SPIO: Ensemble and Selective Strategies via LLM-Based Multi-Agent Planning in Automated Data Science

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have enabled dynamic reasoning in automated data analytics, yet recent multi-agent systems remain limited by rigid, single-path workflows that restrict strategic exploration and often lead to suboptimal outcomes. To overcome these limitations, we propose **SPIO** (**S**equential **P**lan **I**ntegration and **O**ptimization), a framework that replaces rigid workflows with adaptive, multi-path planning across four core modules: data preprocessing, feature engineering, model selection, and hyperparameter tuning. In each module, specialized agents generate diverse candidate strategies, which are cascaded and refined by an optimization agent. **SPIO** offers two operating modes: **SPIO-S** for selecting a single optimal pipeline, and **SPIO-E** for ensembling top-k pipelines to maximize robustness. Extensive evaluations on Kaggle and OpenML benchmarks show that **SPIO** consistently outperforms state-of-the-art baselines, achieving an average performance gain of 5.6%. By explicitly exploring and integrating multiple solution paths, **SPIO** delivers a more flexible, accurate, and reliable foundation for automated data science.

1 Introduction

Automated data analytics and predictive modeling have emerged as pivotal components in modern data-driven research (Erickson et al., 2020; Ebadifard et al., 2023; Hollmann et al., 2023; Guo et al., 2024), empowering practitioners to extract meaningful insights from complex, high-dimensional datasets (Drori et al., 2021; Hollmann et al., 2023; Hassan et al., 2023; Yang et al., 2024). Traditionally, constructing robust analytics pipelines has necessitated extensive manual design and iterative refinement, relying heavily on domain expertise and coding practices with established libraries and frameworks (Bergstra and Bengio, 2012; Elsken et al., 2017; Sparks et al., 2017; De Bie et al., 2021). This conventional paradigm, while effective in certain settings, often imposes rigid workflow structures that limit strategic exploration and impede the integration of diverse solution paths, ultimately constraining the scalability and robustness of predictive systems (Sparks et al., 2017; Xin et al., 2018; Maymounkov, 2018; Nikitin et al., 2022).

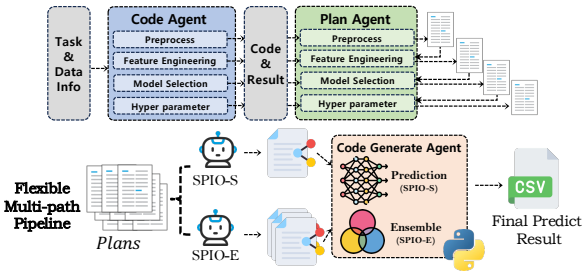


Figure 1: **SPIO framework overview.** Given dataset descriptions, **SPIO** produces baseline code and results for each pipeline module. A sequential planning agent then proposes candidate improvements per module, after which **SPIO** either selects a single best end-to-end path (**SPIO-S**) or ensembles the top-k paths (**SPIO-E**) to generate the prediction file.

Large Language Models (LLMs) have become powerful tools for automating data analytics and predictive modeling, addressing prior challenges via natural language instructions (Zhang et al., 2023; Guo et al., 2024; Xu et al., 2024a; Tayebi Arasteh et al., 2024; Gu et al., 2024). By translating high-level instructions into executable code, LLM-based systems greatly reduce the need for specialized programming skills and simplify the design of complex analytical workflows (Nijkamp et al., 2022; Joel et al., 2024; Xu et al., 2024b). Recent frameworks include Spider2v (Cao et al., 2024), which automates analysis by generating SQL/Python code and manipulating GUI components, and OpenAgents (Xie et al., 2023), a platform that integrates diverse AI agents for unified data analysis and web search (Xie et al., 2023).

More recently, multi-agent systems such as

Agent K v1.0 (Grosnit et al., 2024), AutoKaggle (Li et al., 2024), and Data Interpreter (Hong et al., 2025) automate end-to-end data analytics. Agent K v1.0 coordinates agents for cleaning, feature engineering, modeling, and hyperparameter tuning; AutoKaggle follows a six-stage pipeline from exploration to planning; and Data Interpreter constructs task graphs that decompose subtasks and dependencies to guide actions. More general-purpose agent frameworks, including OpenHands (Wang et al., 2024) and AIDE (Jiang et al., 2025), further incorporate execution-driven loops for autonomous code generation and refinement. However, these methods still (1) rely on single-path execution that limits exploration, (2) follow rigid, predefined workflows that struggle with complex tasks, and (3) lack mechanisms to aggregate multi-dimensional feedback, reducing their ability to exploit nuanced signals for optimal predictions.

To address these limitations, we propose **SPIO** (**S**equential **P**lan **I**ntegration and **O**ptimization), a framework for automated data analytics and predictive modeling. Unlike rigid single-path or fixed multi-agent workflows, **SPIO** organizes the pipeline into four modules: data preprocessing, feature engineering, model selection, and hyperparameter tuning, and performs sequential planning within each module. At each stage, **SPIO** records intermediate outputs such as transformed data summaries or validation scores, which interact with results from previous stages. Based on these accumulated plan records, **SPIO** applies LLM-driven selection and ensembling to optimize end-to-end performance. Code-generation agents first produce executable solutions from a data description D and a task description T. A sequential planning agent then revises these solutions using data statistics for preprocessing and feature engineering, and validation score for model selection and hyperparameter tuning. Finally, **SPIO** supports two variants: **SPIO-S**, which selects a single best pipeline for execution, and **SPIO-E**, which selects the top k pipelines and ensembles their predictions.

Extensive experiments on regression and classification benchmark datasets, including Kaggle and OpenML demonstrate **SPIO**'s superiority over state-of-the-art methods. **SPIO**'s adaptive multi-path reasoning integrates diverse insights, overcoming the limitations of fixed workflows. Consequently, it improves predictive accuracy, adapts to varied data through iterative feedback, and enhances execution reliability. These strengths make **SPIO** a valuable solution for business intelligence, scientific research, and automated reporting, where accuracy and flexibility are essential.

2 Related Work

2.1 Machine Learning and AutoML

Machine learning (ML) develops algorithms that learn from data to make predictions or decisions (Sutton et al., 1998; Bishop and Nasrabadi, 2006; Das et al., 2015; Helm et al., 2020). Despite major advances in ML algorithms (Kirkos et al., 2008; El Naqa and Murphy, 2015; Xie et al., 2016; Sarker, 2021), building effective systems still requires substantial manual effort: practitioners must choose models, craft features, and tune hyperparameters, which is time-consuming and expert-dependent. Automated Machine Learning (AutoML) aims to reduce this burden by automating key steps of pipeline construction (Shen et al., 2018; Wang et al., 2021; Feurer et al., 2022). Early systems such as Auto-WEKA (Thornton et al., 2013) and TPOT (Olson and Moore, 2016) used Bayesian optimization and genetic programming to search predefined model spaces, while later frameworks (e.g., DSM (Kanter and Veeramachaneni, 2015), OneBM (Lam et al., 2017), AutoLearn (Kaul et al., 2017), H2O AutoML (LeDell et al., 2020)) incorporated meta-learning and ensembles. However, many AutoML solutions remain constrained by static search spaces, limiting ptation to diverse data and evolving tasks. Thus, scalability and flexibility remain open challenges.

2.2 Data Science Agents

Large Language Models (LLMs) have accelerated automated data science agents for task understanding, data processing, and predictive modeling. Tools such as LiDA (Dibia, 2023) and GPT4-Analyst (Cheng et al., 2023) automate exploratory data analysis, while recent work further leverages code-interpreting capabilities to enable more flexible and complex analytics (Seo et al., 2025). Multi-agent systems extend this line toward end-to-end pipelines: Agent K (Grosnit et al., 2024) decomposes for cleaning, feature engineering, modeling and tuning. AutoKaggle (Li et al., 2024) follows a multi-phase workflow, and Data Interpreter (Hong et al., 2025) structures subtasks as dependency graphs; DS Agent Survey (Wang et al., 2025) provides a taxonomy linking agent design to workflow stages, and Blackboard DS Discov-

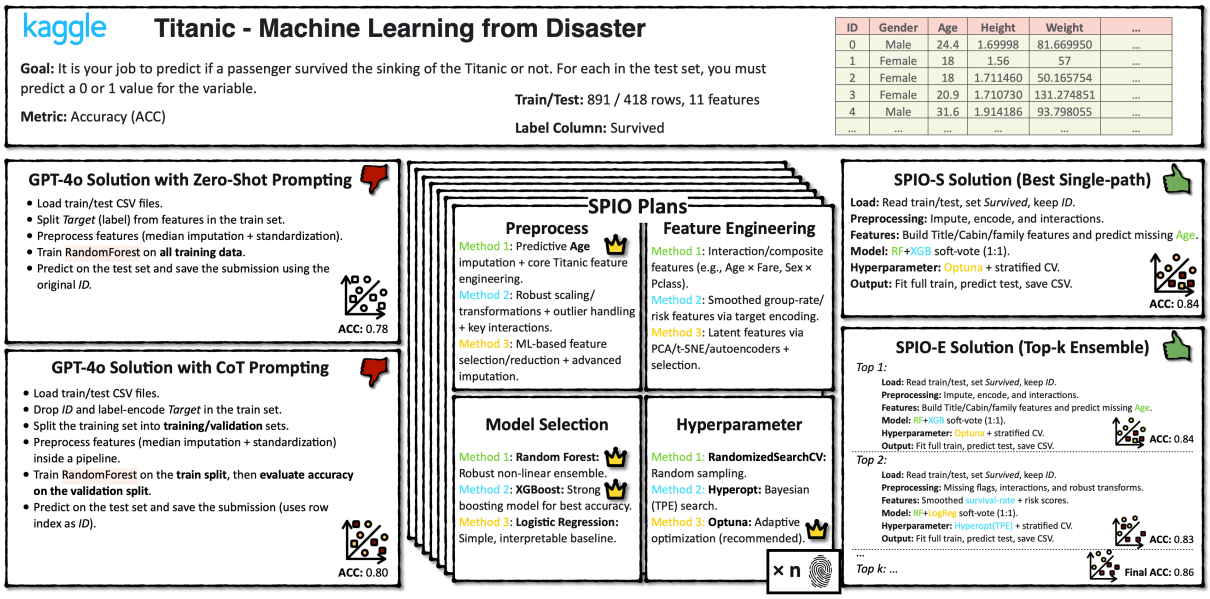


Figure 2: **Sequential plan integration and optimization in SPIO (Titanic example). SPIO** enumerates candidate plans for preprocessing, feature engineering, model selection, and hyperparameter tuning (e.g., predictive Age imputation, interaction/target-encoded features, RF/XGBoost/LogReg choices, and Optuna/Hyperopt search), then composes them into complete pipelines. Compared with GPT-4o baselines (Zero-Shot ACC 0.78; CoT ACC 0.80), **SPIO** selects the best single path (**SPIO-S**, ACC 0.84) or ensembles the top-k pipelines (**SPIO-E**, final ACC 0.86).

ery (Salemi et al., 2025) coordinates specialists via a shared blackboard for scalable data-lake discovery; more general-purpose agent frameworks such as OpenHands (Wang et al., 2024) and AIDE (Jiang et al., 2025) adopt execution-driven iterative loops, refining LLM-generated code through repeated planning, execution, and feedback. Recent benchmarks (Tang et al., 2023; Chan et al., 2024; Huang et al., 2023; Nathani et al., 2025) have begun to standardize the evaluation of LLM-based data science agents on end-to-end AutoML tasks. However, these systems typically follow a single reasoning path with rigid workflows, which can limit adaptability to task-specific nuances. To address these limitations, we propose the **SPIO** framework.

