Be Greedy, Stay Linear: Universally Robust Feature Engineering

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2026

Abstract

Automated feature engineering (FE) for tabular data often assumes that complex downstream models require equally complex selection proxies. We expose this as a fundamental fallacy: the *Selector Overfitting Paradox*. As dimensionality scales, high-capacity proxies overfit to the expanding combinatorial noise, generating bloated representations that actively degrade densely tuned, state-of-the-art predictors. To resolve this, we introduce *LinFE*, a greedy symbolic feature-construction method that enforces a strict linear bottleneck. By utilizing an Ordinary Least Squares (OLS) readout, feature generation transitions from an approximate heuristic into an exact geometric matching pursuit, where the optimal candidate is strictly the one maximizing its orthogonal projection onto the current residual. We further unify the search space by statically mapping categorical variables to continuous residual-mean terminals. Across a comprehensive OpenML benchmark, competing FE methods inject variance and fail to outperform optimized raw baselines. In stark contrast, *LinFE* extracts sparse, structural representations that consistently improve densely tuned LightGBM, CatBoost, and Neural Networks. Our results demonstrate a counterintuitive truth governing high-dimensional learning dynamics: restricting selection capacity liberates representation power, immunizing the search against the curse of dimensionality and distilling only universally robust features.

1. Introduction

Automated feature engineering (FE) suffers from a pervasive evaluation blind spot: it is rarely tested against heavily optimized, state-of-the-art (SOTA) predictors. While existing FE methods artificially boost untuned models by hard-coding basic interactions, they actively degrade densely tuned Gradient-Boosted Decision Trees (GBDTs) and Neural Networks (NNs). By flooding the representation with unaligned redundancy and variance, current methods often perform worse than simply using the raw dataset. The field requires a stricter standard: a feature constructor must universally survive contact with, and strictly improve, highly expressive downstream learners.

Existing greedy FE methods fail this standard because they conflate the model used to *select* features with the model used to *consume* them. By using complex trees as selection proxies, calculating a candidate feature's true loss decrease becomes intractable, forcing reliance on noisy gradient heuristics. A non-convex predictor is a powerful final model, but a flawed, noisy instrument for deciding which symbolic feature to add next.

To untangle these roles, we introduce *LinFE* with an intentional asymmetry: pairing a simple ordinary least-squares (OLS) selector with a complex downstream predictor. Under OLS, the greedy objective becomes mathematically exact. A candidate's reduction in squared error is precisely its orthogonal projection onto the current residual ($\Delta(g) = \langle r, \tilde{g} \rangle^2 \,/\, \|\tilde{g}\|_2^2$), enforcing strictly zero gain for redundant features. This shifts FE from an *approximate* matching pursuit (MP) for *complex*

models into an *exact* MP for *linear* models. Furthermore, we unify all covariates under this shared projection criterion by statically mapping categorical groups to scalar residual atoms, preventing combinatorial explosion.

We empirically answer the critical question: *Does FE still help when the downstream model is densely optimized?* Across 20 OpenML regression benchmarks, we evaluate representations using both fixed-configuration and Optuna-tuned learners. While established baselines succeed on fixed models, they frequently fail to outperform raw features once learners are densely tuned. In stark contrast, *LinFE* consistently improves predictive performance across all optimized model families (NNs, CatBoost, LightGBM, RF). Achieving state-of-the-art aggregate performance, *LinFE* demonstrates that a simple, exact linear selector engineers universally robust features capable of elevating complex SOTA predictors. Our main contributions are listed in Appendix A. Related work about FE can be found in Appendix B.

2. Theory: The Selector Overfitting Paradox

Current benchmark automated FE frameworks assume proxy-alignment (e.g., OpenFE): that a complex downstream model requires an equally complex proxy to select its features. We argue this is a fundamental fallacy. In massive symbolic spaces, high-capacity proxies may not find better signals; they find better noise. We formalize this as the Selector Overfitting Paradox: (1). **Complexity as a Noise Trap:** A high-capacity proxy (e.g., GBDT) acts as a noise-detector. It easily assigns high gain to "lucky" features that happen to partition the specific training sample S_n. This results in features that are statistically lucky but globally irrelevant. (2). **Linearity as a Mechanistic Filter:** A linear proxy is intentionally inflexible. It cannot "save" a weak feature through complex splits. To be selected, a feature must exhibit a consistent, global correlation with the residual. (3). **The Length Paradox:** While a linear proxy forces the generator to construct longer symbolic expressions to linearize relationships, these features are mechanistically robust. Because linear correlation is a sufficient statistic for most learners, these features are "universal." To quantify this, we bound the Universality Regret—the gap between our selected feature and the downstream optimal:

Theorem 1 (The Selector Overfitting Paradox) *Let $\mathcal{G}$ be a symbolic feature space and $\mathcal{D}$ a distribution over downstream model classes $\mathcal{H}_D$. For any $g \in \mathcal{G}$ and model class $\mathcal{H}$, let the true gain be $\Delta(g, \mathcal{H}) = \min_{h\in\mathcal{H}} \mathbb{E}_P[\ell(Y, h(B))] - \min_{h\in\mathcal{H}} \mathbb{E}_P[\ell(Y, h(B \cup \{g\}))]$, and let $g_D^* = \arg\max_{g\in\mathcal{G}} \Delta(g, \mathcal{H}_D)$.*

Let S_n be a dataset of n samples and $\hat{g} = \arg\max_{g\in\mathcal{G}} \widehat{\Delta}(g, \mathcal{H}_P, S_n)$ be the feature selected by proxy class $\mathcal{H}_P$ using empirical gain. For a C-bounded loss ℓ, with probability at least $1-\delta$ over the choice of S_n, the Universality Regret $\mathcal{R}_{univ}(\mathcal{H}_P) = \mathbb{E}_{\mathcal{H}_D\sim\mathcal{D}}[\Delta(g_D^*, \mathcal{H}_D) - \Delta(\hat{g}, \mathcal{H}_D)]$ is bounded by:

$$\mathcal{R}_{univ}(\mathcal{H}_P) \le 4\mathfrak{R}_n(\ell \circ \mathcal{H}_P \circ \mathcal{G}) + \mathbb{E}_{\mathcal{H}_D\sim\mathcal{D}}\left[dist_{\mathcal{G}}(\mathcal{H}_P, \mathcal{H}_D)\right] + 2C\sqrt{\frac{\log(1/\delta)}{2n}} \tag{1}$$

where $dist_{\mathcal{G}}(\mathcal{H}_P, \mathcal{H}_D) = 2\sup_{g\in\mathcal{G}} |\Delta(g, \mathcal{H}_P) - \Delta(g, \mathcal{H}_D)|$ and $\mathfrak{R}_n$ is the Rademacher complexity of the composed proxy-generator function class.

Proof is shown in Appendix F.

Remark 2 ***Complex Proxies vs. Exact Linearity:*** *GBDT proxies (e.g., OpenFE) "shatter" the massive search space $\mathcal{G}$, overfitting to local noise through idiosyncratic splits. This yields bloated*

feature sets that provide marginal utility to regularized models. Conversely, LinFE rejects lucky partitions, extracting a sparse, concentrated representation. ***The Greedy Residual Imperative:*** *Global linear optimization (e.g., FEAT) is insufficient. LinFE enforces an exact greedy matching pursuit, where the gain is strictly orthogonal projection:* $\Delta(g) = \langle r, \tilde{g} \rangle^2 / \|\tilde{g}\|_2^2$. *This ensures features do not just possess linear predictive power, but specifically target the deficiencies of the current coordinate system.* ***Complexity Filtering and Universality:*** *By minimizing proxy capacity* $\mathfrak{R}_n(\mathcal{H}_P)$*, LinFE operates as a strict complexity filter. The features that survive this bottleneck isolate fundamental structural signals, proving universally robust enough to elevate even the most densely tuned state-of-the-art predictors.*

Remark 3 ***Scaling Dynamics of Proxy Complexity*** $\mathfrak{R}_n$**.** *The bound in Theorem 1 reveals why high-capacity proxies fail as the ambient dimension* d *grows. In frameworks like OpenFE, the proxy class* $\mathcal{H}_P$ *consists of GBDTs which "score" a candidate feature* g *by finding optimal splits across the entire feature set* $B \cup \{g\}$*. This causes the Rademacher complexity* $\mathfrak{R}_n(\ell \circ \mathcal{H}_P \circ \mathcal{G})$ *to scale with the joint complexity of the tree search and the symbolic space, making it highly susceptible to "shattering" spurious correlations in high dimensions (*$d \gg n$*).In contrast, LinFE effectively collapses the complexity of the proxy class. By evaluating candidates via strict orthogonalization (*$\tilde{z}_g = (I - P_t)z_g$*), the hypothesis space* $\mathcal{H}_P$ *used to calculate gain is restricted to a simple 1-dimensional projection. Since the Rademacher complexity of linear classes scales much more favorably; often* $O(\sqrt{\log(d)/n})$*;LinFE keeps the first term of the Universality Regret bound small. This allows it to bypass the "Noise Trap" and maintain stability where complex proxies succumb to the aggressive error climb observed in Fig. 1.*

3. *LinFE*: Methods

To circumvent the *Selector Overfitting Paradox*, *LinFE* enforces a strict structural decoupling between feature construction and downstream inference. Operating exclusively as a mechanistic filter, it employs an exact Ordinary Least Squares (OLS) readout to iteratively evaluate and select features based solely on their orthogonal, non-redundant contribution. The complete procedural flow is summarized below, with full derivations, algorithmic details (Algorithms 1–3), and other supplementary materials in Appendix C.

***LinFE* Procedural Summary:**

1. **Categorical Initialization:** To unify the feature space, map each discrete categorical level to a continuous static scalar representing the smoothed mean of the initial continuous residual r_0 (Appendix C.6).
2. **Symbolic Search:** At stage t, generate a population of candidate expressions g using an evolutionary search over a safe, shallow mathematical grammar (Appendix C.5).
3. **Exact Orthogonalization:** For each candidate vector $z_g = g(X)$, isolate its novel contribution by strictly removing the component already spanned by the current representation matrix P_t:$\tilde{z}_g = (I - P_t)z_g$. To bypass expensive matrix inversions, this is computed via an incrementally updated orthonormal basis $\mathcal{Q}_t$, restricting evaluation to $\mathcal{O}(n \cdot m_t)$ time (Appendix C.3).
4. **Residual-Gain Scoring:** Evaluate candidates using the exact finite-sample reduction in squared risk against the current target residual r_t: $\widehat{\Delta}_t(g) = \frac{1}{n} \frac{(r_t^\top \tilde{z}_g)^2}{\tilde{z}_g^\top \tilde{z}_g + \varepsilon}$, where $\varepsilon \to 0^+$ guarantees zero gain for redundant features.

Table 1: **Performance on densely tuned (Optuna) downstream learners across 20 OpenML datasets.** Raw (−) denotes the original dataset without any FE. ↓ indicates that lower values are better. Rank represents the method's average relative standing across all datasets. Norm. stands for min–max normalized RMSE. All values are dataset-level averages over seeds with 95% confidence intervals. Best results are **bolded**, second-best are underlined.

