**Journal of Computer Science and Technology Studies**
ISSN: 2978-8048
DOI: 10.32996/jcsts.2026.5.8.1
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

**| RESEARCH ARTICLE**

# Benchmarking Machine Learning Models for Software Project Schedule Overrun Prediction

**Abdelkarim Ramzy Sweilem [1]✉, Hala Jamil Abdoh [2], and Dr. Ahmad Jamil Abdoh [3]**
[1] *Independent Researcher, Riyadh, Saudi Arabia*
[2] *Independent Researcher, Riyadh, Saudi Arabia*
[3] *3IQ Academy, Amman, Jordan*
**Corresponding Author**: Abdelkarim Ramzy Sweilem, **E-mail**: Abdelkarim@asweilem.com

**| ABSTRACT**
Software projects frequently exceed their planned schedules because project performance is shaped by interacting technical, organizational, resource, risk, and stakeholder-related factors. This study benchmarks six regression algorithms—Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network—to predict the percentage of schedule overrun in software projects and identify the model that provides the most favorable balance between predictive accuracy and interpretability. All six models were developed and evaluated using a publicly available dataset containing 4,517 software-project records. To ensure a consistent comparison, the models used identical predictors, default hyperparameter configurations, and a 10-fold cross-validation procedure. Predictive performance was assessed using root mean squared error, mean absolute error, correlation, and squared correlation. Gradient Boosting Machine achieved the lowest prediction errors, with an RMSE of 4.578 and an MAE of 3.503. Deep Neural Network produced nearly equivalent performance and recorded the highest squared correlation of 0.875. Random Forest ranked third, whereas Support Vector Regression produced the weakest performance under the adopted default settings. Interpretability analysis further identified Risk Factor, Duration Months, and Client Satisfaction Rating as the most consistently influential predictors across the interpretable models. The findings demonstrate the potential of ensemble and deep-learning methods to estimate the magnitude of schedule overrun using structured software-project data. Considering both prediction error and interpretability, Gradient Boosting Machine provided the strongest overall balance and represents a practical candidate for early-warning systems, risk-based project prioritization, schedule-recovery planning, and portfolio-level decision support. Future research should validate the models using independent organizational datasets, optimize their hyperparameters, and apply consistent explainability methods across all algorithms.

**| KEYWORDS**
Artificial Intelligence; Machine Learning; Project Management; Schedule Overrun; Gradient Boosting Machine; Deep Neural Network; Predictive Modeling; Data-Driven Decision-Making.

**| ARTICLE INFORMATION**
**ACCEPTED:** 01 June 2026 **PUBLISHED:** 26 July 2026 **DOI:** 10.32996/jcsts.2026.8.8.13

## 1. Introduction

Software projects are frequently exposed to schedule overruns because of changing requirements, technical uncertainty, resource constraints, interdependent activities, and evolving stakeholder expectations. Schedule delays can increase project costs, reduce client satisfaction, disrupt organizational plans, and postpone the realization of expected business value. Accurate schedule forecasting therefore remains an important component of effective project planning, monitoring, and control [1,3,4].

Traditional schedule-control methods, including baseline comparison, earned value analysis, earned schedule, and periodic progress reporting, provide important information about project performance. However, these approaches mainly depend on predefined plans and observed progress indicators and may have limited ability to capture complex and nonlinear relationships among project risk, duration, effort, cost, team characteristics, and stakeholder-related factors [10–13]. As a result, schedule problems may become visible only after significant deviation has already occurred.

Machine-learning techniques provide an alternative data-driven approach by learning patterns from historical project records and generating predictive estimates for new observations. Such techniques can identify relationships that may be difficult to represent through conventional statistical or schedule-control methods and may support earlier identification of projects with elevated schedule-overrun exposure. The growing application of artificial intelligence and predictive analytics in project management has demonstrated their potential to strengthen risk assessment, forecasting, monitoring, and managerial decision support [14–17].

Previous studies have applied machine-learning, ensemble, hybrid, and deep-learning models to project-delay prediction [19–24,29,30]. However, much of the existing research has focused on construction projects, sector-specific datasets, delay-risk classification, or comparisons involving a limited number of algorithms. Classification-based studies commonly divide projects into categories such as delayed and not delayed, which may conceal differences in the actual severity of schedule deviation. Predicting schedule overrun as a continuous value preserves more information and enables the expected magnitude of the delay to be estimated rather than only identifying its category [25].

Comparatively less attention has been given to controlled benchmarking of traditional statistical, tree-based, ensemble, support-vector, and deep-learning regression models for predicting the continuous percentage of schedule overrun in software projects. Differences in datasets, target definitions, preprocessing procedures, model settings, validation methods, and evaluation metrics also make direct comparison across previous studies difficult. A unified experimental framework is therefore required to evaluate multiple algorithms under consistent conditions.

Accordingly, the primary objective of this study is to benchmark six regression algorithms for predicting the continuous percentage of schedule overrun in software projects and to identify the model that provides the most favorable balance between predictive accuracy and interpretability. The study also aims to compare the models using common regression-performance measures and to determine which project characteristics contribute most strongly to schedule-overrun prediction.

To achieve this objective, the study evaluates Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. All six models are developed using the same dataset, predictor variables, continuous target variable, preprocessing decisions, default hyperparameter configurations, and 10-fold cross-validation procedure. Their predictive performance is assessed using root mean squared error, mean absolute error, correlation, and squared correlation. Model interpretation is additionally examined using regression coefficients, tree structure, and feature-importance outputs where directly available.

The study addresses the following research questions:

RQ1: Which of the six evaluated regression algorithms provides the most accurate prediction of software-project schedule-overrun percentage?

RQ2: How do the models differ in terms of RMSE, MAE, correlation, and squared correlation?

RQ3: Which project characteristics contribute most strongly to schedule-overrun prediction across the interpretable models?

The main contribution of the study is the controlled comparison of six regression approaches under a unified and reproducible experimental framework. Unlike studies that reduce project delay to predefined categories, the present research retains schedule overrun as a continuous percentage, allowing the models to estimate the expected magnitude of schedule deviation. The study also combines predictive-performance evaluation with model-interpretability analysis, thereby addressing both which algorithm performs best and which project characteristics contribute most strongly to its predictions.

From a practical perspective, the findings may support the development of data-driven schedule-monitoring and decision-support tools for project managers and project-management offices. Continuous schedule-overrun estimates may assist organizations in identifying projects with elevated schedule exposure, prioritizing corrective actions, strengthening risk-based governance, and improving portfolio-level monitoring. Nevertheless, predictive outputs are intended to complement professional judgment and established project-control procedures rather than replace them.

The remainder of the paper is organized as follows. Section 2 reviews the literature related to schedule overruns, conventional schedule-control approaches, and machine-learning applications in project-delay prediction. Section 3 describes the dataset, data preparation, predictive algorithms, validation procedure, evaluation measures, and reproducibility considerations. Section 4 presents the empirical performance and interpretability results. Section 5 discusses the findings, compares them with previous studies, and outlines their practical and research implications. Section 6 presents the study limitations, while Section 7 provides the conclusion and directions for future research.

## 2. Background and Related Work

### *2.1 Schedule Overrun in Technical Projects*

A project schedule represents the time-based model through which project activities, dependencies, milestones, and planned completion dates are organized and controlled. An effective schedule provides a baseline against which actual project progress can be monitored and deviations can be identified. Schedule overrun occurs when the actual duration or completion date of a project exceeds the duration or completion date established in the approved schedule baseline [1].

Schedule overrun is particularly significant in technical projects, including information technology implementations, software development, systems integration, digital transformation, data and artificial-intelligence initiatives, and technology infrastructure projects. Such projects commonly involve interdependent technical components, multiple delivery teams, specialized resources, external vendors, testing environments, integration points, and approval dependencies. A delay in one component may therefore affect several dependent activities and generate cumulative effects across the project schedule [2,3].

Unlike projects in which activities can be executed independently, technical projects frequently involve complex logical and technological dependencies. Software components may need to be integrated before system testing can begin, infrastructure environments must be prepared before deployment, and business approval may be required before a solution can progress from development to production. Consequently, a delay affecting requirements, development, integration, testing, cybersecurity approval, infrastructure readiness, or vendor delivery can extend the overall project duration.

Schedule overrun can also produce consequences beyond the delayed completion date. Prolonged projects may require additional human and financial resources, postpone the realization of expected business benefits, increase coordination costs, and expose the project to further changes and risks [3]. Evidence from information-technology projects also indicates that schedule problems may coexist with substantial cost exposure and operational disruption, particularly in complex or poorly controlled initiatives [4].

As schedule pressure increases, project teams may attempt to recover lost time by compressing activities, increasing workload, executing tasks concurrently, or proceeding with incomplete predecessor outputs. Although such actions may provide short-term progress, they may also increase rework, reduce quality, and negatively affect team productivity. Research on software engineering has shown that time pressure is frequently associated with estimation problems and may improve short-term productivity while adversely affecting software quality [5]. Schedule slippage can therefore become self-reinforcing when delayed work creates inefficient sequencing, additional coordination requirements, and further productivity losses [2].

For analytical purposes, schedule overrun can be expressed as the percentage by which the actual project duration exceeds the planned duration:

Schedule Overrun (%) = ((Actual Duration − Planned Duration) / Planned Duration) × 100

A positive value indicates that the project exceeded its planned duration, a value of zero indicates completion according to plan, and a negative value represents completion earlier than planned. Compared with a simple delayed/not-delayed indicator, the percentage measure provides a more informative representation of delay severity because it accounts for differences in planned project duration.

Accurately assessing schedule overrun is therefore essential for project governance, resource planning, risk management, and corrective decision-making. However, schedule performance is influenced by multiple interacting factors that may not be adequately represented through a single variance measure. The following section examines the principal factors identified as contributors to schedule overrun in technical projects.

### *2.2 Factors Influencing Project Schedule Overrun*

Schedule overrun in technical projects is rarely caused by a single isolated event. It typically emerges from the interaction of technical, organizational, managerial, and external factors throughout the project life cycle. These factors may affect individual activities directly or propagate through dependent tasks, resulting in cumulative delays that extend the overall project duration.

Project complexity is one of the principal factors affecting schedule performance. Technical projects frequently contain interconnected systems, specialized technologies, multiple interfaces, and dependencies among business, development, infrastructure, security, testing, and operational teams. As the number of interacting components increases, it becomes more difficult to estimate activity durations, coordinate resources, and anticipate the consequences of changes. Project complexity also amplifies uncertainty because the impact of an event in one workstream may not become visible until it affects subsequent activities [6].

Requirements uncertainty represents another important source of schedule instability. Technical requirements may be incomplete, ambiguous, or subject to change as stakeholders develop a clearer understanding of the expected solution. Uncertain requirements reduce the reliability of initial estimates and may lead to repeated analysis, redesign, redevelopment, and testing. Research on software projects has shown that requirements uncertainty can affect schedule, cost, effort estimation, development processes, and final product quality [7].

Closely related to requirements uncertainty is scope creep, which refers to the gradual addition or modification of project requirements without adequate adjustment to the approved scope, schedule, resources, or budget. Scope creep may arise from

unclear objectives, informal stakeholder requests, insufficient requirements analysis, weak change control, or evolving business expectations. Empirical research in software projects has identified technological, organizational, and human factors as contributors to scope creep and has demonstrated its negative relationship with project success [8]. Consequently, the number, timing, and complexity of change requests may significantly influence schedule performance.

Risk and issue exposure also affect the probability and magnitude of schedule overrun. A risk represents an uncertain event that may affect project objectives, whereas an issue is a condition that has already occurred and requires resolution. When critical risks are not identified or mitigated early, they may develop into operational issues that interrupt project activities. Similarly, unresolved, blocked, or overdue issues can prevent dependent tasks from starting or completing. Research on information-technology project failures indicates that weaknesses in planning, risk identification, governance, communication, and organizational decision-making can interact and create mechanisms that increase the likelihood of poor project outcomes [3].

Resource availability and team capability are additional determinants of schedule performance. Technical initiatives often depend on specialized professionals whose skills may not be easily replaced or scaled. Inadequate staffing, competing priorities, skill gaps, employee turnover, and delayed resource onboarding may reduce productivity and create bottlenecks. Conversely, increasing team size does not always accelerate delivery because larger teams require additional communication, coordination, and integration effort. The schedule impact of team size therefore depends on the project's complexity, work structure, and distribution of responsibilities.

Time pressure may further affect project performance when teams attempt to compensate for unrealistic estimates or accumulated delays. Schedule compression can increase workload, encourage concurrent execution of dependent activities, and reduce the time available for quality assurance and technical review. A systematic review of software-engineering research found that time pressure is closely associated with estimation problems and can influence productivity, quality, and developer well-being [5]. A recent review of Agile software development issues similarly identified delivery pressure and insufficient customer involvement as important sources of wider project problems [9].

Stakeholder participation and decision-making speed can also influence the schedule. Technical projects often require timely clarification, prioritization, acceptance, and approval from business owners, governance committees, security teams, regulators, and operational stakeholders. Delayed decisions may prevent teams from finalizing requirements, approving designs, resolving changes, or progressing between project environments. Limited stakeholder engagement may also increase the probability of late feedback, rejected deliverables, and rework.

External dependencies introduce further uncertainty, particularly when project delivery relies on vendors, implementation partners, cloud providers, equipment suppliers, or third-party systems. Vendor delays, contractual disputes, unavailable environments, integration failures, and delayed approvals can affect activities outside the direct control of the internal project team. The effect may be especially significant when the delayed external deliverable lies on the project's critical path.

Project methodology and governance practices can either reduce or intensify the impact of these factors. Clear roles, realistic plans, formal change control, active risk and issue management, regular progress validation, and timely escalation can help teams identify emerging schedule threats. In contrast, weak governance, unreliable progress reporting, insufficient dependency management, and delayed corrective action may allow minor variances to develop into significant overruns.

These factors are not independent. Requirements changes may increase technical complexity, create additional risks, generate new issues, and place pressure on available resources. Likewise, a delayed vendor deliverable may trigger schedule compression, increase issue aging, and require unplanned change requests. Schedule-overrun prediction should therefore consider the combined and potentially nonlinear relationships among project characteristics, risks, issues, changes, resources, and governance conditions rather than evaluating each factor in isolation.

### *2.3 Traditional Schedule Forecasting Approaches*

Traditional project schedule forecasting relies on structured planning, baseline comparison, expert estimation, and analytical techniques for determining activity sequences, project duration, and expected completion dates. These approaches remain fundamental to project management because they provide the schedule model against which actual performance can be measured and controlled. Common methods include expert judgment, analogous estimation, the Critical Path Method, the Program Evaluation and Review Technique, Earned Value Management, Earned Schedule, and probabilistic schedule-risk analysis [1,10].

Expert judgment and analogous estimation are commonly used during the early stages of a project when detailed activity information is limited. Project managers and subject-matter experts estimate future durations by drawing on professional experience, organizational records, and similarities with previously completed projects. Although these techniques can be applied before a detailed schedule is available, their accuracy depends on the relevance of historical information, the experience of the estimator, and the similarity between the proposed project and previously completed work. They may also be affected by optimism, incomplete information, and poorly calibrated duration estimates [1,12].

The Critical Path Method (CPM) provides a network-based representation of project activities and their logical dependencies. It identifies the longest sequence of dependent activities that determines the minimum project duration and earliest possible

completion date. Activities located on the critical path have little or no scheduling flexibility, and a delay affecting one of these activities may directly influence the planned project completion date [1].

Despite its practical value, CPM is generally deterministic because it commonly represents each activity using a single duration estimate. This assumption simplifies schedule development but may not adequately represent the uncertainty associated with technical activities, resource constraints, external dependencies, emerging risks, and changing requirements. In addition, the apparent critical path may change during execution as actual durations and project conditions deviate from the original baseline [11,12].

The Program Evaluation and Review Technique (PERT) was developed to incorporate uncertainty into activity-duration estimation. Instead of using one deterministic duration, PERT commonly uses optimistic, most likely, and pessimistic estimates to calculate an expected duration. This provides a probabilistic perspective that is more suitable for uncertain activities than a single-point estimate [1,11].

However, conventional PERT is sensitive to the quality of expert estimates and to assumptions concerning activity-duration distributions and independence among activities. Research has indicated that traditional PERT may underestimate schedule variation when its underlying assumptions do not reflect actual project conditions [11,12]. Consequently, although PERT introduces uncertainty into schedule analysis, its output may remain unreliable when the estimates are subjective, poorly calibrated, or based on insufficient historical evidence.

Earned Value Management (EVM) provides another widely established approach for project-performance monitoring and forecasting. EVM integrates scope, schedule, and resource information to compare planned work, completed work, and actual expenditure. Schedule Variance and the Schedule Performance Index can indicate whether the amount of completed work is ahead of or behind the planned level [10].

A limitation of conventional EVM schedule indicators is that they are derived from value-based measures rather than being expressed directly in units of time. Moreover, traditional schedule indicators may lose their usefulness as the project approaches completion because all planned value is eventually earned. Earned Schedule was introduced as an extension of EVM to translate performance information into time-based indicators and support forecasts of the project's expected completion duration [13].

Traditional methods may also be supplemented by Monte Carlo simulation. In schedule-risk analysis, uncertain activity durations and risk events are represented using probability distributions, and the schedule model is simulated repeatedly. The resulting distribution can estimate the probability of completing the project by a particular date and can identify activities that contribute strongly to schedule uncertainty [11]. Unlike deterministic scheduling, this approach produces a range of possible completion outcomes rather than one fixed completion date.

Nevertheless, probabilistic simulation depends on the validity of its input distributions, assumptions regarding correlations, and the quality of risk estimates. If the probability distributions are based primarily on subjective judgment or incomplete data, the simulation may produce precise-looking outputs without necessarily providing accurate forecasts [12]. The method can also require considerable effort to define, validate, and maintain the required inputs.

Overall, traditional schedule-forecasting approaches provide essential mechanisms for planning, baseline control, dependency analysis, progress measurement, and uncertainty assessment. However, they commonly depend on predefined relationships, fixed analytical assumptions, manually selected probability distributions, or expert-generated estimates. These limitations become particularly important in technical projects where schedule performance may be influenced by complex and nonlinear interactions among project characteristics, risks, issues, changes, resources, and delivery conditions.

The need to identify such interactions from historical project data has contributed to the growing use of artificial intelligence and predictive analytics in project management. The following section discusses how these technologies extend traditional schedule-control approaches by learning predictive patterns directly from project data.

### *2.4 Artificial Intelligence and Predictive Analytics in Project Management*

Artificial intelligence refers to a broad group of computational methods that enable systems to perform tasks commonly associated with human intelligence, including pattern recognition, learning, reasoning, prediction, and decision support. Within project management, AI can process historical and current project data to identify relationships that may not be immediately visible through conventional reporting and manual analysis. Its applications can include schedule forecasting, cost estimation, risk identification, resource allocation, progress monitoring, and decision support [14,15].

Predictive analytics represents one of the most relevant AI-enabled capabilities for project management. It combines historical data, statistical techniques, data mining, and machine-learning algorithms to estimate the likelihood or numerical value of future outcomes. Rather than reporting only what has already occurred, predictive analytics attempts to determine what is likely to occur and how strongly the available project conditions support that forecast [16].

Traditional project monitoring is commonly based on comparisons between planned and actual performance. Such comparisons remain essential, but they are generally descriptive or diagnostic because they identify an existing variance after it has appeared. Predictive analytics extends this process by learning from patterns across previously observed projects and using these patterns

to estimate future outcomes for current projects. This distinction can allow project teams to move from reactive schedule control toward earlier and more proactive intervention [14,16].

The potential impact of AI is especially relevant to schedule, cost, and risk management. Fridgeirsson et al. [14] found that project-management experts expected AI to have a substantial influence on these three knowledge areas. Within schedule management, AI was considered particularly relevant to activities involving historical information, schedule baselines, monitoring, and forecast adjustment. These areas generate structured data that can be analyzed to identify recurring relationships between project characteristics and performance outcomes.

In schedule management, predictive models can analyze variables such as planned duration, project complexity, team characteristics, risk exposure, unresolved issues, change requests, and progress information. A model can then estimate an expected delay, schedule-overrun percentage, or probability of meeting the planned completion date. Such estimates do not eliminate schedule uncertainty, but they provide an additional evidence-based indicator that can complement the project baseline, expert assessment, and governance reporting [14,16].

AI can also support continuous forecasting. Traditional forecasts may be prepared periodically and depend heavily on the project manager's interpretation of the latest status information. A data-driven model can be reapplied whenever updated project data become available, enabling the forecast to change as risks materialize, issues remain unresolved, changes are approved, or progress deviates from the expected trajectory. Predictive analytics has consequently been proposed as a means of estimating project outcomes and supporting the practical use of project-management master data [16].

Machine learning further extends conventional statistical analysis by learning relationships directly from data. Depending on the selected algorithm, it can model linear effects, decision rules, nonlinear interactions, and complex combinations of project variables. This capability is important because schedule overrun is rarely determined by one factor in isolation. A moderate level of complexity, for example, may not result in delay unless it is combined with high risk exposure, unresolved issues, insufficient resources, or repeated scope changes [6–9,15].

AI-based forecasting should nevertheless be treated as decision support rather than as a replacement for project-management judgment. A model identifies patterns represented in the data used for its development, but it may not fully account for new technologies, exceptional events, organizational politics, contractual constraints, or other contextual information that has not been captured in the dataset. Project managers therefore remain responsible for interpreting forecasts, validating their relevance, and selecting appropriate corrective actions [14,15,17].

The quality of AI-based predictions also depends on the availability, consistency, completeness, and representativeness of project data. Inaccurate progress records, inconsistent definitions, missing variables, or biased historical outcomes may reduce model reliability. Systematic reviews of AI in project management have therefore identified data availability, organizational readiness, technological maturity, implementation capability, and integration with existing processes as important challenges to effective adoption [15].

Another important consideration is model transparency. Highly accurate predictions may have limited practical value when decision-makers cannot understand the factors contributing to them. Explainable machine-learning methods can help connect model outputs to project activities and characteristics, allowing managers to identify the variables influencing a forecast and to distinguish between forward-looking decision support and retrospective explanation of project performance [17].

Accordingly, AI-based predictive analytics should be integrated with established project-governance practices. Model outputs should be reviewed alongside the approved schedule, risk and issue registers, change records, resource information, and professional judgment. When used in this manner, predictive analytics can strengthen rather than replace traditional project controls by providing an additional early-warning mechanism for emerging schedule threats [14,17].

Although AI offers several approaches to project-delay prediction, previous studies have formulated the prediction problem in different ways. Several studies have applied binary classification to determine whether a project is likely to be delayed, while others have used multiclass classification to estimate delay severity. A smaller number of studies have treated delay prediction as a continuous regression problem by estimating its numerical magnitude [1,7]. The following section examines the distinction between classification-based and regression-based formulations.

### *2.5 Problem Formulation in Previous Delay-Prediction Studies*

The formulation of the prediction target is a fundamental methodological decision in supervised machine learning. It determines the type of output generated by the model, the algorithms that can be applied, and the evaluation metrics used to assess predictive performance. In project-delay research, the prediction problem has generally been formulated either as a classification task or as a regression task [18].

Classification-based approaches assign projects to predefined delay outcomes or severity categories. Regression-based approaches, in contrast, estimate the numerical magnitude of the expected delay, such as delay days, final project duration, schedule-to-completion, or schedule-overrun percentage. Although both formulations can support project control and decision-making, they provide different levels of information and serve different managerial purposes.

### *2.5.1 Classification-Based Prediction*

Classification-based prediction assigns each project to one of a finite number of predefined classes. In binary classification, the model typically determines whether a project is expected to be delayed or not delayed. In multiclass classification, projects may be assigned to categories such as minor, moderate, or major delay.

This formulation is useful for early-warning systems because classification outputs can be communicated through simple labels, risk bands, or traffic-light indicators. Projects classified as high risk can be prioritized for management attention, additional monitoring, or corrective action. Classification performance is commonly evaluated using accuracy, precision, recall, sensitivity, specificity, F1-score, and the area under the receiver operating characteristic curve [18].

Several previous project-delay studies have adopted binary or multiclass classification formulations. Some studies predicted whether a project would experience delay, whereas others divided delay magnitude into predefined severity categories. These approaches demonstrated the potential of machine-learning models to identify projects exposed to schedule risk [19,22].

However, converting a continuous delay outcome into discrete categories results in a loss of numerical information and may reduce the ability of the analysis to distinguish between substantially different outcomes [18,25]. Projects with schedule overruns of 31% and 58%, for example, may both be classified as moderate delay, even though they may require different recovery strategies and levels of management intervention.

Classification results also depend on the thresholds selected to define each class. A small numerical difference near a category boundary may change the assigned label, whereas larger differences within the same category may not be reflected in the model output [25]. Consequently, classification is suitable for risk screening and prioritization, but it may provide insufficient numerical detail for schedule-recovery planning, resource forecasting, and quantitative portfolio analysis.

### *2.5.2 Regression-Based Prediction*

Regression-based prediction estimates a continuous numerical outcome. Depending on the research objective, the target may represent project duration, remaining duration, delay days, schedule-to-completion, time-overrun ratio, or schedule-overrun percentage [18].

Unlike classification, regression does not merely indicate whether a delay is expected or assign a project to a predefined risk category. Instead, it estimates the numerical magnitude of the expected deviation. For example, a regression model may predict that a project will exceed its planned duration by 24.5%, thereby providing greater numerical detail than a general high-delay classification [18].

This level of precision can support resource reallocation, schedule-recovery planning, contingency assessment, stakeholder communication, and portfolio prioritization. Regression outputs also enable project managers to compare expected delay severity across projects with different characteristics and planned durations [14,16,24].

