Eureka: Intelligent Feature Engineering for Enterprise AI Cloud Resource Demand Prediction

Hangxuan Li[*]
School of Computer Science, Fudan University
Shanghai, China
hxli23@m.fudan.edu.cn

Renjun Jia
School of Computer Science and Technology, Tongji University
Shanghai, China
2332101@tongji.edu.cn

Xuezhang Wu[†]
Alibaba Cloud Computing Co. Ltd
Hangzhou, China
weixi.wxz@alibaba-inc.com

Yunjie Qian
Alibaba Cloud Computing Co. Ltd
Hangzhou, China
ramsey.qyj@alibaba-inc.com

Zeqi Zheng
Alibaba Cloud Computing Co. Ltd
Hangzhou, China
zhengzeqi.zzq@alibaba-inc.com

Xianling Zhang
Independent Researcher
United States
lilyzhng.ai@gmail.com

Abstract

The rapid growth of foundation models (LLMs, VLMs) has surged enterprise cloud AI demand, where GPU resources must be dynamically allocated to meet evolving workloads. Accurate demand prediction is critical for efficient and reliable real-world deployment, yet traditional forecasting systems struggle due to sparse historical data and highly volatile workload behaviors. We present **Eureka**, an *LLM-driven agentic framework* that automates feature engineering. Our approach has three main components: a domain knowledge-driven *Expert Agent* that encodes cloud resource expertise to evaluate feature quality, an Automated Feature Generator that explores new feature spaces, lastly a RL Feedback Loop (reinforcement learning) connects the two components and enables continuous learning. Deployed and evaluated on real-world cloud provider datasets, Eureka improves demand fulfillment rate by 16%, and reduces computing resource migration rates by 33%. This work introduces a novel intelligent system for cloud resource prediction and AI supply chain management, advancing the efficiency, scalability, and deployability of foundation models in production environments.

[*]Hangxuan Li and Renjun Jia contributed equally to this work. This work was done during two author's internship at Alibaba Cloud Computing Co. Ltd.

[†]Corresponding author.

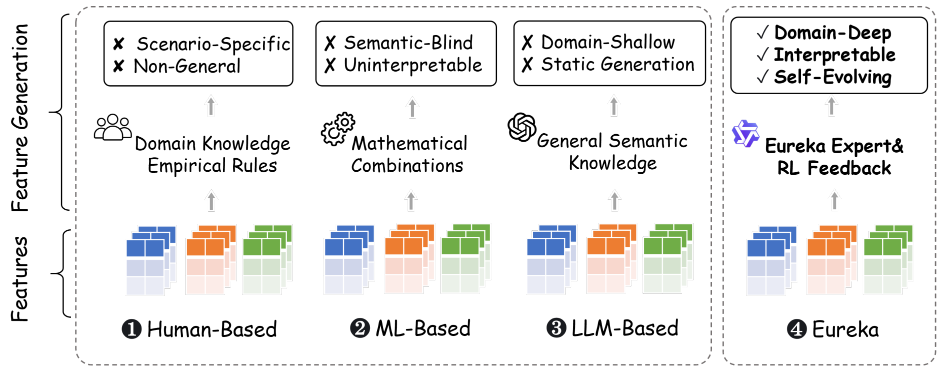


Figure 1: Comparison of different feature generation paradigms, highlighting their respective features, generation methods, and limitations/advantages.

1 Introduction

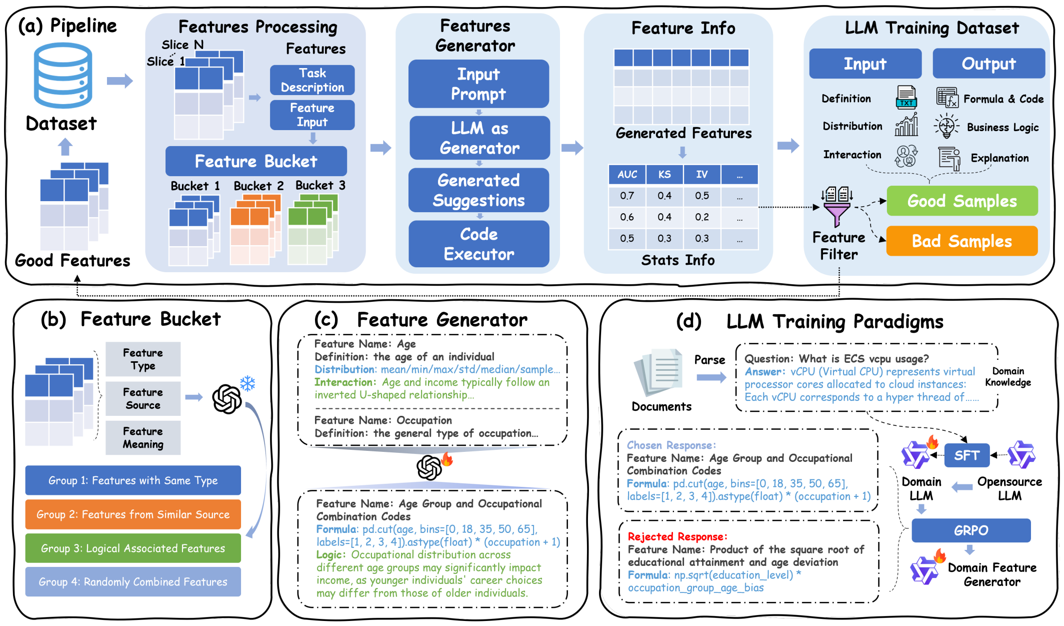
Cloud providers face an operational challenge: ensure sufficient GPU capacity for foundation models workloads (LLMs, VLMs) despite bursty and irregular demand. These workloads often leave GPU clusters underutilized during idle periods and long-queued at peak [14, 17, 19]. Unlike traditional cloud services with stable, repeatable usage patterns, foundation model workloads have fluctuations and changing deployment patterns, making existing resource management and forecasting methods unreliable. This creates a significant trade-off between service reliability and cost efficiency, so accurate demand prediction is essential.

