Memory-Augmented LLM-based Multi-Agent System for Automated Feature Generation on Tabular Data

Anonymous ACL submission

Abstract

Automated feature generation extracts informative features from raw tabular data without manual intervention and is crucial for accurate, generalizable machine learning. Traditional methods rely on predefined operator libraries and cannot leverage task semantics, limiting their ability to produce diverse, high-value features for complex tasks. Recent Large Language Model (LLM)-based approaches introduce richer semantic signals, but still suffer from a restricted feature space due to fixed generation patterns and from the absence of feedback from the learning objective. To address these challenges, we propose a Memory-Augmented LLM-based Multi-Agent System (**MALMAS**) for automated feature generation. MALMAS decomposes the generation process into agents with distinct responsibilities, and a Router Agent activates an appropriate subset of agents per iteration, further broadening exploration of the feature space. We further integrate a memory module comprising procedural memory, feedback memory, and conceptual memory, enabling iterative refinement that adaptively guides subsequent feature generation and improves feature quality and diversity. Extensive experiments on multiple public datasets against state-of-the-art baselines demonstrate the effectiveness of our approach. The code is available at https://anonymous.4open.science/r/MALMAS-63DB

1 Introduction

Recently, the advancement of Automated Machine Learning (**AutoML**) has greatly improved the efficiency of data modeling (Trirat et al., 2025; Guo et al., 2024; Xu et al., 2024; Jeong et al., 2025; Wei et al., 2024). Within this paradigm, *automated feature generation*, which extracts informative features from raw data without manual intervention, has become a key enabler for building accurate and generalizable models.

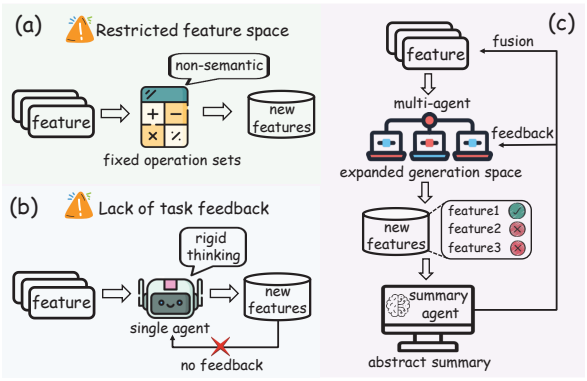

Figure 1: Comparison of traditional, LLM-based, and multi-agent feature generation approaches.

However, traditional automated feature generation methods still suffer from several limitations, which hinder their ability to produce high-quality features effectively. As illustrated in Figure 1(a), these methods apply a predefined set of operators to original features to construct new feature sets (Horn et al., 2019; Kanter and Veeramachaneni, 2015; Zhang et al., 2023). They rely on limited operator sets and do not incorporate feature semantics, which confines the transformation search space to a narrow region.

Recently, large language models (LLMs) have shown strong semantic understanding and generation capabilities (Matarazzo and Torlone, 2025), motivating LLM-based feature generation that leverages task descriptions to propose transformations (Hollmann et al., 2023; Nam et al., 2024). However, these methods typically rely on static, singular generation strategies rooted in rigid thinking, which still constrain the exploration of broader feature spaces. More critically, these methods lack mechanisms to adapt generation strategies based on historical experience or task-specific feedback. Without such adaptive signals, feature generation becomes disconnected from learning performance, leading to inefficient, trial-and-error exploration and limited ability to prioritize high-value transfor-

mations. This feedback-insensitive process undermines learning-goal alignment and hampers methods' performance, as illustrated in Figure 1(b).

To this end, we propose the Memory-Augmented LLM-based Multi-Agent System (**MALMAS**), an automated feature generation framework, as illustrated in Figure 1(c).

Specifically, we decompose feature generation into multiple independent agents with roles grounded in a principled framework along three largely orthogonal dimensions from established feature engineering practice: transformation complexity, data scope, and data-type dependency. Inspired by the categorization of high-value "golden features" (Zhang et al., 2024), this design assigns each agent a clear, specialized responsibility. A Router Agent dynamically selects a subset of agents from a predefined pool based on task metadata and accumulated memory, enabling adaptive allocation of generation effort. Each agent then conducts multi-turn interactions with the LLM, constructing role-specific prompts conditioned on the current feature set and experiential feedback. By exploring complementary regions of the feature space, the agents mitigate feature homogenization in static single-agent strategies (Hollmann et al., 2023) and reduce functional redundancy, thereby broadening the overall search space.

To address the lack of feedback-driven adaptability, we equip MALMAS with a multi-level memory that enables credit assignment and strategy updates across rounds. Procedural memory caches executed transformations to suppress redundant exploration, feedback memory attributes validation utility to generated features, and conceptual memory abstracts reusable heuristics from historical traces for longer-horizon adaptation. A Summary Agent aggregates cross-agent feedback and concepts into a global conceptual memory that conditions subsequent routing and prompting. This design turns per-round evaluations into persistent learning signals, steering generation toward high-yield, task-relevant transformations.

The main contributions of this paper are summarized as follows:

- We propose the first multi-agent framework for automated feature generation, enabling collaborative exploration beyond predefined operators and improving feature diversity.
- We develop a multi-level memory mechanism that integrates procedural traces, feedback, and conceptual abstractions, allowing agents to iteratively refine their strategies.
- We evaluate MALMAS on 16 classification and 7 regression datasets, where it outperforms baselines, and we provide practical analyses demonstrating real-world applicability and efficiency.

2 Related Work

2.1 Traditional and LLM-enhanced AutoML

Before the emergence of LLMs, end-to-end AutoML systems such as Auto-WEKA, Auto-sklearn, H2O AutoML, Google AutoML Tables, and FLAML (Thornton et al., 2013; Feurer et al., 2015; Olson and Moore, 2016; LeDell and Poirier, 2020; Wang et al., 2021; Feurer et al., 2022) mainly focused on pipeline search, hyperparameter optimization, and model selection.

With the advent of LLMs, AutoML has increasingly adopted natural language as an interface for automation. Methods such as Text-to-ML, LLM-Select, DS-Agent, and GL-Agent (Trirat et al., 2025; Guo et al., 2024; Xu et al., 2024; Jeong et al., 2025; Wei et al., 2024) employ LLMs to generate or recommend ML pipelines, leveraging instruction following and agentic workflows to reduce manual pipeline design.

However, most of these systems emphasize model and pipeline configuration, while feature construction remains limited to basic preprocessing operations (Gu et al., 2024). This indicates a gap between LLM-enhanced AutoML pipelines and domain-aware feature engineering.

2.2 Automated Feature Generation

Feature generation is a long-standing and critical step for improving model performance. Traditional methods such as autofeat (Horn et al., 2019), Deep Feature Synthesis (DFS) (Kanter and Veeramachaneni, 2015), and OpenFE (Zhang et al., 2023) apply symbolic transformations over predefined operator sets. DFS, implemented in Featuretools, demonstrates the practicality of compositional operators by automatically generating features from relational data. While these methods are efficient and interpretable, they are constrained by fixed operator libraries and limited adaptation to task-specific semantics. More recently, LLM-based methods enable semantically driven generation (Hollmann et al., 2023; Nam et al., 2024; Abhyankar et al., 2025). CAAFE (Hollmann et al., 2023) uses task descriptions to better align gen-

erated features with downstream objectives, and OCTree combines LLMs with tree-based reasoning to support feature validation and interpretability.

2.3 Multi-Agent Systems

LLM-based multi-agent systems have emerged as a promising paradigm for collaboration, specialization, and iterative reasoning (Wang et al., 2024; Li et al., 2024a). They have been applied to social simulation (e.g., Generative Agents and AgentSociety (Park et al., 2023; Piao et al., 2025)), software development (e.g., AutoGen and CodeAct (Wu et al., 2024; Hong et al., 2024)), and decision-making via multi-agent debate with sparse communication (Chan et al., 2023; Liu et al., 2024; Li et al., 2024b; Liang et al., 2024). Recent methods such as ReAct and Reflexion further highlight the role of memory and feedback-driven reasoning-action loops for continual improvement (Yao et al., 2023; Shinn et al., 2023), and SciAgents extends multi-agent collaboration to scientific discovery (Ghafarollahi and Buehler, 2024). Despite these advances, multi-agent systems for automated feature generation remain underexplored.

3 Problem Formulation

Given a labeled tabular dataset $\mathbb{D} = (X, y)$, where $X \in \mathbb{R}^{m \times n}$ is the feature matrix with m instances and n features, and $y \in \mathbb{R}^m$ is the corresponding label vector. The goal of feature generation is to find a transformation function $T : X \to \widetilde{X}$, where $\widetilde{X} = X \cup T(X)$, that improves the predictive performance of a model f when trained on the enhanced feature space. Formally, the objective is to maximize the validation performance of $\mathcal{F}$:

$$T^* = \arg\max_T \mathcal{E}(\mathcal{F}(X_{\text{val}} \cup T(X_{\text{val}})), Y_{\text{val}}), \quad (1)$$

where $\mathcal{E}$ is the evaluation metric, and $(X_{\text{val}}, Y_{\text{val}})$ is the validation set from cross-validation. Here, T^* denotes the optimal transformation function that produces feature beneficial for model performance.

4 Methodology

Feature engineering for tabular data requires diverse, context-aware transformations, yet most automated methods rely on a single strategy or weakly coupled modules, limiting broad, task-relevant exploration. We propose **MALMAS**, which coordinates specialized agents to generate and refine features. With role-specific agents and shared procedural, feedback, and conceptual memories, MALMAS enables iterative exploration and underpins the pipeline in Figure 2.

4.1 Multi-agent Structure

To address the limited ability of a single generator to deeply explore novel features, MALMAS maintains a pool of specialized agents and employs a Router Agent to activate an appropriate subset per iteration. This design increases the diversity and adaptability of generated features while avoiding unnecessary exploration by inapplicable strategies.