Regression models are commonly evaluated using complementary error-based and explanatory measures. Mean Absolute Error measures the average absolute difference between actual and predicted values, whereas Root Mean Squared Error gives greater weight to larger prediction errors. The squared correlation measures the proportion of variation in the target explained by the model, while correlation evaluates the strength of association between actual and predicted outcomes [18,23].

Compared with classification-oriented research, fewer project-delay studies have estimated delay as a continuous numerical value. Existing regression studies have predicted outcomes such as time-delay magnitude and schedule-to-completion, but several have relied on small or specialized datasets. Furthermore, limited research has compared traditional regression, tree-based ensemble, kernel-based, and deep-learning models using the same continuous schedule-overrun target [19–24].

Regression-based prediction also introduces methodological challenges. The target variable must be measured consistently across projects, and model performance may be affected by extreme values, missing information, and skewed or unevenly distributed target values. Moreover, the managerial significance of a prediction error may differ according to the planned duration and context of the project. Regression performance should therefore be assessed using multiple complementary metrics rather than relying on a single measure [18,23].

The present study formulates schedule-overrun prediction as a supervised continuous regression problem. The target variable, `Schedule_Overrun_%`, represents the percentage by which actual project duration exceeds planned duration. This formulation preserves the numerical magnitude of schedule deviation and enables the evaluated models to distinguish between different levels of overrun without reducing the outcome to predefined categories.

The continuous formulation also supports the direct and consistent comparison of Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network models using RMSE, MAE, $R^2$, and correlation. The empirical studies supporting the distinction between classification- and regression-based delay predictions are reviewed separately in Section 2.8.

### *2.6 Machine Learning Approaches for Schedule Overrun Prediction*

Machine-learning algorithms differ in the assumptions they make, the relationships they can capture, their sensitivity to data characteristics, and their level of interpretability. Comparing models from different methodological families is therefore important when the structure of the prediction problem is not known in advance. A linear model may provide a strong and

interpretable baseline, whereas tree-based, ensemble, kernel-based, and deep-learning models may capture more complex nonlinear relationships and interactions among project variables [18].

In schedule-overrun prediction, model performance may depend on the combined effects of project duration, complexity, team characteristics, risk exposure, unresolved issues, change requests, and other delivery conditions. Some of these relationships may be approximately linear, while others may involve thresholds, interactions, or nonlinear effects. Evaluating multiple algorithmic approaches under the same dataset and validation framework enables a more reliable assessment of which model is best suited to the prediction task.

The present study evaluates six regression models representing distinct learning approaches: Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. The following subsections summarize the theoretical characteristics, strengths, and limitations of each model in the context of schedule-overrun prediction.

### *2.6.1 Linear Regression*

Linear Regression is a supervised statistical-learning method that models the relationship between a continuous dependent variable and one or more independent variables. It assumes that the target can be represented as a linear combination of the predictor variables and an error term [18].

In the context of schedule-overrun prediction, the model estimates the expected change in `Schedule_Overrun_%` associated with changes in project-related variables while holding the remaining predictors constant. For example, the model may estimate how increases in project complexity, risk exposure, unresolved issues, or change requests are associated with changes in the expected schedule-overrun percentage.

The general multiple linear-regression model can be expressed as:

[
\hat{y}=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_px_p
]

where (\hat{y}) represents the predicted schedule-overrun percentage, (\beta_0) is the intercept, (\beta_1,\ldots,\beta_p) are the estimated coefficients, and (x_1,\ldots,x_p) represent the predictor variables.

One of the principal advantages of Linear Regression is its interpretability. The sign and magnitude of each coefficient provide a direct indication of the estimated relationship between a predictor and the target variable. This characteristic makes the model useful as a baseline and supports managerial interpretation of how project factors are associated with schedule performance. Linear Regression is also computationally efficient and can perform well when the underlying relationships are approximately linear and the data satisfy the model assumptions. Its results are relatively easy to communicate to project managers and governance stakeholders because each prediction is derived from a transparent weighted combination of input variables.

However, the model has important limitations in complex project environments. It may not adequately capture nonlinear relationships, interaction effects, threshold behavior, or changing impacts across different project conditions. For example, the effect of unresolved issues may become substantially more severe only when combined with high project complexity or limited team capacity. Such conditional relationships are difficult to represent through a conventional linear model unless interaction terms or transformations are specified explicitly.

Linear Regression may also be sensitive to multicollinearity, extreme observations, heteroscedasticity, and violations of the assumption that the relationship between predictors and the target is linear. Consequently, its performance in this study provides an interpretable benchmark against which the more flexible machine-learning and deep-learning models can be evaluated [18].

### *2.6.2 Regression Tree*

A Regression Tree is a supervised machine-learning method designed to predict continuous numerical outcomes. Unlike Linear Regression, which estimates the target using a single linear equation, a Regression Tree divides the predictor space into a set of smaller and more homogeneous regions. Each observation is assigned to one of these regions according to a sequence of decision rules, and the predicted value is typically calculated as the average target value of the training observations located within the same terminal region [18].

The model begins with the complete dataset and repeatedly selects a predictor and a split point that produce the greatest reduction in prediction error. For regression problems, the splitting process commonly seeks to reduce the residual sum of squares within the resulting regions. The procedure continues recursively until a stopping condition is reached or the tree is subsequently reduced through pruning [18].

A Regression Tree prediction can be expressed as:

[
\hat{f}(x)=\sum_{m=1}^{M}c_m I(x\in R_m)
]

where (R_m) represents one of the (M) terminal regions created by the tree, (c_m) is the predicted value assigned to that region, and (I(x\in R_m)) is an indicator showing whether an observation with characteristics (x) belongs to region (R_m) [18].

The resulting model has a hierarchical structure. Internal nodes represent decision conditions, branches represent the outcomes of those conditions, and terminal nodes contain the final numerical predictions. For example, an initial split may separate projects according to project complexity, followed by additional splits based on risk exposure, unresolved issues, planned duration, or change requests.

In schedule-overrun prediction, a tree may learn that projects with high complexity and a large number of unresolved issues are associated with a higher expected schedule-overrun percentage, whereas projects with lower complexity, fewer risks, and limited change activity are associated with a lower predicted value. Unlike a linear model, the tree does not require the effect of a predictor to remain constant across all observations.

One of the principal advantages of Regression Trees is their ability to capture nonlinear relationships and interaction effects without requiring these relationships to be specified explicitly in advance. A predictor may influence the target differently depending on the values of other predictors. Tree-based partitioning can identify such conditional and threshold-based relationships directly from the training data [18].

Regression Trees are also relatively interpretable because their predictions can be represented as a sequence of understandable decision rules. They can accommodate numerical and categorical predictors and generally require fewer distributional assumptions than conventional linear models [18]. In project-management applications, these characteristics can help decision-makers understand the combinations of project conditions associated with different predicted levels of schedule overrun.

However, an individual Regression Tree may be unstable. Small changes in the training data can result in different split points and a substantially different tree structure. A highly complex tree may closely fit the training observations and overfit the data, whereas an excessively shallow tree may fail to represent important relationships and underfit the prediction problem [18].

Tree complexity is therefore commonly controlled through stopping criteria or pruning. Pruning removes branches that provide limited improvement in predictive performance and helps balance model complexity against the ability to generalize to unseen observations. The preferred tree size may be selected through validation or cross-validation procedures [18].

Nevertheless, even a pruned Regression Tree may provide lower predictive accuracy and stability than ensemble methods that combine multiple trees. Random Forest and Gradient Boosting approaches were developed partly to address the variance and instability associated with individual decision trees [18].

In the present study, the Regression Tree serves as an interpretable nonlinear benchmark. Its performance is compared with Linear Regression to assess whether rule-based partitioning improves schedule-overrun prediction. It is also compared with Random Forest and Gradient Boosting Machine models to determine whether combining multiple trees provides greater predictive accuracy and stability.

### *2.6.3 Random Forest*

Random Forest is an ensemble-learning method that combines predictions from multiple decision trees to improve predictive accuracy and reduce the instability associated with a single Regression Tree. Each tree is trained using a bootstrap sample drawn from the original training dataset, while only a randomly selected subset of predictor variables is considered at each potential split [18].

The use of bootstrap samples creates variation among the individual trees because each tree is developed from a different resampled version of the training data. Random predictor selection introduces additional diversity by preventing the same strong predictors from dominating every tree. These mechanisms reduce the correlation among the individual trees and allow their combined predictions to generalize more effectively to unseen observations [18].

For a regression problem, the Random Forest prediction can be expressed as:

[
\hat{f}{RF}(x)=\frac{1}{B}\sum{b=1}^{B}\hat{f}^{(b)}(x)
]

where (B) represents the total number of trees, (\hat{f}^{(b)}(x)) is the prediction generated by the (b)-th tree for an observation (x), and (\hat{f}_{RF}(x)) is the average prediction produced by the complete forest [18].

Averaging predictions across multiple trees reduces model variance. An individual tree may generate substantially different predictions when small changes occur in the training data, but the average output of a large collection of partially independent trees is generally more stable. Random Forest therefore commonly provides stronger generalization performance than a single Regression Tree while retaining the ability to model nonlinear relationships and complex interactions [18].

In schedule-overrun prediction, different trees may identify different combinations of project conditions. One tree may emphasize project complexity and risk exposure, while another may place greater importance on unresolved issues, planned duration, team characteristics, or change requests. The final prediction represents the average of these different tree-based perspectives.

Random Forest does not require the relationship between the predictors and the target to be linear. It can represent threshold effects, interactions, and conditional relationships without requiring them to be defined manually. For example, the effect of

change requests may differ according to project complexity, issue severity, or the stage at which the changes occur. Such relationships can be captured through the recursive partitioning structure of the individual trees [18].

Another advantage of Random Forest is its relative robustness to noise and overfitting compared with an individual decision tree. Although each tree may fit a particular bootstrap sample closely, the averaging process reduces the influence of extreme or unstable predictions. Increasing the number of trees generally stabilizes the forest, although it also increases computational requirements [18].

Random Forest can also provide measures of variable importance. These measures estimate the extent to which individual predictors contribute to reducing prediction error or improving the quality of the tree splits. In project-management applications, feature importance can help identify project characteristics that are strongly associated with schedule performance. However, importance values should be interpreted carefully because correlated predictors may share or distort the measured contribution of one another [18].

Despite its predictive strengths, Random Forest has limitations. The combined model is less transparent than a single Regression Tree because its prediction is generated by a large number of trees rather than one interpretable sequence of decision rules. Although feature-importance measures provide an overall indication of predictor influence, they do not fully explain the logic behind an individual prediction [18].

Random Forest may also require tuning of parameters such as the number of trees, maximum tree depth, minimum observations per terminal node, and number of predictors considered at each split. Poor parameter selection may lead to unnecessary computational complexity or limit the model's ability to capture relevant patterns.

In the present study, Random Forest represents a bagging-based ensemble approach. Its performance is compared with the individual Regression Tree to assess whether averaging multiple randomized trees improves prediction accuracy and stability. It is also compared with Gradient Boosting Machine, which combines trees sequentially rather than independently, and with the remaining regression models under the same validation and evaluation framework.

### *2.6.4 Support Vector Regression*

Support Vector Regression (SVR) is a supervised machine-learning method that extends the principles of Support Vector Machines to the prediction of continuous numerical outcomes. Rather than attempting to minimize the prediction error for every training observation directly, SVR seeks to construct a function that is sufficiently flat while allowing prediction errors within a predefined tolerance level [18,26].

The general SVR prediction function can be expressed as:

[
f(x)=w^{T}\phi(x)+b
]

where (x) represents the predictor variables, (\phi(x)) maps the original predictors into a transformed feature space, (w) represents the model weights, and (b) is the intercept. The transformation may be performed implicitly through a kernel function, enabling the model to represent nonlinear relationships without calculating the transformed feature space explicitly [18,26].

A central feature of SVR is the epsilon-insensitive loss function:

[
\max\left(0,\left|y-f(x)\right|-\varepsilon\right)
]

where (y) is the actual outcome, (f(x)) is the predicted value, and (\varepsilon) defines an acceptable prediction-error margin. Errors that fall within the epsilon margin are not penalized, whereas deviations outside the margin contribute to the model's loss function [26].

Conceptually, SVR attempts to fit a regression function inside an epsilon-wide tube surrounding the observed target values. Training observations located within this tube do not affect the fitted function directly. Observations located on or outside its boundaries become support vectors and contribute to determining the final prediction function. This property can produce a relatively sparse model because only a subset of the training observations influences the fitted solution [26].

The parameter (C) controls the trade-off between model flatness and the penalty assigned to observations outside the epsilon margin. A large value of (C) imposes a stronger penalty on prediction errors and may cause the model to fit the training data more closely. A smaller value allows greater deviation in exchange for a smoother and potentially more generalizable prediction function. The value of (\varepsilon) determines the width of the error-insensitive region and therefore influences the number of observations that become support vectors [26].

Kernel functions allow SVR to capture nonlinear relationships. Common choices include linear, polynomial, radial basis function, and sigmoid kernels. The radial basis function kernel is frequently used when the relationship between predictors and the target is not expected to be linear. Its kernel parameter, commonly represented by (\gamma), controls the extent to which individual observations influence the prediction surface [18,26].

In schedule-overrun prediction, SVR can model complex relationships among project duration, complexity, risks, unresolved issues, team characteristics, and change requests. A kernel-based model may identify nonlinear patterns in which the effect of one project factor varies depending on the values of other factors.

SVR offers several potential advantages. It can represent nonlinear relationships, operate effectively in high-dimensional predictor spaces, and control model complexity through the combination of the margin, regularization parameter, and kernel settings. The epsilon-insensitive loss function may also reduce the influence of small prediction deviations that are not operationally significant [26].

However, SVR performance is highly dependent on parameter selection. The regularization parameter (C), epsilon margin, kernel type, and kernel parameters must be selected carefully. Inadequate tuning may result in underfitting or overfitting. Cross-validation is therefore commonly used to identify parameter combinations that provide stronger generalization performance [18,26].

SVR is also sensitive to differences in predictor scale because variables with larger numerical ranges may disproportionately influence distance-based kernel calculations. Predictor normalization or standardization is therefore generally required before model training. Furthermore, SVR is less interpretable than Linear Regression or an individual Regression Tree because its predictions are based on support vectors and kernel transformations rather than directly observable coefficients or decision rules [18,26].

The computational cost of SVR may increase as the number of training observations and support vectors grows. This consideration can become important for large datasets or extensive parameter-search procedures. Nevertheless, SVR remains a useful benchmark for determining whether margin-based and kernel-based learning can improve schedule-overrun prediction relative to linear and tree-based approaches.

In the present study, SVR represents the kernel-based regression family. Its performance is evaluated using the same dataset, validation process, and performance measures applied to the other models. This enables a direct assessment of whether its margin-based prediction strategy provides competitive performance for estimating `Schedule_Overrun_%`.

#### *2.6.5 Gradient Boosting Machine*

Gradient Boosting Machine (GBM) is an ensemble-learning method that constructs a strong predictive model by combining a sequence of relatively weak learners, typically shallow decision trees. Unlike Random Forest, in which individual trees are trained largely independently and their predictions are averaged, GBM builds trees sequentially. Each new tree is trained to improve the errors produced by the existing ensemble [18,27].

The boosting process begins with an initial prediction, which may be the average value of the target variable in a regression problem. The algorithm then calculates the difference between the observed and predicted outcomes. A new tree is fitted to information derived from these errors, and its contribution is added to the existing model. This procedure is repeated over multiple iterations so that the ensemble progressively focuses on patterns that were not adequately captured by the preceding trees [18,27].

In the general gradient-boosting framework, the model is improved by minimizing a selected loss function through gradient descent in function space. At each iteration, a weak learner is fitted to the negative gradient of the loss function evaluated using the current model. For squared-error regression, these negative gradients correspond to the residuals between the observed and predicted values [27].

The final GBM prediction can be represented as:

[F_M(x)F_0(x)+\nu\sum_{m=1}^{M}\gamma_m h_m(x)]

where (F_0(x)) is the initial prediction, (M) is the total number of boosting iterations, (h_m(x)) is the decision tree fitted during iteration (m), (\gamma_m) represents the contribution of that tree, and (\nu) is the learning rate controlling the contribution of each new tree [18,27].

The learning rate, also referred to as shrinkage, reduces the contribution of each individual tree. A smaller learning rate generally requires a larger number of trees but allows the model to improve more gradually. This gradual learning process may enhance generalization by preventing individual trees from making excessively large corrections. However, the learning rate and number of trees must be considered jointly because an inadequate combination may result in underfitting or overfitting [18,27].

Tree depth is another important parameter. Shallow trees represent relatively simple relationships and low-order interactions, whereas deeper trees can capture more complex interactions among predictors. In schedule-overrun prediction, these interactions may include conditions in which project complexity becomes particularly influential when combined with high risk exposure, unresolved issues, or repeated change requests.

GBM is well suited to structured project data because it can model nonlinear relationships, threshold effects, and interactions without requiring them to be specified manually. Each successive tree can focus on projects or conditions that were predicted poorly by the preceding ensemble. Consequently, the model can progressively distinguish among different combinations of project characteristics associated with varying levels of schedule overrun [18,27].

For example, an initial tree may identify a general relationship between project complexity and schedule overrun. A subsequent tree may correct errors for projects with high risk exposure, while another may improve predictions for projects containing unresolved issues or extensive change activity. The final prediction incorporates the accumulated corrections produced across all trees.

Another advantage of GBM is its flexibility regarding the choice of loss function. The original gradient-boosting framework can be adapted to squared-error loss, absolute-error loss, robust loss functions, and classification-related loss functions. This allows the learning procedure to be aligned with the characteristics and objectives of the prediction problem [27].

GBM can also provide measures of feature importance based on how frequently and effectively individual predictors contribute to error-reducing splits across the ensemble. In project-management applications, these measures can help identify which project attributes contribute most strongly to schedule-overrun prediction. Feature importance therefore adds a degree of interpretability to an otherwise complex ensemble model.

However, feature-importance values should not automatically be interpreted as causal effects. A variable may receive high importance because it provides useful predictive information, but this does not prove that changing the variable will directly change the project outcome. Importance may also be distributed among correlated predictors, which should be considered when interpreting the results.

GBM has several limitations. Because trees are trained sequentially, the training process may be slower than methods in which trees can be developed independently. The model is also sensitive to hyperparameters such as the number of trees, learning rate, maximum tree depth, minimum leaf size, and loss function. An excessively complex model may overfit the training data, whereas an overly restricted model may fail to capture important patterns [18,27].

Cross-validation is therefore commonly used to determine an appropriate balance between the number of trees, learning rate, and tree complexity. Regularization through shrinkage, shallow trees, subsampling, or early stopping may further improve the model's ability to generalize to unseen observations [18,27].

In the present study, GBM represents the sequential tree-based ensemble family. Its performance is compared with the individual Regression Tree and the bagging-based Random Forest model to determine whether iterative error correction improves schedule-overrun prediction. GBM feature importance is also examined to identify the project variables that contribute most strongly to the resulting predictions.

### *2.6.6 Deep Neural Network*

A Deep Neural Network (DNN) is a supervised learning model composed of an input layer, multiple hidden layers, and an output layer. The term "deep" refers to the use of several successive layers that transform the original input variables into increasingly abstract internal representations before producing the final prediction [18,28].

Each layer consists of interconnected computational units, commonly referred to as neurons. A neuron receives values from the preceding layer, multiplies them by learned weights, adds a bias term, and applies an activation function. For a single neuron, this operation can be expressed as:

[
z_j=\sum_{i=1}^{n}w_{ji}x_i+b_j
]

[
a_j=\phi(z_j)
]

where (x_i) represents an input from the preceding layer, (w_{ji}) is the weight connecting input (i) to neuron (j), (b_j) is the bias term, (z_j) is the weighted input, (\phi(\cdot)) is the activation function, and (a_j) is the resulting neuron output [18,28].

The outputs generated by one layer become the inputs to the following layer. Through this layered structure, a DNN can learn hierarchical representations and complex relationships among predictor variables. The final output layer produces the numerical prediction required for a regression problem, such as the expected schedule-overrun percentage [18,28].

Activation functions introduce nonlinearity into the network. Without nonlinear activation functions, multiple layers would remain equivalent to a single linear transformation and would not provide the flexibility normally associated with deep learning. Common activation functions include the sigmoid function, hyperbolic tangent, and Rectified Linear Unit (ReLU), with ReLU frequently used in hidden layers because of its computational simplicity and effectiveness in gradient-based training [18,28].

For a regression problem, the complete network can be represented conceptually as:

[
\hat{y}
f^{(L)}
\left(
f^{(L-1)}
\left(
\cdots

f^{(2)}
\left(
f^{(1)}(x)
\right)
\right)
\right)
]
where (x) represents the input project variables, (f^{(1)},\ldots,f^{(L)}) represent the transformations learned by the successive network layers, and (\hat{y}) represents the predicted schedule-overrun percentage [18].
During training, the network generates predictions and compares them with the actual target values through a loss function. For regression, the loss may be based on squared prediction error. The objective of training is to identify network weights and biases that minimize this loss across the training observations [18,28].
The error is propagated backward through the network using the backpropagation algorithm. Backpropagation applies the chain rule to calculate how much each parameter contributes to the prediction error. An optimization algorithm then updates the weights and biases in the direction expected to reduce the loss. This process is repeated over multiple training iterations or epochs [18,28].
The general parameter-update process can be represented as:
[\theta_t\eta\nabla_{\theta}L(\theta_t)]
where (\theta_t) represents the network parameters at iteration (t), (L) is the loss function, (\nabla_{\theta}L) is the gradient of the loss with respect to the parameters, and (\eta) is the learning rate controlling the size of each update [18,28].
In schedule-overrun prediction, the input layer may receive project variables such as planned duration, complexity, team characteristics, risk exposure, unresolved issues, and change requests. The hidden layers can learn combinations and interactions among these variables, while the output layer generates the estimated value of `Schedule_Overrun_%`.
One of the principal advantages of DNNs is their ability to approximate complex nonlinear relationships. Multiple hidden layers can represent interactions and patterns that may be difficult to capture through a single linear equation or manually specified transformation. This flexibility can be useful when schedule overrun results from the combined effects of several project characteristics rather than from one factor acting independently [18,28].
DNNs can also learn intermediate representations automatically. Instead of requiring the researcher to specify every possible interaction among project variables, the network can learn weighted combinations of the available inputs during training. This may help identify predictive structures that are not immediately apparent through conventional statistical analysis [28].
However, the flexibility of a DNN also increases the risk of overfitting. A network containing many layers and parameters may learn noise or project-specific patterns in the training data rather than relationships that generalize to unseen observations. Regularization techniques such as weight penalties, dropout, early stopping, and validation-based model selection can be used to reduce this risk [18,28].
DNN performance is also influenced by architectural and training choices, including the number of hidden layers, neurons per layer, activation functions, learning rate, batch size, optimizer, number of epochs, and regularization settings. An inadequate configuration may result in underfitting, overfitting, unstable learning, or slow convergence. Model development therefore commonly requires systematic tuning and validation [18,28].
Another limitation concerns interpretability. The prediction produced by a DNN results from transformations distributed across multiple layers and a potentially large number of learned parameters. Consequently, it is generally more difficult to explain than a Linear Regression coefficient or an individual tree-based decision rule. This black-box characteristic can limit managerial understanding of why a particular schedule-overrun value was predicted [18,28].
DNNs may also require more computational effort and data than simpler models. Their advantages are not guaranteed for every structured dataset, particularly when the dataset is moderate in size or when tree-based algorithms can represent the relevant relationships effectively. Deep learning performance should therefore be evaluated empirically rather than assumed to be superior solely because the model is more complex [18].
In the present study, the DNN represents the deep-learning family and is evaluated under the same validation framework used for the five machine-learning models. Its performance is compared with GBM and the other algorithms to determine whether its multilayer nonlinear structure provides improved schedule-overrun prediction. Because direct interpretation of the internal DNN parameters is limited, the study relies primarily on predictive-performance measures and actual-versus-predicted analysis when assessing the model.

### *2.7 Explainability in AI-Based Project Prediction*

Predictive accuracy is an essential requirement for AI-based project forecasting, but it is not sufficient on its own. Project managers and governance stakeholders also need to understand the factors contributing to a prediction, particularly when the model indicates a substantial schedule overrun or recommends management intervention. Explainability therefore concerns the

extent to which the reasoning, influential variables, or internal behavior of a predictive model can be understood by human decision-makers [17].

In project-management environments, explainability has practical importance because model outputs may influence resource allocation, escalation, contingency planning, vendor management, and schedule-recovery decisions. A forecast indicating that a project is expected to exceed its planned duration may have limited managerial value when the responsible stakeholders cannot determine whether the prediction is associated primarily with project complexity, unresolved issues, risk exposure, change activity, resource constraints, or another delivery condition [17].

Explainability can generally be considered at two levels. Global explainability describes the overall behavior of a model across the complete dataset and identifies the variables that contribute most strongly to its predictions. Local explainability focuses on an individual project and attempts to explain why the model generated a particular prediction for that specific observation [17].

Model interpretability varies substantially across machine-learning approaches. Linear Regression is relatively transparent because each coefficient provides an estimated direction and magnitude of association between a predictor and the target, subject to the assumptions of the model. An individual Regression Tree is also interpretable because its prediction can be traced through a sequence of visible decision rules [18].

Ensemble tree models, such as Random Forest and Gradient Boosting Machine, are more complex because they combine predictions from many individual trees. Their complete decision process cannot usually be represented through one simple rule set. However, they can provide global feature-importance measures that summarize how strongly different predictors contribute to the prediction process [18,27].