To achieve accurate prediction, cloud providers need predictive features for forecasting bursty GPU demand. However, even with proactive GPU provisioning, cloud providers still face online shortages when bursty demand exceeds allocation. Features crafted for conventional steady-state workloads are not suitable for new AI workload scenarios. Without domain expertise and finetuning for feature design, existing feature engineering approaches (see Figure 1) listed below remain limited:

- **(1) Manual FE** relies on domain expertise, resulting in a limited set of features being effective in narrow contexts and fails to generalize.
- **(2) Automated FE (AutoFE)** scales feature generation but lacks semantic understanding, producing features that failed to capture bursty demand patterns [8, 9, 21].
- **(3) LLM-based FE** are static generators that lacks feedbacks for post-deployment iterations, and does not adapt to irregular, bursty patterns in AI workloads [3, 6, 13].

To address these limitations, we introduce **Eureka**, an agentic feature engineering framework that discovers high-quality, logical features even with data scarcity constraints across LLM and VLM applications. The operational challenge it addresses—forecasting GPU demand under bursty, irregular workload patterns—is directly motivated by large-scale VLM/LLM deployment in production. Our system is now fully integrated into the production services. Our key contributions are highlighted below.

- **Eureka Expert (Planning Stage)** encodes domain knowledge as heuristic constraints and design templates to guide exploration of feature interactions and cross-resource dependencies. It steers toward domain-meaningful candidates, filters spurious correlations, and identifies high-order feature dependencies that reflect system dynamics.
- **LLM Feature Factory (Execution Stage)** translates design plans into concrete features via conditioned code generation. Guided by domain heuristics, it explores the feature space and outputs executable implementations that maximize information signal from limited historical data.
- **Self-Evolving Alignment Engine (Refinement Stage)** improves feature quality using dual-channel reward feedback from real-world deployment outcomes. By linking offline optimization with online performance, it adapts features to irregularworkload patterns while preventing overfitting.



AUC	KS	IV	...
0.7	0.4	0.5	...
0.6	0.4	0.2	...
0.5	0.3	0.3	...

Feature Name: Age
Definition: the age of an individual
Distribution: mean/min/max/std/median/sample...
Interaction: Age and income typically follow an inverted U-shaped relationship...

Feature Name: Occupation
Definition: the general type of occupation...

Feature Name: Age Group and Occupational Combination Codes
Formula: pd.cut(age, bins=[0, 18, 35, 50, 65], labels=[1, 2, 3, 4]).astype(float) * (occupation + 1)
Logic: Occupational distribution across different age groups may significantly impact income, as younger individuals' career choices may differ from those of older individuals.

Question: What is ECS vcpu usage?
Answer: vCPU (Virtual CPU) represents virtual processor cores allocated to cloud instances: Each vCPU corresponds to a hyper thread of......

Chosen Response:
Feature Name: Age Group and Occupational Combination Codes
Formula: pd.cut(age, bins=[0, 18, 35, 50, 65], labels=[1, 2, 3, 4]).astype(float) * (occupation + 1)

Rejected Response:
Feature Name: Product of the square root of educational attainment and age deviation
Formula: np.sqrt(education_level) * occupation_group_age_bias

Figure 2: **Domain-specific feature generation framework.** (a) Pipeline: End-to-end workflow for generating and filtering domain-aware features. (b) Feature Bucket: Groups features by type, source, or semantic relationships. (c) Feature Generator: Uses LLMs to generate mathematical formulas and Python code for feature creation. (d) LLM Training Paradigms: Fine-tunes LLMs via SFT [18] and GRPO [16] for feature reasoning.

