CO-EVOLVING LLM AGENTS AND BAYESIAN OPTIMIZERS FOR AUTOMATED FEATURE ENGINEERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Automated feature engineering (AutoFE) involves searching a combinatorial space of compositional transformations under expensive downstream evaluations, making efficient exploration and effective reuse of search experience essential. We introduce CoEvolve, a collaborative evolution framework that couples an LLM agent for local exploration with a global optimizer for selecting promising feature set anchors. Starting from an anchor, the agent invokes FE-MDP, a surrogate network that predicts one-step performance gains conditioned on the current feature set and candidate action, to guide exploratory beam expansion and collect local search experience. A dual-memory mechanism distills this experience into Conditional Agent Memory (CAM), which captures context-specific evidence about effective and ineffective transformations, and Unconditional Agent Memory (UAM), which accumulates reusable knowledge across feature sets. Together, these memories guide the agent in proposing candidate transformations for downstream evaluation. To coordinate global exploration, we introduce the Deep Set Global Optimizer (DSGO), which combines a permutation-invariant Set Transformer encoder with a Bayesian linear regression posterior to estimate downstream performance and predictive uncertainty. DSGO selects promising anchors through expected improvement, directing local exploration toward promising regions of the search space. As newly evaluated candidates expand the search pool, the surrogate models and agent memories are iteratively updated, allowing global optimization and local exploration to inform one another. Experiments on public tabular benchmarks demonstrate that CoEvolve outperforms existing feature selection and generation methods, achieving average relative performance improvements of 21.68% over the raw-feature baseline and 3.59% over the strongest competing baseline on each dataset.

1 INTRODUCTION

Feature engineering shapes the representation through which a predictive model accesses tabular data. Explicit transformations can expose useful interactions and more accessible predictive structure to better support a downstream learning task. Automated feature engineering (AutoFE) aims to carry out this process algorithmically, reducing the reliance on manual experimentation and domain expertise. AutoFE requires generating effective transformations while controlling the number of candidates that must be evaluated. Expansion-and-selection methods such as AutoFeat and OpenFE address this challenge through nonlinear feature construction and selection, and incremental feature evaluation with pruning, respectively(Horn et al., 2019; Zhang et al., 2023). Search-based methods (Xiao et al., 2024; Gao et al., 2025; Gong et al., 2025) instead explore transformations adaptively, using techniques such as reinforcement learning or evolutionary optimization. Nevertheless, composing transformations over an evolving feature space creates a large combinatorial search problem, and evaluating each candidate by training a downstream model remains costly.

Efforts were made to reduce expensive downstream evaluations through learned representations and predictive guidance that illustrates a broader shift from exhaustively constructing features toward learning where to search. LFE (Nargesian et al., 2017) leverages previous feature engineering experiences to recommend useful transformations without explicitly evaluating every candidate on a new task. DIFER(Zhu et al., 2022) combines evolutionary search with an encoder–predictor–decoder. Large language models (LLMs) provide another complementary source of guidance by proposing

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.

transformations informed by feature semantics and previously evaluated examples. CAAFE uses language models for context-aware feature construction (Hollmann et al., 2023), while ELLM-FT combines reinforcement-based data collection with evolutionary populations and LLM generation (Gong et al., 2025). More recently, LLM-FE treats feature engineering as evolutionary program search, and PromptFE iteratively refines feature construction using semantic descriptions and performance feedback (Abhyankar et al., 2026; Zou et al., 2026). These advances motivate a further question: how should a feature-engineering agent coordinate global exploration, decisions within a particular feature set, and the reuse of accumulated experience?

Answering this question requires recognizing two properties of AutoFE. First, transformation utility is contextual: an operation that improves one feature set may become redundant or unhelpful when other features are present. Second, under a fixed downstream learner and evaluation protocol, different transformation histories that yield identical output features define the same predictive candidate, treating them as distinct prediction targets can fragment evaluation evidence. These observations motivate searching over output feature sets while learning both where to begin exploration and which transformations to apply within each set. In addition, the search must connect reasoning to measured outcomes. A promising global region, an apparently useful local action, and a plausible linguistic explanation provide different kinds of evidence. An effective system should allow them to inform one another while retaining actual model evaluation as the source of supervision.

To address these requirements, we introduce **CoEvolve**, a collaborative evolution framework that couples Bayesian global optimization with memory-guided local exploration. CoEvolve represents each candidate by its output feature set and associated expressions, merging candidates with identical canonical outputs. Because the utility of subsequent transformations depends on the current feature set, anchor selection determines the context in which local exploration takes place. We therefore select anchors from a global candidate pool, using predicted feature-set quality and uncertainty to guide the allocation of local search.

The *Deep Set Global Optimizer (DSGO)* supports this selection through a Set Transformer encoder with permutation-invariant pooling (Lee et al., 2019). This architecture accommodates the unordered nature of feature sets while modeling interactions among their constituent features, whose joint predictive value may not be captured by evaluating features independently. A neural prediction head provides supervision from observed cross-validation scores, enabling the encoder to learn representations relevant to downstream performance. Conditioned on these learned representations, a DNGO-style Bayesian linear regression posterior supplies the predictive mean and uncertainty used for acquisition (Snoek et al., 2015). This design combines flexible representation learning with tractable Bayesian inference over the output weights. Expected improvement then prioritizes candidate anchors according to their potential to outperform the best observed feature set, accounting for both predicted performance and uncertainty. The selected anchors provide starting contexts for the LLM agent, linking global performance modeling to targeted local exploration.

Around each anchor, CoEvolve models sequential feature transformations as a Markov decision process. A learned decision model, *FE-MDP*, combines a Deep Sets representation of the current feature set (Zaheer et al., 2017) with a candidate action to predict its immediate performance gain and guide exploratory beam expansion. However, a finite beam guided by learned gain estimates may favor locally attractive transformations while overlooking promising compositions beyond the explored branches. Moreover, its candidate rankings remain subject to prediction error. CoEvolve therefore uses the resulting local experience to inform a dual-memory mechanism, rather than relying solely on beam rankings to determine subsequent search directions. *Conditional Agent Memory (CAM)* summarizes evidence about effective and ineffective transformations in the current context, while *Unconditional Agent Memory (UAM)* consolidates reusable transformation knowledge across feature sets and search rounds. By integrating the current anchor with both memories, the LLM agent synthesizes higher-level search directions that extend the exploration informed by FE-MDP. These directions are instantiated as concrete candidate feature sets and evaluated through downstream cross-validation. New evaluations expand the candidate and observation pools and support iterative updates to both learned models and agent memories. This feedback loop couples the optimizer's evolving view of promising feature sets with the agent's accumulated knowledge of useful transformations, allowing global anchor selection and local exploration to adapt together.

Our contributions are threefold. First, We formulate AutoFE as sequential search over output feature sets, merging candidates with identical canonical outputs and explicitly conditioning transformation

decisions on the current feature set. Second, We develop CoEvolve, which integrates Bayesian anchor selection, learned local gain prediction, and LLM generation guided by conditional and unconditional memories within a shared evaluation-and-update loop. Third, We evaluate CoEvolve on eight public tabular datasets spanning classification and regression. In the primary five-fold cross-validation comparison against 16 feature selection and generation baselines, CoEvolve ranks first on seven datasets and achieves average relative improvements of 21.68% over raw features and 3.59% over the strongest competing baseline on each dataset.

2 Related Work

Learned representations for feature search. LFE learns transformation recommendations from previous tasks using classifiers over feature statistics (Nargesian et al., 2017). DIFER jointly learns an encoder, performance predictor, and decoder to optimize feature expressions through gradient-guided evolution (Zhu et al., 2022). GAINS similarly learns an encoder–evaluator–decoder for feature subsets, starting from high-performing historical subsets and optimizing their embeddings before decoding new selections (Xiao et al., 2023). CoEvolve shares the use of learned performance guidance but assigns it two distinct targets: the absolute performance of an output feature set and the incremental benefit of an action in its current context. Its embeddings support Bayesian acquisition over explicit candidate sets, while discrete transformations and LLM proposals generate new candidates. Thus, its search does not require decoding optimized latent vectors into expressions or subsets.

