AFE-Master: Enhancing LLM-Driven Autonomous Feature Engineering with Domain-Specific Language Parsing and Guided Local Search

Anonymous Author(s)
Affiliation
Address
email

Abstract

Autonomous feature engineering (AFE) is crucial for improving predictive performance on tabular datasets, liberating domain experts from the intensive process of manual feature generation. However, traditional AFE approaches lack the semantic guidance which is required to fully leverage domain knowledge. Although large language models (LLMs) make it possible in principle to mimic human experts and inject expert knowledge and feature-engineering experience, existing LLM-based methods still face significant challenges in creating semantically rich and syntactically complex expert-level features. To address these limitations, we propose AFE-Master, a novel method that combines a domain-specific language (DSL) and Abstract Syntax Trees (ASTs) to enhance LLMs' ability to parse and manipulate complex feature structures in a controlled manner. We further integrate Guided Local Search (GLS) for interpretable, progressive feature optimization, ensuring a smooth transition from simpler features to optimal feature sets. Extensive experiments on multiple popular Kaggle and OpenML datasets and one deployed online A/B test demonstrate that our approach significantly outperforms baselines, improving real-world application's performance and suggesting a promising direction for the next generation of AFE.

1 Introductions

Feature generation is the process of constructing and selecting new features from raw data to better capture underlying patterns and relationships. However, achieving exceptional feature generation typically requires substantial domain expertise and considerable human effort, making the process time-consuming and challenging to scale [Hollmann et al., 2024]. To streamline this, Automated Feature Engineering (AFE) employs algorithms to automatically construct high-quality features while reducing the reliance on human intervention [Zhang et al., 2023].

Traditional AFE methods can be primarily divided into two categories: expansion-selection paradigms and sequential decision paradigms. The former first constructs a large number of new candidate features by combining operators and original features and uses statistical tools and machine learning models to select high-quality features, such as SAFE [Shi et al., 2020], OpenFE [Zhang et al., 2023], and others [Kanter and Veeramachaneni, 2015, Katz et al., 2016, Lam et al., 2017]. The sequential decision paradigm typically attempts to model feature construction as a multi-step decision process, where each step constructs a single feature and often uses methods like reinforcement learning to learn a near-optimal feature generation policy [Wu et al., 2022, Chen et al., 2019, Zhu et al., 2022, Li et al., 2023, Khurana et al., 2018]. However, given the vast and exponentially growing space of high-order features (i.e., features involving multiple operators), these methods still lack comprehensive and systematic search capabilities and cannot explicitly leverage domain knowledge. As a result, they may

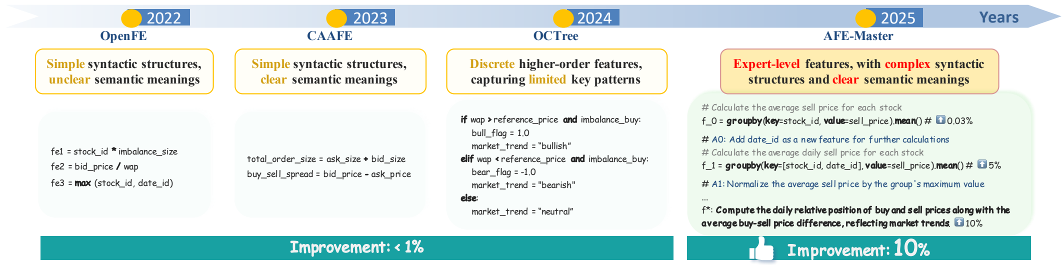

Figure 1: This figure shows the evolution of AFE methods recently. Existing AFE methods struggle to capture key patterns and only result in limited performance improvements. AFE-Master, with expert-level features that combine complex syntactic structures and clear semantics, leads to the most advanced performance boost.

consequently struggle to reliably identify expert-level features in such a sparse and high-dimensional feature space in the same way that human experts do.

To expose fundamental limitations of traditional AFE algorithms, in Figure 1 we present a characteristic case study on the widely-used *Optiver* dataset, which originates from a large-scale and popular Kaggle quantitative trading competition. OpenFE, one of the state-of-the-art AFE algorithm, is expected to comprehensively construct all possible first-order features and select the highest quality ones. However, it primarily identified features with ambiguous semantics, such as multiplying stock IDs by imbalanced size, resulting in only marginal performance improvements. This highlights the inefficiency of traditional AFE methods in identifying expert-level features with rich semantics and complex structures.

In recent years, the application of large language models (LLMs) in data science has gained widespread attention and adoption [Hong et al., 2024, Chen et al., 2024, Guo et al., 2024]. CAAFE [Hollmann et al., 2024] utilizes LLMs to iteratively generate and select new features, guiding feature construction based on task descriptions. OCTree [Nam et al., 2024] proposes a black-box evolutionary approach that optimizes features iteratively and incorporates decision tree representations into prompts, enabling the LLM to construct features by referencing and combining decision tree code fragments.

However, despite these advancements, existing LLM-based AFE methods still face notable limitations in constructing expert-level features. As shown in Figure 1, since CAAFE relies on an almost end-to-end generation of features based on task descriptions, it imposes extremely high demands on the model's reasoning ability, often resulting in the construction of features with simple syntactic structures. On the other hand, As noted by the authors of OCTree, most of the features generated by this method are discrete, making it difficult to fully capture the key patterns in the dataset and unlikely to produce optimal features. In conclusion, neither method achieves feature generation at the level of human experts, that is, the ability to efficiently construct high-order features that integrate domain knowledge while continuously optimizing the features based on domain knowledge and performance feedback.

To tackle the above challenges, we propose *AFE-Master*, a pipeline that turns a naive seed feature into an expert-level indicator through iterative guided refinement. Each feature is first encoded in a lightweight domain-specific language (DSL) that distills the original Python code into its core operators and operands; syntax parsing then unfolds the DSL into an abstract syntax tree (AST), exposing a complete transformation hierarchy for precise LLM analysis. Guided Local Search (GLS) explores this AST with small syntactic or semantic edits: the LLM proposes a handful of candidate refinement plans, executes them in parallel, and—through a reflection loop that repairs ineffective variants—selects the best-performing candidate at each iteration, so the search quickly navigates to a high-quality feature.

Our main contributions are threefold: (1) we introduce a DSL together with AST-based parsing for interpretable and efficient feature representation, enabling LLMs to comprehend and maintain complex features more easily; (2) we integrate GLS for iterative refinement across syntactic and semantic neighborhoods, coupling it with LLM-driven plan execution and a reflection mechanism to boost search efficiency and feature quality by leveraging meaningful reforms and lessons from past failures; and (3) we show that AFE-Master achieves competitive results against existing AFE baselines on 13 popular Kaggle datasets and 10 OpenML datasets when paired with 3 representative downstream models, XGBoost [Chen and Guestrin, 2016], MLP [Gorishniy et al., 2021] and TabPFN [Hollmann et al., 2025]. We also deploy AFE-Master on an online A/B test to demonstrate its practical value for real real-world tabular data task.

2 Preliminaries

In this section, we provide necessary background on Large Language Models (LLMs), automated feature engineering (AFE), domain-specific languages (DSLs), and syntax parsing. These topics collectively lay the foundation for understanding how our proposed method integrates LLMs and syntax-based techniques to construct expert-level features.

2.1 Large Language Models

Large Language Models (LLMs) refer to deep neural network architectures that are trained on massive corpora of text data to effectively capture linguistic patterns, contextual relationships, and broad world knowledge—examples include GPT [Radford, 2018] and BERT [Devlin, 2018]. In recent years, substantial advances in LLM research have dramatically broadened their practical application scope, extending well beyond traditional language generation and dialogue systems to encompass more domain-specific tasks such as automated code generation and data science workflows [Hong et al., 2023, Huang et al., 2023, Guo et al., 2024]. By learning from a diverse mix of textual sources—including programming snippets, scientific articles, and specialized domain corpora—LLMs can develop a surprisingly robust understanding of formal languages and symbolic reasoning, making them increasingly attractive and promising for tasks like autonomous feature engineering (AFE).

2.2 Feature Generation

Feature generation, or feature construction, involves creating new features from raw data in order to enhance a machine learning model's ability to capture salient patterns. Traditionally, this process has often relied heavily on manual domain expertise, where data scientists apply a variety of mathematical transformations, aggregations, or logical operations to derive more informative features. However, such manual approaches tend to be extremely time-consuming, inherently prone to human error, and relatively less scalable when faced with large or highly complex datasets.

Automated Feature Engineering (AFE) methods have therefore been developed to mitigate these limitations. Early approaches such as the expansion-selection paradigm generate numerous candidate features by combining predefined operators and then prune them using statistical or model-based selection [Kanter and Veeramachaneni, 2015, Katz et al., 2016, Lam et al., 2017, Shi et al., 2020, Zhang et al., 2023]. Reinforcement learning-based approaches model feature construction as a sequential decision process, learning policies that select effective transformations step by step [Chen et al., 2019, Li et al., 2023, Khurana et al., 2018]. However, these methods may still face challenges when the search space is large or when nuanced domain knowledge is crucial. The integration of LLMs offers a new avenue for generating more semantically informed features, potentially bridging the gap between fully manual and purely algorithmic approaches [Hollmann et al., 2024, Nam et al., 2024].

2.3 Domain-Specific Language

Domain-Specific Languages (DSLs) [Mernik et al., 2005] are specialized mini-languages precisely tailored to particular problem domains. Unlike general-purpose programming languages, DSLs concentrate on expressing only the concepts and operations most relevant to a given domain, thereby enabling clearer, more concise, and considerably less error-prone code. In the context of feature engineering, a DSL can elegantly encapsulate common transformations (e.g., arithmetic, logical, or statistical operations) while effectively shielding users (or automated systems) from unnecessary low-level details. By providing a structured yet highly flexible way to express feature logic, DSLs help LLMs more reliably understand, generate, and optimize valid feature expressions—especially when combined with formal parsing techniques (Section 2.4).

2.4 Syntax Parsing

Syntax parsing is the process of analyzing a string (or sequence of tokens) according to the grammar rules of a language and constructing a parse tree or AST [Sun et al., 2023]. In the context of a DSL for feature engineering, the goal is to transform textual expressions into a tree structure that precisely captures operator precedence, operand relationships, and logical grouping. An example is presented in Figure 2. By working directly with ASTs, we enable the LLM to perform fine-grained edits on any subtree, such as inserting, removing, or replacing operators, without disturbing unrelated parts of the expression. At the same time, each node carries type annotations and semantic metadata (e.g. numerical/categorical constraints), which lets us automatically verify that every transformation is valid. Finally, defining local edit operations on the AST naturally induces a neighborhood over candidate features, powering our guided local search to efficiently explore high-order variants. Together, these capabilities make feature construction both rigorously structured and easily interpretable by the LLM.

3 Methods

3.1 Overview

In real-world feature-engineering practice, human experts rarely start from scratch; instead, they prefer to fine-tune existing statistics, preserving the underlying signal while continually enhancing expressiveness. AFE-Master follows the same philosophy: the system first uses an LLM to propose a simple seed feature, then converts it into a concise DSL expression and parses it into an AST. During each optimization iteration, the LLM analyzes the AST structure and feature semantics to diagnose potential shortcomings and generates a handful of high-value reform plans. These plans are executed and evaluated in parallel; if a plan fails to yield an improvement, a reflection phase is triggered, where the LLM explains the failure, refines the strategy, and retries. Ultimately, the best-performing variant is adopted as the new feature state and becomes the starting point for the next search round, continuing until a preset maximum number of iterations is reached. Through this proposal–feedback–reflection loop, AFE-Master can—like an experienced feature engineer—evolve simple statistics into complex, high-efficiency professional features.

Figure 2 uses the Optiver stock-trading dataset to illustrate a concrete instance of AFE-Master: within just a few iterations, the system elevates a naive average-price feature into an expert-level indicator that fuses spread information and temporal granularity. The left panel traces the complete optimization path from F_0 to F_5, whereas the right panel zooms in on a single optimization iteration—showing how AFE-Master first simplifies the feature expression via the DSL, represents its detailed structure with an AST, then reasons about improvements, executes candidate plans in parallel with reflection, and finally selects the best refinement. We will elaborate on the key components of our method in the next two subsections, using this example as a running guide.

3.2 Feature Representation through Domain Specific Language and Syntax Parsing

Pure natural-language descriptions struggle to precisely describe the internal structure of complex features, whereas raw Python code entangles high-level logic with low-level implementation details, distracting an LLM from the core semantics [Mirzadeh et al., 2024]. To balance brevity and interpretability, we first build upon existing feature construction operators [Zhang et al., 2023] and introduce a lightweight DSL that adopts a functional syntax, retains only operators and dependent fields, and omits boilerplate implementation details. For instance, the DSL expression of $\mathtt{F}_2$ in Figure 2 (upper right) fits on a single line yet captures the full group-by $\rightarrow$ sliding-window $\rightarrow$ aggregation pipeline, conveying far more information per token than the Python snippet on the left and allowing the LLM to reason and rewrite directly at the expression level. Appendix C lists the complete DSL grammar.

