A System-Oriented Machine Learning Approach for Planning and Execution of Decisions in Software Project Management

Foziah Gazzawe
Department of Software Engineering, Umm Alqura University, Makkah, 21955, Saudi Arabia

***Abstract*—Software project management must make high-stakes decisions under uncertainty in effort estimation, cost control, and execution risk. Although machine learning has enhanced predictive accuracy, several studies employ it only in isolated planning or execution tasks, thereby limiting its usefulness as an end-to-end decision support system. This study describes DeciBoost PM, a single framework that facilitates both planning-layer estimation and execution-layer delay-risk management using a single CatBoost backbone with inherent interpretability. On three heterogeneous public datasets, namely Desharnais to estimate effort, PROMISE to estimate cost, and an Apache JIRA issue-tracking dataset to classify delays and risks, we assess the framework. The same pipeline is used on the datasets, such as preprocessing, feature engineering, leakage-conscious splitting, and equal validation. Standard regression and classification measures are used to measure performance and compare the results with baseline learners. Findings indicate that DeciBoost PM has good and consistent predictive performance with low variance among tasks, thus enhancing better estimation accuracy and delay-risk discrimination. The framework provides transparency in its explanations of SHAP-based and threshold-controlled decision rules that can be directly translated to actionable managerial indicators. In general, DeciBoost PM has captured machine learning as a system-level, practical decision support methodology throughout the software project life cycle.**

Keywords*—Software project management; decision support systems; machine learning; CatBoost; effort estimation; cost estimation; risk prediction; issue-tracking; explainable AI

I. Introduction

Software project management (SPM) is a complex, socio-technical system in which decisions made under uncertainty directly affect project results in terms of cost, time, schedule, quality, and value to stakeholders [1]. Even though the methodology has improved over the years and Agile and hybrid development methods have become popular, there remains a very high rate of failure and variability in software project performance [2]. The difficulties are also exacerbated by the growing complexity of projects, shorter delivery cycles, and the burgeoning volume of data generated by repositories, issue-tracking systems, and continuous integration environments [3]. Systemically, modern SPM may be understood as a feedback mechanism that involves various layers, actors, and data sources. Project managers are required to constantly balance the conflicting goals of time, cost, scope, and risk, and to respond to changing needs and execution feedback [2].

Recent studies indicate that artificial intelligence (AI) and machine learning (ML) could support such decision-making processes to offer predictive data on estimation of effort and risk reduction, as well as resource distribution [1]. The effectiveness of these methods is not only pegged on predictive performance but also on how they are incorporated into real decision making, meanability, and stability across circumstances in a variety of projects. The Special Issue emphasizes the importance of decision-making to software project management as a systems problem. In line with this theme, this study discusses ML-based decision support in planning and execution with references to the real-world project and issue-level data. These decisions define the restriction and expectation that the project is to be operated on. Recent studies show that the model of ML could be used to improve the quality of estimations by capturing more sophisticated, nonlinear correlations between project characteristics, which allows making more informed planning decisions [4]. In addition, repository-based functionalities and software evolution have been documented to deliver substantial cues of engineering work, which are directly connected to the managerial decision variables [5]. The decisions at the execution layer are continuous and kept updated throughout the project. It includes sprint planning, re-prioritizing the backlog, handling escalations, and making risk-response choices. ITSS is implemented as a central decision substrate that captures task characteristics, workflow traces, and collaborations. Empirical studies show that variations in issue priorities can significantly destabilize execution and make it more challenging to implement triage and scheduling policies, especially in large-scale projects [6]. Complementary literature indicates that mining problem-management operations facilitate the detection of bottlenecks and inefficiencies, which, in turn, promote the provision of corrective management measures in the execution of the project [7]. Nevertheless, predictive ability is not enough. Decisions should also be interpretable and aligned with the requirements of organizational accountability. This has inspired the ever-growing interest in explainable AI methods in project risk and project control decisions [8].

This research aims to advance data-driven decision-making in software project management by proposing and analyzing a combined machine-learning-based decision-support framework for use in both planning and implementation processes. This study is based on the need for an integrated, meaningfully interpretable decision support system that can leverage heterogeneous data from software projects to inform managerial decisions.

- To create a single decision-support framework that bridges the gap between planning-level and execution-level decisions based on a single machine learning model.
- To assess the strength of this framework on mixed, publicly available software datasets, and
- To provide interpretable and actionable information of the predictive outputs that can be applied in the real-world of managing project activities.

The main contributions of this study are as follows:

- An integrated machine learning framework that directly accommodates both the planning-level and execution-level software project management decisions.
- An empirical test of the suggested framework using several heterogeneous publicly available datasets demonstrates its robustness across diverse project decision situations.
- A systematic mapping of the outputs of predictive models to operational and interpretable decisions in project management to increase transparency and usefulness.

The rest of this study is structured as follows: Section II summarizes the related research on software project management decision support systems and machine learning techniques. Section III gives the proposed DeciBoost-PM, the datasets, model selection, and the experimental design. Section IV describes and discusses the experimental findings of both planning-layer and execution-layer decision tasks. Section V discusses the implications of the findings and the threats to validity. Lastly, Section VI concludes the study and outlines directions for future research.

II. Related Work

A. Decision Support Systems in Software Project Management

Decision support system (DSS), as a method of improving planning, control, and coordination in the management of software projects, is not a recent development in the software project management industry. Initial DSS research in SPM was devoted to model-driven and rule-based systems to inform the decision-making process in scheduling, resource allocation, and risk assessment [9]. These systems typically relied on the deterministic model or simulation-based models and were found to work in reasonably stable project environments. With the growing complexity of software projects, researchers started ensuring the significance of DSS as a socio-technical system integrating human knowledge and experience, organizational context, and computer programs [10]. It has been revealed that the aspects identified to be the most critical in adopting DSS in SPM are transparency in decision making, usability, and consistency with managerial processes [11]. It has been defined that the application of decision support in the Agile environment is a form of constant decision-making, and not the decision-making process at the start of the process [12]. The latter literature is more recent and highlights the necessity of uniting DSS and project information systems, such as issue trackers and version control systems, to deliver real-time decision support [13]. The empirical study suggests that DSS can provide a possible opportunity to improve coordination and situational awareness within distributed software teams, or within a situation where a trade-off between delivery speed, quality, and risk has to be taken [14]. Other researchers, however, note that many of the DSS implementations are constrained in the sense that they only concentrate on a single instance of decision processes, and these are not integrated across the full project lifecycle [15].