Feature importance in tree-based models is commonly derived from the contribution of each variable to improving the quality of the tree splits or reducing prediction error across the ensemble. Variables that repeatedly generate effective splits may receive higher importance values. In schedule-overrun prediction, such an analysis may indicate whether project complexity, risk-related variables, issue-related variables, change activity, planned duration, or team characteristics contribute strongly to the model's predictive performance [18,27].

Feature importance can help translate an accurate predictive model into actionable project-management insight. For example, if unresolved issues and change requests receive high importance values, project managers may strengthen issue-resolution and change-control practices. If planned duration or project complexity is highly influential, greater attention may be given to estimation quality, dependency analysis, and delivery planning.

Nevertheless, feature-importance values must be interpreted carefully. Importance indicates that a variable contributes useful predictive information, but it does not demonstrate that the variable causes schedule overrun. A highly ranked feature may be associated with the outcome without being directly responsible for it. Consequently, predictive importance should not be interpreted as causal evidence [17,18].

Correlated predictors may also affect the distribution of importance. When two variables contain similar information, the model may divide their contribution between them or assign greater importance to one while reducing the apparent importance of the other. The ranking should therefore be interpreted in relation to the dataset structure, project context, and relationships among the predictors rather than as an absolute measurement of managerial significance [18].

Deep Neural Networks generally present greater explainability challenges. Their predictions are generated through multiple layers, activation functions, and learned parameter combinations. Although this structure allows the model to capture complex nonlinear relationships, it also makes the internal reasoning difficult to express through directly interpretable coefficients or decision rules [28].

The limited transparency of deep models is often described as a black-box characteristic. A DNN may generate an accurate schedule-overrun prediction without providing a straightforward explanation of how each project variable contributed to the final result. This limitation can affect managerial trust, particularly when the prediction differs from professional judgment or when it leads to a significant governance decision [17,28].

Model explainability should therefore be considered together with predictive performance. The most accurate model may not necessarily be the most suitable for every project-management application. A slightly less accurate but more interpretable model may be preferable when transparency, accountability, and stakeholder communication are essential. Conversely, a more complex model may be justified when its improvement in predictive accuracy is substantial and its outputs can be supported by complementary explanation techniques [17].

Explainability does not replace professional interpretation. Model explanations reflect patterns learned from historical data and may be influenced by data quality, missing variables, and measurement practices. Project managers must assess whether the identified features are consistent with the project context and whether additional information outside the dataset should influence the final decision.

In the present study, explainability is addressed primarily through GBM feature-importance analysis. GBM was selected for this purpose because it combines strong nonlinear predictive capability with the ability to estimate the relative contribution of the input variables. The feature-importance results are used to identify which project characteristics contribute most strongly to schedule-overrun prediction.

Direct feature-importance analysis is not used as the primary interpretation method for the DNN because its internal multilayer transformations are substantially less transparent. The DNN is therefore evaluated mainly through RMSE, MAE, $R^2$, correlation, and actual-versus-predicted analysis. This approach enables the study to compare predictive accuracy while also recognizing the interpretability trade-off between ensemble tree models and deep neural networks.

### *2.8 Empirical Studies on Project Delay Prediction*

Recent advances in artificial intelligence have encouraged researchers to investigate the use of machine-learning and deep-learning methods for predicting project delays. Existing empirical studies differ substantially in terms of project domain, dataset size, data source, prediction target, algorithm selection, and evaluation methodology.

A substantial proportion of the previous literature has focused on construction projects and has formulated delay prediction as either a binary classification problem or a multiclass delay-severity problem. Comparatively fewer studies have estimated delay as a continuous numerical outcome. This distinction is important because classification models identify whether delay is expected or assign a project to a predefined risk category, whereas regression models estimate the numerical magnitude of the expected schedule deviation.

The following subsections organize the reviewed empirical literature into traditional machine-learning studies, ensemble and hybrid learning studies, deep-learning studies, and review or conceptual studies. The analysis considers not only the reported predictive performance but also the sample size, prediction type, project domain, model-comparison design, and generalizability of each study.

#### *2.8.1 Traditional Machine Learning Studies*

Gondia et al. [19] developed machine-learning models to predict delay risk in construction projects using historical project information and delay-related risk factors. The study evaluated Decision Tree and Naïve Bayes classifiers using data from 51 construction projects. The prediction problem was formulated as a multiclass classification task in which projects were assigned to predefined delay-risk categories rather than being given a continuous estimate of schedule overrun.

Although both algorithms demonstrated stronger performance during model development, their predictive accuracy decreased when they were applied to unseen testing data. Naïve Bayes achieved a testing accuracy of approximately 51.2%, while the Decision Tree achieved approximately 47.2%. The decline in testing performance suggested that the models had limited ability to generalize beyond the relatively small training sample [19].

The study demonstrated that conventional classification algorithms can support the early identification of project delay risk. However, its small construction-specific dataset limits the generalizability of the results. Furthermore, the categorical target does not indicate the exact magnitude of the anticipated schedule overrun, which reduces its usefulness for detailed schedule forecasting, contingency calculation, and recovery planning [19].

Egwim and Alaka [29] presented a broader comparison of 27 machine-learning classification algorithms for predicting construction-project delay. The evaluated approaches included Perceptron, Support Vector Machine, Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, AdaBoost, XGBoost, and LightGBM. The analysis was based on 120 survey responses describing factors associated with construction delays.

The Perceptron model achieved the strongest reported overall performance, with accuracy, balanced accuracy, ROC-AUC, and F1-score values of approximately 85%. Random Forest also performed more strongly than the standalone Decision Tree, indicating that combining multiple trees may improve predictive stability relative to relying on an individual tree [29].

Despite evaluating a wide range of algorithms, the study remained limited by its survey-based sample and binary prediction target. The resulting models identified whether a project was expected to experience delay but did not estimate the percentage or number of days by which its schedule would be exceeded. In addition, the relatively small and geographically restricted sample limits the extent to which the findings can be generalized to broader technical-project environments [29].

Collectively, these studies demonstrate that traditional machine-learning algorithms can support early delay-risk identification. However, their results are based mainly on small construction datasets and classification-oriented targets. Consequently, they provide limited evidence regarding the ability of machine-learning models to estimate the continuous magnitude of schedule overrun required for quantitative schedule-control decisions [19,29].

#### *2.8.2 Ensemble and Hybrid Learning Studies*

Ensemble-learning methods combine the predictions of multiple individual models to improve predictive accuracy, reduce model instability, and capture relationships that may not be represented adequately by a single learner. Previous project-delay studies have examined bagging, boosting, stacking, and hybrid optimization approaches, particularly within construction-related applications [20,21].

Egwim et al. [20] investigated the use of ensemble machine-learning methods for predicting construction-project delay. The study was based on 120 survey responses and evaluated several standalone and ensemble classifiers, including Decision Tree, Random Forest, Bagging, Extra Trees, AdaBoost, Gradient Boosting Machine, XGBoost, and different Naïve Bayes variants. The prediction task was formulated as a binary classification problem that distinguished between delayed and non-delayed projects [20].

The study also developed a stacking-based ensemble that combined the outputs of multiple base classifiers. The proposed ensemble achieved a reported classification accuracy of approximately 85.5%, outperforming several of the individual learning models evaluated under the same experimental setting [20]. This result suggested that combining diverse algorithms could improve delay-risk classification by allowing the ensemble to benefit from complementary predictive patterns.

However, the study relied on a relatively small survey-based sample and focused on construction-project conditions within a restricted context. Its binary target identified whether a delay was expected but did not estimate the numerical magnitude of the schedule deviation. Therefore, although the model could support early delay-risk identification, it offered limited information for estimating expected delay duration or schedule-overrun percentage [20].

The work of Egwim et al. [20] is closely related to the comparative study by Egwim and Alaka [29]. Both studies were based on a sample of 120 responses and examined similar construction-delay factors. Consequently, their findings should not be treated as completely independent empirical evidence, even though the studies evaluated different model combinations and analytical objectives. Recognizing this overlap is important when assessing the overall strength and diversity of evidence in the literature.

Yaseen et al. [21] proposed a hybrid artificial-intelligence model for predicting delay-risk categories in construction projects. The study used information from 40 construction projects and compared a conventional Random Forest model with a hybrid Random Forest–Genetic Algorithm approach. The Genetic Algorithm was applied to optimize aspects of the Random Forest model and improve its predictive performance [21].

The hybrid model classified projects into three delay-risk categories and achieved a reported testing accuracy of approximately 91.67%, compared with approximately 75% for the standalone Random Forest model [21]. The results indicated that combining an ensemble learner with an optimization algorithm could improve classification performance by identifying more suitable model configurations or predictor combinations.

Despite the strong reported accuracy, the testing set contained only 12 project observations. A high percentage calculated from such a small number of cases may change substantially when one or two observations are classified differently. The limited sample size therefore reduces confidence in the stability and generalizability of the reported result [21].

The study was also classification-oriented. It predicted predefined delay-risk categories rather than the continuous magnitude of schedule overrun. As a result, the model could distinguish between different levels of delay risk but could not directly estimate whether a project would exceed its planned duration by a specific percentage or number of days [21].

Nassar and Elbisy [23] examined continuous time-delay prediction in marine construction projects. Their study considered 43 delay-related factors collected from 30 projects and 103 experts across several Middle Eastern countries. The evaluated models included a General Regression Neural Network, Support Vector Machine, and Tree Boost model [23].

Unlike many previous studies, the prediction task was formulated as a regression problem. The General Regression Neural Network achieved the strongest reported performance, with an RMSE of approximately 7.321, an MAE of approximately 7.050, and a correlation coefficient of approximately 0.926 [23]. This study is particularly relevant because it demonstrates the feasibility of estimating project delay as a continuous outcome rather than merely assigning projects to categorical risk levels.

Nevertheless, the empirical sample remained relatively small and was restricted to marine construction projects. The use of expert assessments also means that part of the predictive information was influenced by professional perceptions rather than being derived exclusively from standardized historical project records. These characteristics may limit the transferability of the findings to broader technical-project portfolios [23].

The evaluation also compared only a limited set of regression approaches. It did not provide a broad benchmark covering linear, individual tree, bagging-based ensemble, boosting-based ensemble, kernel-based, and deep-learning models within one unified experimental framework. Therefore, the relative advantage of different model families remained only partially examined [23].

Overall, ensemble and hybrid studies generally reported stronger performance than individual baseline models, particularly when multiple trees, boosting mechanisms, stacking strategies, or optimization algorithms were employed [20,21]. However, most of these studies continued to focus on construction-related contexts, small samples, and categorical delay prediction. Although Nassar and Elbisy [23] addressed continuous regression, their specialized and limited dataset leaves a need for larger-scale comparative research on numerical schedule-overrun prediction across technical projects.

### *2.8.3 Deep Learning Studies*

Deep-learning approaches have increasingly been investigated for project-delay prediction because of their ability to represent nonlinear relationships, complex interactions, and temporal dependencies. However, existing studies differ in how they formulate the prediction target and in the type of project information processed by the models [22,24].

Alsulamy [22] compared three deep-learning approaches for predicting construction-project delays in Saudi Arabia: Generative Adversarial Networks (GAN), Long Short-Term Memory networks (LSTM), and Multilayer Perceptrons (MLP). The study used data collected from 62 construction projects and categorized time overrun into three classes: minor delay below 30%, moderate delay between 30% and 60%, and major delay above 60% [22].

The study evaluated the models using classification measures including accuracy, precision, sensitivity, specificity, and misclassification error. GAN achieved the strongest reported accuracy of 91%, followed by LSTM at 88% and MLP at 83%. The corresponding misclassification errors were 9%, 12%, and 17%, respectively [22].

The GAN approach was particularly useful for addressing class imbalance because it generated synthetic observations to improve the representation of underrepresented delay categories. LSTM was used for its ability to capture temporal dependencies, whereas MLP provided a more conventional neural-network benchmark [22].

Although the reported results demonstrate the potential of deep-learning models for construction-delay assessment, the study remained a multiclass classification analysis. Its predictions assigned projects to broad delay-severity categories rather than estimating the exact continuous percentage of schedule overrun. For example, two projects with time overruns of 31% and 58% would both be classified as moderate delay despite the substantial difference between their actual outcomes [22].

The study was also based on a relatively small construction-specific sample. Although synthetic data generation may help address class imbalance, it does not replace the need for a large and diverse collection of original project records. The ability of the reported models to generalize to technical projects or other project environments therefore requires further investigation.

Cheng et al. [24] developed a hybrid deep-learning model to estimate the final cost and schedule to completion of construction projects. Their approach combined a conventional Neural Network for processing time-independent project attributes with a Bidirectional Gated Recurrent Unit for processing sequential and time-dependent information. The combined architecture was referred to as NN-BiGRU [24].

The researchers further optimized the hybrid architecture using the Optical Microscope Algorithm, producing the OMA-NN-BiGRU model. The proposed approach achieved a reported Reference Index of 0.977 for final-cost estimation and 0.932 for schedule-to-completion estimation, outperforming the alternative models included in their analysis [24].

An important contribution of the study was its distinction between time-independent and time-dependent project data. Static project characteristics were processed through the Neural Network component, while evolving project information and temporal dependencies were processed through the BiGRU component. This enabled the model to integrate different forms of project information within one predictive architecture [24].

Nevertheless, the prediction target differed from the target used in the present research. Cheng et al. estimated the remaining schedule or schedule to completion, whereas the present study predicts the percentage by which actual project duration exceeds planned duration. Schedule-to-completion estimation and schedule-overrun-percentage prediction are related but distinct forecasting problems.

The hybrid architecture also involved several integrated components, including a Neural Network, BiGRU, and an optimization algorithm. Although this complexity may improve predictive capability, it can increase computational requirements, tuning effort, and difficulty of interpretation. These considerations may affect the practical deployment of the model in project-management environments in which transparency and ease of implementation are important.

Overall, the reviewed deep-learning studies demonstrate that neural architectures can model complex project-delay patterns and may provide strong predictive performance [22,24]. However, the available evidence remains concentrated in construction applications and includes different prediction targets, such as categorical delay severity and schedule to completion. Consequently, direct comparison with continuous schedule-overrun-percentage prediction remains limited.

The present study addresses this limitation by evaluating a Deep Neural Network alongside Linear Regression, Regression Tree, Random Forest, Support Vector Regression, and Gradient Boosting Machine using the same technical-project dataset, target variable, validation procedure, and performance measures. This design makes it possible to determine whether the added complexity of deep learning provides a meaningful predictive advantage over conventional and ensemble machine-learning approaches.

#### *2.8.4 Review and Conceptual Studies*

In addition to empirical delay-prediction research, several review and conceptual studies have examined the broader role of artificial intelligence in construction and project management. These studies provide important theoretical and managerial insights into the potential applications, implementation challenges, and governance requirements of AI-based project prediction. However, they generally do not provide directly comparable experimental results for continuous schedule-overrun prediction.

Abioye et al. [30] conducted a critical review of artificial-intelligence applications in the construction industry. The study examined several AI subfields and application areas, including machine learning, knowledge-based systems, computer vision, robotics, optimization, activity monitoring, risk management, and resource optimization. It also discussed the opportunities and challenges associated with adopting AI in construction environments [30]. The article was published in the Journal of Building Engineering and was designed as a critical literature review rather than an empirical model-comparison study.

The review highlighted the potential of AI to address complex construction-management problems, including time and cost overruns, productivity limitations, safety concerns, and inefficient resource utilization. It also emphasized that the benefits of AI depend on the availability of appropriate data, organizational readiness, technical infrastructure, and the ability to integrate intelligent systems into existing industry practices [30].

Although Abioye et al. [30] provided a comprehensive overview of AI applications, the study did not train or compare schedule-delay prediction models using a common project dataset. It therefore supports the general motivation for applying AI in project

management but does not provide empirical evidence regarding which regression algorithm performs best for estimating the numerical percentage of schedule overrun.

Fridgeirsson et al. [14] qualitatively examined the expected impact of artificial intelligence on project schedule, cost, and risk-management knowledge areas. Their study presented AI as a decision-support capability that may strengthen forecasting, monitoring, risk identification, and project-control activities. It also emphasized that AI should complement professional project-management judgment rather than replace managerial responsibility [14].

This qualitative perspective is relevant to schedule prediction because it connects predictive analytics with practical project-control decisions. However, the study did not provide a controlled comparison of machine-learning and deep-learning algorithms, nor did it report model-performance measures such as RMSE, MAE, or R2. Its contribution is therefore primarily conceptual and managerial rather than predictive and experimental.

Nenni et al. [15] conducted a systematic literature review on the transformation of project management through artificial intelligence. The review examined a large body of previous research and highlighted the potential of AI to support risk management, forecasting, decision-making, and other project-management processes. It also identified challenges related to data quality, organizational adoption, technical integration, and the need for appropriate human oversight [15]. The review analyzed 215 publications, providing broad evidence of the expanding role of AI in project-management research.

However, systematic reviews synthesize evidence reported by previous studies rather than generating new predictions from a single standardized dataset. Consequently, the results reviewed by Nenni et al. [15] were based on different industries, data structures, target variables, algorithms, and evaluation measures. This heterogeneity limits the ability to determine whether one model family is consistently superior for continuous schedule-overrun prediction.

Santos et al. [17] presented an explainable machine-learning approach for project-management control under uncertainty. Their methodology combined project-network simulation, machine-learning models, and Shapley-based explanations to analyze expected project time and cost deviations. The approach demonstrated how explainable AI can support both forward-looking analysis and retrospective investigation of the variables influencing project performance [17].

The work of Santos et al. [17] is important because it moves beyond prediction accuracy and addresses the need to explain model outputs to project stakeholders. Nevertheless, its objective was to develop an explainable project-control methodology based on simulated project networks rather than to benchmark multiple regression algorithms using a large set of historical technical-project records.

Collectively, these review and conceptual studies establish a strong theoretical foundation for using artificial intelligence in project planning, monitoring, risk management, and decision support [14,15,17,30]. They also emphasize important implementation considerations, including data quality, interpretability, organizational readiness, and the continuing role of professional judgment.

However, they do not resolve the empirical question addressed by the present study: how traditional statistical models, individual tree models, ensemble methods, kernel-based learning, and deep neural networks compare when they are trained and evaluated on the same technical-project dataset to predict a continuous schedule-overrun percentage.

The present research complements these conceptual contributions by providing a controlled empirical benchmark using 4,517 project records. Six models are evaluated under the same target definition, data-preparation process, validation procedure, and performance measures, thereby producing evidence that can be compared directly across model families.

**2.9 Comparative Summary of Related Studies**

The reviewed literature demonstrates growing interest in applying artificial intelligence to project-delay prediction. However, the studies differ considerably in their application domains, data sources, sample sizes, prediction targets, selected algorithms, and evaluation measures. These differences make direct comparison of the reported numerical results difficult. Table 1 therefore summarizes the main characteristics of the reviewed studies and highlights the limitations that motivate the present research.

**Table 1**. Comparative Summary of Related Project-Delay Prediction Studies

| Study | Domain / Sample | Models | Prediction Type | Best Reported Result | Key Gap |
|---|---|---|---|---|---|
| **Gondia et al. [19]** | Construction; 51 projects | DT, NB | Multiclass classification | NB testing accuracy: 51.2% | Small dataset; categorical delay output |
| **Egwim et al. [29]** | Construction; 120 survey responses | RF, GBM, XGBoost, bagging, boosting, stacking | Binary classification | Stacking achieved approximately 85.5% accuracy | Survey-based data; no delay-magnitude estimation |
| **Egwim and Alaka [20]** | Construction; 120 survey | 27 classifiers, including | Binary classification | Perceptron achieved 85% accuracy and F1- | Small sample; binary target |

| | | | | | |
|---|---|---|---|---|---|
| | responses | SVM, RF, DT and boosting models | | score | |
| **Yaseen et al. [21]** | Construction; 40 projects | RF, RF-GA | Multiclass classification | RF-GA achieved 91.67% accuracy | Only 12 testing cases; categorical output |
| **Nassar and Elbisy [23]** | Marine construction; 30 projects | GRNN, SVM, Tree Boost | Continuous regression | GRNN: RMSE 7.321, MAE 7.050, R 0.926 | Small and highly specialized dataset |
| **Alsulamy [22]** | Saudi construction; 62 projects | GAN, LSTM, MLP | Multiclass classification | GAN achieved 91% accuracy | No comparison with traditional or ensemble regression models |
| **Cheng et al. [24]** | Construction; two case studies | OMA-NN-BiGRU | Continuous regression | Schedule Reference Index: 0.932 | Predicted schedule-to-completion rather than overrun percentage |
| **Abioye et al. [30]** | Construction-industry literature review | Review of AI techniques | Not applicable | Identified broad AI opportunities in construction management | No empirical benchmarking of delay-prediction models |
| **Present study** | Technical projects; 4,517 records | LR, RT, RF, SVR, GBM, DNN | Continuous regression | Evaluated using RMSE, MAE, $R^2$ and correlation | Predicts numerical schedule-overrun percentage under a unified framework |

Note: LR = Linear Regression; DT = Decision Tree; NB = Naïve Bayes; RF = Random Forest; GA = Genetic Algorithm; RF–GA = Random Forest–Genetic Algorithm; SVM = Support Vector Machine; SVR = Support Vector Regression; GBM = Gradient Boosting Machine; XGBoost = Extreme Gradient Boosting; GRNN = General Regression Neural Network; GAN = Generative Adversarial Network; LSTM = Long Short-Term Memory; MLP = Multilayer Perceptron; NN = Neural Network; BiGRU = Bidirectional Gated Recurrent Unit; OMA = Optical Microscope Algorithm; DNN = Deep Neural Network; RMSE = Root Mean Squared Error; MAE = Mean Absolute Error.

The comparison reveals several recurring patterns in the existing literature. First, most empirical studies were conducted in construction-related environments, including general construction, marine construction, and geographically restricted national contexts [19,24]. Although these studies provide valuable evidence regarding construction-delay risk, their findings may not transfer directly to broader technical-project environments involving software, digital transformation, infrastructure, enterprise systems, and technology-enabled services.

Second, previous studies generally relied on relatively small datasets. The reviewed empirical samples ranged from 30 to 120 projects or survey responses, and some reported testing results were based on very small subsets of observations [19,23,29]. Small samples may limit model stability, increase sensitivity to individual observations, and reduce confidence that the reported results will generalize to projects outside the original context.

Third, most previous studies formulated project-delay prediction as a binary or multiclass classification problem [19,22,29]. Classification models can provide useful early-warning information by identifying whether a project is likely to be delayed or by assigning it to a predefined delay-severity category. However, they do not provide the precise magnitude of the expected schedule deviation. Projects with substantially different overrun percentages may be placed in the same category, reducing the usefulness of the prediction for detailed contingency planning, resource allocation, recovery scheduling, and contractual assessment.

Continuous regression has received comparatively less attention. Nassar and Elbisy [23] predicted time delay as a numerical outcome, while Cheng et al. [24] estimated schedule to completion using a hybrid deep-learning architecture. Nevertheless, both studies were conducted in specialized construction contexts, and their target variables were not fully equivalent to the continuous schedule-overrun percentage examined in the present research.

The reviewed studies also differ substantially in their model-comparison designs. Some evaluated only two algorithms, while others concentrated on classifiers, ensemble models, or specialized hybrid architectures. Few studies compared conventional statistical regression, an individual tree, bagging-based ensembles, boosting-based ensembles, kernel-based learning, and deep neural networks using the same data, target variable, preprocessing procedures, validation method, and evaluation measures.

The variation in evaluation metrics creates an additional comparison challenge. Classification studies commonly reported accuracy, precision, recall, F1-score, sensitivity, specificity, or ROC-AUC [19,22,29]. Regression studies used measures such as

RMSE, MAE, correlation, or study-specific indices [23,24]. Consequently, a classification accuracy of 91% cannot be compared directly with an RMSE or R2 value because these measures evaluate different prediction tasks.

Review and conceptual studies provide broader evidence that artificial intelligence can support project forecasting, monitoring, risk analysis, and decision-making [14,15,30]. They also emphasize challenges related to data quality, organizational readiness, system integration, interpretability, and human oversight. Santos et al. [17] further demonstrated the importance of explaining predictive outputs rather than evaluating models solely through accuracy. However, these studies do not provide a unified empirical benchmark for continuous schedule-overrun prediction using historical technical-project records.

The present study responds to these limitations by evaluating six models from distinct analytical families on a dataset of 4,517 technical-project records. The models include Linear Regression as a conventional statistical baseline, Regression Tree as an interpretable nonlinear model, Random Forest as a bagging-based ensemble, Support Vector Regression as a kernel-based method, Gradient Boosting Machine as a sequential ensemble, and Deep Neural Network as a deep-learning approach.

All six models are trained and evaluated using the same target definition, data-preparation process, validation framework, and performance measures. Schedule overrun is treated as a continuous numerical percentage and model performance is assessed using RMSE, MAE, R2, and correlation. This unified design enables a methodologically consistent comparison and provides clearer evidence regarding the relative predictive capabilities of conventional, ensemble, kernel-based, and deep-learning approaches in technical-project schedule forecasting.

### *2.10 Critical Synthesis of Previous Research*

The reviewed literature demonstrates that artificial intelligence can support the early identification and prediction of project delay. Traditional machine-learning models, ensemble methods, hybrid optimization approaches, and deep-learning architectures have all shown potential for detecting delay-related patterns in project data [19–24]. However, the existing evidence remains fragmented because previous studies differ substantially in project domain, sample size, prediction formulation, data source, model-selection strategy, and evaluation method.

A dominant characteristic of the literature is its concentration on construction-related applications. Most empirical studies examined general construction, marine construction, or geographically restricted construction-project samples [19,24,29]. Construction projects provide an important context for studying delay because they involve complex dependencies, uncertainty, resource constraints, and external risks. Nevertheless, the extent to which findings from these environments can be generalized to technical projects remains uncertain. Technical projects may involve software development, digital transformation, enterprise systems, infrastructure integration, data platforms, and technology-enabled services, each of which has different delivery characteristics and sources of schedule uncertainty.

