

Machine Learning Algorithms for IT Project Risk Level Identification: A Multi-Classifer Framework with Systematic Preprocessing and Per-Class Evaluation

Pastima Simanjuntak^{1,2*}, Sunardi¹, Anton Yudhana¹

¹Doctoral Program in Informatics, Universitas Ahmad Dahlan, Yogyakarta, Indonesia

²Department of Informatics Engineering, Universitas Putera Batam, Kepulauan Riau, Indonesia

ABSTRACT

The persistently high failure rate of Information Technology (IT) projects, driven by inadequate risk identification, remains a critical organizational challenge, and conventional risk assessment based on subjective expert judgment struggles to model the non-linear interdependencies in large-scale project data. This research proposes a multi-classifier machine learning (ML) framework for automated IT project risk level identification (Low, Medium, High), built on two contributions: a systematic, auditable preprocessing protocol (explicit target-leakage screening, a controlled before-after comparison of the Synthetic Minority Over-sampling Technique (SMOTE), and hyperparameter optimization) and a per-class ordinal-consistency evaluation that decomposes misclassifications by their severity relative to the ordinal risk structure. Five classifiers were evaluated on 10,919 historical IT project records: Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), and K-Nearest Neighbors (KNN). The leakage screening identified and removed a variable that deterministically encoded the target label. After SMOTE and hyperparameter tuning, SVM achieved the best single-model performance (accuracy 83.65%, macro F1-score 0.759), followed by RF (82.14%, 0.731), LR (80.82%, 0.744), DT (73.72%, 0.669), and KNN (69.64%, 0.596); a five-model soft-voting ensemble reached 83.42% accuracy and a macro F1-score of 0.757. Across all sixteen model/stage configurations (7,165 aggregate misclassifications), the non-adjacent error rate, a High-risk project predicted as Low-risk or vice versa, was exactly zero, a reliability property invisible to aggregate accuracy or F1-score.

Keywords: IT Project Risk Level; Machine Learning; Multi-class Classification; Systematic Preprocessing; Per-Class Evaluation; Data Leakage; *SMOTE*



This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Corresponding Author:

Pastima Simanjuntak,
Doctoral Program in Informatics,
Universitas Ahmad Dahlan
Jalan Ringrod Selatan Kampus 4, Yogyakarta, Indonesia.
Email: 2436083018@webmail.uad.ac.id

1. INTRODUCTION

Information technology (IT) projects constitute critical enablers of organizational transformation, digital innovation, and competitive advantage across industries. However, despite significant advances in project management methodologies, software engineering practices, and agile approaches, IT project failure rates remain a persistent challenge. Empirical studies have consistently demonstrated that a substantial proportion of IT initiatives fail to achieve their initial cost, schedule, or scope objectives, resulting in consequences such as financial losses, productivity disruption, and strategic opportunity costs [1], [2]. Recent analyses of IT project risks have further confirmed that schedule-related issues and resource allocation failures, which can potentially be detected at an early stage, remain among the most dominant contributors to project failure [3].

At the root of many preventable IT project failures lies inadequate risk identification: the systematic process of detecting, characterizing, and prioritizing potential threats before they materialize into realized incidents [4]. Effective early-stage risk level assignment, classifying projects as Low, Medium, or High Risk, enables project managers to proactively direct mitigation resources, adjust staffing, and escalate governance attention toward the most vulnerable initiatives [5][6]. In practice,

however, risk identification still relies heavily on subjective expert judgment, qualitative risk matrices, and experience-based checklists. These approaches are inherently limited in scalability, reproducibility, and their capacity to model multivariate non-linear interactions embedded in large-scale project data [7][8].

