# American Journal of Data Science and Artificial Intelligence (AJDSAI)

ISSN: 3069-3632 (ONLINE)

VOLUME 2 ISSUE 1 (2026)

PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Data Science and Artificial Intelligence (AJDSAI)

Volume 2 Issue 1, Year 2026
ISSN: 3069-3632 (Online)
DOI: https://doi.org/10.54536/ajdsai.v2i1.7792
https://journals.e-palli.com/home/index.php/ajdsai

# Leakage-Free Early Prediction of Issue Resolution Time in Agile Software Projects: A Comparative Study on Dataset Quality

Subash Kunwar[1*], Bhoj Raj Ghimire[1]

**Article Information**

**Received:** March 21, 2026

**Accepted:** May 16, 2026

**Published:** June 13, 2026

**Keywords**

*Agile Estimation, Issue Resolution Time, Leakage-Free Prediction, Machine Learning, Random Forest, Temporal Validation, XGBoost*

## ABSTRACT

Inaccurate estimation of Issue Resolution Time (IRT) remains a persistent challenge in agile software development, leading to overflow sprints, misallocation of resources, and frequent violations of Service Level Agreement (SLA). Advances in machine learning and predictive analytics offer significant potential to address this challenge by leveraging historical issue-tracking data to generate accurate, data-driven predictions as soon as an issue is created. Early prediction of Issue Resolution Time (IRT) supports sprint planning, risk mitigation, and Service Level Agreement (SLA) compliance in agile software projects. This study proposes a leakage-free prediction framework that restricts features strictly to issue creation time and evaluates Random Forest (RF) and XGBoost under temporal split validation on two datasets: TAWOS (a tracker-native, multi-project dataset) and a Kaggle Jira dataset. A systematic review of 30 papers contextualizes the experimental design by identifying common themes: (i) creation-time metadata strongly predicts resolution duration; (ii) tree ensembles provide dependable baselines; and (iii) temporal and cross-project validation yield more realistic performance estimates than random splits. Experimental results show that XGBoost achieves the best holdout performance on the Kaggle dataset (MAE = 2.278 hours, RMSE = 2.877 hours), while both models perform comparably on TAWOS (MAE ≈ 6.4–6.6 hours). Feature stability analysis across 5-fold temporal cross-validation confirms that creation-time features—particularly Weekday, Hour, Priority, Project_ID, and Assignee_Past_Issues—are consistently the most important predictors, validating the leakage-free design. Residual analysis reveals that XGBoost produces tighter error distributions (Std = 1.93 hours on Kaggle) compared to RF (Std = 5.73 hours), indicating superior prediction consistency.

## INTRODUCTION

Software development teams using agile methodologies need to estimate how long issues and tasks will take to resolve so they can plan realistic sprints and manage risk effectively. Traditional agile estimation techniques based on expert judgment, such as Planning Poker, provide measurable improvements in effort estimation (Komala *et al.,* 2023). More recently, artificial intelligence has been applied to agile project management to enable data-driven decision-making (Haque & Fahad, 2025) and systematic risk management (Bauskar *et al.,* 2024). Recent evidence from Nepal suggests that AI-assisted coding tools are already reshaping software development practice, improving accessibility and rapid prototyping (Poudel *et al.,* 2026).

Previous research in software engineering has been more focused on predicting high-level project efforts using available resources and historical data. However, with the advent of modern project management tools such as Jira, Trello, and ClickUp, which track detailed execution-level data at the issue level, including descriptions, priority, assignee, reporter, and timestamps, the prediction of Issue Resolution Time (IRT) is possible at the time of issue creation. However, the key challenge is to ensure there is no data leakage, i.e., the model must not be able to access data that is available only after the issue is created. Despite the availability of such rich issue-tracking data, organizations continue to struggle with accurate IRT prediction at early lifecycle stages. Existing machine learning approaches frequently incorporate future-derived features during training, leading to inflated accuracy that does not transfer to operational deployment. According to a systematic review by Fernández-Diego et al. (2020); (Pargaonkar, 2023), up to 40% of estimation studies include post-resolution data, producing models that appear accurate in benchmarks but fail in production.

This research addresses these gaps through a leakage-free prediction framework that strictly restricts features to those available at issue creation time. The study is guided by four streams of literature from 30 reviewed papers: (a) issue-level prediction using tracker data (Ardimento *et al.,* 2025); (b) task and project duration estimation (Sousa *et al.,* 2023); (c) agile analytics with explainability (ForouzeshNejad *et al.,* 2025); and (d) systematic reviews discussing evaluation practices and common methodological errors (Rivera Ibarra *et al.,* 2024).

### Research Objectives

The primary objective of this research is to develop and evaluate a leakage-free machine learning framework



[1] Nepal Open University, Manbhawan, Lalitpur, Nepal
[*] Corresponding author's e-mail: subashkunwar1991@gmail.com

for early prediction of Issue Resolution Time in agile software projects. The specific objectives are:

i. To identify creation-time-only features that are predictive of IRT without introducing data leakage, informed by a systematic review of 30 papers.

ii. To compare Random Forest and XGBoost under temporal split validation on two datasets (TAWOS and Kaggle) to establish robust performance baselines.

**Significance of the Study**

This study contributes to the field by providing a methodologically rigorous, leakage-free prediction framework that can be directly deployed for sprint planning and SLA management. By restricting features to creation-time signals and using temporal validation, the results represent realistic, deployment-ready performance estimates rather than optimistic benchmarks. The cross-dataset evaluation on TAWOS (tracker-native, multi-project) and Kaggle (community-sourced) strengthens generalizability claims beyond single-context studies that dominate the current literature.

## LITERATURE REVIEW

The literature relevant to this research spans three interconnected areas.

