Dynamic and Adaptive Feature Generation with LLM

Xinhao Zhang[1] , **Jinghan Zhang**[1] , **Banafsheh Rekabdar**[1] , **Yuanchun Zhou**[2] , **Pengfei Wang**[2,3*] and **Kunpeng Liu**[1]
[1]Portland State University
[2]Computer Network Information Center, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences, Chinese Academy of Sciences
{xinhaoz, jinghanz, kunpeng, rekabdar}@pdx.edu, {zyc, pfwang}@cnic.cn

Abstract

The representation of feature space is a crucial environment where data points get vectorized and embedded for subsequent modeling. Thus, the efficacy of machine learning (ML) algorithms is closely related to the quality of feature engineering. As one of the most important techniques, feature generation transforms raw data into an optimized feature space conducive to model training and further refines the space. Despite the advancements in automated feature engineering and feature generation, current methodologies often suffer from three fundamental issues: lack of explainability, limited applicability, and inflexible strategy. These shortcomings frequently hinder and limit the deployment of ML models across varied scenarios. Our research introduces a novel approach adopting large language models (LLMs) and feature-generating prompts to address these challenges. We propose a dynamic and adaptive feature generation method that enhances the interpretability of the feature generation process. Our approach broadens the applicability across various data types and tasks and offers advantages in terms of strategic flexibility. A broad range of experiments showcases that our approach is significantly superior to existing methods.

1 Introduction

The success of machine learning (ML) algorithms generally depends on three aspects: data processing, feature engineering, and modeling [Jordan and Mitchell, 2015]. Among these, feature engineering is critical, directly influencing ML models' performance and effectiveness. Within feature engineering, feature generation is a vital process of transforming raw features into a structured format and optimizing the feature space by creating new features from the original ones [Nam *et al.*, 2024]. This transformation usually involves mathematical or algorithmic operations on existing features. This optimization can significantly enrich the feature space, enabling ML models to draw and utilize data information more effectively and achieve superior performance.

*Corresponding author.

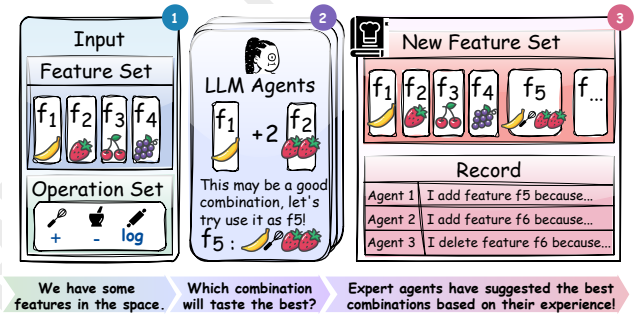

Figure 1: Our goal is to iteratively reconstruct the feature space to create an optimal and explainable feature set that enhances performance on downstream machine learning tasks.

Given the importance of feature engineering and feature generation, extensive research has focused on enhancing both the effectiveness and efficiency of these methods [Mumuni and Mumuni, 2024; Gu *et al.*, 2024]. As deep learning has explosively developed, automated feature engineering has emerged as a mainstream approach due to its convenience and remarkable performance. These auto-engineering algorithms greatly reduce the need for manual calculation and determination, and thus reduce subjective errors and mistakes. In the field of automated feature generation, there are outstanding works such as [Khurana *et al.*, 2018; Zhang *et al.*, 2024b; Wang *et al.*, 2022a]. These methods leverage advanced techniques like reinforcement learning to generate optimal feature sets that enhance the performance of downstream tasks while maintaining a degree of explainability. Despite these methods enriching the feature space and restructuring it into a more refined space structure, like other automated feature generation and broader automated feature engineering techniques, these works have not resolved three fundamental issues: 1) **Lack of Explainability:** Most feature engineering processes that utilize deep learning algorithms are result-oriented "black boxes". The specific operations on features and the process of restructuring feature space are usually hard for researchers to understand. 2) **Limited Applicability:** For different data types and downstream tasks, researchers need to manually select and switch among a wide range of feature engineering methods, which demands a high level of domain knowledge

and expertise from the researchers. This limited applicability can be a significant barrier across varied ML scenarios. 3) **Fixed Strategy:** Existing methods usually require predefined strategies for exploration and generation. Models utilizing these methods are unable to adapt or refine their strategies based on insights acquired about the feature space during the model's learning process. Hence, these models lack the flexibility to dynamically adjust to the evolving understanding of the data.

Our Targets. We aim to develop an automated feature generation method that: (1) offers a **transparent and understandable** feature generation process; (2) adapts to various data types and downstream tasks by selecting best-suited operations **automatically**; (3) continuously learns from the feature space and **refines its strategies** based on new insights. For target (1), we explicitly document the operations applied during the feature generation and how they transform the input data into a structured feature space. By doing so, we intend to replace their lack of interpretability with a more open and interpretable framework. For target (2), we apply a large language model (LLM) that can assess the characteristics of the data and the requirements of the task and dynamically select the most appropriate strategies under proper guidance. For target (3), we develop multiple expert-level LLM agents with different strategies. These agents adjust their operations and strategies in response to evolving data patterns, task requirements, and actions of other agents.

Our Method. We employ a novel approach to automate and enhance the feature generation process by adopting LLMs with expert-level guidance. Our method starts by inputting an original feature set and a predefined operation set into the LLM, which then instantiates several expert agents. Each agent generates new features with its exclusive strategy and integrates these new features with the original features to produce enriched feature sets. The newly created feature sets are applied to downstream tasks to evaluate their performance, and the performance outcomes are fed back to the agents. The agents "communicate" and share their inferences, and reflect on the impact of their own and other agents' operations on the downstream tasks. In this way, the agents refine and optimize their generation strategies to better capture how variations in the feature space impact downstream tasks and, consequently, reconstruct the feature space more adaptively and effectively. The iteration continues with new feature sets until the optimal feature set for the task is found or a predefined number of iterations is reached.