Traditional, non-ML risk management continues to rely heavily on structured multi-criteria decision-making (MCDM) and hybrid optimization approaches [9], and on qualitative frameworks tailored to specific development methods such as Scrum [10]; while valuable, these approaches depend on expert-elicited weights and are not designed to learn directly from historical project records at scale. Detailed taxonomies of realized IT project risks also show that resource, schedule, and vendor-related factors recur across a large sample of documented incidents [3], reinforcing the practical need for models that can automatically flag such patterns as projects unfold [11]. The proliferation of ML has created compelling opportunities to automate risk-level identification through data-driven pattern recognition [12], [13], including hybrid ML models that combine classification with predictive analytics [14]. However, two methodological gaps recur across this literature. First, applied studies frequently report resampling and tuning outcomes without a documented audit for target leakage a scenario in which a feature deterministically or near-deterministically encodes the label, producing spuriously inflated performance that reflects the leakage artifact rather than genuine predictive skill. This risk is amplified whenever engineered composite features are built directly from the same underlying severity indicators used to define the target class. Second, risk categories in this domain are ordinal by construction (Low < Medium < High), yet performance is almost universally reported through aggregate metrics, accuracy, macro precision, recall, and F1-score- that treat all misclassifications as equivalent, obscuring whether a model's errors are confined to adjacent categories or occasionally span the full severity range, a distinction with direct managerial consequence.

Prior ML-based approaches to project risk classification illustrate the promise and the gaps. Multiple classifiers applied to software project datasets have achieved accuracy rates of up to 89% for risk percentage prediction [15]. Gradient Boosting combined with t-SNE dimensionality reduction has achieved 85% accuracy in project risk classification tasks [16]. ML ensemble approaches have been applied to investment project risk level prediction across Low, Medium, and High Risk categories [17], and hybrid ML frameworks have achieved 92% accuracy for joint risk and resource management prediction [18]. Despite this growing evidence, a systematic and unified evaluation of five mainstream classifiers LR, SVM, RF, DT, and KNN for three-class IT project risk level identification, combined with an auditable preprocessing protocol and rigorous per-class, severity-aware error analysis, remains limited in the literature [12].

Risk identification in IT projects has been extensively studied from both qualitative and quantitative perspectives. Studies categorizing IT project risks by type and severity identify schedule and resource-related factors among the most prevalent contributors to project failure [3]. DEMATEL-ISM analyses of IT project risk factors indicate that technical complexity and external dependencies are influential drivers of elevated risk [19]. Systematic reviews of risk management practices across the software development lifecycle indicate a progressive transition from ad-hoc approaches toward structured, data-driven frameworks [4]. Reviews contrasting deterministic and probabilistic risk-management approaches in project-based industries similarly report that manual, experience-based practices remain dominant and hinder automation and knowledge transfer across projects [20], while exploratory studies of risk management frameworks specific to IT companies highlight persistent gaps in the systematic integration of quantitative prediction tools into existing governance processes [21].

Comparative evaluations of ML classifiers across related classification tasks provide methodological grounding for this research. Studies comparing SVM, RF, and LR for health risk classification demonstrate competitive performance among these approaches when supported by appropriate preprocessing [22]. Hybrid AI models, such as a Random Forest optimized by a genetic algorithm, have been applied to classify construction projects by delay-risk level [23], while hybrid ML frameworks have also been proposed for joint resource-allocation and IT project risk prediction [18]. RF has been reported to consistently outperform several standalone classifiers in multi-level risk categorization tasks [13], and ANN-based models applied to multi-level construction-risk classification have achieved accuracy levels around 93% [24], with ANN-based risk-prediction approaches also validated in the related context of global software development [25]. Complementing purely algorithmic comparisons, similarity-based prediction using historical project context has been proposed as an alternative to feature-engineered classifiers for software risk forecasting [26], a rationale consistent with the instance-based KNN classifier evaluated in the present study. None of these studies, however,

report a target-leakage audit or a per-class, ordinal-aware decomposition of misclassification patterns, the gap this study addresses.