Another recurring limitation is the relatively small size of the datasets used in previous empirical studies. The reviewed studies commonly relied on samples ranging from approximately 30 to 120 projects, survey responses, or expert assessments [19,23,29]. Small datasets can make model performance sensitive to the selected training and testing observations. They may also increase the risk that a model captures sample-specific patterns rather than relationships that generalize across a broader population of projects.

The reliability of reported performance is particularly important when testing subsets contain only a small number of observations. For example, the strong accuracy reported by Yaseen et al. [21] was calculated using only 12 testing projects. In such a setting, the classification of one additional project can cause a substantial change in the reported accuracy. Therefore, high performance percentages should be interpreted together with the number of observations on which they were calculated rather than being evaluated in isolation.

The reviewed studies also differed in the origin and structure of their data. Some relied on historical project records, while others used survey responses or expert assessments of delay factors [20,23,29]. Expert-based data can provide valuable knowledge when historical databases are unavailable. However, such data may be affected by respondent perception, recall, professional experience, and contextual interpretation. Models trained on standardized historical project records may provide a different form of evidence from models trained mainly on subjective assessments.

A further limitation concerns the dominant use of classification-based formulations. Several studies predicted whether a project would be delayed or assigned projects to predefined delay-severity categories [19,22,29]. Classification can support early-warning systems by identifying projects requiring attention. However, it reduces a continuous schedule outcome to a limited set of categories and may conceal meaningful differences among projects placed within the same class.

For example, projects with schedule overruns of 31% and 58% may both be classified as having a moderate delay, despite the substantial difference in their managerial and contractual implications. Such categorical predictions provide limited support for determining schedule contingency, estimating recovery effort, prioritizing interventions, or quantifying the expected magnitude of schedule deviation. Treating schedule overrun as a continuous regression target preserves more information and produces a more precise numerical prediction [18,25].

Continuous delay prediction has received comparatively less attention. Nassar and Elbisy [23] estimated project delay numerically, while Cheng et al. [24] predicted schedule to completion using a hybrid deep-learning architecture. These studies

demonstrate the feasibility of regression-based project forecasting. Nevertheless, their application domains were specialized, their datasets were limited, and their target definitions differed from schedule-overrun percentage.

The literature also provides evidence that ensemble and hybrid models can outperform individual baseline models. Egwim et al. [20] reported stronger performance through stacking, while Yaseen et al. [21] improved Random Forest performance by combining it with a Genetic Algorithm. These findings suggest that model diversity, sequential correction, optimization, or aggregation can enhance predictive performance. However, the results were obtained mainly in classification settings and relatively small construction datasets, limiting conclusions about their superiority in continuous technical-project forecasting.

Deep-learning studies have also reported strong results [22,24]. Nevertheless, the complexity of deep architectures does not automatically guarantee superior performance. Their effectiveness depends on dataset size, data structure, model architecture, training configuration, and validation design [18,28]. In structured tabular project data, ensemble tree models may perform competitively because they can capture nonlinear relationships and interactions without requiring the extensive architectural design associated with deep neural networks.

Previous studies rarely compared fundamentally different model families under the same experimental conditions. Some studies focused on conventional classifiers, others on ensembles, and others on specialized hybrid or deep-learning architectures [19,24,29]. As a result, differences in reported performance may reflect not only the capability of the algorithms but also differences in datasets, targets, preprocessing methods, validation procedures, and evaluation measures.

This methodological heterogeneity prevents direct conclusions regarding which model family is most suitable for schedule-overrun prediction. A model achieving high classification accuracy in one dataset cannot be considered superior to a regression model evaluated using RMSE or R2R^2R2 in another dataset. Meaningful benchmarking requires all candidate models to be trained on the same records, predict the same target, undergo the same data preparation, and be assessed through the same validation and performance measures.

Interpretability represents another unresolved issue. Linear Regression and individual decision trees provide relatively understandable coefficients or rules, while ensemble models and deep neural networks are more complex [18,27,28]. Review and conceptual studies emphasize that project stakeholders require not only accurate predictions but also sufficient understanding of the factors associated with those predictions [14,15,17].

Santos et al. [17] demonstrated the value of explainable machine learning in project control by combining predictive methods with Shapley-based explanations. Nevertheless, explainability has not been consistently integrated into project-delay benchmarking studies. Consequently, existing research provides limited evidence about how predictive performance should be balanced against managerial transparency and decision-support requirements.

The literature therefore presents a trade-off among model accuracy, complexity, interpretability, and implementation feasibility. A highly complex model may provide strong predictive performance but be difficult to explain, tune, maintain, or integrate into project-governance processes. Conversely, a simpler model may be easier to understand but fail to capture nonlinear and interaction effects associated with schedule overrun.

Overall, previous research establishes that AI-based methods are relevant to project-delay prediction, but it does not provide a sufficiently broad and controlled comparison for continuous schedule-overrun forecasting in technical projects. The recurring limitations include construction-sector concentration, small or survey-based datasets, categorical prediction targets, inconsistent evaluation methods, limited model-family comparison, and insufficient attention to explainability.

The present study builds on these findings by evaluating six models representing distinct analytical families using a dataset of 4,517 technical-project records. Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network are assessed using the same continuous target, preprocessing procedure, validation framework, and performance measures. This design provides a more consistent basis for comparing predictive accuracy while also considering the interpretability and practical value of the resulting models.

### *2.11 Research Gap*

The reviewed literature confirms the potential of artificial intelligence for project-delay prediction, but several important research gaps remain. Existing studies have primarily focused on construction-related applications, while comparatively limited attention has been given to technical projects involving software development, digital transformation, enterprise systems, infrastructure integration, data platforms, and technology-enabled services [19–24,29].

A second gap concerns dataset scale and diversity. Many previous empirical studies were based on relatively small samples, survey responses, expert assessments, or highly specialized project environments [19,23,29]. These data sources provide useful evidence but may limit model stability and the extent to which reported results can be generalized to broader project portfolios.

A third gap relates to the formulation of the prediction problem. Much of the existing literature treated project delay as a binary or multiclass classification task [19,22,29]. These approaches identify whether delay is expected or assign projects to broad delay-severity categories, but they do not estimate the precise numerical magnitude of schedule overrun. Consequently, their outputs provide limited support for quantitative schedule contingency, recovery planning, resource allocation, and escalation decisions.

Although some studies addressed continuous delay prediction, the available evidence remains limited. Nassar and Elbisy [23] examined numerical time-delay prediction in marine construction, while Cheng et al. [24] estimated schedule to completion using a hybrid deep-learning architecture. However, these studies were based on specialized construction contexts, relatively limited datasets, or target variables that were not fully equivalent to schedule-overrun percentage.

A further methodological gap is the lack of broad and controlled benchmarking across fundamentally different model families. Previous studies typically concentrated on a limited set of classifiers, ensemble models, or specialized hybrid architectures. Few studies have compared a conventional statistical model, an individual tree, a bagging-based ensemble, a boosting-based ensemble, a kernel-based method, and a deep neural network under the same experimental conditions.

This limitation is important because model-performance results reported across different studies cannot be compared directly when the studies use different datasets, target definitions, preprocessing procedures, validation strategies, and evaluation measures. A high classification accuracy reported in one study cannot establish superiority over a regression model assessed using RMSE, MAE, or R2R^2R2 in another study. A unified benchmarking framework is therefore required to isolate model-related performance differences from differences caused by the experimental design.

Another gap concerns the relationship between model complexity and predictive performance. Deep-learning and hybrid architectures are often presented as advanced alternatives to conventional machine-learning methods, but their additional complexity does not necessarily produce a meaningful performance improvement for structured project data [18,28]. Limited evidence is available regarding whether deep neural networks outperform ensemble tree models when both are trained and evaluated on the same technical-project dataset.

Explainability also remains insufficiently integrated into project-delay benchmarking. Existing research has emphasized the importance of understandable predictive outputs for project-control decisions [14,15,17]. However, many empirical studies focus primarily on predictive accuracy without adequately examining which project variables contribute most strongly to the forecasts. This limits the translation of model outputs into practical managerial action.

Accordingly, the principal research gap addressed by the present study is the absence of a unified, large-scale comparison of conventional, tree-based, ensemble, kernel-based, and deep-learning regression models for predicting continuous schedule-overrun percentage in technical projects.

The present study addresses this gap by evaluating six predictive models using 4,517 technical-project records:

- Linear Regression
- Regression Tree
- Random Forest
- Support Vector Regression
- Gradient Boosting Machine
- Deep Neural Network

All models are trained and evaluated using the same target variable, data-preparation process, validation framework, and performance measures. The target is formulated as the continuous percentage of schedule overrun, and performance is assessed using RMSE, MAE, R2R^2R2, and correlation.

The study also examines GBM feature importance to identify the project variables contributing most strongly to schedule-overrun prediction. This provides a managerial interpretation layer in addition to the comparative assessment of predictive accuracy.

By combining a larger technical-project dataset, continuous regression formulation, diverse model families, unified evaluation conditions, and model-interpretation analysis, the present research provides evidence that is currently limited in the project-delay prediction literature.

### *2.12 Positioning and Contribution of the Present Study*

The present study is positioned at the intersection of project-management forecasting, machine learning, deep learning, and decision support. It extends previous project-delay research by examining schedule overrun as a continuous regression problem within the context of technical projects rather than restricting the analysis to construction-delay classification.

The study does not propose a single new algorithm. Instead, its principal contribution lies in establishing a controlled comparative framework through which models from different analytical families can be evaluated under identical experimental conditions. This design enables performance differences to be attributed more clearly to the models themselves rather than to differences in datasets, target definitions, preprocessing procedures, validation strategies, or evaluation measures.

A central contribution of the study is its use of 4,517 technical-project records. The dataset provides a broader empirical basis than the relatively small project samples and survey-based datasets commonly reported in previous delay-prediction studies. It also expands the application domain beyond construction by addressing technical-project environments in which schedule performance may be influenced by complexity, requirements, risks, issues, change activity, team characteristics, and delivery methodology.

The prediction target is defined as the continuous percentage of schedule overrun. This formulation preserves the numerical magnitude of schedule deviation rather than converting the outcome into delayed and non-delayed classes or broad severity

categories. The resulting prediction can therefore provide more precise information for schedule contingency planning, recovery analysis, resource prioritization, escalation, and portfolio-level decision-making.

The study evaluates six models representing distinct predictive approaches: Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. Linear Regression provides an interpretable statistical baseline, while the Regression Tree represents rule-based nonlinear prediction. Random Forest and GBM represent bagging-based and sequential boosting ensembles, respectively. SVR represents margin-based kernel learning, and the DNN represents a multilayer deep-learning architecture.

Evaluating these six models within the same framework provides a broader benchmark than studies focusing on only one algorithm family. It enables direct examination of whether nonlinear approaches outperform a conventional linear baseline, whether ensemble models improve upon an individual tree, whether kernel-based learning is competitive for technical-project data, and whether the additional complexity of deep learning produces a meaningful performance advantage.

Methodological consistency represents another contribution. All models are developed using the same dataset, target variable, predictor set, data-preparation procedures, and validation framework. Their predictive performance is assessed using RMSE, MAE, R2R^2R2, and correlation. This common evaluation framework supports a direct comparison of prediction error, explanatory performance, and agreement between actual and predicted schedule-overrun values.

The study also incorporates safeguards against data leakage. Variables that could disclose or directly derive information from the final schedule outcome are excluded from the predictor set before model training. In particular, variables such as Budget_Overrun_% and Risk_Level_Encoded are removed to reduce the possibility that model performance is artificially inflated by information that would not be independently available when generating a schedule-overrun forecast.

Beyond predictive accuracy, the study addresses the practical interpretation of model outputs. GBM feature-importance analysis is used to identify the project variables that contribute most strongly to schedule-overrun prediction. This analysis does not establish causal relationships, but it provides insight into the variables that carry the greatest predictive value within the fitted model.

The comparison between GBM and DNN is especially relevant to the positioning of the study. Both models can represent complex nonlinear relationships, but they differ in structure, interpretability, tuning requirements, and implementation complexity. Comparing them under identical conditions provides evidence regarding whether deep learning offers a meaningful advantage over a strong tree-based ensemble for structured technical-project data.

The study therefore contributes at several levels. Empirically, it provides evidence based on a comparatively large technical-project dataset. Methodologically, it establishes a unified benchmark across six different model families. Predictively, it evaluates continuous schedule-overrun percentage rather than categorical delay risk. Practically, it supports quantitative schedule decision-making. Interpretively, it supplements performance comparison with GBM feature-importance analysis.

Accordingly, the present study is positioned not merely as another application of machine learning to project delay, but as a comparative decision-support investigation. Its objective is to determine which modeling approach offers the most appropriate balance among predictive accuracy, model complexity, interpretability, and practical applicability for forecasting schedule overrun in technical projects.

The following section presents the research methodology, including the research design, dataset structure, target-variable formulation, data-preparation procedures, model-development process, validation strategy, and performance-evaluation measures.

## 3. Methodology

### *3.1 Research Design*

This study adopts a quantitative, comparative, and predictive research design to evaluate the performance of machine-learning and deep-learning models in estimating schedule overrun in technical projects. The research is formulated as a supervised regression problem in which historical project characteristics are used as predictor variables and the continuous percentage of schedule overrun is used as the target variable[18].

The primary objective is to compare the predictive capabilities of six regression models representing different analytical families: Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. These models were selected to provide a broad comparison among conventional statistical modeling, individual tree-based learning, bagging-based ensembles, boosting-based ensembles, kernel-based learning, and deep-learning approaches.

The study follows a comparative benchmarking approach rather than proposing a new machine-learning algorithm. Each model is trained and evaluated using the same dataset, target-variable definition, predictor set, data-preparation procedures, validation strategy, and performance measures. This standardized experimental design reduces the possibility that observed differences in performance are caused by variations in the underlying data or evaluation conditions.

The unit of analysis is the individual technical-project record. The dataset contains 4,517 project records characterized by schedule-related, organizational, risk-related, issue-related, change-related, and project-specific attributes. These records are used to examine whether historical project characteristics can support the prediction of the numerical magnitude of schedule overrun.

The dependent variable is Schedule_Overrun_%, which represents the percentage by which the actual project duration exceeds the planned duration. Because this variable is continuous, the study applies regression rather than binary or multiclass classification. This formulation enables the models to estimate the expected magnitude of schedule deviation rather than merely determining whether a project is likely to be delayed.

The research is predictive rather than causal. It aims to identify models and variables that provide useful predictive information regarding schedule overrun; it does not attempt to establish that individual predictor variables directly cause project delay. Associations and feature-importance values are therefore interpreted in terms of predictive contribution rather than causal effect [18].

Model performance is evaluated using Root Mean Squared Error, Mean Absolute Error, the squared correlation, and the correlation between actual and predicted values. These measures collectively assess prediction-error magnitude, explanatory capability, and agreement between the observed and estimated schedule-overrun percentages.

The resulting research design supports a controlled assessment of whether more complex nonlinear models provide a meaningful predictive advantage over simpler and more interpretable alternatives. It also enables the study to consider the practical balance among prediction accuracy, model complexity, interpretability, and applicability to technical-project decision support.

3.2 Research Framework and Workflow

The study followed a structured predictive-modeling workflow to ensure that all candidate models were developed and evaluated under consistent experimental conditions. The workflow covered dataset inspection, target-variable formulation, data preprocessing, predictor selection, data partitioning, model development, performance evaluation, comparative analysis, and model interpretation.

The first stage involved inspecting the original dataset to understand its structure, variable types, completeness, and overall suitability for predictive modeling. The records were examined for missing values, duplicated observations, inconsistent categories, invalid values, and variables that could not be used reliably in model development. This stage established the initial quality and scope of the available project data.

The second stage focused on formulating the prediction target. Schedule overrun was represented as a continuous numerical percentage derived from the relationship between actual and planned project durations. This formulation enabled the study to estimate the magnitude of schedule deviation rather than assigning projects to predefined delay categories.

The candidate predictors were subsequently reviewed according to their relevance, availability at the intended prediction point, and potential relationship with the target. Variables that could directly reveal the final project outcome or introduce data leakage were excluded before model training. In particular, Budget_Overrun_% and Risk_Level_Encoded were removed from the final predictor set because of their potential to introduce outcome-related information into the prediction process.

The remaining data were then prepared for modeling. This stage included the treatment of incomplete or invalid observations, the transformation of categorical variables into machine-readable representations, and the application of feature scaling where required by specific algorithms. All preprocessing operations that required learning information from the data were performed using the training data and subsequently applied to the evaluation data to reduce information leakage [18].

After preprocessing, the dataset was partitioned according to the validation strategy described in Section 3.7. The same data partitions and predictor variables were used for all six candidate models to ensure a consistent and fair comparison.

The model-development stage involved training Linear Regression, Regression Tree, Random Forest, and Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network models. These models were selected to represent conventional statistical, individual tree-based, bagging-based ensemble, boosting-based ensemble, kernel-based, and deep-learning approaches.

Each model was configured and trained using the same continuous target variable. Model-specific parameters and hyper parameters were selected according to the procedures described in Sections 3.8 and 3.9. Where preprocessing requirements

differed among algorithms, such as the need for feature scaling in SVR and DNN, these requirements were incorporated within the corresponding model-development process.

The trained models were evaluated using Root Mean Squared Error, Mean Absolute Error, the squared correlation, and correlation. Actual-versus-predicted analyses were also used to examine how closely model predictions followed the observed schedule-overrun values across the evaluation dataset.

The resulting performance measures were compared to identify differences in predictive accuracy, error magnitude, explanatory capability, and agreement between actual and predicted values. The comparison also considered model complexity, interpretability, and practical applicability rather than relying on a single performance measure.

Finally, GBM feature-importance analysis was conducted to identify the project variables that contributed most strongly to schedule-overrun prediction. The feature-importance results were interpreted as indicators of predictive contribution rather than evidence of causal relationships.

The overall research workflow is summarized in Figure 1.

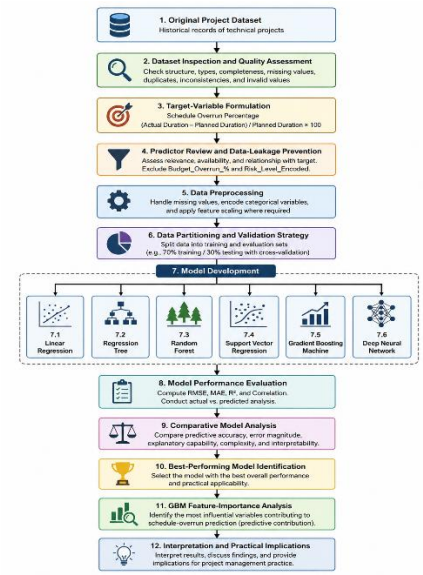


**Figure 1**. Research framework and predictive-modeling workflow used in the study.

### *3.3 Dataset Description*

This section describes the source, scope, structure, and variables of the dataset used to develop and compare the six schedule-overrun prediction models. The dataset contained historical information on technical projects and included project characteristics, planning attributes, execution-related indicators, and duration information required to formulate the continuous prediction target.

#### *3.3.1 Data Source and Scope*

The study used a publicly available dataset containing 4,517 historical software project records. The dataset is available on Kaggle under the title Software Project Dataset and was accessed on 9 May 2026 [31]. Each record represented an individual software project and included information related to project characteristics, planning conditions, execution factors, and final schedule performance.

The dataset covered projects with different types, categories, management methodologies, team sizes, complexity levels, risks, issues, change requests, and project durations. This diversity provided an appropriate basis for examining the ability of different machine-learning algorithms to predict schedule overrun across varying software project conditions.

The scope of the study was limited to project records containing the planning and duration information required to formulate the prediction target. Planned project duration represented the original schedule baseline, whereas actual project duration

represented the time required to complete the project. Actual duration was used exclusively to calculate the dependent variable and was not included among the predictor variables.

The dataset was inspected to assess its suitability for predictive modeling. The inspection considered missing values, duplicated observations, inconsistent categories, invalid values, and the availability of variables at the intended prediction point. Any incomplete or invalid observations were handled according to the preprocessing procedures described in Section 3.5.

The dataset was used to develop and evaluate regression models that predicted schedule overrun as a continuous percentage. Therefore, the prediction task focused on estimating the magnitude of schedule deviation rather than classifying projects into predefined delayed and non-delayed categories.

### *3.3.2 Dataset Variables*

The dataset contained numerical, ordinal, and categorical variables describing the characteristics, planning conditions, and execution environment of each software project. The variables were reviewed according to their relevance to schedule-overrun prediction, their availability at the intended prediction point, and their potential to introduce data leakage.

The final predictor set included project type, project category, project-management methodology, planned project duration, team size, complexity, open risks, open issues, and change requests. These variables represented information that could reasonably be available before the final project outcome was known.

Planned_Duration_Days was retained as a predictor because it represented the original schedule baseline. In contrast, Actual_Duration_Days was used only to calculate the dependent variable and was not provided to the prediction models. Including actual duration as an input would directly reveal information used to construct the prediction target.

The categorical variables Project_Type, Project_Category, and Methodology represented qualitative project characteristics. These variables were converted into numerical representations during preprocessing. The numerical and ordinal predictors represented measurable project conditions, including planned duration, team size, complexity, risks, issues, and change requests.

Schedule_Overrun_% was used as the continuous dependent variable for all six candidate models. Budget_Overrun_% and Risk_Level_Encoded were excluded from the final predictor set because they could contain information closely associated with the final project outcome and therefore increase the risk of data leakage.

The variables considered in the modeling process, together with their data types and roles, are summarized in Table 2.

**Table 2**. Description and Modeling Roles of the Study Variables

| Variable | Type | Description | Modeling Role |
|---|---|---|---|
| **Project_Type** | Categorical | Represents the general type of the technical project. | Predictor |
| **Project_Category** | Categorical | Represents the functional or organizational category of the project. | Predictor |
| **Methodology** | Categorical | Identifies the project-management or development methodology, such as Agile, Waterfall, or Hybrid. | Predictor |
| **Planned_Duration** | Numerical | Represents the originally planned duration of the project. | Predictor |
| **Team_Size** | Numerical | Represents the number of team members assigned to the project. | Predictor |
| **Complexity** | Numerical/Ordinal | Indicates the assessed level of project complexity. | Predictor |
| **Risks** | Numerical | Represents the number or level of risks recorded for the project. | Predictor |
| **Issues** | Numerical | Represents the number of project issues recorded during execution. | Predictor |
| **Change_Requests** | Numerical | Represents the number of formal change requests associated with the project. | Predictor |
| **Actual_Duration** | Numerical | Represents the final duration required to complete the project. | Target calculation only |
| **Schedule_Overrun_%** | Numerical | Represents the percentage deviation of actual duration from planned duration. | Target |

| | | | |
|---|---|---|---|
| **Budget_Overrun_%** | Numerical | Represents the percentage difference between actual and planned project cost. | Excluded |
| **Risk_Level_Encoded** | Numerical/Ordinal | Encoded representation of the project risk level. | Excluded |

**Note:** Actual_Duration_Days was used exclusively to derive Schedule_Overrun_% and was not included in the predictor matrix. Budget_Overrun_% and Risk_Level_Encoded were excluded before model training to reduce the risk of outcome-related data leakage.

**Source:** Prepared by the authors based on the study dataset [31].

### 3.4 Target-Variable Formulation

Schedule overrun was formulated as a continuous numerical percentage representing the deviation between the actual and planned duration of each project. The target variable, Schedule_Overrun_%, was calculated using Equation (1):

$$\text{Schedule\_Overrun\_\%} = \left(\frac{\text{Actual\_Duration\_Days} - \text{Planned\_Duration\_Days}}{\text{Planned\_Duration\_Days}}\right) \times 100$$

This formulation expresses schedule deviation relative to the original planned duration. A percentage-based measure was selected because the same number of delay days may have different implications depending on the planned length of the project. For example, a delay of 30 days represents a substantially larger schedule deviation for a three-month project than for a two-year project.

Positive values of Schedule_Overrun_% indicated that the actual project duration exceeded the planned duration. A value of zero indicated that the project was completed according to its planned duration, whereas negative values, where present, indicated completion earlier than planned.

Actual_Duration_Days was used exclusively in the calculation of the target variable and was excluded from the predictor matrix. Its inclusion as a model input would provide direct information about the final schedule outcome and introduce data leakage.

The continuous target formulation allowed the models to estimate the magnitude of schedule deviation rather than merely determine whether a project was delayed. Consequently, the prediction task was treated as a regression problem. Retaining the continuous form also avoided the loss of information associated with converting numerical outcomes into predefined categories [25].

Because Planned_Duration_Days appeared in the denominator of Equation (1), its values were required to be valid and greater than zero. Records containing missing, invalid, or zero planned-duration values were reviewed and treated during the preprocessing stage described in Section 3.5.

### *3.5 Data Preprocessing*

Before model development, the dataset was preprocessed to improve data quality and ensure compatibility with the input requirements of the six candidate algorithms. The preprocessing procedure consisted of data cleaning, categorical-variable encoding, feature scaling, and outlier treatment. These operations were designed to prepare a consistent modeling dataset while reducing the risk of invalid observations, inappropriate variable representations, and information leakage.

Preprocessing steps that required parameters to be learned from the data were fitted using the training data and subsequently applied to the corresponding evaluation data. This procedure prevented information from the evaluation set from influencing model development [18].

#### *3.5.1 Data Cleaning*

The dataset was initially inspected for missing values, duplicated observations, inconsistent categorical labels, invalid numerical entries, and logically implausible project-duration values. The objective of this stage was to ensure that each observation contained reliable information suitable for target-variable calculation and predictive modeling.