A linear DSL string, however, still masks the internal hierarchy and reusable sub-expressions of a feature. We therefore **parse the DSL into an AST**. In the tree, each node represents an operator, parent nodes denote higher-order compositions, and subtrees correspond to local sub-expressions. This structure offers three benefits: (i) *pluggable analysis*—the LLM can inspect or replace any

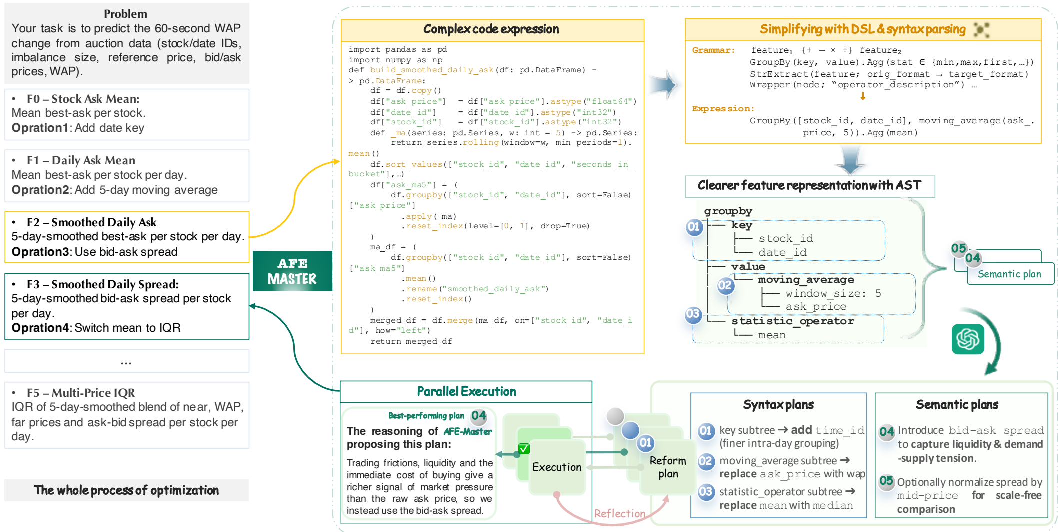

Figure 2: This diagram illustrates the process of feature optimization using AFE-Master. Features are first represented using a DSL and parsed into ASTs. GLS then utilizes LLMs to propose feature improvement plans from both syntactic and semantic perspectives, iteratively optimizing the features.

subtree without affecting the global logic; (ii) *similarity metrics*—tree-edit distance provides a natural measure of syntactic difference, which GLS uses to define neighborhoods; (iii) *step-wise explanation*—every refinement appears as a small set of node insertions or replacements, making the optimization path easy to trace and justify. The boxes at the bottom right of Figure 2 summarize two types of typical AST modification plans: 3 *syntactic* plans that add, swap, or remove specific nodes, and 2 *semantic* plans that adjust the feature's underlying meaning. By coupling the DSL's concise surface form with the AST's hierarchical view, AFE-Master retains semantic fidelity while giving the LLM a feature representation that is both highly operable and readily interpretable.

3.3 Guided Local Search: From Initial Guess to Expert-Level Features

With the AST in place, AFE-Master enters the *Guided Local Search* (GLS) phase. At each round the LLM proposes a handful of local reform plans to the AST; candidates are evaluated in parallel, ineffective variants are revised through a brief reflection–retry loop, and only the best-performing version is retained for the next iteration. Figure 2 shows a typical trajectory on the Optiver dataset: five GLS iterations transform an initial moving-average ask price (F_0) into an expert-level indicator that combines bid–ask spread, temporal bucketing, and an IQR statistic (F_5).

3.3.1 Defining the Neighborhoods of Change

We characterize fine-grained edits to a feature f using two complementary neighborhoods. First, the **syntactic neighborhood** $\mathcal{N}_{\text{syn}}(f)$ relies on the feature's AST and measures difference via a one-step tree-edit distance d_{AST}: for example, replacing the aggregation operator `mean` with `median` or swapping the field `ask_price` for `wap` each counts as a single edit ($d_{\text{AST}} = 1$). Second, the **semantic neighborhood** $\mathcal{N}_{\text{sem}}(f)$ embraces variants that remain close in feature meaning; such candidates can often be produced in one shot by prompting an LLM—for instance, converting ask price to bid–ask spread would require several AST edits in principle, yet the LLM can perform this semantic-level change with a single instruction. Together, these neighborhoods define the overall search space:

$$\mathcal{N}(f) = \mathcal{N}_{\text{syn}}(f) \ \cup \ \mathcal{N}_{\text{sem}}(f),$$

where any $f' \in \mathcal{N}(f)$ is deemed a legitimate local reform of f.

3.3.2 LLM-Driven Reform Plans

Even within the restricted neighborhoods, the number of possible reform plans remains intractable. We therefore enlist a LLM as an "automated feature engineer". Using the feature's AST structure and semantics, and drawing on domain knowledge learned during pre-training, the LLM proposes only a few high-value reform plans. As illustrated in Figure 2, starting from `F2: Smoothed Daily Ask`, the LLM proposes five plans. On the syntactic side it (1) refines the `groupby` key by adding `time_id`, (2) swaps the moving-average field from `ask_price` to `wap`, and (3) replaces the aggregation operator `mean` with `median`. Semantically, it (4) suggests substituting the ask price with the bid–ask spread to capture liquidity pressure and (5) normalizing prices to remove scale effects. Each proposal is backed by explicit reasoning; for example, the spread-based variant (plan 4) is motivated by the observation that "the bid–ask spread directly quantifies the immediate cost of execution and therefore reflects market pressure more faithfully than a single-side quote".

3.3.3 Knowledge-Driven Parallel Plan Execution and Reflection

At each GLS round, the reform plans $\mathcal{P} = \{p_1, \ldots, p_n\}$ proposed by the LLM are applied **in parallel** to the current feature $\mathbf{f}^{(t)}$, producing a candidate set $\tilde{\mathcal{N}}(\mathbf{f}^{(t)}) = \{\mathbf{f}'_1, \ldots, \mathbf{f}'_n\}$. Each candidate is evaluated by the downstream model M on dataset D; the utility $\mathcal{U}(\mathbf{f}', D, M)$ records the validation-set performance gain obtained after adding the new feature. If a reform plan yields no improvement, the system enters a *reflection* phase: the LLM analyses the failure, revises the original plan, or proposes an alternative, and the revised plan is re-evaluated. Every plan is allowed at most k retries. For example, adding `time_id` to the `groupby` key once seemed promising but worsened validation performance; reflection revealed that `time_id` is a continuous field with many unique values, over-partitioning the data. The LLM therefore suggested discretising `seconds_in_bucket` and using the resulting buckets instead, preserving high-frequency temporal structure without fragmenting the groups and ultimately delivering a positive gain. Among all successful candidates, the one with the highest utility is chosen as the next feature state.

$$\mathbf{f}^{(t+1)} = \arg \max_{\mathbf{f}' \in \tilde{\mathcal{N}}(\mathbf{f}^{(t)})} \mathcal{U}(\mathbf{f}', D, M).$$

where $\tilde{\mathcal{N}}(\mathbf{f}^{(t)})$ is the set of candidate features generated in the current round, $\mathbf{f}^{(t+1)}$ denotes the best-performing feature that will serve as the baseline for the next GLS iteration. This combination of plan execution and reflection ensures both fast convergence and semantic integrity, addressing the challenge of efficiently discovering high-order, domain-aware features.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate AFE-Master on 13 representative datasets from Kaggle, covering a diverse range of scales and complexities: (1) three beginner-level Kaggle competition and datasets; (2) eight playground-level monthly Kaggle competitions; and (3) two difficult industrial-grade competitions offering cash prizes. These datasets vary in sample size and feature count, representing applications in real-world domains such as finance, healthcare, agriculture, education, environmental protection, and disaster prediction. They encompass classification tasks, regression tasks, time-series datasets, and non-time-series datasets. All competition datasets are widely popular, with thousands of participating teams. Following the setting in CAAFE, we include textual (language-based) descriptions for each dataset. Detailed dataset characteristics are provided in Appendix B. Additionally, to offer a more comprehensive evaluation, we tested all methods on the 10 OpenML datasets used in CAAFE; results are presented in Section 4.3.

Baseline Models. To assess the effectiveness of **AFE-Master**, we use **XGBoost** [Chen and Guestrin, 2016], **MLP** [Gorishniy et al., 2021], and the recent tabular ML model **TabPFN** [Hollmann et al., 2025] as downstream predictive models. XGBoost is a competitive decision tree–based method, while MLP represents a classic neural network architecture. For comparison, we evaluate the features generated by the following AFE baselines: (1) **OpenFE** [Zhang et al., 2023]: a state-of-the-art traditional AFE method that uses operator-based expansion and feature selection; (2) **CAAFE** [Hollmann et al., 2024]: an LLM-based approach that generates features end-to-end using

Table 1: Performance Improvement of Feature Engineering Methods for Regression Tasks (MAE)

Downstream	Method	optiver	abalone	bike	crab	google	flood	Avg
XGBoost	OpenFE	0.16%	-0.02%	31.71%	-0.04%	2.56%	-0.64%	5.62%
	CAAFE	0.15%	0.02%	0.92%	-0.09%	31.89%	0.00%	5.48%
	OCTree	0.55%	-0.02%	2.04%	0.01%	0.53%	-0.64%	0.41%
	Ours	**9.90%**	**1.39%**	**43.96%**	**0.13%**	**54.08%**	**7.01%**	**19.41%**
MLP	OpenFE	0.60%	0.35%	-28.39%	-0.43%	0.58%	**0.71%**	-4.43%
	CAAFE	2.07%	-1.68%	2.30%	-0.23%	32.14%	0.00%	5.77%
	OCTree	0.50%	-0.29%	-2.71%	-0.49%	0.43%	0.00%	-0.43%
	Ours	**8.67%**	**0.99%**	**20.30%**	**0.03%**	**53.72%**	0.00%	**13.95%**

Table 2: Performance Improvement of Feature Engineering Methods for Classification Tasks (ACC)

Downstream	Method	multi	academic	disease	kidney	smoke	soft	spac	Avg
XGBoost	OpenFE	**0.31%**	0.12%	6.88%	5.61%	0.25%	0.11%	0.70%	2.00%
	CAAFE	-4.06%	0.65%	15.54%	8.45%	-0.63%	0.05%	0.85%	2.98%
	OCTree	0.21%	0.12%	3.44%	-1.40%	0.00%	0.00%	0.00%	0.34%
	Ours	**0.31%**	**2.74%**	**20.64%**	**23.15%**	**0.46%**	**2.11%**	**1.10%**	**7.22%**
MLP	OpenFE	2.76%	0.41%	-3.12%	**13.80%**	0.82%	0.26%	4.75%	2.81%
	CAAFE	4.98%	-2.11%	6.27%	-3.43%	0.37%	-0.89%	5.96%	1.59%
	OCTree	1.07%	-2.26%	4.69%	1.14%	0.37%	0.26%	1.08%	0.91%
	Ours	**5.21%**	**1.02%**	**12.50%**	**13.80%**	**1.51%**	**0.53%**	**6.12%**	**5.81%**

task descriptions; and (3) **OCTree** [Nam et al., 2024]: another LLM-based method that leverages evolutionary search and decision-tree prompts for feature construction. Results on TabPFN are reported in Section 4.3.

Experimental Configuration. For all datasets, we allocate 60% for training, 20% for validation, and 20% for testing. We follow OCTree's setting of using MAE for regression tasks and ACC for classification tasks. During MLP training, we use embeddings for categorical features. Both XGBoost and MLP are tuned using hyperparameter optimization, with details on the search space provided in Appendix A. We select GPT-4o-mini as the LLM used in the dialogue, which is lightweight, cost-effective, and offers fast responses. Despite its smaller size, GPT-4o-mini effectively supports our feature engineering pipeline, generating valuable suggestions and identifying potential improvements.

4.2 Main Results

Tables 1 and 2 demonstrate the effectiveness of AFE-Master across 13 well-known Kaggle datasets with varying difficulty. We evaluate each AFE method based on the relative error reduction compared to the baseline without feature engineering. Our analysis reveals that AFE-Master consistently outperforms existing methods across datasets of different scales, complexities, types, and application domains, achieving significant performance improvements. Specifically, our approach yields an average improvement of 16.68% for regression tasks and 6.52% for classification tasks, demonstrating a significant gain over existing methods. These results highlight the strong feature construction capabilities of our method.

Our method consistently performs well across datasets of varying difficulty levels and distributions by leveraging a concise DSL and a structured AST to effectively decompose and understand complex feature structures. Moreover, the integration of GLS further enables the progressive discovery of high-quality features through small, interpretable edits. In contrast, other LLM-based methods—such as CAAFE and OCTree—can sometimes be constrained by the limited expressiveness of their feature representations or the relative inefficiency of their search algorithms. This comparison suggests that AFE-Master more fully harnesses the reasoning capabilities of LLMs, thereby yielding more meaningful and interpretable feature transformations.