Table 1. Research Gap Statement

Research (Reference)	Year	Dataset / Domain	ML Method / Approach	Prediction Target	Reported Accuracy	Key Contribution	Limitation / Research Gap
Gouthaman and Sankaranarayanan [15]	2021	Software Project Data	Multiple Machine Learning Classifiers	Risk Percentage Prediction	Up to 89%	Established a benchmark by applying multiple ML classifiers for software project risk prediction.	Did not provide a unified evaluation of multiple classifiers for three-class IT project risk level identification.
Bauskar et al. [16]	2024	Software Project Data	Gradient Boosting + t-SNE Dimensionality Reduction	Risk Level Classification	85%	Improved classification performance by integrating Gradient Boosting with t-SNE feature representation.	Limited comparison with other mainstream classifiers and lacked detailed per-class performance analysis.
Diao [17]	2024	Investment Project Data	Machine Learning Ensemble Approach	Project Risk Level Prediction (Low, Medium, High Risk)	87%	Applied ML ensembles to investment project risk level prediction; demonstrated cross-domain applicability.	Focused on investment projects; not specifically designed for IT project risk classification.
Ismawan and Wahyu [18]	2025	IT Project Data	Hybrid Machine Learning Framework	Joint Risk and Resource Management Prediction	92%	Proposed a hybrid ML framework integrating risk and resource management prediction.	Did not conduct systematic comparison among LR, SVM, RF, DT, and KNN with confusion matrix evaluation.
Research Gap Statement		Although prior studies have reported promising results, a systematic, unified evaluation of five mainstream classifiers (LR, SVM, RF, DT, and KNN) specifically for three-class IT project risk level identification, incorporating rigorous per-class confusion matrix analysis, remains absent from the literature.					

Table 1 shows that the existing literature reveals two critical research gaps addressed by this research: first, the absence of a systematic and unified comparative evaluation framework involving five mainstream machine learning classifiers (LR, SVM, RF, DT, and KNN) specifically for three-class IT project risk level identification (Low, Medium, and High Risk) under consistent preprocessing and evaluation conditions; and second, the limited analysis of classification performance at the class level, particularly through per-class F1-score and confusion matrix evaluation across all risk categories.

This research addresses these gaps through the following primary contributions: (1) a systematic, auditable preprocessing protocol comprising explicit target-leakage screening, encoding, a controlled before-after SMOTE comparison, and hyperparameter optimization; (2) a rigorous comparative evaluation of five classification algorithms (LR, SVM, RF, DT, KNN) under unified preprocessing and evaluation conditions, including a soft-voting ensemble; (3) detailed per-class performance analysis (precision, recall, F1-score, confusion matrix) for Low, Medium, and High Risk categories; and (4) a novel per-class ordinal-consistency evaluation that decomposes misclassification patterns according to their severity relative to the ordinal risk structure quantifying, for the first time in this literature to our knowledge, the extent to which classifier errors respect or violate the ordinal proximity of risk categories.

2. RESEARCH METHOD

2.1 Research Design

The research design follows a systematic, auditable sequence of six stages, illustrated in Figure 1: (1) data collection; (2) target-leakage screening and preprocessing (encoding and scaling); (3) stratified train-test splitting with a controlled SMOTE comparison (baseline on the original training data versus SMOTE applied to the training data only); (4) implementation of five classification algorithms (LR, SVM, RF, DT, KNN); (5) hyperparameter optimization and construction of a soft-voting ensemble; and (6) performance evaluation, including per-class confusion matrices, the before/after SMOTE comparison, and the proposed Ordinal Consistency Rate (OCR) analysis. This sequence extends the conventional preprocess-train-evaluate pipeline by making the leakage screening, the SMOTE before/after comparison, and the ordinal-consistency evaluation explicit, reportable stages rather than implicit

choices.