Particular attention was given to Planned_Duration_Days and Actual_Duration_Days because these variables were required to calculate Schedule_Overrun_%. Planned-duration values were required to be greater than zero because Planned_Duration_Days appeared in the denominator of the target-variable equation. Records containing missing, zero, negative, or otherwise invalid duration values were therefore identified and treated before the target variable was calculated.

The categorical variables were also inspected for inconsistent spelling, duplicated labels, unnecessary spaces, and variations in capitalization. Categories referring to the same project characteristic were standardized to prevent them from being interpreted as separate categories during the encoding process.

Duplicated project records were examined to prevent the same observation from contributing more than once to model training or evaluation. Numerical variables were additionally reviewed to identify invalid values that fell outside their logically acceptable ranges.

Following the cleaning process, the resulting dataset contained valid and consistently represented observations suitable for the subsequent encoding, scaling, outlier-treatment, and model-development stages.

### *3.5.2 Categorical-Variable Encoding*

The dataset contained three categorical variables used in model development: Project_Type, Project_Category, and Methodology. Because the candidate algorithms required numerical inputs, the corresponding encoded variables—Project_Type_Encoded, Project_Category_Encoded, and Methodology_Encoded—were used in the predictor matrix.

Each category was represented by a fixed numerical code that remained consistent across all project records. The codes served as machine-readable identifiers for the original categories and were not intended to represent differences in magnitude or importance among them. The original textual variables were retained for interpretation and reporting purposes but were excluded from model training to avoid duplicate representations of the same project characteristic.

The same categorical mappings were applied to both the training and evaluation data. No encoding category was created using information from Schedule_Overrun_%, ensuring that the transformation remained independent of the prediction target. Using fixed mappings also ensured that all six candidate models received identical categorical representations and could therefore be compared under consistent input conditions.

Although the dataset also contained Risk_Level_Encoded, this variable was not included in the final predictor matrix because of its potential to reflect outcome-related project information. Its exclusion is discussed further in Section 3.6.

### *3.5.3 Feature Scaling*

The numerical predictor variables were measured on different scales. For example, Planned_Duration_Days generally contained larger numerical values than variables such as Complexity, Open_Risks, Open_Issues, and Change_Requests. Differences in predictor magnitude can influence algorithms that depend on distance calculations or gradient-based optimization, particularly Support Vector Regression and Deep Neural Networks [18,26,28].

However, no explicit feature-scaling or normalization operator was included in the implemented RapidMiner workflows. All six candidate models were therefore trained using the cleaned numerical variables in their original measurement scales. The same predictor representation was maintained across the models, and no standardization, Z-transformation, or Min–Max normalization was performed before model training.

Because no scaling transformation was fitted, no scaling parameters were derived from either the training or evaluation partitions. The potential influence of unscaled numerical variables was considered when interpreting the results, particularly the comparatively weak performance of the Support Vector Regression model. Nevertheless, the absence of scaling should be regarded only as a possible contributing factor and not as the sole explanation for differences in model performance.

The original units and numerical ranges of the project variables were retained throughout the modeling process.

### 3.5.4 Treatment of Outliers

Potential outliers were examined during the data-quality assessment stage by reviewing the numerical distributions and checking unusually high or low values for logical plausibility. Particular attention was given to project durations, team size, complexity, risks, issues, change requests, and Schedule_Overrun_% values.

Observations identified as invalid or logically impossible were addressed during data cleaning. However, extreme values that represented plausible project conditions were retained because they reflected meaningful cases of substantial schedule deviation and contributed to the practical variability of the dataset.

No automatic removal rule was applied solely on the basis of numerical magnitude. This prevented valid high-overrun projects from being incorrectly excluded and allowed the predictive models to learn from both typical and extreme project outcomes.

### *3.6 Predictor Selection and Data-Leakage Prevention*

The predictor variables were selected according to their relevance to schedule-overrun prediction, availability at the intended prediction point, data completeness, and suitability for consistent use across the six candidate models. The objective was to construct a predictor set that represented project characteristics and execution conditions without directly revealing the final schedule outcome.

The final predictor set consisted of Project_Type_Encoded, Project_Category_Encoded, Methodology_Encoded, Planned_Duration_Days, Team_Size, Complexity, Open_Risks, Open_Issues, and Change_Requests. These variables represented project characteristics, planning conditions, and operational factors that could reasonably contribute to schedule deviation.

Planned_Duration_Days was retained because it represented the original schedule baseline and was available before the final project outcome was known. Although it was also used in calculating Schedule_Overrun_%, it did not contain the actual completion duration and therefore did not directly reveal the target value.

Actual_Duration_Days was excluded from the predictor matrix because it was used to calculate Schedule_Overrun_%. Including this variable would provide direct access to the final project duration and allow the models to reconstruct the target, resulting in data leakage and unrealistically optimistic predictive performance.

Budget_Overrun_% was also excluded because it represented a final performance outcome that could be closely associated with schedule delay. Its inclusion could introduce information that would not necessarily be available at the intended prediction stage. Similarly, Risk_Level_Encoded was removed because the encoded risk classification could reflect aggregated or outcome-related information derived from project conditions already represented by other variables.

The original categorical variables Project_Type, Project_Category, and Methodology were not included together with their encoded versions. Only the encoded variables were used during model training to avoid providing duplicate representations of the same project characteristics.

The same finalized predictor set was used for Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. This consistent feature set ensured that differences in model performance were attributable primarily to the learning algorithms rather than to variations in the input variables.

Predictor selection and leakage prevention were completed before model training and performance evaluation. This separation helped preserve the validity of the comparative analysis and reduced the risk of obtaining misleading evaluation results [18].

### *3.7 Data Partitioning and Validation Strategy*

The dataset was evaluated using 10-fold cross-validation, implemented through the Cross Validation operator in RapidMiner. This validation strategy was selected to provide a more reliable estimate of model performance than relying on a single training–testing split[18].

The dataset was divided into ten non-overlapping subsets, referred to as folds. In each iteration, nine folds were used to train the predictive model, while the remaining fold was used for evaluation. This procedure was repeated ten times until each fold had served once as the evaluation set.

Within the training subprocess, the corresponding algorithm was fitted using only the nine training folds. The resulting model was then transferred to the testing subprocess and applied to the held-out fold. Model performance was calculated by comparing the predicted Schedule_Overrun_% values with the corresponding actual values.

The same 10-fold cross-validation structure, predictor variables, target variable, and performance measures were applied consistently to Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. This ensured that all candidate models were evaluated under equivalent experimental conditions.

The project records were shuffled during partitioning to reduce the possibility that the folds reflected the original ordering of the dataset. Because each observation was used for evaluation once while remaining excluded from the training data in the corresponding iteration, the procedure reduced the risk of obtaining overly optimistic performance estimates.

The final RMSE, MAE, $R^2$, and correlation values were obtained by aggregating model performance across the ten cross-validation iterations. The actual-versus-predicted analyses were based on predictions generated for observations while they were included in the corresponding held-out evaluation folds.

### *3.8 Predictive Model Development*

Six regression algorithms were developed and evaluated to compare their ability to predict Schedule_Overrun_%. The selected models represented different predictive-learning approaches, including a conventional statistical model, an individual decision-tree model, a bagging-based ensemble, a kernel-based regression model, a boosting-based ensemble, and a deep-learning model.

The evaluated algorithms were Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. Each model was trained using the same continuous target variable, predictor set, and 10-fold cross-validation structure to maintain consistent experimental conditions.

The models were implemented independently within the RapidMiner Cross Validation operator. During each validation iteration, the model was trained using nine folds and applied to the remaining held-out fold. Predictions from the evaluation folds were subsequently used to calculate RMSE, MAE, $R^2$, and correlation.

Model-specific configurations were selected according to the characteristics of each learning algorithm. The theoretical basis and role of each model are described in Sections 3.8.1–3.8.6, while the corresponding parameter settings are reported separately in Section 3.9.

#### *3.8.1 Linear Regression*

Linear Regression was used as the conventional statistical baseline for schedule-overrun prediction. The model estimates the relationship between the predictor variables and Schedule_Overrun_% through a linear combination of the input attributes [18]. The general form of the model is expressed in Equation (2):

$$y^=\beta0+\beta1x1+\beta2x2+\cdots+\beta pxp$$

where y^\hat{y}y^ represents the predicted schedule-overrun percentage, β0\beta_0β0 is the intercept, β1,…,βp\beta_1,\ldots,\beta_pβ1,…,βp are the estimated regression coefficients, and x1,…,xpx_1,\ldots,x_px1,…,xp represent the predictor variables [18].

Linear Regression was included to provide an interpretable benchmark against which the performance of the nonlinear and ensemble-based models could be evaluated. Its inclusion also enabled the study to examine whether the relationship between the available projects attributes and schedule overrun could be represented adequately through a linear structure.

The model was trained and evaluated using the same 10-fold cross-validation procedure applied to all candidate algorithms. Its performance therefore provided a baseline for determining whether more complex learning methods offered meaningful improvements in predictive accuracy.

### *3.8.2 Regression Tree*

A Regression Tree was developed to model nonlinear relationships between the predictor variables and Schedule_Overrun_%. Unlike Linear Regression, the model does not assume that the relationship between the predictors and the target follows a linear functional form. Instead, it recursively partitions the predictor space into smaller and more homogeneous regions based on decision rules applied to the input variables [18].

At each internal node, the algorithm selects a predictor and a corresponding split point that reduces the prediction error within the resulting subsets. This recursive partitioning process continues until the stopping conditions specified in the model configuration are reached. Each final region, referred to as a terminal or leaf node, produces a predicted schedule-overrun percentage based on the average target value of the training observations assigned to that region.

The general prediction structure of a regression tree can be expressed as:

$$\hat{y} = \sum_{m=1}^{M} c_m I(x \in R_m)$$

where *y^\hat{y}y^* represents the predicted schedule-overrun percentage, *MMM* is the number of terminal regions, *RmR_mRm* represents the *mmm*-th terminal region, *cmc_mcm* is the predicted value assigned to that region, and *I(x∈Rm)I(x\in R_m)I(x∈Rm)* is an indicator function that equals one when an observation belongs to region *RmR_mRm* and zero otherwise [18].

The Regression Tree was included because it can capture threshold effects and interactions among project variables without requiring these relationships to be specified in advance. For example, the influence of a project-risk factor may differ depending on project duration, team size, or methodology. Such conditional relationships can be represented through successive tree splits. The model also provides an interpretable structure because its predictions can be traced through a sequence of decision rules. However, an individual regression tree may be sensitive to variations in the training data and may produce less stable predictions than ensemble approaches. Its inclusion therefore provided a direct comparison between a single tree and the Random Forest and Gradient Boosting Machine models, which combine multiple decision trees.

The Regression Tree was trained and evaluated using the same predictor set and 10-fold cross-validation procedure applied to the other candidate models. The specific tree settings and stopping parameters are reported in Section 3.9.

### *3.8.3 Random Forest*

A Random Forest regression model was developed to improve predictive stability and reduce the variance associated with an individual regression tree. The model combines predictions from multiple decision trees constructed using different bootstrap samples of the training data and randomly selected subsets of predictor variables [18].

For each tree, a bootstrap sample was drawn from the available training observations. At each split, only a randomly selected subset of predictors was considered when determining the optimal partition. This process introduced diversity among the individual trees and reduced the likelihood that all trees would produce highly similar prediction errors.

The final Random Forest prediction was obtained by averaging the outputs of the individual regression trees, as expressed in Equation (4):

$$\hat{y}_{RF(x)} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_{b(x)}$$

where y^RF(x)\hat{y}_{RF}(x)**y^RF**(**x**) represents the final predicted schedule-overrun percentage for observation xx**x**, BB**B** is the total number of trees, and f^b(x)\hat{f}_{b}(x)**f^b**(**x**) is the prediction generated by the bb**b**-th regression tree.

Random Forest was included because it can model nonlinear relationships and complex interactions among project variables without requiring a predefined functional form. Averaging predictions across multiple trees generally produces more stable and accurate results than relying on a single regression tree [18].

The model also provides variable-importance estimates that can indicate the relative contribution of individual predictors to the prediction process. However, these importance values were treated as measures of predictive contribution rather than evidence of causal relationships.

The Random Forest model was trained and evaluated using the same predictor set and 10-fold cross-validation procedure applied to the other candidate models. The number of trees and other model-specific settings are reported in Section 3.9.

### *3.8.4 Support Vector Regression*

Support Vector Regression was developed to model potentially nonlinear relationships between the predictor variables and Schedule_Overrun_%. SVR extends the principles of support vector machines to continuous prediction problems by identifying a regression function that maintains prediction errors within a specified tolerance while controlling model complexity [26].

The model applies an ε\varepsilonε-insensitive loss function, under which prediction errors smaller than the predefined ε\varepsilonε threshold are not penalized. Observations located outside this tolerance region contribute to the optimization process and may become support vectors that determine the final regression function [26].

The general SVR prediction function is expressed as:

$$\hat{y}(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)K(x_i, x) + b$$

where y^(x)\hat{y}(x)y^(x) represents the predicted schedule-overrun percentage, xix_ixi represents a training observation, αi\alpha_iαi and αi∗\alpha_i^{*}αi∗ are the learned coefficients, K(xi,x)K(x_i,x)K(xi,x) is the kernel function, and bbb is the intercept term.

The kernel function enables SVR to represent nonlinear relationships by measuring similarity between observations in a transformed feature space without explicitly calculating the transformation. This capability made SVR suitable for examining whether complex schedule-overrun patterns could be captured through a kernel-based approach [26].

SVR was included because it can provide effective regression estimates in datasets containing nonlinear relationships and multiple interacting predictors. However, its performance depends strongly on the selected kernel and related hyperparameters, including the regularization parameter, kernel coefficient, and ε\varepsilonε-insensitive margin.

The model was trained using the original numerical predictor scales and evaluated through the same 10-fold cross-validation procedure applied to the other candidate models. The kernel type and model-specific parameter settings are reported in Section 3.9.

### *3.8.5 Gradient Boosting Machine*

A Gradient Boosting Machine was developed to model complex nonlinear relationships between the predictor variables and Schedule_Overrun_%. GBM is an ensemble-learning method that constructs a sequence of regression trees, with each new tree trained to reduce the prediction errors produced by the preceding ensemble [27].

The model begins with an initial prediction and progressively adds weak regression trees. At each boosting iteration, the new tree is fitted to the residual errors, or more generally to the negative gradient of the selected loss function. The contribution of each tree is controlled by a learning-rate parameter, which determines the magnitude of the update added to the existing model.

The general additive structure of the GBM prediction can be expressed as:

$$\hat{F}_{M(x)} = \hat{F}_{0(x)} + \sum_{m=1}^{M} \eta h_{m(x)}$$

where *F^M(x)\hat{F}_{M}(x)F^M(x)* represents the final predicted schedule-overrun percentage after *MMM* boosting iterations, *F^0(x)\hat{F}_{0}(x)F^0(x)* is the initial prediction, *η\etaη* is the learning rate, and *hm(x)h_m(x)hm(x)* represents the regression tree added at the *mmm*-th iteration [27].

Unlike Random Forest, which develops multiple trees independently and averages their predictions, GBM constructs trees sequentially. Each tree focuses on correcting prediction errors that remain after the preceding trees have been added. This sequential error-correction process enables the model to capture nonlinear effects and complex interactions among project variables.

GBM was included because of its strong predictive capability on structured tabular datasets and its ability to combine multiple weak learners into a more accurate ensemble model. It can represent threshold effects and interactions that may not be captured adequately by conventional linear models.

The model also provides feature-importance estimates that indicate the relative contribution of individual predictors to the prediction process. These importance values were used later in the explainability analysis to identify the variables that contributed most strongly to the GBM predictions. They were interpreted as indicators of predictive contribution rather than causal influence.

The GBM model was trained and evaluated using the same predictor set and 10-fold cross-validation procedure applied to the other candidate models. Its number of trees, maximum tree depth, learning rate, and other model-specific settings are reported in Section 3.9.

### *3.8.6 Deep Neural Network*

A Deep Neural Network was developed to capture complex nonlinear relationships and interactions between the predictor variables and Schedule_Overrun_%. DNNs consist of interconnected layers of processing units, where each layer transforms the output received from the preceding layer before passing it to the next layer [28].

The input layer received the finalized predictor variables, while one or more hidden layers progressively learned higher-level representations of the project data. Each neuron calculated a weighted combination of its inputs, added a bias term, and applied an activation function. The final output layer produced a continuous numerical estimate of the schedule-overrun percentage.
The transformation performed by a hidden layer can be expressed as:

$$h^(l) = \varphi\left(W^{(l)h^{l-1}} + b^l\right)$$

where *h(l)h^{(l)}h(l)* represents the output of layer *lll*, *h(l−1)h^{(l-1)}h(l−1)* represents the output of the preceding layer, *W(l)W^{(l)}W(l)* is the weight matrix, *b(l)b^{(l)}b(l)* is the bias vector, and *ϕ\phiϕ* represents the activation function.
The final prediction can be represented as:

$$\hat{y} = W^(L)h^(L-1) + b^L$$

where *y^\hat{y}y^* represents the predicted Schedule_Overrun_%, *h(L−1)h^{(L-1)}h(L−1)* is the output of the final hidden layer, and *W(L)W^{(L)}W(L)* and *b(L)b^{(L)}b(L)* represent the weights and bias of the output layer.
During training, the network adjusted its weights and biases to reduce the difference between the predicted and actual schedule-overrun values. This adjustment was performed iteratively through forward propagation, error calculation, and backpropagation using a gradient-based optimization procedure.
The DNN was included because its multilayer structure can approximate complex nonlinear functions and identify interactions among predictors that may not be captured adequately by linear or individual tree-based models. However, its predictive performance can depend on architectural and training choices, including the number of hidden layers, number of neurons, activation functions, learning rate, and training cycles.
The model was trained using the original predictor scales and evaluated through the same 10-fold cross-validation procedure applied to the other candidate models. The network architecture and model-specific training parameters are reported in Section 3.9.

### *3.9 Hyperparameter Configuration*

The model-specific hyperparameters were defined through the corresponding learner operators in RapidMiner. The selected settings were maintained consistently across all ten cross-validation folds so that each model was trained under the same configuration during every validation iteration.
Hyperparameter configuration was performed separately for each algorithm because the six models differed in their structures and learning mechanisms. The relevant settings included the feature-selection and regularization options for Linear Regression; splitting, depth, pruning, and stopping criteria for the Regression Tree; the number and configuration of trees for Random Forest; the kernel and regularization parameters for Support Vector Regression; the number of trees, maximum depth, and learning rate for Gradient Boosting Machine; and the architecture and training parameters of the Deep Neural Network.
The model configurations were selected to provide stable predictive performance while maintaining a consistent experimental framework. Hyperparameters were specified before the final comparative evaluation, and the same settings were applied throughout the 10-fold cross-validation procedure. Table 3 summarizes the final configuration used for each candidate model.

**Table 3**. Main Hyperparameter Configuration of the Predictive Models

| Model | Main Configuration |
|---|---|
| **Linear Regression** | RapidMiner default configuration; no manual tuning |
| **Regression Tree** | RapidMiner default configuration; no manual tuning |
| **Random Forest** | RapidMiner default configuration; no manual tuning |
| **SVR** | RapidMiner default configuration; no manual tuning |
| **GBM** | RapidMiner default configuration; no manual tuning |
| **DNN** | RapidMiner default configuration; no manual tuning |

All six models were implemented using the default hyperparameter settings provided by their corresponding RapidMiner learner operators. No manual hyperparameter optimization or automated parameter-tuning procedure was performed. The same default configuration for each model was maintained throughout all ten cross-validation folds.

### *3.10 Performance Evaluation Metrics*

The predictive performance of the six regression models was evaluated using Root Mean Squared Error, Mean Absolute Error, the squared correlation, and Pearson correlation. These metrics were selected because they provide complementary information regarding prediction-error magnitude, sensitivity to large errors, explanatory performance, and agreement between actual and predicted schedule-overrun values [18].
The metrics were calculated using the predictions generated from the held-out folds of the 10-fold cross-validation procedure. Consequently, each project observation contributed to performance evaluation only when it was excluded from the corresponding model-training fold.

***1) Root Mean Squared Error***

Root Mean Squared Error measures the square root of the average squared difference between the actual and predicted schedule-overrun percentages. It is expressed as:

$$RMSE = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Where *n* represents the number of evaluated observations, *yi* is the actual Schedule_Overrun_% value for observation i, and y^ is the corresponding predicted value.

Because prediction errors are squared before averaging, RMSE assigns greater weight to relatively large errors. A lower RMSE value therefore indicates better predictive accuracy and fewer substantial deviations between the actual and predicted schedule-overrun percentages.

***2) Mean Absolute Error***

Mean Absolute Error represents the average absolute difference between the actual and predicted values and is calculated as:

$$\mathrm{MAE} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}} |\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i}|$$

Unlike RMSE, MAE does not square the individual prediction errors and is therefore less sensitive to a limited number of large deviations. Because the target variable was expressed as a percentage, MAE could be interpreted directly as the average absolute prediction error in schedule-overrun percentage points. Lower MAE values indicate better model performance.

***3) Squared correlation***

The squared correlation measures the proportion of variation in the actual schedule-overrun values explained by the model. It is expressed as:

$$R\text{^}2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Where *y¯\bar{y}y¯* represents the mean of the actual Schedule_Overrun_% values.

An *R2R^2R2* value closer to 1 indicates that the model explains a larger proportion of the variation in schedule overrun. A value close to zero indicates limited explanatory performance, whereas negative values may occur when a model performs worse than predicting the mean target value.

***4) Pearson Correlation***

Pearson correlation was used to measure the strength of the linear agreement between the actual and predicted schedule-overrun percentages. It is calculated as:

$$r = (\sum\nolimits_{-}(i = 1)\text{^}n\ (y\_i - \bar{y})(\hat{y}\_i - \bar{\hat{y}}))/\sqrt{\left(\sum_{i=1}^{n} (y_i - \bar{y})^2\right)\left(\sum_{i=1}^{n} \left(\hat{y}_i - \bar{\hat{y}}\right)^2\right)}$$

where *y^¯\bar{\hat{y}}y^¯* represents the mean of the predicted values.

Correlation values closer to 1 indicate stronger positive agreement between actual and predicted values. However, correlation alone does not measure the magnitude of prediction errors. A model may produce strongly correlated predictions while still containing systematic overprediction or underprediction. Correlation was therefore interpreted together with RMSE, MAE, and *R2R^2R2*.

No single metric was used independently to identify the best-performing model. RMSE and MAE were minimized, whereas *R2R^2R2* and correlation were maximized. The combined evaluation allowed the models to be assessed in terms of overall error, sensitivity to large prediction deviations, explained variation, and agreement with the observed schedule-overrun values.

***3.11 Model Comparison Procedure***

The six predictive models were compared using the performance results obtained from the same 10-fold cross-validation procedure. The comparison was designed to ensure that each model was evaluated using identical predictor variables, the same continuous target variable, equivalent validation folds, and the same performance measures.

Model performance was compared using RMSE, MAE, R2R^2R2, and correlation. RMSE and MAE were interpreted as error measures, with lower values indicating more accurate predictions. In contrast, R2R^2R2 and correlation were interpreted as agreement measures, with higher values indicating stronger correspondence between the actual and predicted schedule-overrun percentages.

The comparative analysis did not rely on a single performance metric. Greater emphasis was placed on RMSE and MAE because these measures directly represented the magnitude of prediction errors. R2R^2R2 and correlation were used as complementary indicators of the models' ability to reproduce the variation and overall pattern of the observed target values.

The model producing the lowest overall prediction errors while maintaining strong agreement between actual and predicted values was considered the best-performing model. Where two models produced similar results, the comparison also considered model complexity, interpretability, computational requirements, and practical applicability to project-management decision support.

The actual-versus-predicted plots were additionally examined to assess the distribution of predictions across the observed range of Schedule_Overrun_%. These plots supported the numerical comparison by revealing whether a model systematically overpredicted or underpredicted particular schedule-overrun levels.

The final model ranking was based on the combined evidence provided by the numerical performance measures and graphical analyses. This procedure enabled the study to identify both the most accurate model and the practical trade-offs associated with the six predictive approaches.

### *3.12 Explainability Analysis*

An explainability analysis was conducted for the Gradient Boosting Machine because it achieved the strongest overall predictive performance based on the combined evaluation criteria. The objective was to identify the project variables that contributed most strongly to the model's schedule-overrun predictions.

Feature-importance scores were extracted from the trained GBM model. These scores reflected the relative contribution of each predictor to the decision-tree splits and prediction improvements generated across the boosting process. Predictors with higher importance values contributed more frequently or more substantially to reducing prediction error within the ensemble [27].

The predictors were ranked according to their feature-importance scores. This ranking provided an indication of which project characteristics the GBM relied on most heavily when estimating Schedule_Overrun_%. The resulting importance values were presented graphically to facilitate comparison among the predictors.

Feature importance was interpreted as a measure of predictive contribution rather than causal influence. A high importance score indicated that a variable was useful to the model for generating predictions; however, it did not demonstrate that the variable directly caused schedule overrun. Similarly, a lower importance score did not necessarily imply that the corresponding project factor was managerially unimportant [17, 18].

The analysis was performed at the global model level, meaning that the resulting ranking described the overall contribution of predictors across the complete dataset rather than explaining the prediction of an individual project. The findings were subsequently interpreted in relation to project-management practice and the observed predictive behavior of the GBM model [17].

The feature-importance results are reported and discussed in the results section, where the most influential predictors are compared and their practical implications for schedule-risk monitoring are examined.

### *3.13 Software and Computational Environment*

The data-preparation, model-development, cross-validation, prediction, and performance-evaluation procedures were implemented using RapidMiner AI Studio 2026.1.1. Separate modeling workflows were created for Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network.