Method	CatBoost			LightGBM			Random Forest			NNs		
Metric	Rank ↓	RMSE ↓	Norm. ↓	Rank ↓	RMSE ↓	Norm. ↓	Rank ↓	RMSE ↓	Norm. ↓	Rank ↓	RMSE ↓	Norm. ↓
Raw (−)	2.750 ± 0.485	4.462 ± 4.453	0.154 ± 0.092	2.850 ± 0.427	4.612 ± 4.602	0.206 ± 0.104	3.400 ± 0.393	5.023 ± 5.122	0.263 ± 0.126	3.150 ± 0.518	4.909 ± 4.905	0.247 ± 0.113
FEAT	5.700 ± 0.405	8.122 ± 9.972	0.939 ± 0.089	5.450 ± 0.674	8.173 ± 9.898	0.868 ± 0.145	5.500 ± 0.612	8.289 ± 9.964	0.883 ± 0.131	5.200 ± 0.645	7.930 ± 9.269	0.826 ± 0.136
AutoFeat	2.800 ± 0.520	4.467 ± 4.473	**0.141 ± 0.092**	3.200 ± 0.435	4.632 ± 4.607	0.255 ± 0.112	3.500 ± 0.421	5.003 ± 5.072	0.267 ± 0.121	3.150 ± 0.591	4.837 ± 4.812	0.253 ± 0.138
MI-obj	4.350 ± 0.686	5.733 ± 6.223	0.485 ± 0.148	4.150 ± 0.686	5.749 ± 6.161	0.453 ± 0.138	3.800 ± 0.774	5.847 ± 6.303	0.391 ± 0.151	4.050 ± 0.732	5.925 ± 6.414	0.474 ± 0.158
OpenFE	2.950 ± 0.745	5.212 ± 5.676	0.228 ± 0.124	3.000 ± 0.752	5.496 ± 6.115	0.273 ± 0.137	2.900 ± 0.737	5.788 ± 6.502	0.250 ± 0.132	3.550 ± 0.628	5.877 ± 6.489	0.379 ± 0.145
LinFE	**2.450 ± 0.438**	**4.449 ± 4.524**	0.143 ± 0.103	**2.350 ± 0.608**	**4.581 ± 4.630**	**0.198 ± 0.130**	**1.900 ± 0.491**	**4.943 ± 5.119**	**0.127 ± 0.104**	**1.900 ± 0.602**	**4.667 ± 4.690**	**0.139 ± 0.114**

5. **Update & Export:** Permanently add the highest-gain feature to the representation, update r_t, and repeat from Step 2. When the affine OLS validation RMSE stops improving, halt construction and export the frozen representation to the expressive downstream learner.

4. Experiments

Detailed experiment setups can be found in Appendix D (e.g., D.2). Below we share the core findings under the optimized downstream learners. More experiment results (e.g, performance comparison under fixed downstream learners without optimization, significance analysis) are shown in Appendix E.

4.1. Optimized Downstream Learners

Table 1 presents our central evaluation: testing representations under densely tuned downstream learners via *Optuna*. This rigorous regime inherently challenges automated FE, as optimized GBDTs and NNs can typically express or regularize away simple transformations independently.

Consequently, competitor methods (AutoFeat, FEAT, MI-obj, OpenFE) frequently degrade performance (<40% win rate; Figure 2b) by injecting capacity-burdening redundancy. In stark contrast, *LinFE* outperforms all baselines and is the only method that consistently beats the densely tuned raw data. By strictly isolating non-redundant features, *LinFE* achieves a dominant 60%–85% win rate across all downstream families, delivering substantial mean improvements of +6.7% for RF and +7.0% for NNs (Figure 2a). These gains over thoroughly optimized baselines prove that *LinFE* does not merely rescue weak models; it structurally reshapes the coordinate system to elevate even the highest-capacity learners.

4.2. Empirical Results on the Selector Overfitting Paradox

Based on the aggregate performance established in Table 1, we selected OpenFE (the strongest performing SOTA baseline) for a direct comparison with *LinFE* to empirically validate our theory.

Table 2 empirically validates the *Selector Overfitting Paradox*. OpenFE's flexible GBDT proxy (massive $\mathfrak{R}_n$) overfits to local noise, generating a bloated set of shallow features (avg. 85.23) that merely inject variance into tuned models. Conversely, *LinFE*'s strict linear bottleneck extracts a sparse (avg. 10.62) yet mathematically dense (high ASL) set of global structural expressions. This

Table 2: **Comparison of FE methods by ASL and the number of generated features.**

Method	Generated		ASL	
	Avg.	*Med.*	*Avg.*	*Med.*
LinFE	10.62	8.00	10.13	10.33
OpenFE	85.23	50.00	2.83	2.94

reveals a counterintuitive truth: restricting selection capacity does not limit representation power—it acts as a forcing function, distilling only the universally robust features capable of elevating SOTA predictors.

To investigate high-dimensional learning dynamics, Figure 1 visualizes feature-construction efficacy across varying input dimensionalities (using min–max normalized RMSE to isolate relative, scale-independent performance).

These dynamics empirically validate the *Selector Overfitting Paradox* and the mechanism of orthogonal decoupling (discussed in Remark 3). As raw dimensions increase, the combinatorial space of spurious correlations explodes. OpenFE's flexible proxy overfits to this expanding noise floor, causing its error to climb aggressively. Conversely, *LinFE*'s performance remains flat and dominant. By imposing an exact linear bottleneck that reduces selection capacity to $\mathcal{O}(1)$ per candidate, *LinFE* immunizes the search against the curse of dimensionality, extracting only universally robust structural signals. This proves that a strict geometric filter is computationally and theoretically essential for scalable, high-dimensional FE.

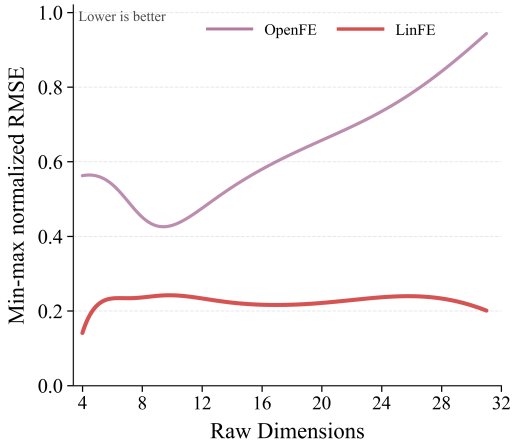


Figure 1: **Dimensional scaling.** Smoothed min–max normalized RMSE across optimized learners.

5. Discussion and Conclusions

The success of *LinFE* resolves the *Selector Overfitting Paradox*, a fundamental tension at the heart of automated FE. For highly tuned expressive models like GBDTs and NNs, the true bottleneck is not a lack of nonlinear candidates, but the injection of proxy-induced noise, particularly as the combinatorial space explodes in higher dimensions. By decoupling representation construction from downstream inference, *LinFE* proves that a strict linear bottleneck is not a limitation, but a vital mechanistic filter. Crucially, this orthogonal decoupling effectively lowers the error floor in scenarios where traditional methods are compromised by selection overfitting. While currently limited to regression, *LinFE* establishes a counterintuitive paradigm for modern tabular learning: *strict orthogonal decoupling liberates representation power.* By remaining greedy and strictly linear during feature construction, we immunize the representation against the curse of dimensionality, allowing SOTA downstream predictors to allocate their full capacity toward learning the true underlying relation, rather than untangling an inefficient coordinate space.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[2] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *The journal of machine learning research*, 13:27–66, 2012.

[5] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34: 18932–18943, 2021.

[6] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004, 2022.

[7] Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Tabular deep learning meets nearest neighbors in 2023. *arXiv preprint arXiv:2307.14338*, 2023.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Franziska Horn, Robert Pack, and Michael Rieger. The autofeat python library for automated feature engineering and selection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 111–120. Springer, 2019.

[10] 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. *Advances in neural information processing systems*, 30, 2017.

[11] William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason H Moore. Learning concise representations for regression by evolving networks of trees. *arXiv preprint arXiv:1807.00981*, 2018.

[12] Yuanfei Luo, Mengshuo Wang, Hao Zhou, Quanming Yao, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. Autocross: Automatic feature crossing for tabular data in real-world applications. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1936–1945, 2019.

[13] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[14] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36:76336–76369, 2023.

[15] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993.

[16] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[17] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[18] Binh Tran, Bing Xue, and Mengjie Zhang. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing*, 8(1):3–15, 2016.

[19] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[20] Hangting Ye, Wei Fan, Xiaozhuang Song, Shun Zheng, He Zhao, Dandan Guo, and Yi Chang. Ptarl: Prototype-based tabular representation learning via space calibration. *arXiv preprint arXiv:2407.05364*, 2024.

[21] Jianan Ye, Zhaorui Tan, Yijie Hu, Xi Yang, Guangliang Cheng, and Kaizhu Huang. Disentangling tabular data towards better one-class anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 13061–13068, 2025.

[22] Tianping Zhang, Zheyu Aqa Zhang, Zhiyuan Fan, Haoyan Luo, Fengyuan Liu, Qian Liu, Wei Cao, and Li Jian. Openfe: Automated feature generation with expert-level performance. In *International Conference on Machine Learning*, pages 41880–41901. PMLR, 2023.

Appendix A. Contributions

Our contributions are:

1. We formulate greedy automated feature construction as an exact residual-gain MP for OLS.
2. We introduce *LinFE*, which leverages exact orthogonal residualization and a linear validation criterion for early stopping.
3. We unify continuous and categorical variables under a shared framework by mapping categorical groups to fixed residual-mean symbolic terminals prior to the search.
4. We demonstrate empirically that exact linear selection engineers universally useful features that survive contact with densely tuned SOTA downstream predictors.

Appendix B. Related Work

Automated FE attempts to navigate the intractable search space of feature transformations. Existing methods generally adopt one of four structural paradigms, each facing distinct challenges when paired with highly expressive downstream learners.

Filter-based Selection. Filter methods evaluate candidates through pure statistical association with the target, aiming to balance relevance against redundancy without directly querying a downstream model's loss [2, 4, 16]. While computationally efficient, mutual information (MI) and statistical filters rely on proxy metrics that often fail to align with the actual error surface of the final predictor, leaving complex conditional redundancies unaddressed.

Global Expansion and Sparse Selection. These methods construct a massive predefined pool of nonlinear transformations and subsequently apply strong regularization (e.g., L_1 penalties) to extract a sparse subset. AutoFeat [9] and AutoCross [12] effectively use an interpretable linear bottleneck to filter this globally expanded space. However, generating the global pool scales poorly with feature dimensionality, and simultaneous sparse selection does not dynamically account for the changing residual as features are added one by one.

Evolutionary Symbolic Search. To avoid static global pools, evolutionary methods dynamically navigate the open-ended space of symbolic expressions. By evolving compact networks of mathematical operators, methods like FEAT [11] and [18] represent the SOTA in finding nonlinear representations with explicit accuracy-complexity trade-offs. Despite their flexibility, these methods typically optimize global fitness metrics over the entire feature set concurrently, making it difficult to guarantee that newly mutated features strictly target the unexplained variance of the existing set.

Greedy Residual Approximation. Taking a step toward MP, greedy methods iteratively ask how much a candidate improves the predictions of the *current* representation. OpenFE [22] represents the forefront of this category. Its FeatureBoost procedure estimates incremental loss reduction by fitting a residual model on top of the current feature set. However, because OpenFE tightly couples the selection logic to a complex GBDT, it is forced to use pruning and approximate attribution (e.g., gradient heuristics) to make the search tractable. LinFE diverges fundamentally from this approach. By replacing the complex tree-based selector with a linear geometric projection, LinFE completely avoids gradient approximations, granting every candidate an exact, closed-form local utility score and mathematically enforcing zero gain for redundant transformations.

Appendix C. Extended Methodology and Theoretical Details

C.1. Problem Formulation

Let $(X, Y) \sim P$, with $Y \in \mathbb{R}$. A representation is a finite tuple of real-valued features

$$F = (f_1, \ldots, f_m), \qquad f_j : \mathcal{X} \to \mathbb{R}.$$

For a readout family $\mathcal{H} = \{\mathcal{H}_m\}_{m \geq 0}$, where $\mathcal{H}_m$ maps $\mathbb{R}^m$ to $\mathbb{R}$, define the predictors available under representation F as

$$\mathcal{F}_{\mathcal{H}}(F) = \{x \mapsto h(f_1(x), \ldots, f_m(x)) : h \in \mathcal{H}_m\}.$$

For loss ℓ, the best achievable risk under F is

$$\mathcal{R}^{\star}_{\mathcal{H},\ell}(F) = \inf_{\hat{y} \in \mathcal{F}_{\mathcal{H}}(F)} \mathbb{E}[\ell(Y, \hat{y}(X))].$$

Definition 4 (Model-relative construction gain) *Given a representation F, a candidate feature $g : \mathcal{X} \to \mathbb{R}$, a readout family $\mathcal{H}$, and a loss ℓ, the construction gain of adding g is*

$$\Delta_{\mathcal{H},\ell}(F, g) = \mathcal{R}^{\star}_{\mathcal{H},\ell}(F) - \mathcal{R}^{\star}_{\mathcal{H},\ell}(F \cup \{g\}). \tag{S.1}$$

We call Eq. (S.1) model-relative because the usefulness of g is measured relative to the readout family used for construction, rather than by marginal association with the target. LinFE uses the affine linear instance of this gain. We focus on regression because squared loss with an affine linear readout gives a closed-form score; classification would require a different residual object, such as a curvature-weighted local approximation, and is outside the scope of this paper.