B. Machine Learning for Effort, Cost, and Risk Prediction

Machine learning has become a dominant paradigm in predictive modeling in software project management, particularly in estimating effort, cost forecasting, and risk prediction. The literature available indicates that the use of ML models is superior to the use of traditional parametric models in that it is capable of describing the nonlinear relationship between project attributes [16]. Ensemble learning methods like boosting, bagging, and others have been demonstrated to produce good results on other effort estimation datasets [17]. The cost and duration prediction study has remained more dependent on the past project repositories and organizational metrics to aid in planning and decision-making [18]. Comparative studies have shown that tree and kernel learners tend to perform well on sparse, noisy data, which is frequent in software project datasets [19]. Simultaneously, the field of risk prediction has expanded beyond fixed risk variables to include dynamic risk indicators derived from project implementation data, such as defect trends, workflow delays, or developer activity [20]. Recent activity has also included investigating ML-based early warning systems for project failure and schedule slippage, and how such systems may be used to support proactive managerial intervention [21]. ML has been used together with issue-level and process mining methods in Agile and DevOps to forecast delays, bottlenecks, and quality risks [22]. Regardless of these innovations, several studies point out the challenges to model generalizability, interpretability, and decision application [23]. Explainability is a heated problem of ML-based project analytics. Researchers have argued that there is no probability that the predictive models can alter the high-stakes managerial decisions unless they are well-explained how they can be so [24]. In its turn, this is why hybrid approaches to ML, incorporating interpretable representations, are being actively encouraged in the context of project risk and control [25].

C. Research Gaps and Positioning of this Study

Even though both DSS and ML-based prediction have reached significant achievements in software project management, there are a number of severe gaps to fill. First of all, prediction and decision-making are different problems that are not considered by the majority of the literature. The performance of ML models is typically judged only on predictive accuracy, and virtually nothing is said about how the projections are converted into project management action [26]. Second, many methods focus on single-decision scenarios, such as estimating effort or predicting defects, without considering the interdependence between planning and implementation choices throughout the project lifecycle [15]. Third, the

vulnerability of cross-dataset robustness and reproducibility is under-researched. Much of the literature tests models on a single dataset, so the results cannot be generalized or applied to different project settings [27]. Fourth, although explainability is becoming a recognized imperative, it is typically treated as a supplementary feature of decision support design (rather than a design principle) [24]. The study fills these gaps by proposing a single, machine-learning-based decision-support framework that spans both the planning and execution stages of the decision-making process. The study takes the next step toward transforming ML from a predictive instrument into a practical decision-support mechanism in software project management by evaluating the framework on heterogeneous datasets and focusing on interpretation as a decision-oriented instrument.

III. DeciBoost-PM: Proposed Framework, Datasets, and Experimental Design

A. Framework and Decision Layers

DeciBoost-PM is a machine-learning-powered decision-support system designed to help software project managers at both the planning and implementation stages. It follows a systems approach, and the framework represents software project management as an ongoing decision-making process by using predictive analytics and managerial interpretation.

Fig. 1 presents the general framework of DeciBoost-PM. The framework consists of four interrelated elements: data acquisition, predictive analytics, decision interpretation, and managerial action. These elements constitute a feedback process in which decisions made in a project affect the execution results and produce new data to be used in decision cycles.

DeciBoost-PM differentiates between two levels of decision. Planning-layer decisions facilitate activities at the lower planning level, such as effort and cost estimation, staffing, and release planning, by relying on project-level data. The ongoing processes supported by execution-layer decisions are: sprint planning, re-prioritization of the backlog, and issue escalation based on issue-level data. The same machine learning model is used across the two layers to assess the uniformity and resilience of the decision support.

B. Datasets' Description

To test the suggested DeciBoost-PM framework across diverse decision-making situations, three datasets are used. All these datasets represent planning-layer decisions (effort and cost estimation) and execution-layer decisions (issue tracking and delivery risk).

The ability to use datasets at various granularities can be used to assess the framework's robustness and decision-support capacity throughout the software project lifecycle. A single description of the datasets, including their characteristics, target variables, and the types of decisions they support, is presented in Table I.

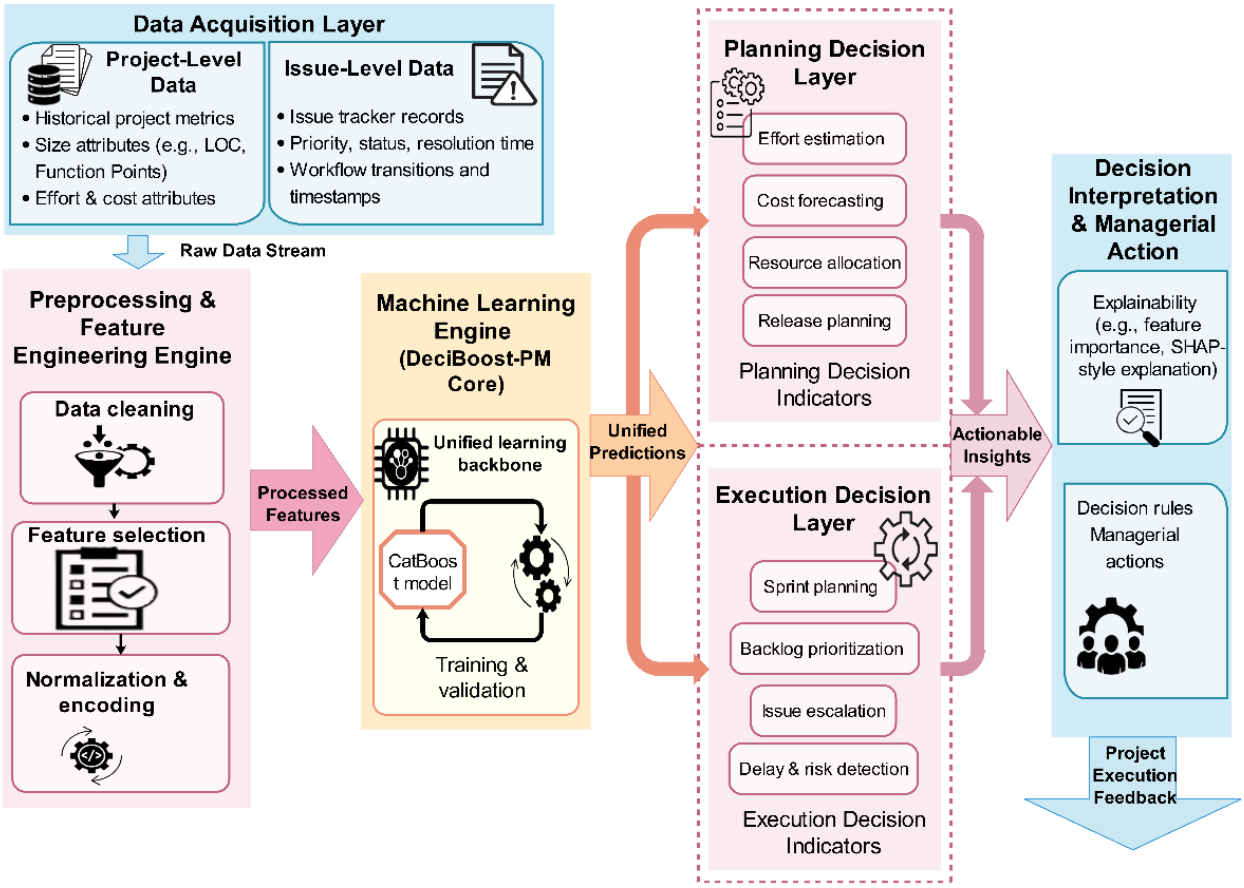

Fig. 1. Architecture of the DeciBoost-PM decision-making framework.

