

Multi-task LSTM-Attention with Adaptive Isolation Forest for Intelligent Project Implementation Monitoring

Xiaocong Ruan

Guangdong University of Technology

Yaojia Wang

Guangdong University of Technology

Rixi Mo

Guangdong University of Technology

Changcheng Shao

Guangdong University of Technology

Zhouqiang Qiu

Guangdong Science and Technology Innovation Monitoring and Research Center

Cheng Zeng

Guangdong University of Technology

Lili Chen

493411276@qq.com

Guangdong Science and Technology Innovation Monitoring and Research Center

Liang Luo

Guangdong Science and Technology Innovation Monitoring and Research Center

Hongsong Zheng

Guangdong Jinke Information Network Center Co., Ltd.

Pinghua Chen

Guangdong University of Technology

Article

Keywords:

Posted Date: April 17th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9252159/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

Multi-task LSTM-Attention with Adaptive Isolation Forest for Intelligent Project Implementation Monitoring

Xiaocong Ruan^{1,†}, Yaojia Wang^{2,†}, Rixi Mo^{3,†}, Changcheng Shao⁴, Zhouqiang Qiu⁵, Cheng Zeng⁶, Lili Chen^{5,*}, Liang Luo⁵, Hongsong Zheng⁷, and Pinghua Chen⁶

¹School of Electromechanical Engineering, Guangdong University of Technology, Guangzhou Guangdong Province 510006, China

²School of Information Engineering, Guangdong University of Technology, Guangzhou Guangdong Province 510000, China

³School of Automation, Guangdong University of Technology, Guangzhou Guangdong Province 510000, China

⁴School of Management, Guangdong University of Technology, Guangzhou Guangdong Province 510520, China

⁵Guangdong Science and Technology Innovation Monitoring and Research Center, Guangzhou Guangdong Province 510033, China

⁶School of Computer Science and Engineering, Guangdong University of Technology, Guangzhou Guangdong Province 510006, China

⁷Guangdong Jinke Information Network Center Co., Ltd., Guangzhou Guangdong Province 510000, China

[†]These authors contributed equally to this work.

*Corresponding author. E-mail: 493411276@qq.com

ABSTRACT

Manual oversight of funded research projects does not scale. Progress delays, budget irregularities, and superficial reporting often go undetected until final acceptance, while existing detection methods struggle to distinguish legitimate schedule adjustments from genuine anomalies. We present the Intelligent Project Monitoring System (IPMS), which couples a feature-decoupled multi-task LSTM-Attention network with an Adaptive Isolation Forest. The LSTM-Attention component models project workflows through finite state machines and predicts milestone deviations; the Adaptive Isolation Forest flags anomalies after a legitimate deviation filter screens out justifiable variances that would otherwise inflate false-positive counts. A multi-head attention module separately tracks how execution performance evolves over the project lifecycle, and all components update online when distribution shifts are detected. On a real-world dataset with approximately 7% confirmed anomalies, IPMS achieved an AUC of 0.924 and a false-positive rate of 4.2%, a 76.9% reduction relative to standard Isolation Forest. Milestone deviation prediction reached an MAE of 1.85 days, 34.2% lower than standard LSTM. Execution profiling scored an MAE of 0.082. Removing the deviation filter alone degraded F1 by 12.3%, and removing multi-scale fusion increased the miss rate for long-duration stalls by 23%.

Introduction

Research Background and Motivation

The implementation phase of funded scientific research projects is where research plans are translated into tangible outcomes, making effective oversight essential for ensuring quality and accountability. As the number of government-funded projects grows, often involving thousands of concurrent projects, hundreds of implementing institutions, and multi-year execution cycles, management agencies face mounting supervisory challenges. The sheer scale of ongoing projects makes manual monitoring impractical. At the same time, implementation effectiveness is exposed to real threats: progress delays, budget misuse, deliverable quality degradation, and execution reporting deficiencies in which implementing teams submit execution reports that lack authenticity, completeness, or internal consistency across reporting periods.

Problem Statement

Despite the critical need for rigorous implementation oversight, existing project monitoring systems are inadequate due to four primary bottlenecks:

The Black-box Problem: Traditional project supervision workflows lack real-time status awareness. Implementation anomalies such as stalled progress or budget irregularities are typically discovered only during periodic inspections or final acceptance, preventing proactive interventions. For instance, a project may have exhausted 80% of its budget within the first quarter while completing only 10% of planned milestones, yet this critical mismatch remains invisible to administrators until the annual inspection months later.

The Misjudgment Problem: Existing anomaly detection methods often fail to distinguish between legitimate deviations and genuine implementation anomalies¹. For example, a biotechnology project may legitimately postpone its clinical trial milestone by six months due to updated regulatory requirements, yet a conventional anomaly detector would flag this delay identically to a project that has genuinely stalled due to mismanagement. This confusion leads to excessively high false positive rates, eroding trust in automated oversight².

The Static Profiling Problem: Project performance assessment conventionally relies on snapshot-based static metrics collected at discrete checkpoints. Such approaches completely overlook the temporal dynamics, evolutionary trends, and multi-dimensional correlations of a project's execution behavior over its lifecycle. For instance, a project that experienced severe delays in its first year but subsequently accelerated and exceeded targets would receive the same "underperforming" label as a project in sustained decline, because snapshot assessments cannot distinguish recovery trajectories from chronic underperformance.

The Fragmentation Problem: Currently, progress monitoring, anomaly detection, and project profiling are treated as isolated silos. No unified framework links these components into a coherent, closed-loop feedback system. As a result, when the anomaly detector flags a budget irregularity, administrators cannot immediately cross-reference it against the workflow status or execution profile to determine whether it reflects a genuine risk or a routine reallocation—they must manually consult separate, disconnected subsystems.

Research Questions

To address the aforementioned gaps, this study aims to answer the following research questions (RQs):

RQ1: How can we construct a real-time monitoring model that covers the entire project implementation workflow to achieve accurate progress state prediction and early warning of anomalies?

RQ2: How can we effectively differentiate legitimate implementation deviations from genuine project anomalies in the detection process to substantially reduce false positive rates?

RQ3: How can temporal behavioral modeling be utilized to realize dynamic profiling and evolutionary tracking of project execution performance?

RQ4: How can we design a unified framework that integrates these three modules into a closed-loop system of monitoring, detection, and analysis?

Main Contributions

Our work proposes a data-driven approach to intelligent project implementation monitoring. The main contributions are as follows: **Framework Contribution:** We propose the Intelligent Project Monitoring System (IPMS) framework, which combines three modules: a feature-decoupled multi-task LSTM-Attention network, an Adaptive Isolation Forest with legitimate deviation filtering, and a multi-dimensional project profiling system driven by dynamic weights.

Methodological Contribution: We introduce a legitimate deviation filtering mechanism that separates reasonable implementation variances (e.g., justified schedule adjustments) from actual anomalies before detection, reducing false positive rates. We also develop an online adaptive learning mechanism with distribution shift detection, allowing the system to adapt continuously in non-stationary environments.

Practical Contribution: We implement a deployable, end-to-end RESTful API system and validate it on a large-scale, real-world project implementation dataset. The system gives science and technology management agencies a working tool for ensuring implementation effectiveness and accountability.

Paper Organization

The remainder of this paper is organized as follows: Section 2 reviews related work in project monitoring, time-series prediction, and execution behavior modeling, explicitly identifying current research gaps. Section 3 formalizes the problem, defines the anomaly taxonomy, and details the architecture and specific algorithms of the IPMS framework. Section 4 presents the experimental setup, main results, ablation studies, and real-world case analyses. Finally, Section 5 concludes the paper, discussing the theoretical and practical implications, the interpretability of the models, current limitations, and avenues for future research.

Related Work

Project Monitoring and Anomaly Detection

Ensuring the implementation effectiveness of funded scientific research projects has historically relied on periodic inspections and manual supervision. Empirical studies have revealed that a substantial fraction of government-funded projects exhibit implementation anomalies such as progress delays, budget irregularities, and insufficient deliverable quality, and that these issues often go undetected until final acceptance³. Organizational factors and institutional relationships have also been shown to influence project execution patterns, introducing biases that traditional rule-based monitoring systems struggle to quantify⁴. Recent analyses of large-scale science and technology programs have demonstrated that strategic misreporting of milestones can persist undetected across multiple reporting cycles, pointing to the need for more sophisticated detection algorithms⁵. The growing volume of project data has further introduced challenges for quality assurance: automated text analysis of project execution reports reveals that a significant fraction contain templated or superficial content that obscures actual progress^{6,7}. In a broader context, analyzing execution behaviors and reporting content has been effective in distinguishing genuine progress from fabricated activities^{8,9}.

To overcome the limitations of manual oversight, machine learning techniques have been increasingly adopted. Graph Convolutional Networks (GCNs) have been proposed to model project execution as multi-relational weighted networks, capturing execution patterns and institutional behaviors more effectively than traditional statistical models¹⁰. Among unsupervised methods, Isolation Forest and its variants have received considerable attention. For data streams, Isolation Forest has been adapted to handle concept drifts^{11,12} and outliers in real-time¹³. Hariri et al. proposed Extended Isolation Forest (EIF) to resolve anomaly score artifacts caused by axis-aligned splits¹⁴. Xiang et al. established an isolation efficiency theory to determine the optimal branching factor, designing OptIForest with clustering-based learning-to-hash that outperformed deep learning baselines on benchmarks¹⁵. Utkin et al. incorporated the Nadaraya-Watson attention mechanism into Isolation Forest, achieving state-of-the-art results without gradient-based optimization¹⁶. However, a significant **research gap** remains: existing anomaly detection algorithms applied to project monitoring generally fail to distinguish between legitimate implementation deviations (e.g., justified schedule adjustments) and genuine anomalous behavior (e.g., budget misuse or fabricated progress). This lack of granularity often results in unacceptably high false positive rates in practical deployments. The present work addresses this gap by introducing a Legitimate Deviation Filtering mechanism that explicitly separates justifiable implementation variances from genuine anomalies before they enter the detection pipeline.

Time Series Prediction and Process Monitoring

Predicting how ongoing processes will progress—and where delays may occur—is a practical need in operational management. Long Short-Term Memory (LSTM) networks are widely used for sequence prediction¹⁷. Adding attention mechanisms to LSTM (LSTM-Attention) has been particularly effective, as it allows models to assign different importance to different temporal steps¹⁸. Cao et al. proposed a hybrid LSTM-Transformer architecture for real-time multi-task prediction in engineering systems, combining online learning with knowledge distillation to achieve both accuracy and computational efficiency¹⁹. Gao et al. introduced UniTS at NeurIPS 2024, a unified multi-task time series model that outperformed 12 forecasting models, 20 classification models, and 18 anomaly detection models across 38 diverse datasets by leveraging task tokenization and domain-transferable representations²⁰. Coupling finite state machine modeling with deep learning also allows for explicit structural representation of workflows while using neural networks to capture temporal dependencies.