C.2. Residual-Gain Identities

The identities in this section are standard consequences of least-squares projection geometry, and are closely related to the classical matching-pursuit and orthogonal matching-pursuit literature [13, 15]. We include them only to make the notation and scoring rule in Section C.3 self-contained. The contribution of LinFE is not these projection identities themselves, but their use as a deliberately simple and exact selection rule for symbolic feature construction before training a separate downstream learner.

Population identity. Let $\ell(y, \hat{y}) = (y - \hat{y})^2$, and let $\mathcal{H}_{\text{lin}}$ be the readout family whose m-dimensional member contains all affine maps from $\mathbb{R}^m$ to $\mathbb{R}$. Assume $Y \in L_2(P)$ and all selected and candidate features are square-integrable.

For a representation F, let $V(F)$ be the closed linear subspace of $L_2(P)$ spanned by the constant random variable 1 and the feature random variables in F. Let Π_F denote orthogonal projection onto $V(F)$, and define

$$r_F = Y - \Pi_F Y.$$

For a candidate g, write $G = g(X)$ and

$$\tilde{G} = G - \Pi_F G.$$

Under squared loss with an affine linear readout, $\mathcal{R}^\star_{\mathcal{H}_{\mathrm{lin}},\ell}(F)$ is the squared distance from Y to $V(F)$, namely

$$\mathcal{R}^\star_{\mathcal{H}_{\mathrm{lin}},\ell}(F) = \|Y - \Pi_F Y\|_2^2 = \|r_F\|_2^2 .$$

If $\left\|\tilde{G}\right\|_2 = 0$, then $G \in V(F)$ almost surely and adding g does not change the best affine readout. Otherwise,

$$V(F \cup \{g\}) = V(F) \oplus \mathrm{span}\{\tilde{G}\},$$

so the decrease in squared-loss risk from adding g is the squared length of the projection of r_F onto $\mathrm{span}\{\tilde{G}\}$:

$$\Delta_{\mathcal{H}_{\mathrm{lin}},\ell}(F, g) = \frac{\left\langle r_F, \tilde{G} \right\rangle^2_{L_2(P)}}{\left\|\tilde{G}\right\|_2^2}. \tag{S.2}$$

Therefore, for any candidate class $\mathcal{G}$, maximizing the squared-loss gain is equivalent to the standard matching-pursuit step of selecting the residualized atom with largest normalized alignment with the current residual:

$$\operatorname*{arg\,max}_{g \in \mathcal{G}: \left\|\tilde{G}\right\|_2 > 0} \Delta_{\mathcal{H}_{\mathrm{lin}},\ell}(F, g) = \operatorname*{arg\,max}_{g \in \mathcal{G}: \left\|\tilde{G}\right\|_2 > 0} \frac{|\left\langle r_F, \tilde{G} \right\rangle_{L_2(P)}|}{\left\|\tilde{G}\right\|_2}. \tag{S.3}$$

Finite-sample identity. Let $A_F = [\mathbf{1}, Z_F]$, $P_F = A_F A_F^+$, $r_F = (I - P_F)y$, and $\tilde{z} = (I - P_F)z$. If $\tilde{z}^\top \tilde{z} > 0$, the exact decrease in empirical squared loss from adding z to the affine linear selector is

$$\widehat{\Delta}_{\mathrm{lin}}(F, z) = \frac{1}{n} \frac{(r_F^\top \tilde{z})^2}{\tilde{z}^\top \tilde{z}}. \tag{S.4}$$

If $\tilde{z}^\top \tilde{z} = 0$, then $\widehat{\Delta}_{\mathrm{lin}}(F, z) = 0$.

The derivation is the finite-dimensional version of the population argument. Since P_F is the orthogonal projector onto $\mathrm{col}(A_F)$, the current least-squares residual sum of squares is $\|r_F\|_2^2$. If $\tilde{z} = 0$, then $z \in \mathrm{col}(A_F)$, so augmenting A_F with z does not change the column space. Otherwise,

$$\mathrm{col}([A_F, z]) = \mathrm{col}(A_F) \oplus \mathrm{span}\{\tilde{z}\}.$$

The projection of y onto the augmented column space is

$$P_F y + \tilde{z} \frac{\tilde{z}^\top y}{\tilde{z}^\top \tilde{z}}.$$

Because $\tilde{z} \perp \mathrm{col}(A_F)$, $\tilde{z}^\top y = \tilde{z}^\top (I - P_F) y = \tilde{z}^\top r_F$. The decrease in residual sum of squares is therefore

$$\frac{(r_F^\top \tilde{z})^2}{\tilde{z}^\top \tilde{z}}.$$

Dividing by n gives the empirical squared-loss decrease.

Thus, for any finite candidate set $\mathcal{G}$, greedy maximization of the finite-sample linear gain is equivalent to selecting the residualized candidate vector with largest normalized dot product with the current residual:

$$\operatorname*{arg\,max}_{z \in \mathcal{G}: \tilde{z} \neq 0} \widehat{\Delta}_{\mathrm{lin}}(F, z) = \operatorname*{arg\,max}_{z \in \mathcal{G}: \tilde{z} \neq 0} \frac{|r_F^\top \tilde{z}|}{\|\tilde{z}\|_2}. \tag{S.5}$$

Redundancy invariance. For any $a \neq 0$ and any vector $v \in \mathrm{col}(A_F)$,

$$\widehat{\Delta}_{\mathrm{lin}}(F, az + v) = \widehat{\Delta}_{\mathrm{lin}}(F, z). \tag{S.6}$$

In particular, every candidate already contained in $\mathrm{col}(A_F)$ has zero gain. Since $(I - P_F)v = 0$,

$$(I - P_F)(az + v) = a(I - P_F)z = a\tilde{z}.$$

Substituting $a\tilde{z}$ into Eq. (S.4) cancels the factor a^2 in the numerator and denominator. If $z \in \mathrm{col}(A_F)$, then $\tilde{z} = 0$, and the finite-sample gain is zero.

C.3. Linear Residual-Gain Scoring

On the construction training split, let $y \in \mathbb{R}^n$ be the target vector. At stage t, let S_t denote the accepted symbolic atoms and define the full current representation as

$$F_t = B \cup Q \cup S_t.$$

Let $Z_t \in \mathbb{R}^{n \times m_t}$ be the design matrix obtained by evaluating all features in F_t on the n construction training samples. LinFE uses an affine OLS selector $A_t = [\mathbf{1}, Z_t], P_t = A_t A_t^+, r_t = (I - P_t)y$, where A_t^+ is the Moore–Penrose pseudoinverse. For a candidate expression g, write $z_g = (g(x_1), \ldots, g(x_n))^\top$ and remove the component already explained by the affine span of the current representation,

$$\tilde{z}_g = (I - P_t)z_g.$$

Computationally, this orthogonalization is realized via an evolving orthonormal basis $\mathcal{Q}_t$ rather than explicit matrix multiplication, as detailed below.

Computational Note & Time Complexity. While the theoretical projection utilizes an $n \times n$ matrix, it is never explicitly instantiated to avoid prohibitive $\mathcal{O}(n^2)$ memory constraints. Instead, mirroring OMP implementations [13], LinFE incrementally maintains an orthonormal basis $\mathcal{Q}_t$ of the current representation space via modified Gram-Schmidt updates. This mathematically exact approach restricts the orthogonal projection of any newly generated candidate vector to strictly $\mathcal{O}(n \cdot m_t)$ operations, where m_t is the number of currently selected features. Consequently, scoring an entire population of C candidates during evolutionary search requires only $\mathcal{O}(C \cdot n \cdot m_t)$ time. This guarantees theoretical scalability for large tabular datasets and fundamentally bypasses the expensive iterative model refitting required by tree-based selectors.

LinFE scores the candidate by the finite-sample residual gain

$$\widehat{\Delta}_t(g) = \frac{1}{n} \frac{(r_t^\top \tilde{z}_g)^2}{\tilde{z}_g^\top \tilde{z}_g + \varepsilon}, \tag{S.7}$$

where $\varepsilon \to 0^+$ is a microscopic numerical stabilizer. This term exists strictly to safely assign zero gain to redundant features (where $\|\tilde{z}_g\|_2 \approx 0$) during floating-point arithmetic, practically preserving the exact geometric projection without introducing approximation heuristics. Thus Eq. (S.7) selects the residualized atom with the largest normalized alignment with the current residual; the exact statement and proof are in Appendix C.2. The score has no complexity penalty and no downstream-model term. Complexity is controlled by the symbolic grammar and by validation stopping.

Algorithm 1: LinFE Main Loop

Require: train/val splits, base terminals B, categories $\mathcal{C}$
Require: safe grammar $\mathcal{G}_{\text{safe}}$, patience p, max stages T
Fit initial affine OLS on B, yielding initial residual r_0
$Q \leftarrow \textsc{CategoricalTerminals}(\mathcal{C}, r_0, \lambda)$ ▷ **Alg. 2**
Let $\mathcal{Q}_0$ be the orthonormal basis of $B \cup Q$, and r_0 its residual
$S_0 \leftarrow \emptyset$, $S_{\text{best}} \leftarrow \emptyset$, $m_{\text{best}} \leftarrow \infty$, $k \leftarrow 0$
for $t = 0, \ldots, T-1$ **do**
  $\mathcal{A}_t \leftarrow \textsc{Search}(\mathcal{G}_{\text{safe}}, B \cup Q, r_t)$ ▷ evolution
  $g_t \leftarrow \textsc{GreedyScore}(\mathcal{A}_t, r_t, \mathcal{Q}_t)$ ▷ **Alg. 3**
  $S_{t+1} \leftarrow S_t \cup \{g_t\}$
  $z \leftarrow g_t(X_{\text{tr}})$
  $\tilde{z} \leftarrow z - \sum_{q \in \mathcal{Q}_t} \langle z, q \rangle q$ ▷ orthogonalize
  $q_{t+1} \leftarrow \tilde{z} / \|\tilde{z}\|_2$ ▷ new basis
  $\mathcal{Q}_{t+1} \leftarrow \mathcal{Q}_t \cup \{q_{t+1}\}$
  $r_{t+1} \leftarrow r_t - \langle r_t, q_{t+1} \rangle q_{t+1}$ ▷ $\mathcal{O}(n)$ update
  $m_t \leftarrow$ val RMSE of affine OLS on $B \cup Q \cup S_{t+1}$
  if $m_t < m_{\text{best}} - \tau$ **then**
    $S_{\text{best}} \leftarrow S_{t+1}$, $m_{\text{best}} \leftarrow m_t$, $k \leftarrow 0$
  else
    $k \leftarrow k + 1$; **if** $k \geq p$ **then break**
return exported representation $B \cup Q \cup S_{\text{best}}$

Algorithm 2: Categorical Terminals

Require: categories $\mathcal{C} = \{C_1, \ldots, C_K\}$, residual r_0, smoothing λ
$Q \leftarrow \emptyset$
for all $C \in \mathcal{C}$ **do**
  Let $\mathcal{L}_C$ be the observed levels of C
  for all $c \in \mathcal{L}_C$ **do**
    $I_c \leftarrow \{i : C_i = c\}$
    $\theta_c \leftarrow \sum_{i \in I_c} r_{0,i} / (\lambda + |I_c|)$
  Define $f_C(C_i) = \theta_{C_i}$
  $Q \leftarrow Q \cup \{f_C(C)\}$
return Q

Algorithm 3: Greedy Feature Scoring

Require: candidate pool $\mathcal{A}_t$, residual r_t, basis set $\mathcal{Q}_t$
for all $g \in \mathcal{A}_t$ **do**
  $z_g \leftarrow g(X_{\text{tr}})$
  $\tilde{z}_g \leftarrow z_g - \sum_{q \in \mathcal{Q}_t} \langle z_g, q \rangle q$
  $\widehat{\Delta}_t(g) \leftarrow \dfrac{(r_t^\top \tilde{z}_g)^2}{n(\tilde{z}_g^\top \tilde{z}_g + \varepsilon)}$
return $\arg\max_{g \in \mathcal{A}_t} \widehat{\Delta}_t(g)$

C.4. LinFE Algorithm

The detailed algorithm of LinFE is given in Algorithms 1–3.