Unlike other approaches that often exhibit performance fluctuations across different datasets, AFE-Master by contrast consistently maintains stable and reliable gains. For instance, although CAAFE performs relatively well on the Google dataset and OpenFE tends to excel on the Bike dataset, their effectiveness varies quite significantly on other benchmarks—perhaps because the features they generate are inherently simpler and thus less capable of modeling more complex, higher-order interactions. Similarly, OCTree's decision-tree–driven prompts tend to favor discrete feature constructions and may also struggle to capture more intricate transformations such as grouped time-series patterns or advanced aggregations.

By optimizing features both syntactically and semantically, AFE-Master ensures a gradual yet flexible refinement process that dynamically adjusts feature structures. This dual strategy allows our method to explore a broader spectrum of high-quality candidates, making it robust across diverse datasets and application domains. These findings reinforce AFE-Master's advantages in automated feature engineering, demonstrating its ability to generate structured, interpretable, and high-performing features for a wide range of machine learning tasks.

4.3 Ablations and Analysis

Performance on Advanced Tabular ML Model. In order to further validate the applicability of our approach to cutting-edge tabular learners, we evaluated each AFE method on TabPFN [Hollmann et al., 2025], a transformer pretrained offline on millions of synthetic tasks and shown to be highly competitive on small tabular benchmarks ($< 10\,\mathrm{k}$ rows). Following the official protocol, we ran all methods on the four Kaggle datasets within this size range. As reported in Table 3, AFE-Master outperforms all baselines by a wide margin, achieving an average error reduction of $+4.52\%$, thereby demonstrating its effectiveness even for state-of-the-art tabular foundation models.

Table 3: Performance Improvement on TabPFN

Method	academic	disease	kidney	spac	Avg
OpenFE	–0.33%	–4.48%	**+4.33%**	+1.72%	+0.31%
CAAFE	+0.79%	+4.45%	+1.44%	+1.32%	+2.00%
OCTree	–0.50%	+4.45%	+2.89%	+1.17%	+2.00%
Ours	**+1.08%**	**+10.44%**	**+4.33%**	**+2.23%**	**+4.52%**

Performance on OpenML Benchmarks. In order to further demonstrate the broad applicability of our approach, we evaluated AFE-Master on 10 well-known OpenML benchmarks previously used by CAAFE. Following our primary experimental protocol—using XGBoost as the downstream learner and GPT-4o-mini for feature construction—we assessed each dataset over three random splits. Table 4 reports the relative error reduction on each benchmark. AFE-Master consistently achieves substantially greater performance gains than existing AFE techniques, markedly reducing the downstream model's prediction error. These results further underscore the robustness and generalization capabilities of AFE-Master across a wide range of tabular tasks.

Table 4: Performance Improvement on 10 OpenML Benchmarks

Dataset	OpenFE	CAAFE	OCTree	Ours
airlines	**4.96%**	-1.11%	-1.48%	3.68%
balance_scale	-11.50%	**100.00%**	8.18%	**100.00%**
breast-w	6.91%	**14.41%**	-7.21%	**14.41%**
chess	3.10%	51.10%	-1.14%	**78.73%**
cmc	**2.64%**	-3.11%	-2.41%	0.47%
credit-g	2.41%	4.90%	3.66%	**5.49%**
diabetes	-1.97%	-3.86%	-11.66%	**1.93%**
eucalyptus	**7.16%**	1.22%	5.95%	1.77%
pc1	2.32%	2.48%	0.00%	**6.97%**
tic-tac-toe	-14.88%	-28.93%	**57.02%**	42.98%
Average	0.12%	13.71%	5.09%	**25.64%**

Effectiveness of the AST component. To isolate the contribution of the AST-based representation, we removed the AST representation of the features from the prompt and asked the LLM to reason directly over the raw DSL string. Tested on all 13 Kaggle datasets used in the main experiments, results are shown in 5. this

Table 5: Ablation (gain) with and without AST. **Reg-Avg**: 6 regression Kaggle datasets; **Cls-Avg**: 7 classification Kaggle datasets; **All-Avg**: overall mean.

Method	Reg-Avg	Cls-Avg	All-Avg
AFE-Master (w/o AST)	9.90%	0.49%	4.83%
AFE-Master	**19.41%**	**7.22%**	**12.85%**

ablated variant suffered a marked drop in performance, confirming that tree-level structure is critical to AFE-Master's gains.

Portability across LLMs. To verify that our method remains effective across various LLMs, we evaluated AFE-Master using Qwen-2.5-7B-Instruct. As shown in Table 6, although the average improvement was lower than with GPT-4o-mini, it still outperformed all other LLM-based baselines (using GPT-4o-mini) and OpenFE. Full results appear in Appendix F.

Table 6: Average Improvement of AFE-Master With Different LLMs on 13 Kaggle Datasets

Method	Avg
AFE-Master (Qwen-2.5-7B-Instruct)	5.10%
AFE-Master (GPT-4o-mini)	**12.85%**

How o1 performs? We also tried a single-turn feature construction with `GPT-o1-preview` on all 13 Kaggle datasets in the main experiments, but this slightly degraded average performance (–0.69%), suggesting that standalone reasoning cannot substitute for downstream feedback and underscoring the necessity of our iterative design. Full results appear in Appendix E

These trends corroborate that our iterative AST + GLS pipeline is essential to stable and effective feature discovery.

5 Online Testing Results

To evaluate the practical value of AFE-MASTER by deploying it in a large-scale online advertising scenario—one of the most profitable tabular data task in the IT industry. Specifically, we test on our industrial advertising platform, which serves billions of users through a mobile app store. The baseline (control group) uses a well-engineered FiBinet model Huang et al. [2019] with 167 expert-crafted features refined over two years. In the experiment group, we add 20 features automatically generated by AFE-MASTER through offline searching, keeping all other model settings unchanged.

We initially allocate 5% of user traffic (millions of users) to each group and observe significant performance improvements in the experiment group over one week. Based on these results, we increase the traffic to 30% for further validation. After another week of observation, as summarized in Table 7, the experiment group demonstrates consistent gains in many key metrics like platform revenue, and Click Through Rate . These results confirm that AFE-MASTER delivers substantial economic benefits at industrial scale. As a result, the experiment group has now serving 100% of the traffic.

Table 7: Online A/B test result. The AFE-MASTER augments the baseline feature set with 20 automatically discovered descriptors, yielding consistent gains on many platform KPIs.

Metric	**Definition (simplified)**	**Lift vs. Baseline**
RPM	Revenue per mille	**+0.63%**
CPM	Cost per mille	**+15.11%**
Revenue	Net revenue collected by the platform	**+0.60%**
CTR	Click through rate	**+3.01%**

6 Conclusion and Limitation

In this paper, we presented *AFE-Master*, a novel automated feature engineering framework that overcomes the limitations of existing LLM-based approaches in crafting semantically rich and syntactically intricate features. By integrating a concise DSL, hierarchical AST representations, and a Guided Local Search strategy, AFE-Master enables progressive, interpretable feature refinement. Empirical results on 13 Kaggle datasets and one deployed online A/B test demonstrate that our method consistently outperforms both traditional AFE algorithms and recent LLM-driven techniques, uncovering expert-level features that boost performance across tree-based and neural network models.

Despite these advances, several avenues remain for future work. On the syntactic side, leveraging AST edit paths more effectively could yield even more efficient transformations. On the semantic front, training specialized models for feature embeddings may support finer-grained, context-aware modifications. In future research, we plan to enrich our operator set, explore more diverse transformation patterns, and integrate deeper domain knowledge to further enhance the stability and interpretability of generated features.

References

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *Advances in Neural Information Processing Systems*, 36, 2024.

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.

Qitao Shi, Ya-Lin Zhang, Longfei Li, Xinxing Yang, Meng Li, and Jun Zhou. Safe: Scalable automatic feature engineering framework for industrial tasks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1645–1656. IEEE, 2020.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE, 2015.

Gilad Katz, Eui Chul Richard Shin, and Dawn Song. Explorekit: Automatic feature generation and selection. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 979–984. IEEE, 2016.

Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*, 2017.

Yu Wu, Xin Xi, and Jieyue He. Afgsl: Automatic feature generation based on graph structure learning. *Knowledge-Based Systems*, 238:107835, 2022.

Xiangning Chen, Qingwei Lin, Chuan Luo, Xudong Li, Hongyu Zhang, Yong Xu, Yingnong Dang, Kaixin Sui, Xu Zhang, Bo Qiao, et al. Neural feature search: A neural architecture for automated feature engineering. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 71–80. IEEE, 2019.

Guanghui Zhu, Zhuoer Xu, Chunfeng Yuan, and Yihua Huang. Difer: differentiable automated feature engineering. In *International Conference on Automated Machine Learning*, pages 17–1. PMLR, 2022.

Liyao Li, Haobo Wang, Liangyu Zha, Qingyi Huang, Sai Wu, Gang Chen, and Junbo Zhao. Learning a data-driven policy network for pre-training automated feature engineering. In *The Eleventh International Conference on Learning Representations*, 2023.

Udayan Khurana, Horst Samulowitz, and Deepak Turaga. Feature engineering for predictive modeling using reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Sirui Hong, Yizhang Lin, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Lingyao Zhang, Mingchen Zhuge, et al. Data interpreter: An llm agent for data science. *arXiv preprint arXiv:2402.18679*, 2024.

Yibin Chen, Yifu Yuan, Zeyu Zhang, Yan Zheng, Jinyi Liu, Fei Ni, and Jianye Hao. Sheetagent: A generalist agent for spreadsheet reasoning and manipulation via large language models. *arXiv preprint arXiv:2403.03636*, 2024.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. *arXiv preprint arXiv:2402.17453*, 2024.

Jaehyun Nam, Kyuyoung Kim, Seunghyuk Oh, Jihoon Tack, Jaehyung Kim, and Jinwoo Shin. Optimized feature generation for tabular data via llms with decision tree reasoning. *arXiv preprint arXiv:2406.08527*, 2024.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

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.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Alec Radford. Improving language understanding by generative pre-training. 2018.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Dong Huang, Qingwen Bu, Jie M Zhang, Michael Luck, and Heming Cui. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*, 2023.

Marjan Mernik, Jan Heering, and Anthony M Sloane. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344, 2005.

Weisong Sun, Chunrong Fang, Yun Miao, Yudu You, Mengzhe Yuan, Yuchen Chen, Quanjun Zhang, An Guo, Xiang Chen, Yang Liu, et al. Abstract syntax tree for programming language understanding and representation: How far are we? *arXiv preprint arXiv:2312.00413*, 2023.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM conference on recommender systems*, pages 169–177, 2019.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly articulate the main contributions and the scope of AFE-Master, which are subsequently supported by the paper's results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated section or discussion on the limitations of the proposed AFE-Master method and the conducted experimental evaluations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The primary contributions of this paper are centered on a novel empirical method and its experimental validation, rather than formal theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides comprehensive details on the experimental setup, datasets (publicly available), and evaluation metrics, sufficient for enabling reproduction of the main findings.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a LLM), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The datasets used (Kaggle, OpenML) are publicly accessible, and the code for AFE-Master and experimental reproduction will be provided in the supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All pertinent details regarding the experimental settings, including data splits, hyperparameter selection, and optimizer configurations, are thoroughly described in the experimental section and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper presents experimental results with appropriate measures of statistical significance over multiple runs, to validate the claims.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details regarding the computational environment, including hardware specifications and approximate execution times, are provided in the appendix to aid reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research has been conducted in strict adherence to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper includes a discussion on the broader impacts, outlining potential benefits in democratizing feature engineering and acknowledging considerations related to automated decision-making systems.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The work focuses on a method (AFE-Master) utilizing LLMs for feature engineering; it does not involve the release of new large-scale pretrained models or datasets that pose a high direct risk for misuse requiring specific safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected? [Yes]

    Justification: All existing assets, specifically the publicly available Kaggle and OpenML datasets, are appropriately credited, and their terms of use are respected; references to their sources are provided.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The new AFE-Master method is thoroughly documented within the paper, and any released code will be accompanied by sufficient documentation for its usage and understanding.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This research does not involve crowdsourcing experiments or any direct research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: As the research does not involve human subjects, Institutional Review Board (IRB) approval was not applicable.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper clearly describes that Large Language Models (LLMs) are an important and core component of the proposed AFE-Master methodology.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Experimental Settings

This section introduces our specific experimental setup. For the downstream XGBoost and MLP Gorishniy et al. [2021] models, we followed the approach of OCTree Nam et al. [2024], utilizing the Optuna library and the same parameter search space to find the optimal hyperparameters. Table 8 presents the hyperparameter search space for XGBoost. Table 9 shows the hyperparameter search space for MLP. For MLP, we trained for 300 epochs with early stopping; if the validation score did not improve within 40 epochs, early stopping was triggered. We used ReduceOnPlateau as the learning rate scheduler.