**Issue-Level Prediction Using Tracker Data**

Ardimento *et al.,* (Ardimento *et al.,* 2025) in their study proved that features provided at issue creation time, namely summary text, issue type, priority, and creation date, were adequate for making predictions regarding resolution turnaround time. ForouzeshNejad *et al.,* (ForouzeshNejad *et al.,* 2025) verified that features provided at issue creation time were essential in developing effective prediction models in agile environments. They used a tree ensemble with SHAP analysis for transparent predictions. Sousa *et al.,* (Sousa *et al.,* 2023) used machine learning techniques for estimating individual task duration in software projects.

**Task and Project Duration Estimation**

Several studies have examined effort and duration estimation using machine learning. Lishner and Shtub (Lishner & Shtub, 2022) used artificial neural networks for project duration prediction with strong temporal modeling capacity but limited interpretability. Previous studies on recurrent neural network–based recommendation systems also demonstrate the effectiveness of sequential learning approaches for predictive analytics in software-related environments (Poudel *et al.,* 2024). Alzeyani and Szabó (Alzeyani & Szabó, 2024) provided a comparative evaluation of model accuracy in agile project management. Satapathy *et al.,* (Satapathy *et al.,* 2016) achieved $R^2 \approx 0.79$ using Random Forest for software effort estimation. Across these studies, MAE and RMSE emerged as the most widely used and robust metrics for IRT prediction baselines (Zakrani *et al.,* 2018), (Abid *et al.,* 2025).

**Systematic Reviews and Methodological Guidance**

Rivera Ibarra *et al.,* (Rivera Ibarra *et al.,* 2024) carried out a systematic mapping of early estimation in agile projects, with special emphasis on the risks of data leakage, for which they recommend the use of features within creation time. Fernández-Diego et al. (2020); (Pargaonkar, 2023) reviewed the literature on effort estimation, from which they concluded that, although random k-fold cross-validation is the most used method, it overestimates accuracy, for which they recommend temporal validation and Leave-One-Project-Out (LOPO) validation. (Pasuksmit et al., 2024); (Uddin *et al.,* 2025) confirmed this, stating that cross-validation with appropriate feature selection is key to accurate estimates.

Aversano, L., et al. (2025). proposed behavioral scoring systems using rough set theory and fuzzy logic for fraud-adjacent transaction risk assessment, illustrating how behavioral analysis can complement traditional feature engineering. explored federated learning approaches for privacy-preserving analytics across distributed datasets, suggesting future directions for decentralized IRT prediction.

Beyond these three streams, dataset representativeness, preprocessing quality, and distributional assumptions are recognized as important factors in prediction model development (ForouzeshNejad *et al.,* 2025), (Priya Varshini *et al.,* 2021). (Priya Varshini *et al.,* 2021) Recommended jointly reporting MAE and RMSE, since RMSE penalizes large deviations more strongly and captures tail risk that MAE alone may obscure.

## MATERIALS AND METHODS

**Research Methodology**

The proposed research methodology follows a seven-phase experimental pipeline designed for leakage-free IRT prediction. It begins with data collection from two datasets (TAWOS and Kaggle), followed by preprocessing to handle missing values, duplicates, and IRT computation. The critical third phase applies leakage-free feature extraction, restricting all features to information available at issue creation time across three categories: temporal, metadata, and historical context. The prepared data then undergoes temporal split validation using a 5-fold expanding window strategy, replacing the random splits that the literature identified as producing inflated accuracy. Two tree-ensemble models, Random Forest and XGBoost, are trained in parallel and evaluated using MAE and RMSE in hours. Finally, a multi-dimensional analysis is conducted covering holdout performance, learning curves, residual distributions, and feature stability to provide a complete picture of prediction reliability across both datasets.

**Datasets**

**TAWOS Dataset**

The Technical and Agile Work-item Open Source (TAWOS) dataset, a tracker-native, multi-project dataset,

is a collection of issue-level records from various open-source projects hosted in publicly available Jira repositories. The features of this dataset are issue type, priority, reporter, assignee, creation date, resolution date, project ID, and textual features such as summary and description. The target variable IRT, or issue resolution time, is defined as the difference between the time of resolution and creation of the issue in hours. Tawos Jira dataset has 458232 records in the issue table.

**Kaggle Jira Dataset**

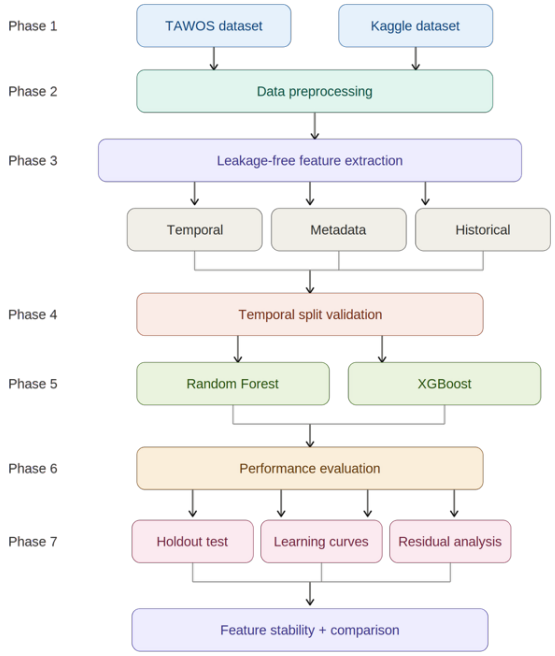


**Figure 1:** Research Methodology

The dataset provided by Kaggle comprises Jira issues for community-driven projects. The dataset is similar in that it also has issue type, priority, and creation and resolution dates. However, it is different in that it is derived from different projects. The Kaggle Jira Dataset has 49147 total records. The use of two datasets for evaluation is important for generalization.

**Data Preprocessing and Feature Engineering**

The data preprocessing stage consisted of several steps applied equally to both data sets. First, missing data in categorical columns, such as assignee and priority, were replaced with the mode, and missing data in numerical columns were replaced with the median. Next, duplicate data points were removed. Lastly, the target variable IRT is calculated as (resolved_date – created_date) in hours. Data points with negative or zero IRT are removed.