2 Related Works

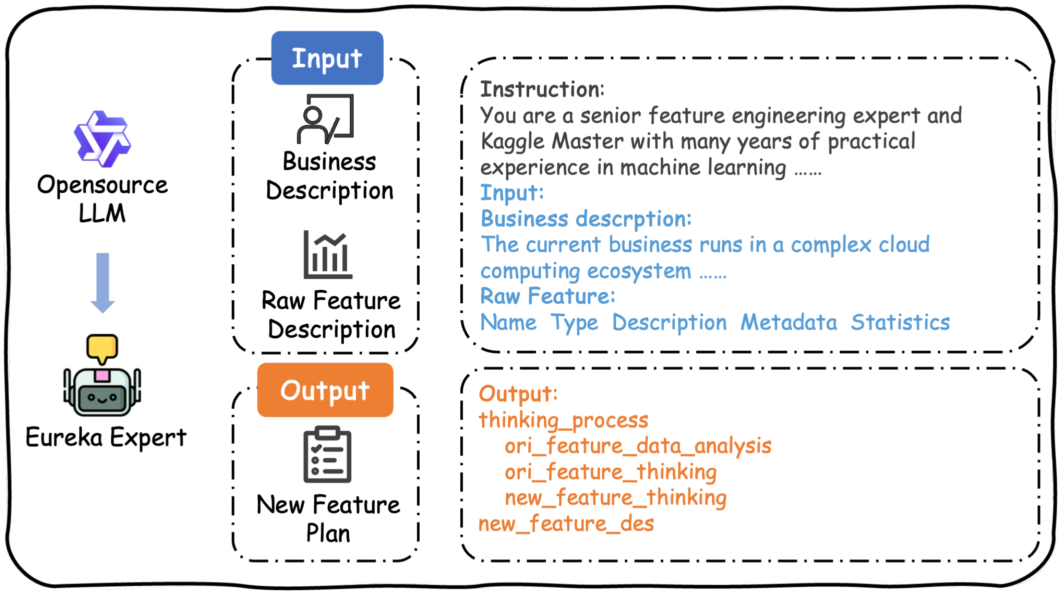
2.1 Automated Feature Engineering

Feature engineering remains a critical bottleneck in machine learning deployment, requiring both statistical expertise and domain knowledge while being largely manual and experience-driven. AutoFE has emerged as a core component of AutoML to reduce iterative trial-and-error costs. FeatureTools is the most widely adopted approach, enabling automated multi-table feature fusion via Deep Feature Synthesis (DFS) [9] for relational database. Additionally there are methods of iterative feature subsampling through bundle search to identify information-rich features [8], and feature boosting with pruning algorithms for efficient, accurate filtering [21, 11].

2.2 LLM for AutoFE

Recent advances in foundation models have enabled their application to automated feature engineering [4]. Hollmann et al. introduce a context-aware method that leverages LLMs to generate semantically meaningful features from task descriptions [6]. Han et al. extract rule-based binary features using prior knowledge and contextual cues to support classification tasks [3]. Nam et al. frame feature generation as a rule synthesis problem, an LLM suggests transformation rules while a decision tree identifies the optimal ones [13].

However, these methods rely on general world knowledge and lack deep domain expertises, making them less effective for complex, specialized datasets. Most importantly, they operates in open-loop manner thus cannot learn from past success or failures, often degrading to fixed combination patterns. To overcome these limitations, we propose **Eureka**, a self-evolving framework that integrates domain-guided knowledge with feedback-driven refinement.



Opensource LLM
Eureka Expert
Input
Business Description
Raw Feature Description
Output
New Feature Plan
Instruction:
You are a senior feature engineering expert and Kaggle Master with many years of practical experience in machine learning ……
Input:
Business descrption:
The current business runs in a complex cloud computing ecosystem ……
Raw Feature:
Name Type Description Metadata Statistics
Output:
thinking_process
ori_feature_data_analysis
ori_feature_thinking
new_feature_thinking
new_feature_des

Figure 3: Eureka Expert

3 Methodology

3.1 Eureka Expert

The Eureka Expert is a domain-specialist LLM model that guides principled feature exploration by encoding domain knowledge and business logic. General-purpose LLMs are insufficient in specialized business settings, so we apply supervised fine-tuning (SFT) to embed this knowledge. The open-source models has been deployed on internal infrastructure to enable task-specific customization and protect data security.

Our automated feature engineering is formulated as a conditional generation problem: given business context and statistical profiles of original features, the objective is to generate a structured feature design plan. This is expressed as $P^* = \arg\max_P p_\theta(P \mid C, M)$, where C, M, and P denote the business scenario description, feature metadata with statistical profiles, and the generated plan (following a predefined schema), respectively; $p_\theta(P \mid C, M)$ is the LLM-parameterized conditional distribution.

The model is fine-tuned on an expert-annotated dataset $\mathcal{D}$ to produce high-quality, domain-aligned feature design plans. Each sample includes an input I_i (role definitions, task requirements, and feature statistics) and an output O_i (expert-generated plan with reasoning). This process teaches structured reasoning and domain-specific construction strategies, formalized as:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\sum_{(I_i,O_i)\in\mathcal{D}} \log p_\theta(O_i \mid I_i) \tag{1}$$

where $\mathcal{D} = \{(I_i, O_i)\}_{i=1}^N$ is the training dataset. Minimizing $\mathcal{L}_{\mathrm{SFT}}$ enables the model to learn expert reasoning and feature construction patterns, embedding domain-specific knowledge for better generalization across business scenarios. The resulting model outputs a reasoned feature design plan for execution by the LLM Feature Factory.