Table 8: XGBoost hyperparameter search space

Parameter	Hyperparameters Space
Max depth	UniformInt [1, 11]
Num estimators	UniformInt [100, 6100, 200]
Min child weight	LogUniformInt [1, 1e2]
Subsample	Uniform [0.5, 1]
Learning rate	LogUniform [1e-5, 0.7]
Col sample by level	Uniform [0.5, 1]
Col sample by tree	Uniform [0.5, 1]
Gamma	LogUniform [1e-8, 7]
Lambda	LogUniform [1, 4]
Alpha	LogUniform [1e-8, 1e2]

Table 9: MLPGorishniy et al. [2021] hyperparameter search space

Parameter	Hyperparameters Space
Num layers	UniformInt [1, 8]
Layer size	UniformInt [16, 1024]
Dropout	Uniform [0, 0.5]
Learning rate	LogUniform [1e-5, 1e-2]
Category embedding size	UniformInt [64, 512]
Learning rate scheduler	[True, False]
Batch size	[256, 512, 1024]

B Datasets

This section introduces the datasets used in our main experiments. As shown in Table 10, we employed 13 datasets of varying complexity, from different scenarios and at different scales, to evaluate the effectiveness of our method.

- academic: Kaggle Academic Dataset
- disease: Kaggle Disease Dataset
- spac: `https://www.kaggle.com/competitions/spaceship-titanic`
- abalone: `https://www.kaggle.com/competitions/playground-series-s4e4`
- bike: `https://www.kaggle.com/competitions/bike-sharing-demand`
- crab: `https://www.kaggle.com/competitions/playground-series-s3e16`
- flood: Kaggle Flood Dataset
- multi: `https://www.kaggle.com/competitions/playground-series-s4e2`
- kidney: `https://www.kaggle.com/competitions/playground-series-s3e12`
- smoke: `https://www.kaggle.com/competitions/playground-series-s3e24`
- soft: `https://www.kaggle.com/competitions/playground-series-s3e23`
- optiver: Kaggle Optiver Dataset
- google: Kaggle Google Brain Dataset

Table 10: Summary of datasets used in main experiments

Dataset Name	Scenario	Feature Count	Sample Count	Complexity
academic	education	19	4,424	beginner
disease	healthcare	10	349	beginner
spac	disaster prediction	14	8,693	beginner
abalone	agriculture	10	90,615	playground
bike	transportation	14	10,886	playground
crab	agriculture	10	74,051	playground
flood	disaster prediction	22	1,117,957	playground
multi	healthcare	18	20,758	playground
kidney	healthcare	8	414	playground
smoke	healthcare	23	38,984	playground
soft	finance	23	101,763	playground
optiver	finance	15	3,335,004	industrial
google	healthcare	8	6,036,000	industrial

C DSL

The syntax of our feature-construction DSL is defined as follows:

- **Discretization**

  ```
  Bin(feature, original_margin, target_set)
  ```

  Split a continuous feature into specified bins.

- **Arithmetic Expressions**

  ```
  feature_a + feature_b,  feature_a - feature_b,
  feature_a * feature_b,  feature_a / feature_b
  ```

  Combine features using the usual operators.

- **Grouping Operations**

  ```
  GroupBy(key, value).Agg(statistic_operator)
  ```

  where
  $$\texttt{Agg} \in \{\texttt{min}, \texttt{max}, \texttt{mean}, \texttt{median}, \texttt{std}, \texttt{rolling\_mean}, \texttt{first}, \dots\}$$
  Compute statistics over grouped data.

- **Time-Series Processing**

  ```
  TimeExtract(feature, 'timeseries_desc')
  ```

  Construct time-series features.

- **String Processing**

  ```
  StrExtract(feature, original_format_desc, target_format_desc)
  ```

  Extract features from string fields based on format descriptions.

- **Generic Wrapper**

  ```
  Wrapper(node, 'operator_desc')
  ```

  Annotate or encapsulate any AST node with a custom operator description.

- **Multi-Feature Loop**

  ```
  for name in [feature1, feature2, ...]:  <DSL expression using name>
  ```

  Apply an operator across multiple features in a loop.

Table 11: Performance of AFE-Master with and without AST for Regression Tasks (MAE)

Method	optiver	abalone	bike	crab	google	flood	Avg
Ours w/o AST	0.36%	0.34%	36.33%	0.03%	20.41%	1.91%	9.90%
Ours	**9.90%**	**1.39%**	**43.96%**	**0.13%**	**54.08%**	**7.01%**	**19.41%**

Table 12: Performance of AFE-Master with and without AST for Classification Tasks (ACC)

Method	multi	academic	disease	kidney	smoke	soft	spac	Avg
Ours w/o AST	-1.87%	1.39%	10.32%	-7.09%	0.25%	0.00%	0.45%	0.49%
Ours	**0.31%**	**2.74%**	**20.64%**	**23.15%**	**0.46%**	**2.11%**	**1.10%**	**7.22%**

D AST Component Ablation

To validate the effectiveness of the AST component, we compared the performance of the model with and without the AST component. In the absence of the AST component, the LLM will directly analyze potential issues in the DSL expression. The results in Table 11 and Table 12 demonstrate that AFE-Master with the AST component significantly outperforms the version without the AST component, highlighting the critical role of the AST component in feature representation and optimization.

E Direct Reasoning with O1

In this section, we performed single-turn feature construction with GPT-o1-preview (an advanced reasoning model) on all 13 Kaggle datasets. All methods used the tuned XGBoost. As shown in Table 14 and Table 13, we found that the average performance across the 13 datasets actually declined slightly, with an average error reduction of –0.69%. This result indicates that standalone reasoning cannot substitute for downstream feedback and further underscores the necessity of our iterative design.

Table 13: Performance Improvement of O1-Direct and AFE methods with GPT-4o-mini for Classification Tasks (ACC)

Method	multi	academic	disease	kidney	smoke	soft	spac	Avg
OpenFE	**0.31%**	0.12%	6.88%	5.61%	0.25%	0.11%	0.70%	2.00%
CAAFE	-4.06%	0.65%	15.54%	8.45%	-0.63%	0.05%	0.85%	2.98%
OCTree	0.21%	0.12%	3.44%	-1.40%	0.00%	0.00%	0.00%	0.34%
o1-direct	-6.56%	-0.49%	12.06%	-5.65%	-0.46%	-0.11%	**2.81%**	0.23%
Ours(GPT-4o-mini)	**0.31%**	**2.74%**	**20.64%**	**23.15%**	**0.46%**	**2.11%**	1.10%	**7.22%**

Table 14: Performance Improvement of O1-Direct and AFE methods with GPT-4o-mini for Regression Tasks (MAE)

Method	optiver	abalone	bike	crab	google	flood	Avg
OpenFE	0.16%	-0.02%	31.71%	-0.04%	2.56%	-0.64%	5.62%
CAAFE	0.15%	0.02%	0.92%	-0.09%	31.89%	0.00%	5.48%
OCTree	0.55%	-0.02%	2.04%	0.01%	0.53%	-0.64%	0.41%
o1-direct	0.55%	-0.19%	-0.10%	-0.19%	-9.44%	-1.27%	-1.77%
Ours	**9.90%**	**1.39%**	**43.96%**	**0.13%**	**54.08%**	**7.01%**	**19.41%**

F Generalizability Analysis

To verify the generalizability of our method, we used Qwen-2.5-7b-instruct as the base model. As demonstrated in Table 16 and Table 15, it also showed strong positive effects, the error reductions of

AFE-Master using Qwen-7b are 5.10%, and AFE-Master using apt-4o-mini is 12.83%. While the average improvement was lower than that of AFE-Master with GPT-4o-mini, it still outperformed other LLM-based methods using GPT-4o-mini and OpenFE.

Table 15: Performance Improvement of AFE-Master with Qwen-7b for Classification Tasks (ACC)

Method	multi	academic	disease	kidney	smoke	soft	spac	Avg
OpenFE	**0.31%**	0.12%	6.88%	5.61%	0.25%	0.11%	0.70%	2.00%
CAAFE	-4.06%	0.65%	15.54%	8.45%	-0.63%	0.05%	0.85%	2.98%
OCTree	0.21%	0.12%	3.44%	-1.40%	0.00%	0.00%	0.00%	0.34%
Ours(Qwen-7b)	-0.10%	0.78%	6.85%	2.81%	**0.50%**	0.00%	0.25%	1.58%
Ours(GPT-4o-mini)	**0.31%**	**2.74%**	**20.64%**	**23.15%**	0.46%	**2.11%**	**1.10%**	**7.22%**

Table 16: Performance Improvement of AFE-Master with Qwen-7b for Regression Tasks (MAE)

Method	optiver	abalone	bike	crab	google	flood	Avg
OpenFE	0.16%	-0.02%	31.71%	-0.04%	2.56%	-0.64%	5.62%
CAAFE	0.15%	0.02%	0.92%	-0.09%	31.89%	0.00%	5.48%
OCTree	0.55%	-0.02%	2.04%	0.01%	0.53%	-0.64%	0.41%
Ours(Qwen-7b)	7.55%	0.38%	12.26%	**0.13%**	27.91%	**7.01%**	9.21%
Ours(GPT-4o-mini)	**9.90%**	**1.39%**	**43.96%**	**0.13%**	**54.08%**	**7.01%**	**19.41%**

G A Brief Qualitative Analysis of AFE-Master's Optimization Efficiency in High-Order Feature Space

In this section, we theoretically explain why AST based GLS ensures efficient exploration and optimization in high-dimensional feature spaces is highly valuable. We attempt to provide a qualitative analysis, hoping it can inspire further relevant discussions.

This analysis is divided into four parts:

1. We point out that when the representational power of the operator set O is unlimited, and the feature order is sufficiently high, there may exist an optimal feature f^* that can approximate the mapping between the original feature set F_o and the label with high accuracy.
2. In the ideal scenario described above, the goal of feature optimization is to optimize the new feature f to approximate f^*. We introduce the concept of the largest common subtree size to quantitatively analyze the similarity $S(f, f^*)$ between f and f^* from a syntactic perspective, thereby providing the optimization objective for high-order feature generation algorithms.
3. We point out that the feature quality verifier used in current LLM-based feature generation algorithms, which is based on the performance of the downstream ML model, is affected by negative factors such as overfitting and contains some error. During the optimization process, there is a certain probability that it will incorrectly accept negative feature optimization operations. Therefore, in the feature improvement operations made by feature generation algorithms for high-order features, a significant portion must be positive improvements. Otherwise, the verifier will accept more negative improvements, preventing the feature optimization algorithm from stably and iteratively improving feature quality.
4. We analyze two feature optimization algorithms: AST+GLS and black-box optimization. We point out that AST+GLS can leverage LLM domain knowledge and feature engineering knowledge to perform more explicit textual reasoning and traverse all subtrees of the syntax tree. This gives it a high probability of proposing effective improvements for the common largest subtree and its subtrees. Even if the verifier occasionally accepts a negative improvement due to error, the single-step edit distance is small, and only minor damage will be done to the common subtree. Therefore, the AST+GLS algorithm holds promise for achieving stable iterative optimization in high-order feature spaces. In contrast, black-box

optimization algorithms typically lack explicit, interpretable textual reasoning during the optimization process. They tend to randomly make uncontrolled modifications to the feature f, and given the vast scale and sparse valid solutions in high-order feature spaces, such random and uncontrolled modifications are unlikely to effectively enlarge the size of the largest common subtree. Considering the verifier's error, more negative operations will likely be accepted, and these uncontrolled operations are more likely to cause significant damage to the common largest subtree, making it difficult for black-box optimization algorithms to stably and iteratively improve feature quality.

G.1 Simplification of Feature Optimization Problem

Approximating the Optimal Feature Given a dataset D with an original feature set F_o, a downstream ML model M, and an operator set O, there exists a optimal new feature set F^* constructed from F_o and O, which can lead to the maximal improvement in the performance of the downstream ML model M. In an ideal scenario, if the representational power of the operator set O is unlimited, this set may contain an optimal feature f^*, which directly describes the relationship between the original feature set F_o and the target label, thereby reducing the prediction error of the downstream ML model M to near 0.

G.2 Feature Similarity Measurement Based on Abstract Syntax Trees – Maximum Common Subtree Size

In this simplified scenario, given a new feature f constructed from F_o and O, the optimization goal for feature f is to maximize its similarity to the optimal feature f^*. For convenience, we measure similarity by comparing the syntactic structures of f and f^*. Specifically, we define the similarity $S(f, f^*)$ of two features as their largest common subtree size, i.e., the number of nodes in their largest common subtree: if f and f^* are identical, their similarity S reaches its maximum value. Thus, the optimization objective can be written as:

$$\begin{gathered}\text{Maximize } S(f, f^*)\\ \text{Minimize size}(f)\end{gathered}$$

The reason for minimizing size(f) is that if f is too large, the downstream ML model needs to reverse-engineer the effective part from it.

When $S(f, f^*) = \text{size}(f^*)$ and $\text{size}(f) = \text{size}(f^*)$, then $f = f^*$, and we have obtained the optimal feature.