3 Methodology

3.1 Preliminaries

Our proposed **SPIO** framework, as shown in Figure 2, integrates sequential planning and optimization into an automated analytics pipeline. The framework accepts two primary inputs: a *Data Description* D (detailing data shape, column types, length, and sample records) and a *Task Description* T (specifying the prediction type, target column, and relevant background). The framework is organized into four key components, each playing a distinct role in the overall process:

1. **Fundamental Code Generation Agents** generate baseline solutions for each module (preprocessing, feature engineering, model selection, hyperparameter tuning) using D and T.

2. **Sequential Planning Agent** reviews the baseline outputs and based on prior plans, produces up to n candidate strategies for further improvement.

3. **SPIO-S: Single-Path Selection** leverages the candidate plans to select one optimal solution path for final execution.

4. **SPIO-E: Ensemble of Sequential Paths** aggregates the top k candidates via an ensemble approach to enhance predictive robustness.

3.2 Fundamental Code Generation Agents

In each module, the fundamental code generation agent establishes a stable baseline by generating executable code and producing a corresponding description of the processed output. We specifically denote each module's output as a tuple (C_i, D_i) or (C_i, V_i), where C_i is the generated code, D_i is the description of the resulting data, and V_i is the validation score (used in modeling and tuning stages). For instance, given D and T, the *data preprocessing agent* produces:

$$(C_{\text{pre}}, D_{\text{pre}}) = \mathcal{G}(D, T), \tag{1}$$

where D_{pre} encapsulates key details for data preprocessing such as data shape, column types, and sample records. The *feature engineering agent* then takes $(C_{\text{pre}}, D_{\text{pre}})$ along with T to generate:

$$(C_{\text{feat}}, D_{\text{feat}}) = \mathcal{G}(C_{\text{pre}}, D_{\text{pre}}, T). \quad (2)$$

For model selection and hyperparameter tuning, the agents output both the generated code and a validation score. Specifically, the *model selection agent* processes $(C_{\text{feat}}, D_{\text{feat}}, T)$ to yield:

$$(C_{\text{model}}, V_{\text{model}}) = \mathcal{G}(C_{\text{feat}}, D_{\text{feat}}, T), \quad (3)$$

and the *hyperparameter tuning agent* refines the model further by producing:

$$(C_{\text{hp}}, V_{\text{hp}}) = \mathcal{G}(C_{\text{model}}, D_{\text{feat}}, T). \quad (4)$$

These outputs form the foundation for subsequent reflective improvement.

3.3 Sequential Planning Agent

After each fundamental code generation step, a sequential planning agent is applied to refine the outputs by proposing alternative strategies. In the data preprocessing and feature engineering modules, the planning agent uses the current code C_i and its corresponding output description D_i (for $i \in \{\text{pre}, \text{feat}\}$), along with D, T, and all previously generated candidate plans $P_{<i}$, to propose up to n alternative strategies:

$$\{P_i^1, P_i^2, \ldots, P_i^n\} = \mathcal{G}(C_i, D_i, D, T, P_{<i}). \quad (5)$$

For the model selection and hyperparameter tuning modules, the planning agent incorporates the validation score V_i (with $i \in \{\text{model}, \text{hp}\}$) as a key performance indicator, generating:

$$\{P_i^1, P_i^2, \ldots, P_i^n\} = \mathcal{G}(C_i, V_i, D, T, P_{<i}). \quad (6)$$

This reflective planning mechanism systematically explores alternative solution paths, thereby supporting subsequent optimization.

3.4 SPIO-S: Single-Path Selection

The **SPIO-S** variant employs an LLM-driven optimization agent to directly identify and select the single optimal solution path from among candidate plans generated across all modules. Formally, let

$$\mathcal{P} = \bigcup_{i \in \{\text{pre, feat, model, hp}\}} \{P_i^1, P_i^2, \ldots, P_i^n\}$$

denote the complete set of candidate strategies. Given the data description D, task description T, and intermediate module outputs, the LLM autonomously evaluates and selects the most effective candidate plan:

$$P^* = \text{SelectBest}_{\text{LLM}}(\mathcal{P}, D, T). \quad (7)$$

The final executable code is subsequently generated by applying the generative model to the chosen plan P^*:

$$C_{\text{final}} = \mathcal{G}(P^*, D, T). \quad (8)$$

Executing C_{final} yields the final predictive outcome.

3.5 SPIO-E: Ensemble of Sequential Paths

The **SPIO-E** variant enhances predictive robustness by utilizing an LLM-driven optimization agent to select the top k candidate plans for ensemble integration. Specifically, given the candidate plan set $\mathcal{P}$, along with descriptions D, T, and intermediate module outputs, the LLM autonomously selects and ranks the most promising k candidates:

$$\{P_1^*, P_2^*, \ldots, P_k^*\} = \text{SelectTopK}_{\text{LLM}}(\mathcal{P}, D, T). \quad (9)$$

For each selected candidate plan P_i^*, executable code is independently generated:

$$C_{\text{final}}^i = \mathcal{G}(P_i^*, D, T). \quad (10)$$

For classification tasks, the ensemble prediction is computed through soft voting:

$$\hat{y} = \arg\max_{c \in \mathcal{C}} \left(\frac{1}{k} \sum_{i=1}^{k} p_i(c) \right), \quad (11)$$

while for regression tasks, predictions are averaged across the k models:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i. \quad (12)$$

4 Experiments

4.1 Setup

Models Used. We utilize `GPT-4o` (Achiam et al., 2023), `Claude 3.5 Haiku` (Anthropic, 2024)[1],and the open-weight `LLaMA3-8B model` (Dubey et al., 2024)[2] for our generation tasks. Following prior LLM-based data-science agents and code-generation frameworks to balance exploration and

[1] https://www.anthropic.com/claude/haiku
[2] https://ai.meta.com/llama/

execution reliability (Cao et al., 2024; Guo et al., 2024; Wu et al., 2024), we adopt a moderate sampling setup for **SPIO** to encourage diverse while maintaining stable, executable code: temperature= 0.5, top_p= 1.0, and max_tokens= 4096.

Experimental Datasets. We evaluate our approach on two widely used benchmarks: OpenML (Vanschoren et al., 2014) and Kaggle.[3] From OpenML, we use 4 popular datasets (3 regression, 1 binary classification), partitioned into 70% train, 10% validation, and 20% test due to the absence of predefined splits. From Kaggle, we use 8 datasets (7 classification, 1 regression). Since these provide leaderboard test sets, we split the given training data into 70% training and 30% validation data. The validation sets are utilized for planning and optimizing, while the leaderboard test sets are reserved for the main experiments.[4]

Baseline Methods and Implementation Details. We compare our approach with two single-inference and five recently proposed representative multi-agent baselines, specifically:

1. **Zero Shot Inference:** The LLM generates code tailored to the task, without intermediate reasoning.

2. **CoT Inference** (Wei et al., 2022): The LLM generates an explicit step-by-step reasoning trace to derive the pipeline, then outputs the final executable code.

3. **Agent K v1.0** (Grosnit et al., 2024): uses a two-stage workflow that first prepares the data and submission artifacts, then trains models with hyperparameter tuning and ensembling.

4. **AutoKaggle** (Li et al., 2024): follows a six-phase workflow: task/background understanding, exploratory analysis, data cleaning, feature engineering, modeling, and final submission generation, coordinated by specialized agents.

5. **OpenHands** (Wang et al., 2024): a general-purpose autonomous agent framework that decomposes user instructions into tool-augmented action sequences. It iteratively plans, executes, and revises code using execution feedback, relying on a unified agent loop rather than task-specific phases.

6. **Data Interpreter** (Hong et al., 2025): constructs high-level task graphs and iteratively refines them into executable action graphs, translating abstract intent into step-by-step code actions guided by intermediate results.

7. **AIDE** (Jiang et al., 2025): performs ML engineering as code optimization, using execution scores to search a solution tree. It iteratively selects promising nodes, applies small fixes or improvements, and evaluates new variants.

Compared to the baselines, our framework **SPIO** utilizes LLM-driven planning ability across four modules (preprocessing, feature engineering, modeling, and hyperparameter tuning), generating up to two candidate plans per module, where **SPIO-S** selects the single best plan and **SPIO-E** ensembles the top two plans.

Evaluation Metrics. For each task, we use standard, task-appropriate metrics. *Root Mean Squared Error* (RMSE) is the primary metric for regression tasks, and for computational convenience, we report RMSE even when the official metric is MSE. We additionally evaluate performance using *ROC* metrics when applicable, and use *Accuracy* (ACC) as the primary metric for classification tasks.

4.2 Experimental Analysis

4.2.1 Main Experiment Results

We compare our proposed framework **SPIO** with several baseline methods. As shown in Table 1, while **ZeroShot** provides quick initial solutions, it typically yields inconsistent results due to its lack of structured reasoning. **CoT** prompting slightly improves interpretability by adopting step-by-step reasoning but still remains limited by its single-path exploration. Multi-agent frameworks such as **Agent K v1.0**, **Data Interpreter**, and **AutoKaggle** further enhance performance by decomposing the workflow into specialized stages and coordinating multiple agents; however, their stage schemas are largely predefined and they typically refine within a fixed pipeline template, which can limit systematic exploration of diverse module-level alternatives and their combinations. **AIDE** and **OpenHands** further enhance performance through execution-driven improvement loops; however, they rely heavily on interaction with runtime feedback rather than explicit plan reasoning, making results more sensitive to execution budgets and offering weaker plan-level traceability and control.