LLM-based feature engineering. CAAFE generates executable feature transformations from dataset context and iteratively evaluates their predictive utility (Hollmann et al., 2023). ELLM-FT combines reinforcement-based collection of transformation examples with evolutionary populations and LLM generation (Gong et al., 2025). LLM-FE formulates feature engineering as evolutionary program search, maintaining multiple populations of evaluated programs to guide subsequent in-context generation (Abhyankar et al., 2026). PromptFE generates canonical feature expressions and updates its prompts using ranked examples and evaluation feedback (Zou et al., 2026). These methods already reuse evaluated candidates and support iterative refinement. CoEvolve's distinction is how this feedback is organized and used to allocate search: DSGO selects global anchors using predicted performance and uncertainty, FE-MDP explores transitions within each anchor, and the LLM uses CAM and UAM to synthesize further search directions from the accumulated evidence.

Memory and experience-guided exploration. Reflexion stores verbal interpretations of environment feedback to inform later agent decisions without updating language-model weights (Shinn et al., 2023). Within AutoFE, the recent TopoFE preprint maintains per-island prompt memories and elite archives, and learns a transfer topology to coordinate exploration across feature families (Li & Ramakrishnan, 2026). SIGMA encodes recent search history through the features exposed to the prompt and restrictions on repeatedly unsuccessful operations (Zheng et al., 2026). CoEvolve organizes experience according to feature-set context: CAM analyzes outcomes around a particular anchor, while UAM extracts reusable heuristics across anchors and rounds. The same measured evaluations also supervise global set-performance and local action-gain models. This coupling connects textual experience reuse with learned search guidance, allowing memory-informed proposals and numerical models to evolve through a shared feedback process.

3 Methodology

We introduce *CoEvolve*, a framework that couples Deep Set Global Optimization (DSGO) with an LLM feature engineering agent. DSGO selects promising feature sets as search conditions, while a learned feature engineering MDP (FE-MDP) explores their local neighborhoods and constructs empirical reasoning evidence. The LLM converts this evidence into a conditional chain-of-thought (CoT) context and uses it, together with accumulated memory, to select further transformations. Their evaluated outcomes update the global optimizer and the FE-MDP, establishing a reciprocal feedback loop between global acquisition, local reasoning, and empirical evaluation. Figure 1 illustrates the overall architecture.

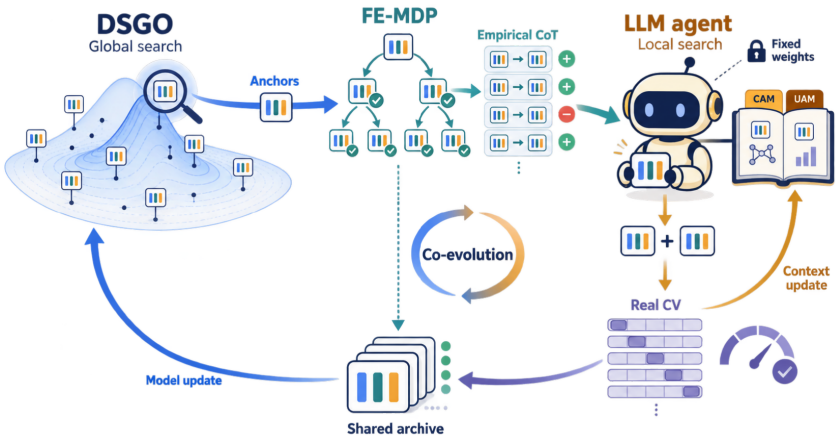


Figure 1: Overview of CoEvolve. DSGO selects promising anchors, then FE-MDP guides local exploration that composes evidences in CAM and UAM, and the LLM agent proposes feature engineering directions using CAM and UAM.

Let $\mathcal{D} = (X, y)$ denote a tabular learning task and S a retained set of K features. Each feature $f \in S$ is a compositional expression e_f over the original inputs. Each feature expression specifies its computational operations and dependencies. CoEvolve identifies a candidate by its canonicalized set of output feature expressions, independently of the search trajectory that produced it. All candidates are evaluated using a fixed downstream model and evaluation protocol. Consequently, candidates with the same canonical output feature set share the same cross-validation score $J(S)$, regardless of their generation histories. We express all scores so that higher values indicate better performance. Given a budget of true CV evaluations, CoEvolve seeks a high-scoring output set by alternating global anchor selection and local exploration. We use t for the outer search round and ℓ for the remaining depth of a local rollout.

3.1 Deep Set Global Optimizer

Learning feature-set performance. DSGO maintains a global candidate pool $\mathcal{P}_t$ and an evaluated pool $\mathcal{V}_t = \{(S_i, y_i)\}_{i=1}^{N_t}$, where $y_i = J(S_i)$. The global pool may contain candidates whose CV scores have not yet been measured. Let $\mathbf{x}_f \in \mathbb{R}^{d_x}$ denote the vector representation of feature expression e_f. A Set Transformer models interactions among these representation vectors, followed by an invariant pool (Lee et al., 2019):

$$\mathbf{h}_\theta(S) = \mathrm{Pool}_{\mathrm{inv}}\left(\mathrm{ST}_\theta(\{\mathbf{x}_f : f \in S\})\right). \tag{1}$$

This representation respects the unordered nature of a feature set while capable of modeling feature interactions. A neural head g_ψ supplies supervision from observed CV performance:

$$\mathcal{L}_{\mathrm{global}}(\theta, \psi) = \frac{1}{N_t} \sum_{i=1}^{N_t} \left(g_\psi(\mathbf{h}_\theta(S_i)) - y_i\right)^2. \tag{2}$$

Bayesian prediction and acquisition. The Set Transformer captures interactions among features, and invariant pooling summarizes them into a set representation. The neural prediction head maps this representation to CV performance during training on the evaluated pool. DSGO then fits a DNGO-style Bayesian linear regression posterior over the head's hidden activations (Snoek et al., 2015). This posterior provides a predicted CV score and associated uncertainty for each candidate, conditional on the learned neural representation.

For anchor selection, DSGO aggregates predictive moments across sampled prior and noise precisions and uses the resulting moment-matched Gaussian approximation to compute expected improvement (EI). EI ranks feature sets in the global candidate pool $\mathcal{P}_t$, including previously evaluated candidates, by their potential to improve on the best observed score. The highest-ranked sets are selected as anchors for local exploration. Selected anchors without cached scores receive CV evaluation before local search, and the selected batch remains fixed for the round. By combining predicted

performance with uncertainty, this procedure allocates local exploration to promising feature sets while retaining opportunities to explore uncertain candidates. Acquisition is computed over explicit candidates using their learned representations; it serves to select search starting points rather than predict the gains of subsequent local feature transformations.

3.2 LLM LOCAL SEARCH WITH FE-MDP, CAM, AND UAM

Feature transformations as an MDP. CoEvolve organizes local feature transformations as an MDP. To control the large action space induced by individual features and feature interactions, it groups the current set S into m clusters, denoted by $\mathcal{C}(S) = \{C_1(S), \ldots, C_m(S)\}$. A unary action selects one cluster, and a binary action selects an ordered pair of clusters. With fewer clusters than features, this reduces operand choices while enabling each action to generate multiple features for subsequent selection. Clustering, cluster indexing, and operator execution are fixed functions of the current set and task configuration, ensuring consistent actions for the same state. We define the finite-horizon state as $s = (S, \ell)$, where ℓ denotes the remaining local-search depth.