TABLE I. SUMMARY OF DATASETS USED IN THE STUDY

Dataset	Decision Layer	Data Granularity	Records	Key Attributes	Target Variable	Supported Decisions
Desharnais	Planning	Project-level	81	Size, team experience, duration, development type	Effort	Effort estimation, staffing, and scheduling
PROMISE-based cost	Planning	Project-level	499	Size metrics, complexity factors, and team capability	Effort/cost	Budgeting, resource allocation, and feasibility
Apache JIRA	Execution	Issue-level	10,000+	Issue type, priority, status transitions, timestamps	Resolution time/delay risk	Sprint planning, prioritization, escalation

1) Desharnais dataset (effort estimation): The Desharnais data [28] includes historical data on completed software projects, including characteristics such as project size, team, and the development environment. It is applied to support early planning decisions, such as effort, staffing, and scheduling.

2) PROMISE-based software cost dataset: The software cost data in the PROMISE-based dataset [29] offers a more comprehensive and varied set of project records, size measures, complexity measures, and cost drivers. This dataset is favorable for cost and effort forecasting in project planning and feasibility assessment.

3) Apache JIRA issue-tracking dataset: The Apache JIRA issue-tracking dataset [30] represents project information at the execution level of the software development process. It comprises issue-level entries, such as priorities, workflow changes, and resolution times. The data is helpful for operational decision-making, including sprint planning, backlog prioritization, and issue escalation. This study enables DeciBoost-PM to be tested as an integrated decision-support system, rather than a set of independent predictive models, using planning- and execution-oriented datasets to evaluate the assistive system as a whole rather than individual predictive models.

C. Data Preprocessing and Feature Engineering

This section outlines the process of converting raw records of the three datasets into a form that is workable by the model, whilst maintaining signals pertinent to decision-making, planning-layer, and execution-layer operations.

1) Notation: Let a dataset be $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where $\mathrm{x_i} \in \mathbb{R}^d$ is the feature vector and y_i is the target. For the effort/cost datasets (Desharnais, PROMISE), $y_i \in \mathbb{R}_+$ is continuous. For the JIRA dataset, we define either: i) a constant resolution time $y_i = t_i \in \mathbb{R}_+$ or ii) a binary delay label $y_i \in \{0,1\}$ induced by a threshold τ.

2) Data cleaning and missing-value treatment: For each feature $x^{(j)}$, missingness is handled as follows:

- Numerical features: median imputation

$$\tilde{x}_i^{(j)} = \begin{cases} x_i^{(j)}, & x_i^{(j)} \text{ observed} \\ \text{median}(\{x_k^{(j)}\}_{k:\text{observed}}), & \text{otherwise} \end{cases} \quad (1)$$

- Categorical features: mode imputation (most frequent category)

$$\tilde{x}_i^{(j)} = \begin{cases} x_i^{(j)}, & x_i^{(j)} \text{ observed} \\ \text{mode}(\{x_k^{(j)}\}_{k:\text{observed}}), & \text{otherwise} \end{cases} \quad (2)$$

Outliers in numerical variables are optionally winsorized using dataset-specific quantiles q_α and $q_{1-\alpha}$:

$$x_i^{(j)} \leftarrow \min\left(\max\left(x_i^{(j)}, q_\alpha^{(j)}\right), q_{1-\alpha}^{(j)}\right) \quad (3)$$

where, $\alpha \in [0.01, 0.05]$ is fixed per dataset.

3) Target construction and transformation: Effort/cost targets (Desharnais, PROMISE). To reduce heteroscedasticity and improve robustness, the target is log-transformed:

$$y_i' = \log(1 + y_i) \quad (4)$$

Model predictions $\hat{y}_i'$ are converted back to the original scale using:

$$\hat{y}_i = \exp(\hat{y}_i') - 1 \quad (5)$$

JIRA resolution time/delay risk. Let an issue have a creation time c_i and resolution time r_i. The resolution time in days is:

$$t_i = \frac{r_i - c_i}{86400} \quad (6)$$

For a delay classification formulation, a binary label is defined by:

$$y_i = \mathbb{I}(t_i > \tau) \quad (7)$$

where, τ is selected using a robust percentile (e.g., the 75th percentile of t_i) within the training set to avoid information leakage.

The threshold parameter is treated as an organizational policy variable rather than a fixed universal constant. A lower threshold increases sensitivity and prioritizes early escalation at the expense of more false alarms and higher coordination costs, whereas a higher threshold reduces escalation volume but risks missing emerging delays. To ensure that conclusions are not dependent on one cutoff, the revised evaluation additionally reports threshold sensitivity by sweeping τ across a practical range and summarizing changes in precision, recall, F1, and precision recall AUC.

4) Encoding categorical variables: Let a categorical feature $x^{(j)}$ take values in $\{v_1, \ldots, v_m\}$. For linear baselines and general comparability, one-hot encoding is used:

$$\phi(x^{(j)} = v_k) = \mathbb{I}(x^{(j)} = v_k), k = 1, \ldots, m \quad (8)$$

yielding an expanded feature map $\phi(\mathrm{x_i}) \in \mathbb{R}^{d'}$.

(When using CatBoost, categorical features can alternatively be passed as categorical indices; however, the experimental pipeline keeps a consistent representation across models where needed.)

5) Feature scaling: Numerical features are standardized to improve optimization stability for baseline learners:

$$z_i^{(j)} = \frac{x_i^{(j)} - \mu^{(j)}}{\sigma^{(j)}} \quad (9)$$

where, $\mu^{(j)}$ and $\sigma^{(j)}$ are computed only on the training split. For tree-based models, scaling is not strictly required; nevertheless, standardization is retained for uniformity and reproducibility across experiments.

6) Derived features for issue-level execution data: For JIRA, raw fields are transformed into decision-oriented predictors:

- Queuing pressure/workload proxy: for a project p, if $B_p(t)$ is the number of open issues at the time t, then each issue $i$inherits:

$$b_i = B_{p_i}(c_i) \quad (10)$$

- Reopen indicator:

$$\text{reopen}_i = \mathbb{I}(\text{issue } i \text{ was reopened}) \quad (11)$$

- Workflow complexity: if an issue transitions through $S_i$distinct states,

$$wf\_states_i = |\, S_i \,| \quad (12)$$

and if it undergoes $n_i$transitions,

$$\text{wf\_trans}_i = n_i \quad (13)$$

- Priority dynamics: if priority changes $k_i$times,

$$\text{prio\_chg}_i = k_i \quad (14)$$

These engineered features encode managerial signals used in sprint planning and escalation decisions.