4.1.1 Parallel Generation Architecture

MALMAS maintains an agent pool $\mathcal{A} = \{A_i\}_{i=1}^K$, where each agent implements a distinct feature transformation strategy. At iteration r, a Router Agent selects an active subset $\mathcal{A}^{(r)} \subseteq \mathcal{A}$, and only the selected agents run in parallel to explore complementary feature interactions, transformations, and compositions. Over multiple rounds, this design adapts to diverse feature types and modeling needs through heterogeneous strategies. The overall process is formulated as:

$$T^{(r)} = \bigcup_{A_i \in \mathcal{A}^{(r)}} T_i^{(r)}, \quad k_r = \left|\mathcal{A}^{(r)}\right|. \quad (2)$$

Here, $T^{(r)}$ denotes the aggregated set of features generated in the r-th round, and $T_i^{(r)}$ represents the subset produced by agent A_i when activated.

In each round, each active agent $A_i \in \mathcal{A}^{(r)}$ independently generates a subset of new features $T_i^{(r)}$ from the current dataset X by applying its designated strategy. Taking the union of the activated agents' outputs yields an enriched and more comprehensive feature space for model training.

4.1.2 Agent Responsibilities

To systematically explore the vast feature space, MALMAS adopts a principled multi-agent framework that decomposes feature generation along three largely orthogonal dimensions from feature engineering practice: transformation complexity, data scope, and data-type dependency. Inspired by the categorization of high-value "golden features" (Zhang et al., 2024), this design encourages agents to explore complementary aspects of the data, increasing feature diversity while reducing functional redundancy.

Each agent A_i applies a distinct transformation strategy $f_i(\cdot)$ to the dataset X, producing a feature subset T_i aligned with its objective:

$$T_i = f_i(X). \quad (3)$$

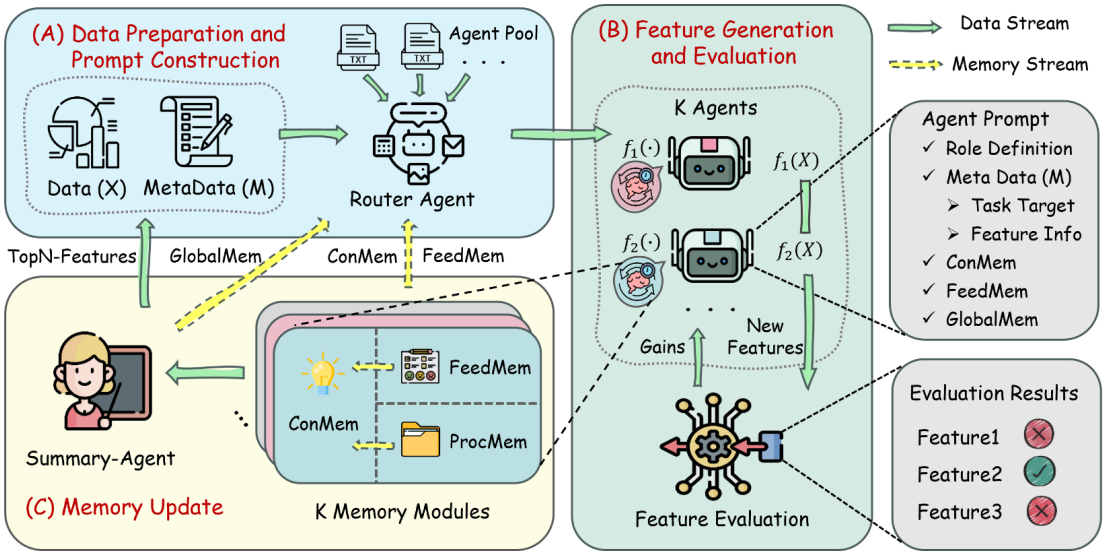

Figure 2: Overview of the proposed MALMAS framework. (A) Agents construct prompts from metadata, selected features, and memory. (B) Agents generate and evaluate candidate features with a downstream model. (C) Agent memories are summarized into a global memory to guide subsequent rounds.

We instantiate a fixed pool of strategy agents as follows, from which the Router Agent activates a subset at each iteration:

- **Unary-Feature Agent**. Applies unary transformations $f_{\text{unary}}(X)$ to individual features to generate basic but informative variants.
- **Cross-Compositional Agent**. Combines multiple inputs $f_{\text{compositional}}(X)$ to capture higher-order interactions.
- **Temporal-Feature Agent**. Extracts temporal patterns $f_{\text{temporal}}(X)$ for time-series data.
- **Aggregation-Construct Agent**. Generates group-level summary features $f_{\text{aggregation}}(X)$.
- **Local-Transform Agent**. Applies region-specific transformations $f_{\text{local-transform}}(X)$ to capture locally informative patterns.
- **Local-Pattern Agent**. Discovers latent patterns within feature subsets via clustering or local interaction modeling $f_{\text{local-pattern}}(X)$.

4.2 Memory Architecture and Management

Feature generation in MALMAS is formulated as an iterative search over transformations, where learning signals from downstream evaluation are *persisted* and *reused* to refine future generation. As illustrated in Figure 2(C), each agent maintains a structured memory state that supports cross-round *credit assignment* and *strategy refinement*, thereby turning expensive feedback into reusable guidance.

Formally, at iteration r, each agent A_i maintains an explicit, structured memory state $\mathcal{M}_i^{(r)} = \{\text{ProcMem}_i^{(r)}, \text{FeedMem}_i^{(r)}, \text{ConMem}_i^{(r)}\}$. At the beginning of round r, the agent retrieves its local memories together with the shared $\text{GlobalMem}^{(r-1)}$ to condition prompt construction; after feature evaluation, it appends new traces and utilities to update $\mathcal{M}_i^{(r)}$. Intuitively, procedural memory captures *what was tried*, feedback memory captures *what worked*, and conceptual memory captures *why it worked* in a compact form.

4.2.1 Procedural Memory

Procedural memory serves as an *execution trace* that records the concrete transformation actions performed by agent A_i, enabling reproducibility and constraining redundant exploration. In iteration r, after generating $n_i^{(r)}$ features:

$$\text{ProcMem}_i^{(r)} = \{(b_j, t_j, f_j, d_j, r)\}_{j=1}^{n_i^{(r)}}, \quad (4)$$

where b_j denotes the base columns, t_j the transformation type, f_j the generated feature name, d_j the transformation description, and r the iteration index. During subsequent rounds, ProcMem_i is used to avoid duplicate transformations and to discourage patterns that repeatedly fail under evaluation.

4.2.2 Feedback Memory

Feedback memory provides a *utility signal* by associating each generated feature with its downstream validation outcome, enabling explicit credit assignment for feature transformations. For agent A_i in

Algorithm 1 Iterative Feature Generation

Input: $X^{(1)}$, y, metadata M, rounds R,
agent pool $\mathcal{A} = \{A_i\}_{i=1}^{K}$, metric E
Output: $X^{(R+1)}$

Initialize memories for all agents $A_i \in \mathcal{A}$
for $r = 1, 2, \ldots, R$ **do**
  $\mathcal{A}^{(r)} \leftarrow \texttt{Route}(M, \text{GlobalMem}^{(r-1)})$
  for each active agent $A_i \in \mathcal{A}^{(r)}$ **do**
    $p_{\text{mem}} \leftarrow \{\text{FeedMem}_i^{(r-1)}, \text{ConMem}_i^{(r-1)}, \text{GlobalMem}^{(r-1)}\}$
    $p \leftarrow \texttt{ConstructPrompt}(M, p_{\text{mem}})$
    $T_i^{(r)} \leftarrow \pi_\theta(p, X^{(r)})$ {Generate $f_i^{(r)}$}
    gains $\leftarrow \mathcal{E}((T_i^{(r)} \cup X^{(r)}), y)$
    $\texttt{Update}$ $(\text{ProcMem}_i^{(r)}, \text{FeedMem}_i^{(r)}, \text{ConMem}_i^{(r)})$
  end for
  $\text{Mem}^{(r)} \leftarrow \bigcup_{A_i \in \mathcal{A}^{(r)}} \left(\text{ConMem}_i^{(r)} \cup \text{FeedMem}_i^{(r)}\right)$
  $\text{GlobalMem}^{(r)} \leftarrow \texttt{Summary}(\text{Mem}^{(r)})$
  $T^{(r)} \leftarrow \bigcup_{A_i \in \mathcal{A}^{(r)}} T_i^{(r)}$
  $F^{(r)} \leftarrow \bigcup_{A_i \in \mathcal{A}^{(r)}} \text{FeedMem}_i^{(r)}$
  $S^{(r)} \leftarrow \texttt{TopN-Features}(T^{(r)}, F^{(r)})$
  $X^{(r+1)} \leftarrow X^{(r)} \cup S^{(r)}$
  $M \leftarrow \texttt{UpdateMetadata}(M, S^{(r)})$
end for
return $X^{(R+1)}$

iteration r with $n_i^{(r)}$ generated features:

$$\text{FeedMem}_i^{(r)} = \{(f_j, m, v_j, e_j, r)\}_{j=1}^{n_i^{(r)}}, \quad (5)$$

where f_j is the feature name, m is the evaluation metric, v_j is the metric value, e_j indicates whether the feature is effective, and r is the iteration index. This memory enables *utility attribution* by linking each feature to validation gain, which biases later rounds toward high-yield transformations and away from noisy or low-impact candidates.

4.2.3 Conceptual Memory

Conceptual memory stores a compact set of reusable heuristics distilled from an agent's historical traces and utilities. After each round, the LLM summarizes $\text{ProcMem}_i^{(r)}$ and $\text{FeedMem}_i^{(r)}$ into rules that guide subsequent generation:

$$\text{ConMem}_i^{(r)} = \text{LLM}\left(\text{ProcMem}_i^{(r)}, \text{FeedMem}_i^{(r)}\right). \quad (6)$$

By compressing experience into high-level guidance, ConMem_i supports strategy adaptation across rounds keeping the prompt context concise.

4.2.4 Global Conceptual Memory

To promote coordination and knowledge transfer across agents, after each iteration the SummaryAgent aggregates agents' local conceptual and feedback memories into a Global Conceptual Memory. This cross-agent consolidation forms a shared prior for the next round, propagating effective transformation heuristics across roles, reducing overlap among agents, and improving the efficiency of subsequent exploration and refinement.