3.2 LLM Feature Factory

The LLM Feature Factory is the execution module that translates the feature design plans of the Eureka Expert into concrete features through code generation and validation. Feature engineering can

Algorithm 1 Three-Phase LLM Feature Engineering

```
Require: Dataset $\mathbf{X}$, LLM $\mathcal{M}$, iterations $K$
Ensure: Enhanced dataset $\mathbf{X}_{enhanced}$
 1: Phase 1: $\mathcal{X} \leftarrow$ partition($\mathbf{X}$) into $\{\mathcal{X}_{num}, \mathcal{X}_{cat}, ...\}$
 2: Initialize $H_0 \leftarrow \emptyset$
 3: for $k = 1$ to $K$ do
 4:     Phase 2: Prepare inputs $(N_j, D_j, H_{k-1}, \text{stats}(\mathbf{X}))$
 5:     Phase 3 - Chain of Thought:
 6:         $R_k \leftarrow \mathcal{M}$.generate_rationale(inputs)
 7:         $F_k \leftarrow \mathcal{M}$.formulate_rule($R_k$)
 8:         $C_k \leftarrow \mathcal{M}$.generate_code($F_k$)
 9:     $\mathbf{x}_{new} \leftarrow$ execute($C_k, \mathbf{X}$)
10:     $P_k \leftarrow$ evaluate($\mathbf{x}_{new}$)
11:     if $P_k >$ threshold then
12:         Update: $\mathbf{X} \leftarrow \mathbf{X} \cup \{\mathbf{x}_{new}\}$, $H_k \leftarrow H_{k-1} \cup \{(N_k, D_k, P_k, S_k)\}$
13:     end if
14: end for
15: return $\mathbf{X}$
```

be formulated as a bilevel optimization problem that discovers a feature $\mathcal{X}'$ through a transformation $\mathcal{F} : \mathcal{X} \to \mathcal{X}'$, where a model f trained in the augmented feature space achieves better performance:

$$\max_{\mathcal{F}} \xi(f^*; \mathcal{D}_{val} \cup \mathcal{F}(\mathcal{D}_{val})) \quad \text{s.t.} \quad f^* \in \arg\max_{f} \xi(f; \mathcal{D}_{train} \cup \mathcal{F}(\mathcal{D}_{train})) \tag{2}$$

Traditional gradient-based methods face challenges due to non-differentiable transformations and computational complexity. To overcome these limitations, we introduce a framework that leverages a LLM as an outer-loop optimizer, reformulating feature engineering as an iterative search process.

Our methodology, outlined in Algorithm 1, operates in three phases. In Phase 1, the dataset is partitioned into subsets by types. Phase 2 prepares inputs including feature metadata, historical records, and statistical summaries. Phase 3 employs a Chain of Thought approach where the LLM generates a transformation rationale, formulates a feature rule, and produces executable code. New features are evaluated and good ones are retained while updating the history for subsequent iterations. This process enables exploration of the transformation space, leveraging semantic understanding to generate meaningful features.

3.3 Self-Evolving Alignment Engine

The Self-Evolving Alignment Engine is an RL-based module that aligns the SFT model's outputs with human reasoning and domain-specific context [15]. It constructs a joint reward function combining quantitative and semantic evaluations, then applies the GRPO algorithm to optimize parameters, enhancing reasoning quality and business relevance while maintaining structured output.

To balance feature performance and reasoning quality, the reward is defined as:

$$R_{\text{total}}(P_i) = \alpha \cdot R_{\text{metric}}(P_i) + \beta \cdot R_{\text{semantic}}(P_i) \tag{3}$$

where $R_{\text{metric}}(P_i) = \mathbf{w}^T \widetilde{\mathbf{m}} i$ quantifies feature performance through normalized indicators, and $R_{\text{semantic}}(P_i)$ evaluates reasoning completeness, causal logic, and domain alignment:

$$R_{\text{semantic}}(P_i) = \frac{K - \text{rank}_{\text{LLM}}\left(P_i \,\middle|\, \{P_j\}_{j=1}^{K}, \mathcal{C}\right)}{K - 1} \tag{4}$$

Here, K the number of candidate features. A stronger LLM performs ranking-based evaluation to generate real-time reward signals without training a separate reward model, consistent with the GRPO framework. The combined reward $R\text{total}$ then guides intra-group preference learning, maintaining training stability and integrating both metric and semantic quality.

Despite involving multiple components , the majority of Eureka's computation occurs offline during monthly retraining cycles used to refresh feature candidates. In deployment, we observe that the

Table 1: Basic information of datasets. Numbers in parentheses denote categorical/numerical features.

Data	# of samples	# of features	Label ratio (%)
Adult	48,842	14 (7/7)	76:24
Bank	45,211	16 (8/8)	88:12
Blood	748	4 (0/4)	76:24
Credit-g	1,000	20 (12/8)	70:30
Diabetes	768	8 (0/8)	65:35
Heart	918	11 (4/7)	45:55
Myocardial	1,700	111 (94/17)	22:78
EGS	4,837	45 (0/45)	12:88

Table 2: Performance comparison across methods and datasets. Bold numbers indicate the best performance for each dataset.

Method	Adult	Bank	Blood	Credit	Diabetes	Heart	Myocardial	EGS
DFS	0.915	0.897	0.637	0.707	0.882	0.921	0.654	0.680
AutoFE	0.872	0.865	0.735	0.676	0.837	0.903	0.674	0.666
OpenFE	0.920	0.923	0.626	0.701	0.817	0.897	0.697	0.658
TabPFN	0.900	0.904	0.715	0.787	**0.888**	0.938	0.676	0.694
CAAFE	0.901	0.905	0.713	0.797	0.886	**0.941**	0.686	0.692
FeatLLM	0.894	0.851	0.678	0.743	0.811	0.881	0.663	0.675
Eureka-Qwen	0.926	0.932	0.738	0.794	0.873	0.926	**0.737**	0.679
Eureka-32B	0.926	0.932	0.716	0.824	0.884	0.936	0.711	0.688
Eureka-gpt4o	**0.928**	**0.934**	**0.745**	**0.838**	0.881	0.938	0.699	**0.699**
Eureka-grok2	**0.928**	**0.934**	0.735	0.815	0.886	**0.940**	0.719	0.686

incremental cost of offline feature search is outweighed by its operational benefits: proactive surge prediction reduces costly GPU under-provisioning events and significantly lowers server migration overhead. This cost–benefit ratio motivated the system's adoption in production at scale.