The feature engineering stage is unique in that it only uses data available during issue creation time. This ensured the leakage-free constraint. There are three types of engineered features:

Temporal features: Weekday (day of the week of issue creation), Hour (hour of issue creation), IsWeekend (binary), and Month are derived from issue creation timestamp.

Metadata features: Issue Type, Priority, and Project_ID are label-encoded. Title Length and Desc Length are character lengths of the issue summary and description, respectively.

Historical context features: Assignee Past Issues (the number of past resolved issues by assignee), Reporter Past Issues (the number of past resolved issues by reporter), Project Past Issues (the number of past resolved issues in the project), and Age Days (the age of the project in days) are calculated using only data from before issue creation date.

**Experimental Setup**

**Models and Hyperparameters**

Two tree-ensemble models were selected based on the literature review findings: Random Forest (RF) and XGBoost (XGB). Random Forest was Configured ured with 100 estimators, maximum depth unrestricted, minimum samples split of 2, and minimum samples leaf of 1, using scikit-learn's RandomForestRegressor. XGBoost was conFigure ured with 100 estimators, maximum depth of 6, learning rate of 0.1, subsample ratio of 0.8, and column sample by tree of 0.8, using the xgboost.XGBRegressor implementation. Both models used default settings for the remaining parameters.

**Validation Strategy**

Consistent with the literature review findings emphasizing temporal validation over random splits, all experiments used temporal split validation. Issues were sorted by creation date, with earlier issues used for training and later issues for testing. For cross-validation, a 5-fold expanding window strategy was employed: in each fold,

the training set consisted of all issues created before the fold boundary, and the test set comprised issues in the subsequent time window. Holdout evaluation used the final temporal segment (most recent issues) as the test set, with all preceding data for training.

**Evaluation Metrics**

Model performance was assessed using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which were reported in hours. MAE captures the average prediction error magnitude, while RMSE penalizes large deviations more heavily and provides insight into tail risk. Residual distributions (actual – predicted) were analyzed to characterize error spread and bias. Feature stability was assessed by counting how frequently each feature appeared in the top-10 importance rankings across the 5 cross-validation folds.

**Literature Analysis Parameters**

Across the 30 reviewed papers, five parameters were systematically analyzed:

(a) Feature timing and leakage control: whether features were restricted to creation time (Rivera Ibarra *et al.,* 2024), (Pargaonkar, 2023).

(b) Validation design: random splits vs. temporal vs. LOPO strategies (Pargaonkar, 2023), (Priya Varshini *et al.,* 2021).

(c) Dataset type and label definition: representativeness and correctness of IRT computation (Sousa *et al.,* 2023), (Jadhav *et al.,* 2023).

(d) Model family: which algorithms were compared and why (Priya Varshini *et al.,* 2021), (Abid & Ali, 2025).

(e) Accuracy metrics and reporting: MAE, RMSE, $R^2$, and supplementary metrics (Priya Varshini *et al.,* 2021), (Abid & Ali, 2025).

**Methods and Algorithms Across Reviewed Papers**

The reviewed papers show a clear trend in favor of data-driven approaches. The majority of the papers describe a standard data science process from data collection and feature construction to modeling and evaluation. Ensemble methods, such as Random Forest and Gradient Boosting, are popular in the reviewed papers because of their robustness in dealing with different data types and overfitting problems (Zakrani *et al.,* 2018), (Alatawi *et al.,* 2023). Neural network models are used in papers dealing with complex non-linear relationships in data (Polu, 2024), while support vector machines and regression methods are used as baseline methods in some papers (Ben Kraiem *et al.,* 2023).

Validation design is a recurring methodological concern.

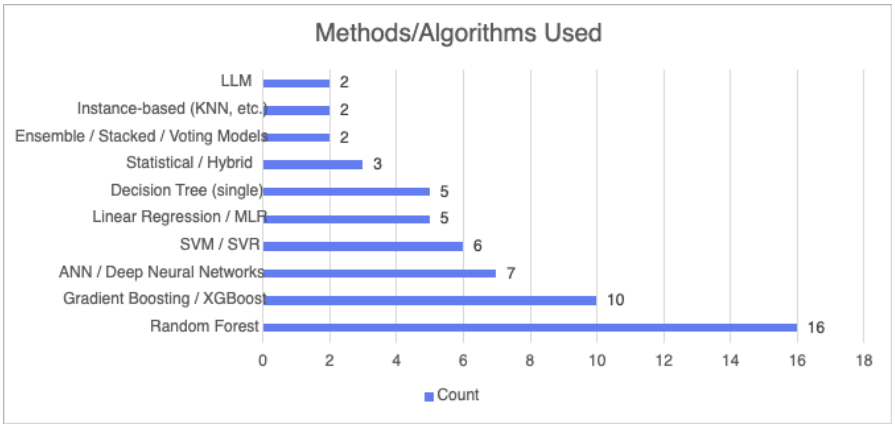


**Figure 2:** Frequency of methods used across 30 reviewed papers.

Random train-test splits, used in approximately 64% of reviewed studies, tend to produce optimistic results that do not generalize to deployment settings. Temporal validation (18%) and LOPO strategies (10%) provide more realistic error estimates (Sousa *et al.,* 2023), (Pargaonkar, 2023).