Contributions. Our main contributions include:

- We propose a novel automated feature generation methodology based on LLMs to restructure the feature space of a dataset. This end-to-end structure enables training with LLM agents on varied feature generation strategies without manual selection of feature operations.
- We introduce a dynamic and adaptive generation process based on feedback from downstream tasks. This method effectively utilizes the in-context learning and reasoning capabilities of LLMs in the feature generation process. It significantly enhances the applicability and effectiveness of the generated features across various machine learning scenarios without the need to create additional ML models.
- We conduct a series of experiments to validate the effectiveness and robustness of our method across different datasets and downstream tasks. Our results demonstrate that our method has clear advantages over existing methods and considerable potential to promote a broader range of feature engineering tasks.

2 Preliminary and Related Work

2.1 Feature Generation

Feature Engineering. Feature engineering is the process of selecting, modifying, or creating new features from raw data to improve the performance of machine learning models [Severyn and Moschitti, 2013]. This process can be described as a function $\phi : \mathcal{F} \rightarrow \mathcal{F}^{'}$ that transforms the original feature set $\mathcal{F}$ into a new feature set $\mathcal{F}^{'}$ through certain operations. Feature generation is a decisive category in the field of feature engineering. Feature generation means generating new attributes from existing features in a dataset through various mathematical or logical transformation operations. Feature generation methods can primarily be categorized into two main classes: (1) latent representation learning based methods, such as deep factorization machines [Guo *et al.*, 2017; Song *et al.*, 2019] and deep representation learning [Zhong *et al.*, 2016; Bengio *et al.*, 2013]. These methods can create complex latent feature spaces; however, their generation processes often lack transparency and are difficult to trace or explain [Schölkopf *et al.*, 2021]; (2) feature transformation-based methods, which generate new features by arithmetic or aggregation operations [Nargesian *et al.*, 2017; Wang *et al.*, 2022c; Wang *et al.*, 2022b]. These methods often require extensive manual operations and rely heavily on domain knowledge to select transformations and adjust parameters.

Automated Feature Generation. With the widespread adoption of large models and various deep learning methods in the feature engineering field, automated feature generation is experiencing significant development. Automated feature generation enhances the feature space by systematically creating and integrating new features to improve model performance [Xiang *et al.*, 2021; Wang *et al.*, 2022a; Pan *et al.*, 2020; Wang *et al.*, 2022a]. For example, [Katz *et al.*, 2016] developed ExploreKit, which generates a large set of candidate features by combining information from the original features. [Khurana *et al.*, 2016] explores the feature space using manually crafted heuristic traversal strategies, while [Shi *et al.*, 2018] proposed a feature optimization approach using deep learning and feature selection to enhance traffic classification performance. Although these methods are more efficient than traditional manual feature engineering and enable faster processing of large datasets, they often ignore the semantic aspects of data. Furthermore, their "black box" operation process makes the results difficult to explain.

2.2 LLMs and Tree of Thoughts

LLMs' Capabilities. LLMs have revolutionized numerous fields with their extraordinary capabilities in in-context learning and step-by-step reasoning [Xie *et al.*, 2024; Zhang *et al.*, 2024a; Wei *et al.*, 2022]. These capabilities enable LLMs to effectively understand and navigate complex tasks given appropriate guidance and precise prompts. Thus, they exhibit professional competence across various specialized domains [Wang *et al.*, 2023; Zhang *et al.*, 2024c; Zhang *et al.*, 2025a; Wang *et al.*, 2025]. Recent advancements in prompt engineering and reasoning structures have further enabled these models to tackle complex data analysis tasks and multi-step feature engineering processes [Peng *et al.*, 2023; Liu *et al.*, 2023].

Prompting. LLM prompting leverages pre-trained models to recall existing knowledge and generate output based on specific guideline prompts [White *et al.*, 2023]. This process interacts closely with LLMs' capabilities for in-context learning and step-by-step reasoning [Chu *et al.*, 2023]. Prompts containing context details guide models to think more coherently when handling long tasks. Moreover, step-by-step reasoning prompts can guide a model through logical thought processes by mimicking human-like reasoning to solve complex problems [Yu *et al.*, 2023; Wei *et al.*, 2022; Diao *et al.*, 2023].

Tree of Thoughts. One line of work in prompting methods focuses on enhancing the structured reasoning capabilities of LLMs to guide them through logical analysis and problem-solving tasks. *Input-Output (IO) Prompting* [Wang *et al.*, 2024; Hao *et al.*, 2023] involves wrapping the input x with task-specific instructions or examples to guide the model to produce the desired output y. This method is straightforward, but its guiding capability is limited in complex tasks. *Chain-of-Thought (CoT) Prompting* [Wei *et al.*, 2022; Besta *et al.*, 2024; Zhang *et al.*, 2025c] involves a sequence of intermediate, coherent language expressions $z_1, \ldots, z_n$ that logically bridge the input x to the output y. This method enhances the model's logical capabilities for complex problem-solving, but it is limited to linear, single-strategy reasoning, capturing only a subset of the possible solutions. *Tree of Thoughts (ToT) Prompting* [Yao *et al.*, 2024] extends the CoT prompting by exploring multiple reasoning paths over thoughts. ToT operates as a search over a tree structure where each node represents a partial solution within the input and contextual thoughts. ToT has advantages over other methods by exploring multiple reasoning paths with different strategies [Zhang *et al.*, 2025b]. Thus, it performs better in deep data analysis and multi-step feature engineering.

3 Methodology

In this section, we design a novel feature generation method called **L**LM **F**eature **G**eneration (**LFG**), which is highly interpretable, dynamically adaptable, and employs an end-to-end approach to significantly enhance downstream task performance. In this method, we apply an LLM as an auto-generator of features. We augment the LLM with the ToT technique to generate detailed inference along with each step of generation so that each decision is explainable. Furthermore, we collect the performance of downstream tasks to offer feedback to the LLM generator. Our entire workflow is divided into two main parts: (1) **Automated Feature Generation with LLM Agents**; (2) **Feedback and Optimization**.