[3] https://www.kaggle.com/

[4] Detailed dataset descriptions are provided in Table 6.

Table 1: **Performance comparison on 8 Kaggle datasets and 4 OpenML datasets for three LLM backends (GPT-4o, Claude 3.5 Haiku, and LLaMA3-8B). Bold** indicates the best; underline indicates the second-best. Human expert denotes the best public Kaggle leaderboard score achieved by a human expert submission (i.e., the top-ranked result at the time of reporting). An asterisk (*) denotes a statistically significant improvement for **SPIO** variants (paired t-test with Holm-Bonferroni correction, $p < 0.05$). Results for OpenHands and AIDE are unavailable under `LLaMA3-8B` due to context-length constraints and the lack of native function-calling support.

Methods	Titanic (ACC↑)	Spaceship (ACC↑)	Monsters (ACC↑)	Academic (ACC↑)	Obesity (ACC↑)	Kc1 (ACC↑)	Bank Churn (ROC↑)	Plate Defect (ROC↑)	House Price (RMSE↓)	Boston (RMSE↓)	Diamond (RMSE↓)	Sat11 (RMSE↓)
GPT-4o												
ZeroShot	0.7464	0.7706	0.7051	0.8269	0.8864	0.8499	0.8192	0.8640	0.1509	4.6387	540.3852	1447.1177
CoT (2022)	0.7440	0.7718	0.7108	0.8280	0.8859	0.8373	0.8719	0.8654	0.1447	4.6387	540.3556	1447.1177
Agent K V1.0 (2024)	0.7608	0.7810	0.6673	0.7879	0.8356	0.8490	0.8796	0.8209	0.1437	3.6797	415.8068	1338.4025
Auto Kaggle (2024)	0.7781	0.7753	0.7253	0.8275	0.8864	0.8422	0.8742	0.8351	0.1441	4.2510	417.9387	1365.7454
OpenHands (2024)	0.7528	0.7860	0.7189	0.8164	0.8815	0.8503	0.8830	0.8632	0.1379	3.4234	569.5314	1355.3216
Data Interpreter (2025)	0.7679	0.7870	0.7289	0.8297	0.8986	0.8482	0.8836	0.8828	0.1415	3.3937	420.9826	1376.4327
AIDE (2025)	0.7775	0.7949	0.7088	0.8340	0.8973	0.8567	0.8877	**0.8856**	0.1310	3.8445	444.0037	1319.8435
SPIO-S (Ours)	0.7847*	0.8010*	0.7316*	0.8341*	0.9071	0.8590	0.8877*	0.8836*	0.1310*	3.0312	404.6253	1290.9073
SPIO-E (Ours)	**0.7871***	**0.8034***	**0.7410***	**0.8359***	**0.9072**	**0.8687**	**0.8885***	0.8843*	**0.1298***	**2.9192**	**398.8893**	**1268.7817**
Claude 3.5 Haiku												
ZeroShot	0.7488	0.7904	0.7089	0.8281	0.8910	0.8594	0.7251	0.8657	0.1473	4.6387	543.1463	1446.9613
CoT (2022)	0.7488	0.7917	0.6994	0.8282	0.8904	0.8547	0.7454	0.8685	0.1467	4.6421	543.1463	1440.0644
Agent K V1.0 (2024)	0.7512	0.7928	0.7127	0.8288	0.8988	0.8639	0.8863	0.8813	0.1407	2.9710	546.9876	1445.1732
Auto Kaggle (2024)	0.7688	0.7884	0.7207	0.8182	0.8933	0.7427	0.8459	0.8243	0.1480	3.2183	789.4067	1288.3061
OpenHands (2024)	0.7727	0.7830	0.7051	0.8267	0.8894	0.8456	0.8656	0.8577	0.1445	3.1952	517.1757	1369.5845
Data Interpreter (2025)	0.7464	0.7940	0.6994	0.8282	0.8951	0.8578	0.8744	0.8642	0.1384	**2.8378**	565.1463	1350.7718
AIDE (2025)	0.7625	0.7893	**0.7316**	0.8335	0.8948	0.8706	**0.8887**	0.8856	0.1392	2.8556	560.4698	1273.8269
SPIO-S (Ours)	**0.7780***	0.7996*	0.7278*	0.8336*	0.9058*	0.8626	0.8812*	**0.8867***	0.1334*	2.8418	514.3608	1279.6327
SPIO-E (Ours)	0.7775*	**0.8027***	0.7297*	**0.8340***	**0.9066***	**0.8723**	0.8863*	0.8830*	**0.1332***	2.8409	**511.2089**	**1270.2695**
LLaMA3-8B												
ZeroShot	0.7410	0.7704	0.6880	0.8148	0.8793	0.8294	0.7783	0.8554	0.1521	4.6387	547.8507	1450.8444
CoT (2022)	0.7434	0.7819	0.7002	0.8181	0.8789	0.8279	0.7839	0.8570	0.1497	4.6387	546.4380	1425.4643
Agent K V1.0 (2024)	0.7512	0.7803	0.7013	0.8052	0.8855	0.8310	0.8671	0.8488	0.1458	4.0728	544.0589	1408.9078
Auto Kaggle (2024)	0.7415	0.7787	0.7018	0.8150	0.8775	0.8293	0.8505	0.8501	0.1469	4.1962	542.8794	1372.6835
Data Interpreter (2025)	0.7536	0.7830	0.7022	0.8187	0.8855	0.8362	0.8695	0.8613	0.1452	4.3823	538.0820	1378.5213
SPIO-S (Ours)	**0.7583***	0.7896*	0.7101	0.8208*	0.8935*	0.8424	0.8733*	0.8613*	0.1398*	3.6467	533.1050	1359.8238
SPIO-E (Ours)	0.7560*	**0.7907***	**0.7115***	**0.8238***	**0.8971***	**0.8474**	**0.8785***	**0.8739**	**0.1388**	**3.4386**	**524.1970**	**1353.8603**
Human Expert	-	0.8218	0.8072	0.8404	0.9116	-	0.9059	0.8898	-	-	-	-

In contrast, **SPIO** overcomes the limitations of single-path and rigid multi-agent workflows by integrating sequential optimization with explicit multi-path exploration across data preprocessing, feature engineering, model selection, and hyperparameter tuning. By generating and evaluating multiple candidate plans at each stage, **SPIO** improves adaptability and reduces the risk of suboptimal local decisions. Crucially, SPIO conditions later-stage decisions on intermediate evidence, rather than optimizing each module in isolation. This staged plan trace also makes the final pipeline more transparent, since SPIO can attribute gains to specific module-level choices.[5] In particular, **SPIO-S** employs an LLM-driven optimization agent to select the most effective end-to-end pipeline, while **SPIO-E** further enhances robustness by ensembling top-ranked plans to exploit complementary modeling strengths. As a result, **SPIO** consistently outperforms strong baselines across diverse datasets, achieving an average performance improvement of 5.6% across 12 benchmark datasets (Table 1), which demonstrates its effectiveness in complex, real-world scenarios.

[5]Experiments on datasets released after the language model's pretraining are provided in the Appendix A.

4.3 Ablation Studies

To further validate the robustness and effectiveness of **SPIO**, we conducted comprehensive ablation studies focusing on three key aspects: (1) Performance Comparison of Each Plan, (2) Ensemble Validation, (3) and Module Impact Analysis.

4.3.1 Performance Comparison of Each Plans

We evaluate the predictive performance of candidate plans ranked by the LLM to verify the efficacy of its selection strategy. We specifically compare the relative ranks and weighted average scores of the top-ranked plans generated at each stage.

As shown in Table 2, experimental results demonstrated that the first-ranked plan consistently achieved superior performance relative to other lower-ranked alternatives across multiple datasets. This finding validates that the LLM-driven ranking method accurately captures the underlying quality of generated plans, confirming the reliability and rationale behind its decisions. Leveraging the

Table 2: **Results of Ablation Studies: (1) *Performance Comparison*, (2) *Ensemble Validation* and (3) *Module Impact* on SPIO-S and SPIO-E Variants Across GPT-4o, Claude 3.5 Haiku, and LLaMA3-8B. Bold** text indicates the best performance. Underlined text indicates the second-best performance.

LLM	GPT-4o		Claude 3.5 Haiku		LLaMA3-8B	
	ACC/ROC ↑	RMSE ↓	ACC/ROC ↑	RMSE ↓	ACC/ROC ↑	RMSE ↓
Performance Comparison						
SPIO-S Top1	**0.8361**	**424.6737**	**0.8344**	**449.2421**	**0.8187**	**474.1788**
SPIO-S Top2	0.8308	448.2788	0.8339	469.6405	0.8025	480.1354
SPIO-S Top3	0.8248	501.7758	0.8203	504.4984	0.8052	496.9982
SPIO-S Top4	0.8193	454.9485	0.8176	596.6416	0.7931	498.5383
Ensemble Validation						
SPIO-S	0.8361	424.6737	0.8344	449.2421	0.8187	474.1788
SPIO-E Ensemble2	**0.8395**	**417.6800**	0.8365	**446.1131**	**0.8224**	**470.4087**
SPIO-E Ensemble3	0.8375	427.0149	**0.8476**	460.0835	0.8140	486.7630
SPIO-E Ensemble4	0.8341	429.5985	0.8323	524.2414	0.8014	486.3129
Module Impact Analysis						
(w/o) Preprocess	0.8310	434.4996	0.8276	472.4470	0.8331	497.3395
(w/o) Feature Eng.	0.8274	458.6899	0.8152	483.1569	0.8258	494.3421
(w/o) Model Select	0.8271	458.4007	0.8272	484.9608	0.8284	503.2921
(w/o) Hyper Param	0.8317	479.2854	0.8230	684.8432	0.8310	526.5389

LLM's top-ranked plan selection ensures that **SPIO** effectively prioritizes the most robust strategies, improving overall predictive performance.

4.3.2 Ensemble Validation

We further explore the optimal number of ensemble (k) by analyzing predictive performance across ensembles comprising two to four candidate plans. Table 2 summarizes the results across different ensemble sizes of each model.

The results clearly indicate that ensembles of two candidate plans consistently achieved the highest predictive accuracy, providing an optimal balance of diversity and precision. Ensembles with more than two plans introduced redundancy without significant performance gains, whereas a single-plan approach lacked the robustness afforded by ensemble diversity. We demonstrate that an ensemble consisting of the Top 1 and Top 2 plans yields the best performance.