C.5. Symbolic Candidate Search

The candidate class is an implicit grammar of expression trees over numerical terminals, fixed categorical residual-mean terminals, and scalar constants. In our experiments, LinFE uses the fixed shallow grammar

$$\begin{aligned} & e ::= u \mid a \mid \phi(e) \mid \psi(e, e), \\ & u \in B \cup Q, \qquad a \in [-2, 2], \\ & \phi \in \{\sin, \cos, \tanh, |\cdot|, \log(1 + |\cdot|), \sqrt{|\cdot|}, (\cdot)^2, -(\cdot)\}, \\ & \psi \in \{+, -, \times, \text{safeDiv}, \max, \min\}. \end{aligned} \tag{S.8}$$

Here B denotes processed numerical raw terminals, Q denotes fixed categorical residual-mean terminals, and a denotes an ephemeral scalar constant sampled from $[-2, 2]$. The maximum operator depth is fixed to $d = 4$, yielding shallow expressions that reduce search variance, limit overfitting, and keep the selected features inspectable. In implementation, the grammar is safe and typed: invalid expressions producing NaNs, infinities, or near-constant outputs are discarded, and categorical terminals are only used through admissible linear combinations or interactions, rather than arbitrary nonlinear wrappers.

Rather than enumerate the grammar, LinFE uses an evolutionary search procedure: it generates a population of valid expressions, screens candidates with a cheap residual-alignment proxy, exactly evaluates a shortlist using Eq. (S.7), and updates the population by selection, mutation, crossover, and elitism. Thus, the symbolic search is heuristic, while the score assigned to every shortlisted candidate

is the exact OLS residual gain relative to the current full representation. The search is repeated at every construction stage against the current residual, so the dictionary is accessed adaptively rather than as a fixed global pool.

In Algorithm 1, B denotes the processed non-categorical raw terminals retained throughout construction, and S denotes the accepted symbolic atoms. Categorical terminals Q are statically generated prior to the search using the initial residual r_0.

Initialization of r_0: We compute r_0 by fitting an affine OLS strictly on the raw continuous features B, yielding $r_0 = (I - P_B)y$. This forces the categorical embeddings to capture only the target variance unexplained by B, mathematically eliminating redundancy between the discrete and continuous bases from the start. For purely categorical datasets ($B = \emptyset$), the initial OLS reduces to an intercept-only model, naturally simplifying r_0 to the mean-centered target, $y - \bar{y}$.

C.6. Categorical Variables as Symbolic Terminals

Algorithm 2 implements the categorical mapping strategy. Rather than expanding a categorical variable into many independent one-hot terminals, we construct a single depth-one tree over that variable prior to the symbolic search. Let r_0 be the initial residual vector obtained by fitting an affine OLS on the continuous base features B. If C has levels $\mathcal{L}_C$, the fitted terminal is

$$f_C(c) = \theta_c, \qquad \theta_c = \frac{\sum_{i:C_i=c} r_{0,i}}{\lambda + |\{i : C_i = c\}|}. \tag{S.9}$$

To prevent target leakage and over-reliance on rare categorical levels, we incorporate standard additive smoothing (e.g., Laplace smoothing) directly into the terminal definition. $\lambda > 0$ is a regularization parameter that shrinks the representation of low-frequency categories toward zero (the expected residual prior).

Consequently, K categorical variables contribute exactly K static functional variables to form the set Q. Crucially, this mapping is performed exactly once before the greedy construction begins. During the main LinFE loop, the set Q acts as a fixed set of continuous terminals alongside B.

To maintain semantic validity and interpretability, LinFE enforces a *safe grammar* for these categorical terminals. While numerical features may undergo arbitrary transformations, the grammar restricts $f_C(C)$ to linear combinations or pairwise interactions (e.g., cross-features like $x_j f_C(C)$ or $f_{C_i}(C_i) f_{C_j}(C_j)$). Arbitrary nonlinear wrappers around categorical means, such as $\sin(f_C(C))$, are strictly prohibited. This ensures that the constructed features logically represent category-specific slopes or scaling factors.

C.7. Stopping and Downstream Export

LinFE uses a linear model twice during construction. First as the residual-gain selector and second as the early-stopping monitor. After each accepted atom, LinFE refits an affine OLS readout on the current training representation ($B \cup Q \cup S$) and evaluates the RMSE on the validation split. If the validation RMSE does not improve for p stages, the algorithm stops and restores the best validation snapshot.

The exported representation strictly mirrors the construction environment. It contains the base non-categorical features B, the fixed categorical residual terminals Q, and the selected symbolic expressions S_{best}. Any symbolic expression utilizing $f_C(C)$ is evaluated using the exact same static mapping established during the initial construction phase. The final downstream learner, i.e., linear,

NNs, CatBoost, LightGBM, or RF, is trained only after this fully instantiated representation has been frozen.

Table 3: OpenML regression datasets used in the final benchmark suite (Feature and sample counts are reported after target removal and our preprocessing (e.g., handling missing value).

Dataset	Version	Abbr.	(Features, Samples)	Cat. Features
abalone	5	AB	(8, 4177)	1
autoMpg	1	AU	(7, 392)	3
Bike_Sharing_Demand	7	BI	(11, 17379)	5
bodyfat	1	BO	(14, 252)	0
california	4	CA	(8, 20640)	0
chscase_census2	1	CH	(7, 400)	0
concrete_compressive_strength	7	CO	(8, 1030)	0
cpu_activity	6	CP	(21, 8192)	0
cpu_small	2	CPU	(12, 8192)	0
energy_efficiency	9	EN	(8, 768)	0
fri_c3_100_25	1	FR	(25, 100)	0
grid_stability	7	GR	(12, 10000)	0
kin8nm	1	KI	(8, 8192)	0
liver-disorders	1	LI	(5, 345)	0
Moneyball	2	MO	(12, 420)	4
QSAR_fish_toxicity	7	QS	(6, 908)	0
sensory	1	SE	(11, 576)	11
stock	1	ST	(9, 950)	0
tecator	1	TE	(124, 240)	0
visualizing_soil	4	VI	(4, 8641)	1

Appendix D. More Experimental Details

D.1. Datasets

Table 3 gives the complete benchmark suite used in the main experiments. The $(\#\text{features}, \#\text{samples})$ column is counted after the same preprocessing pipeline used by the experiments and before feature generation; sample counts are the processed train, validation, and test rows combined. The abbreviations are the column headers used in the per-dataset appendix tables.

D.2. Experimental Setup

Datasets and splits. We evaluate on 20 OpenML [19] regression datasets. Datasets detailed information see Appendix D.1. For each dataset, we use five random seeds, split the processed data into train/validation/test partitions (64/16/20), construct features only from the train/validation side, and report the final test RMSE. All methods use the same splits and preprocessing for a given seed. Experiments were run on a server with 8 NVIDIA A100-SXM4-80GB GPUs.

FE baselines. We compare *LinFE* with four automated FE baselines: an MI-objective variant, FEAT [11], AutoFeat [9], and OpenFE [22]. The MI variant uses the same fixed symbolic-search budget as *LinFE* but replaces residual gain with an MI objective [2, 16]. FEAT, AutoFeat, and OpenFE are run with their released/default feature-generation configurations, with only seed, worker count, and logging controls set by the runner. *LinFE* also uses one fixed construction configuration for all datasets and downstream learners. Detailed FE configurations are in Appendix D.4.

Downstream learners and metrics. To ensure fair comparison, all representations are evaluated under identical downstream settings. We study two regimes: a main regime using *Optuna*-tuned [1] CatBoost [17], LightGBM [10], RF [3], and NNs (MLP and ResNet-style [8]; Appendix D.5), and a secondary regime using fixed, untuned Linear, CatBoost, LightGBM, and RF models. Following standard tabular benchmarks, we evaluate performance using RMSE and dataset-wise min–max normalized RMSE, with mean rank across datasets serving as our primary aggregate statistic [5–7, 14, 20, 21].

D.3. Protocol, Preprocessing, and Reporting

For every dataset and seed, we use the same split for all methods: 80% train/validation and 20% test, followed by a 20% validation split inside the train/validation portion, giving 64/16/20 train/validation/test proportions. Regression splits are random and seeded. Feature generators are fit only on the construction train/validation data and export a frozen representation before any final downstream test evaluation. The test split is used only for reporting the final RMSE of the frozen FE representation paired with the selected downstream learner.

Numerical features are processed with the runner's robust or simple preprocessing strategy, depending on the dataset metadata. Categorical variables are represented according to the method being evaluated: baseline methods receive the processed representation produced by their runner, while LinFE replaces each categorical group by the residual-mean terminal described in Section C.6. All final numbers are averaged over seeds $42, 43, 44, 45, 46$.

For normalized RMSE in Tables 1 and 8, we first average test RMSE over seeds for each dataset, method, and downstream learner. We then min–max normalize RMSE within each dataset and downstream-learner block across the compared methods. Mean ranks are computed in the same paired blocks, so every method is ranked only against methods with valid results for the same dataset and downstream learner.

D.4. Feature Engineering Configurations

Table 4 lists the FE configurations used in the benchmark. All FE methods are run with fixed construction hyperparameters across datasets; downstream hyperparameters are tuned or fixed only after the FE representation is frozen.

D.5. Downstream Learner Configurations

Table 5 gives the fixed downstream configurations. Table 6 gives the Optuna search spaces. Tree-based Optuna learners use 25 trials with a 900-second timeout per dataset, seed, method, and downstream learner. NN Optuna uses 80 trials with a 1800-second timeout and selects between MLP and ResNet-style architectures. Optuna optimizes validation RMSE with a seeded TPE sampler; after selection, the final learner is refit on train+validation and evaluated once on the test split.

D.6. Optuna Significance Test Details

All significance tests use the Optuna tuning configuration. For each dataset, downstream learner, and method, the input value is the seed-averaged test RMSE over the five seeds. Negative values indicate that LinFE has lower RMSE. We apply a one-sided Wilcoxon signed-rank test with alternative median $\rho_{i,d}^{(b)} < 0$. Table 7 reports pooled paired tests over all 80 cases (20 datasets × 4 downstream

Table 4: Feature-generation configurations.

Method	Configuration
LinFE	Pure residual-gain score in Eq. (S.7); population 120; generations 15; tournament size 5; elite size 8; mutation probability 0.35; crossover probability 0.55; constant probability 0.15; maximum symbolic depth 4; maximum tree depth 5; maximum nodes 31; wrapper shortlist 24; archive size 64; duplicate-correlation threshold 0.9995; minimum feature std. 10^{-8}; early stopping with an affine linear validation readout and patience 4.
MI-obj	Same fixed symbolic search budget as LinFE, but the candidate objective is replaced by an MI relevance objective. This isolates the effect of the residual-gain score from the symbolic grammar and search budget.
FEAT	Released FEAT-style evolutionary symbolic FE configuration: population 120; generations 8; maximum depth 4; tournament size 4; mutation and crossover probabilities 0.45 each; constant probability 0.15; internal validation fraction 0.25; near-duplicate correlation threshold 0.9995; minimum feature std. 10^{-8}.
AutoFeat	AutoFeatRegressor feature-generation runner with released/default feature-construction settings; the runner sets only random seed, worker count, and verbosity.
OpenFE	OpenFE released/default regression configuration with fixed seed and worker count. Generated OpenFE representations are frozen before downstream training and are evaluated under the same downstream protocols as all other FE methods.

Table 5: Fixed downstream learner configurations.

Learner	Fixed configuration
Linear	OLS with intercept; no regularization.
CatBoost	RMSE loss; 300 iterations; depth 6; learning rate 0.06; L_2 leaf regularization 3.0; random strength 1.0; Bayesian bootstrap with bagging temperature 0.5; border count 128; seeded training.
LightGBM	Regression objective; 300 estimators; max depth 6; learning rate 0.06; num leaves 31; subsample 0.8; column subsample 0.8; min child samples 20; $L_1 = 10^{-8}$; $L_2 = 1.0$; seeded training.
RF	300 trees; unlimited max depth; min samples split 2; min samples leaf 1; max features 1.0; bootstrap enabled; no max-samples subsampling; seeded training.