The workflows included operators for dataset retrieval, predictor selection, target-role assignment, 10-fold cross-validation, model training, prediction generation, and regression-performance evaluation. The Performance (Regression) operator was used to calculate RMSE, MAE, correlation, and squared correlation from the predictions generated for the held-out validation folds.

The same workflow structure was maintained across the six models, with the corresponding learner operator changed according to the algorithm being evaluated. This consistent implementation reduced procedural differences among the models and supported a fair comparative assessment.

Spreadsheet software was used where required to inspect the dataset structure, verify variable names, organize model-performance results, and prepare comparative tables and figures. The original project records were obtained from the publicly available Kaggle dataset described in Section 3.3.1.

The experiments were conducted locally using RapidMiner AI Studio 2026.1.1 on a computer running Microsoft Windows 11. The computational environment was sufficient for processing the 4,517 project records and executing the 10-fold cross-validation procedures for all six predictive models.

No external cloud-computing environment or distributed-processing infrastructure was used. Conducting all experiments within the same local software environment helped maintain consistency across the model-development and evaluation procedures.

### *3.14 Reliability, Reproducibility, and Ethical Considerations*

Several procedural measures were adopted to support the reliability and reproducibility of the study. All six predictive models were developed using the same dataset, continuous target variable, predictor set, evaluation metrics, and 10-fold cross-validation structure. Maintaining consistent experimental conditions reduced the possibility that differences in model performance resulted from variations in the evaluation procedure rather than from the learning algorithms themselves.

Each project record was used for evaluation once through the held-out folds of the cross-validation process. Model performance was therefore calculated using predictions generated for observations that were not included in the corresponding training fold.

This procedure reduced the risk of overly optimistic performance estimates and provided a more reliable assessment than a single training–testing split [18].

The RapidMiner workflows were implemented separately for each candidate algorithm while preserving the same overall process structure. All models were executed using the default hyperparameter settings provided by their corresponding learner operators, without manual or automated hyperparameter optimization. The software version, operating system, validation procedure, target formulation, preprocessing decisions, and evaluation measures were documented to facilitate replication of the study.

Reproducibility was also supported by the use of a publicly available Kaggle dataset containing 4,517 software project records. Researchers can access the original dataset through the source reported in Section 3.3.1. However, exact replication requires the use of the same selected variables, target-variable calculation, data-cleaning decisions, encoded representations, RapidMiner AI Studio version, and learner configurations.

The study used secondary data obtained from a public data repository and did not involve direct interaction with human participants. No surveys, interviews, experiments involving individuals, or collection of personally identifiable information were conducted. The analysis focused exclusively on project-level attributes and schedule-performance outcomes.

The predictive results were interpreted as decision-support information rather than definitive managerial judgments. Model predictions may help identify projects with potential schedule-overrun risk, but they should not replace professional assessment, contextual knowledge, or established project-governance processes [15, 17].

Feature-importance values were similarly interpreted as indicators of predictive contribution rather than causal evidence. A variable receiving a high importance score was considered influential in the model's prediction process, but this did not demonstrate that the variable independently caused schedule overrun [17, 18].

Potential ethical concerns related to automated decision-making were considered when interpreting the findings. The models should not be used to assign responsibility to individual employees, evaluate personal performance, or make employment-related decisions without additional evidence and human oversight. Their appropriate role is to support early risk identification, planning, monitoring, and informed intervention at the project level [15, 17].

Finally, the findings remain dependent on the scope, quality, and representativeness of the publicly available dataset. Predictions generated in other organizational, industrial, or geographical contexts may require further validation before practical deployment [15].

## 4. Results

This section presents the experimental results obtained from the six regression models evaluated in this study: Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. All models were developed and evaluated under identical experimental conditions using the same dataset, predictor variables, continuous target variable, preprocessing procedure, and 10-fold cross-validation structure.

Model performance was assessed using root mean squared error (RMSE), mean absolute error (MAE), correlation, and squared correlation. RMSE and MAE quantify the magnitude of prediction errors, with lower values indicating better predictive performance. RMSE assigns greater weight to larger errors because prediction errors are squared before aggregation, whereas MAE represents the average absolute difference between the actual and predicted values [18]. Correlation and squared correlation were used to assess the degree of agreement between the actual and predicted schedule-overrun values.

The reported performance values were generated from the out-of-fold predictions obtained through 10-fold cross-validation. In this procedure, the dataset was divided into ten folds, with each fold used once as a validation subset while the remaining nine folds were used for model training. This process was repeated until every observation had been evaluated outside the training subset used to generate its prediction [18].

The results are presented through an overall comparison of model performance, graphical comparisons of the evaluation metrics, actual-versus-predicted results, residual-error analysis, and model-interpretability outputs. The interpretation of the findings and their comparison with previous studies are provided separately in Section 5.

### *4.1 Overall Predictive Performance*

Table 4 presents the predictive performance of the six evaluated algorithms based on the results generated through 10-fold cross-validation.

**Table 4**. Predictive performance of the evaluated models

| Algorithm | RMSE ↓ | MAE ↓ | Squared Correlation ↑ | Correlation ↑ |
|---|---|---|---|---|
| **Linear Regression** | 5.253 | 4.178 | 0.834 | 0.913 |
| **Regression Tree** | 6.024 | 4.431 | 0.787 | 0.887 |
| **Random Forest** | 4.822 | 3.736 | 0.866 | 0.930 |
| **Support Vector Regression** | 11.093 | 8.829 | 0.623 | 0.789 |

| | | | | |
|---|---|---|---|---|
| **Gradient Boosting Machine** | **4.578** | **3.503** | 0.874 | **0.935** |
| **Deep Neural Network** | 4.585 | 3.553 | **0.875** | **0.935** |

As shown in Table 4, Gradient Boosting Machine achieved the lowest RMSE value of 4.578 and the lowest MAE value of 3.503. Deep Neural Network produced nearly identical results, with an RMSE of 4.585 and an MAE of 3.553. The differences between GBM and DNN were only 0.007 for RMSE and 0.050 for MAE.

Deep Neural Network achieved the highest squared-correlation value of 0.875, followed closely by Gradient Boosting Machine with a value of 0.874. Both models recorded the highest correlation value of 0.935.

Random Forest ranked third, achieving an RMSE of 4.822, an MAE of 3.736, a squared correlation of 0.866, and a correlation of 0.930. Linear Regression recorded an RMSE of 5.253 and an MAE of 4.178, while Regression Tree produced an RMSE of 6.024 and an MAE of 4.431.

Support Vector Regression produced the highest prediction errors among the evaluated models, with an RMSE of 11.093 and an MAE of 8.829. It also recorded the lowest squared correlation and correlation values, at 0.623 and 0.789, respectively.

Based on the RMSE and MAE values, the models were ranked as follows: Gradient Boosting Machine, Deep Neural Network, Random Forest, Linear Regression, Regression Tree, and Support Vector Regression. Although DNN achieved a marginally higher squared-correlation value, GBM produced the lowest values for both direct prediction-error metrics.

While Table 4 summarizes the numerical performance of the evaluated models, graphical comparisons provide a clearer visualization of the differences among the algorithms. The following subsections compare the models using each evaluation metric individually to facilitate interpretation of their predictive performance.

### *4.2 Comparative Analysis of Evaluation Metrics*

Figures 4.1–4.3 provide a visual comparison of the six evaluated models based on RMSE, MAE, and squared correlation. These figures complement the numerical results reported in Table 4 and facilitate direct comparison of the differences in predictive performance across the models.

#### *4.2.1 RMSE Comparison*

Figure 4.1 presents the RMSE values obtained by the six regression models. Gradient Boosting Machine recorded the lowest RMSE value of 4.578, followed closely by Deep Neural Network at 4.585 and Random Forest at 4.822. Linear Regression achieved an RMSE of 5.253, while the Regression Tree recorded a higher value of 6.024. Support Vector Regression produced the highest RMSE value of 11.093.

The difference between GBM and DNN was limited to 0.007, indicating nearly identical performance according to this metric. In contrast, the RMSE produced by SVR was more than twice the value obtained by GBM.

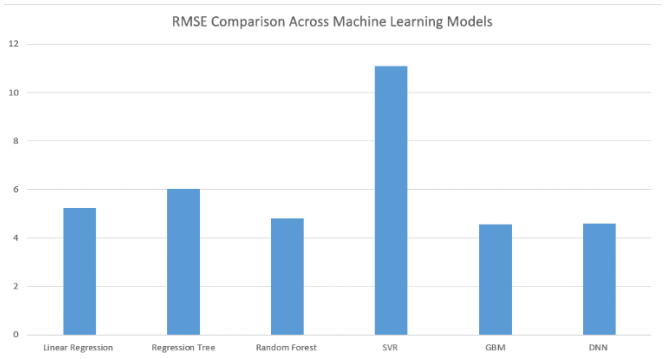


**Figure 4.1**. RMSE comparison across the evaluated machine-learning models.
**Source**: Developed by the authors using RapidMiner Studio.

#### *4.2.2 MAE Comparison*

Figure 4.2 compares the MAE values of the evaluated models. Gradient Boosting Machine achieved the lowest MAE of 3.503, followed by Deep Neural Network with 3.553 and Random Forest with 3.736. Linear Regression and Regression Tree recorded MAE values of 4.178 and 4.431, respectively. Support Vector Regression produced the highest MAE value of 8.829.

The MAE results followed the same overall ordering observed for RMSE. GBM achieved the smallest average absolute deviation between the actual and predicted schedule-overrun values, while SVR showed the largest average prediction error.

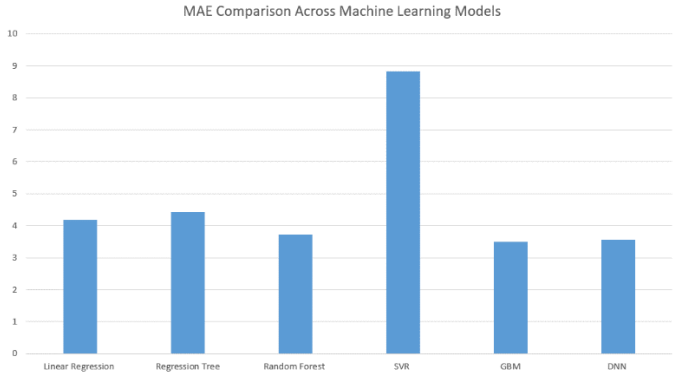


**Figure 4.2**. MAE comparison across the evaluated machine-learning models.
**Source:** Developed by the authors using RapidMiner Studio.

### *4.2.3 Squared-Correlation Comparison*

Figure 4.3 presents the squared-correlation values achieved by the six models. Deep Neural Network recorded the highest squared correlation of 0.875, followed by Gradient Boosting Machine at 0.874 and Random Forest at 0.866. Linear Regression achieved a value of 0.834, whereas the Regression Tree recorded 0.787. Support Vector Regression produced the lowest squared-correlation value of 0.623.

The results indicate that DNN and GBM achieved almost identical agreement between the actual and predicted values. The difference between their squared-correlation values was only 0.001.

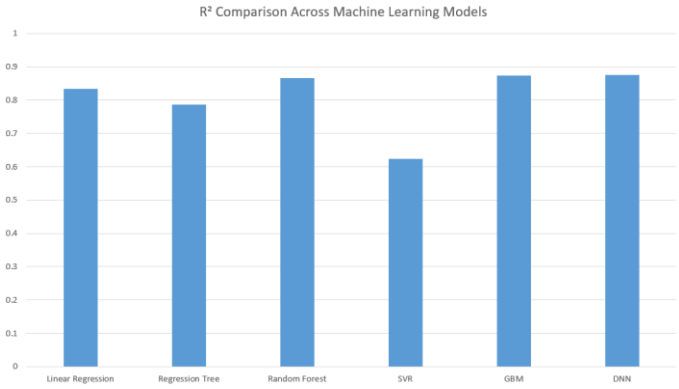


**Figure 4.3**. Squared-correlation comparison across the evaluated machine-learning models.
**Source**: Developed by the authors using RapidMiner Studio.

### *4.2.4 Overall Metric-Based Ranking*

The RMSE and MAE comparisons produced the same model ranking. Gradient Boosting Machine ranked first, followed by Deep Neural Network, Random Forest, Linear Regression, Regression Tree, and Support Vector Regression. The squared-correlation results differed only slightly, with DNN ranking first and GBM ranking second by a margin of 0.001.

Considering the evaluation metrics collectively, GBM and DNN were the two strongest models. GBM achieved the lowest RMSE and MAE, whereas DNN achieved the highest squared correlation. Random Forest consistently ranked third across all reported measures.

## *4.3 Actual-versus-Predicted Analysis*

Actual-versus-predicted plots were used to visually examine the agreement between the observed and predicted schedule-overrun percentages. In each plot, the horizontal axis represents the actual schedule-overrun percentage, while the vertical axis represents the corresponding value predicted by the model. A model demonstrates stronger predictive agreement when its predicted values remain close to the ideal reference line, where the predicted value equals the actual value. The following subsections present the prediction patterns generated by each of the six evaluated models.

### *4.3.1 Linear Regression*

Figure 4.4 presents the actual and predicted schedule-overrun percentages generated by the Linear Regression model. The predicted values followed a clear positive trend as the actual schedule overrun increased, indicating that the model captured the general relationship between the input variables and the target variable. Nevertheless, dispersion remained visible around the

fitted trend, particularly at the lower end of the actual schedule-overrun range. This visual pattern is consistent with the model's RMSE of 5.253, MAE of 4.178, and squared correlation of 0.834.

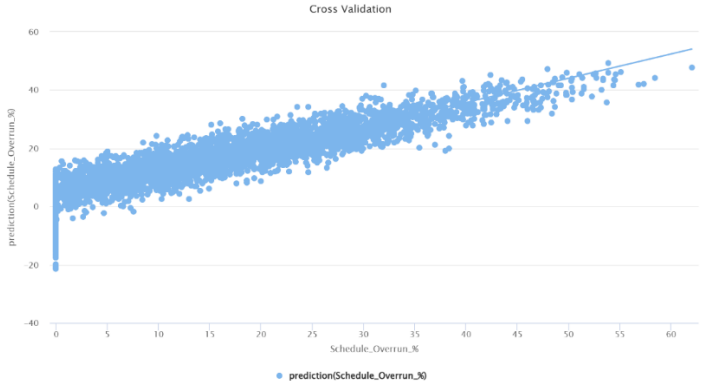


**Figure 4.4**. Actual-versus-predicted schedule-overrun values for the Linear Regression model.
**Source**: Authors' calculations based on the 10-fold cross-validation predictions.

### *4.3.2 Regression Tree*

Figure 4.5 presents the actual and predicted schedule-overrun percentages generated by the Regression Tree model. The predictions followed an overall positive relationship with the observed schedule-overrun values, indicating that the model captured the general trend of the data. However, a noticeably wider dispersion of prediction points was observed compared with the Linear Regression model, particularly for projects with medium and high schedule-overrun percentages. A concentration of predictions was also visible at the lower end of the schedule-overrun range, reflecting the partition-based nature of the Regression Tree algorithm. This visual behaviour is consistent with the model's RMSE of 6.024, MAE of 4.431, and squared correlation of 0.787.

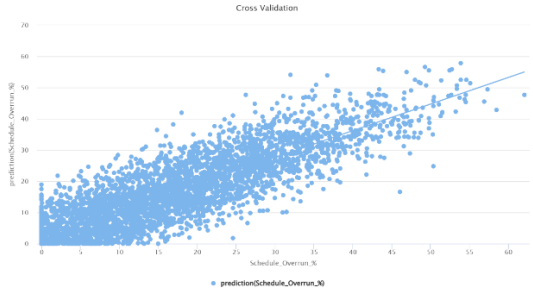


**Figure 4.5**. Actual-versus-predicted schedule-overrun percentages generated by the Regression Tree model.
**Source**: Authors' calculations based on the 10-fold cross-validation predictions.

### *4.3.3 Random Forest*

Figure 4.6 presents the actual and predicted schedule-overrun percentages generated by the Random Forest model. The predicted values exhibited a strong positive relationship with the observed schedule-overrun percentages, with a noticeably tighter distribution around the fitted trend compared with the Regression Tree model. Although a moderate degree of dispersion remained visible, particularly for projects with higher schedule-overrun percentages, the overall prediction pattern demonstrated improved consistency and stability. This visual behaviour is consistent with the model's RMSE of 4.822, MAE of 3.736, and squared correlation of 0.866, indicating substantially better predictive performance than the single Regression Tree model.

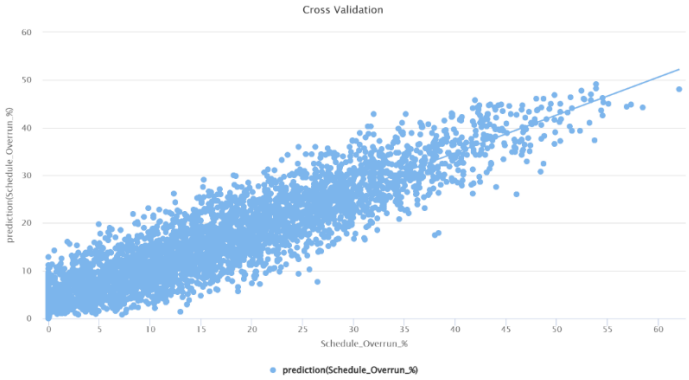


**Figure 4.6**. Actual-versus-predicted schedule-overrun percentages generated by the Random Forest model.
**Source**: Authors' calculations based on the 10-fold cross-validation predictions.

### *4.3.4 Support Vector Regression (SVR)*

Figure 4.7 presents the actual and predicted schedule-overrun percentages generated by the Support Vector Regression (SVR) model. Although the predictions generally followed an increasing trend with the observed schedule-overrun percentages, the model exhibited substantially greater dispersion than the previously evaluated algorithms. Considerable variability was observed across the prediction range, particularly at lower schedule-overrun values where several negative predictions were produced despite the non-negative nature of the target variable. The prediction spread also increased for projects with higher schedule-overrun percentages, indicating weaker agreement between the observed and predicted values. This visual behaviour is consistent with the model's RMSE of 11.093, MAE of 8.829, and squared correlation of 0.623, confirming that SVR achieved the weakest predictive performance among the evaluated models.

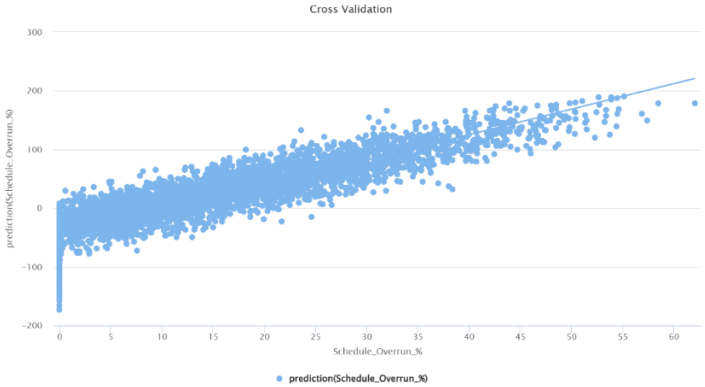


**Figure 4.7**. Actual-versus-predicted schedule-overrun percentages generated by the Support Vector Regression model.
**Source**: Authors' calculations based on the 10-fold cross-validation predictions.

### *4.3.5 Gradient Boosting Machine*

Figure 4.8 presents the actual and predicted schedule-overrun percentages generated by the Gradient Boosting Machine (GBM) model. The predicted values followed the actual schedule-overrun values closely across the entire prediction range, with relatively limited dispersion around the fitted trend. Compared with the Linear Regression and Regression Tree models, the prediction points were more tightly clustered, indicating a stronger agreement between the observed and predicted values. This visual behaviour is consistent with the model's superior predictive performance, reflected by an RMSE of 4.578, MAE of 3.503, and squared correlation of 0.874.

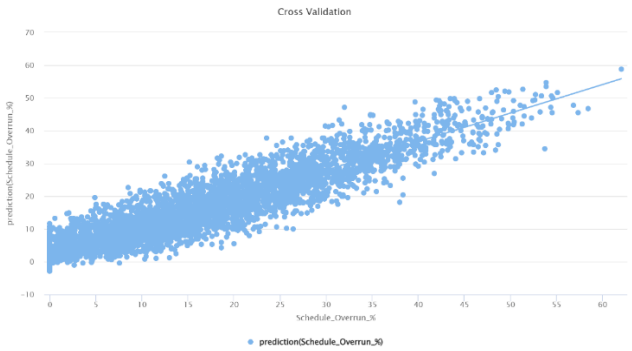


**Figure 4.8**. Actual-versus-predicted schedule-overrun values for the Gradient Boosting Machine model.
**Source**: Authors' calculations based on the 10-fold cross-validation predictions.

### *4.3.6 Deep Neural Network*

Figure 4.9 presents the actual and predicted schedule-overrun percentages generated by the Deep Neural Network (DNN) model. Similar to the Gradient Boosting Machine, the predicted values closely followed the actual schedule-overrun values throughout the prediction range, with only limited dispersion around the fitted trend. The prediction pattern demonstrates a strong agreement between the observed and predicted values, particularly for medium and high schedule-overrun percentages. This visual behaviour is consistent with the model's excellent predictive performance, reflected by an RMSE of 4.585, MAE of 3.553, and squared correlation of 0.875. Although the DNN achieved the highest squared-correlation value among the evaluated models, its improvement over the Gradient Boosting Machine was marginal, confirming that both approaches provided almost identical predictive accuracy.

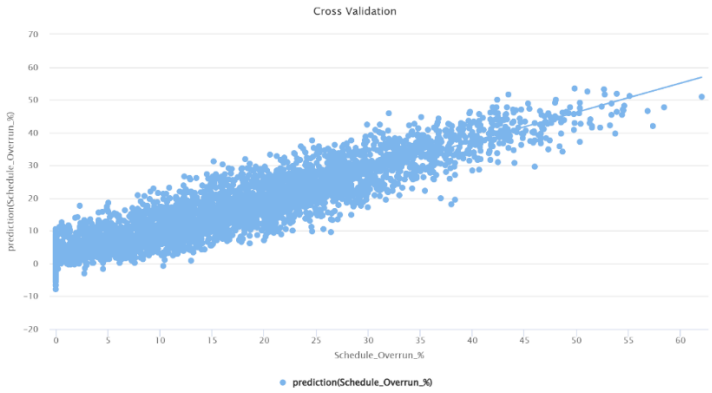


**Figure 4.9**. Actual-versus-predicted schedule-overrun values for the Deep Neural Network model.
**Source**: Authors' calculations based on the 10-fold cross-validation predictions.

### *4.4 Model Interpretability Results*

In addition to evaluating predictive accuracy, model-specific outputs were examined to identify the relative contribution and role of the predictor variables. The available interpretability results included the estimated coefficients of the Linear Regression model, the splitting structure of the Regression Tree, and the predictor-importance weights generated by the Random Forest and Gradient Boosting Machine models. These results provide complementary information regarding how the evaluated algorithms used the project attributes when generating schedule-overrun predictions.

### *4.4.1 Linear Regression Coefficients*

**Table 5**. Estimated coefficients of the Linear Regression model

| Predictor | Coefficient | Standard Error | Standardized Coefficient | t-Statistic | p-Value |
|---|---|---|---|---|---|
| **Functional Size** | 0.000 | 0.000 | 0.007 | 1.221 | 0.222 |
| **Effort Person-Months** | 0.001 | 0.001 | 0.007 | 1.142 | 0.254 |
| **Team Size** | −0.234 | 0.006 | −0.233 | −38.493 | <0.001 |
| **Risk Factor** | 31.340 | 0.338 | 0.562 | 92.595 | <0.001 |
| **Duration Months** | 0.394 | 0.005 | 0.523 | 86.299 | <0.001 |
| **Client Satisfaction** | −4.002 | 0.055 | −0.444 | −73.201 | <0.001 |

| | | | | | |
|---|---|---|---|---|---|
| **Rating** | | | | | |
| **Methodology Encoded** | −0.156 | 0.070 | −0.014 | −2.227 | 0.026 |
| **Intercept** | 4.880 | 0.423 | — | 11.533 | <0.001 |

**Source**: Authors' calculations based on the fitted Linear Regression model.

Table 5 presents the estimated coefficients of the Linear Regression model. Risk Factor recorded the largest positive standardized coefficient at 0.562, followed by Duration Months at 0.523. Both predictors were statistically significant at the 0.001 level. These results indicate that higher risk-factor values and longer project durations were associated with higher predicted schedule-overrun percentages within the fitted linear model.

Client Satisfaction Rating produced the largest negative standardized coefficient at −0.444, while Team Size recorded a standardized coefficient of −0.233. Both coefficients were statistically significant at the 0.001 level. Methodology Encoded also showed a statistically significant negative coefficient, although its standardized magnitude was comparatively small at −0.014. Functional Size and Effort Person-Months produced standardized coefficients of 0.007 and were not statistically significant at the 0.05 level, with p-values of 0.222 and 0.254, respectively. Accordingly, these two predictors showed limited independent linear contributions after the remaining variables were included in the model.

### *4.4.2 Regression Tree Structure*

Figure 4.10 presents the structure of the fitted Regression Tree model. The tree selected Risk Factor as the root splitting variable, indicating that this attribute provided the first separation of the project observations. The next level of the tree used Duration Months to further divide the observations, while Client Satisfaction Rating was used at the lower level to produce the final prediction groups.

The terminal nodes generated different predicted schedule-overrun percentages according to the combinations of the splitting conditions. The highest terminal prediction was 37.614, while the lowest was 1.879. The resulting structure shows that the model relied primarily on Risk Factor, Duration Months, and Client Satisfaction Rating when partitioning the dataset and producing schedule-overrun predictions.