In dynamic environments such as project implementation monitoring, static models inevitably suffer from concept drift. Sahoo et al. proposed the Hedge Backpropagation (HBP) method for Online Deep Learning, enabling adaptive-depth DNNs on streaming data²¹. Recent advances at ICLR 2024 introduced Kalman filter-based Bayesian online learning to track parameter drift in non-stationary environments²². Despite these advances, the integration of feature decoupling (separating static attributes from dynamic temporal behaviors) with online adaptive learning remains underexplored in the specific context of scientific

research project implementation monitoring. Our work fills this gap by combining a Feature Decoupling module—which explicitly separates static project attributes from dynamic execution behaviors—with an online adaptive learning mechanism that continuously recalibrates model parameters as new implementation data arrive.

Project Execution Behavior Analysis and Profiling

Traditional project profiling relies on static, snapshot-based metrics collected at discrete checkpoints, which fail to capture how a project’s execution pattern and performance evolve over time. Sun et al. proposed TREBA, a self-supervised multi-task learning method that reduces the annotation burden for behavior analysis by up to $10\times$ through “task programming”—encoding domain expert knowledge into reusable program structures²³. Arakawa et al. developed Providence, a visual-programming tool that lets experts combine ML algorithms for multimodal human behavior analysis without writing code²⁴. Multi-head attention mechanisms have also improved the interpretability of behavioral models. Still, there is a clear need for methods that can build dynamic, multi-dimensional profiles of scientific research projects—tracking execution efficiency, deliverable quality, budget compliance, and risk factors across the entire implementation lifecycle. To address this need, we design a multi-head attention LSTM behavior model with dynamic weight optimization that constructs evolving, five-dimensional execution profiles throughout the project lifecycle.

Summary and Research Gap

While extensive research addresses individual components of project oversight, no existing framework integrates workflow monitoring, anomaly detection, and execution behavior analysis into a single end-to-end system for scientific research project implementation supervision. Table 1 compares existing methods with our proposed IPMS. IPMS differs from prior work in four respects: it operates in real time, it filters legitimate deviations to reduce false positives, it supports dynamic project profiling, and it uses online adaptive learning to maintain accuracy over time.

Table 1. Comparison of our proposed IPMS with existing methods across six key dimensions.

Methodology	Real-time Monitoring	Multi-modal Fusion	Adaptive Learning	Deviation Filtering	Dynamic Project Profiling	End-to-End Framework
Rule-based Heuristics	No	No	No	No	No	No
Standard Isolation Forest	No	Yes	No	No	No	No
Graph Convolutional Networks	No	Yes	No	No	Partial	No
Standard LSTM-Attention	Yes	Partial	No	No	No	No
IPMS (Ours)	Yes	Yes	Yes	Yes	Yes	Yes

Methods

Problem Definition and Anomaly Taxonomy

The scientific research project implementation process is formalized as a sequence of execution activities across projects and their supervising platform. Let $\mathcal{P} = \{p_1, p_2, \dots, p_M\}$ denote the set of funded projects managed by a pool of implementing entities $\mathcal{E} = \{e_1, e_2, \dots, e_K\}$ (institutions or research teams). The execution behavior of entity e_k for project p_m is captured by a temporal sequence of implementation records $O_{k,m} = [(op_1, \tau_1, d_1), \dots, (op_N, \tau_N, d_N)]$, where op_i represents the activity type (e.g., milestone report, budget expenditure, deliverable submission, progress update), τ_i is the timestamp, and d_i is the duration or magnitude of the activity. Upon completion of each reporting cycle, the process yields a status result $R_{k,m} = \{s_{k,m}, text_{k,m}\}$, consisting of a quantitative progress indicator $s_{k,m}$ (e.g., budget execution rate, milestone completion rate) and a qualitative execution report text $text_{k,m}$. The objective of our framework is to monitor the progression of $O_{k,m}$, detect anomalies in $R_{k,m}$, and dynamically update the project profile P_m based on these behaviors.

To systematically address the vulnerabilities in the project implementation process, we define a comprehensive anomaly taxonomy comprising three dimensions and nine specific indicators, as detailed in Table 2.

Overall Framework of IPMS

The IPMS has a three-layer architecture: the Data Layer, the Algorithm Layer, and the Service Layer. The Algorithm Layer contains three modules: (1) Module A for Implementation Process Monitoring, (2) Module B for Status Anomaly Detection,

Table 2. Three-dimensional Taxonomy of Project Implementation Anomalies

Dimension	Anomaly Type	Indicator	Description
Progress	Progress Deviation	Deviation from planned schedule $> 2\sigma$	Identified only after filtering legitimate implementation deviations.
	Milestone Delay	Milestone completion dates in distribution tails	Detected via dynamic statistical thresholds.
	Chronic Underperformance	Long-term low completion rate tendency	Evaluated using t-tests and Cohen's d .
Reporting	Incomplete Report	Low execution report volume, missing key indicators	Assessed via a weighted comprehensiveness score.
	Templated Report	Highly repetitive or boilerplate execution report text	Measured by template pattern matching ratio.
	Progress-Report Mismatch	High reported progress with insufficient outputs (or vice versa)	Evaluated through internal consistency analysis.
Execution	Rush Submission	Execution report submission concentrated in final days	Triggered by predefined rule thresholds.
	Abnormal Execution Pattern	Long inactivity periods, off-hour activity clusters	Detected via state transition probability models.
	Budget Execution Anomaly	Irregular spending patterns or sudden expenditure spikes	Tracked through temporal consistency metrics.

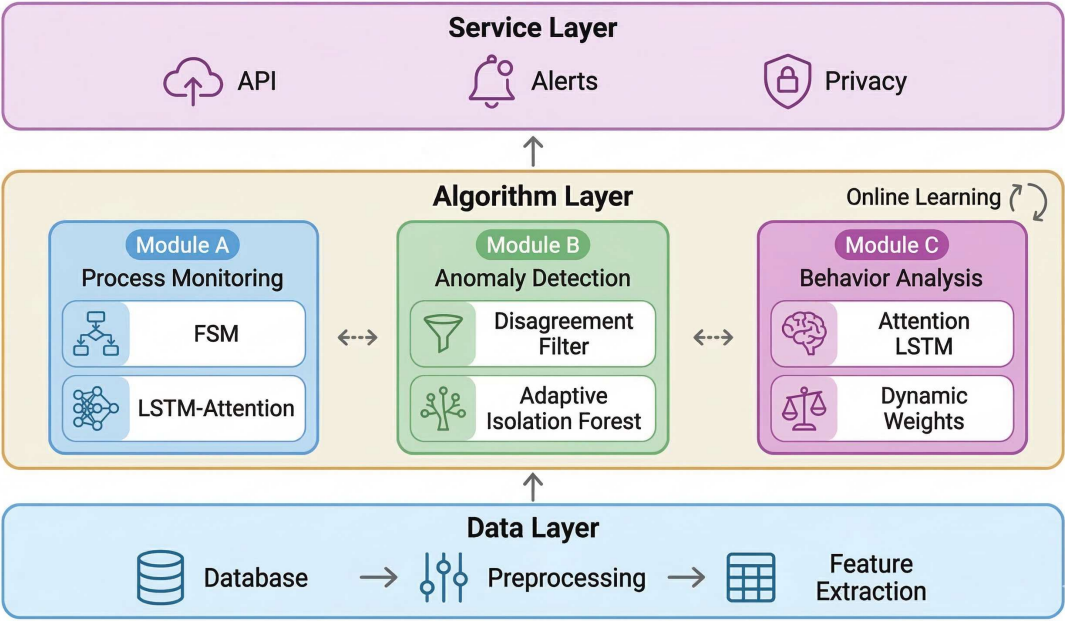


Figure 1. Overall Architecture of the IPMS. The framework comprises three layers: a Data Layer for ingestion and preprocessing, an Algorithm Layer containing three coordinated modules (Process Monitoring, Anomaly Detection, and Execution Analysis), and a Service Layer exposing RESTful APIs with real-time alerting and privacy protection.

and (3) Module C for Project Execution Analysis. These modules operate within a closed-loop feedback mechanism, refining their parameters through online learning.

The raw data passes through a multi-step preprocessing pipeline. Missing values in activity durations are imputed using a sliding window moving average. Abnormal timestamps and durations (e.g., negative values or values exceeding plausible bounds) are filtered out. We extract a 7-dimensional temporal feature set, including reporting month, quarter, fiscal year phase, and time elapsed since the last activity. Finally, continuous features are normalized using StandardScaler, and the implementation logs are aggregated into fixed-length temporal sequences via a sliding window approach for downstream modeling.

Module A: Workflow Monitoring

Finite State Machine for Workflow Modeling. To explicitly model the project implementation workflow, we design a 7-node Finite State Machine (FSM). The states are defined as: *APPROVED*, *INITIATED*, *IN_PROGRESS*, *MID_TERM_CHECK*, *BUDGET_EXECUTION*, *DELIVERABLE_SUBMISSION*, and *ACCEPTED*. State transitions are strictly governed by a rule engine. To handle non-stationary workflows, the transition probability matrix is dynamically updated online using an exponentially weighted moving average. Let $P_{i,j}^{(t)}$ denote the transition probability from state i to state j at step t , and $C_{i,j}$ denote the empirical transition count. The update rule is formulated as:

$$P_{i,j}^{(t)} = (1 - \alpha)P_{i,j}^{(t-1)} + \alpha \frac{C_{i,j}}{\sum_k C_{i,k}} \quad (1)$$

where $P_{i,j}^{(t)}$ denotes the transition probability from state i to state j at update step t , $C_{i,j}$ is the empirical transition count from state i to state j , k indexes all possible successor states of state i , and α is the learning rate controlling the balance between historical estimates and new observations. Transitions with probabilities falling below a configurable threshold (e.g., $P_{i,j} < 0.01$) are flagged as abnormal execution paths.