4 Experiments

4.1 Experimental Setup

Datasets. Evaluations are conducted on 8 classification datasets: Adult [1], Bank [12], Blood [20], Credit-g [5], Diabetes[3], Heart[4], Myocardial [2], and a proprietary EGS (Elastic GPU Service) dataset. The EGS dataset is derived from real GPU usage logs and used for bursty demand prediction, it is a binary classification dataset with monthly granularity, where label 1 indicates a sudden surge in a user's GPU usage during that month, and 0 indicates stable usage.

Baselines. Extensive evaluations are conducted against 5 automated FE methods: Deep Feature Synthesis (DFS) [9], AutoFE [8], OpenFE [21], CAAFE [6], and FeatLLM [3], covering traditional and LLM-based approaches.

Experimental Setup. Features from all methods are combined with original features and evaluated using LightGBM [10] and TabPFN [7]. Performance is measured by ROC-AUC, reporting the best result across both predictors.

4.2 Overall Performance Comparison

Based on the results in Table 2, several key findings emerge. The proposed Eureka framework consistently achieves strong and superior performance across multiple datasets, particularly excelling

[3]https://kaggle.com/datasets/uciml/pima-indians-diabetes-database
[4]https://kaggle.com/datasets/fedesoriano/heart-failure-prediction

Table 3: Ablation study of EUREKA components.

Configuration	ROC-AUC (%)	Δ vs. Eureka (%)
Eureka	69.97	—
w/o Self-Evolving Engine	69.45	-0.52
w/o Eureka Expert	65.91	-4.06
w/o LLM Factory	63.84	-6.13

on feature-rich data such as Myocardial. This advantage comes from Eureka's ability to semantically interpret original features and generate domain-informed feature combinations, which is especially beneficial in complex, high-dimensional settings.

Eureka-Qwen, Eureka-grok2, and Eureka-gpt4o utilize proprietary APIs, while Eureka-32B is a fine-tuned open-source variant based on Qwen3-32B. Among them, Eureka-gpt4o attains the highest accuracy on four datasets (Adult, Bank, Blood, and Credit-g) and remains competitive elsewhere. Eureka-Qwen leads on Myocardial (0.737), highlighting its strength in handling feature-rich data, while Eureka-grok2 performs best on Heart and shows strong results on Adult and Bank. Overall, the consistent superiority of Eureka demonstrates the framework's effectiveness and generalization ability in automated feature engineering.

4.3 Ablation Study

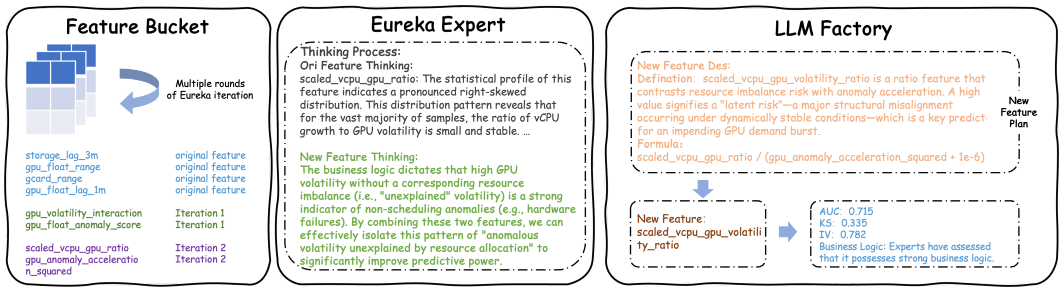
We conducted ablation studies to evaluate each component of Eureka on the EGS dataset by systematically removing core modules. The configurations and performance are detailed in Table 3.

First, the LLM factory is essential, with its removal causing the largest performance drop (6.13%). This underscores the superiority of our automated feature discovery method compared to the traditional manual feature engineering.

Second, domain knowledge injection is critical. Replacing Eureka Expert with a generic LLM degrades performance by 4.06% . This validates that SFT-injected domain insights are vital for generating business-relevant features.

Finally, the self-evolving alignment provides crucial optimization. While its removal causes a 0.52% loss, it leverages dual-channel reward feedback to guide features generation of enhanced interpretability and coherence, enabling the model to internalize successful strategies for subsequent iterations.

4.4 Case Study



storage_lag_3m	original feature
gpu_float_range	original feature
gcard_range	original feature
gpu_float_lag_1m	original feature
gpu_volatility_interaction	Iteration 1
gpu_float_anomaly_score	Iteration 1
scaled_vcpu_gpu_ratio	Iteration 2
gpu_anomaly_acceleratio n_squared	Iteration 2

Thinking Process:
Ori Feature Thinking:
scaled_vcpu_gpu_ratio: The statistical profile of this feature indicates a pronounced right-skewed distribution. This distribution pattern reveals that for the vast majority of samples, the ratio of vCPU growth to GPU volatility is small and stable. ...