The action vocabulary contains unary actions (u, j) and ordered binary actions (b, j, k). Their generated features are

$$G(S,a) = \begin{cases} \{u(f) : f \in C_j(S)\}, & a = (u,j), \\ \{b(f,g) : f \in C_j(S),\ g \in C_k(S)\}, & a = (b,j,k). \end{cases} \tag{3}$$

Thus, a unary operation acts on every feature in its selected cluster, and a binary operation acts on the full Cartesian product of two selected clusters. Operand order is retained for noncommutative operators. An action is admitted only when its full generated set satisfies the prescribed operator-domain and operand constraints. New features are merged with the current feature set through deduplication and K best selection:

$$T(S,a) = \mathrm{KBest}_K\left(S \cup G(S,a)\right), \qquad r(S,a) = J(T(S,a)) - J(S). \tag{4}$$

The transition maps (S, ℓ) to $(T(S,a), \ell - 1)$. It is computed by executing the transformation and selection rules. A rollout terminates at zero remaining depth, an exhausted budget, or a state with no legal actions. Rewards are computed from measured CV scores, with positive values indicating improvement. Throughout the search, the transformation and evaluation rules remain fixed, while updates to the learned models and agent memories guide which actions to explore next.

Conditional one-step gain prediction. FE-MDP learns the immediate reward from a buffer $\mathcal{R}_t$ of evaluated one-step transitions (S, a, S', r), where $S' = T(S,a)$. A Deep Sets encoder (Zaheer et al., 2017) represents the current feature set, and a two-layer MLP combines it with the one-hot action vector:

$$\mathbf{u}_\omega(S) = \rho_\omega\left(\sum_{f \in S} \varphi_\omega(\mathbf{x}_f)\right), \tag{5}$$

$$\widehat{r}_\omega(S,a) = \mathbf{w}_2^\top \nu\left(W_1[\mathbf{u}_\omega(S); \mathrm{onehot}(a)] + \mathbf{b}_1\right) + b_2, \tag{6}$$

$$\mathcal{L}_{\mathrm{local}}(\omega) = \frac{1}{|\mathcal{R}_t|} \sum_{(S,a,S',r) \in \mathcal{R}_t} \left(\widehat{r}_\omega(S,a) - r\right)^2. \tag{7}$$

Here ν is an activation function and $[\cdot;\cdot]$ denotes concatenation. The finite action encoding preserves operator identity and ordered cluster indices. Conditioning on S allows the same action to receive different predictions in different feature contexts. This is immediate-gain regression, with no Bellman bootstrap or long-horizon value target. Appendix B proves that feature engineering effects are conditioned.

Exploratory beam expansion. From an anchor S^{a}, the agent invokes FE-MDP to guide a beam of width B over at most H local steps. At each retained state, action proposals use an ϵ-greedy rule:

$$\pi_\epsilon(a \mid S) = (1-\epsilon)\mathbb{I}[a = a^\star(S)] + \frac{\epsilon}{|\mathcal{A}(S)|}, \qquad a^\star(S) \in \arg\max_{a \in \mathcal{A}(S)} \widehat{r}_\omega(S,a), \tag{8}$$

where $\mathcal{A}(S)$ is the legal-action set and ties follow a fixed rule. Selected actions generate successor featuresets through T. Successfully evaluated successors are deduplicated at each depth, and the

beam retains those with the highest measured CV scores. Only measured score differences between a state and its immediate successor are stored as rewards in $\mathcal{R}_t$. Unevaluated anchors receive CV evaluation before local exploration, and cached scores are reused under the same evaluation protocol.

Conditional and unconditional memory. A finite beam can miss useful compositions, and its action selection depends on imperfect gain estimates. Cluster-wise operations further restrict search granularity by applying an operator to an entire cluster or all feature pairs between two clusters. Features within a cluster may respond differently to the same transformation, so an action can jointly generate useful, redundant, and uninformative features. Instead, these evaluated outcomes play a role analogous to intermediate steps in chain-of-thought (CoT) reasoning (Wei et al., 2022). Candidate transformations instantiate hypotheses, and measured CV changes provide evidence about their consequences. We therefore view FE-MDP-guided exploration as an external source of CoT-like evidence, which are summerized as anchor-conditioned evidence in CAM and reusable heuristics in UAM to inform subsequent LLM reasoning.

Given an anchor and its evaluated local candidates, CAM uses the LLM to summarize which feature changes improved or reduced CV performance in the current context. This summary is stored with the anchor and helps the agent decide which directions to explore next. UAM consolidates reports across feature sets and outer rounds into reusable transformation heuristics. The agent retrieves a compact selection u_t^{ret} from this memory and proposes directions conditioned on both scopes:

$$d \sim \pi_{\mathrm{LLM}}\left(\cdot \mid S^{\mathrm{a}}, c_t(S^{\mathrm{a}}), u_t^{\mathrm{ret}}\right). \tag{9}$$

Here, π_{LLM} denotes the LLM agent's conditional proposal policy: given the anchor, CAM analysis, retrieved UAM guidance, and available action catalog, it generates a candidate search direction d. Each LLM proposal selects a valid action from the same operator and feature-cluster vocabulary used in beam expansion. The action is applied to the anchor, and the resulting feature set is evaluated by CV. Successfully evaluated candidates are added to the search pool alongside the beam-generated candidates. UAM provides reusable guidance, but whether a suggested transformation helps must be verified in the current feature context.

3.3 Coevolution of Global Optimization and Local Exploration

Shared observations, distinct learning targets. Initialization provides legal seed candidates, a nonempty CV pool, and evaluated transformations for the initial FE-MDP fit. At the beginning of each outer round, DSGO is retrained from scratch on the current evaluated pool, rebuilds its Bayesian posterior, and selects anchors from the global pool. Around each anchor, beam exploration supplies measured local evidence. CAM summarizes these outcomes, and UAM incorporates the resulting reports before the LLM proposes additional candidates. Successfully measured results update the shared pools:

$$\mathcal{P}_{t+1} = \mathcal{P}_t \cup \mathcal{P}_t^{\mathrm{beam}} \cup \mathcal{P}_t^{\mathrm{LLM}}, \qquad \mathcal{V}_{t+1} = \mathcal{V}_t \cup \Delta\mathcal{V}_t. \tag{10}$$

Here, $\mathcal{P}_t^{\mathrm{beam}}$ and $\mathcal{P}_t^{\mathrm{LLM}}$ contain candidates registered through successful local evaluations, and $\Delta\mathcal{V}_t$ contains newly measured feature-set scores. Candidates are merged by their canonical output-set identity, without requiring their generation histories.

At the end of each round, FE-MDP continues supervised training on observed one-step performance gains using Eq. (7). The expanded CV pool is used to retrain DSGO at the beginning of the next round using Eq. (2). All evaluated sets are then re-encoded, and the Bayesian posterior is rebuilt before anchor selection.

UAM also evolves as successive rounds explore different anchors. Each beam exploration provides opportunities to observe useful and ineffective transformations, and UAM is updated as new contextual reports become available. By consolidating these reports with earlier experience, UAM accumulates transformation knowledge across feature sets and search rounds. This evolving memory supplies context for subsequent LLM proposals, whose usefulness is verified through CV in the current feature context. CoEvolve thus couples global anchor selection, local exploration, and memory updates through shared evaluation feedback, without requiring LLM weight fine-tuning.

Overall search procedure. CoEvolve begins with seed exploration to initialize the global candidate pool $\mathcal{P}_0$, the evaluated pool $\mathcal{V}_0$, and the agent memories, and to obtain measured one-step gains

for the initial FE-MDP fit. At round t, DSGO is trained on $\mathcal{V}_t$ and its Bayesian posterior is rebuilt. Expected improvement selects up to n_a anchors from $\mathcal{P}_t$. Any selected anchor without a cached score is evaluated before local exploration, and its score is added to the evaluated pool.