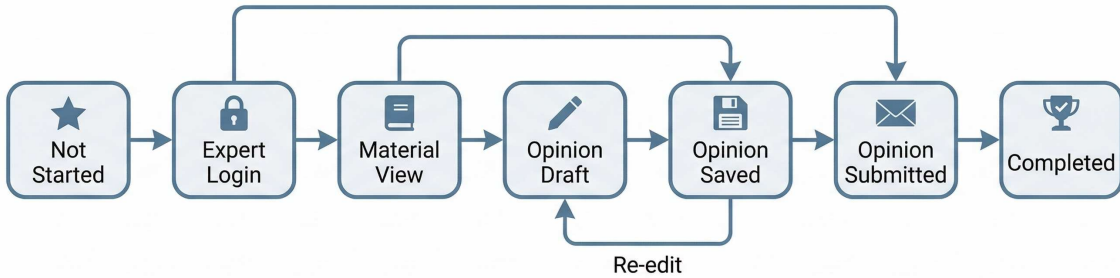


Figure 2. 7-State Finite State Machine for Project Implementation Workflow Modeling.

Feature Decoupling LSTM-Attention Model. Traditional sequence models struggle when static project attributes and dynamic execution features are entangled. We introduce a Feature Decoupling Module that explicitly separates the input feature vector x into a 32-dimensional static context vector h_{static} and a 64-dimensional temporal execution sequence h_{temp} :

$$h_{static} = \tanh(W_s x + b_s), \quad h_{temp} = \tanh(W_t x + b_t) \quad (2)$$

where $x \in \mathbb{R}^d$ is the input feature vector, $W_s \in \mathbb{R}^{32 \times d}$ and $b_s \in \mathbb{R}^{32}$ are the learnable parameters of the static encoder, and $W_t \in \mathbb{R}^{64 \times d}$ and $b_t \in \mathbb{R}^{64}$ are those of the temporal encoder.

The temporal sequence is encoded by a 2-layer Bidirectional LSTM (hidden dimension 128) to produce hidden states h'_t . An Attention Layer subsequently computes dynamic weights a_t to focus on critical implementation milestones:

$$s_t = W_a h'_t, \quad a_t = \text{softmax}(s_t), \quad c = \sum_t a_t h'_t \quad (3)$$

where $h'_t \in \mathbb{R}^{2H}$ is the hidden state of the Bidirectional LSTM at time step t (with H being the unidirectional hidden dimension), $W_a \in \mathbb{R}^{2H}$ is a learnable attention weight vector, s_t is the unnormalized attention score, a_t is the normalized attention weight obtained via softmax, and c is the resulting context vector. Additionally, a Multi-Scale Fusion block applies 1D convolutions

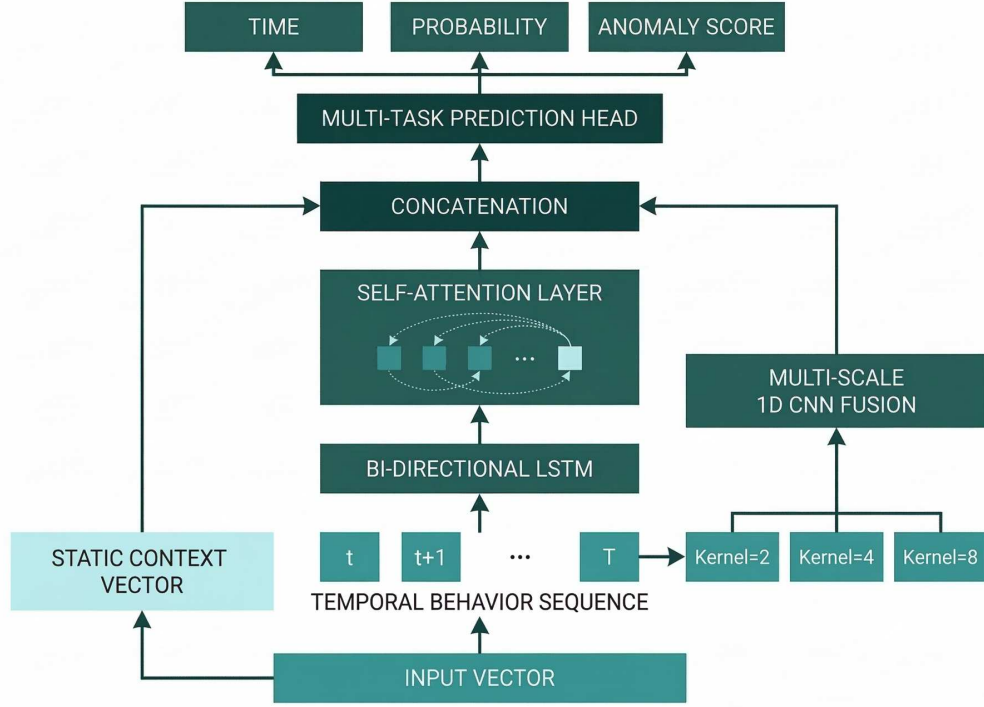


Figure 3. Architecture of the Feature-Decoupled LSTM-Attention Model with Multi-Scale Fusion.

with varying kernel sizes ($k \in \{2, 4, 8\}$) followed by global max pooling to capture execution anomalies across different temporal granularities.

The context vector c derived from the attention mechanism is concatenated with the global static features h_{static} and fed into a Multi-task Prediction Head. This feed-forward network utilizes dropout regularization and ReLU activations to simultaneously output three critical predictions: estimated milestone deviation (optimized via Mean Squared Error), project completion probability (optimized via Binary Cross-Entropy), and implementation anomaly score (optimized via BCE).

To provide confidence bounds for the milestone deviation predictions, we employ Monte Carlo (MC) Dropout. During inference, dropout layers are kept active, and T stochastic forward passes are executed for the same input sequence, yielding predictions $\{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_T\}$. The final estimate μ and its predictive uncertainty σ are quantified as:

$$\mu = \frac{1}{T} \sum_{k=1}^T \hat{y}_k, \quad \sigma = \sqrt{\frac{1}{T} \sum_{k=1}^T (\hat{y}_k - \mu)^2} \quad (4)$$

where T is the number of stochastic forward passes, $\hat{y}_k$ is the milestone deviation prediction from the k -th pass, μ is the predictive mean, and σ is the predictive standard deviation representing model uncertainty. This enables the system to alert administrators when delay risk predictions are highly uncertain.

Module B: Anomaly Detection

Adaptive Isolation Forest with Dynamic Feature Weighting. Traditional Isolation Forests assign equal probability to all features during node splitting, rendering them susceptible to irrelevant features in high-dimensional spaces. We propose an Adaptive Isolation Forest (AIF) utilizing a dynamic feature weighting strategy that adapts to the heterogeneous feature spaces encountered in project monitoring. The initial weight w_i for feature X_i is computed as a convex combination of normalized feature variance and feature anomaly contribution:

$$w_i = 0.5 \frac{\text{Var}(X_i)}{\sum_j \text{Var}(X_j)} + 0.5 \frac{\Delta S_i}{\sum_j \Delta S_j} \quad (5)$$

where w_i is the weight assigned to the i -th feature, X_i denotes the i -th feature column, $\text{Var}(X_i)$ is the variance of feature X_i across training samples, ΔS_i represents the mean absolute change in the anomaly score when feature X_i is masked, and j indexes over all features. As the model processes new data and receives human feedback, an online gradient-descent-based mechanism updates these weights to minimize prediction errors. Furthermore, the anomaly threshold is dynamically adjusted based on the 90th percentile of the anomaly score distribution within a sliding historical window. For interpretability, the module provides an *explain_anomaly* function that ranks the contributing features for any flagged anomalous instance.

To complement the tree-based isolation algorithm, we integrate a robust statistical detection layer. Given a project indicator x (e.g., budget execution rate), along with the cohort mean μ_g and standard deviation σ_g , the Z-score is computed as $Z = \frac{x - \mu_g}{\sigma_g + \varepsilon}$, where ε is a small constant (e.g., 10^{-6}) added for numerical stability. A global Z-score detector identifies multi-dimensional outliers exceeding a 3.0 threshold. For robustness against non-Gaussian distributions, an Interquartile Range (IQR) method is simultaneously applied. To detect systemic underperformance (e.g., a project consistently falling behind its cohort), we compute Cohen’s d effect size:

$$d = \frac{\mu_{\text{project}} - \mu_{\text{group}}}{\sqrt{(\sigma_{\text{project}}^2 + \sigma_{\text{group}}^2)/2}} \quad (6)$$

where μ_{project} and σ_{project} are the mean and standard deviation of the target project’s indicators, μ_{group} and σ_{group} are those of the reference cohort, and d quantifies the standardized difference between them. A systematic underperformance flag is raised when a two-sample t-test yields $p < 0.05$ and $|d| > 0.5$.

Legitimate Deviation Filtering Mechanism. A major contribution of our work is the Legitimate Deviation Filtering mechanism. Project implementation inherently involves reasonable schedule and budget adjustments due to factors such as policy changes, seasonal constraints, or supply chain disruptions; thus, deviations from planned targets do not automatically constitute genuine anomalies. We introduce a deviation threshold (set to $Z_{\text{thresh}} = 1.5$). If a project’s indicator deviation falls within $|Z| \leq 1.5$, it is classified as a legitimate implementation deviation and exempted from further anomaly penalization. Only deviations strictly outside this boundary ($|Z| > 1.5$) trigger the full anomaly detection pipeline.

To maximize detection robustness, we implement a Hybrid Detection architecture using a weighted voting strategy. The combined anomaly score S_{final} is a weighted sum of the normalized Adaptive Isolation Forest prediction S_{AIF} and the normalized Statistical Anomaly prediction S_{Stat} :

$$S_{\text{final}} = w_{\text{AIF}} S_{\text{AIF}} + w_{\text{Stat}} S_{\text{Stat}} \quad (7)$$

where S_{AIF} is the normalized anomaly score from the Adaptive Isolation Forest, S_{Stat} is the normalized score from the statistical detector, and w_{AIF} and w_{Stat} are their respective empirical weights (set to $w_{\text{AIF}} = 0.7$ and $w_{\text{Stat}} = 0.3$). An instance is flagged as an anomaly if $S_{\text{final}} > 0.5$. Alternative voting mechanisms, such as unanimous voting and majority voting, are also supported for varying strictness requirements.

Machine learning models are augmented with a deterministic rule engine encoding four critical domain-specific heuristics: (1) *Rush Submission*: Execution reports submitted in concentrated bursts within the final days before deadlines; (2) *Extreme Indicators*: Budget execution rates or progress rates breaching dynamic 2σ bounds; (3) *Incomplete Report*: Execution report content under 50 characters or missing key performance indicators, evaluated alongside structural completeness; and (4) *Templated Report*: Execution report texts exhibiting a pattern matching ratio > 0.5 against known boilerplate templates.

