature_count": 12,
  "task": "classification",
  "metric": "auc",
  "parent_code": "operator_arity2(X, selected_indices)",
  "metrics": {"peak": 0.10, "support": 0.08},
  "bindings": [{"ids": [3, 7], "gain": 0.02}],
  "coverage": {"requested": 64, "completed": 64}
}
```

Anonymous indices and gains are still data-dependent. This interface reduces semantic exposure and contextual requirements, without a formal privacy guarantee.

G.2 Renaming and Column-Permutation Checks

All eight representative datasets pass byte-identical prompt checks under semantic renaming. Complete families of fixed operators also pass column-permutation checks. These conclusions are restricted to the tested prompt constructor and complete fixed families. Finite evaluation quotas, stochastic exploration, and protected-binding order can still change realized search trajectories.

G.3 Anonymous Inputs for an External Generator

CAAFE is run with anonymous inputs on four classification datasets, five outer units each, retaining its native ten-round proposal setting. All 20 searches complete. Runs producing no new columns remain in the evaluation using the original representation. Prediction means and standard deviations follow Appendix D.1. Anonymous generation produces features on some datasets, whereas adult produces none. This characterizes generation behavior; it does not alone establish predictive benefit

Table 25: Anonymous CAAFE: feature generation and measured predictive results.

Dataset	New columns	No new / 5	LightGBM	XGBoost	MLP
Academic	1.4	0	0.772 ± 0.000	0.774 ± 0.000	0.708 ± 0.007
adult	0.0	5	0.871 ± 0.000	0.873 ± 0.000	0.828 ± 0.002
balance-scale	3.8	0	0.925 ± 0.000	0.931 ± 0.000	0.969 ± 0.002
vehicle	2.6	1	0.769 ± 0.000	0.765 ± 0.000	0.825 ± 0.001

or information leakage. The balance-scale comparison pairs semantic and anonymous CAAFE on

Table 26: Paired semantic and anonymous CAAFE on balance-scale. Differences use unrounded values.

Model	Semantic CAAFE	Anonymous CAAFE	Anonymous minus semantic
lightgbm	0.974 ± 0.000	0.925 ± 0.000	-0.050
xgboost	0.976 ± 0.000	0.931 ± 0.000	-0.045
mlp	0.981 ± 0.003	0.969 ± 0.002	-0.012

the same current outer units and predictors. Anonymous accuracy is lower for each model. Semantic column information can thus be useful in this setting. The other three anonymous datasets lack same-protocol semantic controls, so the comparison is neither a four-dataset paired test nor evidence of pretraining contamination.

H Discovery Efficiency and Computational Cost

H.1 Candidate and Selection Counts

The counting unit is a unique training output that has completed actual fine evaluation. Different bindings with identical outputs count once. Shared initialization outputs are excluded; a candidate is positive when its unclipped signed gain exceeds zero. Newly selected features are the frozen output set minus initialization outputs, rather than the total number of final columns. Failed requests contribute token usage when recorded; missing usage stays unknown.

The independent efficiency pilot comprises 15 datasets, three search repeats per method, an 80/20 split, internal three-fold evaluation, and 50 request opportunities per repeat. Its 90 searches use shared initialization. Counts are pooled within dataset before computing method ratios, then ratios are averaged equally across datasets. This measures proposal utilization and is separate from the main five-unit predictive evaluation.

H.2 Per-Dataset Yield and Token Records

Table 27: Per-dataset efficiency after pooling three search repeats. E: EvoFG; D: Direct.