4.3 Iterative Feature Generation

This section describes the feature generation mechanism of our multi-agent system, as shown in Figure 2. Across iterative rounds, agents leverage local and global memories to refine their strategies.

In each iteration r, each active agent $A_i \in \mathcal{A}^{(r)}$ independently executes a fixed sequence of steps, as detailed in Algorithm 1:

- **Prompt Construction**: Each agent constructs a prompt from statistics and metadata M, effective features from $\text{FeedMem}_i^{(r-1)}$, and distilled guidance from $\text{ConMem}_i^{(r-1)}$ and $\text{GlobalMem}^{(r-1)}$.
- **Feature Generation and Evaluation**: Conditioned on the prompt, the agent uses π_θ to propose a transformation, instantiates it as $f_i^{(r)}$, evaluates the resulting features under $\mathcal{E}$, and stores the feedback in $\text{FeedMem}_i^{(r)}$.
- **Memory Update**: To guide the next round, the agent updates $\text{ProcMem}_i^{(r)}$ and $\text{ConMem}_i^{(r)}$ with attempted operations, effective transformations, and newly identified patterns.

At the end of iteration r, the system selects top-performing features generated by the activated agents and integrates them into the dataset. Specifically, each active agent $A_i \in \mathcal{A}^{(r)}$ applies $\texttt{TopN-Features}$ to $T_i^{(r)}$ to retain the highest-ranked features under $\mathcal{E}$, and the selected features are aggregated to expand the global dataset. This iterative selection-and-aggregation procedure accumulates high-quality transformations and yields progressive improvements in model performance.

5 Experiments

5.1 Experimental Setup

5.1.1 Datasets

Following prior work (Hollmann et al., 2023; Abhyankar et al., 2025), we evaluated our method on 16 classification and 7 regression datasets sourced from Kaggle and UCI. Following (Nam et al., 2024), we used a 60–40 train–test split and repeated each experiment three times with different random seeds. Dataset details are in Appendix A.

Table 1: Performance (AUC) of all methods on 16 classification datasets. Best results are in bold, second-best are underlined. Results are averaged across three random train-test splits using the XGBoost classifier. "N/A" indicates that the running time exceeded 12 hours.

Datasets	Base	Traditional Methods			LLM-based Methods			MALMAS
		DFS	AutoFeat	OpenFE	CAAFE	OCTree	LLMFE	
Adult	0.849±0.009	0.857±0.001	0.849±0.009	0.849±0.009	0.868±0.005	0.845±0.011	0.853±0.011	**0.875±0.010**
Balance	0.908±0.009	0.989±0.009	0.908±0.009	0.908±0.009	**1.000±0.000**	0.933±0.031	0.994±0.008	**1.000±0.000**
Bank	0.869±0.007	0.891±0.014	0.869±0.007	**0.904±0.003**	0.874±0.022	0.873±0.010	0.875±0.012	0.895±0.002
Banknote	0.995±0.002	0.998±0.001	0.994±0.003	0.998±0.001	0.993±0.002	0.993±0.004	0.991±0.002	**0.999±0.001**
Breast_W	0.989±0.002	0.988±0.003	0.989±0.002	0.988±0.003	**0.992±0.001**	0.986±0.003	**0.992±0.002**	**0.992±0.001**
Car_Eval	0.982±0.004	0.978±0.002	0.982±0.004	0.994±0.004	0.988±0.006	0.975±0.007	0.986±0.004	**0.999±0.000**
Cdc	0.863±0.001	0.863±0.002	N/A	0.864±0.001	0.866±0.002	0.865±0.007	0.864±0.001	**0.867±0.001**
Credit_G	0.756±0.004	0.758±0.011	0.756±0.004	0.755±0.011	0.751±0.008	0.754±0.011	0.748±0.018	**0.775±0.002**
Heart	0.915±0.001	0.916±0.008	0.914±0.001	0.920±0.008	0.909±0.002	0.912±0.007	0.911±0.004	**0.923±0.001**
Jungle	0.970±0.000	0.975±0.000	0.970±0.000	0.980±0.000	0.983±0.005	0.972±0.003	0.981±0.006	**0.993±0.000**
Myocardial	0.802±0.003	0.800±0.010	N/A	0.803±0.002	0.805±0.003	0.803±0.003	0.805±0.002	**0.809±0.003**
Pima	0.809±0.007	0.810±0.003	0.805±0.006	0.815±0.008	0.810±0.007	0.810±0.005	0.810±0.008	**0.823±0.003**
Student	0.978±0.001	0.977±0.000	0.978±0.001	0.983±0.000	0.979±0.001	0.978±0.001	0.978±0.001	**0.984±0.000**
Churn	0.829±0.002	0.833±0.003	0.829±0.002	0.828±0.001	0.827±0.003	0.825±0.002	0.829±0.001	**0.835±0.001**
Titanic	0.843±0.007	0.839±0.005	0.843±0.007	0.816±0.005	0.843±0.006	0.847±0.004	0.849±0.004	**0.872±0.008**
Wine	0.878±0.001	0.885±0.005	0.879±0.002	**0.891±0.007**	0.878±0.001	0.869±0.006	0.879±0.003	0.886±0.003
MeanRank	4.37	3.69	4.75	3.12	3.57	4.81	3.75	**1.12**

Table 2: Performance (NRMSE) of all methods on 7 regression datasets. Best results are in bold, second-best are underlined (Lower is better). Results are averaged across three random train-test splits using XGBoost regressor.

Datasets	Base	Traditional Methods			LLM-based Method	MALMAS
		DFS	AutoFeat	OpenFE	LLMFE	
Airfoil	0.015±0.001	0.016±0.001	0.015±0.001	0.014±0.000	0.015±0.001	**0.013±0.000**
Bike	0.230±0.001	0.225±0.003	0.230±0.001	**0.213±0.002**	0.225±0.003	0.215±0.001
Crab	0.220±0.003	0.217±0.002	0.220±0.003	0.214±0.002	0.218±0.002	**0.213±0.002**
Insurance	0.367±0.006	0.365±0.010	0.367±0.006	0.381±0.006	0.358±0.007	**0.355±0.002**
House	0.173±0.005	0.179±0.019	0.173±0.005	0.160±0.001	0.165±0.005	**0.155±0.004**
Energy	0.060±0.002	**0.046±0.003**	0.060±0.002	0.054±0.003	0.058±0.005	0.050±0.007
Medical	0.368±0.003	0.373±0.002	0.368±0.003	0.377±0.001	0.370±0.006	**0.355±0.003**
MeanRank	3.86	3.29	3.86	2.86	3.14	**1.29**

5.1.2 Baselines

We compared MALMAS against a range of automated feature engineering baselines, including traditional methods such as AutoFeat (Horn et al., 2019), OpenFE (Zhang et al., 2023), and DFS (Kanter and Veeramachaneni, 2015), and LLM-based approaches such as CAAFE (Hollmann et al., 2023), OCTree (Nam et al., 2024), and LLMFE (Abhyankar et al., 2025). The configurations of all baseline methods are detailed in Appendix B.1.

5.1.3 Evaluation Metrics

For classification tasks, we adopted the area under the AUC as the primary evaluation metric, and additionally reported accuracy (ACC) as a complementary measure, as shown in Appendix C.1. For regression tasks, we used the normalized root mean squared error (NRMSE) as the evaluation metric. Following prior work (Abhyankar et al., 2025), we also adopted mean rank as a global indicator to compare the overall effectiveness.

5.1.4 MALMAS Configuration

Across all experiments, MALMAS uses a fixed multi-agent configuration with R=4 iterative rounds, where agents generate candidate features, evaluate them. All methods are evaluated with the same downstream model, XGBoost (Chen and Guestrin, 2016). LLM-based results in the main text use DeepSeekV3; additional details are deferred to Appendix B.2.

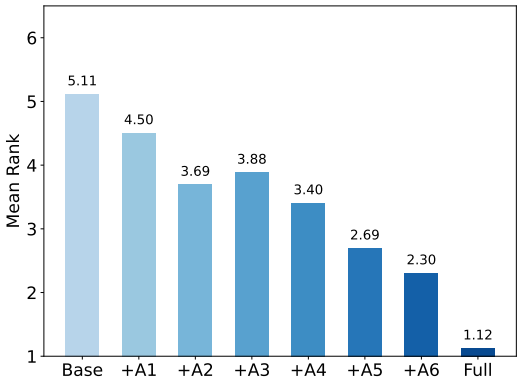

Figure 3: Mean rank (lower is better) across different ablation configurations of MALMAS.

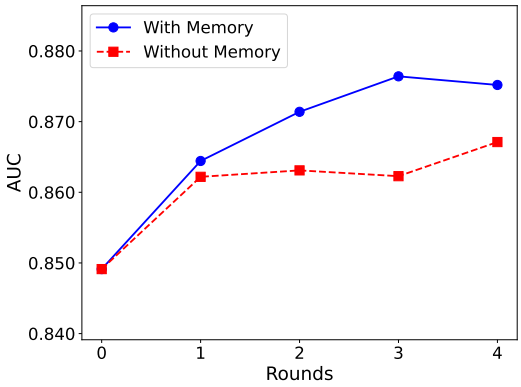

Figure 4: AUC performance on the Adult dataset across different rounds with and without memory.

5.2 Overall Performance

Table 1 summarizes the AUC performance of all evaluated methods across 16 classification datasets. Overall, MALMAS achieves the highest average AUC, consistently outperforming both traditional feature engineering methods and recent LLM-based approaches. MALMAS consistently improves upon the base model, ranking first or second on most benchmark datasets and exhibiting strong generalization across diverse real-world domains. Although LLM-based methods such as OCTree and LLMFE benefit from semantic-aware transformations, they still underperform compared to MALMAS in terms of overall average AUC. These results clearly and collectively underscore the effectiveness of memory-enhanced multi-agent collaboration in facilitating high-quality feature discovery.