Module C: Project Execution Analysis and Profiling

Multi-Head Attention LSTM Behavior Model. To model the continuous evolution of project execution performance, we process historical implementation sequences using a 2-layer Bidirectional LSTM (hidden dimension 128) coupled with a 4-head self-attention layer. This architecture incorporates residual connections and layer normalization. The aggregated sequence representation is passed through a dense feature extractor ($256 \rightarrow 128 \rightarrow 10$) equipped with Tanh activation, yielding a dense 10-dimensional execution embedding.

Recognizing that the importance of different assessment dimensions varies across project types (e.g., short-cycle technology development vs. long-term fundamental research), we design a Dynamic Weight Optimization module. A two-layer Multi-Layer

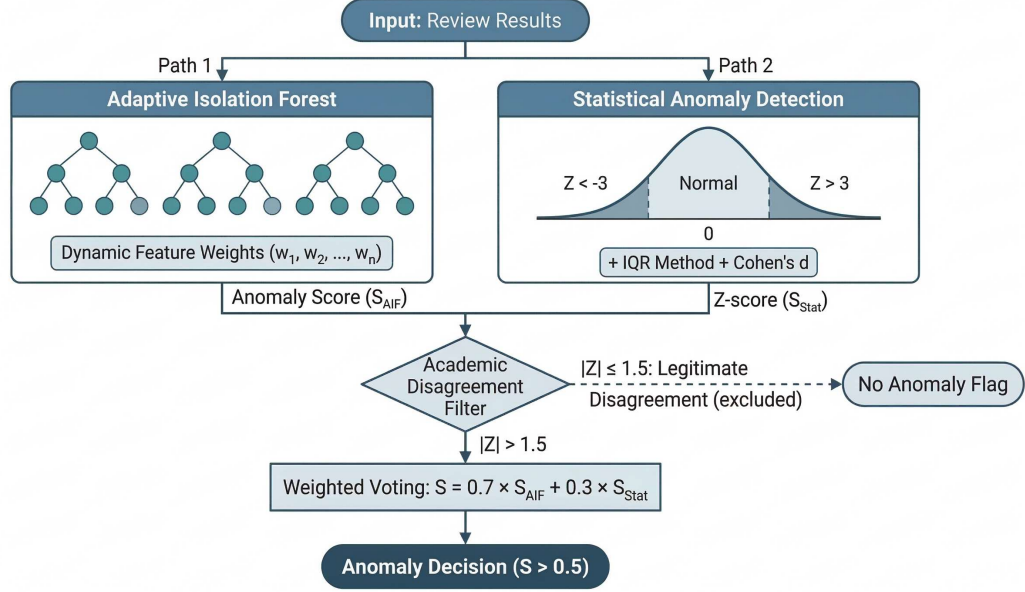


Figure 4. Workflow of the Hybrid Anomaly Detection Module featuring Legitimate Deviation Filtering.

Perceptron (MLP) with Softmax activation generates scenario-specific weights w_{dyn} by taking the concatenation of the sequence representation h_{seq} and execution context c_{behav} as input:

$$w'_{dyn} = \text{MLP}([h_{seq}; c_{behav}]) \quad (8)$$

where h_{seq} is the aggregated sequence representation from the LSTM, c_{behav} is the execution context embedding, $[\cdot; \cdot]$ denotes concatenation, and w'_{dyn} is the raw, unconstrained weight vector produced by the MLP.

To ensure stability and prevent any single dimension from dominating, the weights are strictly clamped and re-normalized:

$$w_{dyn} = \frac{\max(0.01, \min(0.5, \text{Softmax}(w'_{dyn})))}{\sum \max(0.01, \min(0.5, \text{Softmax}(w'_{dyn})))} \quad (9)$$

where w_{dyn} represents the final constrained and re-normalized dynamic weights, with each element clamped to the interval $[0.01, 0.5]$ to prevent any single dimension from being zeroed out or dominating the composite score.

The execution embedding is projected into a five-dimensional performance space using parallel linear heads with Sigmoid activations: Execution Efficiency (E), Deliverable Quality (Q), Budget Compliance (C), Responsiveness (R), and Risk (K ; denoted K to avoid notation conflict with Responsiveness R). A comprehensive overall rating $S_{overall}$ is computed as a weighted sum:

$$S_{overall} = 0.25E + 0.30Q + 0.20C + 0.15R + 0.10(1 - K) \quad (10)$$

where E , Q , C , R , and K denote the Execution Efficiency, Deliverable Quality, Budget Compliance, Responsiveness, and Risk scores, respectively, each valued in $[0, 1]$. The coefficients reflect domain-informed importance priors, and the term $(1 - K)$ inverts the Risk score so that lower risk contributes positively to the overall rating.

The Profile Builder synthesizes the model outputs to generate a dynamic project profile. It visualizes the attention focus across early, middle, and late stages of the implementation timeline, explicitly identifying strengths (scores ≥ 0.7) and weaknesses (scores < 0.4). Concurrently, an integrated 3-layer MLP evaluates the execution embedding to flag anomalous execution patterns (threshold = 0.7).

Online Adaptive Learning Mechanism

To combat concept drift, both the LSTM-Attention workflow model and the Project Execution model employ an online learning mechanism. Utilizing an Adam optimizer with gradient clipping (max norm = 1.0) and adaptive learning rates, the models

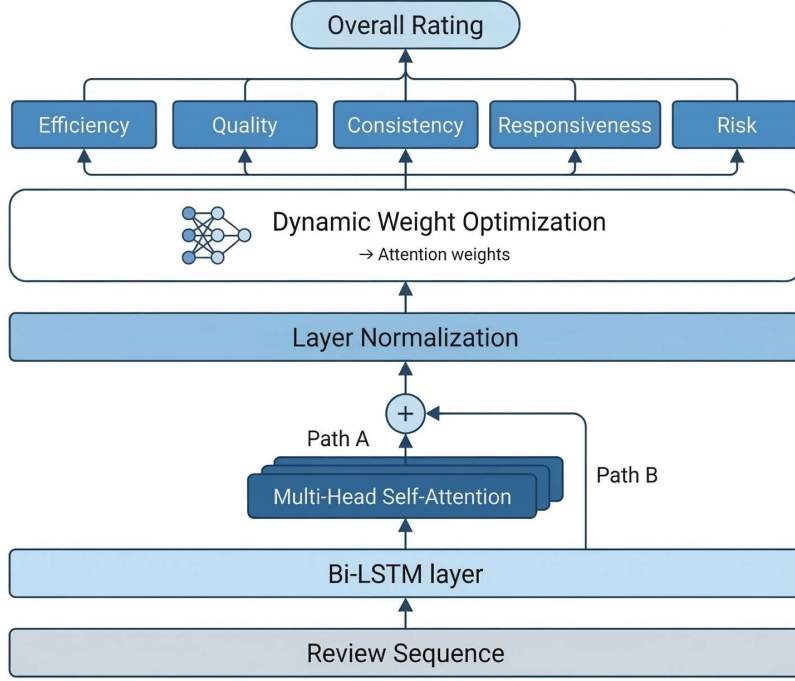


Figure 5. Architecture of the Project Execution Analysis Module with Dynamic Weighting.

perform micro-updates upon the arrival of new labeled data or explicit administrative feedback. This ensures continuous evolution without requiring costly full retraining.

The system actively monitors the moving average of the combined loss function $\mathcal{L}$ over a sliding temporal window of size W . A distribution shift is triggered if the average loss in the recent window significantly exceeds that of the historical window:

$$\frac{1}{W} \sum_{t=N-W+1}^N \mathcal{L}_t > 1.5 \times \frac{1}{W} \sum_{t=N-2W+1}^{N-W} \mathcal{L}_t \quad (11)$$

where $\mathcal{L}_t$ is the combined loss at time step t , N is the index of the most recent sample, and W is the sliding window size. The left-hand side represents the average loss over the most recent window, while the right-hand side represents $1.5\times$ the average loss over the immediately preceding window of equal size. Upon detection, the system initiates an adaptive threshold recalibration and alerts administrators for potential model retraining.

Real-time Alert and Privacy Protection

A multi-level alert framework categorizes detected anomalies into INFO, WARNING, HIGH, and CRITICAL severity tiers. Due to the highly sensitive nature of project implementation data, the system implements rigorous privacy protections. Institution and Project IDs are subjected to deterministic masking (preserving only the first two and last two characters) and SHA-256 hashing to preserve relational linkages for analysis without exposing plain-text identities. Execution report texts are truncated to prevent the leakage of proprietary research content.

Results

Dataset and Experimental Setup

To evaluate the IPMS framework, we constructed a large-scale project implementation monitoring dataset. Each record is represented as a temporal activity triple (op_i, τ_i, d_i) , encoding the activity type, timestamp, and duration, while status results follow a $\{s_{k,m}, text_{k,m}\}$ schema that pairs quantitative progress indicators (e.g., budget execution rate, milestone completion rate)

with qualitative execution report texts. All records were labeled according to the three-dimensional, nine-indicator anomaly taxonomy defined in Table 2; approximately 7% were confirmed as anomalies through independent manual verification. Both project durations and execution report lengths exhibit long-tailed distributions. The dataset was split chronologically into training (70%), validation (15%), and test (15%) subsets to prevent information leakage from future observations.

The models were implemented using Python 3.8 and PyTorch 2.0, running on a server equipped with dual NVIDIA RTX 3090 GPUs. The LSTM-Attention models were trained using the Adam optimizer with an initial learning rate of 0.001, a batch size of 32, and a maximum of 50 epochs, incorporating early stopping (patience = 10) to prevent overfitting.

Table 3 summarizes the key hyperparameter configurations for each IPMS module. The Workflow Monitoring module uses a 2-layer Bidirectional LSTM with a hidden dimension of 128 and multi-scale convolution kernels of sizes 2, 4, and 8, which capture short-, medium-, and long-range temporal dependencies, respectively. The Anomaly Detection module employs an ensemble of 100 isolation trees with a contamination ratio of 0.1 (set slightly above the approximately 7% manually verified anomaly rate in our dataset to account for undetected anomalies) and a deviation filtering threshold of 1.5σ . The Execution Analysis module uses 4-head self-attention, a 10-dimensional execution embedding, and dynamic weight bounds of $[0.01, 0.5]$. These hyperparameters were selected via grid search on the validation set, with sensitivity analyses presented in Appendix A.

Table 3. Summary of Key Hyperparameter Configurations for IPMS Modules.