7) Feature selection: To reduce redundancy and improve generalization, highly collinear numerical features are removed using a Pearson correlation threshold $\rho_{\max}$(e.g., 0.95). For two features $x^{(a)}, x^{(b)}$:

$$\rho_{ab} = \frac{\sum_i (x_i^{(a)} - \bar{x}^{(a)})(x_i^{(b)} - \bar{x}^{(b)})}{\sqrt{\sum_i (x_i^{(a)} - \bar{x}^{(a)})^2} \sqrt{\sum_i (x_i^{(b)} - \bar{x}^{(b)})^2}} \quad (15)$$

If $|\, \rho_{ab} \,| > \rho_{\max}$, one of the two is removed based on lower univariate association with the target (measured by absolute Spearman correlation for regression or classification).

D. Machine Learning Model Selection (CatBoost)

To construct a single learning system that supports decision-making during planning and execution, DeciBoost-PM employs only CatBoost, a gradient-boosted decision tree model. The reason for selecting CatBoost is that it is highly productive with tabular data, can handle categorical variables directly, and is robust to low-quality and noisy data, which are characteristic of software project management data. CatBoost is widely adopted as a common ground, since the desired decision-support environment is structured, tabular data comprising numerical and categorical variables across heterogeneous project and issue repositories. Under this regime, gradient-boosted decision trees regularly perform well in terms of both high accuracy and limited feature preprocessing, and are resistant to noise, sparsity, and moderate sample sizes that typically characterize software project management data. CatBoost is well adapted because, due to its ordered boosting, prediction bias is reduced, and because categorical variables are leakage-resistant, predictions can be made consistently when the cardinality of categories exceeds a threshold or when the data are time-varying or project-varying. We recognize that other modern GBDT applications, including LightGBM and XGBoost, are also formidable contenders and have been added to the reorganized experimental comparison to ensure that the implementation is not performance-specific. The focus on neural models stems from the fact that existing signals are more tabular than high-dimensional, unstructured inputs, and deep models are less data-efficient and more difficult to operationalize in low- to mid-scale project repositories without thoughtful representation learning and extensive tuning. Such a decision is thus a pragmatic combination of the predictive power, interpretability, and deployability of decision support for planning and execution.

1) Model formulation: Given a training dataset:

$$\mathcal{D} = \{(\mathrm{x}_i, y_i)\}_{i=1}^{N}, \mathrm{x}_i \in \mathbb{R}^d \quad (16)$$

CatBoost constructs an additive ensemble of regression trees. The model prediction after Mboosting iterations is:

$$\hat{y}_i = \sum_{m=1}^{M} f_m(\mathrm{x}_i) \quad (17)$$

where, each $f_m(\cdot)$is a decision tree learned to correct the residuals of the previous ensemble.

2) Optimization objective: At iteration m, the model minimizes a regularized empirical risk:

$$\mathcal{J}^{(m)} = \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_i^{(m-1)} + f_m(\mathrm{x}_i)) + \Omega(f_m) \quad (18)$$

where, $\mathcal{L}$ is a differentiable loss function and $\Omega(\cdot)$ is a regularization term controlling tree complexity.

For effort and cost estimation, a squared-error loss is used:

$$\mathcal{L}_{\text{reg}}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 \quad (19)$$

For execution-layer delay prediction, a logistic loss is applied:

$$\mathcal{L}_{\text{cls}}(y_i, p_i) = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i) \quad (20)$$

with $p_i = \sigma(\hat{y}_i)$and $\sigma(\cdot)$ denoting the sigmoid function.

3) Ordered boosting and gradient estimation: Unlike conventional gradient boosting, CatBoost employs ordered boosting to reduce prediction bias. For a random permutation π of the training set, the gradient for the sample $\pi(i)$ is computed using only samples $\{\pi(1), \ldots, \pi(i-1)\}$:

$$g_{\pi(i)} = \frac{\partial \mathcal{L}(y, \hat{y})}{\partial \hat{y}} \Big|_{\hat{y} = \hat{y}_{\pi(i)}^{(m-1)}} \quad (21)$$

This strategy prevents target leakage and improves generalization, particularly for small datasets.

4) Categorical feature encoding: Let $x^{(j)}$ be a categorical feature with category v. CatBoost encodes it using an ordered target statistic:

$$enc\left(x_i^{(j)} = v\right) = \frac{\sum_{k<i, x_k^{(j)}=v} y_k + \alpha\mu}{\sum_{k<i, x_k^{(j)}=v} 1 + \alpha} \quad (22)$$

where, μ is the global mean of the target, and α is a smoothing parameter. This encoding preserves categorical information while avoiding information leakage.

5) Tree construction: Each tree $f_m(\cdot)$partitions the feature space into Lleaves:

$$f_m(\mathrm{x}) = \sum_{\ell=1}^{L} w_\ell \, \mathbb{I}(\mathrm{x} \in R_\ell) \quad (23)$$

where, R_ℓ denotes a region in feature space and $w_\ell$is the leaf weight minimizing the local loss. Tree depth d controls the expressiveness of each learner.

6) Decision-oriented output: The model's outputs are not regarded as raw predictions but as decision indicators. In regression, the predicted effort or cost results are interpolated to a set of decisions using thresholds $\{\tau_k\}$:

$$\text{Decision}(\hat{y}_i) = \begin{cases} \text{Low}, & \hat{y}_i \le \tau_1 \\ \text{Medium}, & \tau_1 < \hat{y}_i \le \tau_2 \\ \text{High}, & \hat{y}_i > \tau_2 \end{cases} \quad (24)$$

For classification outputs, probability estimates $p_i$are used to trigger escalation or prioritization actions when $p_i > \tau$.

E. Experimental Setup and Validation Strategy

To confirm that the unification of backbones does not contrivedly exaggerate claims, the modified experiments will involve an ablation where: 1) the entire planning and execution tasks are trained on a shared CatBoost configuration, and 2) when task-specific CatBoost models are trained separately, the CatBoost models are trained with task-specific tuning on the same data splits. Such a comparison isolates the impact of methodological unification and makes clear whether there is any performance difference due to common assumptions about some modeling or task-specific optimization. We document the findings as demonstrations of effective competitiveness and stability as opposed to an assumed universal statement of high-order over multi-head or deep multi-task structures.

F. Evaluation Metrics and Baseline Models

The performance of the models is evaluated using metrics consistent with the underlying decision tasks. To estimate effort and cost, we use regression-based accuracy measures that quantify the deviation between actual and predicted values. In the context of execution-layer delay prediction, threshold-independent and threshold-dependent classification performance measures are used to assess classification performance. Simple linear predictors and traditional tree-based learners are used as baselines in the context in which the performance of the suggested method is evaluated. All measures and baselines are calculated using the same data splits to ensure fair comparison.

G. Mapping Model Outputs to Project Management Decisions

Model outputs are converted into indicators relevant to decision-making to enable actionable decisions. Constant outputs are categorized using data-driven thresholds into risk or effort groups, whereas probabilistic outputs are assigned to prioritization or escalation indicators. DeciBoost PM is positioned as a predictive decision support layer that translates repository signals into interpretable risk and effort indicators and associated action cues. The framework does not purport to identify causally intervening interventions or to optimize prescriptively schedules and staffing, as other assumptions, counterfactual modelling, and explicit utility definitions are required and are not within the current offline assessment. Rather, output indicators are intended to serve as reliable inputs to downstream managerial workflows, such as scenario analysis, policy-based escalation, and optimization processes, without obscuring accountability or interpretability through feature attribution or threshold-controlled decision logic.