As shown in Table 2, MALMAS also achieves the lowest mean NRMSE on almost regression tasks, indicating its strong and reliable feature generation capability beyond classification. Although some LLM-based baselines such as CAAFE and OCTree do not support regression, MALMAS still outperforms LLMFE by a large margin, confirming its advantage in continuous-value prediction.

5.3 Ablation Study

To gain deeper insights into the contributions of the multi-agent and memory modules, we further analyzed the results in Figure 3. From "Base" to "+A6," the mean rank decreases from 5.11 to 2.30, indicating that expanding the agent pool broadens the feature search space and improves diversity, which in turn enhances downstream performance.

Beyond this, the "Full" configuration—which incorporates the memory module on top of all six agents—achieves a dramatic mean rank reduction to 1.12. This demonstrates the role of memory in accumulating cross-round information and refining feature-generation strategies. Specifically, procedural memory records attempted features to reduce redundancy, feedback memory stores downstream performance to guide the next round of exploration, and conceptual memory abstracts cross-round patterns summarized by the Summary Agent into a global conceptual memory shared across agents.

We observe a slight non-monotonicity from +A2 to +A3. This can plausibly occur because adding agents expands the candidate pool but may introduce higher-variance transformations that, under a fixed top-N budget, occasionally replace more robust features; mean-rank aggregation is also sensitive to small dataset-level fluctuations.

5.4 Parameter Sensitivity

The number of generation rounds R is a key parameter in MALMAS, governing iterative feature refinement. Memory facilitates this process by guiding feature reuse, evaluation, and abstraction. Figure 4 reports the AUC on the Adult dataset across rounds. With memory enabled, AUC increases from 0.85 to nearly 0.88 as R grows, indicating that iterative generation can leverage accumulated feedback to uncover richer feature interactions. In contrast, without memory, performance plateaus after the first two rounds and slightly drops at round three, suggesting less directed exploration and limited gains in later rounds. Moreover, improvements diminish and plateau around rounds three to four, implying that MALMAS reaches a sufficiently rich feature set; dynamic scheduling could improve efficiency by using more agents/rounds early for exploration and focusing on high-value features later.

Table 3: Classification performance (AUC) of H2O and DS-Agent on benchmark tabular datasets. "w/o" indicates training with original features, while "w/" indicates training with our derived features. Results are reported as mean ± standard deviation over three runs.

Datasets	H2O		DS-Agent	
	w/o	w/	w/o	w/
Adult	0.876±0.003	**0.881±0.001**	0.871±0.002	**0.880±0.001**
Bank	0.864±0.010	**0.899±0.029**	0.866±0.012	**0.892±0.021**
Breast_W	0.989±0.002	**0.992±0.003**	0.985±0.004	**0.990±0.002**
Churn	0.844±0.003	**0.846±0.001**	**0.846±0.003**	0.845±0.004
Titanic	0.859±0.003	**0.869±0.001**	0.855±0.001	**0.866±0.003**

5.5 Integration with Classical AutoML

To further validate the effectiveness of our derived features in an end-to-end setting, we integrate them into classical AutoML pipelines based on H2O AutoML and DS-Agent (LeDell and Poirier, 2020; Guo et al., 2024). Table 3 reports the AUC performance of both methods on multiple tabular benchmark datasets. For each method, "w/o" uses the original features only, whereas "w/" augments them with our derived features.

In our experiments, H2O AutoML was run with a time budget of 2 hours for each run on each dataset, using five-fold cross-validation and "AUC" as the primary metric for model selection. For the DS-Agent pipeline, we used the same data splits and metric, and adopted DeepSeek-V3 as the LLM backbone to generate derived features.

Across all datasets, incorporating our derived features consistently improves the performance of both H2O AutoML and DS-Agent. This demonstrates that our feature derivation method integrates well with established end-to-end AutoML frameworks, yielding robust and reproducible gains.

5.6 Discussion on Feature Generalization

To assess whether MALMAS-generated features generalize beyond a single downstream model, we evaluated them across multiple classifiers, including XGBoost, LightGBM, Random Forest, and MLP (Chen and Guestrin, 2016; Ke et al., 2017; Liu et al., 2012). Table 8 shows that MALMAS consistently achieves the highest AUC, with a mean rank of 1.00 (vs. 2.40 for the next-best method), indicating stable performance across diverse learning architectures and suggesting that the generated features capture broadly informative patterns rather than being tuned to a specific model. Compared with conventional approaches that may favor a particular algorithm, MALMAS remains effective under different decision boundaries and inductive biases; for example, both tree-based and neural models benefit from the enriched feature space. This cross-model consistency provides a reliable foundation in pipelines where model choice may vary across deployment settings. By mitigating dependence on any single learner, MALMAS can reduce repeated feature engineering and yield a more portable feature base for classification tasks.

5.7 Computational and Token Cost

To assess the feasibility of our method, we measured the average runtime and token usage of MALMAS on 16 classification datasets using the DeepSeek-V3 API as the LLM backbone. On average, each dataset required 0.452 hours of computation and 147.57k tokens for feature generation, with an estimated cost of $0.17, as detailed in Appendix C.6. These results indicate that MALMAS incurs modest overhead and can be readily embedded within existing AutoML pipelines.

6 Conclusion

We propose **MALMAS**, a memory-augmented multi-agent framework for automated feature generation, and validate its effectiveness through extensive experiments. By assigning distinct roles to specialized agents, MALMAS enables parallel and diverse exploration of the feature space, addressing the limitations of single-strategy approaches. Its memory module allows agents to retain useful signals and improve generation strategies across iterations. Together, these components provide a scalable and interpretable approach for producing high-quality, task-relevant features. We further present practical analyses demonstrating real-world applicability and efficiency.

7 Limitations

MALMAS is designed for labeled tabular datasets and relies on downstream evaluation signals; its effectiveness may degrade when labels are scarce or evaluation budgets are limited. While our framework targets tabular feature engineering, its applicability to other modalities or structured domains remains unexplored. Moreover, as the candidate feature pool grows, repeated downstream training and validation can become a computational bottleneck, making performance sensitive to the available evaluation budget. Finally, although MALMAS provides transformation descriptions and memory traces, the overall LLM-driven generation process does not guarantee full interpretability of every derived feature.

8 Ethical Considerations

MALMAS is an automated feature generation framework for tabular data. Its outputs may inherit or amplify biases present in the input data, and the downstream evaluation signal may inadvertently favor transformations that correlate with sensitive attributes when such attributes are present or can be proxied. In addition, because MALMAS generates transformation programs, it may propose invalid or data-leaking features if the schema is ambiguous or the data pipeline is misconfigured. To mitigate these risks, we recommend applying strict schema constraints (e.g., explicitly marking protected attributes and leakage-prone fields), enforcing execution-time validation and leakage checks, and conducting fairness and privacy audits when deploying MALMAS in high-stakes settings. Finally, our experiments use publicly available datasets and do not involve human subjects.

References

Nikhil Abhyankar, Parshin Shojaee, and Chandan K. Reddy. 2025. Llm-fe: Automated feature engineering for tabular data with llms as evolutionary optimizers. *Preprint*, arXiv:2503.14434.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *Preprint*, arXiv:2308.07201.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

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.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Alireza Ghafarollahi and Markus J. Buehler. 2024. Sciagents: Automating scientific discovery through multi-agent intelligent graph reasoning. *Preprint*, arXiv:2409.05556.

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. *Preprint*, arXiv:2411.10478.

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. In *Proceedings of the 41st International Conference on Machine Learning*, pages 16813–16848.

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

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

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

Daniel P. Jeong, Zachary C. Lipton, and Pradeep Ravikumar. 2025. Llm-select: Feature selection with large language models. *Preprint*, arXiv:2407.02694.

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.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on*

Neural Information Processing Systems 2017, pages 3146–3154.

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

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024a. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9.

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. 2024b. Improving multi-agent debate with sparse communication topology. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7281–7294.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904.

Tongxuan Liu, Xingyu Wang, Weizhe Huang, Wenjiang Xu, Yuting Zeng, Lei Jiang, Hailong Yang, and Jing Li. 2024. Groupdebate: Enhancing the efficiency of multi-agent debate using group discussion. *Preprint*, arXiv:2409.14051.

Yanli Liu, Yourong Wang, and Jian Zhang. 2012. New machine learning algorithm: Random forest. In *Information Computing and Applications - Third International Conference, ICICA 2012*, volume 7473, pages 246–252.

Andrea Matarazzo and Riccardo Torlone. 2025. A survey on large language models with some insights on their capabilities and limitations. *Preprint*, arXiv:2501.04040.

Jaehyun Nam, Kyuyoung Kim, Seunghyuk Oh, Jihoon Tack, Jaehyung Kim, and Jinwoo Shin. 2024. Optimized feature generation for tabular data via llms with decision tree reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pages 92352–92380. Curran Associates, Inc.

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.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, Chen Gao, Fengli Xu, Fang Zhang, Ke Rong, Jun Su, and Yong Li. 2025. Agentsociety: Large-scale simulation of llm-driven generative agents advances understanding of human behaviors and society. *Preprint*, arXiv:2502.08691.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

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.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. 2025. Automl-agent: A multi-agent llm framework for full-pipeline automl. *Preprint*, arXiv:2410.02958.

Chi Wang, Qingyun Wu, Markus Weimer, and Erkang Zhu. 2021. Flaml: A fast and lightweight automl library. In *Proceedings of Machine Learning and Systems*, volume 3, pages 434–447.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Front. Comput. Sci.*, 18(6).

Lanning Wei, Huan Zhao, Xiaohan Zheng, Zhiqiang He, and Quanming Yao. 2024. A versatile graph learning approach through llm-based agent. *Preprint*, arXiv:2309.04565.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Tianping Zhang, Zheyu Aqa Zhang, Zhiyuan Fan, Haoyan Luo, Fengyuan Liu, Qian Liu, Wei Cao, and Li Jian. 2023. Openfe: Automated feature generation

with expert-level performance. In *International Conference on Machine Learning*, pages 41880–41901.

Yanlin Zhang, Ning Li, Quan Gan, Weinan Zhang, David Wipf, and Minjie Wang. 2024. Elf-gym: Evaluating large language models generated features for tabular prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 5420–5424.