Recent studies have incorporated explainability methods

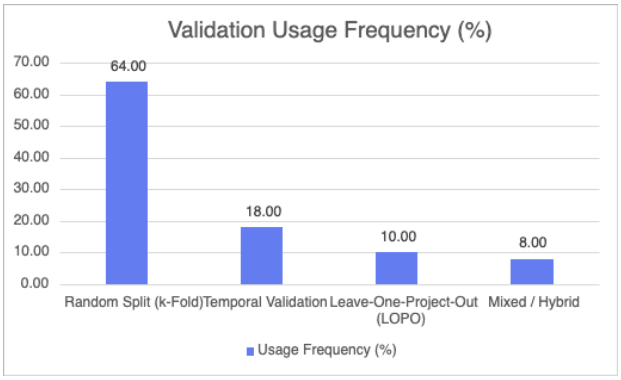


**Figure 3:** Validation usage frequency across reviewed papers.

such as SHAP values and feature importance analysis (Bauskar *et al.,* 2024), (ForouzeshNejad *et al.,* 2025), reflecting a shift toward interpretable models that provide actionable insights for agile practitioners (Meharunnisa *et al.,* 2023).

## RESULTS AND DISCUSSION

### Literature Review Findings

### Model Performance Across Reviewed Papers

Among the 30 analyzed papers, tree-based ensemble models consistently achieved most reliable and interpretable performance. Studies reported $R^2$ values ranging between 0.74 and 0.80 for Random Forest and Gradient Boosting, outperforming linear models and neural networks. Hybrid stacking of RF with GB improved MAE and RMSE by 8–12% over individual models (Jadhav *et al.,* 2023).

Performance across the reviewed literature can be summarized in three tiers. Top-performing methods (RF, GB, and hybrid stacked ensembles) achieved the best balance of accuracy and interpretability, with $R^2 \approx 0.77$–0.79 (Satapathy *et al.,* 2016), (ForouzeshNejad *et al.,* 2025). Moderately strong methods (ANN, SVM) achieved $R^2 \approx$ 0.65–0.72 but required larger samples and longer tuning cycles (Lishner & Shtub, 2022), (Polu, 2024). Baseline methods (Linear Regression, Decision Trees, KNN) served as benchmarks, with MAE and RMSE typically 20–30% higher than ensemble models (Jadhav *et al.,* 2023).

### Feature Importance from Literature

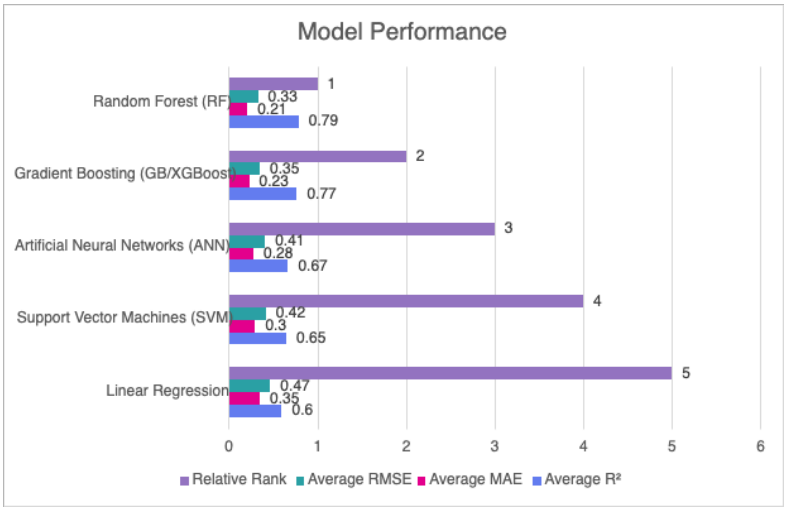


**Figure 4:** Comparative model performance across 30 reviewed papers.

Across the reviewed studies, metadata features (issue type, priority, reporter) and temporal features (creation day, sprint context) were consistently identified as dominant predictors. Table I summarizes the feature importance findings.

### Experimental Results

**Table 1:** Feature Importance Summary from Literature Review

| Feature | Importance Summary |
|---|---|
| Issue Type | Strongest metadata predictor for IRT |
| Priority | Consistently impacts duration estimation |
| Reporter/Assignee | Correlates with experience and speed |
| Creation Time | Key temporal indicator for early prediction |
| Component/Project | Useful for cross-project generalization |

### Holdout Performance

Table II shows the results of the holdout method in terms of MAE and RMSE for both models on both datasets using temporal split validation. For the TAWOS dataset, Random Forest had an MAE of 6.636 hours and an RMSE of 7.901 hours, while XGBoost had an MAE of 6.432 hours and an RMSE of 8.127 hours. This shows that both models have similar performance on the TAWOS dataset, indicating that both models have similar prediction challenges on this heterogeneous multi-project dataset.

For the Kaggle dataset, XGBoost had much better performance than Random Forest, with an MAE of 2.278 hours (which is 55% less than RF's 5.084 hours) and an RMSE of 2.877 hours (which is 51% less than RF's 5.844 hours). This means that the model's predictions are accurate up to 3 hrs, which is exactly what is needed for sprint planning.

### Learning Progress Across Folds

Figure ures 7 and 8 show the MAE progression across 5 temporal folds. On TAWOS, both models exhibit an initial increase in MAE at fold 2 (RF: 7.350, XGB: 7.356 hours), likely due to a distributional shift in the early temporal segments, followed by a steady decline as the training set expands. By fold 5, RF achieves 6.594 hours, and XGB

achieves 6.751 hours, demonstrating convergence.
On Kaggle, XGBoost shows a pronounced downward trend from fold 3 onward, reaching an MAE of 3.066 hours at fold 5 compared to RF's 5.110 hours. This confirms that XGBoost benefits more from additional training history on this dataset, consistent with gradient boosting's sequential error-correction mechanism.