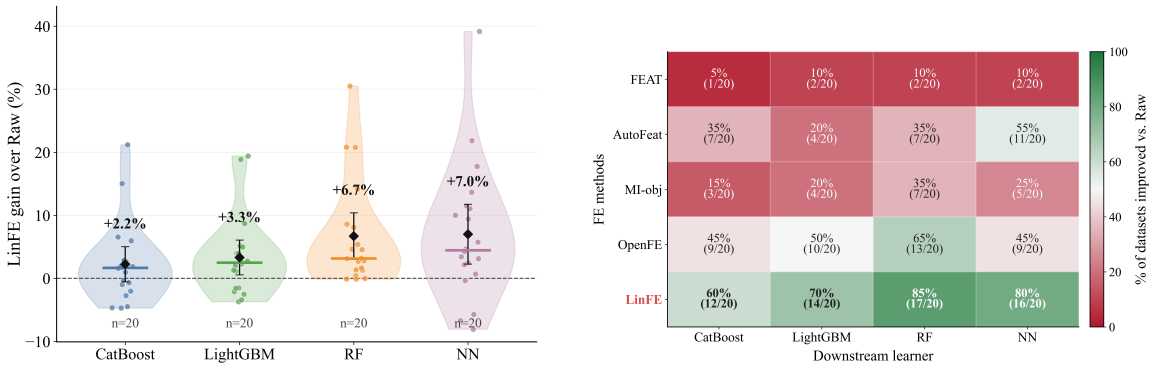
learners) and applies Holm correction across the five baselines. The Raw row corresponds to the pooled row in Table 9.

Table 6: Optuna downstream search spaces. All ranges are inclusive; log indicates log-uniform sampling.

Learner	Optuna search space
CatBoost	Iterations $[150, 800]$; depth $[2, 8]$; learning rate $[0.01, 0.20]$ log; L_2 leaf regularization $[10^{-3}, 20]$ log; random strength $[10^{-8}, 10]$ log; bagging temperature $[0, 1]$; border count $[32, 255]$.
LightGBM	Estimators $[150, 800]$; max depth $[2, 8]$; learning rate $[0.01, 0.20]$ log; num leaves $[8, 255]$; subsample $[0.60, 1.00]$; column subsample $[0.60, 1.00]$; min child samples $[5, 80]$; L_1 $[10^{-8}, 5]$ log; L_2 $[10^{-3}, 20]$ log.
RF	Trees $[150, 800]$; max depth in $\{\mathrm{None}, 4, 6, 8, 12, 16, 24, 32\}$; min samples split $[2, 40]$; min samples leaf $[1, 20]$; max features in $\{\sqrt{\cdot}, \log_2(\cdot), 0.5, 0.75, 1.0\}$; bootstrap in $\{\mathrm{true}, \mathrm{false}\}$; if bootstrap is true, max samples in $[0.60, 1.00]$.
NNs	Architecture in {MLP, ResNet}; width in $\{32, 64, 128, 256, 512\}$; MLP hidden layers in $\{0, 1, 2, 3\}$; ResNet blocks in $\{1, 2, 3\}$; batch size in $\{64, 128, 256, 512\}$; learning rate $[10^{-4}, 3 \cdot 10^{-3}]$ log; weight decay $[10^{-8}, 10^{-2}]$ log; dropout $[0, 0.35]$; epochs $[80, 400]$; early-stopping patience in $\{10, 20, 40\}$.

Table 7: Pooled Optuna paired significance of LinFE against each baseline over 80 cases (20 datasets × 4 downstream learners). W/T/L counts cells where LinFE wins, ties, or loses. Relative changes are percentages; negative values mean lower RMSE for LinFE.

Baseline	Pairs	W/T/L	Mean $\Delta\%$	Median $\Delta\%$	p	Holm p
Raw	80	59/0/21	-4.81	-2.73	3.0×10^{-7}	6.1×10^{-7}
FEAT	80	74/0/6	-33.04	-32.78	1.3×10^{-13}	6.5×10^{-13}
AutoFeat	80	59/0/21	-5.49	-2.69	2.5×10^{-8}	1.0×10^{-7}
MI-obj	80	61/0/19	-14.08	-14.30	1.6×10^{-7}	4.9×10^{-7}
OpenFE	80	55/0/25	-7.91	-1.85	1.3×10^{-5}	1.3×10^{-5}



(a) LinFE's relative RMSE gain over Raw.

(b) Win rate of FE methods against Raw.

Figure 2: **Paired evaluation of FE methods under dense downstream tuning.** All comparisons are paired by dataset and seed. **(a)** Relative RMSE improvements of LinFE over Raw. **(b)** Head-to-head win rates of FE methods against Raw across downstream model families.

Appendix E. More Experiment Results

E.1. Optimized Downstream Learners

Figure 2 dissects this advantage via paired relative gains and head-to-head win rates. Panel (b) reveals a stark contrast in robustness: while established FE baselines frequently degrade the performance of densely tuned models (often exhibiting win rates well below 40%), LinFE is the exception, maintaining a dominant 60%–85% win rate across all downstream families. Panel (a) confirms that these are not marginal gains, with mean improvements reaching +6.7% and +7.0% for RF and NNs, respectively. Crucially, these gains are not caused by a weak downstream model being rescued by obvious nonlinearities. Instead, they appear after the downstream learner has already been thoroughly optimized, indicating that LinFE successfully reshapes the coordinate system in a way that remains structurally useful even for high-capacity learners.

E.2. Fixed Downstream Learners

To ensure that the performance disparities observed in Table 1 are not merely artifacts of the downstream hyperparameter optimization process, we introduce a supplementary evaluation with fixed downstream learners. This controlled setting decouples the efficacy of the FE methods from downstream tuning dynamics, providing a standard environment where established FE methods typically demonstrate clear utility. For a focused comparison, we benchmark our LinFE against Raw features and the two most competitive baselines from the optimized study, AutoFeat and OpenFE. The exact fixed configurations are detailed in Appendix D.5.

Table 8 confirms that the conventional FE methods evaluated in our benchmark are highly competent. Under fixed-configuration downstream models, baselines like AutoFeat and OpenFE consistently reduce test error compared to the raw dataset across multiple learners. This controlled setting proves that we are not evaluating intentionally weakened FE algorithms. Rather, their utility is fundamentally tied to the capacity of the downstream model. The divergence occurs exclusively

Table 8: **Performance on fixed-configuration downstream learners across 20 OpenML datasets.** Raw (−) denotes the original dataset without any FE. ↓ indicates that lower values are better. Rank represents the method's average relative standing across all datasets. Norm. stands for min–max normalized RMSE. All values are dataset-level averages over seeds with 95% confidence intervals. Best results are **bolded**, second-best are underlined.

Method	Linear†			CatBoost			LightGBM			Random Forest		
Metric	Rank ↓	RMSE ↓	Norm. ↓	Rank ↓	RMSE ↓	Norm. ↓	Rank ↓	RMSE ↓	Norm. ↓	Rank ↓	RMSE ↓	Norm. ↓
Raw (−)	2.562 ± 0.428	2.960 ± 2.576	0.519 ± 0.189	3.025 ± 0.352	4.808 ± 4.903	0.634 ± 0.166	3.125 ± 0.375	4.806 ± 4.757	0.679 ± 0.158	3.075 ± 0.428	4.873 ± 4.915	0.626 ± 0.194
AutoFeat	2.438 ± 0.389	2.957 ± 2.572	0.541 ± 0.163	2.725 ± 0.479	**4.706 ± 4.752**	0.537 ± 0.174	2.475 ± 0.490	**4.685 ± 4.619**	0.425 ± 0.182	2.575 ± 0.410	4.805 ± 4.830	0.396 ± 0.147
OpenFE	3.250 ± 0.490	2.723 ± 2.611	0.634 ± 0.224	**1.850 ± 0.537**	5.276 ± 5.690	**0.298 ± 0.188**	**2.200 ± 0.579**	5.545 ± 6.090	**0.400 ± 0.195**	2.350 ± 0.574	6.354 ± 7.550	0.487 ± 0.207
LinFE	**1.750 ± 0.607**	**2.508 ± 2.570**	**0.254 ± 0.216**	2.400 ± 0.459	4.771 ± 4.882	0.448 ± 0.161	**2.200 ± 0.417**	4.811 ± 4.798	0.425 ± 0.172	**2.000 ± 0.450**	**4.799 ± 4.904**	**0.268 ± 0.152**

† For the Linear downstream, statistics exclude specific datasets where baselines experienced severe numerical instability (see Appendix E).

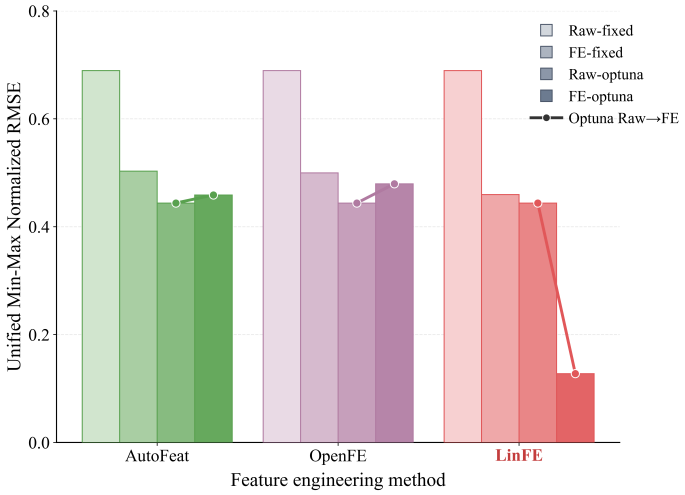


Figure 3: **Unified comparison of feature engineering impact under fixed (no tuning) versus Optuna-tuned downstream regimes.** Test RMSE is averaged across the four regime models, min–max normalized per dataset, and finally averaged over all 20 benchmarks. *Raw* denotes the baseline without feature engineering. *fixed* downstream learner includes CatBoost, LightGBM and RF. *optuna* downstream learner includes NNs, CatBoost, LightGBM and RF.

when the final learner is densely tuned; however, LinFE consistently circumvents this failure mode because its exact orthogonalization assigns strictly zero value to candidate features already spanned by the current representation.

Figure 3 visually summarizes this divergence. The bar heights confirm that traditional FE significantly improves *fixed* models, yet the connecting lines reveal a stark reversal under *Optuna* tuning, where the optimized raw baseline suddenly becomes much harder to beat. This empirical phenomenon captures the paradox introduced in Section 1: *most automated FE methods flood the representation with extra coordinates but lack a mechanism to control redundancy, actively burdening the downstream model selection*.

Table 9: **Statistical significance of LinFE vs. Raw under Optuna tuning.** W/T/L: datasets where LinFE wins, ties, or loses against Raw. Mean/Median $\Delta\%$: relative RMSE change (negative is better). p-values: one-sided Wilcoxon signed-rank test with Holm correction across families; significant results (< 0.05) are **bolded**. *All Optuna cells* indicates the global paired diagnostic.

Downstream	Pairs	W/T/L	Mean $\Delta\%$	Median $\Delta\%$	p	Holm p
CatBoost	20	12/0/8	-2.24	-1.66	0.115	0.115
LightGBM	20	14/0/6	-3.31	-2.50	**0.0107**	0.0215
RF	20	17/0/3	-6.69	-3.16	$\mathbf{1.3\times10^{-5}}$	5.3×10^{-5}
NNs	20	16/0/4	-7.00	-4.45	**0.00279**	0.00837
All Optuna cells	80	59/0/21	-4.81	-2.73	$\mathbf{3.0 \times 10^{-7}}$	–

E.3. Paired Significance Under Optuna

We next test whether LinFE's gains over the raw optimized learner are statistically reliable. For each dataset and Optuna downstream learner, we compare the seed-averaged test RMSE of LinFE and Raw on the same split. The test statistic is the paired relative RMSE change $\rho_{i,d} = \frac{\mathrm{RMSE}_{i,d}^{\mathrm{LinFE}} - \mathrm{RMSE}_{i,d}^{\mathrm{Raw}}}{\mathrm{RMSE}_{i,d}^{\mathrm{Raw}}}$, so negative values indicate that LinFE improves over Raw. We apply a one-sided Wilcoxon signed-rank test with alternative median $\rho_{i,d} < 0$. The downstream-specific p-values are Holm-corrected over the four Optuna downstream learners. Appendix D.6 gives additional paired tests against all other FE baselines.

Table 9 shows that the gains are not driven by a single downstream family. LinFE significantly improves over Raw for LightGBM, RF, and NNs after Holm correction, and the pooled paired diagnostic is highly significant across all 80 cases (20 datasets × 4 downstream learners). CatBoost remains positive in mean and median paired RMSE, but is not significant($p < 0.05$) at the dataset level. More experiment results are shown in Appendix E.