**Figure 4.10**. Structure of the fitted Regression Tree model.
**Source**: Authors' calculations based on the Regression Tree model.

### *4.4.3 Random Forest Feature Importance*

Figure 4.11 presents the predictor-importance weights generated by the Random Forest model. Risk Factor recorded the highest importance weight (0.170), followed by Functional Size (0.164) and Duration Months (0.160). Effort Person-Months and Team Size also contributed substantially, with importance weights of 0.138 and 0.130, respectively. In contrast, Project Size Category Encoded (0.003) and Methodology Encoded (0.040) showed relatively limited contributions to the prediction process. Overall, the Random Forest model distributed predictive importance across several project characteristics rather than relying on a single predictor.

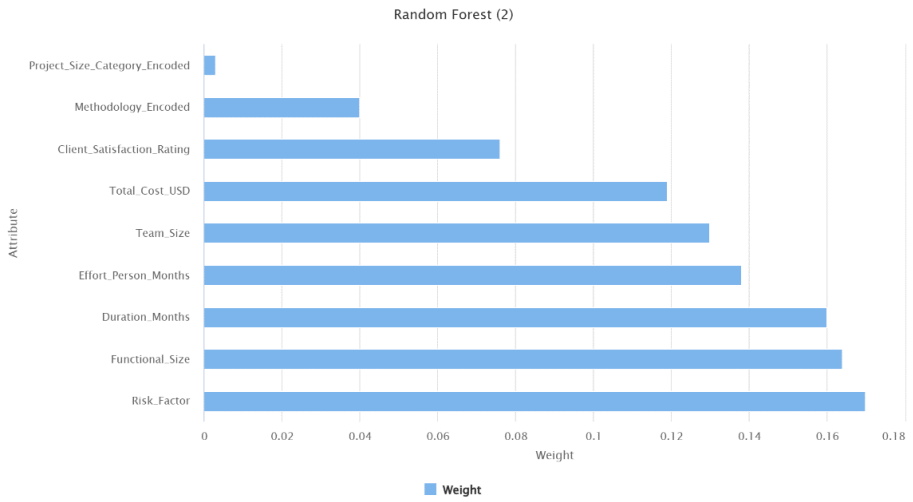


**Figure 4.11**. Predictor-importance weights generated by the Random Forest model.
**Source**: Authors' calculations based on the fitted Random Forest model.

**Table 6**. Predictor-importance weights generated by the Random Forest model

| Predictor | Importance Weight |
|---|---|
| **Risk Factor** | 0.170 |
| **Functional Size** | 0.164 |
| **Duration Months** | 0.160 |
| **Effort Person-Months** | 0.138 |
| **Team Size** | 0.130 |
| **Total Cost USD** | 0.119 |
| **Client Satisfaction Rating** | 0.076 |
| **Methodology Encoded** | 0.040 |
| **Project Size Category Encoded** | 0.003 |

**Source**: Authors' calculations based on the fitted Random Forest model.

Table 6 provides the exact importance weights corresponding to Figure 4.11. The ranking confirms that Risk Factor, Functional Size, and Duration Months were the three most influential predictors in the Random Forest model, whereas Project Size Category Encoded contributed only marginally. These findings demonstrate that the model relied primarily on project risk, project scale, and project duration when estimating schedule-overrun percentages.

### *4.4.4 Gradient Boosting Machine Feature Importance*

Figure 4.12 presents the predictor-importance weights generated by the Gradient Boosting Machine (GBM) model. Risk Factor recorded the highest importance weight (10,091,477), followed by Duration Months (9,212,199) and Client Satisfaction Rating (6,269,215.5). Team Size also contributed to the prediction process, although with a substantially smaller importance weight (1,410,415). In contrast, Functional Size, Effort Person-Months, Total Cost USD, Methodology Encoded, and Project Size Category Encoded received zero importance, indicating that they were not selected by the boosting process when constructing the final ensemble model.

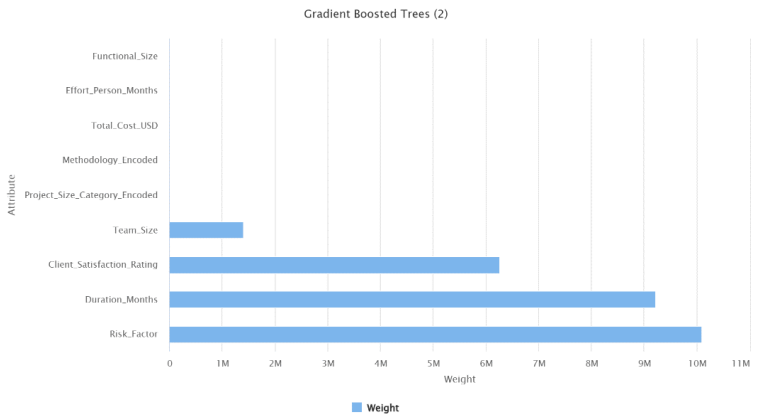


**Figure 4.12**. Predictor-importance weights generated by the Gradient Boosting Machine model.
**Source**: Authors' calculations based on the fitted Gradient Boosting Machine model.

**Table 7**. Predictor-importance weights generated by the Gradient Boosting Machine model

| Predictor | Importance Weight |
|---|---|
| **Risk Factor** | 10,091,477 |
| **Duration Months** | 9,212,199 |
| **Client Satisfaction Rating** | 6,269,215.5 |
| **Team Size** | 1,410,415 |
| **Functional Size** | 0 |
| **Effort Person-Months** | 0 |
| **Total Cost USD** | 0 |
| **Methodology Encoded** | 0 |
| **Project Size Category Encoded** | 0 |

Table 7 provides the exact importance values corresponding to Figure 4.12. The ranking demonstrates that the Gradient Boosting Machine relied primarily on Risk Factor, Duration Months, and Client Satisfaction Rating when generating schedule-overrun predictions. Unlike the Random Forest model, which distributed predictive importance across a broader set of variables, the GBM model concentrated its predictive power on a smaller subset of highly informative predictors. This finding is also consistent with the Regression Tree structure presented earlier, where the same three variables formed the primary decision hierarchy.

### *4.4.5 Cross-Model Comparison of Predictor Importance*

The interpretability outputs showed both similarities and differences in how the evaluated models used the predictor variables. Risk Factor was consistently identified as a highly influential predictor across Linear Regression, Regression Tree, Random Forest, and Gradient Boosting Machine. It recorded the largest standardized coefficient in the Linear Regression model, appeared as the root splitting variable in the Regression Tree, and received the highest importance weight in both Random Forest and GBM. Duration Months also demonstrated consistent predictive relevance. It recorded the second-largest positive standardized coefficient in Linear Regression, was selected at the second level of the Regression Tree, and ranked among the three most influential variables in both ensemble models. Client Satisfaction Rating similarly appeared among the principal predictors across the models, although its contribution differed in direction and magnitude depending on the modeling approach.

The models differed more noticeably in their treatment of the remaining predictors. Random Forest distributed importance across a relatively broad set of project attributes, including Functional Size, Effort Person-Months, Team Size, and Total Cost USD. In contrast, GBM concentrated its importance on Risk Factor, Duration Months, Client Satisfaction Rating, and Team Size, while assigning zero importance to the other five predictors.

Table 8 summarizes the relative role of the principal predictors across the four interpretable models.

**Table 8**. Comparative role of predictors across the interpretable models

| Predictor | Linear Regression | Regression Tree | Random Forest | GBM |
|---|---|---|---|---|
| **Risk Factor** | Strongest positive standardized coefficient | Root splitting variable | Highest importance | Highest importance |
| **Duration Months** | Second-strongest positive coefficient | Second-level split | Third-highest importance | Second-highest importance |
| **Client Satisfaction Rating** | Strong negative coefficient | Lower-level split | Moderate importance | Third-highest importance |
| **Team Size** | Moderate negative coefficient | Not selected | High importance | Fourth-highest importance |
| **Functional Size** | Not statistically significant | Not selected | Second-highest importance | Zero importance |
| **Effort Person-Months** | Not statistically significant | Not selected | Moderate-to-high importance | Zero importance |
| **Total Cost USD** | Not included among significant linear effects | Not selected | Moderate importance | Zero importance |
| **Methodology Encoded** | Small significant effect | Not selected | Low importance | Zero importance |
| **Project Size Category Encoded** | — | Not selected | Lowest importance | Zero importance |

**Source**: Authors' synthesis of the fitted model outputs.

The comparison indicates that Risk Factor, Duration Months, and Client Satisfaction Rating were the most consistently influential variables across the interpretable models. However, the importance assigned to the remaining variables varied according to the structure and learning mechanism of each algorithm. A broader interpretation of these similarities and differences is provided in Section 5.

### 4.5 Summary of Results

This section summarized the empirical results obtained from the six regression models evaluated under identical experimental conditions using 10-fold cross-validation. The comparison demonstrated clear differences in predictive performance across the evaluated algorithms.

Gradient Boosting Machine achieved the lowest prediction errors, with an RMSE of 4.578 and an MAE of 3.503. Deep Neural Network produced closely comparable results, recording an RMSE of 4.585 and an MAE of 3.553. Although DNN achieved the highest squared-correlation value of 0.875, compared with 0.874 for GBM, the difference between the two models was marginal. Both models also recorded a correlation value of 0.935.

Random Forest ranked third, with an RMSE of 4.822, an MAE of 3.736, and a squared correlation of 0.866. Linear Regression achieved moderate predictive performance, followed by the Regression Tree. Support Vector Regression produced the highest RMSE and MAE values and the lowest squared correlation, indicating the weakest predictive performance among the evaluated models.

The actual-versus-predicted plots were consistent with the numerical performance measures. GBM and DNN showed the closest overall agreement between the observed and predicted schedule-overrun percentages, followed by Random Forest. Linear Regression captured the general prediction trend but exhibited greater dispersion, while the Regression Tree produced less consistent predictions. SVR showed the widest deviations between the actual and predicted values.

The model-interpretability results identified several predictors that contributed consistently to schedule-overrun prediction. Risk Factor emerged as the most influential variable across the Linear Regression, Regression Tree, Random Forest, and GBM outputs. Duration Months also showed a strong and consistent contribution, while Client Satisfaction Rating appeared among the principal predictors across multiple models. Random Forest distributed predictive importance across a broader range of variables, whereas GBM concentrated its predictive importance on a smaller subset dominated by Risk Factor, Duration Months, Client Satisfaction Rating, and Team Size.

Overall, the results identified GBM and DNN as the strongest predictive models, with GBM achieving the lowest direct prediction errors and DNN recording the marginally highest squared correlation. The interpretation and implications of these findings, including their relationship to previous research and their practical relevance to software project management, are discussed in Section 5.

## 5. Discussion

This section interprets the empirical findings presented in Section 4 and discusses their implications for predicting schedule overrun in software projects. The discussion considers the comparative performance of the six evaluated models, the close results obtained by Gradient Boosting Machine and Deep Neural Network, the influence of the principal predictor variables, and

the practical relevance of the findings for project monitoring and decision support. The results are also compared with previous studies to identify areas of consistency and difference within the project-delay prediction literature.

### *5.1 Comparative Performance of the Predictive Models*

The results demonstrated that the selection of the learning algorithm had a noticeable effect on the accuracy of schedule-overrun prediction. Gradient Boosting Machine achieved the lowest RMSE and MAE values, while Deep Neural Network produced almost identical error values and recorded the highest squared correlation. Random Forest also performed strongly and ranked third across the principal evaluation measures. Linear Regression and Regression Tree produced moderate results, whereas Support Vector Regression demonstrated the weakest predictive performance among the six evaluated models.

Gradient Boosting Machine achieved an RMSE of 4.578, an MAE of 3.503, a squared correlation of 0.874, and a correlation of 0.935. Its strong performance indicates that the relationship between the project characteristics and schedule-overrun percentage was not entirely linear. Gradient boosting constructs a sequence of weak prediction models, typically decision trees, in which each additional learner is developed to reduce the errors remaining from the preceding ensemble. This sequential error-correction mechanism enables GBM to capture nonlinear relationships and interactions among predictor variables that may not be adequately represented by a conventional linear model [27].

The Deep Neural Network produced closely comparable results, with an RMSE of 4.585, an MAE of 3.553, a squared correlation of 0.875, and a correlation of 0.935. Neural networks can approximate complex nonlinear relationships through multiple interconnected layers that progressively transform the input variables into the final prediction [28]. The high squared-correlation value achieved by the DNN indicates that it captured a substantial degree of agreement between the observed and predicted schedule-overrun values. However, its RMSE and MAE remained marginally higher than those of GBM.

The numerical difference between GBM and DNN was limited to 0.007 in RMSE and 0.050 in MAE, while the difference in squared correlation was only 0.001 in favor of DNN. These differences are extremely small and should not be interpreted as evidence of a substantial performance gap between the two models. GBM may be considered the best-performing model according to the direct error measures, whereas DNN achieved a marginally higher squared correlation. Accordingly, both algorithms demonstrated highly competitive predictive performance under the experimental conditions adopted in this study. Because no statistical significance test was conducted to compare the cross-validated prediction errors, the superiority of GBM over DNN should be interpreted as numerical rather than statistically confirmed. The results establish that GBM produced the lowest observed RMSE and MAE values, but they do not demonstrate that the difference would necessarily remain significant across different samples or datasets.

Random Forest ranked third, producing an RMSE of 4.822, an MAE of 3.736, a squared correlation of 0.866, and a correlation of 0.930. Its performance was substantially stronger than that of the single Regression Tree. A Random Forest combines predictions from multiple decision trees trained using resampled observations and subsets of predictor variables. Aggregating the predictions of multiple trees can reduce the instability and variance associated with an individual tree, thereby producing more consistent predictions [18]. This ensemble structure provides a reasonable explanation for the improvement observed over the standalone Regression Tree in the present study.

The feature-importance results also showed that Random Forest distributed predictive importance across a broader range of project characteristics than GBM. Risk Factor, Functional Size, Duration Months, Effort Person-Months, Team Size, and Total Cost USD all contributed to the Random Forest prediction process. This broader distribution suggests that Random Forest captured information from multiple predictor combinations rather than concentrating its predictive structure on only a small number of variables.

Linear Regression achieved an RMSE of 5.253, an MAE of 4.178, a squared correlation of 0.834, and a correlation of 0.913. These results indicate that the model captured a considerable degree of association between the project characteristics and schedule-overrun percentage. Nevertheless, its prediction errors were higher than those produced by GBM, DNN, and Random Forest. Linear Regression assumes that the expected target value can be represented as a linear combination of the predictor variables. Consequently, its predictive capacity can be restricted when the underlying relationships include nonlinear effects, threshold behaviour, or interactions among variables [18].

The Linear Regression actual-versus-predicted plot showed a clear positive trend, confirming that the model captured the general direction of variation in schedule overrun. However, the greater dispersion around the fitted trend compared with the strongest nonlinear models suggests that a purely linear representation was insufficient to capture all patterns contained in the dataset. At the same time, the Linear Regression model provided directly interpretable coefficients, making it useful for examining the direction and relative magnitude of associations between individual predictors and the target variable.

The Regression Tree produced an RMSE of 6.024, an MAE of 4.431, a squared correlation of 0.787, and a correlation of 0.887. Although the model successfully identified meaningful splitting variables, including Risk Factor, Duration Months, and Client Satisfaction Rating, its prediction accuracy was lower than that of Linear Regression and the two ensemble tree-based models. Regression trees divide the predictor space into a finite number of regions and assign the same predicted value to observations

located within each terminal region. This piecewise-constant prediction structure can limit precision when the target variable is continuous and contains gradual variation [18].

The Regression Tree structure nevertheless provided a clear representation of the principal decision hierarchy in the dataset. Risk Factor formed the root split, followed by Duration Months and Client Satisfaction Rating. These variables also appeared prominently in the Linear Regression and ensemble-model interpretability results, demonstrating consistency in the predictors selected by different learning approaches. However, the stronger results of Random Forest and GBM illustrate the predictive advantage of combining multiple trees instead of relying on a single partitioning structure.

Support Vector Regression recorded an RMSE of 11.093, an MAE of 8.829, a squared correlation of 0.623, and a correlation of 0.789. These were the weakest results among the evaluated models. Its actual-versus-predicted plot showed substantially wider dispersion than the other models and included several negative predictions for observations with relatively low actual schedule-overrun percentages.

The performance of SVR depends on several modeling choices, including the kernel function, regularization parameter, epsilon-insensitive loss margin, and scaling of the input variables [18], [26]. The weak performance observed in this study should therefore not be interpreted as evidence that SVR is generally unsuitable for regression problems. Rather, it indicates that the default configuration applied in the current experiment was less compatible with the structure and scale of the dataset. Because all learners were intentionally evaluated using their default hyperparameter settings, no attempt was made to optimize the SVR kernel, cost parameter, epsilon value, or related settings. A tuned SVR configuration might consequently produce different results.

The comparison also demonstrates that model complexity alone did not determine predictive performance. DNN was one of the strongest models, but GBM achieved marginally lower prediction errors despite having a fundamentally different structure. Similarly, Random Forest substantially outperformed the single Regression Tree, showing that the manner in which multiple learners are combined can be as important as the complexity of an individual learner.

Overall, the findings indicate that nonlinear ensemble and neural-network approaches were more effective than the simpler individual models under the conditions examined in this study. GBM achieved the lowest direct prediction errors, DNN recorded the highest squared correlation, and Random Forest provided the third-best performance while retaining useful predictor-importance outputs. Linear Regression offered moderate accuracy together with high interpretability, while the Regression Tree provided a transparent decision structure at the cost of reduced predictive precision. SVR demonstrated the weakest performance under its default configuration.

The close performance of GBM and DNN also indicates that model selection should not be based exclusively on a single evaluation metric. Predictive error, interpretability, computational requirements, implementation complexity, reproducibility, and ease of integration into project-management decision-support environments should all be considered. Under these considerations, GBM offers a particularly favorable balance between predictive accuracy and model interpretability, while DNN provides nearly equivalent accuracy but a less directly interpretable prediction mechanism.

### *5.2 Comparison with Previous Studies*

The findings of the present study are broadly consistent with previous research demonstrating the potential of machine-learning techniques for predicting project delays. Earlier studies have applied a variety of classification, regression, ensemble, and deep-learning approaches to delay prediction across construction and related project environments [19]–[24], [29]. However, direct numerical comparison should be made cautiously because the studies differ in their datasets, industrial contexts, target-variable formulations, validation procedures, and evaluation metrics.

Gondia et al. developed machine-learning models to support construction-project delay-risk prediction using project risk factors and historical performance data [19]. Their work demonstrated that delay risk can be estimated using data-driven relationships among multiple interconnected project characteristics. The present findings support this general conclusion, as all six evaluated models captured at least some predictive relationship between the project attributes and schedule-overrun percentage. Nevertheless, the current study differs by formulating the target as a continuous schedule-overrun percentage rather than as a categorical delay-risk outcome.

The strong performance of GBM and Random Forest is also consistent with research emphasizing the effectiveness of ensemble learning for project-delay prediction. Egwim et al. developed an ensemble-of-ensembles framework that incorporated bagging, boosting, and stacking techniques and reported improved delay-prediction performance following model optimization [20]. Similarly, Yaseen et al. combined Random Forest with a Genetic Algorithm and reported that the hybrid RF-GA model achieved stronger predictive performance than the comparative approaches considered in their study [21]. These findings support the present observation that combining multiple learners can produce more accurate and stable predictions than relying on a single Regression Tree.

The current results further show that the two ensemble tree-based models did not perform equally. GBM achieved lower RMSE and MAE values than Random Forest, despite both models relying on multiple decision trees. This difference may reflect their distinct learning mechanisms. Random Forest constructs and aggregates multiple trees in parallel, whereas GBM develops trees sequentially to correct the errors remaining from earlier learners [18], [27]. Under the present experimental conditions, the

sequential error-correction mechanism of GBM appears to have captured the predictive patterns more effectively. This interpretation is limited to the dataset and default configurations used in the study and should not be treated as a general conclusion that boosting always outperforms bagging.

The strong performance of the DNN is consistent with the increasing use of deep-learning approaches for project-delay and schedule-estimation problems. Alsulamy evaluated deep-learning algorithms for predicting construction-project delays in Saudi Arabia and demonstrated the potential of advanced neural architectures for identifying complex delay patterns [22]. Cheng et al. also proposed a hybrid deep-learning model that integrated sequential and non-sequential project data for cost and schedule estimation [24]. The DNN result in the present study supports the broader conclusion that neural networks can model nonlinear relationships between project characteristics and schedule outcomes.

However, the DNN did not clearly outperform GBM. Although it achieved the highest squared correlation, its RMSE and MAE were marginally higher. This result indicates that the additional structural complexity of a neural network does not automatically guarantee lower prediction error, particularly when the available dataset is structured tabular data rather than a large sequential, image, or text dataset. The finding therefore complements previous deep-learning studies by showing that a tree-based ensemble can remain highly competitive with a neural network when predicting schedule overrun from conventional project attributes.

Egwim and Alaka compared 27 machine-learning algorithms for construction-delay prediction and identified a Perceptron model as the strongest classifier in their experimental setting [29]. Their findings provide further evidence that neural learning methods can perform effectively in project-delay applications. However, their study treated delay prediction as a classification task and evaluated the models using accuracy, balanced accuracy, area under the receiver operating characteristic curve, and F1-score. In contrast, the present study retained schedule overrun as a continuous variable and evaluated prediction magnitude using RMSE and MAE. Consequently, the relative performance rankings should not be compared directly, because classification and regression address different prediction objectives.

The decision to retain a continuous target variable represents an important distinction from several previous delay-risk studies. Categorizing projects as delayed or not delayed can support high-level risk screening, but it removes information regarding the magnitude of the expected overrun. Dichotomizing or categorizing continuous variables can reduce information and statistical sensitivity [25]. By predicting schedule-overrun percentage directly, the present study provides an estimate of the expected severity of delay rather than only assigning a project to a predefined delay category.

The performance of SVR differed from findings in studies where support-vector methods demonstrated competitive delay-prediction capability. For example, Nassar and Elbisy evaluated Support Vector Machine alongside General Regression Neural Network and Tree Boost models for predicting time delays in marine construction projects [23]. The present SVR model, by contrast, produced the highest error values among the six algorithms. This inconsistency is not necessarily contradictory, because support-vector performance is sensitive to data scaling, kernel selection, regularization, epsilon settings, sample characteristics, and target formulation [18, 26]. Differences between marine construction data and software-project data may also contribute to the contrasting results.

Another difference concerns hyperparameter treatment. Several previous studies employed optimization, hybridization, or specialized model architectures to improve performance [20], [21, 24]. The present research intentionally retained the default learner configurations to ensure that all six algorithms were compared under a consistent and reproducible baseline. This design improves the fairness of the benchmarking procedure but may disadvantage models that are especially sensitive to parameter selection, such as SVR and DNN. Therefore, the reported rankings represent baseline performance under the adopted settings rather than the maximum achievable performance of each algorithm.

The predictor-importance findings also correspond with the broader literature on project-delay determinants. Previous research has emphasized the influence of project risk, uncertainty, complexity, requirements changes, and time pressure on project outcomes [3], [5]–[8], [11]. In the present study, Risk Factor and Duration Months appeared consistently among the most influential predictors across the interpretable models. This agreement suggests that the machine-learning models identified patterns that are conceptually compatible with established project-management research, although the observed feature importance should be interpreted as predictive association rather than evidence of causality.

Overall, the comparison with previous studies supports three principal observations. First, machine-learning methods can provide meaningful predictions of project-delay outcomes across different sectors and target formulations [19]–[24], [29]. Second, ensemble and deep-learning approaches frequently demonstrate strong performance when project outcomes depend on nonlinear relationships and interactions among multiple variables [20]–[22], [24]. Third, differences in model rankings across studies are expected because predictive performance depends on data characteristics, preprocessing, target definition, validation design, and hyperparameter configuration. The present study contributes to this literature by comparing six regression algorithms under identical conditions and by predicting the magnitude of software-project schedule overrun as a continuous percentage rather than reducing the outcome to a categorical delay label.

### *5.3 Interpretation of the Key Predictors*

The model-interpretability results identified Risk Factor, Duration Months, and Client Satisfaction Rating as the most consistently influential predictors of schedule-overrun percentage. Team Size also contributed to the prediction process, although its relative importance differed across the evaluated models. The consistency of these variables across Linear Regression, Regression Tree, Random Forest, and Gradient Boosting Machine suggests that they captured important predictive patterns within the software-project dataset.

#### *5.3.1 Risk Factor*

Risk Factor emerged as the most consistently influential predictor. It recorded the largest positive standardized coefficient in Linear Regression, formed the root split of the Regression Tree, and received the highest importance weight in both Random Forest and GBM. This cross-model agreement indicates that variation in project risk was strongly associated with the predicted magnitude of schedule overrun.

This finding is consistent with project-management research that identifies uncertainty, ambiguity, and interconnected risk conditions as major challenges to project planning and execution [3, 6, 11]. Software and information-technology projects may be particularly exposed to changing requirements, technical uncertainty, resource dependencies, and implementation complexity, all of which can reduce schedule reliability [4, 7, 8].

The importance assigned to Risk Factor also supports the argument that schedule performance cannot be evaluated solely through planned dates and activity durations. Effective prediction requires consideration of the uncertainty surrounding project execution. Previous research on artificial intelligence in project management has similarly emphasized the potential value of data-driven risk assessment for improving schedule monitoring and decision support [14, 16, 17].

However, the present findings should be interpreted as evidence of predictive association rather than causality. The models indicate that Risk Factor was highly useful for predicting schedule overrun, but they do not independently establish that increases in the recorded risk measure directly caused the observed overruns. Causal interpretation would require a different research design and additional control of potential confounding factors.