A Dataset Specifications

This section provides detailed specifications of all datasets used in our experiments, including the number of features, sample sizes, data sources, and official names. The datasets are categorized into classification and regression tasks as summarized in Table 4.

To ensure a consistent and model-compatible feature representation, missing values in categorical features were imputed with a placeholder category "NA", while missing or infinite values in numerical features were replaced with zeros. All categorical variables were then encoded using `LabelEncoder`, mapping unseen categories during transformation to a fallback code of -1. This preprocessing ensured that all features were numerical and suitable for downstream learning algorithms.

B Implementation Details

B.1 Baseline Configurations

We implement and evaluate a variety of feature engineering baselines, spanning traditional symbolic approaches and recent LLM-based methods, to compare against our proposed MALMAS framework. All methods share the same downstream pipeline, using unified preprocessing and XGBoost as the default model to ensure fairness. Below, we summarize the baseline configurations.

AutoFeat

AutoFeat is a symbolic feature engineering method that constructs new features using mathematical transformations such as polynomials, logarithms, and interactions. We adopt the open-source autofeat package and configure it to perform a single transformation step.

OpenFE

OpenFE is an automated feature construction framework that combines feature boosting and pruning to identify informative transformations. We use the open-source openfe package with default settings.

Deep Feature Synthesis (DFS)

DFS generates new features by applying aggregation and transformation operations over feature primitives. Following standard practice, we use mean, standard deviation as aggregators, and add_numeric, subtract_numeric as transformation primitives.

CAAFE

CAAFE employs large language models to generate features via iterative sampling guided by control instructions. We use the official implementation with the number of iterations set to 10.

OCTree

OCTree performs evolutionary search over operator trees to generate feature transformations. We use the official implementation for classification tasks and set the iteration count to 20.

LLMFE

LLMFE uses a single-round prompt-based LLM generation pipeline without iterative feedback. We configure the method to sample 20 candidate features per seed.

All LLM-based methods are implemented using the DeepSeek API with a temperature of 1.0.

B.2 Implementation Details of MALMAS

MALMAS is a multi-agent, memory-augmented LLM framework for iterative feature generation. Unless otherwise specified, we use DeepSeekV3 as the backbone with temperature 1.0. We run R=4 interaction rounds, select the top-3 features per round, and apply a minimum effective-feature threshold of 2 during conceptual summarization. For all experiments, we use XGBoost as the downstream classifier with 500 trees and a learning rate of 0.02; for the relatively simple Car Evaluation and Banknote Authentication datasets, we reduce the number of trees to 50 to mitigate overfitting. This configuration is kept consistent across all baseline methods. We also evaluate GPT-4.1-mini on 16 classification datasets, with full results reported in Appendix C.2. All experiments are conducted on a machine with an Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz and an NVIDIA RTX A6000.

C Additional Results

C.1 Supplementary Accuracy Results

While AUC was used as the primary evaluation metric in our main experiments due to its robust-

Table 4: Dataset Details and Sources

Datasets	Name	#Features	#Samples	Source
Classification Datasets				
Adult	Adult Census Income	15	32561	Kaggle
Balance	Balance Scale	5	625	Kaggle
Bank	Bank Marketing	21	41188	Kaggle
Banknote	Banknote Authentication	5	1372	UCI
Breast_W	Breast Cancer Wisconsin (Original)	9	699	UCI
Car_Eval	Car Evaluation	7	1209	Kaggle
Cdc	diabetes health indicators dataset	21	253680	Kaggle
Credit_G	German Credit Data	21	1000	Kaggle
Heart	Heart Disease	12	918	Kaggle
Jungle	Jungle Chess 2 Moves	7	44819	OpenML
Myocardial	myocardial infarction complications	111	1700	UCI
Pima	Pima Indians Diabetes	9	768	Kaggle
Student	Student Performance Factors	20	6607	Kaggle
Churn	Telco Customer Churn	21	7043	Kaggle
Titanic	Titanic Dataset	12	891	Kaggle
Wine	Wine Quality	13	6497	Kaggle
Regression Datasets				
Airfoile	Airfoil Self-Noise	7	1504	UCI
Bike	Bike Sharing	13	17379	UCI
Crab	Crab Age Prediction	9	3893	Kaggle
Insurance	Healthcare Insurance	7	1338	Kaggle
Housee	House Price Prediction	36	1460	Kaggle
Energy	Energy Efficiency	9	768	UCI
Medical	Medical Cost Personal Datasets	7	1338	Kaggle

ness against class imbalance—common in many real-world classification datasets—we additionally report results based on Accuracy (ACC) to further validate the effectiveness and generalizability of the MALMAS framework.

Table 5 reports the average ACC, with the following key observations:

- MALMAS achieves the highest average accuracy, outperforming both traditional and LLM-based baselines.
- It ranks first on 11 out of 16 datasets and achieves top-2 performance on 4 datasets, demonstrating strong overall robustness.
- Compared to other LLM-based methods, MALMAS benefits from multi-agent collaboration and memory-guided prompt evolution, resulting in more diverse and relevant feature generation.

These results confirm that MALMAS delivers consistent classification performance not only under class-imbalance-aware metrics like AUC, but also under general-purpose metrics.

C.2 Performance Analysis Using GPT-4.1-Mini

To validate the generalizability of our framework beyond a specific backbone, we also conducted experiments using GPT-4.1 Mini as the LLM for all LLM-based methods. Table 6 reports the AUC performance across 16 classification datasets.

Overall, MALMAS achieves the best average rank (1.37), outperforming both traditional baselines (DFS, AutoFeat, OpenFE) and other LLM-based methods (CAAFE, OCTree, LLMFE). While the performance gap between different LLM-based methods narrows under a weaker backbone, MALMAS still maintains a consistent lead. This suggests that our multi-agent collaboration and mem-

Table 5: Performance (ACC) of all methods on 16 classification datasets. Best results are in bold, second-best are underlined. Results are averaged across three random train-test splits using the XGBoost classifier. "N/A" indicates that the running time exceeded 12 hours.

Datasets	Base	Traditional Methods			LLM-based Methods			MALMAS
		DFS	AutoFeat	OpenFE	CAAFE	OCTree	LLMFE	
Adult	0.814±0.015	0.810±0.005	0.814±0.015	0.814±0.015	**0.822±0.007**	0.803±0.014	0.816±0.013	**0.822±0.013**
Balance	0.849±0.008	0.961±0.004	0.849±0.008	0.849±0.008	**1.000±0.000**	0.868±0.020	0.987±0.019	**1.000±0.000**
Bank	0.873±0.009	0.887±0.008	0.873±0.009	0.884±0.001	0.869±0.010	0.881±0.019	0.868±0.009	**0.897±0.007**
Banknote	0.978±0.003	0.978±0.005	0.978±0.001	0.991±0.004	0.978±0.000	0.973±0.005	0.983±0.003	**0.993±0.006**
Breast_W	0.952±0.009	0.954±0.005	0.952±0.009	0.956±0.007	**0.967±0.004**	0.956±0.002	0.958±0.004	0.960±0.003
Car_Eval	0.901±0.003	0.917±0.006	0.901±0.003	0.961±0.012	0.924±0.019	0.899±0.003	0.916±0.018	**0.989±0.003**
Cdc	0.857±0.002	0.857±0.001	N/A	**0.862±0.001**	0.859±0.001	0.855±0.002	0.861±0.001	0.861±0.001
Credit_G	0.744±0.016	0.746±0.012	0.744±0.016	0.738±0.005	**0.757±0.006**	0.741±0.018	0.748±0.024	0.747±0.010
Heart	0.843±0.017	0.841±0.005	0.843±0.010	0.858±0.003	0.831±0.013	0.852±0.010	0.841±0.003	**0.864±0.004**
Jungle	0.855±0.000	0.868±0.002	0.855±0.000	0.882±0.003	0.901±0.022	0.866±0.017	0.895±0.024	**0.948±0.001**
Myocardial	0.790±0.006	0.791±0.010	N/A	0.792±0.002	0.794±0.003	0.795±0.003	0.795±0.002	**0.799±0.003**
Pima	0.736±0.011	0.753±0.008	0.740±0.005	0.756±0.007	0.738±0.013	0.754±0.018	0.747±0.011	**0.759±0.009**
Student	0.930±0.003	0.929±0.004	0.930±0.003	0.939±0.002	0.934±0.002	0.930±0.003	0.931±0.003	**0.943±0.002**
Churn	0.787±0.001	**0.792±0.002**	0.787±0.001	0.787±0.003	0.784±0.003	0.789±0.002	0.787±0.002	0.791±0.004
Titanic	0.768±0.034	0.782±0.002	0.768±0.034	0.709±0.051	0.774±0.021	0.777±0.036	0.781±0.019	**0.816±0.007**
Wine	0.860±0.001	0.868±0.007	0.861±0.002	**0.872±0.006**	0.864±0.005	0.854±0.005	0.863±0.001	0.869±0.002
MeanRank	5.32	3.81	5.31	3.31	3.44	4.63	3.57	**1.38**

ory mechanisms provide robust benefits even when the underlying LLM capacity is limited.

However, the overall performance of LLM-based methods, including MALMAS, tends to degrade slightly compared to results under stronger LLMs such as DeepSeekV3. For instance, methods like LLMFE and OCTree show more pronounced fluctuations and fall behind traditional methods on some datasets. This observation highlights an important insight: LLM-based feature generation is partially constrained by the expressive and reasoning capabilities of the underlying language model. Therefore, stronger LLMs contribute positively to semantic feature transformation, but architectural design remains critical for consistent gains.

These results serve as complementary evidence to our main experiments in the paper, demonstrating that MALMAS is not only effective with powerful LLMs, but also remains competitive and stable under smaller LLM configurations.

C.3 Details of the Ablation Study

The ablation study results, as shown in Table 7, provide a comprehensive evaluation of the incremental improvements in feature generation performance achieved by progressively adding agents and the memory module in the MALMAS.