FE-MDP then guides beam exploration around each anchor. Selected successors receive CV evaluation or reuse cached scores, and measured performance determines beam retention. CAM summarizes the resulting anchor-specific observations, while UAM consolidates these reports with knowledge accumulated across earlier anchors and rounds. Conditioned on the anchor, CAM, UAM, and the available action catalog, the LLM proposes additional transformations whose resulting feature sets are assessed by CV.

Successfully evaluated candidates from both beam exploration and LLM proposals expand the global pool, including candidates not retained in the beam. Their measured scores update the evaluated pool. These additions yield $\mathcal{P}_{t+1}$ and $\mathcal{V}_{t+1}$ according to Eq. (10), with duplicates merged by canonical output-set identity. FE-MDP continues training on the accumulated measured one-step gains, while the expanded $\mathcal{V}_{t+1}$ supports DSGO training at the start of the next round. After $T_{\max}$ rounds, CoEvolve returns the best evaluated feature set:

$$(\widehat{S}, \widehat{y}) \in \arg \max_{(S,y)\in\mathcal{V}_{T_{\max}}} y. \tag{11}$$

Local exploration thus supplies new candidates and observations for global optimization, while global anchor selection provides new contexts for local exploration. As successive beam explorations produce further contextual reports, UAM evolves alongside the numerical models and informs subsequent LLM proposals. The summarized CoEvolve search procedure is available in Appendix C.

4 Experiments

We evaluate CoEvolve through three complementary experiments: a comparison with feature selection and generation baselines, an analysis of performance across CoEvolve rounds, and ablations of the LLM agent and DSGO. During CoEvolve's search and optimization, candidate feature sets are assessed using 5-fold cross-validation (CV) on the training partition. After search, we select the feature set with the highest training CV score and fix its feature transformations. For the main results, this selected feature set is evaluated using a separate 5-fold CV procedure within the test partition, and we report the mean score across the five folds. An additional comparison using test evaluation without cross-validation is reported in Appendix A.

4.1 Experimental Setup

Datasets. We consider eight tabular benchmarks appearing in prior feature engineering and selection studies (Gong et al., 2025; Ying et al., 2024). OpenML-586, OpenML-589, OpenML-607, and OpenML-620 are regression tasks; UCI Credit, SpamBase, svmguide3, and German Credit are classification tasks. Table 1 reports the sample and input-feature counts for the supplied dataset versions. OpenML datasets appear first, in ascending identifier order, followed by the four classification datasets.

Baselines. We compare against Raw, which uses the original features, and 16 named baselines. Feature selection baselines include K-Best, mRMR (Peng et al., 2005), Lasso (Tibshirani, 1996), LassoNet (Lemhadri et al., 2021), RFE (Guyon et al., 2002), GFS (Fan et al., 2021), MCDM (Hashemi et al., 2022), MARLFS (Liu et al., 2019), SARLFS (Liu et al., 2021), and VTFS (Ying et al., 2024). We also include AFT (autofeat) (Horn et al., 2019), LDA (latent Dirichlet allocation) (Blei et al., 2003), ERG (feature expansion followed by selection) (Gong et al., 2025), TTG (Khurana et al., 2018), NFS (Chen et al., 2019), and ELLM-FT (Gong et al., 2025). Method aliases follow the benchmark conventions of ELLM-FT and VTFS.

4.2 Experiment 1: Comparison with Existing Methods

Table 1 shows that CoEvolve improves over Raw on all eight datasets and achieves the highest reported score on seven. Its mean per-dataset relative gains are 21.68% over Raw and 3.59%

	Regression (OpenML)				Classification			
Dataset	586	589	607	620	UCI Credit	SpamBase	svmguide3	German Credit
Samples	1,000	1,000	1,000	1,000	30,000	4,601	1,243	1,001
Features	25	25	50	25	24	57	21	24
Raw	0.5495	0.5095	0.5173	0.5101	0.8019	0.9268	0.7781	0.6488
K-Best	0.5671	0.5530	0.5334	0.5503	0.7996	0.9080	0.7702	0.6481
mRMR	0.5671	0.5528	0.5406	0.5503	0.7956	0.9191	0.7742	0.6610
Lasso	0.6067	0.5974	0.5815	0.5799	0.7794	0.9172	0.7443	0.6710
LassoNet	0.5769	0.5528	0.5464	0.5597	0.7413	0.8962	0.7887	0.6307
RFE	0.5655	0.5503	0.5512	0.5590	0.8010	0.9212	0.7890	0.6665
GFS	0.4302	0.5974	0.4250	0.4685	0.8049	0.9044	0.8177	0.6235
MCDM	0.5795	0.5543	0.5556	0.5566	0.7461	0.9011	0.7726	0.6641
MARLFS	0.5603	0.5194	0.5429	0.4664	**0.8077**	0.9094	0.7912	0.6575
SARLFS	0.5723	0.5375	0.5399	0.5727	0.8009	0.9094	0.7892	0.6378
VTFS	0.6425	<u>0.6972</u>	<u>0.6601</u>	<u>0.6200</u>	0.7943	<u>0.9513</u>	<u>0.8498</u>	<u>0.7788</u>
AFT	0.5521	0.5120	0.5085	0.5210	0.8030	0.9213	0.7947	0.6552
LDA	0.5366	0.5027	0.5188	0.4985	0.8013	0.9201	0.7776	0.6359
ERG	0.6240	0.6100	0.5290	0.5460	0.8030	0.9200	0.8190	0.6710
TTG	0.5440	0.5030	0.5180	0.5110	0.8040	0.9170	0.7920	0.6630
NFS	0.5460	0.5020	0.5180	0.5150	0.8050	0.9220	0.7920	0.6760
ELLM-FT	<u>0.6585</u>	0.6890	0.6550	0.6194	0.7952	0.9281	0.7851	0.7053
CoEvolve	**0.7279**	**0.7036**	**0.6911**	**0.6857**	<u>0.8061</u>	**0.9520**	**0.8622**	**0.7835**
Δ_{Raw} (%)	+32.46	+38.10	+33.60	+34.43	+0.52	+2.72	+10.81	+20.77
Δ_{best} (%)	+10.54	+0.92	+4.70	+10.60	−0.20	+0.07	+1.46	+0.61

Table 1: Experiment 1: five-fold cross-validation at evaluation. Bold marks the highest reported score while underline indicates the second best. The last two rows give CoEvolve's relative gains: $100(s_{\text{CoEvolve}}/s_{\text{ref}} - 1)$. The best baseline is selected separately for each dataset, excluding Raw and CoEvolve. Negative gains are retained.

over the strongest named baseline. On the four OpenML tasks, CoEvolve improves over Raw by 32.46%–38.10%, indicating substantial benefits from changing the input representation in these runs. Against stronger feature selection and generation baselines, the largest margins occur on OpenML-620 (10.60% over VTFS) and OpenML-586 (10.54% over ELLM-FT). CoEvolve also exceeds ELLM-FT on every dataset, with a mean relative gain of 6.72%.

4.3 Experiment 2: Progress Across CoEvolve Rounds

We next examine the evolution of the reported best test performance across eight coevolve rounds. Each round permits retaining up to 20 candidates; this retention limit does not specify the number of newly generated candidates or downstream evaluations. Figure 2 reproduces the recorded trajectories for OpenML-620, OpenML-607, and svmguide3. Across the three datasets, CoEvolve progressively discovers better feature sets, achieving relative improvements of 2.94%–8.09% between rounds 1 and 8. Progress occurs through intermittent gains rather than steady increases, with further improvements emerging after several rounds of unchanged performance. These delayed gains suggest that an early plateau does not necessarily indicate that the search has exhausted useful transformations. They also motivate CoEvolve's iterative search design: successive rounds select anchors using updated performance observations and guide local exploration with accumulated experience. Overall, the results demonstrate the value of continued search within the round budget.