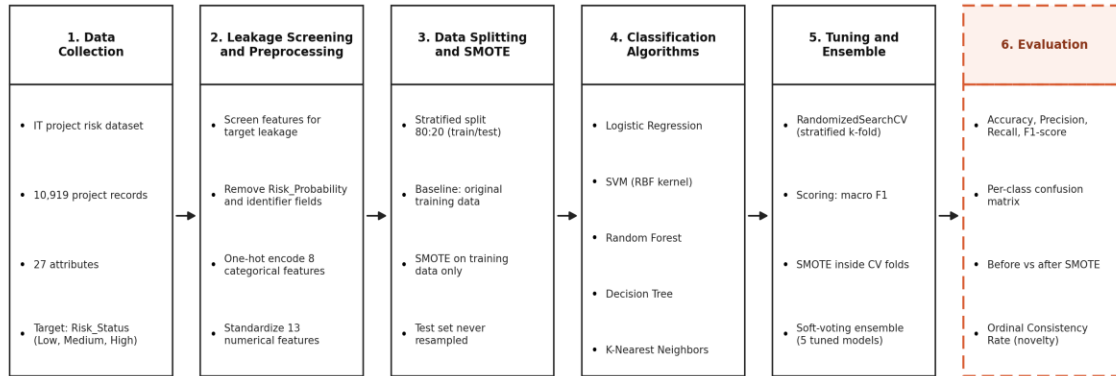


Figure 1. Research design of the proposed multi-classifier framework

2.2 Data Collection

The dataset comprises 10,919 IT project records from Kaggle with 27 attributes spanning project characteristics (Duration_days, Budget_IDR, Team_Size, Complexity), management attributes (Methodology, Vendor_Type, Priority, Project_Type), and quantitative risk indicators (Technical_Difficulty, Requirements_Stability, Resource_Availability, Stakeholder_Engagement, Schedule_Adherence, Budget_Risk, Previous_Project_SuccessRate, Change_Request_Count, Number_of_Vendors, Change_Request_Rate_per_30days, Security_Requirement, Remote_Team, Regulatory). The target variable, Risk_Status, encodes three ordinal structured classes: High Risk ($n = 747$, 6.84%), Medium Risk ($n = 8,375$, 76.71%), and Low Risk ($n = 1,797$, 16.46%), reflecting a real-world class imbalance of approximately 11.21 (majority-to-minority) [25], [26].

The degree of imbalance is summarised by the imbalance ratio (IR), defined as the size of the majority class divided by the size of the minority class:

$$IR = \frac{N_{maj}}{N_{min}} = \frac{8,375}{747} \approx 11.21 \quad (1)$$

2.3 Data Preprocessing

2.3.1 Target-Leakage Screening (Systematic Preprocessing)

Prior to feature selection, every candidate feature was screened for a deterministic or near-deterministic relationship with the target label. This screening identified that the variable Risk_Probability was mapped to Risk_Status through non-overlapping thresholds (Low: 0.0214-0.3300; Medium: 0.3302-0.6600; High: 0.6602-0.9787). Because a classifier including this variable could reconstruct the label with near-perfect accuracy without learning a generalizable risk pattern, Risk_Probability was excluded from the feature space, along with non-informative identifier fields (Project_ID, Project_Name, Start_Date, End_Date). This audit step is reported explicitly because omitting it can produce misleadingly high performance that reflects the leaked variable rather than genuine model skill, a risk that applies equally to any engineered feature built from the same underlying severity components used to define the target [27].

Formally, the screening showed that the label of each record could be written as a deterministic step function of its Risk_Probability value p :

$$y = \begin{cases} \text{Low, if } p \leq \tau_1 \\ \text{Medium, if } \tau_1 < p \leq \tau_2 \\ \text{High, if } p > \tau_2 \end{cases} \quad (2)$$

where $\tau_1 \approx 0.330$ and $\tau_2 \approx 0.660$ are the empirical class boundaries. Because Eq. (2) recovers y exactly from a single input, any model given access to p can reproduce the label without learning a generalizable risk pattern. The same check, i.e., testing whether $y = g(x_j)$ holds for a simple function g of a single feature x_j , was applied to every candidate feature.

2.3.2 Preprocessing, Encoding, and Splitting

- Step 1 - Encoding: 13 numerical features (Duration_days, Budget, Team_Size, Technical_Difficulty, Requirements_Stability, Resource_Availability, Stakeholder_Engagement, Schedule_Adherence, Budget_Risk, Previous_Project_SuccessRate, Change_Request_Count, Number_of_Vendors, Change_Request_Rate_per_30days) were standardized (z-score). Eight categorical features (Complexity, Project_Type, Methodology, Vendor_Type, Security_Requirement, Remote_Team, Regulatory, Priority) were one-hot encoded.

The z-score standardization of numerical feature j for record i is given by

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \quad (3)$$

where μ_j and σ_j are the mean and standard deviation of feature j estimated on the training subset and then applied unchanged to the test subset, so that no test-set information enters the scaling.