Overall Performance Improvement: Starting from the "Bas" configuration, where the model is trained solely on raw features, the mean rank is 5.11. This serves as the baseline, highlighting the limitations of using untransformed raw features without any feature generation strategy. The introduction of each agent role (from +A1 to +A6) brings about noticeable improvements in performance, demonstrating the positive impact of specialized feature generation strategies.

For example, when the first agent (+A1) is introduced, the mean rank improves to 4.50, but further improvements are not always linear. As additional agents are incorporated, the performance fluctuates, reaching a mean rank of 2.30 with the inclusion of all six agents (+A6). This trend suggests that the incorporation of agents with different roles progressively improves feature generation, as reflected by the decreasing mean rank at each step.

Impact of Memory Module: The most significant improvement is observed when the full memory module is incorporated (Full configuration), resulting in the best performance with a mean rank of 1.12. The inclusion of the memory module, which integrates procedural memory, feedback memory, and conceptual memory, enables the system to iteratively refine feature generation strategies based on past experiences. This feedback loop allows the agents to adapt and enhance their strategies, contributing to the significant performance boost from +A6 to the Full configuration.

These results clearly demonstrate the effectiveness of the multi-agent and memory-augmented design of MALMAS. The gradual addition of agents and the final memory module significantly en-

Table 6: Performance comparison (AUC) of all methods across 16 classification datasets. Best results are in bold, second-best are underlined. Results are averaged across three random train-test splits. The GPT-4.1 Mini model was used as the backbone for the LLM-based methods. "N/A" indicates that the running time exceeded 12 hours.

Datasets	Base	Traditional Methods			LLM-based Methods			MALMAS
		DFS	AutoFeat	OpenFE	CAAFE	OCTree	LLMFE	
Adult	0.849±0.009	0.857±0.001	0.849±0.009	0.849±0.009	**0.868±0.005**	0.845±0.011	0.853±0.011	0.860±0.013
Balance	0.908±0.009	0.989±0.009	0.908±0.009	0.908±0.009	**1.000±0.000**	0.989±0.007	0.994±0.008	**1.000±0.000**
Bank	0.869±0.007	0.891±0.014	0.869±0.007	**0.904±0.003**	0.873±0.009	0.869±0.007	0.867±0.010	0.885±0.011
Banknote	0.995±0.002	0.998±0.001	0.994±0.003	0.998±0.001	0.993±0.002	0.992±0.004	0.992±0.005	**0.999±0.001**
Breast_W	0.989±0.002	0.988±0.003	0.989±0.002	0.988±0.003	**0.991±0.001**	0.989±0.002	**0.991±0.002**	**0.991±0.001**
Car	0.982±0.004	0.978±0.002	0.982±0.004	0.994±0.004	0.993±0.002	0.983±0.007	0.991±0.001	**0.999±0.000**
Cdc	0.863±0.001	0.863±0.002	N/A	0.864±0.001	0.865±0.001	**0.866±0.002**	0.864±0.001	**0.866±0.001**
Credit_G	0.756±0.004	0.758±0.011	0.756±0.004	0.755±0.011	0.760±0.009	0.754±0.011	0.756±0.003	**0.764±0.004**
Heart	0.915±0.001	0.916±0.008	0.914±0.001	0.920±0.008	0.912±0.004	0.912±0.003	0.913±0.004	**0.923±0.003**
Jungle	0.970±0.000	0.975±0.000	0.970±0.000	0.980±0.000	0.983±0.005	0.974±0.002	0.981±0.006	**0.988±0.003**
Myocardial	0.802±0.003	0.800±0.010	N/A	0.803±0.002	0.805±0.002	0.802±0.003	0.806±0.001	**0.808±0.003**
Pima	0.809±0.007	0.810±0.003	0.805±0.006	0.815±0.008	0.801±0.007	0.809±0.007	0.813±0.011	**0.817±0.007**
Student	0.978±0.001	0.977±0.000	0.978±0.001	**0.983±0.000**	0.980±0.001	0.978±0.001	0.978±0.004	0.982±0.001
Churn	0.829±0.002	0.833±0.003	0.829±0.002	0.828±0.001	0.830±0.002	0.827±0.002	0.830±0.003	**0.834±0.002**
Titanic	0.843±0.007	0.839±0.005	0.843±0.007	0.816±0.005	0.849±0.011	0.843±0.007	0.846±0.010	**0.855±0.002**
Wine	0.878±0.001	0.885±0.005	0.879±0.002	**0.891±0.007**	0.878±0.005	0.878±0.001	0.879±0.001	0.882±0.002
MeanRank	4.94	4.32	5.12	3.81	3.44	5.12	3.87	**1.37**

hances the performance of model by enabling a more diverse and refined feature generation process. The final Full configuration, which integrates all components, shows the highest performance across all datasets, reaffirming the importance of both the multi-agent collaboration and the memory mechanism in driving high-quality feature discovery. The steady decrease in mean rank as each agent and memory component is added suggests that the MALMAS framework provides a robust and adaptive solution for feature generation.

C.4 Details of the Parameter Sensitivity

Figure 5 presents the AUC scores across four feature generation rounds for multiple datasets, comparing models *with* and *without* memory. Each round corresponds to an additional iteration of feature generation and refinement by the MALMAS framework. Key findings from this evaluation are summarized below:

Impact of Memory on Performance

Models *with memory* consistently outperform their *without memory* counterparts across all rounds and most datasets. This demonstrates the effectiveness of memory-augmented feature generation in enabling progressive refinement. In contrast, models without memory show limited or stagnant improvement, particularly in early rounds, due to the lack of retained knowledge from prior iterations.

Performance Trends Across Rounds

In datasets such as *Adult*, *Breast_W*, and *Jungle*, AUC scores steadily increase across rounds when memory is used, highlighting the cumulative benefits of iterative refinement. Without memory, the improvements are slower or negligible, reflecting the difficulty of building upon previously learned features without retention mechanisms.

Dataset-Specific Behavior

On simpler datasets like *Banknote*, both configurations achieve high AUC scores with a small performance gap. This suggests that memory may be less critical for less complex tasks. In contrast, for challenging datasets such as *Churn* and *Credit_G*, the performance gap becomes more pronounced, indicating that memory is especially valuable for learning from intricate or nuanced data patterns.

Overall Implications

The *with memory* models exhibit a clear trend of continuous improvement across rounds, whereas *without memory* models often plateau. This reflects the advantage of a memory-augmented framework in progressively enriching the feature space. The ability to incorporate and refine past knowledge plays a pivotal role in enhancing model performance, especially in complex scenarios.

In summary, these results provide strong empirical evidence for the benefits of memory in iterative feature generation. The memory-enhanced design

Table 7: Performance comparison across different steps in the ablation study for each dataset using DeepSeekv3.

Dataset	Base	+A1	+A2	+A3	+A4	+A5	+A6	Full
Adult	0.849±0.009	0.853±0.110	0.858±0.009	0.867±0.012	0.861±0.015	0.865±0.005	0.867±0.020	0.875±0.010
Balance	0.908±0.009	0.908±0.009	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000
Bank	0.869±0.007	0.878±0.023	0.878±0.006	0.878±0.006	0.888±0.011	0.881±0.004	0.884±0.011	0.895±0.002
Banknote	0.995±0.002	0.992±0.003	0.995±0.004	0.997±0.003	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.001
Breast_W	0.989±0.002	0.989±0.002	0.990±0.001	0.989±0.003	0.989±0.003	0.990±0.003	0.992±0.003	0.992±0.001
Car_Eval	0.982±0.004	0.991±0.001	0.996±0.001	0.998±0.001	0.995±0.002	0.997±0.002	0.999±0.001	0.999±0.000
Cdc	0.863±0.001	0.864±0.001	0.865±0.001	0.865±0.002	0.865±0.001	0.865±0.001	0.866±0.001	0.867±0.001
Credit_G	0.756±0.004	0.756±0.001	0.752±0.008	0.750±0.013	0.757±0.013	0.759±0.009	0.765±0.010	0.775±0.002
Heart	0.915±0.001	0.915±0.001	0.910±0.002	0.908±0.010	0.911±0.010	0.913±0.008	0.907±0.008	0.923±0.001
Jungle	0.970±0.000	0.972±0.001	0.993±0.003	0.978±0.012	0.980±0.010	0.987±0.007	0.986±0.005	0.993±0.000
Myocardial	0.802±0.003	0.802±0.003	0.804±0.001	0.804±0.001	0.804±0.002	0.806±0.002	0.806±0.001	0.809±0.003
Pima	0.809±0.007	0.809±0.011	0.811±0.013	0.815±0.009	0.817±0.007	0.821±0.005	0.824±0.005	0.823±0.003
Student	0.978±0.001	0.979±0.001	0.980±0.001	0.980±0.001	0.978±0.001	0.980±0.004	0.982±0.000	0.984±0.000
Churn	0.829±0.002	0.833±0.001	0.833±0.001	0.831±0.003	0.825±0.009	0.833±0.001	0.828±0.007	0.835±0.001
Titanic	0.843±0.007	0.856±0.016	0.859±0.015	0.852±0.006	0.865±0.008	0.861±0.007	0.868±0.012	0.872±0.008
Wine	0.878±0.001	0.879±0.001	0.881±0.002	0.881±0.001	0.882±0.004	0.881±0.004	0.883±0.003	0.886±0.003
MeanRank	5.11	4.50	3.69	3.88	3.40	2.69	2.30	1.12

of MALMAS significantly boosts learning capacity and generalization, enabling sustained performance gains over multiple rounds. This underscores the critical role of memory in advanced feature engineering systems.

C.5 Generalization Experiment Setup

For the generalization experiments across downstream models, all tree-based classifiers, including XGBoost, LightGBM, CatBoost, and Random Forest, were configured with 500 trees and a learning rate of 0.02, where applicable. Other hyperparameters were kept at their default settings as provided by each library. For two relatively simple datasets, car_evaluation and banknote_authentication, the number of trees was reduced to 50 to prevent overfitting due to their low data complexity. This configuration was consistently applied to all classification experiments to ensure fair evaluation of feature transferability across different model architectures.