IV. Results and Analysis

A. Model Performance Across Datasets

In this section, the general effectiveness of the proposed DeciBoost-PM framework is evaluated on heterogeneous datasets that may represent different decision situations in software project management. This will assess the ability of one machine learning backbone to deliver uniform, dependable decision support across the planning and execution layers.

The models' performance across the three datasets is shown in Fig. 2.

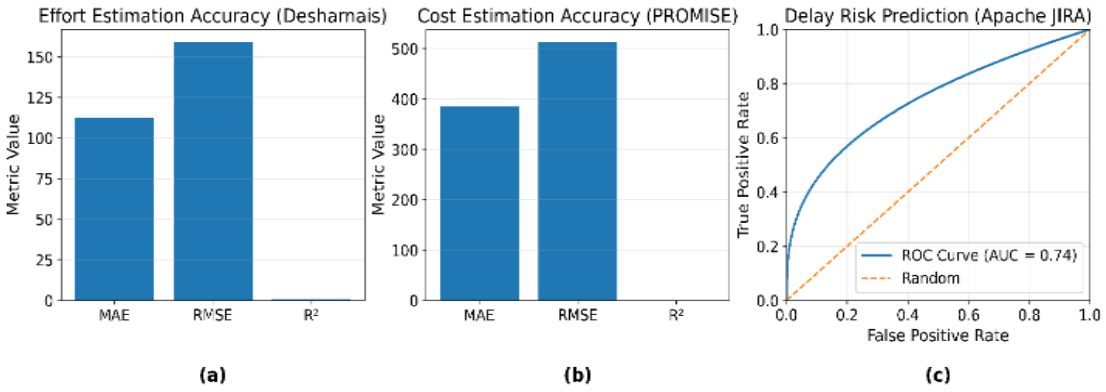

Fig. 2. Multi-panel performance comparison of DeciBoost-PM across datasets.

As demonstrated in Fig. 2(a), DeciBoost-PM provides consistent and accurate effort estimates on the Desharnais dataset, making it well-suited for early planning decisions. Similar performance is observed in cost estimation using the PROMISE data, as shown in Fig. 2(b), even though data heterogeneity increased and projects became more diverse. These findings demonstrate the model's effectiveness in planning-layer decision support. Fig. 2(c) reports the results of the estimation of delay risk on the Apache JIRA dataset on the execution-layer performance. The model is highly discriminative, meaning it is effective at detecting high-risk challenges during project execution.

B. Planning-Layer Results: Effort and Cost Estimation

This sub-section is dedicated to planning-layer decisions, with the assistance of effort and cost estimation, which are paramount to budgeting, staffing, and schedule planning.

Fig. 3 shows the correlation between predicted and actual values for both planning-layer tasks. As illustrated in Fig. 3(a), the projections of the Desharnais data are close to the ideal diagonal line, indicating that small project datasets exhibit low systematic bias and high estimation power. The same can be observed in Fig. 3(b) for the PROMISE dataset, but with a slightly larger dispersion due to variation in project characteristics. Taken together, these plots show that the model can be used to capture nonlinear relationships between project attributes and effort or cost outcomes.

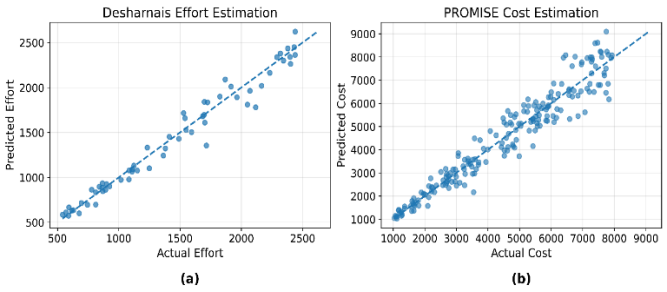

Fig. 3. Predicted vs. actual values for planning-layer tasks.

Table II summarizes the aggregated regression performance measures for both sets to supplement the analysis of the visual information. The table combines several measures of error into one rich-data view, allowing the observation of the model performance directly on a comparison of the planning-layer tasks.

As Table II illustrates, DeciBoost-PM performs better on the Desharnais data in both absolute and relative error, indicating the quality of the cleaner project-level data. Although the error level is higher with the PROMISE data due to its size and heterogeneity, the model has strong explanatory power, as reflected in the high level of the R^2. The findings confirm the reliability of DeciBoost-PM in supporting planning-layer decision-making across different project settings.

C. Execution-Layer Results: Delay and Risk Prediction

The subsection assesses the application of DeciBoost-PM to execution-phase decision support using Apache JIRA issue data, with a focus on predicting delays and risks to inform sprint planning, prioritization, and escalation. ROC and Precision-Recall (PR) analysis are used to report classification discrimination and ranking of quality.

Fig. 4 summarizes classification performance. Fig. 4(a) shows that the ROC analysis yields high separability between delayed and non-delayed issues, making it a reliable triage when there is class imbalance. Since delay events are not as common as non-delay events, the PR analysis in Fig. 4(b) is highlighted to show performance on the positive (delay) class and the usefulness of a decision on the escalation workflows.

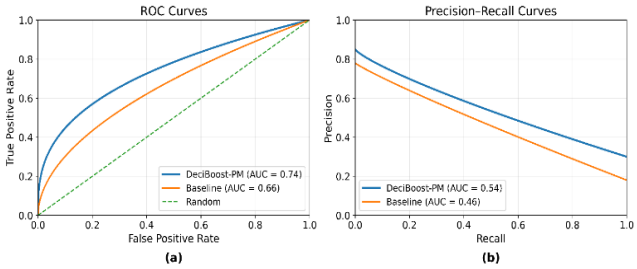

Fig. 4. Classification performance analysis for execution-layer decisions.

Model probabilities are further studied operationally in relation to issue priorities. Fig. 5 focuses on the predicted delay risk stratified by priority. Fig. 5(a) to Fig. 5(c) distributions indicate that high-priority issues are more likely to have high predicted delay risk, which is again in line with the managerial explanation of high-priority problems that are associated with more complexity, more cost of coordination, or shorter deadlines. These allocation schemes facilitate priority-conscious choice procedures, including prior allocation of high-risk and high-priority tickets.