4.3.3 Module Impact Analysis

We assess the contribution of each module within **SPIO** by removing, in turn, the data preprocessing, feature engineering, modeling, and hyperparameter tuning components and comparing performance.

Table 2 results in a distinct contrast in how different components support **SPIO**'s performance. When feature engineering or hyperparameter tuning is removed, the system's gains largely disappear, resulting in a pronounced reduction in overall predictive accuracy. Meanwhile, eliminating data preprocessing or the modeling module reduces performance to a lesser extent, though the drop remains clearly observable. This contrast suggests that feature engineering and hyperparameter tuning are the primary drivers behind **SPIO**'s robust, high-quality results across varied datasets.

5 Qualitative Study

To complement quantitative benchmarks, we conducted an expert qualitative study to assess the practical acceptability of **SPIO** pipelines. We considered 10 distinct LLM–datasets (e.g., GPT-4o-Titanic) pairs and recruited 10 AI practitioners and graduate-level participants, yielding 100 dataset–participant evaluation pairs. For each dataset, we evaluated six items ranging from the preprocessing outcomes to the optimal methods selected by **SPIO-S** and **SPIO-E**, and scored each item using four predefined evaluation criteria. In addition, we included a single forced-choice question to collect expert votes on the most highly rated item among the Top-1 to Top-3 candidates.

Evaluation Dimensions All Likert items are grouped into four subjective dimensions:

- **Plausibility:** Is the **SPIO** pipeline reasonable and safe?
- **Interpretability:** Are **SPIO**'s explanations

Table 3: **Expert subjective ratings (mean±std) for the final optimal pipelines.** * indicates **SPIO-E** significantly outperforms **SPIO-S** (paired t-test with Holm-Bonferroni correction; $p < 0.05$).

Dimension	SPIO-S	SPIO-E
Plausibility	4.38 ± 0.56	$4.54 \pm 0.29^*$
Interpretability	4.21 ± 0.45	$4.37 \pm 0.30^*$
Diversity & Coverage	4.10 ± 0.47	$4.35 \pm 0.30^*$
Usability & Trust	4.28 ± 0.46	$4.55 \pm 0.26^*$

and rationales clearer than typical AutoML tools?

- **Diversity and Coverage:** Does the multi-path ensemble explore distinct strategies and cover the space well?
- **Usability and Trust:** Would the participant trust and use **SPIO** in practice or research?

Result of Metric-Wise Plan Rating : Participants rated the metric-wise plans of final **SPIO-S** and **SPIO-E** pipelines on four dimensions using five-point Likert items. As shown in Table 3, All stage-wise components average above 4.0, and final pipelines score higher: **SPIO-S** = 4.24±0.18 and **SPIO-E** = 4.45±0.15.

Additionally, Table 3 figures out that **SPIO-E** is consistently higher across all dimensions, with statistically significant gains (paired t-test with Holm-Bonferroni correction; $p < 0.05$). Finally, to test alignment between the LLM's internal ranking and expert discrete choices, participants selected the best plan among the anonymized top-3 **SPIO-E** candidates per dataset: Plan 1 was chosen 72/100 times (72%), versus 22 and 6 for Plan 2 and plan 3, indicating strong agreement with the top-1 recommendation generated from LLM.[6]

6 Discussion

Multi-Path Planning improved performance. Our experiments and ablation studies show that **SPIO**'s sequential multi-path planning substantially improves automated analytics and predictive modeling, enhancing not only accuracy but also interpretability and robustness. Compared to conventional single-path workflows, **SPIO** explores multiple candidate strategies at each stage. This systematic yet flexible design produces robust, well-tuned models that better reflect real-world data complexities and user-specific analytical goals.

[6]Detailed qualitative evaluation results are provided in Appendix C.

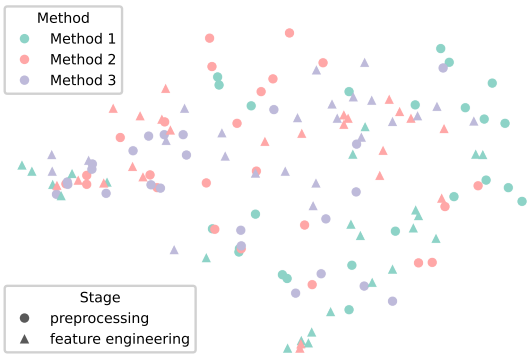


Figure 3: **Distribution of Preprocessing and Feature Engineering Method.** PCA projection of embedded preprocessing and Feature engineering methods.

Diverse Planning Across Modules. To illustrate that **SPIO** explores diverse strategies rather than following a fixed workflow, we embed the natural-language plans generated in each module using *text-embedding-ada-002* from OpenAI and project them to 2D with PCA for visualization. Figure 3 shows that plans in *Preprocessing* and *Feature Engineering* spread broadly in the embedding space, suggesting **SPIO** adapts its decision to dataset characteristics instead of collapsing to a single preferred pattern.[7]

7 Conclusion

SPIO advances automated data science by integrating sequential planning with multi-path exploration, improving predictive accuracy by an average of 5.6% over state-of-the-art baselines. **SPIO** integrates sequential planning, multi-path exploration, and ensemble optimization across four modules, using LLM-driven decisions to generate candidate strategies that capture complex data and improve robustness. Two variants, **SPIO-S** and **SPIO-E**, offer flexible use. Our ablation studies validate these gains: top-ranked plans outperform lower-ranked alternatives, and a Top-$k = 2$ ensemble yields the most reliable improvement, while removing key modules degrades performance. Our discussion highlights that **SPIO** explores diverse module-level strategies rather than collapsing to a fixed workflow, supporting robustness and interpretability in practice. Future work will extend **SPIO** to dynamic, interactive data environments to enhance utility in large-scale, real-world deployments.

[7]Additional module visualizations (model selection and hyperparameter tuning) are provided in Appendix D.

Limitation

Although our baseline comparisons show that **SPIO** generally achieves stronger predictive performance than single-pass prompting and recent multi-agent frameworks, it does not guarantee that the selected or ensembled pipeline is always globally optimal for every dataset. Since **SPIO** relies on LLM-generated plans and LLM-driven ranking to compose end-to-end solutions across preprocessing, feature engineering, model selection, and hyperparameter tuning, the final outcome can be sensitive to the LLM backend, prompt context, and generation stochasticity. In future work, we plan to develop more verification- and feedback-driven plan selection, improve efficiency through early pruning and cost-aware search, and expand the supported toolchains to enhance robustness and generalization across broader real-world settings.

Ethics Statement

SPIO can enhance automated data science through LLM-based planning and code generation, but also poses ethical and safety risks. Generated pipelines may encode bias or lead to unfair outcomes, and errors such as target leakage or poor validation can yield misleading results. **SPIO** is a research framework, not a fully audited system, so practitioners must implement governance and guardrails, including constraint-aware planning and automated checks, for responsible use.

Use of AI Assistants

We used general-purpose AI assistants only for feedback on writing quality and clarity, and not for experiment design, data analysis, or result generation. All scientific content was written and revised by the authors.

Reproducibility Statement

We present comprehensive information on architectures, datasets, and evaluation protocols both in the main text and the Appendix. Additionally, our code and scripts required to replicate all experiments are available at https://anonymous.4open.science/r/SPIO-E89C.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sonnet Anthropic. 2024. Model card addendum: Claude 3.5 haiku and upgraded claude 3.5 sonnet. *URL https://api. semanticscholar. org/CorpusID*, 273639283.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The journal of machine learning research*, 13(1):281–305.

Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Wenjing Hu, Yuchen Mao, and 1 others. 2024. Spider2-v: How far are multimodal agents from automating data science and engineering workflows? *Advances in Neural Information Processing Systems*, 37:107703–107744.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, and 1 others. 2024. Mle-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*.

Liying Cheng, Xingxuan Li, and Lidong Bing. 2023. Is gpt-4 a good data analyst? *arXiv preprint arXiv:2305.15038*.

Sumit Das, Aritra Dey, Akash Pal, and Nabamita Roy. 2015. Applications of artificial intelligence in machine learning: review and prospect. *International Journal of Computer Applications*, 115(9).

T De Bie, L De Raedt, J Hernández-Orallo, HH Hoos, P Smyth, and CKI Williams. 2021. Automating data science: Prospects and challenges. 65 (3), 76–87.

Victor Dibia. 2023. Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. *arXiv preprint arXiv:2303.02927*.

Iddo Drori, Yamuna Krishnamurthy, Remi Rampin, Raoni de Paula Lourenco, Jorge Piazentin Ono, Kyunghyun Cho, Claudio Silva, and Juliana Freire. 2021. Alphad3m: Machine learning pipeline synthesis. *arXiv preprint arXiv:2111.02508*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Nassi Ebadifard, Ajitesh Parihar, Youry Khmelevsky, Gaétan Hains, Albert Wong, and Frank Zhang. 2023. Data extraction, transformation, and loading process automation for algorithmic trading machine learning modelling and performance optimization. *arXiv preprint arXiv:2312.12774*.

Issam El Naqa and Martin J Murphy. 2015. What is machine learning? In *Machine learning in radiation oncology: theory and applications*, pages 3–11. Springer.

Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. 2017. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2022. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research*, 23(261):1–61.

Antoine Grosnit, Alexandre Maraval, James Doran, Giuseppe Paolo, Albert Thomas, Refinath Shahul Hameed Nabeezath Beevi, Jonas Gonzalez, Khyati Khandelwal, Ignacio Iacobacci, Abdelhakim Benechehab, and 1 others. 2024. Large language models orchestrating structured reasoning achieve kaggle grandmaster level. *arXiv preprint arXiv:2411.03562*.

Yang Gu, Hengyu You, Jian Cao, Muran Yu, Haoran Fan, and Shiyou Qian. 2024. Large language models for constructing and optimizing machine learning workflows: A survey. *ACM Transactions on Software Engineering and Methodology*.

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

Md Mahadi Hassan, Alex Knipper, and Shubhra Kanti Karmaker Santu. 2023. Chatgpt as your personal data scientist. *arXiv preprint arXiv:2305.13657*.

J Matthew Helm, Andrew M Swiergosz, Heather S Haeberle, Jaret M Karnuta, Jonathan L Schaffer, Viktor E Krebs, Andrew I Spitzer, and Prem N Ramkumar. 2020. Machine learning and artificial intelligence: definitions, applications, and future directions. *Current reviews in musculoskeletal medicine*, 13(1):69–76.

Noah Hollmann, Samuel Müller, and Frank Hutter. 2023. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *Advances in Neural Information Processing Systems*, 36:44753–44775.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, and 1 others. 2025. Data interpreter: An llm agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2023. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*.