Among the two datasets, the TAWOS data demonstrate stable, reliable, and consistent error distribution. This focuses on the importance of temporal consistency and feature-level quality in agile issue resolution time prediction

**Residual Distribution Analysis**

**Table 2:** Holdout MAE and RMSE (Hours) — Temporal Split

| Dataset | Model | MAE (h) | RMSE (h) |
|---|---|---|---|
| TAWOS | Random Forest | 6.636 | 7.901 |
| TAWOS | XGBoost | 6.432 | 8.127 |
| Kaggle | Random Forest | 5.084 | 5.844 |
| Kaggle | XGBoost | 2.278 | 2.877 |

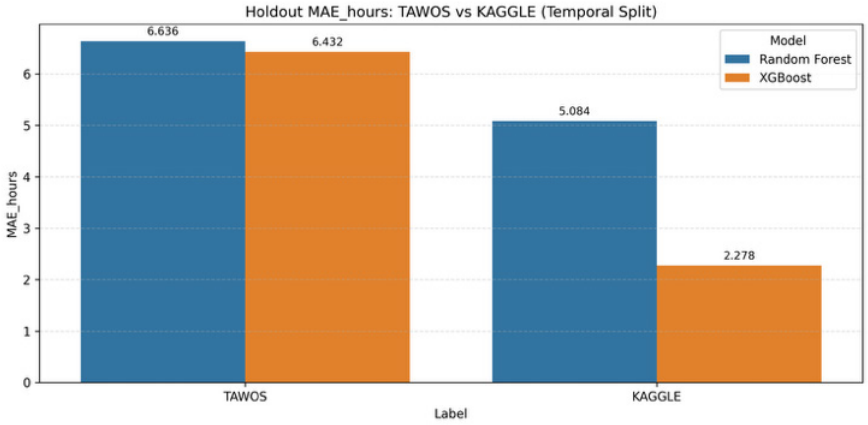


**Figure 5:** Holdout MAE (hours): TAWOS vs Kaggle (Temporal Split).

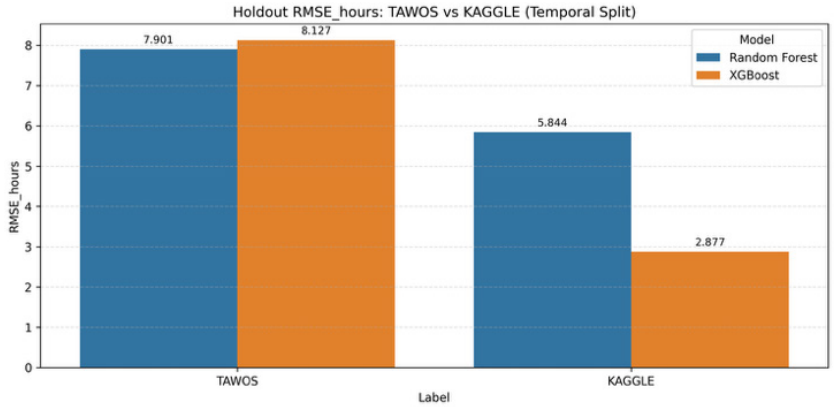


**Figure 6:** Holdout RMSE (hours): TAWOS vs Kaggle (Temporal Split).

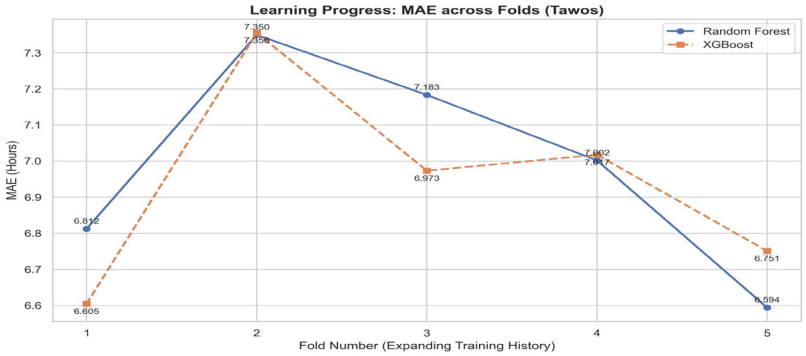


**Figure 7:** Learning progress: MAE across folds (TAWOS).

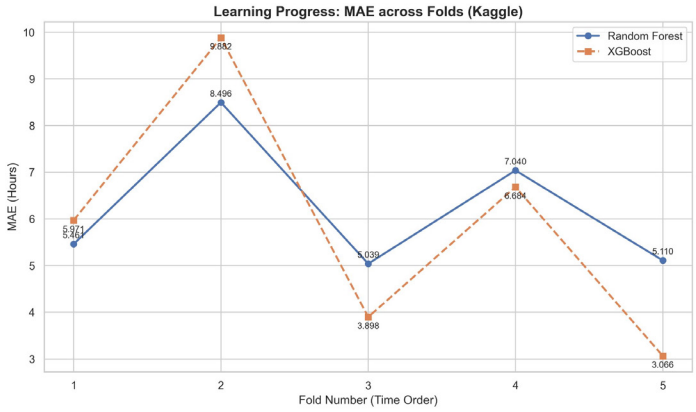


**Figure 8:** Learning progress: MAE across folds (Kaggle).

Figure ures 9 and 10 present the residual (actual – predicted) density plots for both models on each dataset. On TAWOS, both RF (Std: 7.91 hours) and XGBoost (Std: 8.02 hours) exhibit similar right-skewed distributions centered near zero, indicating comparable prediction uncertainty with a tendency to under-predict resolution times for long-running issues.

On Kaggle, the difference is striking: XGBoost's residual distribution is substantially tighter (Std: 1.93 hours) with errors concentrated around zero, while RF shows a wider, bimodal spread (Std: 5.73 hours) with significant mass at −7 and +5 hours. This confirms XGBoost's superior consistency on the Kaggle dataset and highlights that RF may struggle with certain issue subpopulations.