4.4 Experiment 3: Component Ablation

Figure 3 compares CoEvolve with two variants that remove the LLM agent or DSGO. The variant without the LLM agent adapts the DNGO-LS paradigm (Yan et al., 2021): it retains Bayesian global anchor selection through DSGO and replaces the agent's local exploration mechanism with

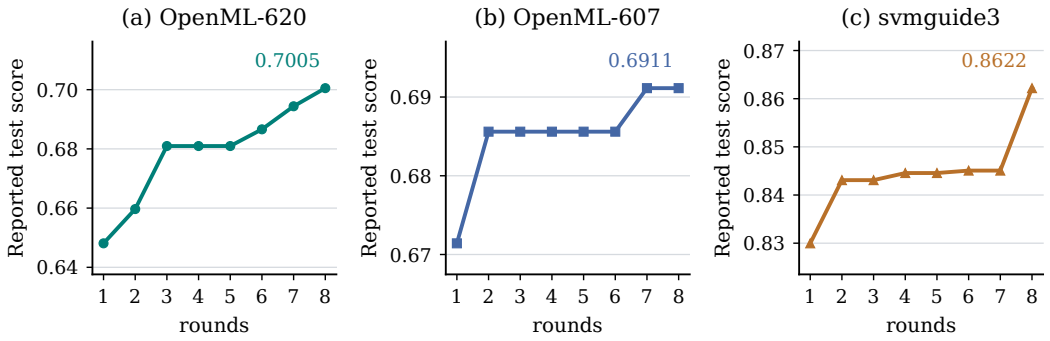


Figure 2: Best test scores across eight CoEvolve rounds, retaining up to 20 candidates per round. The curves show that continued search can discover better feature sets even after temporary performance plateaus.

neighborhood search in the learned feature-set embedding space. The variant without DSGO retains agent-guided local exploration but selects anchors according to their observed training CV scores.

CoEvolve outperforms these variants on eight and seven datasets, with mean relative improvements of 3.96% and 3.58%, respectively. The gains over DNGO-LS suggest that the proposed local exploration mechanism adds value beyond numerical prediction and embedding proximity. The gains over the variant without DSGO further support using learned global optimization to identify promising contexts for local exploration.

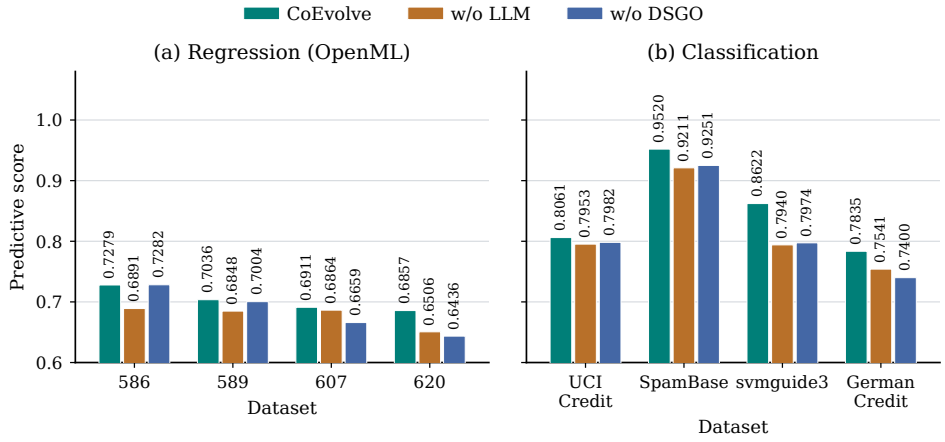


Figure 3: Ablation study: The bars compare the full CoEvolve framework with two variants: one without the LLM agent and one without DSGO. Higher scores indicate better performance.

5 Conclusion

We presented CoEvolve, an automated feature engineering framework that couples DSGO-based global optimization with memory-guided LLM local search. FE-MDP guides exploration to collect evaluated local evidence, while CAM (Conditional Agent Memory) and UAM (Unconditional Agent Memory) consolidate contextual observations and reusable experience for LLM proposals. Shared evaluation feedback updates the numerical models and agent memories, enabling global anchor selection and local exploration to evolve together. Across eight tabular benchmarks, CoEvolve achieves the highest scores on seven datasets, with mean relative improvements of 21.68% over the Raw baseline and 3.59% over the strongest baseline on each dataset. Search trajectories and ablations further support the benefits of iterative exploration and combining both search components.

AI USE STATEMENT

We used OpenAI ChatGPT/Codex to assist in coding CoEvolve's conceptual framework and methodology, including the output-feature-set formulation, the coordination of global and local search, and the roles of CAM and UAM. AI assistance also included refining mathematical formulations, providing feedback on experimental design, and analyzing and interpreting author-provided results from baseline comparisons, search trajectories, and component ablations. Additional uses included literature retrieval and citation checking, manuscript polishing, and generating tables and figures.

The LLM agent used for feature search and memory updates is a component of CoEvolve and is described in Section 3. The authors retain responsibility for methodological decisions, verification of AI-assisted content and code, and the accuracy, originality, and integrity of the reported results and final manuscript.

ETHICS STATEMENT

This work studies automated feature engineering on publicly available tabular benchmarks. Although CoEvolve aims to improve predictive performance, generated features may preserve or amplify existing biases and encode proxies for sensitive attributes. The reported performance gains do not establish fairness or suitability for consequential decisions, including credit assessment. Applying CoEvolve to sensitive data requires appropriate data-use permissions and controls over information exposed to the LLM and retained in agent memories.

REPRODUCIBILITY STATEMENT

We provide detailed algorithmic descriptions, hyperparameter settings, dataset preprocessing procedures, and additional experimental results in the appendix and accompanying source package. The implementation and configuration files required to reproduce the experiments are also included. Section 3 describes the output-feature-set representation, DSGO, FE-MDP, CAM, UAM, and their iterative update procedure. Dataset characteristics, baseline references, and score aggregation are provided in Section 4.1. Section 4 distinguishes 5-fold CV on the training partition for feature search from a separate 5-fold CV evaluation within the test partition, using the fixed feature transformations selected by training CV. Table 1 provides per-dataset results, while Figures 2 and 3 report search progress and component ablations. Appendix A presents the supplementary comparison without test-stage cross-validation.

REFERENCES

Nikhil Abhyankar, Parshin Shojaee, and Chandan K. Reddy. LLM-FE: Automated feature engineering for tabular data with LLMs as evolutionary optimizers. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=qvI35hkpOO.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. URL https://www.jmlr.org/papers/v3/blei03a.html.

Xiangning Chen, Qingwei Lin, Chuan Luo, Xudong Li, Hongyu Zhang, Yong Xu, Yingnong Dang, Kaixin Sui, Xu Zhang, Bo Qiao, Weiyi Zhang, Wei Wu, Murali Chintalapati, and Dongmei Zhang. Neural feature search: A neural architecture for automated feature engineering. In *2019 IEEE International Conference on Data Mining*, pp. 71–80, 2019. doi: 10.1109/ICDM.2019.00017. URL https://www.microsoft.com/en-us/research/publication/neural-feature-search-a-neural-architecture-for-automated-feature-engineering/.

Wei Fan, Kunpeng Liu, Hao Liu, Ahmad Hariri, Dejing Dou, and Yanjie Fu. AutoGFS: Automated group-based feature selection via interactive reinforcement learning. In *Proceedings of the 2021 SIAM International Conference on Data Mining*, pp. 342–350, 2021. doi: 10.1137/1.9781611976700.39. URL https://doi.org/10.1137/1.9781611976700.39.

Wanfu Gao, Zengyao Man, Hanlin Pan, and Kunpeng Liu. Dual-agent reinforcement learning for automated feature generation. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 5190–5198, 2025. doi: 10.24963/ijcai.2025/578.