Sathvik Joel, Jie Wu, and Fatemeh Fard. 2024. A survey on llm-based code generation for low-resource and domain-specific programming languages. *ACM Transactions on Software Engineering and Methodology*.

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

Ambika Kaul, Saket Maheshwary, and Vikram Pudi. 2017. Autolearn—automated feature generation and selection. In *2017 IEEE International Conference on data mining (ICDM)*, pages 217–226. IEEE.

Efstathios Kirkos, Charalambos Spathis, and Yannis Manolopoulos. 2008. Support vector machines, decision trees and neural networks for auditor selection. *Journal of computational Methods in Sciences and Engineering*, 8(3):213–224.

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

Erin LeDell, Sebastien Poirier, and 1 others. 2020. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020, page 24.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, and 1 others. 2024. Autokaggle: A multi-agent framework for autonomous data science competitions. *arXiv preprint arXiv:2410.20424*.

Petar Maymounkov. 2018. Koji: Automating pipelines with mixed-semantics data sources. *arXiv preprint arXiv:1901.01908*.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, and 1 others. 2025. Mlgym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*.

Nikolay O Nikitin, Pavel Vychuzhanin, Mikhail Sarafanov, Iana S Polonskaia, Ilia Revin, Irina V Barabanova, Gleb Maximov, Anna V Kalyuzhnaya, and Alexander Boukhanovsky. 2022. Automated evolutionary approach for the design of composite machine learning pipelines. *Future Generation Computer Systems*, 127:109–125.

Randal S Olson and Jason H Moore. 2016. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pages 66–74. PMLR.

Alireza Salemi, Mihir Parmar, Palash Goyal, Yiwen Song, Jinsung Yoon, Hamed Zamani, Hamid Palangi, and Tomas Pfister. 2025. Llm-based multi-agent blackboard system for information discovery in data science. *arXiv preprint arXiv:2510.01285*.

Iqbal H Sarker. 2021. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160.

Wonduk Seo, Seungyong Lee, Daye Kang, Zonghao Yuan, and Seunghyun Lee. 2025. Vispath: Automated visualization code synthesis via multi-path reasoning and feedback-driven optimization. *arXiv e-prints*, pages arXiv–2502.

Zhenqian Shen, Yongqi Zhang, Lanning Wei, Huan Zhao, and Quanming Yao. 2018. Automated machine learning: From principles to practices. *arXiv preprint arXiv:1810.13306*.

Evan R Sparks, Shivaram Venkataraman, Tomer Kaftan, Michael J Franklin, and Benjamin Recht. 2017. Keystoneml: Optimizing pipelines for large-scale advanced analytics. In *2017 IEEE 33rd international conference on data engineering (ICDE)*, pages 535–546. IEEE.

Richard S Sutton, Andrew G Barto, and 1 others. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Xiangru Tang, Yuliang Liu, Zefan Cai, Yanjun Shao, Junjie Lu, Yichi Zhang, Zexuan Deng, Helan Hu, Kaikai An, Ruijun Huang, and 1 others. 2023. Ml-bench: Evaluating large language models and agents for machine learning tasks on repository-level code. *arXiv preprint arXiv:2311.09835*.

Soroosh Tayebi Arasteh, Tianyu Han, Mahshad Lotfinia, Christiane Kuhl, Jakob Nikolas Kather, Daniel Truhn, and Sven Nebelung. 2024. Large language models streamline automated machine learning for clinical studies. *Nature Communications*, 15(1):1603.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855.

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

Dakuo Wang, Josh Andres, Justin D Weisz, Erick Oduor, and Casey Dugan. 2021. Autods: Towards human-centered automation of data science. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–12.

Peiran Wang, Yaoning Yu, Ke Chen, Xianyang Zhan, and Haohan Wang. 2025. Large language model-based data science agent: A survey. *arXiv preprint arXiv:2508.02744*.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2024. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xueqing Wu, Rui Zheng, Jingzhen Sha, Te-Lin Wu, Hanyu Zhou, Mohan Tang, Kai-Wei Chang, Nanyun Peng, and Haoran Huang. 2024. Daco: Towards application-driven and comprehensive data analysis via code generation. *Advances in Neural Information Processing Systems*, 37:90661–90682.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, and 1 others. 2023. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*.

Doris Xin, Stephen Macke, Litian Ma, Jialin Liu, Shuchen Song, and Aditya Parameswaran. 2018. Helix: Holistic optimization for accelerating iterative machine learning. *arXiv preprint arXiv:1812.05762*.

Jinglue Xu, Jialong Li, Zhen Liu, Nagar Anthel Venkatesh Suryanarayanan, Guoyuan Zhou, Jia Guo, Hitoshi Iba, and Kenji Tei. 2024a. Large language models synergize with automated machine learning. *arXiv preprint arXiv:2405.03727*.

Shuyuan Xu, Zelong Li, Kai Mei, and Yongfeng Zhang. 2024b. Aios compiler: Llm as interpreter for natural language programming and flow programming of ai agents. *arXiv preprint arXiv:2405.06907*.

Yazheng Yang, Yuqi Wang, Yaxuan Li, Sankalok Sen, Lei Li, and Qi Liu. 2024. Unleashing the potential of large language models for predictive tabular tasks in data science. *arXiv preprint arXiv:2403.20208*.

Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. 2023. Automl-gpt: Automatic machine learning with gpt. *arXiv preprint arXiv:2305.02499*.

A Datasets Released After 2025

Table 4: **Performance on datasets released after LLM pre-training**. Results on three Kaggle datasets published in 2025 (RainFall, BackPack, and Podcast), which were unavailable during the pre-training of the evaluated language models. Human expert denotes the best public Kaggle leaderboard score achieved by a human expert submission (i.e., the top-ranked result at the time of reporting). **Bold** denotes the best result, and underline denotes the second-best result. Results for OpenHands and AIDE are unavailable under `LLaMA3-8B` due to context-length constraints and the lack of native function-calling support.

Methods	RainFall (ACC) ↑	BackPack (RMSE) ↓	Podcast (RMSE) ↓
GPT-4o			
ZeroShot	0.8641	38.8250	12.3384
CoT	0.8655	38.8012	12.3158
Agent K V1.0	0.8691	38.7277	12.2849
Auto Kaggle	0.8879	38.7353	12.3042
OpenHands	0.8882	38.8392	12.4421
Data Interpreter	0.8884	38.7168	12.2549
AIDE	0.8932	38.8318	12.3371
SPIO-S (Ours)	0.8995	38.7071	12.1873
SPIO-E (Top 2 Ensemble)	**0.9004**	**38.6823**	**12.1072**
Claude 3.5 Haiku			
ZeroShot	0.8439	38.8138	12.3419
CoT	0.8497	38.8038	12.3242
Agent K V1.0	0.8526	38.7419	12.2750
Auto Kaggle	0.8719	38.7290	12.2388
OpenHands	0.8789	38.9418	12.4609
Data Interpreter	0.8731	38.7249	12.1948
AIDE	0.8812	38.9195	12.5523
SPIO-S (Ours)	0.8874	38.7071	12.1914
SPIO-E (Top 2 Ensemble)	**0.8923**	**38.6672**	**12.1823**
LLaMA3-8B			
ZeroShot	0.8420	38.8450	12.3700
CoT	0.8470	38.8320	12.3550
Agent K V1.0	0.8501	38.7890	12.2980
Auto Kaggle	0.8549	38.7840	12.3090
Data Interpreter	0.8564	38.7740	12.2760
SPIO-S (Ours)	0.8608	38.7520	12.2450
SPIO-E (Top 2 Ensemble)	**0.8738**	**38.7380**	**12.2278**
Human Expert	0.9065	38.6162	11.4483

Most datasets used in our main experiments were released prior to 2025. As a result, one might raise concerns that such datasets could have been included in the pre-training corpora of large language models, potentially introducing unintended data leakage. To explicitly address this concern, we additionally evaluate our framework on three Kaggle datasets released in 2025, consisting of one classification task and two regression tasks. These datasets are used solely to verify that the observed performance gains stem from the proposed methodology rather than memorization or prior exposure during model pre-training.

Specifically, we include the RainFall (classification), BackPack (regression), and Podcast (regression) datasets, all of which were published in 2025 and were therefore unavailable during the pre-training of the evaluated language models. Experimental results on these datasets demonstrate that **SPIO** continues to outperform strong baselines, confirming that the performance improvements are driven by structured multi-path planning and optimization rather than reliance on pre-trained knowledge. Notably, on the RainFall classification dataset, **SPIO-E** with `GPT-4o` achieves performance within approximately the **top 9%** of the public Kaggle leaderboard at the time of submission, further highlighting the practical competitiveness and robustness of our approach on contemporary, real-world data.

B Supplementary of Main Experiments

Dataset Information Tables 5 and 6 summarize the datasets used in our main experiments, including their sizes, feature dimensionality, task types, and data sources. These benchmarks cover a diverse set of real-world supervised learning tasks from Kaggle and OpenML, spanning binary and multi-class classification as well as regression, with dataset sizes ranging from hundreds to millions of instances.

Failure Rate Analysis. We report the failure rate at K attempts (FR@K) for baseline methods and **SPIO** variants. FR@K is defined as the proportion of tasks that are *not* successfully completed within at most K execution attempts. For example, FR@3 denotes the fraction of tasks that still fail after three execution attempts. Table 7 summarizes FR@K for representative baseline methods and **SPIO-S** using `GPT-4o` and `Claude 3.5 Haiku`. Several single-path or weakly validated approaches (e.g., Agent K v1.0 and AutoKaggle) exhibit relatively high initial failure rates (FR@1), indicating vulnerability to cascading errors caused by an incorrect first-generation output. More execution-driven methods, such as AIDE and OpenHands, also show non-negligible FR@1, reflecting instability in early iterations despite iterative refinement. In contrast, **SPIO-S** rapidly reduces failures, achieving FR@3 = 0 under both LLMs despite a non-zero FR@1. This behavior demonstrates effective early recovery enabled by **SPIO**'s multi-path candidate generation and selection mechanism. All evaluated methods converge to zero failure rates by $K = 10$.

Table 5: **Dataset statistics used in the main experiments.** We report dataset sizes, feature dimensionality, task types, and the fraction of missing values in the training and test splits.