Dataset	Positive: E/D	Selected: E/D	Yield ratio	Selection ratio	Token saving
Academic	2125/35	55/20	60.71	2.75	60.07%
Disease	1915/45	60/25	42.56	2.40	51.15%
adult	1709/37	58/17	46.19	3.41	65.59%
airfoil_self_noise	522/27	59/27	19.33	2.19	41.82%
balance-scale	221/50	60/42	4.42	1.43	11.88%
car	440/38	59/28	11.58	2.11	42.43%
cpu_small	2442/47	60/15	51.96	4.00	69.72%
diabetes	2921/53	57/29	55.11	1.97	35.53%
forest-fires	1653/38	56/23	43.50	2.43	48.26%
heart	1375/28	58/19	49.11	3.05	61.70%
housing	1774/44	60/30	40.32	2.00	39.07%
insurance	909/46	58/40	19.76	1.45	16.22%
kddCup09	2835/44	56/16	64.43	3.50	69.39%
pc1	2626/51	58/25	51.49	2.32	47.90%
vehicle	1536/33	60/15	46.55	4.00	70.18%

Dataset-equal averaging differs from pooling all candidates into one ratio: $25003/616 \approx 40.59$, whereas the mean of dataset ratios is approximately 40.47. Similarly, selected-feature yield is computed from within-dataset ratios, rather than $874/371$. Token efficiency includes only recorded usage and excludes no run simply because its feature yield is low. Incomplete token logs affect 14 searches across ten datasets. The identifiers are search seeds in the independent pilot, not outer-fold indices in the main benchmark. Missing token consumption is not estimated or replaced by zero.

Table 28: Searches with incomplete token-usage records.

Dataset	Search seed	Method
balance-scale	42	Direct
balance-scale	62	Direct
heart	52	EvoFG
forest-fires	52	EvoFG
forest-fires	62	Direct
airfoil_self_noise	42	Direct
airfoil_self_noise	62	EvoFG
vehicle	52	EvoFG
Academic	52	EvoFG
kddCup09	62	EvoFG
Disease	42	EvoFG
Disease	62	EvoFG
insurance	62	EvoFG
cpu_small	52	EvoFG

Consequently, the reported saving describes recorded tokens per selected feature, not a complete monetary or compute budget.

H.3 Search Wall-Clock and Downstream Evaluation Costs

Search wall-clock time is computed from the start and end events of each outer search, including LLM generation/waiting, binding optimization, and search-stage model evaluation. EvoFG, Direct, and Direct-Batch share the 50-proposal ceiling, but trigger different amounts of evaluator work. The shared scheduler introduces resource contention. This appendix retains the full 52-dataset timing study, which is broader than the selected 27-dataset predictive benchmark.

Table 29: Observed search wall-clock time under shared scheduling, for the full 52-dataset timing archive.

Method	Datasets	Search units	Median (min)	IQR (min)
EvoFG	52	255	28.32	10.43–67.05
Direct	52	260	1.75	1.47–2.35
Direct-Batch	52	260	6.37	4.97–9.51

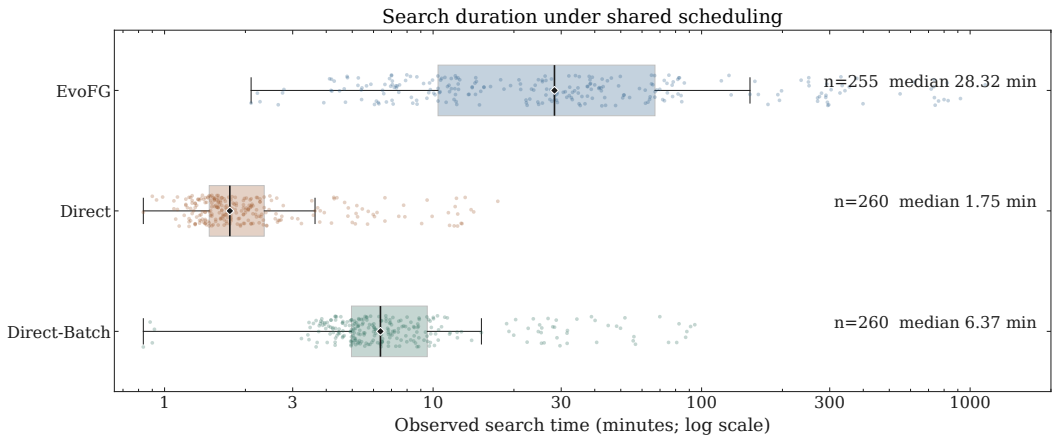


Figure 4: Observed search duration under shared scheduling. Points are individual search units; boxes show the interquartile interval and black lines/diamonds mark medians. Time is shown on a logarithmic axis.