Nanxu Gong, Chandan K. Reddy, Wangyang Ying, Haifeng Chen, and Yanjie Fu. Evolutionary large language model for automated feature transformation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(16):16844–16852, 2025. URL https://ojs.aaai.org/index.php/AAAI/article/view/33851.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002. doi: 10.1023/A:1012487302797. URL https://doi.org/10.1023/A:1012487302797.

Amin Hashemi, Mohammad Bagher Dowlatshahi, and Hossein Nezamabadi-pour. Ensemble of feature selection algorithms: A multi-criteria decision-making approach. *International Journal of Machine Learning and Cybernetics*, 13(1):49–69, 2022. doi: 10.1007/s13042-021-01347-z. URL https://link.springer.com/article/10.1007/s13042-021-01347-z.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.03403.

Franziska Horn, Robert Pack, and Michael Rieger. The autofeat Python library for automated feature engineering and selection. arXiv preprint arXiv:1901.07329, 2019. URL https://arxiv.org/abs/1901.07329. Author manuscript; ECML PKDD Workshop on Automating Data Science.

Udayan Khurana, Horst Samulowitz, and Deepak Turaga. Feature engineering for predictive modeling using reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. doi: 10.1609/aaai.v32i1.11678. URL https://ojs.aaai.org/index.php/AAAI/article/view/11678.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani. LassoNet: A neural network with feature sparsity. *Journal of Machine Learning Research*, 22(127):1–29, 2021. URL https://jmlr.org/papers/v22/20-848.html.

Sha Li and Naren Ramakrishnan. TopoFE: Topology-aware LLM-guided automated feature engineering, 2026. URL https://arxiv.org/abs/2607.23286v1. arXiv:2607.23286v1.

Kunpeng Liu, Yanjie Fu, Pengfei Wang, Le Wu, Rui Bo, and Xiaolin Li. Automating feature subspace exploration via multi-agent reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 207–215, 2019. doi: 10.1145/3292500.3330868. URL https://doi.org/10.1145/3292500.3330868.

Kunpeng Liu, Pengfei Wang, Dongjie Wang, Wan Du, Dapeng Oliver Wu, and Yanjie Fu. Efficient reinforced feature selection via early stopping traverse strategy. In *2021 IEEE International Conference on Data Mining*, pp. 399–408, 2021. doi: 10.1109/ICDM51629.2021.00051. URL https://doi.org/10.1109/ICDM51629.2021.00051.

Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B. Khalil, and Deepak Turaga. Learning feature engineering for classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 2529–2535, 2017. doi: 10.24963/ijcai.2017/352. URL https://doi.org/10.24963/ijcai.2017/352.

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. doi: 10.1109/TPAMI.2005.159. URL https://ieeexplore.ieee.org/document/1453511/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat, and Ryan P. Adams. Scalable Bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2171–2180, 2015. URL https://proceedings.mlr.press/v37/snoek15.html.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. doi: 10.1111/j.2517-6161.1996.tb02080.x. URL https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Meng Xiao, Dongjie Wang, Min Wu, Pengfei Wang, Yuanchun Zhou, and Yanjie Fu. Beyond discrete selection: Continuous embedding space optimization for generative feature selection. In *2023 IEEE International Conference on Data Mining (ICDM)*, pp. 688–697, 2023. doi: 10.1109/ICDM58522.2023.00078. URL https://doi.org/10.1109/ICDM58522.2023.00078.

Meng Xiao, Dongjie Wang, Min Wu, Kunpeng Liu, Hui Xiong, Yuanchun Zhou, and Yanjie Fu. Traceable group-wise self-optimizing feature transformation learning: A dual optimization perspective. *ACM Transactions on Knowledge Discovery from Data*, 18(4), 2024. doi: 10.1145/3638059.

Shen Yan, Kaiqiang Song, Fei Liu, and Mi Zhang. CATE: Computation-aware neural architecture encoding with transformers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11670–11681. PMLR, 2021. URL https://proceedings.mlr.press/v139/yan21c.html.

Wangyang Ying, Dongjie Wang, Haifeng Chen, and Yanjie Fu. Feature selection as deep sequential generative learning. *ACM Transactions on Knowledge Discovery from Data*, 18(9):221:1–221:21, 2024. doi: 10.1145/3687485. URL https://arxiv.org/abs/2403.03838.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html.

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. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 41880–41901, 2023. URL https://proceedings.mlr.press/v202/zhang23ay/zhang23ay.pdf.

Xuan Zheng, Kento Uchida, and Shinichi Shirakawa. SIGMA: SHAP-guided implicit-trajectory generation for metadata-free LLM-based AutoFE. arXiv preprint arXiv:2608.17948, 2026. URL https://arxiv.org/abs/2608.17948.

Guanghui Zhu, Zhuoer Xu, Chunfeng Yuan, and Yihua Huang. DIFER: Differentiable automated feature engineering. In *Proceedings of the First International Conference on Automated Machine Learning*, volume 188 of *Proceedings of Machine Learning Research*, pp. 17/1–17. PMLR, 2022. URL https://proceedings.mlr.press/v188/zhu22a.html.

Yufeng Zou, Jean Utke, Diego Klabjan, and Han Liu. PromptFE: Automated feature engineering by prompting. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 653–681, 2026. URL https://aclanthology.org/2026.eacl-long.28/.

A TEST EVALUATION WITHOUT CROSS-VALIDATION

Dataset and baseline descriptions are provided in Section 4.1. Table 2 presents the separately recorded experiment without test-stage cross-validation. CoEvolve again obtains the highest available score on all five paired datasets. Its mean relative gains are 11.57% over Raw and 4.66% over the strongest available baseline. The latter rounds to 4.7% at one decimal place and applies specifically to this five-dataset comparison. The largest relative margins over VTFS occur on OpenML-586 (11.97%) and OpenML-620 (8.94%); the classification margins are 0.25% on SpamBase and 0.49% on svmguide3. CoEvolve obtains 0.8159 on OpenML-589, again without a reported baseline. The two experiments show a consistent ordering on the paired datasets under their respective recorded evaluation procedures. The unreported ELLM-FT scores and absent uncertainty estimates limit stronger conclusions about LLM-agent superiority or statistical reliability.

	Regression (OpenML)				Classification			
Dataset	586	589	607	620	UCI Credit	SpamBase	svmguide3	German Credit
Samples	1,000	1,000	1,000	1,000	30,000	4,601	1,243	1,001
Features	25	25	50	25	24	57	21	24
Raw	0.6721	0.6739	0.6960	0.6428	**0.8073**	0.9488	0.8213	0.7197
K-Best	0.6962	0.6892	0.7120	0.6660	0.8061	0.9433	0.7952	0.7137
mRMR	0.7208	0.6906	0.7097	0.6660	0.8027	0.9477	0.7791	0.7322
Lasso	0.6972	0.7191	0.7270	0.6942	0.7784	0.9290	0.8074	0.6766
LassoNet	0.6972	0.7000	0.7090	0.6829	0.7525	0.9195	0.7881	0.5892
RFE	0.7018	0.6905	0.7130	0.6678	0.8023	0.9443	0.8197	0.7657
GFS	0.5068	0.7191	0.5040	0.6742	0.8023	0.9358	0.8037	0.6656
MCDM	0.7057	0.6923	0.7119	0.6873	0.7589	0.9316	0.7987	0.7010
MARLFS	0.7005	0.6639	0.7173	0.6630	0.8042	0.9423	0.8287	0.7116
SARLFS	0.7037	0.6639	0.7078	0.6600	0.7929	0.9388	0.8030	0.7356
VTFS	0.7331	0.7182	0.7502	0.7194	0.8069	0.9509	0.8483	0.7197
AFT	0.6750	0.6664	0.5933	0.6333	0.8008	0.9276	0.8440	0.7261
LDA	0.6708	0.6716	0.6967	0.6392	0.7990	0.9345	0.8233	0.7178
ERG	0.7140	0.7510	0.7190	0.6190	0.7540	0.9310	0.7890	0.7340
TTG	0.6580	0.6770	0.6640	0.6120	0.7470	0.9270	0.8290	0.7540
NFS	0.6600	0.6770	0.6640	0.6140	0.7410	0.9310	0.8250	0.7600
CoEvolve	**0.8209**	**0.8159**	**0.7624**	**0.7837**	0.7954	**0.9533**	**0.8525**	**0.7767**
Δ_{Raw} (%)	$+22.14$	$+21.07$	$+9.54$	$+21.92$	-1.48	$+0.47$	$+3.79$	$+7.92$
Δ_{best} (%)	$+11.97$	$+8.64$	$+1.63$	$+8.94$	-1.43	$+0.25$	$+0.49$	$+1.44$