E.4. Optuna Downstream Learner Results

The detailed performance results among all FE methods (i.e., Raw without FE, FEAT, AutoFeat, MI-Objective FE, OpenFE and LinFE) and across multiple downstream learners (i.e., CatBoost, LightGBM, Random Forests, and NNs) under dense Optuna tuning are shown in Table 10, 11, 12, 13.

E.5. Fixed Downstream Learner Results without Tuning

The detailed performance results among all FE methods (i.e., Raw without FE, AutoFeat, OpenFE, and LinFE) and across multiple downstream learners (i.e., Linear Regression, CatBoost, LightGBM, and Random Forests) under fixed configurations are shown in Table 14, 15, 16, 17.

It should be noted that for the fixed Linear readout, numerical instability (rank-deficiency) occasionally occurs when a feature generator produces highly collinear coordinates. We report these as failures ($\times$) and, to maintain a valid comparison, we only compute aggregate statistics over datasets where all compared methods produced stable numerical results.

Table 10: Dataset-level RMSE for CatBoost-Optuna downstream. Each entry reports mean ± std over random seeds. Bold indicates the best method and underline indicates the second-best method within each dataset. The table is split into two blocks, each showing up to 10 datasets.

Method	AB	AU	BI	BO	CA	CH	CO	CP	CPU	EN
Raw (−)	2.1358 ± 0.0890	2.9391 ± 0.2518	**41.5236 ± 1.4331**	**1.0269 ± 0.7832**	0.1303 ± 0.0032	**0.5124 ± 0.0341**	**4.1262 ± 0.6073**	2.4626 ± 0.3019	2.8262 ± 0.0832	0.2899 ± 0.0427
FEAT	2.2340 ± 0.0515	3.5017 ± 0.3050	102.1076 ± 6.9617	1.0944 ± 0.8721	0.2254 ± 0.0052	0.5273 ± 0.0399	9.9335 ± 2.1514	3.1542 ± 0.4362	4.7346 ± 1.3007	1.5437 ± 1.1608
AutoFeat	2.1461 ± 0.0821	2.9391 ± 0.2518	41.7539 ± 0.8335	**1.0269 ± 0.7832**	0.1312 ± 0.0051	**0.5124 ± 0.0341**	4.1288 ± 0.6269	2.3426 ± 0.1104	2.8984 ± 0.2823	**0.2829 ± 0.0580**
MI-obj	2.1651 ± 0.0953	3.2412 ± 0.1542	60.9030 ± 7.6425	1.0695 ± 0.9381	0.1833 ± 0.0156	0.5248 ± 0.0277	6.2320 ± 0.4513	2.7714 ± 0.1163	3.3549 ± 0.1588	0.4416 ± 0.0380
OpenFE	**2.0236 ± 0.0929**	2.8950 ± 0.2643	55.4965 ± 4.1089	1.0999 ± 0.9251	**0.1277 ± 0.0024**	0.5190 ± 0.0340	4.4290 ± 0.6915	**2.2703 ± 0.0605**	**2.7304 ± 0.0585**	1.1637 ± 0.4592
LinFE	2.0775 ± 0.0646	**2.4976 ± 0.2549**	42.6638 ± 1.0784	1.0728 ± 0.8601	0.1279 ± 0.0028	0.5228 ± 0.0504	4.3203 ± 0.5991	2.3010 ± 0.1098	2.7777 ± 0.0750	0.2871 ± 0.0381

Method	FR	GR	KI	LI	MO	QS	SE	ST	TE	VI
Raw (−)	0.7463 ± 0.1247	0.0068 ± 0.0002	0.0879 ± 0.0015	3.0015 ± 0.2993	24.0282 ± 2.6078	0.8866 ± 0.0863	0.6842 ± 0.0279	**0.6826 ± 0.0571**	1.0800 ± 0.2875	0.0575 ± 0.0060
FEAT	0.9115 ± 0.1361	0.0250 ± 0.0048	0.2205 ± 0.0051	3.0683 ± 0.2823	23.6903 ± 2.6643	0.9980 ± 0.0472	0.7387 ± 0.0390	1.4105 ± 0.6230	1.4220 ± 0.4445	0.9068 ± 0.3667
AutoFeat	0.7843 ± 0.0734	0.0067 ± 0.0002	0.0937 ± 0.0017	2.9578 ± 0.2500	24.0282 ± 2.6078	**0.8865 ± 0.0881**	0.6842 ± 0.0279	**0.6826 ± 0.0571**	1.0022 ± 0.3428	0.0575 ± 0.0097
MI-obj	**0.5133 ± 0.1367**	0.0123 ± 0.0006	0.2091 ± 0.0027	3.1217 ± 0.3876	26.2901 ± 2.3271	0.9728 ± 0.0811	**0.6822 ± 0.0275**	1.0738 ± 0.1595	**0.7869 ± 0.1937**	0.1179 ± 0.0167
OpenFE	0.7592 ± 0.1067	**0.0064 ± 0.0003**	**0.0823 ± 0.0021**	**2.9140 ± 0.2527**	24.2977 ± 2.6458	0.8997 ± 0.1080	0.6937 ± 0.0425	0.7771 ± 0.0731	0.9967 ± 0.3866	0.0666 ± 0.0202
LinFE	0.7810 ± 0.1983	0.0066 ± 0.0003	0.0865 ± 0.0026	2.9423 ± 0.2335	**23.3462 ± 2.2148**	0.8926 ± 0.0807	0.6909 ± 0.0334	0.6846 ± 0.0364	0.8511 ± 0.2236	**0.0541 ± 0.0065**

Table 11: Dataset-level RMSE for LightGBM-Optuna downstream. Each entry reports mean ± std over random seeds. Bold indicates the best method and underline indicates the second-best method within each dataset. The table is split into two blocks, each showing up to 10 datasets.

Method	AB	AU	BI	BO	CA	CH	CO	CP	CPU	EN
Raw (−)	2.1205 ± 0.0901	2.9768 ± 0.2031	**42.9255 ± 0.5058**	1.4217 ± 0.7446	0.1296 ± 0.0013	0.5119 ± 0.0325	4.5185 ± 0.6544	2.4019 ± 0.1718	2.8749 ± 0.2073	0.3347 ± 0.0376
FEAT	2.2158 ± 0.0566	3.4670 ± 0.2134	101.0941 ± 7.8755	**1.2083 ± 0.8833**	0.2252 ± 0.0059	**0.5095 ± 0.0324**	10.1693 ± 1.9955	3.2328 ± 0.4465	4.7513 ± 1.2961	1.5194 ± 1.1153
AutoFeat	2.1185 ± 0.0895	2.9768 ± 0.2031	43.0195 ± 0.6950	1.4217 ± 0.7446	0.1309 ± 0.0019	0.5119 ± 0.0325	**4.3265 ± 0.6627**	2.3887 ± 0.1544	2.9187 ± 0.2689	0.3404 ± 0.0365
MI-obj	2.1617 ± 0.0865	3.3759 ± 0.2578	60.0083 ± 5.9085	1.3284 ± 0.7057	0.1834 ± 0.0160	0.5170 ± 0.0208	6.3699 ± 0.5426	2.7408 ± 0.1086	3.3673 ± 0.1728	0.4439 ± 0.0361
OpenFE	**2.0304 ± 0.0843**	2.9965 ± 0.1766	60.1836 ± 8.5913	1.4379 ± 0.8381	0.1275 ± 0.0014	0.5174 ± 0.0391	4.3785 ± 0.5306	**2.2911 ± 0.1250**	**2.7475 ± 0.0893**	1.0322 ± 0.2945
LinFE	2.0716 ± 0.0576	**2.7178 ± 0.1822**	43.5750 ± 0.9030	1.3484 ± 0.9217	**0.1270 ± 0.0016**	0.5227 ± 0.0592	4.3402 ± 0.7271	2.3372 ± 0.1251	2.8376 ± 0.1596	**0.3221 ± 0.0627**

Method	FR	GR	KI	LI	MO	QS	SE	ST	TE	VI
Raw (−)	0.7123 ± 0.0485	0.0078 ± 0.0003	0.1148 ± 0.0035	**2.8710 ± 0.2393**	24.8213 ± 2.4241	0.8962 ± 0.0728	0.7058 ± 0.0153	0.8237 ± 0.0525	1.0092 ± 0.1524	0.0639 ± 0.0138
FEAT	0.9415 ± 0.1521	0.0252 ± 0.0047	0.2204 ± 0.0050	3.1389 ± 0.2720	25.1254 ± 2.0692	0.9997 ± 0.0599	0.7298 ± 0.0374	1.4765 ± 0.5920	1.5346 ± 0.3815	0.8675 ± 0.4240
AutoFeat	0.7213 ± 0.0785	0.0075 ± 0.0003	0.1239 ± 0.0030	2.9576 ± 0.3371	24.8213 ± 2.4241	0.9099 ± 0.0804	0.7058 ± 0.0153	0.8237 ± 0.0525	1.3323 ± 0.4934	0.0754 ± 0.0123
MI-obj	**0.5315 ± 0.1466**	0.0124 ± 0.0007	0.2094 ± 0.0026	3.0447 ± 0.3325	26.9115 ± 1.9323	0.9707 ± 0.0649	**0.6785 ± 0.0355**	1.1658 ± 0.1626	**0.8584 ± 0.1786**	0.1093 ± 0.0154
OpenFE	0.6925 ± 0.0603	**0.0065 ± 0.0002**	**0.0921 ± 0.0033**	3.0345 ± 0.3948	24.8960 ± 2.0791	0.9089 ± 0.1056	0.7212 ± 0.0159	**0.7256 ± 0.0500**	1.0439 ± 0.2059	0.0531 ± 0.0102
LinFE	0.7386 ± 0.2041	0.0074 ± 0.0003	0.0931 ± 0.0018	2.9687 ± 0.1788	**24.1427 ± 2.4511**	**0.8859 ± 0.0663**	0.7169 ± 0.0266	0.7908 ± 0.0553	1.0347 ± 0.2437	**0.0515 ± 0.0137**

Table 12: Dataset-level RMSE for Random Forest-Optuna downstream. Each entry reports mean ± std over random seeds. Bold indicates the best method and underline indicates the second-best method within each dataset. The table is split into two blocks, each showing up to 10 datasets.

Method	AB	AU	BI	BO	CA	CH	CO	CP	CPU	EN
Raw (−)	2.1132 ± 0.0824	3.1928 ± 0.1789	48.9400 ± 2.8806	1.3755 ± 1.0801	0.1430 ± 0.0036	0.5175 ± 0.0297	5.4237 ± 0.6918	2.5204 ± 0.1322	2.9360 ± 0.1144	0.5106 ± 0.0403
FEAT	2.2147 ± 0.0607	3.7752 ± 0.2094	101.7437 ± 7.6520	1.2176 ± 1.0944	0.2242 ± 0.0062	**0.5090 ± 0.0335**	10.1865 ± 1.9124	3.2467 ± 0.4277	4.8427 ± 1.1985	1.5321 ± 1.1375
AutoFeat	2.1205 ± 0.0897	3.1928 ± 0.1789	**48.3640 ± 2.8471**	1.3755 ± 1.0801	0.1527 ± 0.0073	0.5175 ± 0.0297	5.6196 ± 0.6149	2.4841 ± 0.0907	2.9397 ± 0.1184	0.4947 ± 0.0433
MI-obj	2.1640 ± 0.0982	3.2563 ± 0.1293	61.7235 ± 7.0347	**1.1905 ± 0.7785**	0.1835 ± 0.0157	0.5115 ± 0.0245	6.7189 ± 0.4744	2.7851 ± 0.0896	3.3697 ± 0.1553	0.4924 ± 0.0657
OpenFE	**2.0274 ± 0.0818**	3.1036 ± 0.2595	64.4026 ± 19.7369	1.3335 ± 0.9011	0.1368 ± 0.0027	0.5192 ± 0.0280	**5.1606 ± 0.4804**	2.4562 ± 0.1273	**2.8602 ± 0.1051**	0.8829 ± 0.2345
LinFE	2.0559 ± 0.0663	**2.5298 ± 0.4013**	48.9847 ± 3.1682	1.3118 ± 0.8429	**0.1365 ± 0.0033**	0.5176 ± 0.0417	5.2726 ± 0.5879	**2.4398 ± 0.0951**	2.9240 ± 0.1124	**0.4667 ± 0.0388**