#### *5.3.2 Duration Months*

Duration Months was the second most consistently influential predictor. It recorded the second-largest positive standardized coefficient in Linear Regression, appeared at the second level of the Regression Tree, ranked among the three most important variables in Random Forest, and received the second-highest importance weight in GBM.

Longer project duration can increase exposure to uncertainty because projects remain active across a greater number of planning and execution periods. During this extended period, changes in requirements, resource availability, technology, dependencies, and stakeholder expectations may accumulate. Traditional scheduling research has long recognized that uncertainty in activity duration and project execution can affect the reliability of planned completion dates [1, 11, 12].

The result does not imply that long projects must necessarily experience greater percentage overruns. Rather, Duration Months provided substantial information to the models when combined with the remaining project characteristics. Its importance may reflect nonlinear interactions with risk, team structure, customer satisfaction, and project scale. This is particularly relevant to the tree-based models, in which variable importance may arise from repeated splits and interactions rather than from a single independent relationship.

The prominence of project duration also supports the use of continuous schedule-overrun prediction. A delay of a fixed number of days has a different operational meaning for a short project than for a long project. Predicting the overrun as a percentage preserves this proportional interpretation and allows projects of different planned durations to be compared more meaningfully.

#### *5.3.3 Client Satisfaction Rating*

Client Satisfaction Rating was another influential predictor, although its role differed across the models. In Linear Regression, it produced a substantial negative standardized coefficient, indicating that higher satisfaction ratings were associated with lower predicted schedule-overrun percentages after controlling for the remaining predictors. It also appeared in the lower levels of the Regression Tree, ranked at a moderate level in Random Forest, and received the third-highest importance weight in GBM.

This variable may reflect broader conditions within the relationship between the project team and the client. Lower satisfaction can coincide with unclear expectations, repeated revisions, communication difficulties, acceptance disputes, or dissatisfaction with delivery progress. Requirements uncertainty and scope change are recognized challenges in software-project delivery and can affect planning stability and schedule outcomes [7, 8].

Nevertheless, the direction of interpretation should be treated carefully. Client dissatisfaction may precede schedule problems, but it may also arise as a consequence of delayed or underperforming delivery. The dataset and predictive design do not establish the temporal or causal direction of this relationship. Accordingly, Client Satisfaction Rating should be viewed as an informative project-health indicator rather than as an isolated cause of schedule overrun.

From a practical perspective, the result suggests that stakeholder and client feedback may provide useful information for early schedule-risk monitoring. A decline in satisfaction, when combined with elevated risk and extended duration, could indicate a need for closer review of project expectations, communication practices, requirements stability, and delivery progress.

#### *5.3.4 Team Size*

Team Size showed a more variable pattern across the models. In Linear Regression, it produced a statistically significant negative coefficient, while it received a relatively high importance weight in Random Forest and ranked fourth in GBM. It was not selected in the displayed Regression Tree structure.

The negative Linear Regression coefficient indicates that, after accounting for the remaining predictors, larger teams were associated with lower predicted schedule-overrun percentages in the fitted linear model. One possible interpretation is that larger teams provided greater delivery capacity or access to a wider range of technical skills. However, this interpretation should not be generalized without caution because team size can also introduce coordination, communication, and dependency challenges.

The differing treatment of Team Size across the models illustrates an important distinction between linear coefficients and tree-based feature importance. A regression coefficient describes the estimated direction and magnitude of a variable's linear association while holding the other variables constant. Feature-importance values indicate how useful a variable was for prediction, including its possible nonlinear effects and interactions, but they do not directly identify whether the effect was positive or negative [18].

Consequently, the Random Forest and GBM importance results do not contradict the negative Linear Regression coefficient. Team Size may have contributed through threshold effects or interactions with project duration, functional size, effort, and risk. For example, additional team members may benefit some projects but create coordination costs in others, depending on project complexity and organizational conditions.

#### *5.3.5 Differences in the Remaining Predictors*

The remaining predictors showed less consistent importance across the models. Functional Size and Effort Person-Months were not statistically significant in Linear Regression, but they received comparatively high importance weights in Random Forest. In contrast, GBM assigned zero importance to both variables.

This difference suggests that Functional Size and Effort Person-Months may have provided predictive value through nonlinear relationships or interactions captured by Random Forest, despite showing limited independent linear effects. Random Forest distributes predictions across many randomized trees and may therefore use variables that contribute only within particular regions or combinations of the predictor space [18].

Total Cost USD also contributed to Random Forest but received zero importance in GBM. Methodology Encoded showed a statistically significant but small linear coefficient and low Random Forest importance, while GBM did not use it. Project Size Category Encoded received the lowest Random Forest importance and zero GBM importance.

These differences demonstrate that feature-importance rankings are model-dependent. A variable considered useful by one algorithm may be ignored by another because the algorithms construct predictions differently. Importance values should therefore not be interpreted as universal properties of the variables. They describe how each fitted model used the available information under the experimental conditions of the study.

Overall, the interpretability results indicate that Risk Factor, Duration Months, and Client Satisfaction Rating formed the most stable group of predictors across the evaluated models. Their repeated appearance across linear, single-tree, bagging, and boosting approaches strengthens confidence that they represent meaningful predictive indicators of software-project schedule overrun. Team Size provided additional information, although its role was more dependent on the selected algorithm. These findings support the development of schedule-monitoring approaches that combine traditional progress information with risk conditions, project-duration characteristics, and stakeholder-related indicators.

### *5.4 Practical Implications for Project Management*

The findings have several practical implications for organizations seeking to improve software-project schedule monitoring and decision support. The results indicate that machine-learning models can complement conventional scheduling techniques by identifying patterns in historical project data and estimating the expected magnitude of schedule overrun. Such predictions may support earlier intervention than approaches that rely exclusively on periodic status reporting or retrospective schedule-variance analysis [14–17].

#### *5.4.1 Early Warning and Proactive Intervention*

The continuous prediction of schedule-overrun percentage can provide project managers with an early-warning indicator of potential schedule deterioration. Rather than classifying a project only as delayed or not delayed, the model can estimate the expected severity of the overrun. This distinction allows management teams to differentiate between projects requiring routine monitoring and those requiring immediate corrective action.

For example, a project predicted to exceed its planned duration by a small percentage may be managed through closer progress monitoring, while a project with a substantially higher predicted overrun may require formal escalation, schedule recovery planning, scope review, or resource reallocation. The use of predictive analytics in this manner can support proactive rather than reactive project control [14,16,17].

Predictions should nevertheless be treated as decision-support inputs rather than final managerial decisions. Model outputs must be reviewed alongside project context, recent operational developments, contractual obligations, critical dependencies, and professional judgment. A high predicted overrun should trigger investigation, not automatic intervention without validation.

### *5.4.2 Risk-Based Project Prioritization*

Risk Factor was consistently identified as one of the most influential predictors across the interpretable models. This finding suggests that organizations should integrate predictive schedule monitoring with their existing risk-management processes. Projects with both elevated risk indicators and high predicted schedule-overrun percentages may warrant greater management attention than projects showing only one of these warning signals.

Portfolio and project-management offices could use the predictions to prioritize governance reviews, escalation meetings, and recovery support. When multiple projects compete for limited management or technical resources, predicted schedule exposure can assist in directing attention toward the projects with the highest expected impact.

This approach is consistent with the increasing role of artificial intelligence and predictive analytics in strengthening project governance, risk monitoring, and data-supported decision-making [14–17,30]. However, the prediction model should complement—not replace—formal risk registers, issue logs, milestone reviews, and stakeholder assessments.

### *5.4.3 Monitoring Project Duration and Client Satisfaction*

Duration Months and Client Satisfaction Rating also appeared consistently among the most influential predictors. Their importance indicates that schedule monitoring should extend beyond task-completion data and include broader indicators of project conditions.

For long-duration projects, managers may benefit from conducting more frequent schedule-risk reassessments because extended execution periods increase exposure to changing requirements, resource constraints, technical dependencies, and stakeholder expectations. Long projects should not automatically be considered delayed, but their duration may increase the need for stronger change control, dependency management, and periodic reforecasting.

Client Satisfaction Rating may also function as an early project-health indicator. A decline in satisfaction can signal concerns regarding communication, requirements alignment, delivery quality, progress visibility, or acceptance expectations. When declining satisfaction occurs alongside increasing project risk or predicted schedule overrun, management may need to review stakeholder engagement, scope clarity, decision turnaround times, and change-request handling.

Because client dissatisfaction may be either a precursor to or a consequence of delivery problems, this indicator should not be interpreted causally. Its principal value lies in strengthening the overall monitoring picture when combined with operational and schedule-related information.

### *5.4.4 Resource and Team Management*

Team Size contributed to the predictions, although its role varied across the models. This result suggests that resource decisions should not be based solely on the assumption that adding personnel will necessarily accelerate delivery. Additional team members may increase delivery capacity, but they can also introduce communication overhead, coordination requirements, onboarding effort, and dependency complexity.

Project managers should therefore evaluate team size together with project duration, risk level, functional scope, technical complexity, and required skills. Predictive models may assist by identifying historical combinations of these characteristics that were associated with higher or lower schedule-overrun percentages. Nevertheless, the model should not be used to prescribe a specific staffing level without considering the nature of the project and the capabilities of the assigned team.

### *5.4.5 Selection of a Predictive Model for Operational Use*

The close performance of GBM and DNN indicates that organizations have more than one viable modeling option. DNN achieved the highest squared correlation, whereas GBM produced the lowest RMSE and MAE. However, predictive accuracy is not the only consideration in operational deployment.

GBM offers an advantageous balance between prediction accuracy and interpretability. Its feature-importance output provides information about the variables that contributed most strongly to the predictions. This can help project managers understand why a project may have received a high predicted overrun and can support communication with senior management and stakeholders.

DNN provides nearly equivalent predictive accuracy but is less directly interpretable without additional explanation techniques. It may be appropriate when prediction accuracy is prioritized and sufficient technical capability exists to implement model-explanation methods. Random Forest also represents a practical alternative because it achieved strong performance and distributed importance across a broader range of project characteristics.

Linear Regression may remain useful where transparency and ease of explanation are prioritized over achieving the lowest possible prediction error. Its coefficients provide direct information regarding the direction of associations. The Regression Tree also offers a highly visible decision structure, although its lower predictive accuracy limits its suitability as the primary forecasting model.

Under the conditions of the present study, GBM appears to provide the most favorable operational balance because it achieved the lowest direct prediction errors while retaining useful interpretability. This conclusion applies to the evaluated dataset and default model configurations and should be reassessed through external validation before organizational deployment.

#### *5.4.6 Integration into Project-Control Processes*

A predictive model could be integrated into an existing project-management information system or portfolio dashboard. Project data could be updated periodically, allowing the model to generate revised schedule-overrun predictions throughout the project lifecycle. The resulting dashboard could present:

- predicted schedule-overrun percentage;
- expected completion exposure;
- confidence or uncertainty indicators, where available;
- principal contributing predictors;
- changes in the prediction over time; and
- recommended level of managerial review.

The model output could then be discussed during status reviews, steering-committee meetings, risk workshops, or portfolio-governance sessions. Historical prediction trends may also help determine whether corrective actions are reducing the expected overrun.

Operational use would require clearly defined data-governance procedures. Input variables must be updated consistently, definitions must remain stable, and missing or inaccurate project records must be controlled. Model performance should also be monitored after deployment because project characteristics, organizational practices, and data distributions may change over time.

#### *5.4.7 Responsible Use of Predictive Outputs*

The use of artificial intelligence in project management should remain subject to professional oversight. Predictions may be influenced by limitations in the historical data, measurement choices, or organizational biases. Therefore, model outputs should not be used as the sole basis for evaluating individual employees, approving penalties, or making high-impact contractual decisions.

Managers should be able to review the evidence supporting a prediction and challenge outputs that conflict with recent project information. Maintaining transparency, human oversight, periodic model evaluation, and clear accountability is important when integrating predictive analytics into project governance [14,15,17,30].

Overall, the practical value of the proposed approach lies in enhancing the timeliness and consistency of schedule-risk assessment. When combined with managerial judgment and established control practices, the model can help organizations identify potential overruns earlier, prioritize intervention, and allocate governance attention more effectively.

### *5.5 Research Contributions*

This study makes several methodological, empirical, and practical contributions to the application of machine learning in software-project schedule-overrun prediction.

First, the study provides a controlled comparative evaluation of six regression algorithms: Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network. All models were evaluated using the same dataset, predictor variables, continuous target variable, preprocessing decisions, 10-fold cross-validation structure, and performance measures. Maintaining identical experimental conditions allowed the observed differences in performance to be attributed primarily to the learning characteristics of the algorithms rather than to variations in data preparation or validation procedures.

Second, the study predicts the magnitude of schedule overrun as a continuous percentage rather than reducing the outcome to a binary or multiclass delay category. This formulation preserves information about the severity of the expected overrun and enables managers to distinguish between projects with minor and substantial schedule exposure. Retaining the continuous form of the target also avoids the information loss that can occur when numerical variables are divided into predefined categories [25].

Third, the research provides a comparative baseline using the default configurations of all six algorithms. Although hyperparameter optimization may improve individual model performance, the use of default settings supports methodological consistency and reproducibility. The resulting rankings therefore represent the baseline predictive capability of the algorithms under standardized conditions rather than their maximum possible performance after model-specific optimization.

Fourth, the study combines predictive-performance evaluation with model-interpretability analysis. RMSE, MAE, correlation, and squared correlation were used to compare predictive accuracy, while actual-versus-predicted plots provided visual evidence of prediction behavior. Linear Regression coefficients, the Regression Tree structure, and the feature-importance outputs of Random Forest and GBM were also examined. This combination allowed the study to address not only which model performed best, but also which project characteristics contributed most strongly to the predictions.

Fifth, the cross-model interpretability analysis identified Risk Factor, Duration Months, and Client Satisfaction Rating as the most consistently influential predictors. Their repeated appearance across models with different learning mechanisms strengthens confidence that they represent meaningful predictive indicators within the analyzed dataset. At the same time, differences in the treatment of Functional Size, Effort Person-Months, Team Size, and Total Cost USD demonstrate that predictor importance is dependent on the structure of the fitted algorithm.

Sixth, the study provides empirical evidence that a tree-based ensemble can achieve predictive performance comparable to that of a Deep Neural Network when applied to structured software-project data. GBM achieved the lowest RMSE and MAE, while DNN recorded the marginally highest squared correlation. This finding is practically relevant because GBM retained useful feature-importance information while producing prediction accuracy nearly identical to that of the more complex neural-network model.

Seventh, the findings contribute to the practical development of AI-supported project-control systems. The predicted schedule-overrun percentage could be incorporated into project or portfolio dashboards to support early warning, risk-based prioritization, recovery planning, governance escalation, and resource-allocation decisions. This application is consistent with the growing role of artificial intelligence and predictive analytics in project monitoring and decision support [14–17,30].

Finally, the study contributes specifically to the software-project context. Much of the existing project-delay prediction literature has focused on construction projects or categorical delay-risk outcomes. By evaluating multiple regression algorithms using software-project characteristics and a continuous schedule-overrun target, the present research extends the evidence base for data-driven schedule forecasting in technology-project environments.

Overall, the main contribution of the study lies in integrating controlled algorithm benchmarking, continuous delay-magnitude prediction, visual model diagnostics, and predictor interpretation within a single reproducible framework. The findings provide both an empirical comparison of predictive methods and a foundation for developing practical schedule-overrun decision-support tools.

### *5.6 Discussion Summary*

The discussion showed that GBM and DNN achieved the strongest predictive performance, with only marginal differences between them. GBM recorded the lowest RMSE and MAE, while DNN achieved the highest squared correlation. Random Forest ranked third, whereas SVR produced the weakest results under the default configuration.

The findings were generally consistent with previous studies supporting the use of ensemble and deep-learning methods for project-delay prediction. The interpretability analysis further identified Risk Factor, Duration Months, and Client Satisfaction Rating as the most consistent predictors across the evaluated models.

From a practical perspective, the results support the use of machine-learning models as early-warning and decision-support tools for project managers and PMOs. Among the evaluated algorithms, GBM provided the most favorable balance between prediction accuracy and interpretability. However, the results should be interpreted within the dataset and experimental settings used in this study. The main limitations are presented in the following section.

## 6. Limitations

This study has several limitations that should be considered when interpreting the findings. First, the analysis relied on a single publicly available software-project dataset [31]. Although the dataset provided sufficient records for model development and comparison, it may not fully represent the diversity of software projects across different organizations, countries, industries, and delivery environments.

Second, the models were evaluated using 10-fold cross-validation on the same dataset, without validation on an independent external dataset. The reported results therefore reflect internal predictive performance and may differ when the models are applied to new organizational data.

Third, all algorithms were implemented using their default hyperparameter settings. This supported a consistent and reproducible baseline comparison, but it may not represent the highest performance achievable by each model. Algorithms such as SVR and DNN may particularly benefit from parameter optimization, feature scaling, and architectural tuning.

Fourth, the analysis was limited to the predictor variables available in the dataset. Additional factors such as requirements volatility, team experience, stakeholder engagement, dependency complexity, issue-resolution time, and change-request history could improve schedule-overrun prediction.

Finally, the interpretability comparison was limited to Linear Regression, Regression Tree, Random Forest, and GBM because direct explanatory outputs were not available for SVR and DNN within the implemented workflows. Future research could apply techniques such as SHAP, permutation importance, or sensitivity analysis to provide consistent explanations across all models.

These limitations do not invalidate the findings, but they restrict their generalizability. Future studies should validate the models using additional real-world datasets, optimize the model configurations, and evaluate their performance in operational project-management environments.

## 7. Conclusion

This study compared six regression algorithms for predicting schedule-overrun percentage in software projects using a common dataset, identical predictor variables, default model configurations, and 10-fold cross-validation. The evaluated models were

Linear Regression, Regression Tree, Random Forest, Support Vector Regression, Gradient Boosting Machine, and Deep Neural Network.

Gradient Boosting Machine achieved the lowest prediction errors, with an RMSE of 4.578 and an MAE of 3.503. Deep Neural Network produced nearly identical performance and recorded the highest squared correlation of 0.875. Random Forest ranked third, while Support Vector Regression produced the weakest results under the adopted default settings. Because the differences between GBM and DNN were very small, their relative performance should be interpreted as numerically close rather than decisively different.

The interpretability analysis showed that Risk Factor, Duration Months, and Client Satisfaction Rating were the most consistently influential predictors across the interpretable models. Team Size also contributed to prediction, although its importance varied according to the learning algorithm. These findings indicate that schedule-overrun prediction can benefit from combining conventional project characteristics with risk and stakeholder-related indicators.

Overall, GBM provided the most favorable balance between predictive accuracy and interpretability. Its ability to generate accurate continuous schedule-overrun estimates and identify influential predictors makes it a suitable candidate for project-management decision-support applications. Such a model could support early warning, project prioritization, schedule-recovery planning, and portfolio-level monitoring.

The findings remain limited to the dataset and experimental conditions used in this study. Future research should validate the models using independent organizational datasets, apply systematic hyperparameter optimization, include additional project-risk and delivery variables, and use consistent explainability methods for all algorithms. Evaluating the model within an operational project-management environment would also help determine its practical effectiveness and generalizability.

**8. Data Availability Statement**

The dataset used in this study is publicly available on Kaggle and can be accessed at: https://www.kaggle.com/datasets/alphagamingsdf/software-project-dataset (accessed on 9 May 2026).

**Funding**: This research received no external funding.

**Conflicts of Interest**: The authors declare no conflict of interest.

**ORCID iD: 0009-0002-3614-4770**



**References**

[1] Project Management Institute. (2019). Practice standard for scheduling (3rd ed.).
[2] Grey, S. (2010). Origins of schedule overrun. In Proceedings of the PMI Global Congress 2010—Asia Pacific. Project Management Institute.
[3] Schmidt, J. (2023). Mitigating risk of failure in information technology projects: Causes and mechanisms. Project Leadership and Society, 4, 100097. https://doi.org/10.1016/j.plas.2023.100097
[4] Flyvbjerg, B., & Budzier, A. (2011). Why your IT project may be riskier than you think. Harvard Business Review, 89, 23–25.
[5] Kuutila, M., Mäntylä, M., Farooq, U., & Claes, M. (2020). Time pressure in software engineering: A systematic review. Information and Software Technology, 121, 106257. https://doi.org/10.1016/j.infsof.2020.106257
[6] Pich, M. T., Loch, C. H., & De Meyer, A. (2002). On uncertainty, ambiguity, and complexity in project management. Management Science, 48, 1008–1023. https://doi.org/10.1287/mnsc.48.8.1008.163
[7] Haleem, M., Farooqui, M. F., & Faisal, M. (2021). Tackling requirements uncertainty in software projects: A cognitive approach. International Journal of Cognitive Computing in Engineering, 2, 180–190. https://doi.org/10.1016/j.ijcce.2021.10.003
[8] Komal, B., Janjua, U. I., Anwar, F., Madni, T. M., Cheema, M. F., Malik, M. N., & Shahid, A. R. (2020). The impact of scope creep on project success: An empirical investigation. IEEE Access, 8, 125755–125775. https://doi.org/10.1109/ACCESS.2020.3007098
[9] Meckenstock, J.-N. (2024). Shedding light on the dark side—A systematic literature review of the issues in agile software development methodology use. Journal of Systems and Software, 211, 111966. https://doi.org/10.1016/j.jss.2024.111966
[10] Project Management Institute. (2020). The standard for earned value management.
[11] Liu, J.-Y. (2012). Schedule uncertainty control: A literature review. Physics Procedia, 33, 1842–1848. https://doi.org/10.1016/j.phpro.2012.05.293
[12] Trietsch, D., & Baker, K. R. (2012). PERT 21: Fitting PERT/CPM for use in the 21st century. International Journal of Project Management, 30, 490–502. https://doi.org/10.1016/j.ijproman.2011.09.004
[13] Lipke, W. (2003). Schedule is different. The Measurable News, 31–34.

[14] Fridgeirsson, T. V., Ingason, H. T., Jonasson, H. I., & Gunnarsdottir, H. (2023). A qualitative study on artificial intelligence and its impact on the project schedule, cost and risk management knowledge areas as presented in PMBOK®. Applied Sciences, 13, 11081. https://doi.org/10.3390/app131911081
[15] Nenni, M. E., De Felice, F., De Luca, C., & Forcina, A. (2025). How artificial intelligence will transform project management in the age of digitization: A systematic literature review. Management Review Quarterly, 75, 1669–1716. https://doi.org/10.1007/s11301-024-00418-z
[16] Schützko, F., & Timinger, H. (2023). Predictive analytics for project management. In Proceedings of the 2023 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC) (pp. 1–8). IEEE. https://doi.org/10.1109/ICE/ITMC58018.2023.10332400
[17] Santos, J. I., Pereda, M., Ahedo, V., & Galán, J. M. (2023). Explainable machine learning for project management control. Computers & Industrial Engineering, 180, 109261. https://doi.org/10.1016/j.cie.2023.109261
[18] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning: With applications in R (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1
[19] Gondia, A., Siam, A., El-Dakhakhni, W., & Nassar, A. H. (2020). Machine learning algorithms for construction projects delay risk prediction. Journal of Construction Engineering and Management, 146, 04019085. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001736
[20] Egwim, C. N., Alaka, H., Toriola-Coker, L. O., Balogun, H., & Sunmola, F. (2021). Applied artificial intelligence for predicting construction projects delay. Machine Learning with Applications, 6, 100166. https://doi.org/10.1016/j.mlwa.2021.100166
[21] Yaseen, Z. M., Ali, Z. H., Salih, S. Q., & Al-Ansari, N. (2020). Prediction of risk delay in construction projects using a hybrid artificial intelligence model. Sustainability, 12, 1514. https://doi.org/10.3390/su12041514
[22] Alsulamy, S. (2025). Comparative analysis of deep learning algorithms for predicting construction project delays in Saudi Arabia. Applied Soft Computing, 172, 112890. https://doi.org/10.1016/j.asoc.2025.112890
[23] Nassar, A. H., & Elbisy, A. M. (2024). A machine learning approach to predict time delays in marine construction projects. Engineering, Technology & Applied Science Research, 14, 16125–16134. https://doi.org/10.48084/etasr.8173
[24] Cheng, M.-Y., Vu, Q.-T., & Gosal, F. E. (2025). Hybrid deep learning model for accurate cost and schedule estimation in construction projects using sequential and non-sequential data. Automation in Construction, 170, 105904. https://doi.org/10.1016/j.autcon.2024.105904
[25] Altman, D. G., & Royston, P. (2006). The cost of dichotomising continuous variables. BMJ, 332, 1080. https://doi.org/10.1136/bmj.332.7549.1080
[26] Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), Advances in neural information processing systems 9 (pp. 155–161). MIT Press.
[27] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29, 1189–1232. https://doi.org/10.1214/aos/1013203451
[28] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521, 436–444. https://doi.org/10.1038/nature14539
[29] Egwim, C. N., & Alaka, H. (2021). A comparative study on machine learning algorithms for predicting construction projects delay. In Proceedings of the Environmental Design and Management International Conference—EDMIC 2021 (pp. 1–24).
[30] Abioye, S. O., Oyedele, L. O., Akanbi, L., Ajayi, A., Davila Delgado, J. M., Bilal, M., Akinade, O. O., & Ahmed, A. (2021). Artificial intelligence in the construction industry: A review of present status, opportunities and future challenges. Journal of Building Engineering, 44, 103299. https://doi.org/10.1016/j.jobe.2021.103299
[31] AlphaGamingSDF. (n.d.). Software project dataset [Data set]. Kaggle. Retrieved May 9, 2026, from https://www.kaggle.com/datasets/alphagamingsdf/software-project-dataset