G.3 Feature Quality Verifier Based on Downstream ML Model Scores with Error

After determining the optimization objective, before analyzing the optimization algorithms, we first need to analyze the quality verifier of the feature V used by the LLM-based AFE method, which relies on changes in the scores of the downstream ML model in the validation set. As mentioned above, feature quality has already been measured using $S(f, f^*)$. Therefore, the objective of this feature quality verifier can be defined as evaluating the similarity between the feature f and the optimal feature f^* by inspecting the downstream model score. Considering negative factors such as the risk of overfitting, this verifier may have some error. That is, when the verifier compares the quality of a feature f with the feature f' obtained by executing a certain feature optimization operation on f (i.e., comparing $S(f, f^*)$ and $S(f', f^*)$), it may give an incorrect answer with some probability. In this setting, the optimization algorithm must have a high optimization efficiency, which means that with each round of optimization, it should have a high probability of discovering a better feature f' than f. Otherwise, due to the verifier's error, there is a significant chance that it will accept a new feature that is worse than the original feature.

G.4 Qualitative Comparison Between AST-based Local Reform Algorithm and Black-box Optimization Algorithm in High-dimensional Feature Space

Based on the analysis of the verifier above, the quality of a feature generation algorithm largely depends on the probability of discovering better features in each round of optimization. In practical optimization, for the proposed AST+GLS algorithm (AFE-Master), since we improve features by using an LLM to traverse each subtree of the AST and combine the LLM's semantic knowledge for reasoning, proposing locally optimized solutions for these subtrees, this method has two advantages:

1. The LLM will definitely traverse the largest common subtree between f and f^*, as well as its subtrees, and propose improvement options to these effective structures.
2. The local optimization solutions typically have a smaller edit distance, so they do not significantly disrupt the largest common subtree of f and f^*.

Since the proposed local reform options are derived from the LLM's reasoning based on both domain knowledge and feature engineering knowledge, they generally have strong rationality and a high probability of making effective improvements, thereby enlarging the common subtree between the optimized feature f and f^*.

Therefore, each round of optimization with AFE-Master has a high probability of proposing improvements that can expand the common subtree between the optimized feature and the optimal feature f^*, i.e., increasing $S(f, f^*)$. The verifier V will likely accept these valid solutions that expand the common subtree. Even if the verifier occasionally makes an error and accepts some invalid options, because the edit distance of the local optimization is small, these invalid changes will not cause significant damage to the effective subtrees in the original feature. In this way, we can iteratively improve the feature f in a positive direction.

In contrast, if we use a black-box optimization algorithm, since the LLM does not transform the feature's linear expression into a syntax tree and does not analyze each reform option for the AST individually, the model does not perform much explicit textual reasoning. Instead, it may tend to randomly add or update operands from the original feature set F_o and operators from the operator set O into the new feature f's linear expression. In this case, the edit distance of the optimization for the feature's syntax tree is highly uncontrollable. When f has higher order, due to the complex feature structure and lack of effective decomposition and detailed analysis, the algorithm is more likely to make random changes to the valid subtrees. Considering that the search space for feature engineering grows exponentially with the size of the feature, this optimization method is unlikely to accidentally expand the size of the effective subtree. Effective options are sparse, and the verifier V, under the influence of errors, will likely accept many invalid or negative feature improvements. Furthermore, due to the uncontrollable edit distance, a single negative optimization step may cause significant damage to the valid subtree. Therefore, black-box optimization methods face significant difficulties in performing effective optimization in high-dimensional feature space and are more likely to generate negative or ineffective high-order features, with a high risk of overfitting.

H Case Study

In this section, we present a case study to illustrate how AFE-Master optimizes a feature in optiver dataset. We provide the DSL expression and the underlying idea for each optimized feature.

Feature 0: `average_near_price_per_stock`

DSL expression:

```
GroupBy(stock_id, ask_price).Agg(mean)
```

Thought: Considering the data distribution, especially the potential issue of uneven grouping due to a small number of stocks, we can add a new grouping criterion, `date_id`, to the existing feature `average_ask_price_per_stock`. This will allow for further refinement when calculating the average ask price for each stock, breaking it down by date to reduce the imbalance caused by having a small number of stocks in each group. Specifically, the implementation will involve using both `stock_id` and `date_id` as grouping criteria in the `groupby` operation.

Note Each feature's "Thought" originates from an improvement suggestion made by the LLM after analyzing a corresponding subtree in the syntax tree. After all proposed improvements are implemented and evaluated, the one that performs best on the validation set is adopted.

Syntax Tree for Feature 0

```
└── groupby
    ├── keys
    │   └── stock_id
    ├── value
    │   └── ask_price
    └── statistic_operator
        └── mean
```

Other Proposed Improvements

value: To gain a complementary perspective on market dynamics, we propose replacing the feature `ask_price` with `bid_price` in the value field of the `groupby` operation. This change enables the analysis to focus on buyer-side pricing behavior, which can be particularly insightful in assessing demand-side pressure and liquidity conditions.

statistic_operator: Instead of using the mean, which can be sensitive to outliers, we suggest using the median as the statistical operator. The median provides a more robust measure of central tendency, especially in the presence of skewed distributions or extreme price values, thereby improving the reliability of summary statistics in volatile trading environments.

groupby: To improve comparability across different groups and mitigate scale differences in price levels, we suggest incorporating a normalization step within or after the `groupby` operation. This can involve min–max scaling or z-score standardization applied to the aggregated values.

Semantic Rewrite: For the subtree computing the bid–ask spread, rewrite it as

```
Spread(feature) := ask_price - bid_price
```

then feed `Spread(feature)` into downstream operators. This makes the computation's intent explicit and reusable.

Feature Momentum: Introduce a "momentum" operator on the spread, e.g.

```
Momentum(feature, lag=1) := Spread(feature) - Spread(feature).lag(1)
```

to capture short-term directional changes.

Global Normalization: After all aggregations, apply an overall z-score normalization across stocks and dates:

$$\texttt{Normalize}(X) := \frac{X - \mu_{\text{all}}}{\sigma_{\text{all}}}$$

to ensure features are on a common scale for the downstream model.

Feature 1: `average_ask_price_per_stock_per_date`

DSL expression:

```
GroupBy([stock_id, date_id], ask_price).Agg(mean)
```

Thought: We can further reduce the noise in the results by introducing a moving average to smooth the ask price. When calculating `average_ask_price_per_stock_per_date`, we will first apply a moving average to the `near_price` and then compute the mean of that.

Feature 2: `average_ask_price_per_stock_per_date_smoothed`

DSL expression:

```
GroupBy([stock_id, date_id],
  moving_average(ask_price, window_size=5)).Agg(mean)
```

Thought: Given that the current feature `average_ask_price_per_stock` is calculated based on the mean of the ask price, we could attempt to introduce a combination of price-related features to enhance the diversity and robustness of the feature. Based on the data distribution, it is recommended to replace `ask_price` with a combination of `bid_price` and `ask_price`, calculating their difference. This would provide a more comprehensive reflection of market price dynamics.

Feature 3: `average_ask_bid_price_per_stock_per_date_smoothed`

DSL expression:

```
GroupBy([stock_id, date_id],
  moving_average(ask_price - bid_price, window_size=5)).Agg(mean)
```

Thought: Since the calculation of the bid–ask spread may not be significantly affected by extreme values, we can introduce additional statistical measures to more comprehensively reflect the characteristics of the spread. Specifically, when calculating the bid–ask spread, in addition to using the mean, we can also introduce quartiles (Q1 and Q3) to capture the distribution characteristics of the spread. We can calculate the quartiles (Q1 and Q3) of the `ask_price` and `bid_price` for each stock on each date and use Q3–Q1 to represent the range of changes in the spread.

Feature 4: `iqr_ask_bid_price_per_stock_per_date_smoothed`

DSL expression:

```
GroupBy([stock_id, date_id],
  moving_average(ask_price - bid_price, window_size=5)).Agg(IQR)
```

Thought: In order to capture more distribution patterns, we can attempt additional price-related feature combinations such as `near_price`, `wap`, `far_price`, etc.

Feature 5: `iqr_prices_per_stock_per_date_smoothed`

DSL expression:

```
for price in [near_price, wap, far_price]:
  GroupBy([stock_id, date_id], moving_average(price, window_size=5)).Agg(IQR)
```

I Details of Data Description

The data description of each dataset is shown as below.

Abalone The task is to predict the age of abalones based on their physical measurements. Determining an abalone's age traditionally requires cutting the shell, staining it, and counting the number of rings through a microscope. We aim to predict age using more easily obtainable physical measurements to streamline this process.

Listing 1: Dataset description for abalone.


```
[
    {
        "fe_name": "id",
        "desc": "Unique identifier for each abalone specimen in the dataset."
    },
    {
        "fe_name": "Sex",
        "desc": "This feature represents the gender of the abalone, with three categories."
    },
    {
        "fe_name": "Length",
        "desc": "The length feature denotes the longest measurement of the abalone shell, from the
            apex to the base, measured in millimeters."
    },
    {
        "fe_name": "Diameter",
        "desc": "Diameter represents the measurement of the abalone shell perpendicular to its length,
             also measured in millimeters."
    },
    {
        "fe_name": "Height",
        "desc": "This feature signifies the height of the abalone shell, measured perpendicular to the
             plane formed by the length and diameter, in millimeters."
    },
    {
        "fe_name": "Whole weight",
        "desc": "Whole weight indicates the total weight of the abalone, encompassing both the meat
            and the shell, measured in grams."
    },
```

```
    {
        "fe_name": "Whole weight_1",
        "desc": "This feature represents the weight of the abalone meat only, measured in grams. It
            indicates the amount of meat extracted from the shell."
    },
    {
        "fe_name": "Whole weight_2",
        "desc": "Viscera weight signifies the weight of the abalone gut after bleeding, measured in
            grams. It provides insights into the weight of the internal organs of the abalone."
    },
    {
        "fe_name": "Shell weight",
        "desc": "This feature represents the weight of the abalone shell only, excluding the meat,
            measured in grams."
    },
    {
        "fe_name": "Rings",
        "desc": "Rings signify the number of rings present on the abalone shell, serving as an
            indicator of the abalone's age (add 1.5 to get the age in years). This is the target label
            ."
    }
]
```

Academic The task is to predict students' academic success based on their personal, social and academic characteristics. The dataset contains comprehensive information about students' backgrounds, academic performance, and various socio-economic factors that may influence their educational outcomes. The dataset was preprocessed using the same method as OCTreeNam et al. [2024]. We used the preprocessed dataset for training and prediction.