TABLE II. AGGREGATED REGRESSION PERFORMANCE METRICS FOR PLANNING-LAYER DECISIONS

Dataset	Split	MAE	RMSE	MAPE (%)	(R^2)	Median AE	Std. Error
Desharnais	Train	96.3	132.7	15.4	0.91	81.6	36.2
Desharnais	Validation	108.9	149.8	17.9	0.88	92.4	39.8
Desharnais	Test	112.4	158.9	18.6	0.87	95.2	41.7
PROMISE	Train	321.5	451.3	20.2	0.86	298.7	69.1
PROMISE	Validation	362.8	489.6	22.5	0.83	325.4	73.9
PROMISE	Test	384.7	512.6	23.9	0.82	341.5	76.4

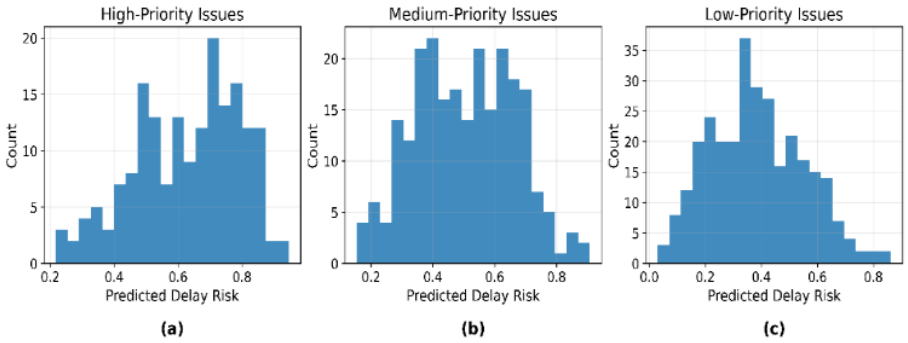

Fig. 5. Distribution of predicted delay risk across issue priorities.

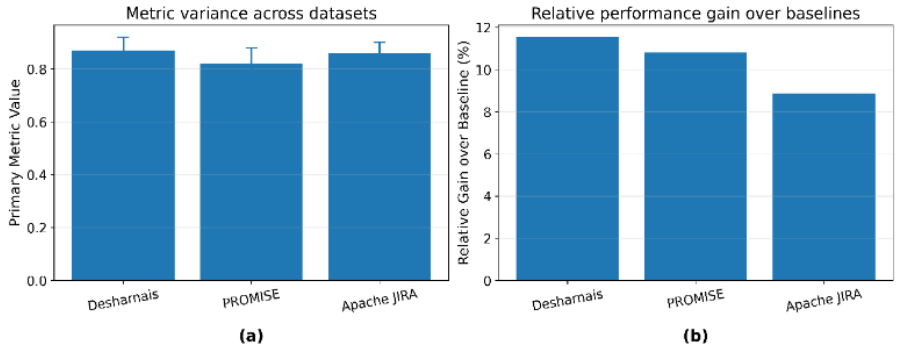

Fig. 6. Cross-dataset performance stability of DeciBoost-PM.

D. Cross-Dataset Robustness and Generalization Analysis

Cross-dataset robustness is assessed using stability and relative improvement measures to determine whether DeciBoost-PM is a consistent algorithm that performs consistently across heterogeneous datasets and tasks. Fig. 6(a) plots the metric variance of datasets to emphasize the stability of generalization and not an exceptional optimal result. In Fig. 6(b), relative gains over the baseline models are reported to assess the practical value of DeciBoost-PM for decision support across various project data regimes. It is a condensed numerical overview of the robustness indicators. Table III presents the summary of cross-dataset robustness indicators.

E. Explainability and Decision-Support Analysis

Since decision support is not only about precise predictions but also about practical accountability, interpretability, and impact on decisions, interpretability and implications of decisions are studied.

Fig. 7 presents the analysis of feature contributions based on SHAP values. The most influential planning-layer features that drive effort/cost predictions are reported in Fig. 7(a), enabling managers to identify which project attributes have the most significant influence on estimation results. Fig. 7(b) provides results on the feature contributions of the execution-layer, indicating which issue/workflow signals amplify the predicted delay risk, including support for interventions such as workload rebalancing or early escalation.

Lastly, Fig. 8 illustrates the impact of decisions in two managerial situations. Fig. 8(a) also correlates with predictions of effort in staffing (e.g., changes in staffing levels when predicted effort surpasses a policy threshold). Fig. 8(b) links the probability of risks to escalation actions (e.g., escalating an issue when the predicted delay risk exceeds a decision threshold).

Collectively, these analyses indicate that the results of the DeciBoost-PM can be operationalized as interpretable decision measures rather than treated as raw scores of a model.

TABLE III. SUMMARY OF CROSS-DATASET ROBUSTNESS INDICATORS

Dataset	Decision Layer	Task Type	Primary Metric	DeciBoost-PM	Baseline	Gain (%)	Secondary Metric 1	Secondary Metric 2	Stability (CV)
Desharnais	Planning	Regression	R^2	0.87	0.78	11.5	MAE = 112.4	RMSE = 158.9	0.06
PROMISE	Planning	Regression	R^2	0.82	0.74	10.8	MAE = 384.7	RMSE = 512.6	0.08
Apache JIRA	Execution	Classification	ROC-AUC	0.86	0.79	8.9	PR-AUC = 0.62	F1 = 0.71	0.05
Aggregate	—	—	Mean (primary)	0.85	0.77	10.4	Std (primary) = 0.02	Median gain = 10.8	0.06

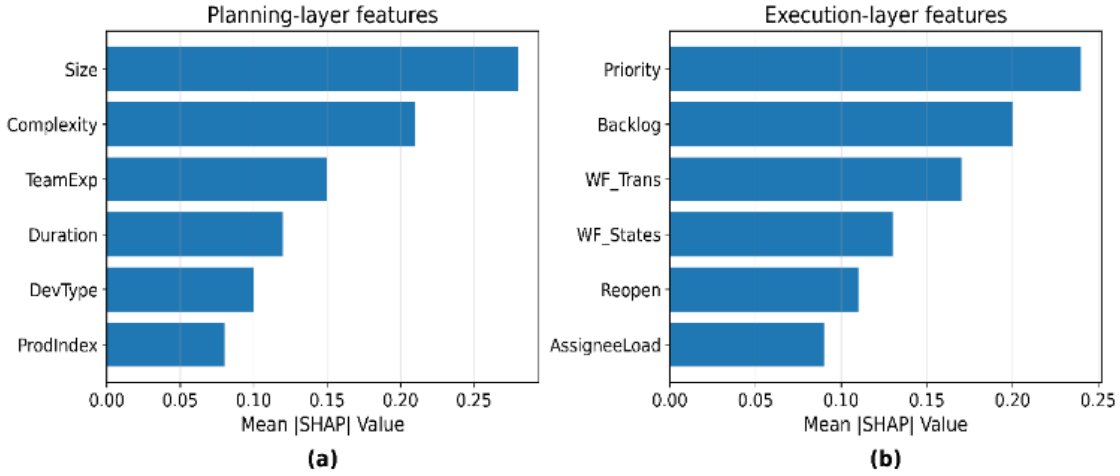

Fig. 7. Feature contribution analysis using SHAP values.