- Step 2 - Stratified split: Data were partitioned 80:20 into training (8,735 records) and test (2,184 records) subsets using stratified sampling on Risk_Status to preserve class proportions (random_state = 42).

- Step 3 - Controlled SMOTE comparison: Each classifier was trained and evaluated twice on the identical test set: once on the original imbalanced training data ("before SMOTE"), and once with SMOTE applied exclusively to the training data or training folds ("after SMOTE"), isolating SMOTE's specific, quantified contribution rather than reporting only a final resampled result. The test set was never resampled.

For each minority-class sample $\mathbf{x}_i$, SMOTE [28] generates a synthetic sample by interpolating between $\mathbf{x}_i$ and one of its k nearest minority-class neighbors, denoted $\hat{\mathbf{x}}_i$:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + \lambda \cdot (\hat{\mathbf{x}}_i - \mathbf{x}_i), \lambda \sim U(0, 1) \quad (4)$$

where λ is drawn from a uniform distribution on $[0, 1]$. Synthetic samples are generated until each minority class reaches the size of the majority class.

This controlled comparison is treated as necessary rather than optional: simulation studies of class-imbalance corrections in logistic-regression-based risk prediction report that resampling can systematically distort predicted class probabilities and, in some settings, degrade model calibration more than it improves discrimination [29]. Accordingly, SMOTE's effect is measured empirically per classifier rather than assumed to be beneficial by default.

2.4 Classification Algorithms

Five classification algorithms were implemented and compared within the proposed framework, each representing a different machine learning paradigm for IT project risk classification. LR was employed as a linear probabilistic classifier that models the relationship between project features and risk levels while providing interpretable probability estimates [15]. SVM was utilized to identify optimal decision boundaries and effectively handle non-linear risk patterns through the Radial Basis Function (RBF) kernel [22][15]. RF, an ensemble learning method based on multiple decision trees, was applied to improve prediction robustness, reduce overfitting, and provide insights into feature importance [6][23]. In addition, DT was used as a rule-based classifier capable of generating transparent and easily interpretable decision rules [16], while KNN was implemented as a non-parametric, instance-based method that classifies projects according to the similarity of neighboring data points [30][31]. A soft-voting ensemble combining the five tuned classifiers with equal weighting was constructed as a sixth, composite configuration.

The ensemble averages the class probabilities of the $M = 5$ tuned classifiers with equal weights and selects the class with the highest mean probability [37]:

$$\hat{y} = \arg \max_c \frac{1}{M} \sum_{m=1}^M P_m(c | \mathbf{x}) \quad (5)$$

where $P_m(c | \mathbf{x})$ is the probability of class c given by classifier m .

2.5 Hyperparameter Optimization

Hyperparameters were optimized using RandomizedSearchCV with stratified k-fold cross-validation and macro F1-score as the scoring criterion, with SMOTE embedded inside the cross-

validation pipeline so that resampling occurred only within training folds. Table 2 reports the optimized hyperparameters obtained for each classifier.

Table 2. Optimized hyperparameters per classifier

Model	Optimized Hyperparameters
Logistic Regression	C = 5; solver = lbfgs; penalty = l2; class_weight = balanced
SVM (RBF)	C = 1; gamma = 0.1; class_weight = None
Random Forest	n_estimators = 200; max_depth = 30; min_samples_split = 2; min_samples_leaf = 4; max_features = sqrt
Decision Tree	max_depth = 10; min_samples_split = 2; min_samples_leaf = 8; criterion = entropy; class_weight = balanced
K-Nearest Neighbors	n_neighbors = 3; weights = uniform; p = 1 (Manhattan distance)

2.6 Performance Evaluation Metrics

Model performance was assessed using accuracy, macro-averaged precision, recall, and F1-score, and full 3x3 confusion matrices, computed on the identical held-out test set (n = 2,184) for every classifier and stage. Macro-averaging was prioritized for model comparison given the pronounced class imbalance [29].