Listing 2: Dataset description for academic success.

```
[
    {
        "fe_name": "id",
        "desc": "Unique identifier for each student record."
    },
    {
        "fe_name": "Age_at_enrollment",
        "desc": "Student's age when enrolling in the program."
    },
    {
        "fe_name": "Gender",
        "desc": "Student's gender (1 - male, 0 - female)."
    },
    {
        "fe_name": "International",
        "desc": "Whether the student is international (1 - yes, 0 - no)."
    },
    {
        "fe_name": "Marital_status",
        "desc": "Student's marital status (1 - single 2 - married 3 - widower 4 - divorced 5 -facto
            union 6 - legally separated)."
    },
    {
        "fe_name": "Previous_qualification",
        "desc": "Student's previous qualifications (1 - Secondary education 2 - Higher education -
            bachelor's degree 3 - Higher education - degree 4 - Higher education - master's 5 - Higher
             education - doctorate 6 - Frequency of higher education 9 - 12th year of schooling - not
            completed 10 - 11th year of schooling - not completed 12 - Other - 11th year of schooling
            14 - 10th year of schooling 15 - 10th year of schooling - not completed 19 - Basic
            education 3rd cycle (9th/10th/11th year) or equiv. 38 - Basic education 2nd cycle (6th/7th
            /8th year) or equiv. 39 - Technological specialization course 40 - Higher education -
            degree (1st cycle) 42 - Professional higher technical course 43 - Higher education -
            master (2nd cycle))."
    },
    {
        "fe_name": "Curricular_units_1st_sem_approved",
        "desc": "Number of curricular units approved in the 1st semester."
    },
    {
        "fe_name": "Curricular_units_1st_sem_grade",
        "desc": "Grade average in the 1st semester (between 0 and 20)."
    },
    {
        "fe_name": "Curricular_units_2nd_sem_approved",
        "desc": "Number of curricular units approved in the 2nd semester."
    },
    {
        "fe_name": "Curricular_units_2nd_sem_grade",
        "desc": "Grade average in the 2nd semester (between 0 and 20)."
    },
    {
        "fe_name": "Father_qualification",
        "desc": "Father's education level (1 - Secondary Education - 12th Year of Schooling or Eq. 2 -
             Higher Education - Bachelor's Degree 3 - Higher Education - Degree 4 - Higher Education -
             Master's 5 - Higher Education - Doctorate 6 - Frequency of Higher Education 9 - 12th Year
             of Schooling - Not Completed 10 - 11th Year of Schooling - Not Completed 11 - 7th Year (
```

```
            Old) 12 - Other - 11th Year of Schooling 13 - 2nd year complementary high school course 14 - 10th Year of Schooling 18 - General commerce course 19 - Basic Education 3rd Cycle (9th/10th/11th Year) or Equiv. 20 - Complementary High School Course 22 - Technical-professional course 25 - Complementary High School Course - not concluded 26 - 7th year of schooling 27 - 2nd cycle of the general high school course 29 - 9th Year of Schooling - Not Completed 30 - 8th year of schooling 31 - General Course of Administration and Commerce 33 - Supplementary Accounting and Administration 34 - Unknown 35 - Can't read or write 36 - Can read without having a 4th year of schooling 37 - Basic education 1st cycle (4th/5th year) or equiv. 38 - Basic Education 2nd Cycle (6th/7th/8th Year) or Equiv. 39 - Technological specialization course 40 - Higher education - degree (1st cycle) 41 - Specialized higher studies course 42 - Professional higher technical course 43 - Higher Education - Master (2nd cycle) 44 - Higher Education - Doctorate (3rd cycle))."
    },
    {
        "fe_name": "Father_occupation",
        "desc": "Category indicating father's occupation (0 - Student 1 - Representatives of the Legislative Power and Executive Bodies, Directors, Directors and Executive Managers 2 - Specialists in Intellectual and Scientific Activities 3 - Intermediate Level Technicians and Professions 4 - Administrative staff 5 - Personal Services, Security and Safety Workers and Sellers 6 - Farmers and Skilled Workers in Agriculture, Fisheries and Forestry 7 - Skilled Workers in Industry, Construction and Craftsmen 8 - Installation and Machine Operators and Assembly Workers 9 - Unskilled Workers 10 - Armed Forces Professions 90 - Other Situation 99 - (blank) 101 - Armed Forces Officers 102 - Armed Forces Sergeants 103 - Other Armed Forces personnel 112 - Directors of administrative and commercial services 114 - Hotel, catering, trade and other services directors 121 - Specialists in the physical sciences, mathematics, engineering and related techniques 122 - Health professionals 123 - teachers 124 - Specialists in finance, accounting, administrative organization, public and commercial relations 131 - Intermediate level science and engineering technicians and professions 132 - Technicians and professionals, of intermediate level of health 134 - Intermediate level technicians from legal, social, sports, cultural and similar services 135 - Information and communication technology technicians 141 - Office workers, secretaries in general and data processing operators 143 - Data, accounting, statistical, financial services and registry-related operators 144 - Other administrative support staff 151 - personal service workers 152 - sellers 153 - Personal care workers and the like 154 - Protection and security services personnel 161 - Market-oriented farmers and skilled agricultural and animal production workers 163 - Farmers, livestock keepers, fishermen, hunters and gatherers, subsistence 171 - Skilled construction workers and the like, except electricians 172 - Skilled workers in metallurgy, metalworking and similar 174 - Skilled workers in electricity and electronics 175 - Workers in food processing, woodworking, clothing and other industries and crafts 181 - Fixed plant and machine operators 182 - assembly workers 183 - Vehicle drivers and mobile equipment operators 192 - Unskilled workers in agriculture, animal production, fisheries and forestry 193 - Unskilled workers in extractive industry, construction, manufacturing and transport 194 - Meal preparation assistants 195 - Street vendors (except food) and street service providers)."
    },
    {
        "fe_name": "Nacionality",
        "desc": "Student's nationality indicator (1 - Portuguese; 2 - German; 6 - Spanish; 11 - Italian; 13 - Dutch; 14 - English; 17 - Lithuanian; 21 - Angolan; 22 - Cape Verdean; 24 - Guinean; 25 - Mozambican; 26 - Santomean; 32 - Turkish; 41 - Brazilian; 62 - Romanian; 100 - Moldova (Republic of); 101 - Mexican; 103 - Ukrainian; 105 - Russian; 108 - Cuban; 109 - Colombian)."
    },
    {
        "fe_name": "Displaced",
        "desc": "Whether student is displaced (1 - yes 0 - no)."
    },
    {
        "fe_name": "Debtor",
        "desc": "Whether student has debt (1 - yes 0 - no)."
    },
    {
        "fe_name": "Tuition_fees_up_to_date",
        "desc": "Whether tuition fees are paid on time (1 - yes 0 - no)."
    },
    {
        "fe_name": "Scholarship_holder",
        "desc": "Whether student holds a scholarship (1 - yes 0 - no)."
    },
    {
        "fe_name": "Daytime_by_evening_attendance",
        "desc": "Attendance during day/evening classes (1 - daytime 0 - evening)."
    },
    {
        "fe_name": "Target",
        "desc": "The target variable indicating academic success classification, with three categories."
    }
}
]
```


Bike The task is to predict the number of bicycle rentals based on environmental and temporal factors. The dataset provides comprehensive information about weather conditions, time-related features, and historical rental patterns to help understand and forecast bike sharing demand.

Listing 3: Dataset description for bike sharing.

```
[
```

```
    {
        "fe_name": "Year",
        "desc": "The year when the rental occurred (four-digit integer)."
    },
    {
        "fe_name": "Month",
        "desc": "Month of the year (1 to 12)."
    },
    {
        "fe_name": "Day",
        "desc": "Day of the month (1 to 31)."
    },
    {
        "fe_name": "season",
        "desc": "Season of the year (1: Spring, 2: Summer, 3: Fall, 4: Winter)."
    },
    {
        "fe_name": "holiday",
        "desc": "Whether the day is a holiday."
    },
    {
        "fe_name": "workingday",
        "desc": "Whether the day is neither a weekend nor holiday."
    },
    {
        "fe_name": "weather",
        "desc": "Weather condition (1: Clear, Few clouds, Partly cloudy, Partly cloudy, 2: Mist +
            Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist, 3: Light Snow, Light Rain +
            Thunderstorm + Scattered clouds, Light Rain + Scattered clouds, 4: Heavy Rain + Ice
            Pallets + Thunderstorm + Mist, Snow + Fog )."
    },
    {
        "fe_name": "temp",
        "desc": "Temperature in Celsius."
    },
    {
        "fe_name": "atemp",
        "desc": "\"Feels like\" temperature in Celsius."
    },
    {
        "fe_name": "humidity",
        "desc": "Relative humidity."
    },
    {
        "fe_name": "windspeed",
        "desc": "Wind speed."
    },
    {
        "fe_name": "casual",
        "desc": "Number of non-registered user rentals initiated."
    },
    {
        "fe_name": "registered",
        "desc": "Number of registered user rentals initiated."
    },
    {
        "fe_name": "count",
        "desc": "Total number of bicycle rentals, this is the target variable we are trying to predict
            ."
    }
]
```

Crab The task is to predict the age of crabs based on their physical measurements and characteristics. Understanding a crab's age is crucial for marine biology research and sustainable fishing practices. The dataset provides comprehensive physical measurements and biological indicators that can help determine a crab's age without invasive methods.

Listing 4: Dataset description for crab age.

```
[
    {
        "fe_name": "id",
        "desc": "Unique identifier for each crab specimen in the dataset."
    },
    {
        "fe_name": "Sex",
        "desc": "Gender of the crab."
    },
    {
        "fe_name": "Length",
        "desc": "Length of the crab shell in meters (in Feet; 1 foot = 30.48 cms)."
    },
    {
        "fe_name": "Diameter",
        "desc": "Diameter of the crab shell in meters, perpendicular to length (in Feet; 1 foot =
            30.48 cms)."
    },
    {
```

```
        "fe_name": "Height",
        "desc": "Height of the crab in meters, providing volumetric information (in Feet; 1 foot = 30.48 cms)."
    },
    {
        "fe_name": "Weight",
        "desc": "Weight of the crab (in ounces; 1 Pound = 16 ounces)."
    },
    {
        "fe_name": "Shucked Weight",
        "desc": "Weight without the shell (in ounces; 1 Pound = 16 ounces)."
    },
    {
        "fe_name": "Viscera Weight",
        "desc": "Weight that wraps around your abdominal organs deep inside body (in ounces; 1 Pound = 16 ounces)."
    },
    {
        "fe_name": "Shell Weight",
        "desc": "Weight of the shell alone (in ounces; 1 Pound = 16 ounces)."
    },
    {
        "fe_name": "Age",
        "desc": "Age of the crab in months, this is the target variable we are trying to predict."
    }
]
```

Disease The task is to predict patient health outcomes based on various symptoms and medical indicators. The dataset provides comprehensive medical records including patient symptoms, demographic information, and vital signs to help identify potential health risks. The dataset was also preprocessed using the same method as OCTreeNam et al. [2024]. We used the preprocessed dataset for training and prediction.

Listing 5: Dataset description for disease symptoms.

```
[
    {
        "fe_name": "index",
        "desc": "Unique identifier for each patient record"
    },
    {
        "fe_name": "Age",
        "desc": "Patient's age in years"
    },
    {
        "fe_name": "Gender",
        "desc": "Patient's gender"
    },
    {
        "fe_name": "Fever",
        "desc": "Whether the patient has fever"
    },
    {
        "fe_name": "Cough",
        "desc": "Whether the patient has cough"
    },
    {
        "fe_name": "Fatigue",
        "desc": "Whether the patient experiences fatigue"
    },
    {
        "fe_name": "Difficulty Breathing",
        "desc": "Whether the patient has breathing difficulties"
    },
    {
        "fe_name": "Disease",
        "desc": "The name of the disease or medical condition"
    },
    {
        "fe_name": "Blood Pressure",
        "desc": "Patient's blood pressure status"
    },
    {
        "fe_name": "Cholesterol Level",
        "desc": "Patient's cholesterol level"
    },
    {
        "fe_name": "Outcome Variable",
        "desc": "Patient's health outcome. This is the target variable we are trying to predict."
    }
]
```

Flood The task is to predict flood probability based on various environmental and human factors. The dataset provides comprehensive information about natural and man-made factors that influence flood risks in different regions.

Listing 6: Dataset description for flood.

```
[
    {
        "fe_name": "id",
        "desc": "Unique identifier for each record in the dataset."
    },
    {
        "fe_name": "MonsoonIntensity",
        "desc": "Higher volumes of rain during monsoons increase the probability of floods."
    },
    {
        "fe_name": "TopographyDrainage",
        "desc": "The drainage capacity based on the region's topography. Efficient drainage can help
            drain rainwater and reduce the risk of floods."
    },
    {
        "fe_name": "Deforestation",
        "desc": "The extent of deforestation in the area. Deforestation reduces the soil's ability to
            absorb water, increasing surface runoff and the risk of floods."
    },
    {
        "fe_name": "ClimateChange",
        "desc": "The impact of climate change on the region. Climate change can lead to more extreme
            precipitation patterns, including torrential rains that can cause floods."
    },
    {
        "fe_name": "WetlandLoss",
        "desc": "Wetlands act as natural sponges, absorbing excess water and helping to prevent floods
            ."
    },
    {
        "fe_name": "CoastalVulnerability",
        "desc": "Low-lying coastal areas are prone to flooding from storm surges and sea level rise."
    },
    {
        "fe_name": "RiverManagement",
        "desc": "The quality and effectiveness of river management practices. Proper river management,
             including dredging and bank maintenance, can improve water flow and reduce floods."
    },
    {
        "fe_name": "DamsQuality",
        "desc": "The quality and maintenance status of dams. Well-maintained dams can control floods,
            and dams with structural problems can break and cause catastrophic floods."
    },
    {
        "fe_name": "DrainageSystems",
        "desc": "Well-maintained and adequately sized drainage systems help drain rainwater and reduce
             the risk of floods."
    },
    {
        "fe_name": "DeterioratingInfrastructure",
        "desc": "Clogged culverts, damaged drainage channels, and other deficient infrastructure can
            increase the risk of floods."
    },
    {
        "fe_name": "Urbanization",
        "desc": "The level of urbanization in the region. Urban areas have impermeable surfaces (
            asphalt, concrete), which reduce water infiltration, raising the risk of floods."
    },
    {
        "fe_name": "PopulationScore",
        "desc": "Densely populated areas can suffer more severe losses."
    },
    {
        "fe_name": "AgriculturalPractices",
        "desc": "The types and sustainability of agricultural practices. The intensification of
            agriculture can lead to deforestation, excessive use of fertilizers and pesticides, and
            inappropriate irrigation practices, reducing soil biodiversity and increasing the risk of
            floods."
    },
    {
        "fe_name": "Encroachments",
        "desc": "The degree of encroachment on flood plains and natural waterways. Construction in
            flood-prone areas impedes the natural flow of water and increases the risk of floods."
    },
    {
        "fe_name": "InadequatePlanning",
        "desc": "Urban planning that does not consider the risk of flooding increases the
            vulnerability of communities."
    },
    {
        "fe_name": "PoliticalFactors",
        "desc": "Factors such as corruption and a lack of political will to invest in drainage
            infrastructure can make it difficult to manage flood risk."
    },
    {
```

```
        "fe_name": "IneffectiveDisasterPreparedness",
        "desc": "The lack of emergency plans, warning systems, and simulations increases the negative
            impact of floods."
    },
    {
        "fe_name": "Watersheds",
        "desc": "Regions with more watersheds may have a higher or lower risk of flooding, depending
            on various factors."
    },
    {
        "fe_name": "Landslides",
        "desc": "Steep slopes and unstable soils are more prone to landslides."
    },
    {
        "fe_name": "Siltation",
        "desc": "The extent of siltation in rivers and reservoirs. The accumulation of sediments in
            rivers (siltation) reduces drainage capacity and increases the risk of floods."
    },
    {
        "fe_name": "FloodProbability",
        "desc": "The overall probability of flooding in the region. This is the target variable for
            predictive analysis."
    }
]
```

Google The task is to predict pressure in a ventilator system based on various input parameters and time-series measurements. The dataset provides detailed readings from mechanical ventilation systems, which is crucial for optimizing patient care in intensive care units.