3.1 Important Definitions

We first define the feature set and operation set, along with their corresponding mathematical symbols.

Feature Set. Let $\mathcal{D} = \{\mathcal{F}, \mathbf{y}\}$ be the dataset. Here, $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is the feature set, where each column represents a feature f_i, and each row represents a data sample; $\mathbf{y}$ is the corresponding target label set for the samples. Our main purpose is to reconstruct the feature space of the dataset $\mathcal{D} = \{\mathcal{F}, \mathbf{y}\}$.

Operation Set. The operation set $\mathcal{O}$ consists of mathematical operations performed on existing features to generate new ones. Operations are of two types: unary and binary. Unary operations include "square", "exp", "log", etc.; binary operations include "plus", "multiply", "divide", etc.

Optimization Objective. Our objective is to construct an optimal and interpretable feature space that can enhance the performance of downstream tasks. Given the feature set $\mathcal{F}$ and the operation set $\mathcal{O}$, our optimization goal is to find a reconstructed feature set $\hat{\mathcal{F}}$:

$$\mathcal{F}^* = \underset{\hat{\mathcal{F}}}{\operatorname{argmax}}\ \theta_{\mathcal{R}}(\hat{\mathcal{F}}, \mathbf{y}), \tag{1}$$

where $\mathcal{R}$ is a downstream ML task (e.g., classification, regression, ranking, detection), θ is the performance metric of $\mathcal{R}$, and $\hat{\mathcal{F}}$ is a reconstructed feature set derived from $\mathcal{F}$. Here, $\hat{\mathcal{F}}$ is generated by applying operations from $\mathcal{O}$ to the original feature set $\mathcal{F}$ using a certain algorithmic structure.

3.2 Automated Feature Generation with LLM Agents

In this section, we introduce the concept of LLM agents and detail their role in feature generation. We utilize an LLM to generate several agents for feature-generation tasks. Here, each agent acts as an automated feature generator, which applies specific operations on a given feature set to generate new features or delete existing features based on iterative prompts. These agents can emulate expert-level logical reasoning and decision-making capabilities, thus optimizing the feature set for input data of downstream tasks and enhancing their performance.

Agents. We first define an agent $\mathcal{A}_l$ that operates on a feature subset $\mathcal{F}_l = \{f_1, \ldots, f_n\}$ by applying operations to the features f_i and f_j. The $\mathcal{F}_l$ is a subset of $\mathcal{F}$. The agent draws operations from an operation set $\mathcal{O}$, to generate new features:

$$\mathcal{F}_l^{'} = \mathcal{F}_l \cup \{g_1, \ldots, g_k, \ldots\}. \tag{2}$$

Each new feature g_k is produced by applying an operation o to the features f_i and f_j:

$$g_k = o(f_i, f_j). \tag{3}$$

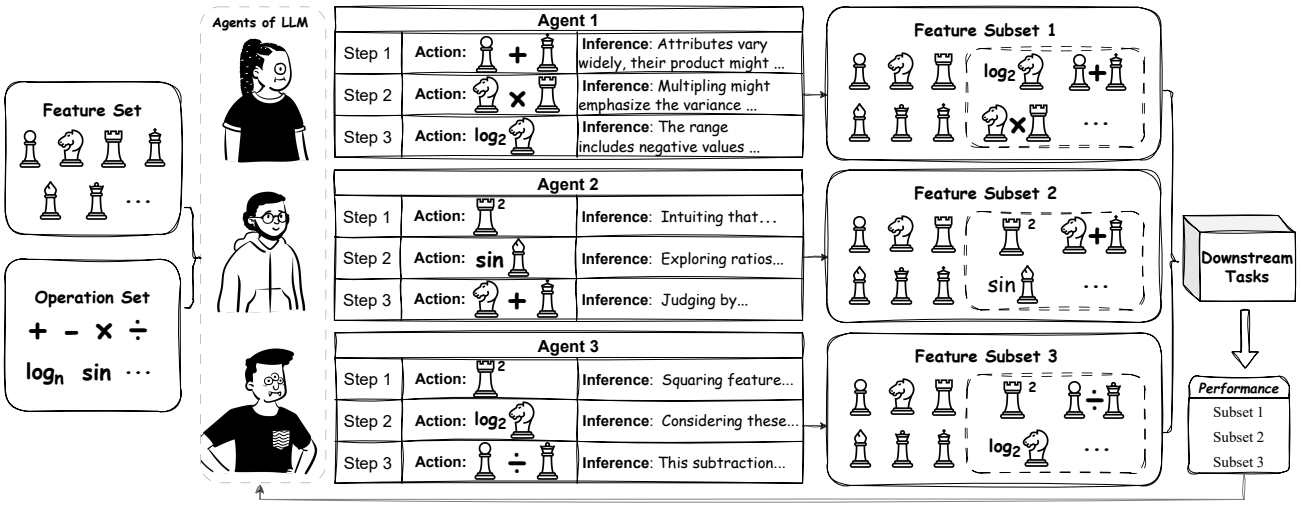

Figure 2: The framework of LFG. First, we input the original feature set and operation set into the LLM's context window. Second, we guide the LLM in creating three expert agents. Each agent generates new features with operations from the operation set, and then combines new features with original features to create a new feature subset. Then, each of these subsets is individually evaluated on a downstream task. After that, we provide the performance of each feature subset in downstream tasks as feedback to the respective agents and iterate the process until the best feature subset is found or the maximum number of iterations is reached.

Here, $\mathcal{F}_l$ is the initial feature subset for $\mathcal{A}_l$; $\mathcal{F}_l^{'}$ is the output feature subset with original and all new features from $\mathcal{A}_l$; and g_k is the new feature from applying operation o to features $f_i, f_j \in \mathcal{F}_l$. Thus, we complete one generation node s_l of $\mathcal{A}_l$, and layer $Layer_1$ consists of l total generation nodes.