Table 30: Per-dataset median search time in minutes for the full timing archive.

Dataset	EvoFG	Direct	Direct-Batch	Units E/D/B
Academic	82.70	2.25	8.85	5/5/5
adult	61.80	1.97	9.63	5/5/5
airfoil_self_noise	4.17	1.20	3.48	5/5/5
airlines	77.85	0.93	0.90	4/5/5
balance-scale	4.48	2.72	4.93	5/5/5
bank-marketing	72.73	2.10	9.80	5/5/5
bike	18.37	1.42	4.53	5/5/5
blood-transfusion	2.27	1.30	4.97	5/5/5
breast-w	7.70	1.45	5.13	4/5/5
car	14.32	1.68	8.50	5/5/5
cdc diabetes	736.13	11.98	48.25	5/5/5
cmc	24.73	1.85	6.77	5/5/5
communities	192.28	2.28	10.48	5/5/5
compass	41.42	1.75	7.03	5/5/5
covtype	732.63	12.82	82.40	3/5/5
cpu_small	24.55	1.67	6.47	5/5/5
crab	13.98	1.35	4.95	5/5/5
credit	35.32	1.65	7.92	5/5/5
diabetes	11.18	1.48	5.27	5/5/5
diamonds	67.12	2.35	10.22	5/5/5
Disease	7.45	1.30	4.17	5/5/5
electricity	25.48	2.02	6.35	5/5/5
Enefit	294.96	7.45	44.88	4/5/5
eucalyptus	37.68	1.65	6.82	5/5/5
eyemoves	43.88	1.97	6.47	5/5/5
forest-fires	7.10	1.27	3.87	5/5/5
heart	12.18	1.45	5.30	5/5/5
Higgs	762.83	12.35	75.32	5/5/5
house_16H	51.47	1.97	9.50	5/5/5
housing	32.03	2.05	7.33	5/5/5
insurance	5.65	1.30	3.77	5/5/5
jannis	270.38	5.70	23.62	5/5/5
jungle_chess	75.55	3.78	19.97	5/5/5
kdd_ipums_la	34.97	2.20	6.80	5/5/5
kddCup09	71.63	1.70	7.17	5/5/5
kidney-stone	5.42	1.50	4.47	5/5/5
MagicTelescope	27.60	2.08	6.00	5/5/5
MiniBooNE	354.80	5.00	28.75	5/5/5
myocardial	290.50	2.63	11.05	5/5/5
NHANES	6.55	1.73	4.33	5/5/5
pc1	24.00	1.70	5.75	5/5/5
pharyngitis	10.43	1.53	4.72	5/5/5
phoneme	7.27	1.58	5.58	5/5/5
plasma_retinol	7.17	1.43	4.33	5/5/5
pol	42.62	1.53	5.48	5/5/5
rl	22.25	1.53	5.13	5/5/5
road-safety	270.42	5.17	27.90	5/5/5
spaceship-titanic	32.73	1.82	6.93	5/5/5
Tesla_Stock	13.62	1.25	4.67	5/5/5
tic-tac-toe	11.68	1.33	4.92	5/5/5
vehicle	39.18	1.63	6.90	5/5/5
wine	10.43	1.50	4.97	5/5/5

Stage elapsed time is distinct from summing parallel task durations. EvoFG spends more conventional computation on each structural proposal and has longer observed search time despite greater feature yield. These records support a trade-off between LLM opportunities and program evaluation; they do not establish end-to-end acceleration under equal exclusive resources.

Table 31: Elapsed time of scheduled evaluation stages; these are not sums of parallel task durations.

Stage	Wall time (s)	Hours
Downstream model evaluation	42349.50	11.764
Initial mechanism queue tail	281.61	0.078
Mechanism retry stage	11210.21	3.114

H.4 Interpreting Computational Cost

Mixed CPU/GPU scheduling, concurrency, and model loading affect elapsed time. TabPFN preprocessing and memory limits also contribute to evaluation cost. Cumulative fitting logs cannot be summed repeatedly at each event. We therefore report discovery yield under a fixed LLM request budget, recorded token cost per selected feature, and observed wall-clock time separately. No one quantity is used as a substitute for total computational efficiency.