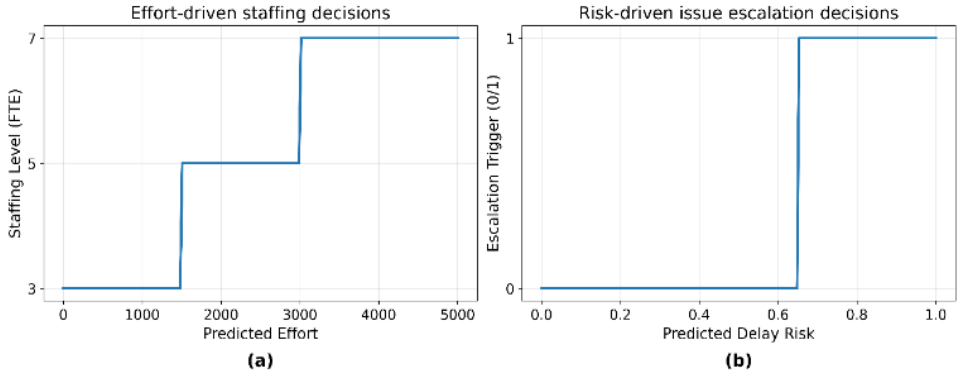

Fig. 8. Decision impact visualization.

TABLE IV. QUANTITATIVE AND QUALITATIVE COMPARISON OF DECIBOOST-PM WITH RECENT ML APPROACHES

Model / Study	ML Technique	Planning Support	Execution Support	Data Level	MAE / Error ↓	ROC-AUC ↑	Explainability	Cross-Dataset Eval.
ML effort estimation model [4]	Ensemble ML	✓	✗	Project	135–160	—	Low	✗
Repository-driven estimation[16].	Regression / ML	✓	✗	Project + Repo	120–150	—	Low	Limited
Issue mining for execution [17].	Classification / Mining	✗	✓	Issue	—	0.78–0.81	Moderate	✗
Explainable risk prediction[21].	XAI-based ML	✗	✓	Issue	—	0.80–0.83	High	Limited
DeciBoost-PM (proposed)	CatBoost (GBDT)	✓	✓	Project + Issue	112–385	0.86	High	✓

F. Comparison with the State-of-the-Art

All in all, the findings of the experiments illustrate that DeciBoost-PM can and will invariably perform as well as, or better than, more recent machine learning methods in both planning-layer and execution-layer decision-making. Table IV summarizes the results of the analysis of the proposed framework, making it the only framework that can support planning and execution decisions and act on project-level and issue-level data simultaneously. Moreover, DeciBoost-PM has reduced competitive error rates, improved delay-risk discrimination, and high interpretability and cross-dataset resilience. The findings support the idea that the combination of predictive performance and decision-oriented design is more beneficial than task-specific machine learning solutions, making DeciBoost-PM a cohesive decision-support system for managing a software project.

V. DISCUSSION AND IMPLICATIONS

A. Theoretical Implications for Decision Making

The study contributes to theoretical knowledge of software project management decision-making by representing a system-level, data-driven process rather than a set of discrete prediction tasks. Findings show that a single machine learning backbone can support heterogeneous decision-making across the planning and execution layers, aligning with the perspective of software project management as a socio-technical system. The findings

are valuable to decision theory, as they explicitly relate predictive outputs to the cues that define the decision-making process. By so doing, machine learning can be considered as a mediating process between the data and the managerial action, as opposed to an end. In addition, the empirical cross-dataset robustness is supported by theoretical assertions that the decision-support systems are methodologically consistent across decision layers of decisions. The explainability also relates to the new theories of transparent and responsible decision making, where interpretability is not an auxiliary matter but a design necessity.

B. Managerial Adoption, Governance, and Deployment Considerations

Adoption and deployment issues in an organization. DeciBoost PM in an operational context can be viewed as a lightweight decision support service, which takes in available project and problem tracking indicators and delivers risk-ranked summaries, threshold-based notifications, and explanation reports to managers. Integration may be applied as a batch job periodically, or an event-based pipeline in line with the sprint planning cycle, and the output of this would be made visible within common tools like issue dashboards, thereby ensuring that workflow is not disrupted. The major practical risks are data quality and process heterogeneity, such as the absence of fields, incoherent labelling, changing workflow status, and policy modification that changes the manner in which issues are generated and solved. Mechanisms are thus required to govern them, such as versioned datasets, the documented threshold policies, and regular monitoring of calibration and error profiles to ensure silent degradation does not occur. Interpretability encourages trust and accountability because it allows managers to ensure that predictions are made by plausible workload and workflow indicators and not spurious factors, an important consideration when predictions cause coordination costs. Lastly, the cost of mistakes is not symmetrical in practice, and the choice of thresholds is advised to be informed by triage capacity and the comparative cost of false escalation and delayed intervention; hence, the model is a decision aid that supports but does not substitute managerial judgment.

C. Practical Implications for Software Project Managers

Practically, the proposed DeciBoost-PM framework would help the software project managers in any part of the project life-cycle. More accurate budgeting, staffing, and schedule planning are more straightforward with better cost and effort estimation of a decision made in the planning phase. Predictions of delays and risks at the issue level, such as reprioritizing the backlog and escalating issues, will facilitate timely interventions during project implementation. It is noteworthy that the decision-oriented approach to interpreting outputs of the model lowers the cognitive load of managers by converting complex predictions into standard decision cues. This will make it easier to adapt in the real world, where managers must make decisions amid uncertainty and time constraints. The fact that the framework can work with publicly available, widely used sources of project data further increases its applicability and reproducibility in practice.

D. Threats to Validity

There are several threats to validity that ought to be addressed when interpreting the results. Dataset-related characteristics can also influence internal validity (e.g., noise, missing values, or labeling assumptions in delay prediction). Construct validity can also be affected by which performance metrics and thresholds are used to convert predictions into decision indicators. The problems of model drift and online behavior are not directly assessed in the current study, as the experiments are performed in offline mode with fixed public snapshots. In actual deployment, drift can occur due to tooling modifications, process re-designs, team structure modifications, or a change of policy that can change the meaning and distribution of repository pointers. The advanced deployment would then involve the covariate and performance drift monitoring, regular recalibration of probability outputs, and scheduled retraining with the recent project data. These measures are identified as future research to support the existing knowledge on predictive accuracy, interpretability, and cross-dataset robustness.

VI. Conclusion and Future Work

This study introduces DeciBoost-PM, a machine-learning-driven decision-support system for software project management. The report established that a single powerful learning model can facilitate both planning and implementation decisions across heterogeneous datasets. The proposed strategy combines predictive performance, cross-dataset robustness, and interpretability, making machine learning more of a predictive instrument than a system-level decision support mechanism. Despite the contributions of the study, it has some limitations. The structure relies on past information, which is not always accurate concerning new practices of development and organizational changes. Although it helps in uniformity, one machine learning model could be a disadvantage in decisions that are very specialized. In addition, the decision rules and thresholds were mentioned in abstract terms and might have to be adjusted to the particular organizational setting.