3.3 Feedback and Optimization

We now have a new feature set composed of all l subsets, denoted as $\mathcal{F}_l^{'} = \{\mathcal{F}_l, g_1, \ldots, g_k\}$. After the agents generate all l subsets, we move forward to evaluate the performance of these subsets in downstream tasks and the effectiveness of the generated nodes at this layer. We then gather performance metrics from downstream tasks as feedback $\theta = \{\theta_t\}_{i=1}^{T}$, for each iteration t corresponding to each generation layer. This feedback reflects the effectiveness of the newly created feature subset and illustrates the agents' contribution to the task. The agents evaluate their performance by comparing the metric θ_t with $\theta_{(t-1)}$. If θ_t significantly improves over $\theta_{(t-1)}$, the feature generation strategy is considered "effective". The agents' self-evaluation positively correlates with the increase in these metrics; a larger improvement signifies a more successful feature generation strategy.

Furthermore, each agent shares its reasoning process during generating features, including the logic behind selecting specific operations. This transparency enables agents to understand and adopt diverse strategies to achieve collaborative improvement. Through self-evaluation and peer interactions, agents identify which strategies successfully enhanced performance and which may require adjustments or replacements. In the subsequent rounds, agents generate increasingly effective feature subsets adopting $\mathcal{F}_l^{'}$ (from the previous round) and $\mathcal{O}$ to form the next layer of generated nodes. The iterative process continues until the optimal feature set $\hat{\mathcal{F}}$ is found or the maximum number of iterations T is reached:

$$\hat{\mathcal{F}} = \underset{\mathcal{F} \in \{\mathcal{F}_l^{\text{opt}} | l \in L\}}{\arg\max} \ \theta(\mathcal{F}) \tag{4}$$

where $\{\mathcal{F}_l^{\text{opt}} \mid l \in L\}$ is the set of optimal feature subsets from each iteration l, and $\theta(\mathcal{F})$ is the performance score of a feature set $\mathcal{F}$.

Monte Carlo Tree Search. In order to further optimize the feature space and balance exploration with exploitation, we employ an improved Monte Carlo Tree Search (MCTS) [Yao *et al.*, 2024; Browne *et al.*, 2012] to explore new or underutilized feature space structures after the initial T iterations. We adopt this MCTS framework because of its random sampling-based search strategy, which enhances the probability of finding superior feature combinations. In our case, this flexible searching method constructs a tree from all agents' generation nodes and adjusts the search direction based on the performance feedback. Here, we define the root node as the original feature set given to the LLM, with each generation node representing a tree node. The search covers all generation nodes in the last iteration layer $Layer_T$ as leaf nodes.

We start with node evaluation by calculating the improvement in the downstream task's performance compared to the parent node. The evaluation reflects the effectiveness of newly generated features. Then, the MCTS iteratively builds a search tree by cycling through four phases: selection, expansion, evaluation, and searching. In the selection phase, children nodes are recursively selected from the root utilizing the Upper Confidence Bound (UCB) [Auer *et al.*, 2002]:

$$UCB(i) = w_i + C\sqrt{\frac{2\ln s_i^{'}}{s_i}}, \tag{5}$$

where s_i and $s_i^{'}$ are the visit counts for node i and its parent respectively, C is a hyperparameter balancing exploration

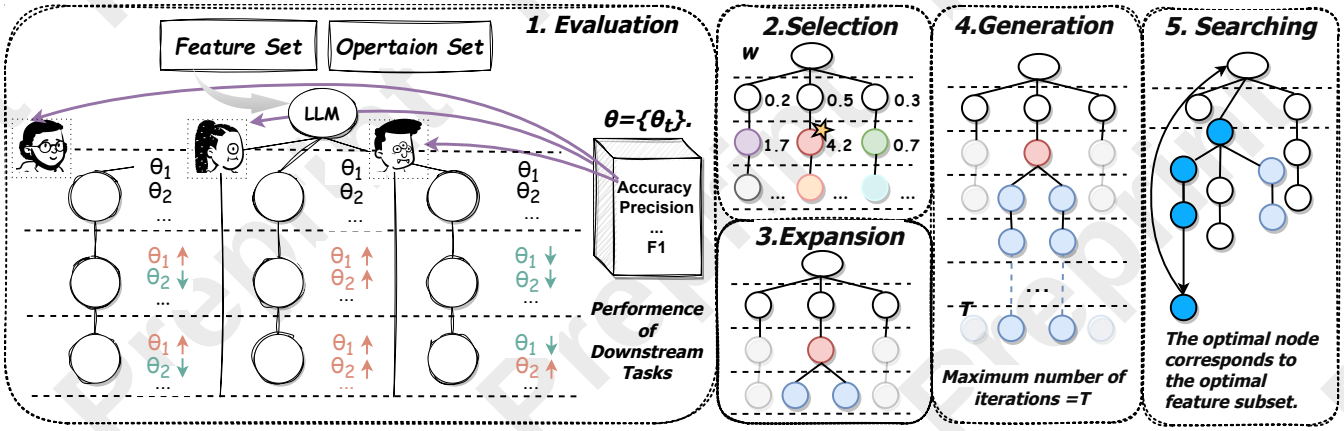

Figure 3: The framework of MCTS for feature generation includes five stages: 1) Performance Evaluation, 2) Node Selection, 3) Node Expansion, 4) Node Generation, 5) Optimal Subset Searching.

and exploitation. The value w_i represents the average performance improvement of all descendant nodes of node i, computed based on changes in the downstream task performance metric θ. Specifically, for each node i, the value of w_i is calculated as the average performance improvement observed from its child nodes in the downstream tasks. This is given by:

$$w_i = \frac{1}{|\mathcal{C}_i|} \sum_{j \in \mathcal{C}_i} (\theta_t^{(j)} - \theta_{(t-1)}^{(j)}). \tag{6}$$

Here, $\mathcal{C}_i$ is the set of child nodes of node i, and $|\mathcal{C}_i|$ is its cardinality (i.e., the number of child nodes of i). $\Delta\theta^{(j)} = (\theta_t^{(j)} - \theta_{(t-1)}^{(j)})$ represents the performance improvement contributed by child node $j \in \mathcal{C}_i$, where $\theta_t^{(j)}$ and $\theta_{(t-1)}^{(j)}$ are the performance metrics associated with the state after exploring child j and the state before, respectively.