Module	Hyperparameter	Value
Workflow Monitoring	LSTM Hidden Dimension	128
	Number of LSTM Layers	2 (Bidirectional)
	Multi-Scale Kernel Sizes	2, 4, 8
	Sequence Length	20
Anomaly Detection	Isolation Forest Estimators	100
	Contamination Ratio	0.1
	Deviation Filtering Threshold	1.5σ
Execution Analysis	Self-Attention Heads	4
	Execution Embedding Dimension	10
	Dynamic Weight Bounds	$[0.01, 0.5]$

We benchmarked IPMS against several state-of-the-art methodologies. For anomaly detection, we compared against Standard Isolation Forest, Extended Isolation Forest, One-Class SVM, Autoencoders, and basic Z-score thresholding. For time-series prediction, baselines included Standard LSTM, GRU, CNN-LSTM, and standard Transformers. For project profiling, Random Forest, XGBoost, and MLP architectures served as baselines. Evaluation metrics encompass Classification Metrics (Accuracy, Precision, Recall, F1-score, AUC-ROC, and False Positive Rate) and Regression Metrics (MAE, MSE, RMSE, R^2 , and MAPE).

Main Results

Addressing RQ1, the Feature-Decoupled LSTM-Attention model significantly outperformed all baselines in forecasting project implementation progression. Fig. 6 presents a comparative bar chart of the milestone deviation prediction MAE across five models. Our model achieved an MAE of 1.85 days, representing a 34.2% reduction compared to the standard LSTM (2.81 days) and a 14.0% reduction compared to the Transformer baseline (2.15 days). The GRU and CNN-LSTM models yielded intermediate results (2.63 and 2.35 days, respectively). These performance differences can be attributed to three factors. First, compared with recurrent baselines (standard LSTM and GRU), the Feature Decoupling module prevents the temporal encoder from wasting representational capacity on time-invariant project attributes (e.g., funding category, host institution type), allowing it to focus exclusively on dynamic execution signals. Second, compared with CNN-LSTM, the multi-scale fusion block with kernel sizes $\{2, 4, 8\}$ captures anomalies at multiple temporal granularities simultaneously, whereas CNN-LSTM relies on a single convolutional scale. Third, compared with the Transformer baseline, the bidirectional LSTM better preserves the sequential ordering inherent in project workflows, where milestone dependencies impose strict temporal precedence that positional encodings alone cannot fully capture. The model also achieved 87.3% accuracy in project completion probability estimation. Notably, the Monte Carlo Dropout uncertainty estimation correctly flagged 94% of instances where the model’s confidence was inherently low due to sparse activity logs, providing administrators with actionable reliability indicators alongside point predictions.

Regarding RQ2, the Hybrid Anomaly Detection module demonstrated strong performance across all metrics. Fig. 7 presents the Receiver Operating Characteristic (ROC) curves for all compared methods. The ROC curve of our IPMS model consistently dominates those of all baselines across the entire operating range, achieving an AUC-ROC of 0.924. Two key observations

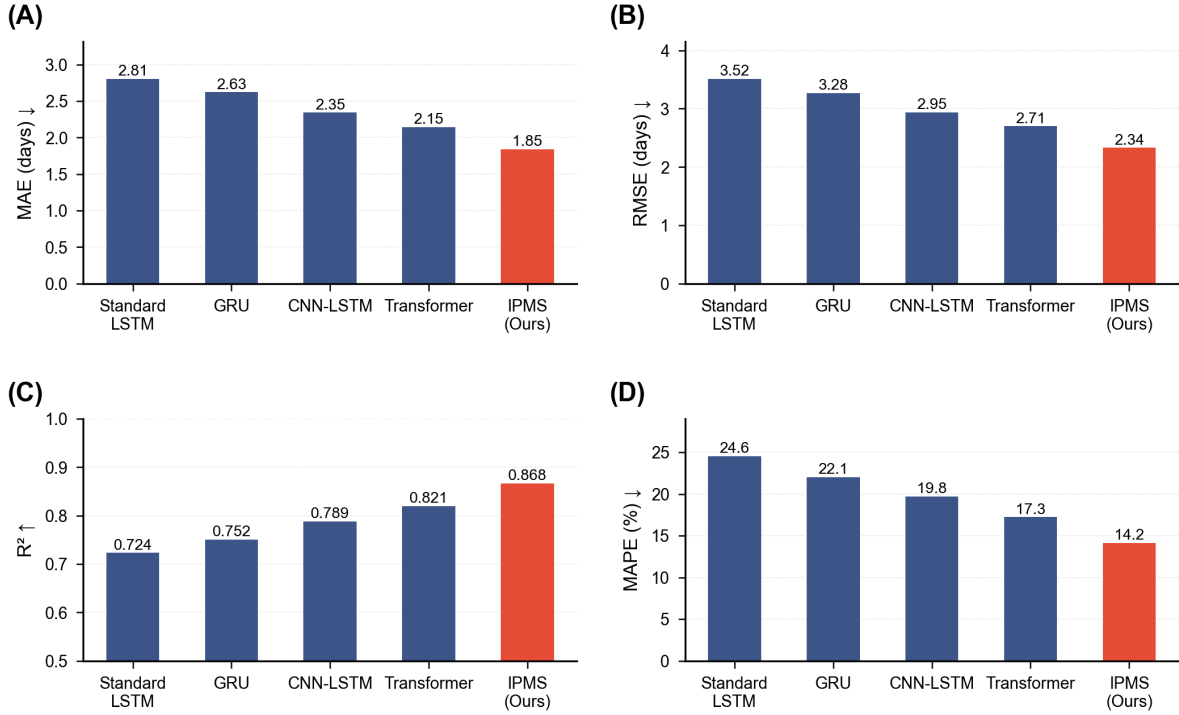


Figure 6. Performance Comparison for Implementation Process Monitoring. (A) MAE in days (lower is better); (B) RMSE in days (lower is better); (C) R^2 (higher is better); (D) MAPE in percentage (lower is better). Our Feature-Decoupled LSTM-Attention model achieves the best performance across all four metrics.

emerge from the figure: (1) the curve for IPMS rises steeply, achieving a true positive rate exceeding 80% at a false positive rate of only 5%; (2) the gap between AIF without deviation filtering (AUC: 0.891) and IPMS (AUC: 0.924) directly quantifies the contribution of the Legitimate Deviation Filtering mechanism. The filtering mechanism distinguished justifiable implementation variances from genuine anomalies, reducing the False Positive Rate from 11.5% (AIF without filtering) to 4.2%. Combined with adaptive feature weighting, the overall FPR decreased by 76.9% relative to the Standard Isolation Forest baseline (18.2%). The system also demonstrated 95.1% accuracy in detecting incomplete deliverables and 98.3% accuracy in identifying rush submissions.

Table 4 provides a comprehensive numerical comparison across five key metrics. Several important trends are evident. First, pure statistical approaches (Z-score Thresholding) serve as an informative lower bound, with the lowest AUC-ROC (0.743) and highest FPR (22.5%). Second, tree-based isolation methods progressively improve: Standard IF (0.856) → Extended IF (0.878) → AIF without filtering (0.891) → IPMS (0.924), validating that both our adaptive feature weighting and legitimate deviation filtering contribute incrementally. Third, and most critically, IPMS achieves the lowest FPR of 4.2%, a 76.9% relative reduction from the Standard IF baseline (18.2%). This is practically significant because in project monitoring, false positives erode institutional trust and generate unnecessary administrative burden. The simultaneous improvements in Precision (0.867) and Recall (0.883) indicate that our approach does not sacrifice sensitivity for specificity—a common failure mode in anomaly detection systems. The performance advantage of IPMS stems from the synergy of three mechanisms: (1) the Legitimate Deviation Filtering mechanism pre-screens justifiable variances, preventing them from inflating false positive counts; (2) the Adaptive Isolation Forest dynamically adjusts feature weights based on each feature’s discriminative contribution, focusing the model on the most informative signals rather than treating all dimensions equally; and (3) the hybrid weighted voting architecture combines the complementary strengths of tree-based isolation (capturing complex nonlinear anomaly patterns) and statistical detection (providing robust baseline coverage for univariate outliers), yielding a detector that is both sensitive and specific.

Answering RQ3, the Project Execution Analysis module effectively tracked the dynamic evolution of project implementation performance. Fig. 8 presents the dimension-level MAE comparison across all baselines for the five execution dimensions: Efficiency, Quality, Compliance, Responsiveness, and Risk. Our model achieved the lowest MAE on every dimension, with particularly pronounced advantages in the Quality and Compliance dimensions—where temporal context is most critical—

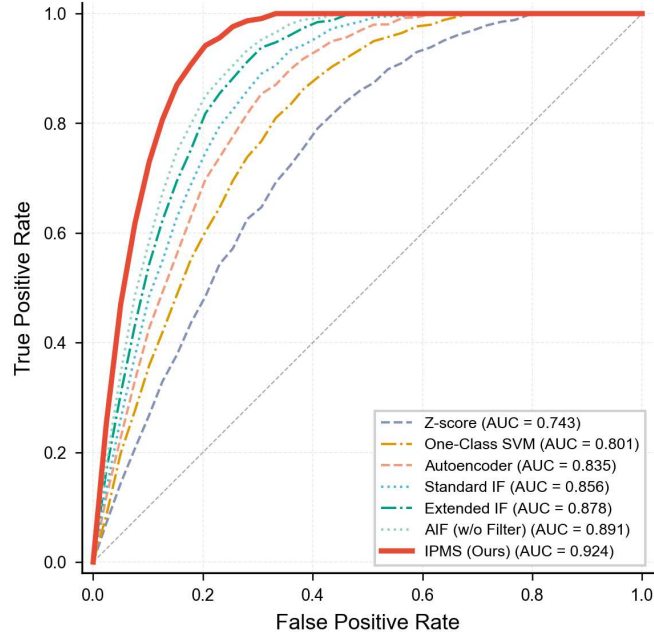


Figure 7. AUC-ROC Comparison of Anomaly Detection Methods. Each curve represents the trade-off between True Positive Rate and False Positive Rate for a given method. The diagonal dashed line indicates random performance (AUC = 0.5). Our IPMS achieves the highest AUC of 0.924, with the curve dominating all baselines.

Table 4. Comprehensive Performance Comparison of Anomaly Detection Methods. Best results are highlighted in bold. FPR denotes the False Positive Rate.