New Feature Thinking:
The business logic dictates that high GPU volatility without a corresponding resource imbalance (i.e., "unexplained" volatility) is a strong indicator of non-scheduling anomalies (e.g., hardware failures). By combining these two features, we can effectively isolate this pattern of "anomalous volatility unexplained by resource allocation" to significantly improve predictive power.

New Feature Des:
Defination: scaled_vcpu_gpu_volatility_ratio is a ratio feature that contrasts resource imbalance risk with anomaly acceleration. A high value signifies a "latent risk"—a major structural misalignment occurring under dynamically stable conditions—which is a key predict for an impending GPU demand burst.
Formula:
scaled_vcpu_gpu_ratio / (gpu_anomaly_acceleration_squared + 1e-6)

New Feature:
scaled_vcpu_gpu_volatili ty_ratio

AUC: 0.715
KS: 0.335
IV: 0.782
Business Logic: Experts have assessed that it possesses strong business logic.

Figure 4: The process of obtaining high-quality features.

To illustrate Eureka's practical impact, we present a production case study demonstrating Eureka's effectiveness in predicting a GPU demand burst, which is undetectable using traditional monitoring system. As shown in Figure 4, Eureka discovers interpretable, high-quality features by synthesizing domain knowledge rather than exhaustive feature combinations.

Table 4: Quantifiable business impact of EUREKA at a leading cloud provider.

Impact area	**Key business metric**	**Improvement**
Operational adoption	Adoption rate of generated warnings	**91%**
Demand fulfillment	Demand fulfillment rate (top-tier customers)	**+16%**
Resource efficiency	Reduction in server migration loss rate	**-33%**

Initially, the system identified two features with weak correlations: `gpu_anomaly_acceleration_squared` and `scaled_vcpu_gpu_ratio`. However, Eureka Expert leverages domain knowledge to reinterpret `scaled_vcpu_gpu_ratio`: a high ratio indicates customers' preparation (data pipeline setup, resource provisioning) preceding GPU-intensive workloads, creating a structural resource imbalance in anticipation of a major GPU demand surge. This "preparatory imbalance" is a subtle precursor that statistical models typically miss.

Guided by this insight, the LLM Feature Factory generates candidate features, and through validation by the Self-Evolving Alignment Engine, the new feature `scaled_vcpu_gpu_volatility_ratio` was identified as optimal to quantify the "preparatory imbalance" phase. It divides the high `scaled_vcpu_gpu_ratio` (the preparation signal) by near-zero `gpu_anomaly_acceleration_squared` (indicating a stable GPU state), amplifying the signal for pre-surge detection. In practice, this feature surges days before demand peaks while standard metrics remain flat, enabling timely forecasting from Eureka.

5 Application in Practice

The Eureka framework has been fully integrated in a major cloud provider's GPU resource management system, serving as a core module for enterprise AI computing demand forecasting. Its primary objective is to forecast GPU demand surges for the upcoming month through the integration of multi-source data streams, thereby generating early-warning signals for resource planning.

To achieve this, Eureka creates a systematic pipeline: training data is sourced from historical demand records and supplemented with domain expert inputs, while the model is retrained monthly based on prediction quality to ensure continuous adaptation to evolving demand patterns. Validation results from June to September 2025 demonstrate the framework's efficacy: Eureka generated 165 warning records, of which 91% were adopted by the GPU resource operations team as actionable recommendations, confirming its business impact (detailed in Table 4).

The Eureka framework has demonstrated significant business benefits in industrial applications. On the demand side, its ability to forecast sudden demand surges for top-tier customers (highest two levels) has increased their demand fulfillment rate by 16%, effectively safeguarding service quality for critical clients. On the supply side, Eureka has resolved the long-standing issue of high loss rates during cloud server migrations. By generating granular "customer + region" level demand forecasts, the system can reroute in-transit equipment before demand occurs, ensuring critical resources are deployed in advance while reducing unnecessary reallocations. Leveraging Eureka's migration strategy, GPU server migration rate has been reduced by by 33%, directly improving operational efficiency and reducing costs. These improvements significantly exceed the modest offline computational overhead introduced by the Eureka pipeline, highlighting a favorable cost–benefit tradeoff in production environments.

Eureka's advanced feature-engineering approach goes beyond traditional univariate time-series forecasting. The framework specifically exploits cross-feature interactions and temporal patterns for comprehending and addressing complex resource demand dynamics.

Beyond immediate operational gains, Eureka has proven its ability to deliver accurate forecasts and optimize resource utilization. It has demonstrated long-term value and is positioned to remain a cornerstone of the cloud GPU ecosystem. Its predictive framework is being extended across elastic computing to unify resource planning, improve utilization efficiency, enable proactive capacity management at scale, and other broader applications across other domains.

6 Conclusion

In this paper, we address the challenge of predicting enterprise GPU demand for cloud providers in the foundation model era (LLMs, VLMs). We propose Eureka, an agentic feature engineering framework that incorporates domain knowledge and continuous learning. Through its three-component architecture, Eureka combines the generative power of LLMs with domain expertises. We evaluated our approach on public benchmarks and a proprietary, real-world dataset from a leading cloud provider, assessing its performance under production workloads. The results demonstrate that Eureka outperforms traditional methods and existing AutoFE baselines, achieving a 4.95% improvement in ROC-AUC on the GPU burst demand prediction task.