In the tree, we select the most promising node s_l. We define the selection criterion for each node as:

$$s_l^{\text{select}} = \arg\max_{s_l \in L} \{\phi(s_l) + \psi(s_l)\}, \tag{7}$$

where $\psi(s_l)$ and $\phi(s_l)$ represent the current and future value of the node. We apply further strategies to generate and expand feature subsets upon selecting these nodes.

On the selected nodes s_l^{select}, more complex or novel feature operations are implemented to generate additional feature subsets. These newly generated subsets, $\mathcal{F}_l^{'}$, are then used for further evaluation in downstream tasks to validate their efficacy. The process is mathematically defined as:

$$\mathcal{F}_l' = \mathcal{G}(\mathcal{F}_l^{\text{select}}, \mathcal{O}'), \tag{8}$$

where $\mathcal{G}$ is the transformation function generating $\mathcal{F}_l^{'}$ by applying operations from a selected subset $\mathcal{O}' \subseteq \mathcal{O}$ (the predefined complete operation set) to $\mathcal{F}_l^{\text{select}}$. At the end of the MCTS process, the optimal feature subset is determined by choosing the leaf node with the highest w_i value, representing the feature set that maximized performance improvement during simulations:

$$\mathcal{F}^* = \mathcal{F}_{i^*}, \quad \text{where} \quad i^* = \arg\max_{i \in \text{LeafNodes}} w_i. \tag{9}$$

Datasets	Samples	Features	Class
Ion	351	34	2
Ama	1,500	10,000	2
Aba	4,177	8	3
Dia	441,455	330	2

Table 1: Datasets descriptions.

This $\mathcal{F}^*$ represents the optimal feature subset identified by the MCTS, which should then be validated further in real-world tasks to confirm its efficacy.

4 Experiments

In this section, we present three experiments to demonstrate the effectiveness of the LFG. First, we compare LFG against several baseline methods on multiple downstream classification tasks. Second, we perform a robustness check on LFG's performance improvement. Finally, we further study and analyze iterative performance improvements in experimental results and discuss their reasons.

4.1 Experimental Setup

Datasets. We evaluate the LFG method on four real-world datasets from UCI, including *Ionosphere (Ino)* [Sigillito *et al.*, 1989], *Amazon Commerce Reviews (Ama)* [Liu, 2011], and *Abalone (Aba)* [Nash *et al.*, 1995], as well as *Diabetes Health Indicators Dataset (Dia)* [Teboul, 2022] from Kaggle. The detailed information is shown in Table 1. For each dataset, we randomly selected 55% of the data for training.

Metrics. We evaluate the model performance by the following metrics: *Overall Accuracy (Acc)* measures the proportion of true results (both true positives and true negatives) in the total dataset. *Precision (Prec)* reflects the ratio of true positive predictions to all positive predictions for each class. *Recall (Rec)*, also known as sensitivity, reflects the ratio of true positive predictions to all actual positives for each class. *F-Measure (F1)* is the harmonic mean of precision and recall, providing a single score that balances both metrics.

Downstream Tasks. We apply the LFG model across a range of classification models, including *Random Forests (RF)*, *Decision Tree (DT)*, *K-Nearest Neighbor (KNN)* and *Multilayer Perceptrons (MLP)*. We compare the performance outcomes in these models both with and without our method.

Baseline Models. We compare the LFG method with raw data (Raw), the Least Absolute Shrinkage and Selection Operator (Lasso), and Feature Engineering for Predictive Modeling using Reinforcement Learning [Khurana *et al.*, 2018] (RL). Here, we set the same operation set for both LFG and RL method to consist of *square root, square, cosine, sine, tangent, exp, cube, log, reciprocal, sigmoid, plus, subtract, multiply*, and *divide*. For our LLM, we perform all the experiments on the OpenAI API, GPT-3.5 Turbo model [OpenAI, 2024].

4.2 Experimental Results

Overall Performance. Table 2 shows the overall performance results for LFG and the baseline models.

(1) Comparison with baseline models. From the table, we can see that the LFG method consistently surpasses baseline methods across a variety of metrics and datasets. Specifically, we show the results of LFG in 3 iterations of generation, denoted as LFG-3, and compare it with the full LFG in $T \leq 10$. On dataset *Ion*, the LFG achieves the best accuracy of 95.6%, which is a 6.4% increase over raw data on task RF, and 4.5% over the RL method. On dataset *Ama*, the LFG-3 improves the accuracy of raw data of 59.6% to 61.5% while LFG performs an increase of 4.3% on RF. Beyond accuracy, LFG consistently demonstrates significant improvements over baseline methods across other metrics and datasets. For instance, on dataset *Ion*, the highest recall improvement of 9.8% on DT reflects LFG's ability to effectively handle positive samples. Similarly, the improvements in F1, such as increasing from 0.476 to 0.607 on dataset *Ama* on KNN and from 0.855 to 0.932 on dataset *Ion* on DT, showcase the model's adaptability across diverse tasks and feature distributions.

(2) Performance across different models and metrics. In precision, the LFG also shows superior performance. For example, on dataset *Ion*, the LFG reaches a precision of 90.6%, surpassing the highest precision of 86.7% among all baselines on KNN. This result indicates that the LFG is effective in reducing misclassifications. Regarding the metric recall, the LFG increases the *Ion*'s recall by 9.8% compared to vanilla data and by 4.3% compared to best of baselines on DT task, which demonstrates that LFG brings higher improvement on identifying positive samples. On the *Ama* dataset, our method significantly improves F1. The 3-iteration version, LFG-3, increased the F1 from an initial 63.8% to 67.6%. The full LFG model further boosted this performance, reaching a final F1 of 68.0%. As the harmonic mean of precision and recall, this substantial improvement in F1 showcases the model's capability to effectively balance between reducing misclassifications and minimizing missed classifications.