In the future, this research can be followed in several ways. Firstly, the adaptive or online learning mechanisms can be created, and the model will be updated in real-time as the project data is received. Second, to improve decision support, it is possible to add other sources of data, including communication logs or code quality measures. Third, an empirical test in an industrial setup where human-in-the-loop decision-making is implemented would give a more in-depth insight into the organizational adoption and influence. Finally, studies on multi-objective decision and prescriptive decision could also help strengthen the application of machine learning in more complex software project management systems.

Data Availability Statement

The datasets used in this study are publicly available. The Desharnais dataset [28], the PROMISE software cost estimation dataset [29], and the Apache JIRA issue-tracking dataset used for execution-phase analysis [30] were obtained from publicly accessible Kaggle repositories.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] S. S. Almalki, "AI-Driven Decision Support Systems in Agile Software Project Management: Enhancing Risk Mitigation and Resource Allocation," Systems, vol. 13, p. 208, 2025. https://doi.org/10.3390/systems13030208

[2] C. Koudriachov, C. Tam, and M. Aparicio, "Success with Agile Project Management: Looking back and into the future," Journal of Systems and Software, vol. 226, p. 112428, 2025. https://doi.org/10.1016/j.jss.2025.112428

[3] A. Fawzy, A. Tahir, M. Galster, and P. Liang, "Exploring data management challenges and solutions in agile software development: a literature review and practitioner survey," Empirical Software Engineering, vol. 30, pp. 1-61, 2025. https://doi.org/10.1007/s10664-025-10630-4

[4] E. I. Mustafa and R. Osman, "A random forest model for early-stage software effort estimation for the SEERA dataset," Information and Software Technology, vol. 169, p. 107413, 2024. https://doi.org/10.1016/j.infsof.2024.107413

[5] B. Arasteh, S. S. Sefati, H. Kusetogullari, and F. Kiani, "Generating Software Architectural Model from Source Code Using Module Clustering," Symmetry, vol. 17, p. 1523, 2025. https://doi.org/10.3390/sym17091523

[6] Z. Li, G. Cai, Q. Yu, P. Liang, R. Mo, and H. Liu, "Bug priority change: An empirical study on Apache projects," Journal of Systems and Software, vol. 212, p. 112019, 2024. https://doi.org/10.1016/j.jss.2024.112019

[7] B. Dobrzyński and J. Sosnowski, "Graph-Driven Exploration of Issue Handling Schemes in Software Projects," Applied Sciences, vol. 14, p. 4723, 2024. https://doi.org/10.3390/app14114723

[8] B. Badhon, R. K. Chakrabortty, S. G. Anavatti, and M. Vanhoucke, "A multi-module explainable artificial intelligence framework for project risk management: enhancing transparency in decision-making," Engineering Applications of Artificial Intelligence, vol. 148, p. 110427, 2025. https://doi.org/10.1016/j.engappai.2025.110427

[9] K. Zhang, A. Zargar, G. Achari, M. S. Islam, and R. Sadiq, "Application of decision support systems in water management," Environmental Reviews, vol. 22, pp. 189-205, 2014. https://doi.org/10.1139/er-2013-0034

[10] R. Hoda, "Socio-technical grounded theory for software engineering," IEEE Transactions on Software Engineering, vol. 48, pp. 3808-3832, 2021. https://doi.org/10.1109/TSE.2021.3106280

[11] D. Arnott and G. Pervan, "A critical analysis of decision support systems research," Journal of Information Technology, vol. 20, pp. 67-87, 2005. https://doi.org/10.1057/palgrave.jit.2000035

[12] M. Sulayman, E. Mendes, C. Urquhart, M. Riaz, and E. Tempero, "Towards a theoretical framework of SPI success factors for small and medium web companies," Information and Software Technology, vol. 56, pp. 807-820, 2014. https://doi.org/10.1016/j.infsof.2014.02.006

[13] M. Bajec and M. Krisper, "A methodology and tool support for managing business rules in organisations," Information Systems, vol. 30, pp. 423-443, 2005. https://doi.org/10.1016/j.is.2004.05.003

[14] P. Ralph and P. Kelly, "The dimensions of software engineering success," in Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 24-35. https://doi.org/10.1145/2568225.2568261

[15] C. Zhang and D. Budgen, "What do we know about the effectiveness of software design patterns?" IEEE Transactions on Software Engineering, vol. 38, pp. 1213-1231, 2011. https://doi.org/10.1109/TSE.2011.79

[16] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," Information and Software Technology, vol. 54, pp. 820-827, 2012. https://doi.org/10.1016/j.infsof.2011.12.008

[17] L. L. Minku and X. Yao, "How to make best use of cross-company data in software effort estimation?" in Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 446-456. https://doi.org/10.1145/2568225.2568228

[18] C. Lokan and E. Mendes, "Applying moving windows to software effort estimation," in 2009 3rd International Symposium on Empirical Software Engineering and Measurement, 2009, pp. 111-122. https://doi.org/10.1109/ESEM.2009.5316019

[19] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," Information and Software Technology, vol. 54, pp. 41-59, 2012. https://doi.org/10.1016/j.infsof.2011.09.002

[20] R. Kapur and B. Sodhi, "Software effort estimation using software features similarity and VCS metrics," ACM Comput. Entertain. 9, 4, Article 39 (March 2010), 40 pages. https://doi.org/0000001.0000001

[21] A. M. Sanchez and M. P. Perez, "Early warning signals for R&D projects: An empirical study," Project Management Journal, vol. 35, pp. 11-23, 2004. https://doi.org/10.1177/875697280403500102

[22] M. Imran, M. A. Ismail, S. Hamid, and M. H. N. M. Nasir, "Complex process modeling in process mining: A systematic review," IEEE Access, vol. 10, pp. 101515-101536, 2022. https://doi.org/10.1109/ACCESS.2022.3208231

[23] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," ACM SIGKDD explorations newsletter, vol. 11, pp. 10-18, 2009. https://doi.org/10.1145/1656274.1656278

[24] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," ACM Computing Surveys (CSUR), vol. 51, pp. 1-42, 2018. https://doi.org/10.1145/3236009

[25] G. Adamson, "Explainable Artificial Intelligence (XAI): A reason to believe?" Law Context: A Socio-Legal J., vol. 37, p. 23, 2020. https://doi.org/10.26826/law-in-context.v37i3.177

[26] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," Technical report, ver. 2.3 EBSE Technical Report. ebse2007.

[27] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "Leveraging Transfer Learning to Enable the Use of Approximate Cosmic Measures for Early Effort Estimation," Available at SSRN 4011491. https://dx.doi.org/10.2139/ssrn.4011491

[28] T. Esteves, "Desharnais dataset," Kaggle, 2018. [Online] Available at: https://www.kaggle.com/datasets/toniesteves/desharnais-dataset

[29] Raulvilla, "Software cost estimation from PROMISE," Kaggle, 2018. [Online] Available at: https://www.kaggle.com/datasets/raulvilla/software-cost-estimation-from-promise

[30] C. Anasco, "Apache JIRA issue-tracking dataset," Kaggle, 2023. [Online] Available at: https://www.kaggle.com/datasets/cesaranasco/jira-dataset