## Feature Stability Analysis

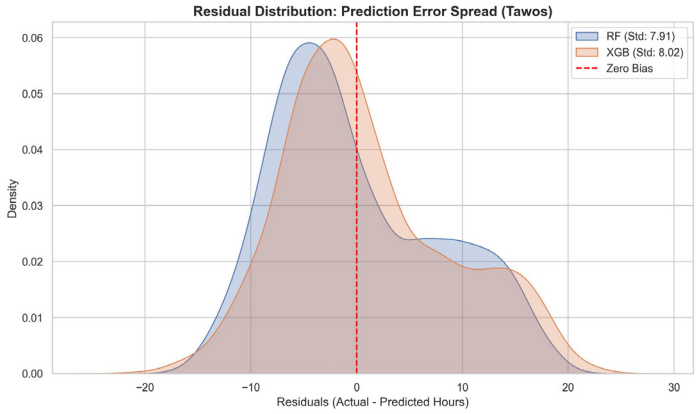


**Figure 9:** Residual distribution: prediction error spread (TAWOS).

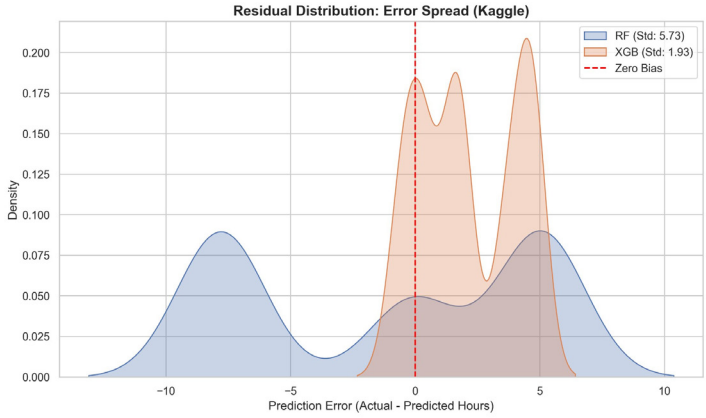


**Figure 10:** Residual distribution: prediction error spread (Kaggle).

The stability of feature stability was evaluated using the number of times each feature occurred in the top-10 importance rankings among the 5-fold cross-validation runs. Features appearing in all 5 runs were considered to be perfectly stable.

On TAWOS, the most stable features for RF were Weekday, Hour, Project_ID, Assignee_Past_Issues, Priority, and IsWeekend (all 5/5 folds). XGBoost showed

similar stability with Hour, Weekday, Assignee_Past_Issues, Project_ID, and Priority achieving 5/5 folds. On Kaggle, IsWeekend, Assignee_Past_Issues, and Type were perfectly stable for RF, while Weekday, Type, Project_ID, IsWeekend, and Age_Days achieved 5/5 for XGBoost.