Listing 7: Dataset description for google brain.

```
[
    {
        "fe_name": "id",
        "desc": "Unique identifier for each measurement record"
    },
    {
        "fe_name": "breath_id",
        "desc": "Globally-unique time step for breaths"
    },
    {
        "fe_name": "R",
        "desc": "Lung attribute indicating how restricted the airway is (in cmH2O/L/S). Physically,
            this is the change in pressure per change in flow (air volume per time). Intuitively, one
            can imagine blowing up a balloon through a straw. We can change R by changing the diameter
             of the straw, with higher R being harder to blow."
    },
    {
        "fe_name": "C",
        "desc": "Lung attribute indicating how compliant the lung is (in mL/cmH2O). Physically, this
            is the change in volume per change in pressure. Intuitively, one can imagine the same
            balloon example. We can change C by changing the thickness of the balloon's latex, with
            higher C having thinner latex and easier to blow."
    },
    {
        "fe_name": "time_step",
        "desc": "The actual time stamp."
    },
    {
        "fe_name": "u_in",
        "desc": "The control input for the inspiratory solenoid valve. Ranges from 0 to 100."
    },
    {
        "fe_name": "u_out",
        "desc": "The control input for the exploratory solenoid valve. Either 0 or 1."
    },
    {
        "fe_name": "pressure",
        "desc": "The airway pressure measured in the respiratory circuit, measured in cmH2O, this is
            the target variable we are trying to predict."
    }
]
```

Kidney The task is to predict the presence of kidney stones based on urinalysis results. The dataset provides detailed measurements of various physical and chemical properties of urine samples to help identify kidney stone formation risk.

Listing 8: Dataset description for kidney stone.

```
[
    {
        "fe_name": "id",
        "desc": "Unique identifier for each urine sample in the dataset"
    },
```

```
    {
        "fe_name": "gravity",
        "desc": "Specific gravity of urine, indicating density relative to water"
    },
    {
        "fe_name": "ph",
        "desc": "pH value of urine, measuring acidity/alkalinity level"
    },
    {
        "fe_name": "osmo",
        "desc": "Osmolarity of urine in milliosmoles (mOsm), a unit used in biology and medicine but
            not in physical chemistry. Osmolarity is proportional to the concentration of molecules in
             solution."
    },
    {
        "fe_name": "cond",
        "desc": "Conductivity of urine in milliMho (mMho), measuring ionic content. One Mho is one
            reciprocal Ohm. Conductivity is proportional to the concentration of charged ions in
            solution"
    },
    {
        "fe_name": "urea",
        "desc": "Urea concentration in urine in millimoles per liter (mmol/L)"
    },
    {
        "fe_name": "calc",
        "desc": "Calcium concentration in urine in millimoles per liter"
    },
    {
        "fe_name": "target",
        "desc": "Binary indicator for kidney stone presence (0: absent, 1: present). This is the
            target label we are trying to predict."
    }
]
```

Multi The task is to predict the obesity level of individuals based on their lifestyle and health-related features. The dataset provides detailed information on various aspects of individuals, including demographic data, physical measurements, dietary habits, lifestyle factors, and other relevant features such as family history of overweight and primary mode of transportation. These features collectively help in assessing the risk and severity of obesity.

Listing 9: Dataset description for multi class obesity.

```
[
    {
        "fe_name": "id",
        "desc": "A unique identifier for each data sample."
    },
    {
        "fe_name": "Gender",
        "desc": "The gender of the individual."
    },
    {
        "fe_name": "Age",
        "desc": "The age of the individual, measured in years."
    },
    {
        "fe_name": "Height",
        "desc": "The height of the individual, measured in meters."
    },
    {
        "fe_name": "Weight",
        "desc": "The weight of the individual, measured in kilograms."
    },
    {
        "fe_name": "family_history_with_overweight",
        "desc": "Indicates whether the individual has family member suffered or suffers from
            overweight."
    },
    {
        "fe_name": "FAVC",
        "desc": "Indicates whether the individual frequently consumes high-calorie foods."
    },
    {
        "fe_name": "FCVC",
        "desc": "Frequency of consumption of vegetables."
    },
    {
        "fe_name": "NCP",
        "desc": "The number of main meals consumed daily."
    },
    {
        "fe_name": "CAEC",
        "desc": "Consumption of food between meals."
    },
    {
        "fe_name": "SMOKE",
```

```
        "desc": "Indicates whether the individual smokes."
    },
    {
        "fe_name": "CH2O",
        "desc": "The amount of water consumed daily."
    },
    {
        "fe_name": "SCC",
        "desc": "Calories consumption monitoring."
    },
    {
        "fe_name": "FAF",
        "desc": "Physical activity frequency."
    },
    {
        "fe_name": "TUE",
        "desc": "Time using technology devices."
    },
    {
        "fe_name": "CALC",
        "desc": "The frequency of alcohol consumption."
    },
    {
        "fe_name": "MTRANS",
        "desc": "The primary mode of transportation used by the individual."
    },
    {
        "fe_name": "NObeyesdad",
        "desc": "The target variable, representing the individual's obesity level."
    }
]
```


Optiver The task is to predict short-term price movements of stocks using auction data. The dataset includes features like stock ID, date ID, imbalance size, reference price, bid and ask prices, and weighted average price (WAP). The target variable represents the 60-second future move in WAP relative to a synthetic index.

Listing 10: Dataset description for optiver.


```
[
    {
        "fe_name": "stock_id",
        "desc": "A unique identifier for the stock. Not all stock IDs exist in every time bucket."
    },
    {
        "fe_name": "date_id",
        "desc": "A unique identifier for the date. Date IDs are sequential & consistent across all
           stocks."
    },
    {
        "fe_name": "seconds_in_bucket",
        "desc": "The number of seconds elapsed since the beginning of the day's closing auction,
           always starting from 0."
    },
    {
        "fe_name": "imbalance_size",
        "desc": "The amount unmatched at the current reference price (in USD)."
    },
    {
        "fe_name": "imbalance_buy_sell_flag",
        "desc": "An indicator reflecting the direction of auction imbalance (1 for buy-side imbalance,
            -1 for sell-side imbalance, 0 for no imbalance)."
    },
    {
        "fe_name": "reference_price",
        "desc": "The price at which paired shares are maximized, the imbalance is minimized and the
           distance from the bid-ask midpoint is minimized, in that order. Can also be thought of as
           being equal to the near price bounded between the best bid and ask price."
    },
    {
        "fe_name": "matched_size",
        "desc": "The amount that can be matched at the current reference price (in USD)."
    },
    {
        "fe_name": "far_price",
        "desc": "The crossing price that will maximize the number of shares matched based on auction
           interest only. This calculation excludes continuous market orders."
    },
    {
        "fe_name": "near_price",
        "desc": "The crossing price that will maximize the number of shares matched based on auction
           and continuous market orders."
    },
    {
        "fe_name": "bid_price",
        "desc": "Price of the most competitive buy level in the non-auction book."
    },
    {
```

```
        "fe_name": "bid_size",
        "desc": "The dollar notional amount on the most competitive buy level in the non-auction book."
    },
    {
        "fe_name": "ask_price",
        "desc": "Price of the most competitive sell level in the non-auction book."
    },
    {
        "fe_name": "ask_size",
        "desc": "The dollar notional amount on the most competitive sell level in the non-auction book."
    },
    {
        "fe_name": "wap",
        "desc": "Weighted Average Price, calculated based on bid and ask prices and sizes. All price-related columns are converted to a price move relative to the stock WAP (Weighted Average Price) at the beginning of the auction period."
    },
    {
        "fe_name": "target",
        "desc": "The target variable. The 60 second future move in the WAP of the stock, less the 60 second future move of the synthetic index. The unit of the target is basis points, which is a common unit of measurement in financial markets. A 1 basis point price move is equivalent to a 0.01% price move."
    }
]
```

Smoke The task is to predict the smoking status of individuals based on their personal, physical, and health-related information. The dataset provides comprehensive records of various factors that can influence smoking habits, helping to understand and classify smoking status.

Listing 11: Dataset description for smoker status.

```
[
    {
        "fe_name": "id",
        "desc": "Unique identifier for each individual in the dataset."
    },
    {
        "fe_name": "age",
        "desc": "The age of the individual in years."
    },
    {
        "fe_name": "height_cm",
        "desc": "The height of the individual in centimeters."
    },
    {
        "fe_name": "weight_kg",
        "desc": "The weight of the individual in kilograms."
    },
    {
        "fe_name": "waist_cm",
        "desc": "Waist circumference length in centimeters."
    },
    {
        "fe_name": "eyesight_left",
        "desc": "Measurement of the individual's left eye vision, typically a decimal value where higher values indicate better vision."
    },
    {
        "fe_name": "eyesight_right",
        "desc": "Measurement of the individual's right eye vision, typically a decimal value where higher values indicate better vision."
    },
    {
        "fe_name": "hearing_left",
        "desc": "Measurement of the individual's left ear hearing ability, typically a decimal value where higher values indicate better hearing."
    },
    {
        "fe_name": "hearing_right",
        "desc": "Measurement of the individual's right ear hearing ability, typically a decimal value where higher values indicate better hearing."
    },
    {
        "fe_name": "systolic",
        "desc": "The systolic blood pressure of the individual."
    },
    {
        "fe_name": "relaxation",
        "desc": "The diastolic blood pressure of the individual."
    },
    {
        "fe_name": "fasting_blood_sugar",
        "desc": "The fasting blood sugar level of the individual."
    },
    {
```

```
        "fe_name": "cholesterol",
        "desc": "The cholesterol level in the individual's blood."
    },
    {
        "fe_name": "triglyceride",
        "desc": "The triglyceride level in the individual's blood."
    },
    {
        "fe_name": "HDL",
        "desc": "The high-density lipoprotein (good cholesterol) level in the individual's blood."
    },
    {
        "fe_name": "LDL",
        "desc": "The low-density lipoprotein (bad cholesterol) level in the individual's blood."
    },
    {
        "fe_name": "hemoglobin",
        "desc": "The hemoglobin level in the individual's blood."
    },
    {
        "fe_name": "urine_protein",
        "desc": "The protein level in the individual's urine."
    },
    {
        "fe_name": "serum_creatinine",
        "desc": "The serum creatinine level in the individual's blood."
    },
    {
        "fe_name": "AST",
        "desc": "The aspartate aminotransferase (AST) level in the individual's blood."
    },
    {
        "fe_name": "ALT",
        "desc": "The alanine aminotransferase (ALT) level in the individual's blood."
    },
    {
        "fe_name": "Gtp",
        "desc": "The gamma-glutamyl transferase (GTP) level in the individual's blood."
    },
    {
        "fe_name": "dental_caries",
        "desc": "Whether the individual has dental caries (1 for yes, 0 for no)."
    },
    {
        "fe_name": "smoking",
        "desc": "The smoking status of the individual (1 for smoker, 0 for non-smoker). This is the
            target variable we are trying to predict."
    }
]
```

Soft The task is to predict the presence of defects in software code based on various code metrics and characteristics. The dataset provides detailed records of code features, helping to understand and classify code quality.