Robustness Check. We validate the experiments through five-fold cross-validation. The LFG method demonstrates robust performance, achieving notable metric improvements. With LFG-3, it gains 2.72% in accuracy and approximately 5% in precision, recall, and F1. The full LFG model further boosts these gains to 4.81% in accuracy and approximately 7% in other metrics. This consistent enhancement is driven by the extended iterations, which continue to refine the feature space, and it illustrates the adaptability, effectiveness, and reliability of the LFG method.

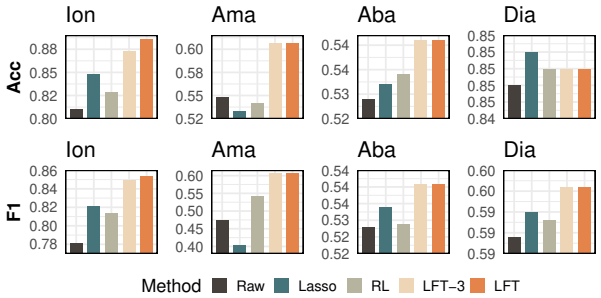

Figure 4: Comparison on KNN (Accuracy and F1).

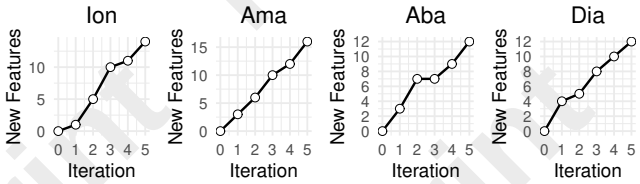

Figure 5: Increase of feature numbers.

Iterative Performance Improvements. We then compare the performance improvements of LFG-3 and LFG. On one hand, the incremental improvement from LFG-3 to LFG illustrates the continuous enhancement in all metrics, indicating that during additional iterations, the model refines and adjusts its generation strategy to explore better solutions for the feature space. On the other hand, the results of LFG-3 with only three iterations already show notable enhancement in all datasets. This shows that the model effectively and rapidly finds the zones of optimal feature space within a limited number of steps. As shown in Figure 5, we can see that the average number of features in a feature subset increases continuously in the first 5 rounds of generation. The increase in features implies that the feature generation algorithm is actively finding new, potential feature combinations in the feature space, effectively expanding the feature space.

5 Conclusion

In this paper, we present a novel feature generation method LFG utilizing LLMs to enhance automated feature engineering's interpretability, adaptability, and strategic flexibility. Our target in designing the model is to address traditional feature generation challenges, including a lack of explainability, limited applicability, and rigid strategy formulation. Our approach utilizes expert-level LLM agents to overcome these. These problems can significantly limit the broader deployment of feature engineering on machine learning tasks in diverse scenarios. Thus, we present LFG with a dynamic, adap-

Metrics	Model	RF				DT				KNN				MLP			
		Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia
Acc	Raw	0.892	0.596	0.539	0.859	0.873	0.536	0.489	0.794	0.810	0.548	0.529	0.847	0.892	0.639	0.546	0.862
	Lasso	0.892	0.627	0.537	0.860	0.880	0.554	0.491	0.796	0.848	0.533	0.532	0.849	0.905	0.671	0.552	0.863
	RL	0.911	0.607	0.559	0.859	0.892	0.548	0.508	0.796	0.829	0.542	0.534	0.848	0.911	0.616	0.558	0.864
	LFG-3	0.918	0.615	0.564	0.860	0.905	0.590	0.523	0.800	0.873	0.607	0.541	0.848	0.918	0.676	0.572	0.866
	LFG	**0.956**	**0.639**	**0.564**	**0.861**	**0.937**	**0.590**	**0.523**	**0.800**	**0.886**	**0.607**	**0.541**	**0.848**	**0.943**	**0.680**	**0.573**	**0.866**
		Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia
Prec	Raw	0.884	0.598	0.533	0.681	0.899	0.537	0.492	0.588	0.867	0.605	0.525	0.643	0.900	0.638	0.541	0.697
	Lasso	0.866	0.624	0.531	0.684	0.857	0.554	0.493	0.590	0.865	0.544	0.532	0.647	0.904	0.671	0.549	0.703
	RL	0.919	0.610	0.554	0.683	0.892	0.548	0.495	0.590	0.856	0.542	0.528	0.642	0.915	0.616	0.544	0.708
	LFG-3	0.915	0.623	0.562	0.688	0.892	0.591	0.524	0.592	0.891	0.608	0.544	0.645	0.940	0.676	0.560	0.719
	LFG	**0.962**	**0.639**	**0.562**	**0.690**	**0.929**	**0.591**	**0.524**	**0.592**	**0.906**	**0.608**	**0.544**	**0.645**	**0.953**	**0.681**	**0.562**	**0.719**
		Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia
Rec	Raw	0.880	0.597	0.545	0.570	0.837	0.537	0.493	0.597	0.768	0.548	0.533	0.577	0.892	0.639	0.544	0.595
	Lasso	0.887	0.622	0.541	0.575	0.874	0.554	0.496	0.599	0.802	0.510	0.542	0.580	0.885	0.672	0.552	0.591
	RL	0.911	0.607	0.559	0.568	0.892	0.548	0.508	0.598	0.829	0.542	0.534	0.579	0.911	0.616	0.558	**0.589**
	LFG-3	0.904	0.618	0.564	0.568	0.902	0.591	0.526	0.601	0.828	0.607	0.546	0.582	0.897	0.676	0.573	0.562
	LFG	**0.944**	**0.639**	**0.564**	**0.570**	**0.935**	**0.591**	**0.526**	**0.601**	**0.828**	**0.607**	**0.546**	**0.582**	**0.921**	**0.681**	**0.565**	0.562
		Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia	Ion	Ama	Aba	Dia
F1	Raw	0.882	0.595	0.537	0.587	0.855	0.536	0.492	0.592	0.781	0.476	0.528	0.592	0.889	0.638	0.541	**0.618**
	Lasso	0.875	0.622	0.533	**0.594**	0.865	0.554	0.493	0.594	0.821	0.403	0.534	0.595	0.893	0.671	0.550	0.614
	RL	0.909	0.602	0.556	0.584	0.890	0.548	0.498	0.594	0.814	0.542	0.529	0.594	0.909	0.615	0.544	0.611
	LFG-3	0.909	0.612	0.562	0.584	0.897	0.590	0.525	0.596	0.849	0.607	0.541	0.598	0.911	0.676	0.563	0.576
	LFG	**0.952**	**0.639**	**0.562**	0.587	**0.932**	**0.590**	**0.525**	**0.596**	**0.854**	**0.607**	**0.541**	**0.598**	**0.935**	**0.680**	**0.563**	0.576