Dataset	#Train	#Test	#Features	Label	Task	Null Ratio (Train/Test)
Kaggle Datasets						
Titanic	891	418	11	Survived	Classification	0.088 / 0.090
House Prices	1460	1459	80	SalePrice	Regression	0.067 / 0.068
Spaceship Titanic	8693	4277	13	Transported	Classification	0.021 / 0.020
Monsters	371	529	6	Type	Classification	0.000 / 0.000
Academic Success	76518	51012	37	Target	Classification	0.000 / 0.000
Bank Churn	165034	110023	13	Exited	Classification	0.000 / 0.000
Obesity Risk	20758	13840	17	NObeyesdad	Classification	0.000 / 0.000
Plate Defect	19219	12814	35	Class	Multi-class	0.000 / 0.000
BackPack	3694318	200000	9	Price	Regression	0.019 / 0.021
Rainfall	2190	730	11	Rainfall	Classification	0.000 / 0.000
Podcast	750000	250000	10	Listening_Time_minutes	Regression	0.031 / 0.031
OpenML Datasets						
Boston Housing	506	–	13	MEDV	Regression	0.000 / –
Diamonds	53940	–	9	Price	Regression	0.000 / –
KC1	2109	–	21	Defects	Classification	0.000 / –
SAT11	4440	–	117	Runtime	Regression	0.000 / –

Table 6: **Summary of datasets used in our experiments**. The table includes benchmark datasets from Kaggle and OpenML, covering a diverse range of supervised learning tasks.

Dataset	Link
Kaggle Datasets	
Titanic	https://www.kaggle.com/competitions/titanic
House Prices	https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques
Spaceship Titanic	https://www.kaggle.com/competitions/spaceship-titanic
Monsters	https://www.kaggle.com/competitions/ghouls-goblins-and-ghosts-boo
Academic Success	https://www.kaggle.com/competitions/playground-series-s4e6
Bank Churn	https://www.kaggle.com/competitions/playground-series-s4e1
Obesity Risk	https://www.kaggle.com/competitions/playground-series-s4e2
Plate Defect	https://www.kaggle.com/competitions/playground-series-s4e3
BackPack	https://www.kaggle.com/competitions/playground-series-s5e2
Rainfall	https://www.kaggle.com/competitions/playground-series-s5e3
Podcast	https://www.kaggle.com/competitions/playground-series-s5e4
OpenML Datasets	
Boston Housing	https://www.openml.org/d/531
Diamonds	https://www.openml.org/d/42225
KC1	https://www.openml.org/d/1067
SAT11	https://www.openml.org/d/41980

To further analyze robustness under constrained model capacity, we measure FR@K at each pipeline stage using `LLaMA-3 8B` (Table 8). The uniformly high initial failure rates (FR@1 = 1.0 across all stages) reflect the difficulty for a smaller LLM to generate fully executable plans or code on the first attempt without iterative refinement. Nevertheless, failure rates decrease sharply with K, often reaching zero by $K = 20$ and, in some cases, much earlier. Bold entries denote the lowest FR@K values within each column, corresponding to the earliest recovery among pipeline stages.

Token Usage Analysis. Figure 4 illustrates the step-wise token usage of all compared frameworks, decomposed into input tokens (prompts, intermediate plans, and execution context) and output tokens (generated code and predictions). Token counts are shown on a logarithmic scale to accommodate the wide variation across methods.

Single-pass baselines such as ZeroShot and CoT exhibit low and stable token usage due to the absence of iterative planning. In contrast, multi-agent pipelines (Agent K, AutoKaggle, and Data Interpreter) consume substantially more tokens as tasks are decomposed into multiple stages with intermediate representations. OpenHands and AIDE show higher token consumption, with interactions dominated by execution and environment feedback rather than explicit algorithmic planning.

SPIO and its variants maintain bounded per-step token usage while supporting sequential planning and selective exploration across modules. This

Table 7: **Failure rate at K attempts (FR@K) across methods using `GPT-4o` and `Claude 3.5 Haiku` (lower is better).**

Score	FR@1	FR@3	FR@5	FR@10
GPT-4o				
ZeroShot	0.08	0	0	0
CoT	0	0	0	0
Agent K V1.0	0.33	0.16	0.08	0
Data Interpreter	0	0	0	0
Auto Kaggle	0.50	0.08	0.08	0
AIDE	0.33	0.08	0	0
OpenHands	0.50	0.16	0.08	0
SPIO-S (Ours)	0.33	0	0	0
Claude 3.5 Haiku				
ZeroShot	0	0	0	0
CoT	0.25	0	0	0
Agent K V1.0	0.17	0.08	0	0
Data Interpreter	0.08	0.08	0	0
Auto Kaggle	0.58	0.17	0	0
AIDE	0.33	0.17	0	0
OpenHands	0.58	0.08	0	0
SPIO-S (Ours)	0.33	0.08	0	0

Table 8: **Failure rate of `LLaMA3-8B` at K attempts of each stage.**

Step	FR@1	FR@3	FR@5	FR@10	FR@15	FR@20
Feature Engineering	1.00	0.89	0.80	0.70	0.35	0
Preprocess	1.00	0.84	0.75	0.56	0.25	0
HyperParameter	1.00	0.80	0.69	0.47	0.25	0
Model Selection	1.00	0.73	0.59	0.35	0.10	0

result indicates that structured, stage-aware multi-path planning can achieve a favorable balance between expressiveness and token efficiency compared to unconstrained autonomous-agent frameworks.

Performance–Token Cost Trade-off. Figure 5 analyzes the trade-off between predictive performance and token cost across different frameworks. Each point represents the average performance (ACC/ROC for classification and RMSE for regression) as a function of the corresponding token consumption, plotted on a logarithmic scale. Single-pass baselines achieve low token cost but relatively weaker performance, whereas multi-agent systems improve accuracy at the expense of increased token usage. OpenHands and AIDE incur substantially higher token costs with comparatively limited performance gains. In contrast, **SPIO-S** and **SPIO-E** consistently achieve superior performance under comparable or lower token budgets, illustrating that structured multi-path planning yields a more favorable performance–efficiency balance.

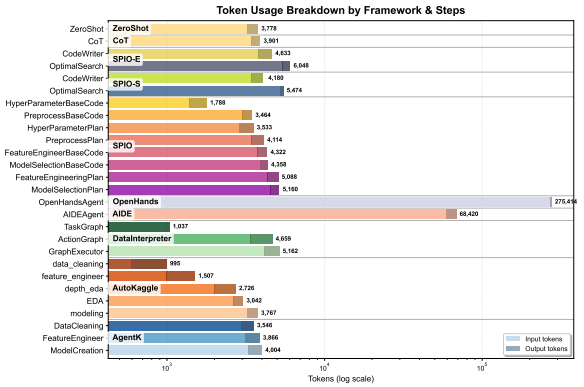


Figure 4: **Token usage breakdown by framework and steps.** Input and output tokens are shown as stacked bars on a log-scaled x-axis.

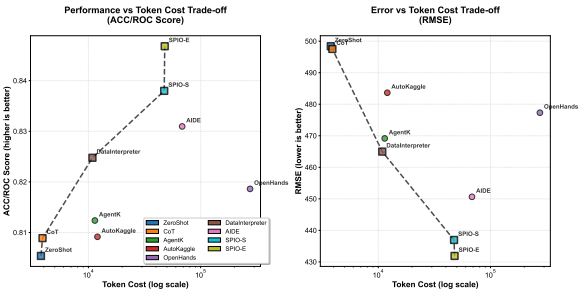


Figure 5: **Performance versus token cost trade-off across different frameworks.** Left: classification performance measured by ACC/ROC (higher is better). Right: regression performance measured by RMSE (lower is better). Token cost is shown on a logarithmic scale.

Qualitative Planning Analysis. We present a concrete comparative example of **SPIO**'s multi-stage, multi-candidate reasoning on the *Monsters* dataset (`Claude-3.5 Haiku` variant). This dataset has a moderate size and feature complexity, making it suitable for analyzing the interpretability of alternative solution paths. At each critical stage—preprocessing, feature engineering, model selection, and hyperparameter optimization—**SPIO** enumerates multiple candidate strategies, evaluates them on validation splits, and retains only the highest-performing pipelines. Table 9 summarizes the top four ranked solution paths (Top 1–Top 4) automatically generated and selected.

Key Distinctions Across Ranked Paths. The top-ranked pipelines differ along four principal axes: (i) the depth and rigor of normalization and distribution diagnostics, (ii) the sophistication of categorical and probabilistic encoding strategies, (iii) the exploitation of interaction or polynomial

Table 9: **Representative high-performing solution paths generated by SPIO on the *Monsters* dataset using Claude 3.5 Haiku**.

Rank	Score	Pipeline Configuration
Top 1	0.7973	**Preprocess:** feature normalization, distribution-aware scaling **Feature:** interaction terms, polynomial expansion **Model/HPO:** XGBoost + Random Forest ensemble (0.5/0.5), Optuna
Top 2	0.6667	**Preprocess:** categorical encoding, feature generation **Feature:** probabilistic encoding **Model/HPO:** XGBoost + SVM ensemble (0.5/0.5), Bayesian optimization (Hyperopt)
Top 3	0.6800	**Preprocess:** robust scaling, outlier handling **Feature:** anomaly-aware feature construction **Model/HPO:** Random Forest + SVM ensemble (0.5/0.5), grid search (CV)
Top 4	0.7255	**Preprocess:** feature normalization **Feature:** probabilistic encoding **Model/HPO:** XGBoost + Random Forest ensemble (0.5/0.5), Bayesian optimization (Hyperopt)

feature expansions, and (iv) the ensemble structure and chosen hyperparameter optimization backend.

Interpretive Factors Driving Top Performance. The Top 1 solution outperforms lower-ranked alternatives primarily due to:

1. **Meticulous normalization and outlier handling**, which reduces variance amplification in downstream interaction terms;
2. **Explicit modeling of nonlinear feature interactions** via polynomial and cross features, improving class separability;
3. **Balanced tree-ensemble composition** (heterogeneous inductive biases of XGBoost and Random Forest), yielding a favorable bias–variance trade-off; and
4. **Efficient hyperparameter optimization** (Optuna), producing well-tuned depth, learning rate, and regularization settings within the allocated budget.

Lower-ranked pipelines rely more heavily on categorical embeddings or simpler ensemble combinations (e.g., Random Forest with SVM) and employ less exhaustive interaction generation. As a result, they achieve modestly lower validation accuracy despite remaining methodologically sound. This case study illustrates how **SPIO**'s evidence-driven "generate–evaluate–select" loop at each stage produces robust, high-quality solutions that surpass fixed or less adaptive baselines. The diversity of candidate plans—ranging from normalization-centric to probabilistic encoding and anomaly-aware strategies—also contributes to robustness, as even non-top pipelines remain competitive.