Listing 12: Dataset description for software defects.

```
[
    {
        "fe_name": "id",
        "desc": "Unique identifier for each record in the dataset."
    },
    {
        "fe_name": "loc",
        "desc": "McCabe's line count of code."
    },
    {
        "fe_name": "v_g",
        "desc": "McCabe's cyclomatic complexity."
    },
    {
        "fe_name": "ev_g",
        "desc": "McCabe's essential complexity."
    },
    {
        "fe_name": "iv_g",
        "desc": "McCabe's design complexity."
    },
    {
        "fe_name": "n",
        "desc": "Halstead's total operators + operands."
    },
    {
        "fe_name": "v",
        "desc": "Halstead's volume."
    },
    {
        "fe_name": "l",
        "desc": "Halstead's program length."
```

```
    },
    {
        "fe_name": "d",
        "desc": "Halstead's difficulty."
    },
    {
        "fe_name": "i",
        "desc": "Halstead's intelligence."
    },
    {
        "fe_name": "e",
        "desc": "Halstead's effort."
    },
    {
        "fe_name": "b",
        "desc": "Halstead's metric related to program bugs or errors."
    },
    {
        "fe_name": "t",
        "desc": "Halstead's time estimator, representing the estimated time required to write the code."
    },
    {
        "fe_name": "lOCode",
        "desc": "Halstead's line count of code, representing the number of lines of code."
    },
    {
        "fe_name": "lOComment",
        "desc": "Halstead's count of lines of comments, representing the number of comment lines."
    },
    {
        "fe_name": "lOBlank",
        "desc": "Halstead's count of blank lines, representing the number of empty lines."
    },
    {
        "fe_name": "locCodeAndComment",
        "desc": "Combined count of lines of code and comments."
    },
    {
        "fe_name": "uniq_Op",
        "desc": "The number of unique operators in the code."
    },
    {
        "fe_name": "uniq_Opnd",
        "desc": "The number of unique operands in the code."
    },
    {
        "fe_name": "total_Op",
        "desc": "The total number of operators in the code."
    },
    {
        "fe_name": "total_Opnd",
        "desc": "The total number of operands in the code."
    },
    {
        "fe_name": "branchCount",
        "desc": "The number of branches in the flow graph, indicating the complexity of the control flow."
    },
    {
        "fe_name": "defects",
        "desc": "Module has/has not one or more reported defects. This is the target variable we are trying to predict."
    }
]
```

Spac The task is to predict whether a passenger was transported to an alternate dimension during the Spaceship Titanic's collision with the spacetime anomaly. To help you make these predictions, you're given a set of personal records recovered from the ship's damaged computer system.

Listing 13: Dataset description for spaceship titanic.

```
[
    {
        "fe_name": "PassengerId",
        "desc": "A unique Id for each passenger. Each Id takes the form gggg_pp where gggg indicates a group the passenger is travelling with and pp is their number within the group. People in a group are often family members, but not always."
    },
    {
        "fe_name": "HomePlanet",
        "desc": "The planet the passenger departed from, typically their planet of permanent residence."
    },
    {
        "fe_name": "CryoSleep",
        "desc": "Indicates whether the passenger elected to be put into suspended animation for the duration of the voyage. Passengers in cryosleep are confined to their cabins."
```

```
    },
    {
        "fe_name": "Cabin",
        "desc": "The cabin number where the passenger is staying. Takes the form deck/num/side, where side can be either P for Port or S for Starboard."
    },
    {
        "fe_name": "Destination",
        "desc": "The planet the passenger will be debarking to."
    },
    {
        "fe_name": "Age",
        "desc": "The age of the passenger."
    },
    {
        "fe_name": "VIP",
        "desc": "Whether the passenger has paid for special VIP service during the voyage."
    },
    {
        "fe_name": "RoomService",
        "desc": "Amount the passenger has billed at the RoomService, one of the spaceship's luxury amenities."
    },
    {
        "fe_name": "FoodCourt",
        "desc": "Amount the passenger has billed at the FoodCourt, one of the spaceship's luxury amenities."
    },
    {
        "fe_name": "ShoppingMall",
        "desc": "Amount the passenger has billed at the ShoppingMall, one of the spaceship's luxury amenities."
    },
    {
        "fe_name": "Spa",
        "desc": "Amount the passenger has billed at the Spa, one of the spaceship's luxury amenities."
    },
    {
        "fe_name": "VRDeck",
        "desc": "Amount the passenger has billed at the VRDeck, one of the spaceship's luxury amenities."
    },
    {
        "fe_name": "Name",
        "desc": "The first and last names of the passenger."
    },
    {
        "fe_name": "Transported",
        "desc": "Whether the passenger was transported to another dimension. This is the target, the column you are trying to predict."
    }
]
```

J Details of Prompts

In Listing 14, We use the core prompts provided below to guide an LLM in analyzing the syntax tree of a given feature and the dataset's content. LLM first identifies and interprets all subtrees. Then, it generates improvement options for each subtree, providing corresponding methodological justifications.

Listing 14: The prompt for expression generation.

```
You are an expert in the field of feature engineering. The purpose of feature engineering is to construct new features and iteratively optimize these features to obtain a set of high-quality expert features, reduce the relationship between input and output, and improve the model prediction score. There is an existing feature:
{expr}
The syntax tree of this feature is:
{ast}
Please perform the following operations in sequence:
(1) List all the subtrees of this tree (excluding individual leaf nodes), analyze the meaning of each subtree, and explain the reason for designing this subtree in this way.
(2) Provide all possible local improvement options and corresponding ideas for each subtree, in the format: If there is xxx (data distribution level) problem, yyy local optimization method should be used to solve it.
Note:
- ** For step (2), when designing improvement options for a certain operator node, you can only operate in the neighborhood of this node, including designing a wrapper for this node, replacing the operator of this node, or adding/deleting/modifying operands for this node. But you cannot modify any downstream operators of this node, nor can you modify any operator nodes at the same level as this node. ** For example, when designing modification options for the groupby node, do not attempt to modify any operators or operands in its key, value, or operator child nodes, such as adding or deleting grouping criteria; when designing modification options for the groupby-value node, do not attempt to modify its sibling nodes, groupby-key/operator.
The improvement operations you can choose are:
{operators}
```

```

The content of the dataset is as follows:
{raw_data_info}

Please ensure that your output follows the following JSON format and that you can list all subtrees
    ↪completely, including {subtree_names}:
[
    "analysis_of_syntax_tree_and_subtrees": [
      {{
        "subtree_name": "Subtree 1 - Name",
        "subtree_analysis": "Analysis of the meaning of this subtree",
      }}
      ...
    ],
    "subtree_improvement_options": [
      {{
        "subtree_name": "Subtree 1 - Name",
        "improvement_suggestions": [
          {{
            "improvement_option": 1,
            "operator": "Which improvement operation to use",
            "condition": "If there is xx data distribution problem, consider doing xxx optimization (
                ↪not involving specific feature names)"
          }},
          {{
            "improvement_option": 2,
            "operator": "Which improvement operation to use",
            "condition": "..."
          }}
        ]
      }},
      {{
        "subtree_name": "Subtree 2 - Name",
        "improvement_suggestions": [
          {{
            "improvement_option": 1,
            "operator": "Which improvement operation to use",
            "condition": "..."
          }},
          {{
            "improvement_option": 2,
            "operator": "Which improvement operation to use",
            "condition": "..."
          }}
        ]
      }}
      ...
        ]
      }}
]
```

In Listing 15, we use this prompt to guide the LLM to perform step-by-step reasoning using the provided data distribution to refine the feature optimization scheme and provide a specific local optimization expression.

Listing 15: The prompt for feature optimization.

```
You are an expert-level data analyst participating in a feature generation task to generate new
    ↪features for a tabular dataset and iteratively improve their quality.
The original feature set and related data distribution of the tabular dataset are as follows:
{raw_data_info}
There is a new feature with the expression: {expr}
The overall task is to iteratively improve this feature through local optimizations. The available
    ↪optimization operators are:
{operators}
There is a local optimization scheme (improvement_option) designed for this feature:
{options}

Your tasks are as follows:
(1) Perform step-by-step reasoning based on the data distribution and provide a specific
    ↪implementation method for the optimization scheme.
(2) Provide a specific local optimization expression.

Please note:
(1) The optimization scheme and its implementation should focus on feature engineering, that is,
    ↪improving the quality of features by combining features and operators. Therefore, any data
    ↪preprocessing tools such as missing value imputation, feature encoding, standardization, extreme
    ↪ value scaling, etc., should not be involved. Even if the option mentions doing similar
    ↪operations, it should be achieved through feature engineering methods.
(2) Your task is to improve the above new feature, not to construct another new feature based on it.
    ↪Therefore, you cannot use old features to participate in feature construction (otherwise, an
    ↪error will occur because the feature cannot be found in the dataframe).
Please ensure that your output follows the following JSON format:
[
  "selected_option": {{
    "thought": "Perform step-by-step reasoning based on the data distribution and provide a specific
        ↪implementation method for the optimization scheme",
  }}
]
```

In Listing 16, our prompt directs the LLM to apply a provided local optimization scheme to a new feature's expression and its abstract syntax tree (AST). The optimization is to be executed using a simplified subset of Python syntax, and the result should include the updated expression and AST.

Listing 16: The prompt for execution.

```
You are an expert-level feature engineer participating in a feature generation task to generate new features for a tabular dataset and iteratively improve their quality.

There is a new feature with the expression: {expr}
Its corresponding abstract syntax tree structure is as follows:
{ast}
A feature engineering expert has analyzed the overall and local structure of the new feature in detail and provided an optimization scheme:
{action}

Your tasks are as follows:
(1) Execute the local optimization scheme provided by the expert and provide the expression and abstract syntax tree structure after execution.

The expression uses a simplified subset of Python syntax, with the specific syntax as follows:
Discretization: Bin(feature, original_margin, target_set)
Arithmetic operations: feature1 +-*/ feature2
Grouping operations: groupby(key, value, statistic_operator)
Note: statistic_operator in ['min', 'max', 'mean', 'medium', 'std', 'rolling_mean', 'first', ....]
Time series processing: TimeExtract(feature, original_format, target_format)
String processing: StrExtract(feature, original_format_desc, target_format_desc)
Loop: for name in [feature1, feature2, ...]

When generating the syntax tree, please note:
1. You are not allowed to use any other expressions not listed.
2. The structure style of the syntax tree should be consistent with the original syntax tree. Please note that you should minimize the difference between the new feature syntax tree and the original feature syntax tree.

When generating the expression of the new feature, please note:
1. The syntax tree is represented by a tree hierarchy connected by vertical and horizontal lines, where each branch node corresponds to an operator, and each leaf node corresponds to a feature.
2. Unless the syntax tree contains a for loop, the entire syntax tree constructs only one new feature. If there are multiple parallel feature nodes (leaf nodes) among the child nodes of a branch node, they are represented as brunch_node: leaf_node1, leaf_node2, ..., and these nodes are connected by their parent node operator (such as Add, Mul). If the parent node is a keyword of an operator, such as groupby-key, for-iter, these features are included in the same list.

Please ensure that your output follows the following JSON format:
{{
    "new_ast": "The updated syntax tree, a tree structure connected by vertical and horizontal lines. Each level is represented by |---and |___, and the nodes at each level are arranged in hierarchical order. While ensuring the implementation of the optimization scheme, the structure of the syntax tree should be as consistent as possible with the original feature's syntax tree.",
    "new_feature_expression": "The expression of the new feature, following the above Python syntax subset, without any function definitions, library imports, or any unlisted functions and operations.",
}}
```

In Listing 17, the prompt directs the LLM to evaluate and refine feature optimization strategies for a tabular dataset. It requires the model to analyze the shortcomings of previously attempted optimization methods in relation to the data distribution, summarize key takeaways from these failures, and update the original optimization option without altering its fundamental position or type.

Listing 17: The prompt for continual refinement through self-reflection.

```
You are an expert-level data analyst participating in a feature generation task to generate new features for a tabular dataset and iteratively improve their quality.
The original feature set and data distribution of the tabular dataset are as follows:
{raw_data_info}
There is a new feature with the expression: {expr}
Its syntax tree (AST) structure is:
{ast}
The overall task is to iteratively improve this feature through local optimizations. The available optimization operators are:
{operators}
A feature engineering expert has analyzed the overall and local structure of the new feature in detail and provided a local optimization option:
{improvement_option}
and tried several specific implementation methods:
{specific_method}
We applied these implementation methods to improve the new feature and tested its effect on downstream machine learning models, but found that the scheme did not result in a positive improvement for the new feature.
Your task is to:
(1) Reflect on the shortcomings of the specific implementation methods in combination with the data distribution.
(2) Summarize the experience and update the option with the lessons learned from the failed attempts.

Please note:
```

```
(1) When updating the optimization option, you must not change the local position of the optimization or the type of optimization operator. For example, if the original option replaces the feature of the key subtree, then your improved scheme should still try to replace the feature of the key subtree, but just adjust the specific replacement method, such as trying to replace it with other features.
(2) Your task is not to improve this option, but to summarize the lessons learned from the failed attempts to prevent making the same mistakes in the next implementation. Therefore, you cannot change the meaning of the original option, nor can you directly provide a new specific implementation method.
(3) Your reflection process should be as rigorous as possible, analyzing only the shortcomings of the attempted schemes without extending to similar untried schemes. For example, when you find that adding a certain feature A is ineffective, you should point out that A is ineffective, but do not infer that adding a similar feature B is also ineffective. When you find that replacing features C and D is ineffective, you should point out that replacing both C and D simultaneously is ineffective, but do not infer that replacing C and replacing D are both ineffective.
Please ensure that your output follows the following JSON format:
{{
"specific_method_summary": "Description of the several ineffective implementation methods that have been tried",
"specific_method_reflection": "Reflection on why these methods were ineffective, analyzing the shortcomings of the existing implementation methods in combination with the data distribution, and providing some improvement suggestions",
"updated_option": "Updated feature optimization scheme, in the format: The original option was xxx. For this option, we have tried xxx and other schemes, but they were ineffective. We believe this is because xxx. Next, we suggest xxx (directions to try).",
}}
```