Table 2: Overall performance on downstream tasks. The best results are highlighted in **bold**, and the runner-up results are highlighted in underline. (Higher values indicate better performance.)

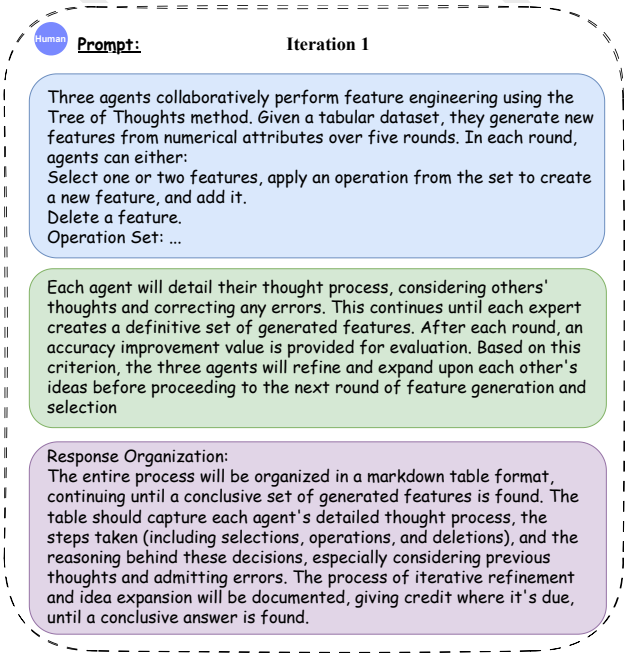

Figure 6: The exemplary prompt for LLM to generate agents and features. The *Feature Engineering* is shown in blue, the *Iterative Refinement and Evaluation* is shown in green and the *Markdown and Organization* is shown in purple. The LLM follows the instructions to start the first iteration and continues to generate features with the downstream tasks' performance data.

tive, automated feature generation approach to enhance interpretability and extend utility across various data types and tasks. Moreover, we present extensive experimental results demonstrating that our approach outperforms various existing methods. For future work, we plan to extend this automated feature engineering paradigm across various techniques and machine learning tasks and investigate transformations our approach applies to the feature space to gain deeper insights into its underlying mechanisms and impacts.

6 Limitations and Ethics Statements

While our LFG shows significant advancements and wide adaptability, there are several limitations that require further exploration, including: (1) high computational demands and limited scalability with very large or complex datasets. With LLM agents being the generator, the computational demands are comparatively higher than traditional methods; (2) the effectiveness of the generated features heavily relies on the quality of input data, which can affect the model's performance in scenarios with poorly curated or noisy datasets; (3) the current work focuses solely on the tabular feature generation process. More complex generation methods on different data are not considered. Extending the LFG to handle feature generation in non-tabular scenarios remains a challenge.

The LFG framework can enhance the effectiveness and explainability of the feature generation process. However, as the framework uses pre-trained GPT-3.5 Turbo as the generation model, it may inherit the ethical concerns associated with GPT-3.5 Turbo, such as responding to harmful queries or exhibiting biased behaviors.

Acknowledgements

Pengfei Wang is supported by the National Natural Science Foundation of China (Grant Nos. 62406306 and 92470204).

References

[Auer *et al.*, 2002] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[Besta *et al.*, 2024] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38(16), pages 17682–17690, 2024.

[Browne *et al.*, 2012] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[Chu *et al.*, 2023] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*, 2023.

[Diao *et al.*, 2023] Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*, 2023.

[Gu *et al.*, 2024] Yang Gu, Hengyu You, Jian Cao, Muran Yu, Haoran Fan, and Shiyou Qian. Large language models for constructing and optimizing machine learning workflows: A survey. *arXiv preprint arXiv:2411.10478*, 2024.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[Hao *et al.*, 2023] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

[Jordan and Mitchell, 2015] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[Katz *et al.*, 2016] 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, 2016.

[Khurana *et al.*, 2016] Udayan Khurana, Deepak Turaga, Horst Samulowitz, and Srinivasan Parthasrathy. Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 1304–1307. IEEE, 2016.

[Khurana *et al.*, 2018] 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(1), 2018.

[Liu *et al.*, 2023] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, page 100017, 2023.

[Liu, 2011] Zhi Liu. Amazon Commerce Reviews Set. https://archive.ics.uci.edu/dataset/215/amazon+commerce+reviews+set, 2011. Accessed: 2025-01-21.

[Mumuni and Mumuni, 2024] Alhassan Mumuni and Fuseini Mumuni. Automated data processing and feature engineering for deep learning and big data applications: a survey. *Journal of Information and Intelligence*, 2024.

[Nam *et al.*, 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. *Advances in Neural Information Processing Systems*, 37:92352–92380, 2024.