**Discussion**

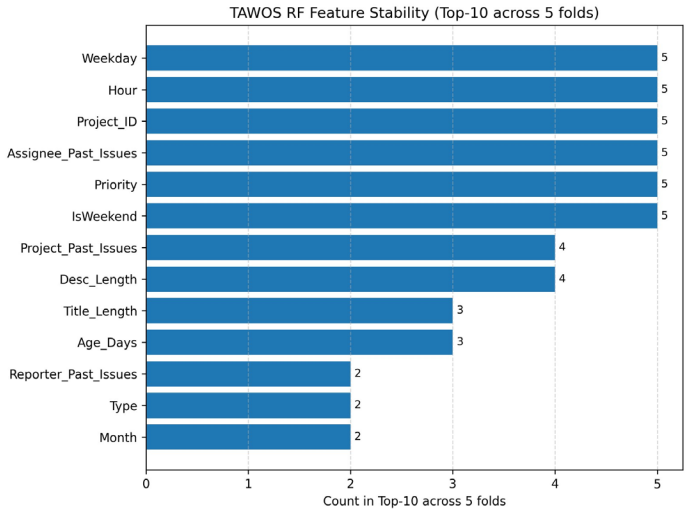


**Figure 11:** TAWOS RF feature stability (top-10 across 5 folds).

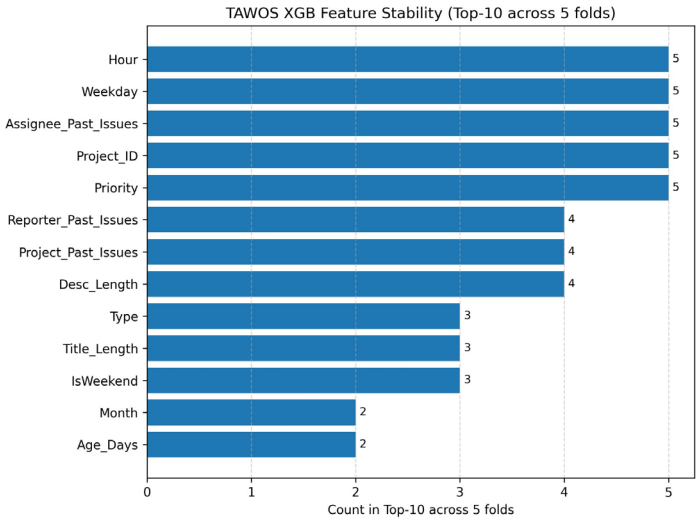


**Figure 12:** TAWOS XGBoost feature stability (top-10 across 5 folds).

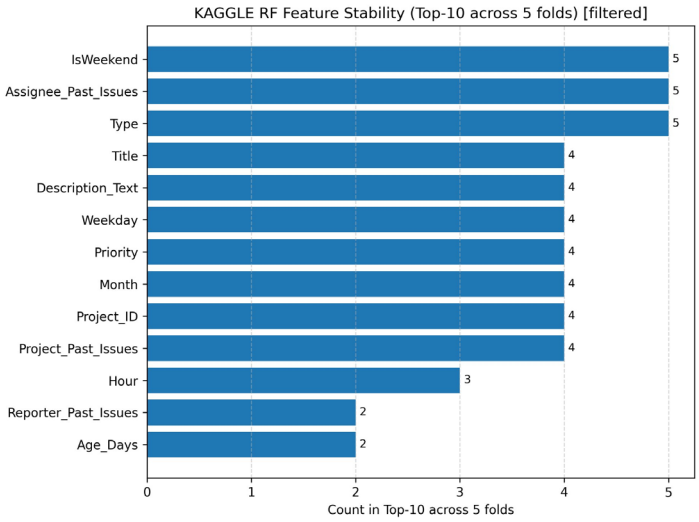


**Figure 13:** Kaggle RF feature stability (top-10 across 5 folds).

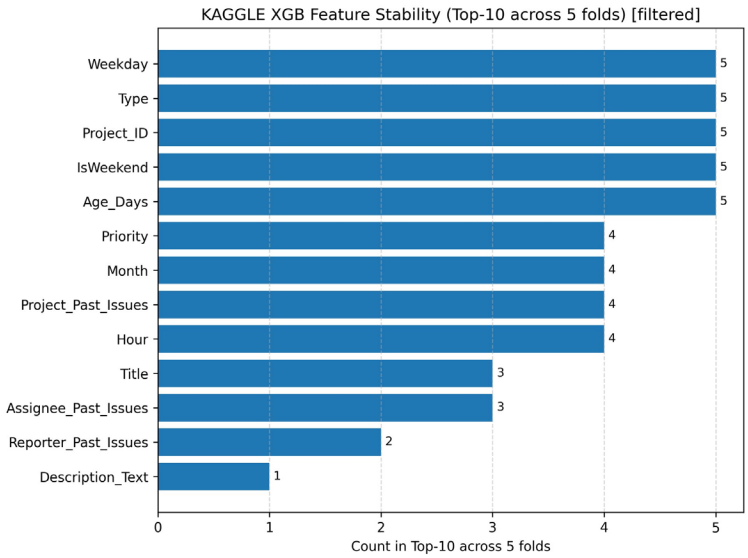


**Figure 14:** Kaggle XGBoost feature stability (top-10 across 5 folds).

**Table 3:** Most Stable Features Across Both Datasets (Fold Count out of 5)

| Feature | TAWOS (RF / XGB) | Kaggle (RF / XGB) |
|---|---|---|
| Weekday | 5 / 5 | 4 / 5 |
| Priority | 5 / 5 | 4 / 4 |
| Project_ID | 5 / 5 | 4 / 5 |
| Assignee_Past_Issues | 5 / 5 | 5 / 3 |
| IsWeekend | 5 / 3 | 5 / 5 |
| Hour | 5 / 5 | 3 / 4 |

### Interpretation of Experimental Results

The experimental results confirm the central hypothesis that creation-time features are sufficient for meaningful IRT prediction without requiring post-creation information. The performance gap between TAWOS (MAE ≈ 6.4–6.6 hours) and Kaggle (XGBoost MAE = 2.278 hours) can be attributed to dataset characteristics: TAWOS spans multiple heterogeneous projects with diverse workflows and team structures, introducing greater variability that is inherently harder to predict. The Kaggle dataset, while still multi-project, may contain more homogeneous resolution patterns.

The substantial advantage of XGBoost over Random Forest on Kaggle (55% lower MAE) is consistent with gradient boosting's sequential error-correction mechanism, which is particularly effective when systematic patterns exist in the residuals. On TAWOS, where residual patterns are more diffuse, both models converge to similar performance, suggesting that the prediction floor for this dataset is approximately 6.4–6.6 hours MAE with creation-time features.

### Comparison with Literature

The literature review found that ensemble models achieve $R^2 \approx 0.77$–$0.79$ across reviewed studies, with MAE typically reported as normalized values. Our results are consistent with these findings: tree ensembles outperform all other model families, and temporal validation produces more conservative but realistic estimates than random splits. Importantly, our use of temporal validation means the reported MAE values represent deployment-realistic performance, unlike many studies that use random splits and may overestimate accuracy.

The feature stability analysis provides evidence that was largely absent from the reviewed literature. While prior studies identified important features in aggregate, few examined whether the same features remain important across different temporal segments of the data. Our finding that Weekday, Priority, Project_ID, and Assignee_Past_Issues are stable across all 5 folds and both datasets provides strong validation for these features as reliable creation-time predictors.

### Methodological Implications

The residual analysis reveals an important practical finding: RF's bimodal residual distribution on Kaggle (Figure 10) suggests that certain issue subpopulations are systematically mispredicted. This is not captured by aggregate MAE/RMSE alone, reinforcing the literature recommendation to report distributional error characteristics alongside point metrics (Priya Varshini *et al.,* 2021).

The learning curve analysis (Figure s. 7–8) demonstrates that model performance stabilizes with sufficient training

history, but the initial folds show high variance—an important consideration for deployment in new projects with limited historical data. This suggests that a minimum training window (approximately 3 folds of temporal data) is needed before predictions become reliable.

### Limitations

There are some limitations in the present study. First of all, only two types of model families, RF and XGBoost, are compared in the present study, and other types of models, such as neural and deep learning models, might be more suitable in finding non-linear dependencies in the data over time. Second, the textual features used in the present study, such as Title_Length and Desc_Length, are quite naive and might be improved by using NLP embeddings of issue descriptions (Ardimento *et al.,* 2025). Third, in the present study, only temporal split validation is used, and Leave-One-Project-Out validation should be performed in future studies to show the generalizability of the results to other projects.

## CONCLUSION

This study shows that leakage-free prediction of Issue Resolution Time is possible using creation time features and temporal validation. Based on 30 papers examined in this study and experimental validation on two datasets, we have the following findings:

Tree-based ensemble models (Random Forest, XGBoost) offer the most reliable trade-off in terms of accuracy, robustness, and interpretability for IRT prediction. XGBoost yields the best performance on the Kaggle dataset (MAE=2.278 hours), whereas both models have similar performance on the more heterogeneous TAWOS dataset (MAE around 6.4–6.6 hours).

Creation time features, namely temporal features (Weekday, Hour, IsWeekend), issue metadata (Priority, Project_ID), and historical context (Assignee_Past_Issues), were always among the most important features, regardless of the model used on both datasets, which is consistent with our leakage-free model design principle.