Table 2: Supplementary experiment: test evaluation without cross-validation. Bold marks the highest reported score. The last two rows give CoEvolve's relative gains: $100(s_{\text{CoEvolve}}/s_{\text{ref}} - 1)$. The best baseline is selected separately for each dataset, excluding Raw and CoEvolve. Negative gains are retained.

B THE FEATURE ENGINEERING EFFECTS ARE CONDITIONED

This appendix provides a reproducible example in which the same feature-engineering (FE) action improves performance under one anchor but decreases performance under another. Here, an *anchor* is the current output feature set, not a subset of data rows. The example separates two statements: the absolute performance of an output feature set is independent of its construction history under a fixed evaluation protocol, whereas the improvement produced by an action depends on its starting set. A useful FE decision must therefore account for the complete generation-and-selection transition, rather than assign an unconditional benefit to an arithmetic operator.

B.1 Synthetic Data and Two Anchors

We generate $n = 1{,}000$ independent observations from

$$\begin{aligned} f_1, f_2, f_3, f_4 &\overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1), \\ Y &= f_1 + f_2 + \tfrac{1}{2} f_4. \end{aligned} \tag{12}$$

There is no additional response noise. Thus, f_3 is independent of the response, while f_4 provides predictive information not available in (f_1, f_2). Both anchors use the same observations and response values:

$$S_A = \{f_1, f_2, f_3\}, \qquad S_B = \{f_1, f_2, f_4\}. \tag{13}$$

They differ in only one feature: the irrelevant f_3 in S_A is replaced by the informative f_4 in S_B.

B.2 An Identical Generation-and-Selection Action

For either anchor, the same semantic action a generates

$$g = f_1 + f_2, \tag{14}$$

adds g to the current feature set, and retains three features using univariate K-Best selection. The term *same action* refers to this identical transformation and selection rule, not necessarily to the same numerical index in a state-dependent cluster-action catalog. The generated feature has exactly the same definition and values under both anchors.

For transparency, this diagnostic uses `SelectKBest(f_regression, k=3)`. The univariate regression F-statistic ranks candidates by their squared sample correlation with the response. This is a deliberately simplified selector, rather than the mutual-information selector used in the main CoEvolve o1 method. Likewise, we use R^2 here for an analytically interpretable regression example; these numbers are not the main benchmark's $1 - \mathrm{RAE}$ results, nor are they the results of running the full CoEvolve search.

At the population level, the relevant correlations are

$$\begin{aligned} &\mathrm{Corr}(f_1, Y) = \mathrm{Corr}(f_2, Y) = \tfrac{2}{3}, \\ &\mathrm{Corr}(f_3, Y) = 0, \qquad \mathrm{Corr}(f_4, Y) = \tfrac{1}{3}, \\ &\mathrm{Corr}(g, Y) = \tfrac{2\sqrt{2}}{3}. \end{aligned} \tag{15}$$

Consequently, a marginal ranking can prefer g, f_1, f_2 even though g is redundant given (f_1, f_2). In the actual sample used below, every training fold selects

$$\widehat{S}'_{A,k} = \widehat{S}'_{B,k} = \{f_1, f_2, g\}, \qquad k = 1, \ldots, 5. \tag{16}$$

The action therefore removes an irrelevant variable from S_A, but removes a genuinely informative variable from S_B.

B.3 Leakage-Controlled Cross-Validation Protocol

We use the same five folds for both anchors and both representations, generated by `KFold(n_splits=5, shuffle=True, random_state=42)`. The downstream estimator is a random forest regressor with 100 trees, a minimum of five training samples per leaf, and `random_state=42`. Other settings follow scikit-learn 1.5.1 defaults, including bootstrap sampling, unrestricted tree depth, squared-error splitting, and `max_features=1.0`. Each evaluation fits a fresh estimator.

Feature selection is fitted separately within each training fold, using only that fold's training responses. Let I_k^{tr} and I_k^{va} denote the training and held-out indices for fold k. The fold-specific transition is

$$\widehat{S}'_{S,k} = \mathrm{KBest}_{I_k^{\mathrm{tr}},3}\big(S \cup \{g\}\big). \tag{17}$$

The selected columns are used to train the forest on I_k^{tr} and predict the unchanged held-out observations I_k^{va}. Computing the fixed, elementwise sum $g = f_1 + f_2$ does not require response values or fitted preprocessing statistics.

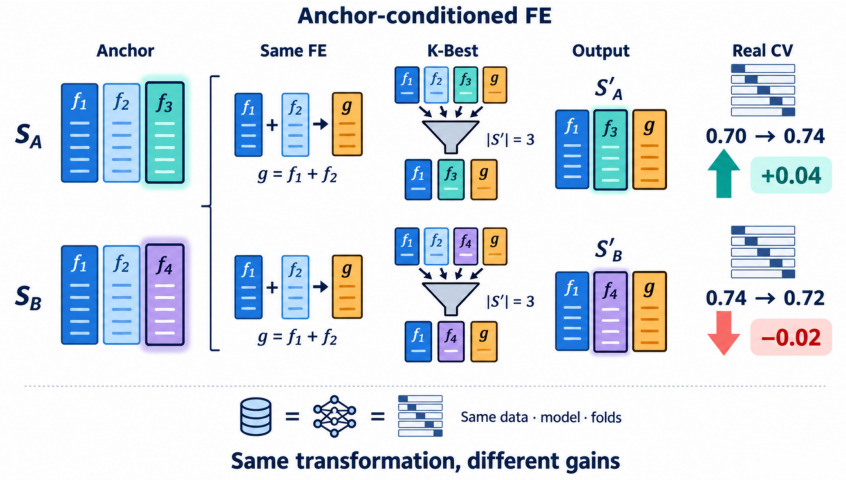


Figure 4: The effect of feature engineering operations are conditioned on feature sets/ anchors.

Anchor	Before FE	After FE	Paired gain
S_A	0.858055	0.865528	+0.007473
S_B	0.954073	0.865528	−0.088545

Table 3: Measured mean five-fold R^2 in the synthetic example. Both anchors become $\{f_1, f_2, g\}$ after generation and fold-local selection. These are reproducible synthetic measurements, not OpenML benchmark results or schematic figure annotations.

For each fold, the score is

$$R_k^2 = 1 - \frac{\sum_{i \in I_k^{\mathrm{va}}} (y_i - \widehat{y}_i)^2}{\sum_{i \in I_k^{\mathrm{va}}} (y_i - \overline{y}_{I_k^{\mathrm{va}}})^2}. \tag{18}$$

We report the arithmetic mean of the five fold scores, not a pooled out-of-fold R^2. Writing $R_k^2(S)$ for a forest trained and evaluated with set S, the paired action gain is

$$\widehat{\Delta}(S, a) = \frac{1}{5} \sum_{k=1}^{5} \left[R_k^2\big(\widehat{S}'_{S,k}\big) - R_k^2(S) \right]. \tag{19}$$

This formulation also covers cases where selection produces different sets in different folds. In this experiment, Equation (16) makes the post-action set identical across all folds and both anchors.