[Nargesian *et al.*, 2017] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B Khalil, and Deepak S Turaga. Learning feature engineering for classification. In *Ijcai*, volume 17, pages 2529–2535, 2017.

[Nash *et al.*, 1995] Warwick Nash, Tracy Sellers, Simon Talbot, Andrew Cawthorn, and Wes Ford. Abalone. https://archive.ics.uci.edu/dataset/1/abalone, 1995. Accessed: 2025-01-21.

[OpenAI, 2024] OpenAI. OpenAI API. https://openai.com/index/openai-api/, 2024. Accessed: 2025-01-21.

[Pan *et al.*, 2020] Tongyang Pan, Jinglong Chen, Jingsong Xie, Zitong Zhou, and Shuilong He. Deep feature generating network: A new method for intelligent fault detection of mechanical systems under class imbalance. *IEEE Transactions on Industrial Informatics*, 17(9):6282–6293, 2020.

[Peng *et al.*, 2023] Letian Peng, Yuwei Zhang, and Jingbo Shang. Generating efficient training data via llm-based attribute manipulation. *arXiv preprint arXiv:2307.07099*, 2023.

[Schölkopf *et al.*, 2021] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[Severyn and Moschitti, 2013] Aliaksei Severyn and Alessandro Moschitti. Automatic feature engineering for answer selection and extraction. In *Proceedings of*

the 2013 Conference on Empirical Methods in Natural Language Processing, pages 458–467, 2013.

[Shi *et al.*, 2018] Hongtao Shi, Hongping Li, Dan Zhang, Chaqiu Cheng, and Xuanxuan Cao. An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification. *Computer Networks*, 132:81–98, 2018.

[Sigillito *et al.*, 1989] V. Sigillito, S. Wing, L. Hutton, and K. Baker. Ionosphere. https://archive.ics.uci.edu/dataset/52/ionosphere, 1989. Accessed: 2025-01-21.

[Song *et al.*, 2019] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1161–1170, 2019.

[Teboul, 2022] Alex Teboul. Diabetes Health Indicators Dataset. https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset, 2022. Accessed: 2025-01-21.

[Wang *et al.*, 2022a] Dongjie Wang, Yanjie Fu, Kunpeng Liu, Xiaolin Li, and Yan Solihin. Group-wise reinforcement feature generation for optimal and explainable representation space reconstruction. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1826–1834, 2022.

[Wang *et al.*, 2022b] Limin Wang, Shuai Zhang, Musa Mammadov, Kuo Li, Xinhao Zhang, and Siyuan Wu. Semi-supervised weighting for averaged one-dependence estimators. *Applied Intelligence*, pages 1–17, 2022.

[Wang *et al.*, 2022c] LiMin Wang, XinHao Zhang, Kuo Li, and Shuai Zhang. Semi-supervised learning for k-dependence bayesian classifiers. *Applied Intelligence*, pages 1–19, 2022.

[Wang *et al.*, 2023] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*, 2023.

[Wang *et al.*, 2024] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):1–26, 2024.

[Wang *et al.*, 2025] Zaitian Wang, Jinghan Zhang, Xinhao Zhang, Kunpeng Liu, Pengfei Wang, and Yuanchun Zhou. Diversity-oriented data augmentation with large language models. *arXiv preprint arXiv:2502.11671*, 2025.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[White *et al.*, 2023] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.

[Xiang *et al.*, 2021] Ziyu Xiang, Mingzhou Fan, Guillermo Vázquez Tovar, William Trehern, Byung-Jun Yoon, Xiaofeng Qian, Raymundo Arroyave, and Xiaoning Qian. Physics-constrained automatic feature engineering for predictive modeling in materials science. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35(12), pages 10414–10421, 2021.

[Xie *et al.*, 2024] Henry J Xie, Jinghan Zhang, Xinhao Zhang, and Kunpeng Liu. Scoring with large language models: A study on measuring empathy of responses in dialogues. *arXiv preprint arXiv:2412.20264*, 2024.

[Yao *et al.*, 2024] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[Yu *et al.*, 2023] Zihan Yu, Liang He, Zhen Wu, Xinyu Dai, and Jiajun Chen. Towards better chain-of-thought prompting strategies: A survey. *arXiv preprint arXiv:2310.04959*, 2023.

[Zhang *et al.*, 2024a] Jinghan Zhang, Xiting Wang, Yiqiao Jin, Changyu Chen, Xinhao Zhang, and Kunpeng Liu. Prototypical reward network for data-efficient rlhf. *arXiv preprint arXiv:2406.06606*, 2024.

[Zhang *et al.*, 2024b] Xinhao Zhang, Zaitian Wang, Lu Jiang, Wanfu Gao, Pengfei Wang, and Kunpeng Liu. Tfwt: Tabular feature weighting with transformer. *arXiv preprint arXiv:2405.08403*, 2024.

[Zhang *et al.*, 2024c] Xinhao Zhang, Jinghan Zhang, Fengran Mo, Yuzhong Chen, and Kunpeng Liu. Retrieval-augmented feature generation for domain-specific classification, 2024.

[Zhang *et al.*, 2025a] Jinghan Zhang, Fengran Mo, Xiting Wang, and Kunpeng Liu. Blind spot navigation in llm reasoning with thought space explorer, 2025.

[Zhang *et al.*, 2025b] Jinghan Zhang, Xiting Wang, Weijieying Ren, Lu Jiang, Dongjie Wang, and Kunpeng Liu. Ratt: A thought structure for coherent and correct llm reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39(25), pages 26733–26741, 2025.

[Zhang *et al.*, 2025c] Xinhao Zhang, Jinghan Zhang, Fengran Mo, Dongjie Wang, Yanjie Fu, and Kunpeng Liu. Leka: Llm-enhanced knowledge augmentation. *arXiv preprint arXiv:2501.17802*, 2025.

[Zhong *et al.*, 2016] Guoqiang Zhong, Li-Na Wang, Xiao Ling, and Junyu Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4):265–278, 2016.