Method	FR	GR	KI	LI	MO	QS	SE	ST	TE	VI
Raw (−)	0.9223 ± 0.1633	0.0125 ± 0.0006	0.1448 ± 0.0042	2.8569 ± 0.1603	25.1004 ± 2.8662	0.9156 ± 0.0757	0.6968 ± 0.0264	0.7737 ± 0.0496	1.3240 ± 0.3681	0.0493 ± 0.0272
FEAT	1.0076 ± 0.2671	0.0254 ± 0.0043	0.2205 ± 0.0052	3.0592 ± 0.2302	25.7253 ± 1.7497	1.0068 ± 0.0324	0.7399 ± 0.0310	1.5634 ± 0.6111	2.0796 ± 0.8167	0.8691 ± 0.4114
AutoFeat	0.8633 ± 0.0750	0.0124 ± 0.0006	0.1517 ± 0.0054	2.8432 ± 0.1564	25.1004 ± 2.8662	0.9338 ± 0.0846	0.6968 ± 0.0264	0.7737 ± 0.0496	1.3516 ± 0.4297	0.0665 ± 0.0619
MI-obj	**0.7708 ± 0.1682**	0.0146 ± 0.0004	0.2096 ± 0.0026	3.0527 ± 0.2565	26.6225 ± 2.8832	0.9739 ± 0.0727	**0.6889 ± 0.0353**	1.2330 ± 0.1401	**0.9251 ± 0.2552**	0.0464 ± 0.0117
OpenFE	0.9088 ± 0.1625	**0.0106 ± 0.0004**	0.1167 ± 0.0038	2.8404 ± 0.1958	25.1719 ± 3.1971	0.9166 ± 0.1002	0.7169 ± 0.0353	**0.7225 ± 0.0606**	1.4268 ± 0.4143	0.0392 ± 0.0108
LinFE	0.8475 ± 0.1634	0.0107 ± 0.0002	**0.1147 ± 0.0049**	**2.8180 ± 0.1436**	**24.7763 ± 2.4961**	**0.9004 ± 0.0820**	0.6976 ± 0.0316	0.7320 ± 0.0472	1.2826 ± 0.3850	**0.0343 ± 0.0138**

Table 13: Dataset-level RMSE for NN-Optuna downstream. Each entry reports mean ± std over random seeds. Bold indicates the best method and underline indicates the second-best method within each dataset. The table is split into two blocks, each showing up to 10 datasets.

Method	AB	AU	BI	BO	CA	CH	CO	CP	CPU	EN
Raw (−)	2.0693 ± 0.0972	2.9755 ± 0.2576	46.9736 ± 2.3880	1.3859 ± 0.6356	0.1641 ± 0.0312	0.5305 ± 0.0600	5.4371 ± 0.7843	2.7105 ± 0.3337	3.0004 ± 0.0937	0.7109 ± 0.0502
FEAT	2.1900 ± 0.0711	3.4991 ± 0.3260	94.5587 ± 5.9984	1.1415 ± 0.5064	0.2128 ± 0.0113	0.5270 ± 0.0508	10.2518 ± 2.5391	3.9878 ± 0.4220	5.3920 ± 1.1311	1.8031 ± 1.0476
AutoFeat	2.0780 ± 0.0882	2.7593 ± 0.1559	45.9504 ± 1.3081	1.3686 ± 0.5189	0.1509 ± 0.0048	0.5364 ± 0.0412	5.1328 ± 0.7766	2.5022 ± 0.1871	3.0751 ± 0.3113	0.6677 ± 0.1471
MI-obj	2.1429 ± 0.1002	3.4686 ± 0.3858	63.3188 ± 8.0019	**1.0498 ± 0.3406**	0.1933 ± 0.0229	0.5814 ± 0.0829	6.5151 ± 0.6080	2.9439 ± 0.1960	3.7020 ± 0.4341	0.9633 ± 0.2613
OpenFE	**2.0554 ± 0.1006**	2.9884 ± 0.4887	64.6451 ± 16.0036	1.3600 ± 0.3746	0.1447 ± 0.0009	0.5295 ± 0.0532	5.2821 ± 0.8033	**2.4392 ± 0.1272**	2.9838 ± 0.3126	1.7473 ± 0.6015
LinFE	2.0765 ± 0.0768	**2.6463 ± 0.1781**	**44.7933 ± 1.5816**	1.2264 ± 0.4193	**0.1350 ± 0.0036**	**0.5268 ± 0.0702**	**4.6942 ± 0.4852**	2.4396 ± 0.1096	**2.9260 ± 0.0711**	**0.5556 ± 0.0479**

Method	FR	GR	KI	LI	MO	QS	SE	ST	TE	VI
Raw (−)	0.8719 ± 0.1475	0.0062 ± 0.0006	0.0709 ± 0.0034	**2.9976 ± 0.1439**	24.0726 ± 2.4579	0.9657 ± 0.0933	0.7383 ± 0.0370	**0.6847 ± 0.0689**	1.3925 ± 0.2834	0.4164 ± 0.1750
FEAT	1.5103 ± 1.2142	0.0241 ± 0.0054	0.2188 ± 0.0058	3.0486 ± 0.2960	24.9951 ± 3.3100	1.0225 ± 0.0591	0.7478 ± 0.0369	1.0763 ± 0.2546	1.4081 ± 0.4797	0.9890 ± 0.2994
AutoFeat	0.8977 ± 0.1312	0.0059 ± 0.0003	0.0735 ± 0.0052	3.0764 ± 0.1284	23.9714 ± 2.0729	1.0042 ± 0.1785	**0.7145 ± 0.0315**	0.6932 ± 0.0542	1.6966 ± 0.3712	0.3904 ± 0.1075
MI-obj	**0.5135 ± 0.1085**	0.0118 ± 0.0006	0.2087 ± 0.0030	3.1424 ± 0.2608	25.7834 ± 3.3474	0.9874 ± 0.0912	0.7148 ± 0.0202	0.9711 ± 0.2049	**0.9841 ± 0.4580**	0.2942 ± 0.0323
OpenFE	0.9896 ± 0.2810	0.0061 ± 0.0004	0.0880 ± 0.0214	3.0670 ± 0.1124	24.4949 ± 2.4463	0.9442 ± 0.0965	0.7620 ± 0.0541	0.9654 ± 0.1257	1.4332 ± 0.2393	0.6107 ± 0.8158
LinFE	0.9218 ± 0.1828	**0.0059 ± 0.0002**	**0.0687 ± 0.0020**	3.1984 ± 0.3773	**23.2466 ± 1.5654**	**0.9102 ± 0.0692**	0.7222 ± 0.0435	0.7400 ± 0.1028	1.2616 ± 0.2436	**0.2534 ± 0.0779**

Table 14: Dataset-level RMSE for fixed Linear downstream. Each entry reports mean ± std over random seeds. Bold indicates the best method and underline indicates the second-best method within each dataset. The table is split into two blocks, each showing up to 10 datasets. For fixed Linear, × marks a seed-instability failure: although some random seeds may be normal, one or more seeds numerically exploded, so the full run is not treated as a valid five-seed result and is excluded for fairness.

Method	AB	AU	BI	BO	CA	CH	CO	CP	CPU	EN
Raw (−)	2.1706 ± 0.0989	2.7730 ± 0.1287	139.9660 ± 1.3762	0.9368 ± 0.4039	0.2198 ± 0.0030	**0.5139 ± 0.0378**	10.4351 ± 0.4102	2.9590 ± 0.0650	3.4565 ± 0.0487	2.8424 ± 0.1729
AutoFeat	2.1387 ± 0.1044	2.7730 ± 0.1287	139.7616 ± 1.3976	0.9368 ± 0.4039	0.1917 ± 0.0026	**0.5139 ± 0.0378**	6.8735 ± 0.3356	2.7556 ± 0.0944	3.2863 ± 0.1051	2.8544 ± 0.1459
OpenFE	×	2.9427 ± 0.2883	×	0.9513 ± 0.3992	0.2728 ± 0.0988	0.5142 ± 0.0382	×	4.9243 ± 3.8261	3.5055 ± 0.8639	×
LinFE	**2.1086 ± 0.0913**	**2.5374 ± 0.1950**	**90.7067 ± 1.1210**	**0.8910 ± 0.5000**	**0.1637 ± 0.0025**	0.5318 ± 0.0531	**6.2254 ± 0.5207**	**2.5166 ± 0.0776**	**3.0257 ± 0.0553**	**0.4329 ± 0.0478**

Method	FR	GR	KI	LI	MO	QS	SE	ST	TE	VI
Raw (−)	1.1321 ± 0.1482	0.0221 ± 0.0003	0.2023 ± 0.0007	**2.9225 ± 0.3138**	21.9310 ± 2.3200	0.9895 ± 0.0784	**0.7476 ± 0.0194**	2.3667 ± 0.0234	**1.0930 ± 0.2969**	5.0863 ± 0.0589
AutoFeat	1.1149 ± 0.1483	0.0183 ± 0.0002	0.1988 ± 0.0011	2.9406 ± 0.3128	21.9310 ± 2.3200	0.9864 ± 0.0971	**0.7476 ± 0.0194**	2.3667 ± 0.0234	1.4620 ± 0.6209	5.0876 ± 0.0567
OpenFE	1.1444 ± 0.1400	0.0121 ± 0.0003	0.2751 ± 0.1382	2.9415 ± 0.2941	22.0025 ± 2.4095	1.0086 ± 0.1014	0.7483 ± 0.0204	**1.0041 ± 0.0587**	1.1039 ± 0.2583	0.2175 ± 0.0736
LinFE	**1.0623 ± 0.2536**	**0.0087 ± 0.0003**	**0.1412 ± 0.0453**	2.9499 ± 0.2563	**21.8064 ± 1.5334**	**0.9751 ± 0.0961**	0.7518 ± 0.0248	1.1421 ± 0.1753	1.4483 ± 0.8658	**0.1729 ± 0.0128**

Table 15: Dataset-level RMSE for fixed CatBoost downstream. Each entry reports mean ± std over random seeds. Bold indicates the best method and underline indicates the second-best method within each dataset. The table is split into two blocks, each showing up to 10 datasets.

Method	AB	AU	BI	BO	CA	CH	CO	CP	CPU	EN
Raw (−)	2.1279 ± 0.0900	2.8342 ± 0.2236	47.0956 ± 0.7181	1.4780 ± 0.8643	0.1422 ± 0.0028	0.5295 ± 0.0295	4.4532 ± 0.5263	2.4660 ± 0.0973	2.8926 ± 0.0804	0.3348 ± 0.0362
AutoFeat	2.0837 ± 0.0897	2.9617 ± 0.1632	**45.6806 ± 0.6622**	1.4980 ± 0.7041	0.1387 ± 0.0021	0.5295 ± 0.0295	**4.4361 ± 0.5167**	2.3821 ± 0.0947	2.8058 ± 0.0864	0.3563 ± 0.0464
OpenFE	**2.0228 ± 0.0837**	2.8458 ± 0.2112	56.0699 ± 1.6705	1.4465 ± 0.8617	**0.1373 ± 0.0022**	**0.5203 ± 0.0335**	4.6379 ± 0.5389	**2.3589 ± 0.0704**	**2.7752 ± 0.0539**	1.2822 ± 0.2544
LinFE	2.0849 ± 0.0793	**2.5015 ± 0.2781**	46.8204 ± 1.1298	**1.4406 ± 0.8777**	0.1386 ± 0.0032	0.5279 ± 0.0574	4.5965 ± 0.5714	2.4246 ± 0.0856	2.8283 ± 0.0571	**0.3265 ± 0.0316**

Method	FR	GR	KI	LI	MO	QS	SE	ST	TE	VI
Raw (−)	0.7598 ± 0.0800	0.0081 ± 0.0002	0.1113 ± 0.0033	3.0171 ± 0.2403	23.6180 ± 2.2999	0.8781 ± 0.0873	0.7227 ± 0.0260	0.7449 ± 0.0786	1.8008 ± 0.4094	0.1551 ± 0.0097
AutoFeat	0.7749 ± 0.0944	0.0075 ± 0.0002	0.1037 ± 0.0034	3.0201 ± 0.2252	**22.9243 ± 1.9709**	0.8793 ± 0.0807	0.7217 ± 0.0183	0.7494 ± 0.0458	1.8556 ± 0.5463	0.2068 ± 0.0131
OpenFE	**0.7452 ± 0.0908**	**0.0072 ± 0.0002**	**0.0928 ± 0.0032**	**2.9993 ± 0.2243**	23.4522 ± 1.7801	**0.8776 ± 0.0794**	0.7449 ± 0.0606	0.7364 ± 0.0381	**1.6234 ± 0.3418**	**0.1493 ± 0.0071**
LinFE	0.7581 ± 0.1460	0.0078 ± 0.0003	0.1040 ± 0.0038	3.0729 ± 0.2821	23.6191 ± 2.3621	0.8802 ± 0.0744	**0.6965 ± 0.0377**	**0.7294 ± 0.0529**	1.6950 ± 0.5034	0.1619 ± 0.0167

Table 16: Dataset-level RMSE for fixed LightGBM downstream. Each entry reports mean ± std over random seeds. Bold indicates the best method and underline indicates the second-best method within each dataset. The table is split into two blocks, each showing up to 10 datasets.