The two anchors and the action are fixed in this diagnostic. There is no iterative FE search or test-based candidate reranking in the reproduction code. The held-out portions are internal CV validation folds, not an additional independent test dataset. The example is explanatory rather than a confirmatory experiment.

B.4 Measured Results

Table 3 reports the measured results of this single synthetic experiment. The same action has a positive mean gain for S_A and a negative mean gain for S_B. Both post-action scores are identical because the selected matrices have the same columns in the same order, and use the same folds, estimator settings, and random seeds.

For completeness, Table 4 gives the individual fold scores. The small improvement for S_A is not positive in every fold; the example establishes the observed mean contrast, not statistical significance or a universal benefit.

Fold	S_A before FE	S_B before FE	Shared output
1	0.857692	0.955528	0.870216
2	0.878282	0.955159	0.889817
3	0.857912	0.945705	0.862432
4	0.831338	0.955381	0.826488
5	0.865051	0.958589	0.878686

Table 4: Fold-wise R^2. The shared output is $\{f_1, f_2, g\}$. Reported means and gains are computed before rounding.

B.5 Why the Starting Feature Set Matters

Anchor A: no predictive information is lost. Replacing f_3 by g removes an irrelevant feature and exposes the target's additive component directly to the downstream learner. Although g contains no new information beyond (f_1, f_2), a split on g can represent a boundary $f_1 + f_2 \leq c$ that is oblique in the original two-dimensional coordinates. This can simplify approximation by a finite axis-aligned random forest. The observed improvement may reflect both this representation change and the removal of an irrelevant input. Without ablations, their separate contributions cannot be identified.

Anchor B: marginal filtering removes complementary signal. Before the action, S_B contains all variables in the response function. After selection, f_4 is absent. Its marginal association is weaker than those of g, f_1, f_2, so a univariate selector can discard it even though it contains information that the retained features cannot recover. The conditional expectation and residual variance after the action are

$$\begin{aligned} \mathbb{E}[Y \mid f_1, f_2, g] &= f_1 + f_2, \\ \mathrm{Var}(Y \mid f_1, f_2, g) &= \tfrac{1}{4}, \\ \mathrm{Var}(Y) &= \tfrac{9}{4}. \end{aligned} \tag{20}$$

Hence, under the population distribution and squared-error prediction, the Bayes population R^2 using $\{f_1, f_2, g\}$ is

$$R^2_{\text{Bayes}} = 1 - \frac{1/4}{9/4} = \frac{8}{9}. \tag{21}$$

The same population benchmark applies to S_A, since f_3 is independent noise, while using all of S_B permits a Bayes population R^2 of one. These are population reference values, not strict upper bounds on individual finite-sample CV scores or guarantees about the fitted random forest.

Implication for CoEvolve o1. The example is consistent with optimizing over output sets: identical final sets have identical evaluated performance under the fixed protocol. Nevertheless, action gains are differences from different baselines:

$$\boxed{r(S, a) = F(T(S, a)) - F(S), \qquad r(S_A, a) \neq r(S_B, a).} \tag{22}$$

The arithmetic expression does not acquire a different numerical meaning under a different anchor. Rather, its utility as part of a complete FE transition depends on the information and representation already present, and on what the selection step retains or removes.

Accordingly, FE-MDP should predict state-conditioned rewards, and CAM should preserve the anchor and the complete observed feature-set change when presenting empirical evidence to the LLM. UAM may retain a reusable hypothesis, such as checking redundancy and complementary information before accepting a generated feature, but should not turn this example into an unconditional rule that adding two features is beneficial. For this one-step diagnostic, the immediate reward and the anchor-relative gain coincide; this need not hold for deeper beam descendants.

C Overall CoEvolve Search Procedure

Algorithm 1 summarizes the complete search. Pool unions merge candidates by canonical output-set identity, and $\mathcal{V}[S]$ denotes the cached CV score of S. TopEI selects up to n_a anchors from the entire

Algorithm 1 CoEvolve: global anchor selection and memory-guided local search

Require: Task $\mathcal{D}$, rounds $T_{\max}$, anchor count n_a, beam settings (H, B, q, η)
Ensure: Best evaluated feature set $\widehat{S}$

Initialize $\mathcal{P}_0$, $\mathcal{V}_0$, CAM, and UAM through seed exploration
Train FE-MDP on measured one-step gains from seed exploration
for $t = 0, \ldots, T_{\max} - 1$ **do**
  $\mathcal{P} \leftarrow \mathcal{P}_t$, $\mathcal{V} \leftarrow \mathcal{V}_t$
  Train a new DSGO on $\mathcal{V}$ and rebuild its Bayesian posterior
  $\mathcal{A}_t \leftarrow \textsc{TopEI}(\mathcal{P}, \mathcal{V}, \mathrm{DSGO}, n_a)$
  Evaluate uncached anchors; collect new (S, y) pairs in $\Delta\mathcal{V}_t^{\mathrm{a}}$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \Delta\mathcal{V}_t^{\mathrm{a}}$
  for each anchor $S^{\mathrm{a}} \in \mathcal{A}_t$ **do**
    $\mathcal{B}_{S^{\mathrm{a}}} \leftarrow \textsc{BeamExploreCV}(S^{\mathrm{a}}, \text{FE-MDP}, \mathcal{V})$
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{S : (S, y) \in \mathcal{B}_{S^{\mathrm{a}}}\}$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{B}_{S^{\mathrm{a}}}$
  end for
  Update CAM from nonempty anchor-conditioned outcomes $\{\mathcal{B}_{S^{\mathrm{a}}}\}$
  Update UAM by consolidating the new CAM reports with earlier experience
  for each anchor $S^{\mathrm{a}} \in \mathcal{A}_t$ **do**
    $c \leftarrow \textsc{GetCAM}(\mathrm{CAM}, S^{\mathrm{a}})$, $u \leftarrow \textsc{Retrieve}(\mathrm{UAM}, S^{\mathrm{a}})$
    $\mathcal{L}_{S^{\mathrm{a}}} \leftarrow \textsc{ProposeLLM}(S^{\mathrm{a}}, c, u, \mathcal{A}_{\mathrm{rank}}(S^{\mathrm{a}}))$
    for each valid proposed action $a \in \mathcal{L}_{S^{\mathrm{a}}}$ **do**
      $S' \leftarrow T(S^{\mathrm{a}}, a)$, $y' \leftarrow \textsc{CachedOrCV}(S', \mathcal{V})$
      Skip this action if execution or CV evaluation fails
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{S'\}$, $\mathcal{V} \leftarrow \mathcal{V} \cup \{(S', y')\}$
      Record the measured gain $y' - \mathcal{V}[S^{\mathrm{a}}]$ for FE-MDP training
    end for
  end for
  Continue training FE-MDP on accumulated measured one-step gains
  $\mathcal{P}_{t+1} \leftarrow \mathcal{P}$, $\mathcal{V}_{t+1} \leftarrow \mathcal{V}$
end for
return $\widehat{S}$ from $(\widehat{S}, \widehat{y}) \in \arg\max_{(S,y)\in\mathcal{V}_{T_{\max}}} y$

global pool and keeps this batch fixed for the round. GETCAM returns an empty report when none is available. BEAMEXPLORECV uses the configured beam settings, evaluates selected successors with cache reuse, retains beam states by measured scores, and records observed one-step gains. It returns all successfully evaluated feature-set–score pairs, including candidates pruned from the beam. The displayed pool updates summarize insertions made as successful observations become available.

Local exploration supplies global candidates and training observations, while global selection supplies new contexts for local exploration. UAM accumulates the contextual reports produced by successive beam explorations, providing evolving guidance for subsequent LLM proposals.