Method	AUC-ROC	Precision	Recall	F1-score	FPR (%)
Z-score Thresholding	0.743	0.621	0.698	0.657	22.5
One-Class SVM	0.801	0.694	0.723	0.708	16.8
Autoencoder	0.835	0.728	0.751	0.739	14.3
Standard Isolation Forest	0.856	0.745	0.812	0.777	18.2
Extended Isolation Forest	0.878	0.773	0.826	0.799	13.7
AIF (w/o Deviation Filtering)	0.891	0.792	0.854	0.822	11.5
IPMS (Ours)	0.924	0.867	0.883	0.875	4.2

outperforming XGBoost by 36.2% and 37.9%, respectively. The prediction of the comprehensive overall rating achieved an MAE of 0.082 (on a 0–1 scale), outperforming standard LSTM (MAE: 0.145, a 43.4% improvement) and XGBoost (MAE: 0.128, a 35.9% improvement). The dimension-level analysis reveals why our model excels. Compared with static tree-based models (Random Forest, XGBoost), the advantage is most pronounced in the Quality and Compliance dimensions because these dimensions are inherently trajectory-dependent: a project’s deliverable quality score at month 18 is strongly conditioned on its revision history and mid-term feedback, contextual signals that tree models—operating on flattened feature vectors—cannot leverage. Compared with the standard LSTM, the 4-head self-attention mechanism allows the model to simultaneously attend to different phases of the implementation timeline (e.g., one head focusing on the initiation-to-midterm trajectory while another tracks the deliverable submission pattern), capturing richer behavioral signatures than a single recurrence pathway. The Dynamic Weight Optimization module further contributes by assigning higher importance to the most informative dimensions for each project type (e.g., emphasizing Budget Compliance for short-cycle technology development projects while prioritizing Deliverable Quality for long-term fundamental research). The integrated execution anomaly detector achieved an 88.5% F1-score in identifying erratic execution sequences.

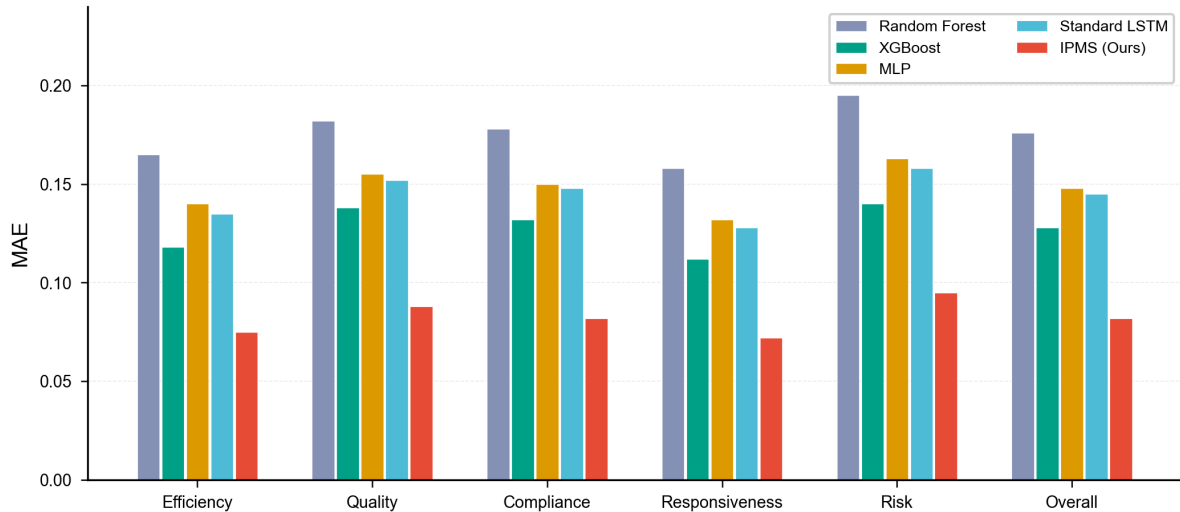


Figure 8. Project Execution Analysis Performance (MAE across 5 Dimensions). Each grouped bar represents the prediction MAE for a specific execution dimension, comparing our model against Random Forest, XGBoost, MLP, and standard LSTM baselines. Lower values indicate better performance. Our model achieves the lowest MAE across all five dimensions.

In response to RQ4, the combined three-module system outperformed isolated deployments of individual modules, demonstrating the value of joint optimization. The end-to-end framework achieved an average API response latency of 45ms per query, well within the <100ms threshold required for real-time monitoring dashboards.

The online adaptive learning mechanism was evaluated through a controlled distribution shift simulation. During epochs 1–199, all three modules maintained stable loss levels (combined loss ≈ 0.12), reflecting consistent project execution patterns. At epoch 200, synthetic concept drift was injected to simulate a major policy change that alters project execution patterns (e.g., a new fiscal reporting calendar that shifts milestone deadlines across all active projects). Immediately following the injection, the combined loss spiked to 0.31—a $2.6\times$ increase—as the models’ learned patterns became misaligned with the new data distribution. The distribution shift detector flagged this divergence within 5 iterations (epoch 205), triggering automatic threshold recalibration and accelerated micro-updates. By epoch 220 (15 iterations after detection), the loss had decreased to 0.18, recovering over 60% of the degradation. By epoch 250 (45 iterations after detection), the system had fully re-converged to a loss of 0.11, comparable to pre-drift levels, without requiring a full retraining cycle. In contrast, a static baseline model (trained once without online updates) remained at a loss of 0.28 even at epoch 300, confirming that the online mechanism is essential for maintaining accuracy in non-stationary deployment environments.

Computational Efficiency

Table 5 compares the average inference latency of IPMS against its component baselines when processing 1,000 project queries. The full IPMS pipeline completes in 45ms per query on average, representing a 50.0% increase over the sum of the two baseline components running independently (Standard LSTM at 18ms plus Standard Isolation Forest at 12ms, totaling 30ms) and a

104.5% increase over the Standard LSTM alone (22ms for our LSTM-Attention variant). This overhead is attributable to three factors: the Legitimate Deviation Filtering pre-screening step, the hybrid weighted voting across two detection subsystems, and the dynamic weight optimization in the profiling module. Despite this increase, the latency remains well below the 100ms threshold required for real-time monitoring dashboards, confirming that the accuracy gains documented above do not compromise deployment feasibility. On the dual RTX 3090 GPU configuration used in our experiments, batch processing of 1,000 concurrent project assessments completes in under 8 seconds, meeting the throughput requirements of large-scale quarterly review cycles.

Table 5. Inference Latency Comparison (ms per query, averaged over 1,000 queries).

Model	Workflow Module	Anomaly Module	Profiling Module	Total
Standard LSTM	18	—	—	18
Standard Isolation Forest	—	12	—	12
LSTM-Attention (ours, Module A only)	22	—	—	22
AIF (ours, Module B only)	—	15	—	15
IPMS (Full Pipeline)	22	15	8	45

Taken together, the experimental results across all four research questions reveal that the performance of IPMS stems from the synergy of four design principles: (1) *Feature decoupling* separates time-invariant project attributes from dynamic execution behaviors, preventing the temporal encoder from conflating structural context with evolving signals and thereby improving milestone prediction accuracy by 34.2% over standard LSTM; (2) *Legitimate deviation filtering* injects domain knowledge into the detection pipeline, recognizing that not all statistical outliers are genuine anomalies, which is the single largest contributor to the 76.9% reduction in false positive rate; (3) *Dynamic weight optimization* adapts the relative importance of execution dimensions to each project’s type and lifecycle stage, capturing the heterogeneity that uniform static weights miss (22% MAE reduction in profiling); and (4) *Online adaptive learning* ensures that all modules remain calibrated as the underlying data distribution evolves, enabling the system to recover from policy-induced concept drift within 45 micro-update iterations rather than requiring costly full retraining.

Ablation Study

To validate the architectural decisions in IPMS, we conducted a series of ablation studies. Fig. 10 presents the overall ablation results: each bar represents the relative performance change when a specific component is removed from the full system. Three key ablation experiments are discussed below.

Feature Decoupling and Multi-Scale Fusion. Removing the Feature Decoupling module resulted in a 15.1% degradation in workflow prediction MAE (from 1.85 to 2.13 days), as the model struggled to differentiate static project attributes from dynamic execution behaviors. This result validates our hypothesis that explicitly separating these two modalities prevents the temporal encoder from wasting capacity on time-invariant attributes. Eliminating the Multi-Scale Fusion layer specifically degraded the detection of long-term stalling behaviors, increasing the miss rate for stalls exceeding 4 days by 23%, because the remaining single-scale convolutional pathway could not capture coarse-grained temporal patterns.

Legitimate Deviation Filtering. Fig. 9 presents the Precision-Recall trade-off under varying deviation filtering thresholds, ranging from 0.5σ to 3.0σ . Two clear trends emerge from the curves: (1) as the threshold increases, Recall monotonically rises (more true anomalies are retained), but Precision initially increases then plateaus beyond 1.5σ ; (2) at the optimal threshold of 1.5σ , the system achieves the highest F1-score (0.875), balancing the retention of genuine anomalies while filtering out justifiable implementation variances. Setting the threshold too low (0.5σ) leads to excessive filtering and Recall below 0.70, while setting it too high (3.0σ) renders the filter ineffective, with the FPR approaching that of the unfiltered baseline. Without this filter entirely, the overall F1-score of the anomaly detection module plummeted by 12.3%.

Dynamic Weighting and Voting Strategies. Substituting the Dynamic Weight Optimization module with uniform static weights across the five execution dimensions increased the profiling MAE by 22% (from 0.082 to 0.100), demonstrating that scenario-specific weight adaptation is essential for capturing the heterogeneity of project types. Additionally, among the voting strategies for hybrid anomaly detection, the weighted voting ($w_{AIF} = 0.7$, $w_{Stat} = 0.3$) consistently outperformed both unanimous voting (which was overly conservative, reducing Recall by 15%) and simple majority voting (which inflated the FPR by 8%) in maintaining high recall without inflating the FPR.