Eureka has proven its value in business environments. Since its integration into a major cloud provider's GPU resource management system, warnings generated by Eureka have achieved a 91% adoption rate. The system has increased the demand fulfillment rate for top-tier customers by 16%, safeguarding business stability, and reduced the GPU server migration loss rate by 33% through forecasting, leading to improvements in resource utilization and operational efficiency.

In conclusion, Eureka presents a novel and effective solution for cloud resource prediction. More broadly, it demonstrates a scalable paradigm for integrating LLM-based agentic framework with domain knowledge in enterprise applications. While our evaluation focuses on GPU demand forecasting in cloud AI services, extending Eureka to other resource types (e.g., memory, network) or domains (e.g., database provisioning, serverless scaling) remains important directions. Additionally, the current framework relies on human expert-defined reward heuristics, so future work could explore self-supervised or adaptive reward learning to further reduce dependency on manual input. Overall, we believe this work paves the way for building more adaptive and knowledge-aware AI systems for supply chain and resource management, which empowers the foundation models deployments in real-world.

References

[1] Barry Becker and Ronny Kohavi. Uci machine learning repository: Adult data set, 2007.

[2] S. E. Golovenkin, V. V. Voino-Yasenetsky, V. O. Shulman, and D. Groen. Chronic heart failure data set for predictive modeling. *Scientific Data*, 7(1):1–8, 2020.

[3] Sungwon Han, Jinsung Yoon, Sercan Ö. Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. In *International Conference on Machine Learning*, 2024.

[4] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023.

[5] Hans Hofmann. Uci machine learning repository: German credit data, 2020.

[6] Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. In *The Thirty-seventh Annual Conference on Neural Information Processing Systems*, 2023.

[7] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025.

[8] F. Horn, Robert T. Pack, and Michael Rieger. The autofeat python library for automated feature engineering and selection. In *PKDD/ECML Workshops*, 2019.

[9] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2015.

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

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

[12] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[13] Jaehyun Nam, Kyuyoung Kim, Seunghyuk Oh, Jihoon Tack, Jaehyung Kim, and Jinwoo Shin. Optimized feature generation for tabular data via LLMs with decision tree reasoning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[14] OpenAI, Josh Achiam, Steven Adler, and et al. Gpt-4 technical report. 2024.

[15] Jiang X et al. Ouyang L, Wu J. Training language models to follow instructions with human feedback. In *Advances in neural information processing systems*, 2022.

[16] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

[17] Gemini Team, Rohan Anil, Sebastian Borgeaud, and et al. Gemini: A family of highly capable multimodal models, 2025.

[18] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652, 2021. URL `https://arxiv.org/abs/2109.01652`.

[19] An Yang, Anfeng Li, Baosong Yang, and et al. Qwen3 technical report. 2025.

[20] I-Cheng Yeh, King-Jang Yang, and Tao-Ming Ting. Uci machine learning repository: Blood transfusion service center data set, 2008.

[21] T. Zhang, Zheyu Zhang, Zhiyuan Fan, Haoyan Luo, Feng Liu, Li-Yu Daisy Liu, Qian Liu, Wei Cao, and Jian Li. Openfe: Automated feature generation with expert-level performance. In *International Conference on Machine Learning*, 2022.

A Details of Eureka

A.1 Eureka Expert

The Eureka Expert is a domain-specialist LLM that guides principled feature exploration by encoding domain knowledge and business logic. General-purpose LLMs are insufficient in specialized business settings, so we apply supervised fine-tuning (SFT) to embed this knowledge. The open-source models has been deployed on internal infrastructure to enable task-specific customization and ensure data security.

Our automated feature engineering is formulated as a conditional generation problem: given business context and statistical profiles of original features, the objective is to generate a structured feature design plan. This is expressed as:

$$P^* = \arg\max_{P} \, p_\theta(P \mid C, M) \tag{5}$$

where C, M, and P represent the business scenario description, feature metadata with statistical profiles, and generated feature design plan (following a predefined JSON schema), respectively. Here, $p_\theta(P \mid C, M)$ is the conditional probability modeled by the LLM with parameters θ.

The model is fine-tuned on an expert-annotated dataset $\mathcal{D}$ to produce high-quality, domain-aligned feature design plans. Each sample includes an input I_i (role definitions, task requirements, and feature statistics) and an output O_i (expert-generated plan with reasoning). This process teaches structured reasoning and domain-specific construction strategies, formalized as:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{(I_i, O_i)\in\mathcal{D}} \log p_\theta(O_i \mid I_i) \tag{6}$$

where $\mathcal{D} = \{(I_i, O_i)\}_{i=1}^{N}$ is the training dataset. Minimizing $\mathcal{L}_{\text{SFT}}$ encourages the model to learn expert reasoning and feature construction patterns, injecting domain-specific knowledge into its parameters. By covering diverse business scenarios and high-quality feature sets in $\mathcal{D}$, the fine-tuned model generalizes across different domains and prediction tasks. The SFT process is conducted on open-source LLMs within secure internal infrastructure, ensuring enterprise data protection. This module outputs a feature design plan with reasoning, which is passed to the LLM Feature Factory for implementation.