Across all downstream classifiers—Logistic Regression (Table 9), LightGBM (Table 10), Random Forest (Table 11) and MLP(Table 12) MALMAS consistently achieves the highest average AUC and the lowest mean rank. This indicates that the features generated by our multi-agent framework generalize robustly across diverse model architectures. Notably, while traditional methods such as OpenFE and AutoFeat perform competitively on simpler datasets, they fail to match MALMAS on complex ones. LLM-based baselines, including OCTree and LLMFE, benefit from semantic reasoning but still fall short in overall performance. These results confirm that the proposed framework is model-agnostic and maintains strong discriminative capability regardless of the downstream classifier.

C.6 Additional Results on Time and Token Usage

To evaluate the computational efficiency of our LLM-based feature generation process, we measured the time cost (in hours) and token usage (in thousands) on all 16 classification datasets. The experiments were conducted using the DeepSeek-V3 API as the LLM backbone and XGBoost as the downstream model, with all hyperparameters kept consistent with the main experiments. As shown in Table 13, the average generation time per dataset was approximately 0.452 hours, and the average token usage was around 147.57k tokens. These results demonstrate that our method is both time-efficient and computationally affordable, especially considering that feature generation is a one-time offline process. The variation in cost across datasets primarily reflects differences in metadata length and data complexity, but remains within acceptable bounds for real-world AutoML scenarios.

Table 8: AUC performance of all methods across four classifiers (Logistic Regression, XGBoost, LightGBM, Random Forest and MLP), averaged over 16 classification datasets. Best results are in bold; second-best are underlined.

Model	Base	Traditional Methods			LLM-based Methods			MALMAS
		DFS	AutoFeat	OpenFE	CAAFE	OCTree	LLMFE	
Logistic Regression	0.814	0.808	0.806	0.823	0.819	0.815	0.814	**0.845**
XGBoost	0.890	0.898	0.889	0.8903	0.898	0.890	0.896	**0.908**
LightGBM	0.891	0.898	0.891	0.896	0.899	0.891	0.898	**0.906**
Random Forest	0.887	0.892	0.889	0.889	0.900	0.889	0.900	**0.905**
MLP	0.863	0.867	0.862	0.866	0.869	0.861	0.870	**0.874**
MeanRank	5.20	3.60	5.80	3.80	2.40	5.20	3.00	**1.00**

Table 9: Performance (AUC) of all methods on 16 classification datasets. Best results are in bold, second-best are underlined. Results are averaged across three random train-test splits using the Logistic Regression classifier. "N/A" indicates that the running time exceeded 12 hours.

Datasets	Base	Traditional Methods			LLM-based Methods			MALMAS
		DFS	AutoFeat	OpenFE	CAAFE	OCTree	LLMFE	
Adult	0.777±0.021	0.782±0.020	0.801±0.011	0.791±0.022	0.812±0.011	0.801±0.003	0.778±0.022	**0.813±0.011**
Balance	0.863±0.018	0.873±0.028	0.896±0.018	0.867±0.018	0.999±0.002	0.877±0.023	0.987±0.016	**1.000±0.000**
Bank	0.890±0.016	0.896±0.008	0.896±0.014	0.893±0.010	0.886±0.010	0.897±0.002	0.887±0.008	**0.901±0.003**
Banknote	0.993±0.001	0.993±0.001	0.998±0.002	**0.999±0.001**	0.995±0.002	0.990±0.005	0.998±0.002	**0.999±0.001**
Breast_W	0.963±0.002	0.963±0.002	0.963±0.002	0.940±0.012	0.963±0.002	0.960±0.004	0.962±0.002	**0.971±0.002**
Car_Eval	0.689±0.008	0.688±0.005	0.689±0.008	0.797±0.003	0.716±0.015	0.693±0.014	0.697±0.016	**0.911±0.054**
Cdc	0.790±0.002	0.794±0.001	N/A	0.792±0.001	0.793±0.001	0.796±0.001	**0.799±0.001**	**0.799±0.001**
Credit_G	0.723±0.009	0.716±0.005	0.724±0.013	0.719±0.006	0.722±0.014	0.716±0.005	0.712±0.004	**0.729±0.002**
Heart	0.849±0.007	0.848±0.006	0.708±0.007	0.851±0.004	0.831±0.028	0.836±0.001	0.853±0.010	**0.857±0.004**
Jungle	0.678±0.003	0.678±0.003	0.702±0.002	0.709±0.013	0.681±0.003	0.678±0.003	0.685±0.019	**0.710±0.002**
Myocardial	0.764±0.001	0.765±0.010	N/A	0.766±0.002	0.766±0.002	**0.767±0.003**	0.766±0.001	0.766±0.003
Pima	0.779±0.016	0.719±0.029	0.751±0.037	0.751±0.015	0.728±0.074	0.765±0.009	0.713±0.054	**0.781±0.005**
Student	0.864±0.012	0.829±0.004	0.861±0.015	**0.936±0.013**	0.871±0.027	0.864±0.012	0.852±0.005	0.879±0.004
Churn	0.802±0.003	0.800±0.004	0.795±0.009	0.799±0.005	0.791±0.005	0.799±0.004	0.796±0.004	**0.806±0.002**
Titanic	0.785±0.001	0.782±0.009	0.748±0.005	0.741±0.009	0.754±0.039	**0.791±0.010**	0.743±0.005	**0.791±0.022**
Wine	0.816±0.008	0.810±0.007	0.811±0.004	0.815±0.005	0.812±0.006	0.815±0.006	0.812±0.002	**0.820±0.006**
Mean	0.814	0.808	0.806	0.823	0.819	0.815	0.814	**0.845**
MeanRank	4.25	4.75	4.43	3.68	4.37	3.81	4.31	**1.12**

Table 10: Performance (AUC) of all methods on 16 classification datasets. Best results are in bold, second-best are underlined. Results are averaged across three random train-test splits using the LightGBM classifier. "N/A" indicates that the running time exceeded 12 hours.

Datasets	Base	Traditional Methods			LLM-based Methods			MALMAS
		DFS	AutoFeat	OpenFE	CAAFE	OCTree	LLMFE	
Adult	0.852±0.012	0.853±0.006	0.852±0.012	0.852±0.012	**0.863±0.013**	0.840±0.006	0.853±0.014	0.854±0.024
Balance	0.917±0.008	0.990±0.006	0.918±0.008	0.917±0.008	**1.000±0.000**	0.937±0.043	0.993±0.009	**1.000±0.000**
Bank	0.878±0.008	0.896±0.013	0.879±0.009	0.900±0.006	0.882±0.019	0.882±0.007	0.870±0.011	**0.902±0.007**
Banknote	0.995±0.001	0.998±0.002	0.996±0.000	**0.999±0.001**	0.995±0.001	0.995±0.001	0.994±0.002	**0.999±0.001**
Breast_W	0.990±0.002	0.989±0.002	0.989±0.002	0.990±0.003	**0.993±0.001**	0.988±0.004	**0.993±0.001**	0.992±0.001
Car_Eval	0.981±0.003	0.989±0.002	0.981±0.003	0.997±0.001	0.989±0.004	0.980±0.001	0.986±0.004	**0.999±0.000**
Cdc	0.858±0.002	0.861±0.002	N/A	0.860±0.001	0.863±0.001	0.862±0.001	**0.864±0.001**	**0.864±0.001**
Credit_G	0.750±0.006	0.749±0.012	0.751±0.007	0.762±0.019	0.755±0.005	0.747±0.005	0.744±0.004	**0.767±0.014**
Heart	0.917±0.003	0.917±0.008	0.917±0.004	**0.922±0.008**	0.912±0.003	0.919±0.009	0.914±0.002	0.920±0.003
Jungle	0.974±0.000	0.979±0.000	0.974±0.000	0.984±0.000	0.988±0.005	0.977±0.005	0.987±0.006	**0.996±0.000**
Myocardial	0.799±0.001	0.795±0.010	N/A	0.799±0.002	0.803±0.002	0.800±0.003	0.804±0.001	**0.805±0.003**
Pima	0.805±0.001	0.813±0.005	0.805±0.003	0.814±0.008	0.804±0.009	0.808±0.010	0.813±0.001	**0.818±0.005**
Student	0.981±0.001	0.979±0.000	0.981±0.001	0.984±0.000	0.981±0.001	0.981±0.001	0.981±0.000	**0.986±0.000**
Churn	0.828±0.002	0.828±0.001	0.827±0.001	0.827±0.002	0.824±0.004	0.825±0.002	0.827±0.000	**0.828±0.002**
Titanic	0.842±0.011	0.842±0.005	0.842±0.011	0.843±0.009	0.848±0.004	0.849±0.001	0.854±0.006	**0.876±0.012**
Wine	0.886±0.000	0.890±0.005	0.886±0.000	**0.892±0.006**	0.885±0.003	0.874±0.009	0.885±0.004	0.886±0.003
Mean	0.891	0.898	0.891	0.896	0.899	0.891	0.898	**0.906**
MeanRank	4.56	3.69	4.63	2.94	3.19	4.31	3.43	**1.32**

Table 11: Performance (AUC) of all methods on 16 classification datasets. Best results are in bold, second-best are underlined. Results are averaged across three random train-test splits using the RandomForest classifier. "N/A" indicates that the running time exceeded 12 hours.