All plans are generated under identical inference settings (temperature 0.5, maximum tokens 4096). For each ranked plan, we log preprocessing transformations, feature engineering steps, selected model families, ensemble weights, and hyperparameter optimization configurations (algorithm, budget, and final parameters). Serialized JSON descriptors are retained for auditability.

C Qualitative Expert Evaluation

Participants and Procedure. We recruited 10 participants who either work in AI-related professional roles or hold graduate-level degrees in relevant fields. Each participant evaluated **SPIO**-generated pipelines on 10 datasets (the same datasets used in the quantitative benchmarks), resulting in a total of 100 dataset–participant evaluation pairs.

For each dataset, **SPIO** generates plans for the main stages of the data science pipeline—preprocessing, feature engineering, model selection, and hyperparameter tuning—as well as two final "optimal" pipelines: one selected by the sequential single-path framework (**SPIO-S**) and one selected by the ensemble framework (**SPIO-E**). Participants completed twenty 5-point Likert-scale questions (1 = strongly disagree, 5 = strongly agree) and one multiple-choice question per dataset.

Stage-wise Results. Table 10 reports the average subjective ratings for each **SPIO** pipeline component and for the two final optimal pipelines, aggregated across all datasets and participants.

Plan-Choice Task (Discrete Agreement with LLM Ranking). While Likert-scale ratings capture graded preferences, they do not directly assess whether the LLM's internal ranking of candidate plans aligns with experts' discrete choices. To evaluate this alignment, we included a multiple-choice plan-selection task.

Table 10: **Average subjective ratings (mean and standard deviation) for each SPIO pipeline component and final optimal method (1 = strongly disagree, 5 = strongly agree).**

Pipeline component	Mean	Std. dev.
Preprocessing	4.07	0.17
Feature engineering	4.13	0.17
Model selection	4.20	0.17
Hyperparameter selection	4.09	0.19
Optimal method (SPIO-S)	4.24	0.18
Optimal method (SPIO-E)	4.45	0.15

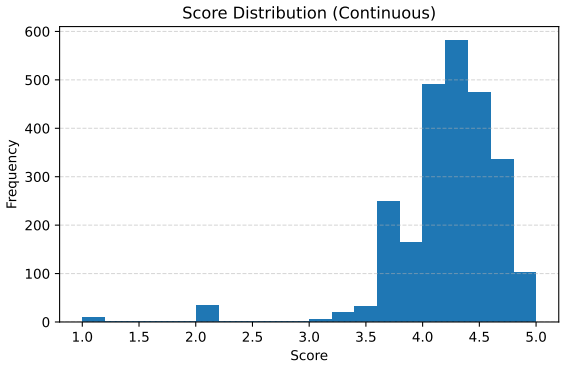


Figure 6: **Histogram of subjective Likert ratings aggregated over participants, datasets, pipeline stages, and final methods.**

For each dataset, **SPIO-E** generates three candidate "optimal" pipelines (Plan 1–Plan 3), corresponding to the top-3 combinations according to the LLM's internal scoring. As shown in Figure 7, participants were shown these three anonymized pipelines and asked to select the one they considered best suited for the given dataset. This procedure yields 10 datasets $\times$ 10 participants = 100 independent selection tasks. Aggregated results show that Plan 1 was selected in 72 cases, whereas Plan 2 and Plan 3 were chosen 22 and 6 times, respectively.

Statistical Testing. To compare **SPIO-S** and **SPIO-E** on the final optimal pipelines, we conducted paired t-tests for each subjective evaluation dimension and applied the Holm–Bonferroni correction to control the family-wise error rate across dimensions. We report statistically significant differences at $p < 0.05$ after correction.

D Diversity Visualization Details

Embedding and PCA Procedure. We visualize the diversity of **SPIO** generated plans by embedding each plan's textual description using the *text-embedding-ada-002* model from OpenAI. For a given set of plan texts within a module, we obtain a fixed-dimensional embedding vector for each plan and apply principal component analysis (PCA) to project the embeddings into two dimensions for visualization.

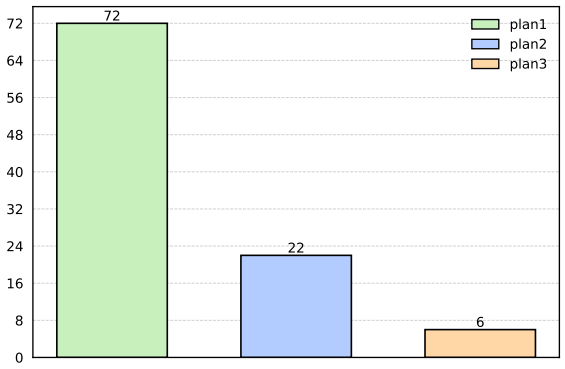


Figure 7: **Expert choices among the top-3 final plans generated by SPIO-E (Plan 1-3)**. Bars show how often each plan was chosen versus not chosen across all choice tasks.

These visualizations are intended as qualitative indicators of diversity. A broad spatial spread and the presence of multiple regions in the two-dimensional projection suggest that the generated plans differ substantially in their semantic content, rather than being minor variations of a single template.

Additional Module Distributions. In addition to the preprocessing and feature engineering visualizations presented in the main paper (Figure 3), we also visualize the distributions of plans generated for the *Model Selection* and *Hyperparameter Tuning* modules. Figure 8 illustrates how the top candidate plans emphasize different model families and hyperparameter optimization strategies. Although individual plans may prioritize specific design choices, the candidate sets collectively span a wide range of approaches, indicating that **SPIO** explores diverse alternatives rather than converging on a single preferred tool or configuration.

Interpretation Notes. The PCA-based visualizations provide qualitative evidence of plan diversity and should not be interpreted as calibrated or quantitative diversity metrics. In particular, distances in the two-dimensional projection do not necessarily preserve all relationships in the original high-dimensional embedding space, and the embedding model itself induces a specific semantic geome-

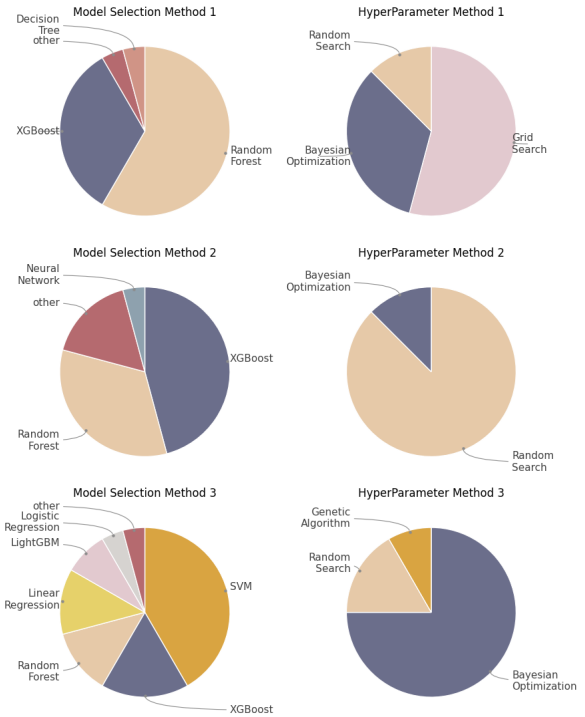


Figure 8: **Distribution of Model and Hyperparameter selection.** Distribution of methods across the top-3 candidate model-selection and hyperparameter plans.

try. Nevertheless, the consistently broad spread observed across modules suggests that **SPIO** does not collapse to a rigid, single-path pipeline, but instead generates multiple distinct and semantically diverse candidate strategies.

E Used Libraries in each Module

Table 11: **Whitelisted Python libraries/classes by pipeline step**. Only these modules (and Python standard library utilities) are invoked by agent-generated code.

Step	Libraries / Classes / Functions
Preprocessing	`numpy`, `pandas`
Feature Engineering	`numpy`, `pandas`, `sklearn.impute:SimpleImputer`, `sklearn.preprocessing:LabelEncoder`, `MinMaxScaler`, `PolynomialFeatures`, `RobustScaler`, `StandardScaler`, `warnings`
Model Selection	`numpy`, `pandas`, `sklearn.compose:ColumnTransformer`, `sklearn.ensemble:GradientBoostingRegressor`, `RandomForestClassifier`, `RandomForestRegressor`, `sklearn.impute:SimpleImputer`, `sklearn.linear_model:Lasso`, `LinearRegression`, `LogisticRegression`, `Ridge`, `sklearn.metrics:accuracy_score`, `classification_report`, `mean_absolute_error`, `mean_squared_error`, `r2_score`, `roc_auc_score`, `sklearn.model_selection:train_test_split`, `sklearn.multioutput:MultiOutputClassifier`, `sklearn.pipeline:Pipeline`, `sklearn.preprocessing:LabelEncoder`, `OneHotEncoder`, `StandardScaler`
Hyperparameter Tuning	All of the above + `sklearn.model_selection:GridSearchCV`, `sklearn.svm:SVC`, `xgboost:XGBClassifier`

F Prompt

Table 12: **Prompt for Planning Agent.**

Prompt Template
Your task: • Your task is to perform: `{task}`. • Focus solely on `{task}`. Performing any unrelated task is strictly prohibited. • Generate a simple and executable code for `{task}`, including explanations of the data and task. • **Important:** Never drop any rows with missing values in the test data. The length of the test set must remain unchanged. • Your final output should save the preprocessed data as CSV files. **Input files (read using `read_csv`):** • Train data file path: `{train_input_path}` • Test data file path: `{test_input_path}` **Output files (save using `to_csv`) — only for preprocessing or feature engineering:** • Train data output path: `{train_output_path}` • Test data output path: `{test_output_path}` **Summarized Original Data:** `{data}` **Task:** `{task}` **Previous Code (excluding preprocessing):** `{code}`

Table 13: **Prompt for Code Agent.**

Prompt Template

Your task:
• You are an expert in `{task}`. Your goal is to solve the task using the given information.
• Propose up to **two alternative methods** (plans) to achieve the task.
• For each method, clearly explain:
  – In which scenario it is most suitable
  – Why it is recommended
• Identify and recommend the best solution based on the provided context.
Preprocessing Code: `{pp_code}`
Summarized Data: `{pp_data_result}`
Task Description: `{task}`

Table 14: **Prompt for SPIO-S Optimal Method Agent.**

Prompt Template
Your task: Select one best plan from each of the following: preprocessing, feature engineering, model selection, and hyperparameter tuning. Combine these into a single pipeline aimed at achieving the highest validation score. For model selection, you may use multiple models in an ensemble — each model should have equal weighting. Your goal is to determine the best-performing pipeline configuration. **Data Summary:** `{data}` **Task Description:** `{task}` **Preprocessing Plans:** `{preprocess_plan}` **Feature Engineering Plans:** `{feature_engineer_plan}` **Model Selection Plans:** `{model_select_plan}` **Hyperparameter Tuning Plans:** `{hyper_opt_plan}`