Method	AB	AU	BI	BO	CA	CH	CO	CP	CPU	EN
Raw (−)	2.1448 ± 0.1003	2.9006 ± 0.4226	45.1056 ± 0.8046	1.7204 ± 0.8019	0.1336 ± 0.0030	0.5674 ± 0.0299	4.5527 ± 0.5545	2.3919 ± 0.1693	2.9133 ± 0.2629	0.4100 ± 0.0252
AutoFeat	2.1357 ± 0.0755	2.9635 ± 0.3339	**43.7265 ± 0.7807**	**1.6355 ± 0.7461**	0.1315 ± 0.0014	0.5674 ± 0.0299	4.5184 ± 0.6658	2.3717 ± 0.1982	2.8403 ± 0.2120	0.4287 ± 0.0650
OpenFE	**2.0947 ± 0.0894**	2.9109 ± 0.4143	60.1957 ± 4.7225	1.7381 ± 0.7938	**0.1294 ± 0.0011**	**0.5529 ± 0.0300**	**4.5119 ± 0.5014**	**2.3095 ± 0.1482**	**2.7916 ± 0.1640**	0.9500 ± 0.2532
LinFE	2.1336 ± 0.0556	**2.7403 ± 0.1928**	45.0591 ± 0.7886	1.7039 ± 0.7566	0.1320 ± 0.0023	0.5574 ± 0.0461	4.6729 ± 0.7475	2.3839 ± 0.1494	2.8315 ± 0.1546	**0.3497 ± 0.0445**

Method	FR	GR	KI	LI	MO	QS	SE	ST	TE	VI
Raw (−)	0.8683 ± 0.1589	0.0082 ± 0.0002	0.1322 ± 0.0035	3.0963 ± 0.1915	24.4748 ± 2.0899	0.9135 ± 0.0901	0.7367 ± 0.0303	0.8671 ± 0.1072	1.7380 ± 0.2238	0.4516 ± 0.0976
AutoFeat	0.9059 ± 0.1634	0.0072 ± 0.0002	0.1025 ± 0.0033	**3.0845 ± 0.2532**	**23.9012 ± 1.9890**	0.9350 ± 0.0832	**0.7229 ± 0.0260**	0.7568 ± 0.0406	1.8506 ± 0.3384	0.1062 ± 0.0285
OpenFE	0.8827 ± 0.1498	**0.0067 ± 0.0001**	**0.0963 ± 0.0038**	3.1035 ± 0.1702	24.3791 ± 2.2414	0.9289 ± 0.0841	0.7411 ± 0.0324	**0.7252 ± 0.0633**	1.7811 ± 0.2904	**0.0653 ± 0.0215**
LinFE	**0.8341 ± 0.1710**	0.0080 ± 0.0005	0.0994 ± 0.0019	3.1016 ± 0.2406	25.3674 ± 2.3464	**0.9052 ± 0.0627**	0.7282 ± 0.0461	0.8242 ± 0.0573	**1.7287 ± 0.1998**	0.0670 ± 0.0161

Table 17: Dataset-level RMSE for fixed Random Forest downstream. Each entry reports mean ± std over random seeds. Bold indicates the best method and underline indicates the second-best method within each dataset. The table is split into two blocks, each showing up to 10 datasets.

Method	AB	AU	BI	BO	CA	CH	CO	CP	CPU	EN
Raw (−)	2.1616 ± 0.0851	3.2344 ± 0.3116	46.5881 ± 1.1631	1.0404 ± 0.9699	0.1447 ± 0.0030	0.5202 ± 0.0294	**5.2504 ± 0.5606**	2.4911 ± 0.1291	2.9183 ± 0.1205	0.4337 ± 0.0390
AutoFeat	2.0928 ± 0.0723	3.2937 ± 0.2780	**45.7546 ± 1.4609**	1.0472 ± 0.9383	0.1403 ± 0.0016	0.5202 ± 0.0294	5.2645 ± 0.5840	2.4388 ± 0.0874	2.8683 ± 0.0846	0.4512 ± 0.0349
OpenFE	**2.0505 ± 0.0708**	3.2270 ± 0.3046	75.8377 ± 10.6935	1.0747 ± 0.9532	**0.1366 ± 0.0009**	**0.5201 ± 0.0307**	5.2819 ± 0.5616	**2.4102 ± 0.0537**	**2.8430 ± 0.0662**	1.0343 ± 0.2203
LinFE	2.0876 ± 0.0486	**2.5875 ± 0.4788**	46.7085 ± 1.3590	**1.0155 ± 0.9542**	0.1385 ± 0.0025	0.5302 ± 0.0478	5.4107 ± 0.6116	2.4329 ± 0.0777	2.9073 ± 0.1039	**0.4187 ± 0.0488**

Method	FR	GR	KI	LI	MO	QS	SE	ST	TE	VI
Raw (−)	0.8117 ± 0.0454	0.0119 ± 0.0001	0.1402 ± 0.0032	3.0073 ± 0.1735	24.9291 ± 2.7344	0.9061 ± 0.0958	0.7097 ± 0.0259	0.8198 ± 0.1021	1.3089 ± 0.3930	0.0371 ± 0.0099
AutoFeat	0.8614 ± 0.0950	0.0104 ± 0.0001	0.1255 ± 0.0025	2.9890 ± 0.1534	24.5706 ± 2.2461	**0.8950 ± 0.0834**	0.7149 ± 0.0371	**0.7506 ± 0.0447**	1.2448 ± 0.2691	0.0607 ± 0.0297
OpenFE	**0.8074 ± 0.0481**	**0.0101 ± 0.0001**	**0.1150 ± 0.0033**	2.9877 ± 0.2144	24.9506 ± 2.7776	0.9103 ± 0.0983	0.7236 ± 0.0521	0.8136 ± 0.0841	1.3012 ± 0.3972	0.0371 ± 0.0061
LinFE	0.8243 ± 0.1965	0.0111 ± 0.0004	0.1158 ± 0.0023	**2.9609 ± 0.1875**	**24.2638 ± 2.4841**	0.8975 ± 0.0852	**0.7074 ± 0.0415**	0.7894 ± 0.0693	**1.1328 ± 0.3103**	**0.0311 ± 0.0122**

Appendix F. Proof of the Theory

Proof 1. Decomposition of Regret: For any fixed downstream class $\mathcal{H}_D \in \mathcal{D}$ and a feature $\hat{g}$ selected on S_n, we decompose the regret relative to the downstream-optimal feature g_D^* by bridging through the population proxy utility $\Delta(\cdot, \mathcal{H}_P)$:

$$\begin{aligned}\text{Regret}(\mathcal{H}_D) &= \Delta(g_D^*, \mathcal{H}_D) - \Delta(\hat{g}, \mathcal{H}_D) \\ &= \underbrace{\Delta(g_D^*, \mathcal{H}_D) - \Delta(g_D^*, \mathcal{H}_P)}_{(A)} + \underbrace{\Delta(g_D^*, \mathcal{H}_P) - \Delta(\hat{g}, \mathcal{H}_P)}_{(B)} + \underbrace{\Delta(\hat{g}, \mathcal{H}_P) - \Delta(\hat{g}, \mathcal{H}_D)}_{(C)}\end{aligned}$$

2. Bounding the Inductive Mismatch (A + C): By applying the definition of the supremum over the feature set $\mathcal{G}$, we bound the transfer loss:

$$\begin{aligned}(A) + (C) &\leq |\Delta(g_D^*, \mathcal{H}_D) - \Delta(g_D^*, \mathcal{H}_P)| + |\Delta(\hat{g}, \mathcal{H}_P) - \Delta(\hat{g}, \mathcal{H}_D)| \\ &\leq 2 \sup_{g \in \mathcal{G}} |\Delta(g, \mathcal{H}_P) - \Delta(g, \mathcal{H}_D)| = \text{dist}_{\mathcal{G}}(\mathcal{H}_P, \mathcal{H}_D)\end{aligned}$$

3. Bounding Selection Overfitting (B): Let $g_P^* = \arg\max_{g \in \mathcal{G}} \Delta(g, \mathcal{H}_P)$. Since $\Delta(g_P^*, \mathcal{H}_P) \geq \Delta(g_D^*, \mathcal{H}_P)$, then $(B) \leq \Delta(g_P^*, \mathcal{H}_P) - \Delta(\hat{g}, \mathcal{H}_P)$. Introducing the empirical proxy gain $\widehat{\Delta}(g, \mathcal{H}_P, S_n)$:

$$\begin{aligned}(B) \leq &\left[\Delta(g_P^*, \mathcal{H}_P) - \widehat{\Delta}(g_P^*, \mathcal{H}_P, S_n)\right] + \left[\widehat{\Delta}(g_P^*, \mathcal{H}_P, S_n) - \widehat{\Delta}(\hat{g}, \mathcal{H}_P, S_n)\right] \\ &+ \left[\widehat{\Delta}(\hat{g}, \mathcal{H}_P, S_n) - \Delta(\hat{g}, \mathcal{H}_P)\right]\end{aligned}$$

Because $\hat{g}$ maximizes $\widehat{\Delta}$ by construction, the middle term is ≤ 0. Thus:

$$(B) \leq 2 \sup_{g \in \mathcal{G}} |\widehat{\Delta}(g, \mathcal{H}_P, S_n) - \Delta(g, \mathcal{H}_P)|$$

4. Concentration and Complexity: Let $\mathcal{F} = \{(x, y) \mapsto \ell(y, h(B(x), g(x))) : h \in \mathcal{H}_P, g \in \mathcal{G}\}$. Define the uniform deviation $\Phi(S_n) = \sup_{f \in \mathcal{F}} |\mathbb{E}_P[f] - \mathbb{E}_{S_n}[f]|$.

Assuming the loss ℓ is bounded in $[0, C]$, changing a single sample z_i modifies $\Phi(S_n)$ by at most C/n. By McDiarmid's inequality, for any $\delta > 0$, with probability at least $1 - \delta$:

$$\Phi(S_n) \leq \mathbb{E}_{S_n}[\Phi(S_n)] + C\sqrt{\frac{\log(1/\delta)}{2n}}$$

By standard symmetrization, $\mathbb{E}_{S_n}[\Phi(S_n)] \leq 2\mathfrak{R}_n(\mathcal{F})$. Since Term $(B) \leq 2\Phi(S_n)$ (noting that gain estimation error is bounded by risk estimation error), we combine these results to find that with probability $1 - \delta$:

$$(B) \leq 4\mathfrak{R}_n(\mathcal{F}) + 2C\sqrt{\frac{\log(1/\delta)}{2n}}$$

5. Final Assembly: Substituting the high-probability bound for Term (B) back into the regret decomposition and taking the expectation over $\mathcal{H}_D \sim \mathcal{D}$ for the inductive mismatch terms:

$$\mathcal{R}_{univ}(\mathcal{H}_P) \leq 4\mathfrak{R}_n(\ell \circ \mathcal{H}_P \circ \mathcal{G}) + \mathbb{E}_{\mathcal{H}_D \sim \mathcal{D}}\left[\text{dist}_{\mathcal{G}}(\mathcal{H}_P, \mathcal{H}_D)\right] + 2C\sqrt{\frac{\log(1/\delta)}{2n}}$$

■