Datasets	Base	Traditional Methods			LLM-based Methods			MALMAS
		DFS	AutoFeat	OpenFE	CAAFE	OCTree	LLMFE	
Adult	0.867±0.007	0.851±0.005	0.866±0.005	0.866±0.008	**0.870±0.003**	0.861±0.005	0.869±0.010	**0.870±0.011**
Balance	0.829±0.007	0.972±0.014	0.843±0.002	0.837±0.007	**1.000±0.000**	0.871±0.047	0.982±0.025	**1.000±0.000**
Bank	0.891±0.006	0.901±0.011	0.891±0.005	**0.908±0.007**	0.898±0.005	0.885±0.012	0.880±0.007	0.906±0.007
Banknote	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**
Breast_W	0.989±0.003	0.988±0.003	0.986±0.003	0.989±0.002	0.990±0.001	0.987±0.003	**0.991±0.003**	**0.991±0.000**
Car_Eval	0.993±0.001	0.989±0.002	0.993±0.001	0.997±0.001	0.992±0.004	0.990±0.003	0.996±0.001	**0.999±0.000**
Cdc	0.832±0.002	0.834±0.002	N/A	0.834±0.002	0.836±0.001	0.830±0.001	**0.837±0.001**	**0.837±0.001**
Credit_G	0.768±0.007	0.759±0.005	0.759±0.008	0.767±0.013	0.758±0.019	0.755±0.015	0.765±0.005	**0.769±0.004**
Heart	0.919±0.006	0.916±0.005	**0.923±0.006**	0.921±0.006	0.917±0.002	**0.923±0.007**	0.916±0.002	0.919±0.004
Jungle	0.943±0.000	0.955±0.000	0.949±0.000	0.963±0.001	0.979±0.009	0.954±0.016	0.972±0.010	**0.993±0.000**
Myocardial	0.801±0.001	0.799±0.010	N/A	0.803±0.002	0.803±0.001	0.802±0.002	0.804±0.002	**0.805±0.001**
Pima	0.824±0.008	0.823±0.011	0.827±0.006	**0.829±0.006**	0.825±0.002	0.826±0.008	0.826±0.016	**0.829±0.005**
Student	0.965±0.001	0.952±0.001	0.964±0.002	0.961±0.001	0.962±0.004	0.965±0.001	0.961±0.001	**0.969±0.000**
Churn	0.819±0.001	0.810±0.002	**0.821±0.001**	0.818±0.000	0.810±0.004	0.819±0.001	**0.821±0.001**	**0.821±0.002**
Titanic	0.853±0.011	0.831±0.013	0.862±0.004	0.821±0.019	0.868±0.019	0.860±0.006	0.872±0.014	**0.879±0.022**
Wine	0.899±0.006	0.897±0.005	0.899±0.006	0.893±0.006	**0.900±0.006**	0.888±0.013	0.896±0.006	0.893±0.006
Mean	0.887	0.892	0.889	0.889	0.900	0.889	0.900	**0.905**
MeanRank	3.88	4.62	3.94	3.38	2.94	4.25	2.87	**1.44**

Table 12: Performance (AUC) of all methods on 16 classification datasets. Best results are in bold, second-best are underlined. Results are averaged across three random train-test splits using the MLP classifier. "N/A" indicates that the running time exceeded 12 hours.

Datasets	Base	Traditional Methods			LLM-based Methods			MALMAS
		DFS	AutoFeat	OpenFE	CAAFE	OCTree	LLMFE	
Adult	0.834±0.007	0.829±0.007	0.836±0.006	0.835±0.008	0.843±0.007	0.828±0.006	0.833±0.012	**0.845±0.005**
Balance	0.898±0.008	0.952±0.007	0.896±0.006	0.897±0.006	**0.960±0.000**	0.907±0.024	0.954±0.009	**0.960±0.000**
Bank	0.860±0.011	0.869±0.013	0.862±0.010	**0.876±0.010**	0.862±0.015	0.850±0.008	0.861±0.012	0.875±0.005
Banknote	0.960±0.000	0.972±0.000	0.965±0.000	**0.984±0.000**	0.980±0.000	0.967±0.000	0.980±0.000	**0.984±0.000**
Breast_W	0.950±0.002	0.950±0.002	0.950±0.001	0.950±0.001	**0.953±0.002**	0.948±0.004	0.952±0.003	0.952±0.002
Car_Eval	0.958±0.000	0.954±0.001	0.958±0.000	0.959±0.000	0.958±0.000	0.956±0.002	0.957±0.000	**0.960±0.000**
Cdc	0.818±0.001	0.821±0.002	N/A	0.820±0.002	0.823±0.001	0.822±0.001	**0.824±0.001**	**0.824±0.001**
Credit_G	0.737±0.009	0.738±0.007	0.733±0.005	0.738±0.009	0.735±0.005	0.728±0.004	0.737±0.003	**0.743±0.004**
Heart	**0.890±0.008**	0.886±0.008	**0.890±0.008**	**0.890±0.008**	0.888±0.009	0.887±0.012	0.889±0.009	0.888±0.007
Jungle	0.934±0.000	0.937±0.001	0.934±0.000	0.941±0.001	0.949±0.007	0.939±0.007	0.949±0.006	**0.958±0.000**
Myocardial	0.759±0.004	0.755±0.006	N/A	0.759±0.002	0.763±0.002	0.760±0.004	0.764±0.001	**0.766±0.003**
Pima	0.792±0.006	0.790±0.010	0.792±0.004	0.792±0.009	0.788±0.004	0.792±0.007	**0.797±0.010**	0.793±0.007
Student	0.948±0.001	0.947±0.001	0.949±0.001	**0.950±0.001**	0.949±0.000	0.948±0.001	0.948±0.000	**0.950±0.001**
Churn	0.796±0.003	**0.797±0.001**	**0.797±0.002**	0.794±0.000	0.792±0.001	0.795±0.001	**0.797±0.001**	**0.797±0.002**
Titanic	0.820±0.008	0.809±0.004	0.818±0.010	0.826±0.009	0.822±0.016	0.828±0.008	0.830±0.010	**0.837±0.016**
Wine	0.840±0.004	0.844±0.005	0.839±0.004	**0.847±0.006**	0.838±0.003	0.832±0.008	0.840±0.003	0.840±0.005
Mean	0.863	0.867	0.862	0.866	0.869	0.861	0.870	**0.874**
MeanRank	3.87	3.75	4.00	2.75	2.68	4.26	2.50	**1.44**

Table 13: Computation time and token usage on 16 classification datasets.

	Adult	Balancee	Bank	Banknote	Breast_W	Car_Eval	Cdc	Credit_G
Time (h)	0.22	0.12	0.21	0.11	0.20	0.11	2.43	0.31
Tokens (k)	111	88	141	93	141	97	164	181

	Heart	Jungle	Myocaridial	Pima	Student	Churn	Titanic	Wine	Mean
Time (h)	0.19	0.63	1.32	0.21	0.31	0.17	0.35	0.35	0.452
Tokens (k)	105	118	463	126	153	94	142	144	147.57

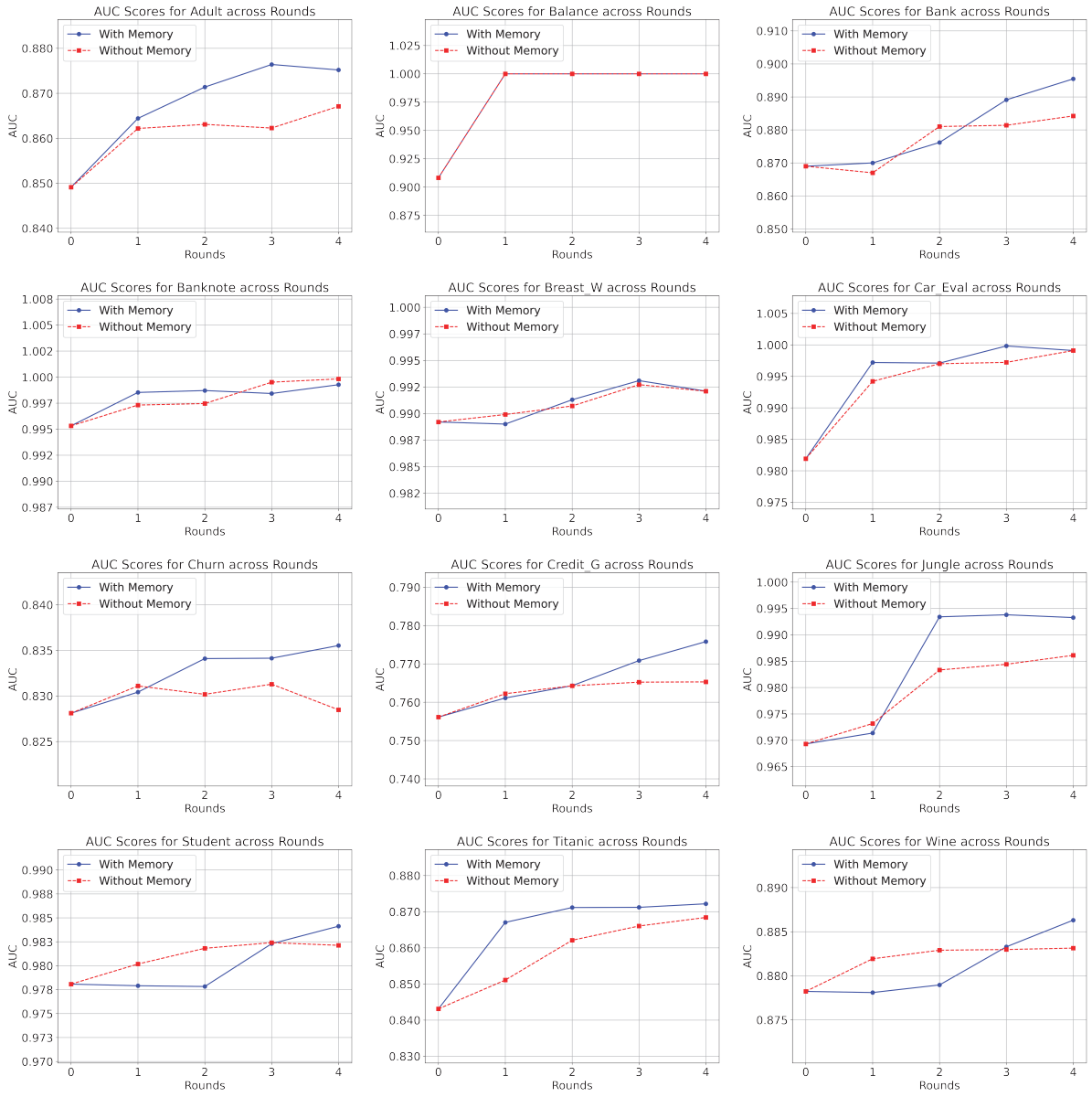

Figure 5: AUC scores for multiple datasets across different rounds with and without memory