Table 15: **Prompt for SPIO-E Optimal Method Agent.**

Prompt Template

Your task: Select the top **two** pipelines by combining one plan from each module: preprocessing, feature engineering, model selection, and hyperparameter tuning. For model selection, you may include ensembles — each model must be equally weighted. Return the result strictly in **JSONL format** (a list of JSON objects), without any additional explanation or text. Each value must exactly match the corresponding original content — no summarization, simplification, or paraphrasing is allowed. All fields (`"preprocess"`, `"feature_engineering"`, `"model_selection"`, `"optimal_hyper_tool"`) must be clearly and precisely filled.
Data Summary: `{data}`
Task Description: `{task}`
Preprocessing Plans: `{preprocess_plan}`
Feature Engineering Plans: `{feature_engineer_plan}`
Model Selection Plans: `{model_select_plan}`
Hyperparameter Tuning Plans: `{hyper_opt_plan}`
Example Output Format:
`[`
`  {`
`  "rank": "top1",`
`  "best_combine": {`
`  "preprocess": "...",`
`  "feature_engineering": "...",`
`  "model_selection": "...",`
`  "optimal_hyper_tool": "..."`
`  }`
`  },`
`  ...`
`]`
Ensure that the output is valid JSON that can be parsed using `json.load()`.

G Case Study

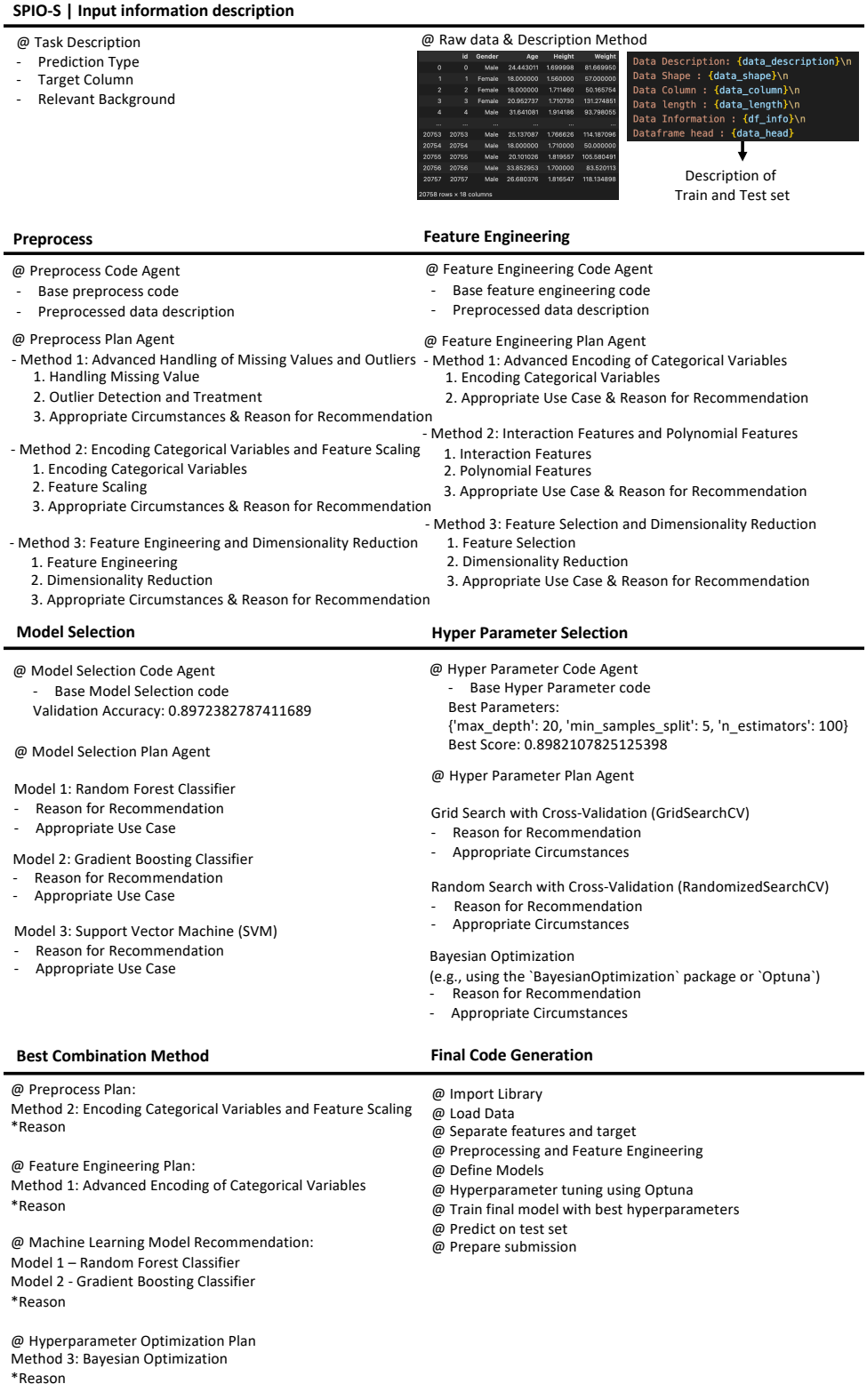
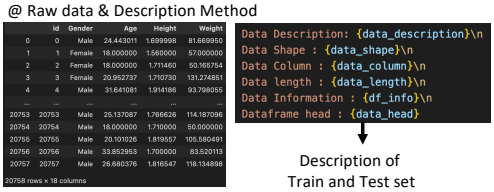


Figure 9: **An overview of the SPIO-S pipeline, illustrating how agents perform preprocessing, feature engineering, model selection, and hyperparameter tuning**. Among all possible strategies, the system identifies a single best combination plan, which is then used to generate the final executable code.

SPIO-E | Input information description

@ Task Description
- Prediction Type
- Target Column
- Relevant Background

@ Raw data & Description Method



	id	Gender	Age	Height	Weight
0	0	Male	24.443011	1.699998	81.669950
1	1	Female	18.000000	1.560000	57.000000
2	2	Female	18.000000	1.711460	50.165754
3	3	Female	20.952737	1.710730	131.274851
4	4	Male	31.641081	1.914198	93.798055
...	...	...	...	...	...
20753	20753	Male	25.137087	1.766626	114.187096
20754	20754	Male	18.000000	1.710000	50.000000
20755	20755	Male	20.101026	1.819557	105.580491
20756	20756	Male	33.852953	1.700000	83.520113
20757	20757	Male	26.680376	1.816547	118.134898

20758 rows × 18 columns

```
Data Description: {data_description}\n
Data Shape : {data_shape}\n
Data Column : {data_column}\n
Data length : {data_length}\n
Data Information : {df_info}\n
Dataframe head : {data_head}
```

Description of
Train and Test set

Preprocess

@ Preprocess Code Agent
- Base preprocess code
- Preprocessed data description

@ Preprocess Plan Agent
- Method 1: Advanced Imputation and Encoding
  1. Missing Value Imputation
  2. Encoding Categorical Variables
  3. Appropriate Use Case & Reason for Recommendation
- Method2 :Feature Scaling and Normalization
  1. Feature Scaling
  2. Normalization
  3. Appropriate Use Case & Reason for Recommendation
- Method3 : Outlier Detection and Handling
  1. Outlier Detection
  2. Handling Outliers
  3. Appropriate Use Case & Reason for Recommendation

Feature Engineering

@ Feature Engineering Code Agent
- Base feature engineering code
- Preprocessed data description

@ Feature Engineering Plan Agent
- Method 1: Advanced Imputation and Encoding
  1. Missing Value Imputation
  2. Encoding Categorical Variables
  3. Appropriate Use Case & Reason for Recommendation
- Method2 :Feature Scaling and Normalization
  1. Feature Scaling
  2. Normalization
  3. Appropriate Use Case & Reason for Recommendation
- Method3 : Outlier Detection and Handling
  1. Outlier Detection
  2. Handling Outliers
  3. Appropriate Use Case & Reason for Recommendation

Model Selection

@ Model Selection Code Agent
- Base Model Selection code

Validation Accuracy: 1

@ Model Selection Plan Agent

Model 1: Random Forest Classifier
- Reason for Recommendation
- Advantages

Model 2: Gradient Boosting Classifier
- Reason for Recommendation
- Advantages

Model 3: Support Vector Machine (SVM)
- Reason for Recommendation
- Advantages

Hyper Parameter Selection

@ Hyper Parameter Code Agent

Best parameters found: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}
Best Score: 0.8949805219271387

@ Hyper Parameter Plan Agent

GridSearchCV (Scikit-learn)
- Reason for Recommendation
- Circumstances for Use

RandomizedSearchCV (Scikit-learn)
- Reason for Recommendation
- Circumstances for Use

Bayesian Optimization
(e.g., using the `hyperopt` or `Optuna` library)
- Reason for Recommendation
- Circumstances for Use

Best Combination Method Agent

@ Top1
```
"rank": "top1",
  "best_combine": {
    "preprocess": "Advanced Imputation and Encoding",
    "feature_engineering": "Advanced Imputation and Encoding",
    "model_selection": "Random Forest Classifier and Gradient Boosting
                        Classifier with ensemble weight 0.5 each",
    "optimal_hyper_tool": "GridSearchCV (Scikit-learn)"
```

@ Top2
```
"rank": "top2",
  "best_combine": {
    "preprocess": "Feature Scaling and Normalization",
    "feature_engineering": "Advanced Imputation and Encoding",
    "model_selection": "Random Forest Classifier and Gradient Boosting
                        Classifier with ensemble weight 0.5 each",
    "optimal_hyper_tool": "RandomizedSearchCV (Scikit-learn)"
```

Final Code Generation Agent

@ Import Library
@ Load Data
@ Prepare Data
@ Identify categorical and numerical columns
@ First pipeline
  @ Model
  @ Pipeline
@ Second pipeline
  @ preprocessing
  @ Model
  @ Pipeline
@ Hyper parameter tuning using Optuna
@ Train pipelines
@ Predict and ensemble results
@ Ensemble Predictions

Figure 10: **An overview of the SPIO-E pipeline, illustrating how agents perform preprocessing, feature engineering, model selection, and hyperparameter tuning**. Based on their outputs, the system selects the 2 best combination plans and produces the final code for execution.