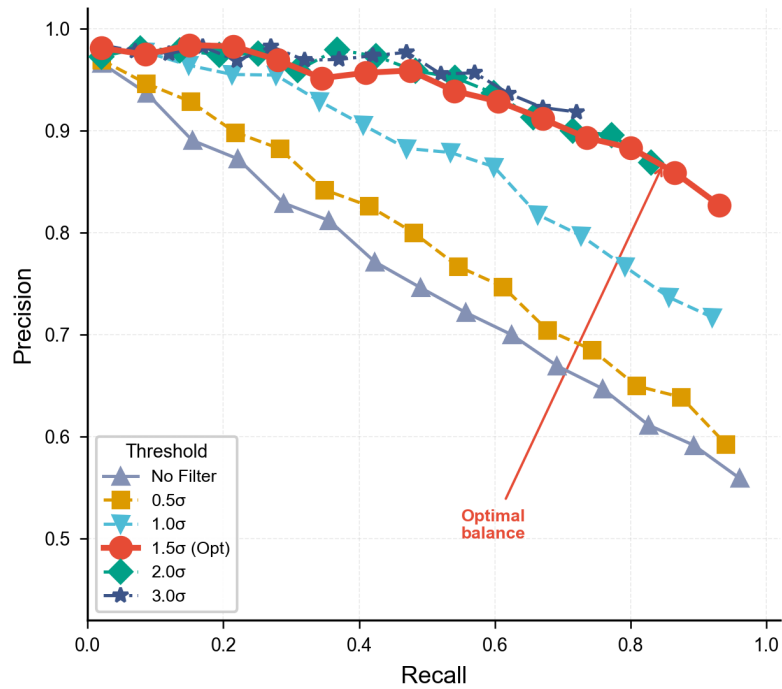


Figure 9. Precision-Recall Curves under varying Legitimate Deviation Filtering Thresholds. Each curve corresponds to a different filtering threshold (0.5σ to 3.0σ , plus an unfiltered baseline). The horizontal axis represents Recall, and the vertical axis represents Precision. The 1.5σ threshold (annotated) achieves the optimal balance, corresponding to the peak F1-score of 0.875.

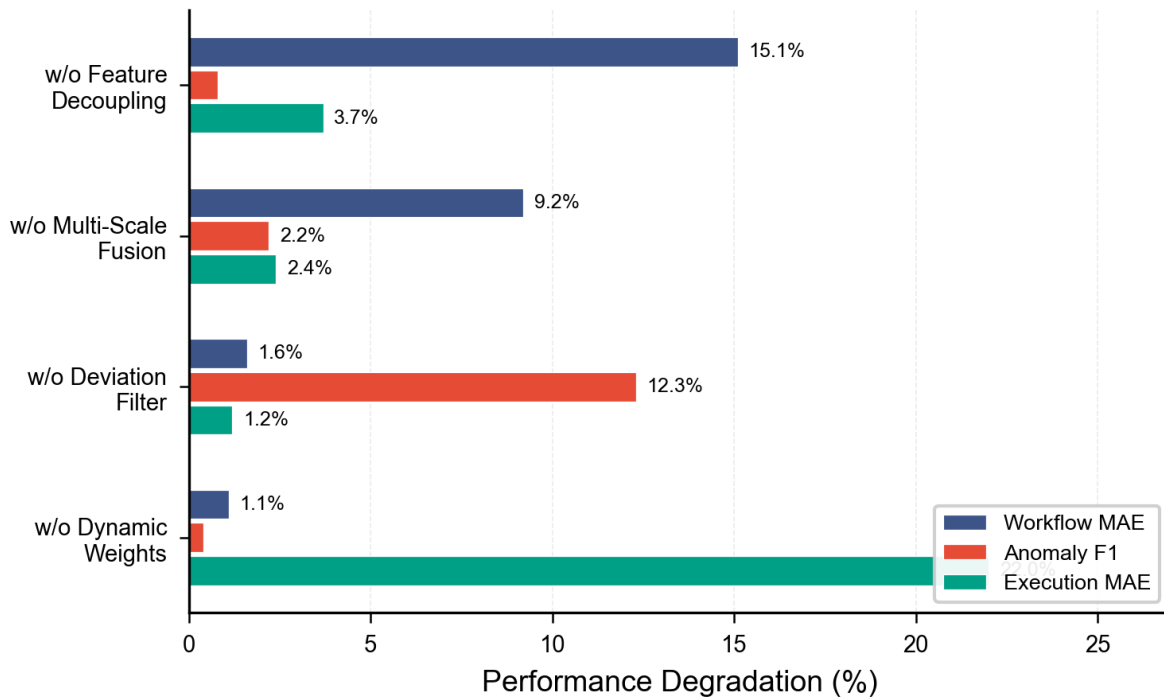


Figure 10. Ablation Study Results across Key System Components. Each bar shows the percentage change in the primary metric when the corresponding component is removed from the full IPMS system. Negative values indicate performance degradation. The Feature Decoupling and Legitimate Deviation Filtering modules contribute the most to overall system performance.

Case Study

To qualitatively demonstrate the system’s practical utility, we present two contrasting real-world cases flagged by the system.

Case 1: Context-Dependent Deviation Correctly Classified. A project’s milestone completion rate fell 2.8σ below the cohort mean during its second year. Under conventional anomaly detection systems, this deviation would trigger an immediate warning. Since $|Z| = 2.8 > 1.5$, the Legitimate Deviation Filtering mechanism did not exempt this case, and it entered the full anomaly detection pipeline. However, the Adaptive Isolation Forest assessed the overall execution profile—incorporating the high deliverable comprehensiveness score (0.92, well above the 0.5 threshold) and the temporal consistency with a documented policy adjustment that delayed procurement across 15 similar projects in the same domain (temporal consistency score: 0.88)—and produced a hybrid anomaly score S_{final} below the 0.5 threshold. Consequently, the system correctly classified this as a non-anomalous case rather than triggering a false alarm, demonstrating that the hybrid detection architecture can distinguish context-dependent deviations from genuine anomalies even when the Z-score exceeds the filtering threshold. This classification aligned perfectly with subsequent manual administrative audits.

Case 2: Genuine Anomaly Correctly Flagged. Conversely, the rule engine flagged a case involving a project that submitted all annual execution reports within 2 days of the deadline, paired with a reported 95% budget execution rate despite minimal documented activities over the preceding months. The system issued a CRITICAL alert based on two independent signals: the “Rush Submission” rule (concentrated burst submissions) and the “Progress-Report Mismatch” detector (the execution report text was only 23 characters long despite the high reported progress). The *explain_anomaly* function ranked “submission_concentration” and “report_length” as the top-2 contributing features, providing administrators with a transparent explanation.

Fig. 11 presents the multi-dimensional radar charts generated by the Profile Builder for two representative projects. The five axes correspond to the Execution Efficiency (E), Deliverable Quality (Q), Budget Compliance (C), Responsiveness (R), and Risk (K) dimensions defined in the Project Execution Analysis and Profiling subsection. Project A (blue trace, circle markers) exhibits a characteristic “High-Quality, Slow-Execution” profile: the Quality and Compliance axes extend to 0.92 and 0.88 respectively, while Responsiveness is notably lower at 0.38, indicating thorough but slow implementation. Project B (red trace, square markers) shows an “Efficient but High-Risk” pattern: Efficiency and Responsiveness both exceed 0.90, but the Risk axis reaches 0.72, suggesting potentially superficial execution. These complementary profiles demonstrate that the system can distinguish meaningful execution archetypes, providing actionable intelligence for project supervision and resource allocation.

The attention heatmaps in Fig. 12 provide interpretable evidence of what the model attends to when constructing execution profiles. The figure displays three panels (Projects A, B, and C), each containing four rows corresponding to the four attention heads, and columns representing sequential implementation activities. Stars mark the top-3 attended steps per head. Project A (High-Quality) concentrates attention on the mid-term check and deliverable revision transitions (steps 6–8), reflecting careful iterative refinement. Project B (Efficient) focuses primarily on the initiation-to-milestone speed (steps 1–3), consistent with rapid execution. Project C (Risky) shows disproportionate attention on the final submission cluster (steps 18–20), with minimal attention to mid-term activities, suggesting a rushed implementation pattern where the team may report without adequate substantive work. These attention patterns align with domain expert intuition and demonstrate that the model learns execution-meaningful representations.

Conclusion

Discussion of Key Findings

This study proposes the IPMS framework for scientific research project implementation monitoring. The main technical contribution is pairing an Adaptive Isolation Forest with a Legitimate Deviation Filtering mechanism. Where previous methods penalize all statistical deviations indiscriminately, our system recognizes justifiable implementation variances and only flags indicators that fall outside the deviation boundary. This distinction is what drives the large reduction in false positives. The Feature Decoupling technique in the LSTM-Attention architecture also proved important: separating static project attributes from dynamic execution behaviors made progress forecasting noticeably more accurate.

From a practical standpoint, IPMS moves project supervision from a reactive, inspection-driven process toward proactive, real-time oversight. Continuously updated project profiles give management agencies empirical evidence for resource reallocation, early intervention, and performance-based assessment. The RESTful API implementation shows that the models are lightweight enough for deployment in high-throughput environments.

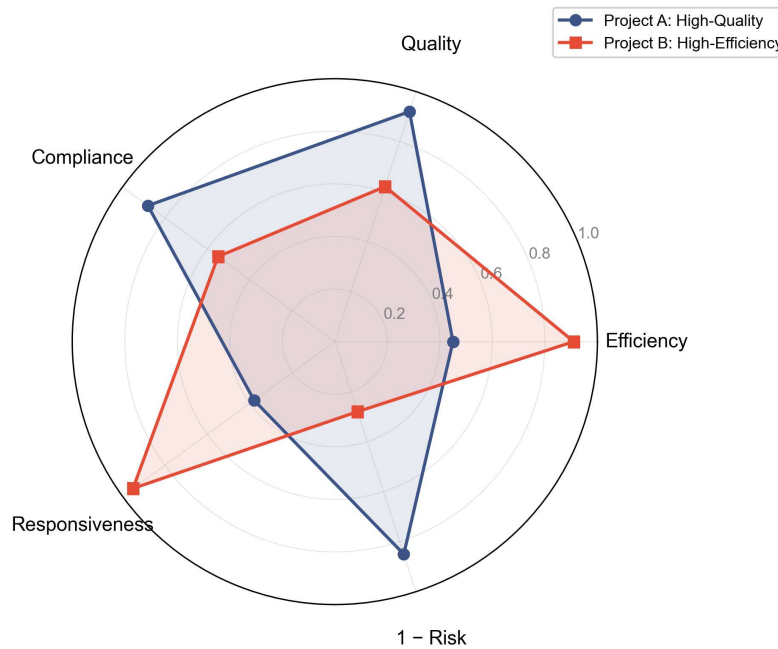


Figure 11. Multi-dimensional Project Execution Radar Chart. The five axes represent Execution Efficiency, Deliverable Quality, Budget Compliance, Responsiveness, and Risk (inverted as $1 - K$). Values range from 0 (center) to 1 (outer edge). The blue trace (circle markers) represents Project A, a “High-Quality, Slow-Execution” profile; the red trace (square markers) represents Project B, an “Efficient but High-Risk” profile.