A.2 Self-Evolving Alignment Engine

The Self-Evolving Alignment Engine is a reinforcement learning (RL)-based module. Its objective is to align the SFT model's outputs with human-preferred reasoning and domain-specific business context[15]. This enables the model to internalize the generation patterns of both high- and low-quality features, thereby improving subsequent generation and generalization. To achieve this, the Self-Evolving Alignment Engine constructs a combined reward function using quantitative metrics and semantic evaluations to assess candidate feature design plans. It then applies the GRPO algorithm to iteratively optimize parameters. This guides the model to preserve structured output while enhancing reasoning quality, business logic relevance, and internalizing high-quality generation patterns.

To balance feature performance and reasoning quality, the Self-Evolving Alignment Engine decomposes the reward function into a metric-based component and a semantic-based component. These are combined through weighted aggregation to produce the final optimization signal.

$$R_{\text{total}}(P_i) = \alpha \cdot R_{\text{metric}}(P_i) + \beta \cdot R_{\text{semantic}}(P_i) \tag{7}$$

The metric-based reward R_{metric} quantifies feature performance using a set of normalized statistical and modeling indicators, while the semantic-based reward R_{semantic} evaluates reasoning quality and domain-specific business alignment. R_{metric} is computed as the weighted dot product:

$$R_{\text{metric}}(P_i) = \mathbf{w}^T \widetilde{\mathbf{m}}_i \tag{8}$$

$\mathbf{w} = [w_{\text{a}}, w_{\text{k}}, w_{\text{i}}, w_{\text{t}}, w_{\text{imp}}, -w_{\text{r}}]^T$ is the weight vector, and $\widetilde{\mathbf{m}}_i$ is the normalized metric vector with all metric values scaled to $[0, 1]$.

In addition to metric-based performance, the semantic-based reward R_{semantic} measures the quality of the model's reasoning based on completeness, causal logic soundness, and alignment with business context. It is computed by employing a stronger LLM to perform group-wise ranking over candidates, with the resulting rank mapped to a scalar score in $[0, 1]$. The reward is formally defined as:

$$R_{\text{semantic}}(P_i) = \frac{K - \text{rank}_{\text{LLM}}\left(P_i \,\middle|\, \{P_j\}_{j=1}^{K}, \mathcal{C}\right)}{K - 1} \tag{9}$$

Here, K is the number of candidate features, and $\text{rank}_{\text{LLM}}(\cdot)$ is the rank assigned by the stronger LLM evaluator. This evaluator assesses the candidate set $\{P_j\}_{j=1}^{K}$ based on criteria $\mathcal{C}$: (1) reasoning completeness, (2) causal logic soundness, and (3) domain alignment. Inspired by RLAIF, we use the LLM evaluator to directly produce real-time reward signals, circumventing the need to train a separate reward model. This aligns with the GRPO framework by leveraging candidate groups and relative rankings.

After defining the reward signals, model training proceeds under the GRPO framework. The combined reward R_{total} serves as the optimization signal for intra-group preference learning, which circumvents explicit reward model training. This design preserves GRPO stability while internalizing both metric-based and semantic quality criteria.

In our implementation, we set $K = 5$ to balance preference estimation accuracy and computational cost. The weights α and β controlling the trade-off between metric-based and semantic-based rewards are tuned on a held-out validation set and set to $\alpha = 0.6$ and $\beta = 0.4$ in our main experiments. All metric values are min–max normalized to $[0, 1]$, and the semantic ranking is obtained from a GPT-4-based evaluator with a fixed evaluation prompt encoding the criteria in $\mathcal{C}$. Candidate generation and LLM-based evaluation are executed in batched mode to reduce latency. Training employs the Adam optimizer with a learning rate of 2×10^{-5}, a group batch size of $N_g = 64$, and a clipping parameter of 0.2 for stable updates.

B Details of Experiment

B.1 Datasets

Our experimental evaluation employs eight distinct datasets for classification tasks: (1) Adult[1] for predicting whether an individual's annual income exceeds \$50,000; (2) Bank[12] for forecasting client subscriptions to a term deposit; (3) Blood[20] for classifying whether blood donors will return; (4) Credit-g [5]for assessing an individual's credit risk as good or bad; (5) Diabetes for diagnosing the presence of diabetes mellitus; (6) Heart for predicting the presence of coronary artery disease; (7) Myocardial[2] for identifying individuals with chronic heart failure; and (8) EGS proprietary dataset, which constitutes a closed-source internal dataset derived from Alibaba Cloud's Elastic Compute Service operational metrics. This specialized dataset facilitates predictive modeling of GPU usage burst patterns by leveraging historical cloud service utilization data to forecast the probability of significant GPU usage fluctuations (both upward and downward transitions) within a one-month prediction horizon.

B.2 Baselines

To comprehensively evaluate our feature engineering approach, we selected five representative automated feature engineering methods as baselines, encompassing both traditional and large language model (LLM)-based paradigms: (1) Deep Feature Synthesis (DFS) [9], which automates feature generation through relational data traversal and mathematical transformations; (2) AutoFE [8], a Python library that systematically generates and selects features through iterative engineering processes; (3) OpenFE [21], an automated feature generation framework capable of achieving expert-level performance through efficient candidate generation and selection; (4) CAAFE [6], a context-aware feature engineering method that utilizes LLMs to generate semantically meaningful features based on dataset descriptions; and (5) FeatLLM [3], which demonstrates LLMs' capability to automatically engineer features for few-shot tabular learning. These baselines were selected to represent the evolution of automated feature engineering from traditional symbolic approaches to LLM-driven methods.