Using temporal validation yields more conservative but realistic performance estimates than random validation splits, which is in line with recommendations in the literature. The distribution of residuals and learning curves should be monitored in addition to aggregate metrics (MAE, RMSE).

Some of the possible research directions for the future include: (i) extending the validation to LOPO validation schemes using the multi-project structure of TAWOS to assess the cross-project generalizability of the proposed approach; (ii) incorporating NLP embeddings of issue descriptions and summaries as features to better capture their content; (iii) using explainable AI techniques such as SHAP values to offer explanations for individual prediction results in the context of sprint planning decisions; (iv) considering hybrid models using ensemble trees in combination with deep learning for dealing with complex temporal dependencies; (v) using the model in a real-time Jira plugin with online retraining to assess the performance in practice.

## REFERENCES


Abdelali, Z., *et al.,* (2019). Investigating the use of random forest in software effort estimation. Procedia *Computer Science, 343–352.*

Abid, M., & Ali, M. L. (2025). Enhancing software effort estimation: A comparative analysis of machine learning models with correlation-based feature selection. *Sustainable Machine Intelligence Journal, 12,*1–19.

Abid, M., *et al.,* (2025). Enhancing software effort estimation in healthcare informatics. Sustainable *Machine Intelligence Journal, 10,* 50–66.

Alatawi, M. N., *et al.,* (2023). A data-driven artificial neural network approach to software project risk assessment. *IET Software, 2023*(1).

Alzeyani, E. M. M., & Szabó, C. (2024). Comparative evaluation of model accuracy for predicting selected attributes in agile project management. *Mathematics, 12*(16).

Ardimento, P., *et al.,* (2025). A novel LLM-based classifier for predicting bug-fixing time in bug tracking systems. *Journal of Systems and Software, 230.*

Aversano, L., *et al.,* (2025). Time series forecasting for bug resolution using machine learning and deep learning models. *Frontiers in Big Data*, 8.

Bauskar, S. R., *et al.,* (2024). Predictive analytics for project risk management using machine learning. *Journal of Data Analysis and Information Processing, 12*(4), 566–580.

Ben Kraiem, I., *et al.,* (2023). A comparative study of machine learning algorithm for predicting project management methodology. *Procedia Computer Science,* 665–675.

Burga, R., *et al.,* (2022). Examining the transition to agile practices with information technology projects. *International Journal of Project Management, 40*(1), 76–87.

Fernández-Diego, M., *et al.,* (2020). An update on effort estimation in agile software development: A systematic literature review. *IEEE Access, 8*, 166768–166800.

ForouzeshNejad, A. A., *et al.,* (2025). Data-driven predictive modelling of agile projects using explainable artificial intelligence. *Electronics, 14*(13).

Haque, E., & Fahad, F. M. (2025). Artificial intelligence in project management: Enhancing decision-making, efficiency and risk management. *Strategic Data Management and Innovation, 2*(1), 62–77.

Jadhav, A., *et al.,* (2023). Effective software effort estimation leveraging machine learning for digital transformation. *IEEE Access, 11,* 83523–83536.

Komala, C. R., *et al.,* (2023). Innovative cost estimation for agile technology: A novel energy storage technique incorporating modified planning poker. *International Journal of Renewable Energy Research, 13*(4).

Kula, E., *et al.,* (2022). Factors affecting on-time delivery in large-scale agile software development. IEEE Transactions on Software Engineering, 48(9), 3573–

3592.

Lishner, I., & Shtub, A. (2022). Using an artificial neural network for improving the prediction of project duration. *Mathematics, 10*(22).

Meharunnisa, *et al.,* (2023). Analysis of software effort estimation by machine learning techniques. *Ingenierie des Systemes d'Information, 28*(6), 1445–1457.

Pargaonkar, S. (2023). A comprehensive research analysis of software development life cycle (SDLC) agile & waterfall model. *International Journal of Scientific and Research Publications, 13*(8), 120–124.

Pasuksmit, J., *et al.,* (2024). A systematic literature review on reasons and approaches for accurate effort estimations in agile. *ACM Computing Surveys, 56*(11).

Polu, O. R. (2024). Machine learning for predicting software project failure risks. *International Journal of Computer Engineering and Technology, 15*(4), 950–959.

Poudel, S., Maharjan, S., & Luitel, K. (2026). Vibe coding in Nepal: Opportunities and challenges in leveraging AI tools for software development. *American Journal of Applied Research and AI, 1*(1), 44–52. https://doi.org/10.54536/ajarai.v1i1.7057

Poudel, S., Maharjan, S., & Subedi, B. (2024). Session based recommender system using recurrent neural network. *International Journal of Research Publications.* https://doi.org/10.47119/IJRP1001551820247093

Priya Varshini, A. G., *et al.,* (2021). Estimating software development efforts using a random forest-based stacked ensemble approach. *Electronics, 10*(10).

Rivera Ibarra, J. G., *et al.,* (2024). Early estimation in agile software development projects: A systematic mapping study. *Informatics, 11*(4).

Satapathy, S. M., *et al.,* (2016). Early stage software effort estimation using random forest technique based on use case points. *IET Software, 10*(1), 10–17.

Sousa, A. O., *et al.,* (2023). Applying machine learning to estimate the effort and duration of individual tasks in software projects. *IEEE Access, 11*, 89933–89946.

Uddin, S., *et al.,* (2025). Machine learning and deep learning in project analytics: Methods, applications and research trends. *Production Planning and Control, 36*(7), 873–892.

Vyas, M., & Hemrajani, N. (2021). Predicting effort of agile software projects using linear regression, ridge regression and logistic regression. *International Journal of Technical and Physical Problems of Engineering, 47*, 14–19.

Zakrani, A., *et al.,* (2018). Software development effort estimation using random forests: An empirical study and evaluation. *International Journal of Intelligent Engineering and Systems, 11*(6), 300–311.