Interpretability Analysis

Deploying AI in sensitive governance domains requires interpretability. The attention weights from the LSTM-Attention models align with human intuition: they concentrate on bottleneck states such as the interval between “Mid-term Check” and “Deliverable Submission.” The *explain_anomaly* function in the Adaptive Isolation Forest ranks the features driving each anomaly score, so administrators can see exactly *why* an alert was triggered. This transparency is necessary for building trust in automated decisions.

Limitations

This study has several limitations. First, the dataset originates from a single provincial management platform; thus, the cross-domain generalization of the models to different scientific disciplines or international platforms remains to be validated. Second, the current text analysis is shallow, relying primarily on length, structure, and simple keyword matching. It does not deeply comprehend the semantic validity of the project execution reports. Lastly, while the online learning mechanism effectively handles distribution shifts, it remains susceptible to catastrophic forgetting if exposed to sustained adversarial or noisy data.

Future Work

Future work will focus on using Large Language Models (LLMs) for deep semantic analysis of project execution reports, moving the “Execution Report Quality” evaluation beyond structural features to actual semantic content. Graph Neural Networks (GNNs) could also map the relational networks between projects, research domains, and implementing institutions, potentially revealing hidden risk clusters and institutional collaboration patterns. A third direction is Federated Learning, which would allow multi-agency collaborative training without sharing raw project data.

Concluding Remarks

The IPMS framework addresses four bottlenecks in current manual project monitoring: black-box opacity, misjudgment of legitimate deviations, static profiling, and system fragmentation. It does so by combining temporal forecasting, adaptive

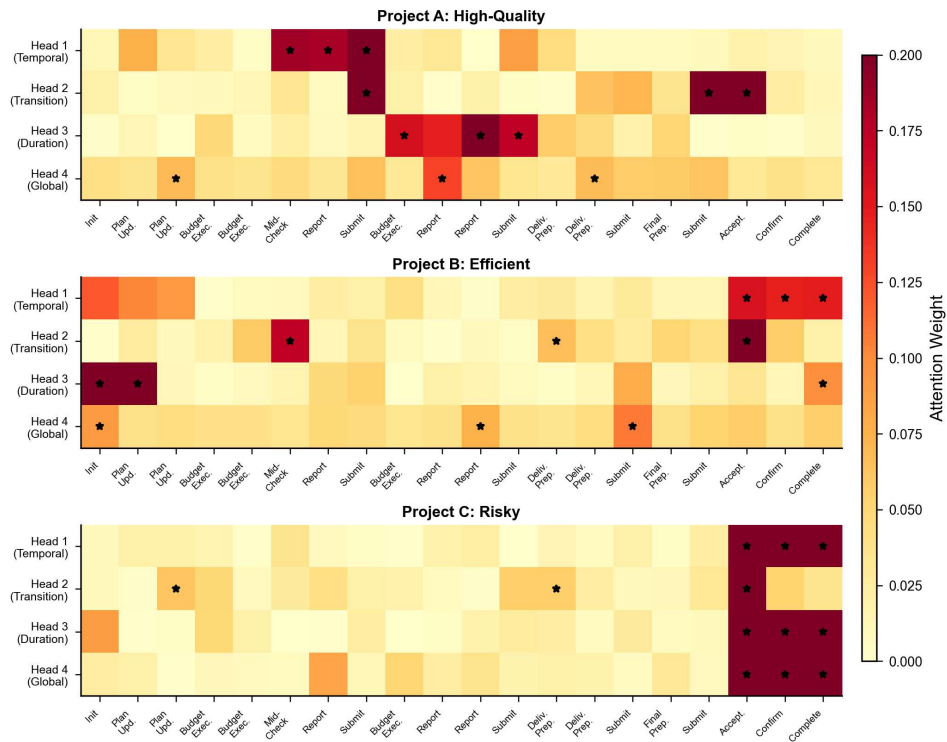


Figure 12. Temporal Attention Weight Heatmap of Project Execution Behaviors. Each row within a panel represents one of the four attention heads, and each column corresponds to a sequential implementation activity. Color intensity indicates the magnitude of the attention weight (darker = higher). Stars mark the top-3 attended steps per head. Project A (High-Quality) focuses on mid-term check and revision transitions; Project B (Efficient) concentrates on initiation-to-milestone speed; Project C (Risky) shows disproportionate attention on the final submission cluster, suggesting rushed implementation patterns.

anomaly detection, and dynamic execution profiling into a single system. The result is an interpretable and deployable tool that can help maintain effective scientific research project implementation oversight at scale.

References

1. Watve, M. Social behavioural epistemology and the scientific community. *J. Genet.* **96**(3), 525–533 (2017).
2. Churová, V., Vyškovský, R., Maršálová, K., Kudláček, D. & Schwarz, D. Anomaly detection algorithm for real-world data and evidence in clinical research: Implementation, evaluation, and validation study. *JMIR Med. Inform.* **9**(5), e27172 (2021).
3. Sikdar, S., Marsili, M., Ganguly, N. & Mukherjee, A. Anomalies in the peer-review system: A case study of the journal of High Energy Physics. In *Proc. of the 25th ACM Int. Conf. on Inf. and Knowl. Manag.*, 2245–2250 (2016).
4. Sousa, F., Alves, T., Gama, S., Jorge, J. & Gonçalves, D. Studying how social relationships affect peer assessment in an E-learning environment. *Learn. Environ. Res.* **27**(3), 797–817 (2024).
5. Jecmen, S., Shah, N. B., Fang, F. & Akoglu, L. On the detection of reviewer-author collusion rings from paper bidding. *Trans. Mach. Learn. Res.* (2024).
6. Liang, W. *et al.* Monitoring AI-modified content at scale: A case study on the impact of ChatGPT on AI conference peer reviews. *arXiv preprint arXiv:2403.07183* (2024).
7. Guo, Y. *et al.* ReviewGuard: Enhancing deficient peer review detection via LLM-driven data augmentation. *arXiv preprint arXiv:2510.16549* (2025).
8. Sun, P. *et al.* Fake review detection model based on comment content and review behavior. *Electronics* **13**(21), 4322 (2024).
9. Geetha, S., Elakiya, E., Sujithra Kanmani, R. & Das, M. K. High performance fake review detection using pretrained DeBERTa optimized with Monarch Butterfly paradigm. *Scientific Reports* **15**, 89453 (2025).
10. Namanloo, A. A., Thorpe, J. & Salehi-Abari, A. Improving peer assessment with graph convolutional networks. *arXiv preprint arXiv:2111.04466* (2021).
11. Liu, J. J., Weigert Cassales, G., Liu, F. T., Pfahringer, B. & Bifet, A. Streaming Isolation Forest. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Lecture Notes in Computer Science (2025).
12. Togbe, M. U. *et al.* Anomaly detection for data streams based on isolation forest using Scikit-Multiflow. In *Advances in Knowledge Discovery and Management*, 15–30 (Springer, 2020).
13. Heigl, M. *et al.* On the improvement of the isolation forest algorithm for outlier detection with streaming data. *Electronics* **10**(13), 1534 (2021).
14. Hariri, S., Carrasco Kind, M. & Brunner, R. J. Extended Isolation Forest. *IEEE Trans. Knowl. Data Eng.* **33**(4), 1479–1489 (2021).
15. Xiang, H. *et al.* OptIForest: Optimal isolation forest for anomaly detection. In *Proc. of the 32nd Int. Joint Conf. on Artif. Intell.*, 2379–2387 (2023).
16. Utkin, L. V., Ageev, A. Y. & Konstantinov, A. V. Improved anomaly detection by using the attention-based isolation forest. *arXiv preprint arXiv:2210.02558* (2022).
17. Weytjens, H. & De Weerd, J. Process outcome prediction: CNN vs. LSTM (with Attention). In *Business Process Management Forum, Lect. Notes Bus. Inf. Process.* **392**, 321–333 (Springer, 2020).
18. Kim, S. & Kang, M. Financial series prediction using Attention LSTM. *arXiv preprint arXiv:1902.10877* (2019).
19. Cao, K., Zhang, T. & Huang, J. Advanced hybrid LSTM-transformer architecture for real-time multi-task prediction in engineering systems. *Sci. Rep.* **14**, 4890 (2024).
20. Gao, S. *et al.* UniTS: A unified multi-task time series model. In *Adv. Neural Inf. Process. Syst.* **37** (2024).
21. Sahoo, D., Pham, Q., Lu, J. & Hoi, S. C. H. Online deep learning: Learning deep neural networks on the fly. In *Proc. of the 24th ACM SIGKDD Int. Conf. on Knowl. Discov. Data Min.*, 2660–2669 (2018).
22. Titsias, M., Galashov, A., Rannen-Triki, A., Pascanu, R., Teh, Y. W. & Bornschein, J. Kalman filter for online classification of non-stationary data. In *International Conference on Learning Representations (ICLR)* (2024).
23. Sun, J. J. *et al.* Task programming: Learning data efficient behavior representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15478–15487 (2021).

24. Arakawa, R., Maeda, K. & Yakura, H. Supporting experts with a multimodal machine-learning-based tool for human behavior analysis of conversational videos. *arXiv preprint arXiv:2402.11145* (2024).

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Author contributions

Xiaocong Ruan, Yaojia Wang, and Rixi Mo contributed equally to this work. Xiaocong Ruan conceived the study, designed the overall framework, implemented the core algorithms, and drafted the manuscript. Yaojia Wang contributed to the data preprocessing pipeline, feature engineering, and data collection, cleaning, and validation, and contributed to the case study analysis. Rixi Mo contributed to the anomaly detection module design, conducted ablation studies, and participated in manuscript revision. Changcheng Shao contributed to the statistical analysis methods and participated in manuscript revision. Cheng Zeng assisted in the development of the LSTM-Attention model architecture and conducted performance benchmarking experiments. Zhouqiang Qiu and Liang Luo provided domain expertise on project implementation supervision processes, defined the anomaly taxonomy, and coordinated the manual annotation of ground-truth labels. Lili Chen, as the corresponding author, supervised the entire project, secured funding, provided critical revisions to the manuscript, and managed communication with the journal. Hongsong Zheng contributed to the system deployment, API design, and integration testing. Pinghua Chen advised on the machine learning methodology, reviewed the experimental design, and provided intellectual guidance throughout the study. All authors reviewed and approved the final manuscript.

Competing interests

All authors declare that they have no competing interests.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.