2.7 Proposed Per-Class Ordinal-Consistency Evaluation

Beyond conventional metrics, this research introduces a decomposition of confusion-matrix errors with respect to the ordinal structure of Risk_Status (Low < Medium < High). Every off-diagonal confusion-matrix cell is classified as an adjacent error (High<->Medium or Medium<->Low, confusion with the immediately neighboring severity level) or a non-adjacent error (High<->Low, a two-step severity jump that skips the intermediate category). The Ordinal Consistency Rate (OCR) is defined as one minus the proportion of total misclassifications that are non-adjacent. An OCR of 1.0 indicates that, whenever a classifier errs, it never confuses the two most severity-distant categories, a property directly relevant to the asymmetric managerial cost of risk-classification errors in project governance. [33][34].

3. RESULTS AND DISCUSSION

3.1 Baseline Performance (Before SMOTE)

Table 3. Baseline classification performance before SMOTE

Model	Accuracy	Precision (macro)	Recall (macro)	F1-score (macro)
Logistic Regression	0.8668	0.8067	0.7450	0.7727
SVM (RBF)	0.8562	0.8181	0.6871	0.7384
Random Forest	0.8425	0.8265	0.6084	0.6757
Decision Tree	0.7825	0.6424	0.6546	0.6483
K-Nearest Neighbors	0.7953	0.6801	0.5745	0.6132

Table 3 shows that at baseline, aggregate accuracy ranges from 78.3% (DT) to 86.7% (LR); however, macro recall is markedly lower across all classifiers (0.57-0.75), reflecting weak sensitivity to the High and Low minority classes despite superficially high accuracy, and the well-documented accuracy-imbalance divergence that motivates the controlled SMOTE comparison.

3.2 After SMOTE Comparison

Table 4. Classification performance after SMOTE

Model	Accuracy	Precision (macro)	Recall (macro)	F1-score (macro)
Logistic Regression	0.8063	0.6865	0.8588	0.7416
SVM (RBF)	0.8237	0.7140	0.8188	0.7547
Random Forest	0.8288	0.7154	0.7364	0.7252
Decision Tree	0.7541	0.6126	0.6814	0.6402

Model	Accuracy	Precision (macro)	Recall (macro)	F1-score (macro)
K-Nearest Neighbors	0.6291	0.5582	0.7677	0.5800

Table 4 shows that SMOTE improved macro recall for LR, SVM, and RF by 11.4, 13.2, and 12.8 percentage points respectively, at a 1.4-6.1 point accuracy cost a favourable trade-off given that a false-negative High-risk classification is typically more costly than a false alarm. KNN, however, exhibited a marked accuracy collapse (79.5% → 62.9%) despite a recall gain, indicating high sensitivity to the altered neighborhood structure introduced by synthetic minority samples. DT also declined modestly on most metrics after SMOTE, consistent with single-tree models' susceptibility to overfitting on resampled data.

3.3 Hyperparameter-Tuned Models and Ensemble

Table 5. Tuned and ensemble classifier performance

Model	Accuracy	Precision (macro)	Recall (macro)	F1-score (macro)
Logistic Regression (Tuned)	0.8082	0.6891	0.8596	0.7441
SVM (Tuned)	0.8365	0.7427	0.7787	0.7587
Random Forest (Tuned)	0.8214	0.7001	0.7723	0.7308
Decision Tree (Tuned)	0.7372	0.6300	0.7621	0.6694
K-Nearest Neighbors (Tuned)	0.6964	0.5625	0.6696	0.5955
Soft-Voting Ensemble (5 models)	0.8342	0.7283	0.7970	0.7573

Table 5 shows that after tuning, SVM achieves the highest macro F1-score (0.7587), narrowly ahead of the five-model ensemble (0.7573) and Random Forest (0.7308). DT and KNN, despite tuning gains of +2.92 and +1.55 macro-F1 points respectively over their after-SMOTE default configurations, remain the two weakest classifiers (0.6694 and 0.5955). DT's single-tree structure and KNN's distance-based mechanism both appear less able to exploit the imbalanced, high-dimensional (one-hot-encoded) feature space than the linear, margin-based, and ensemble-tree alternatives.

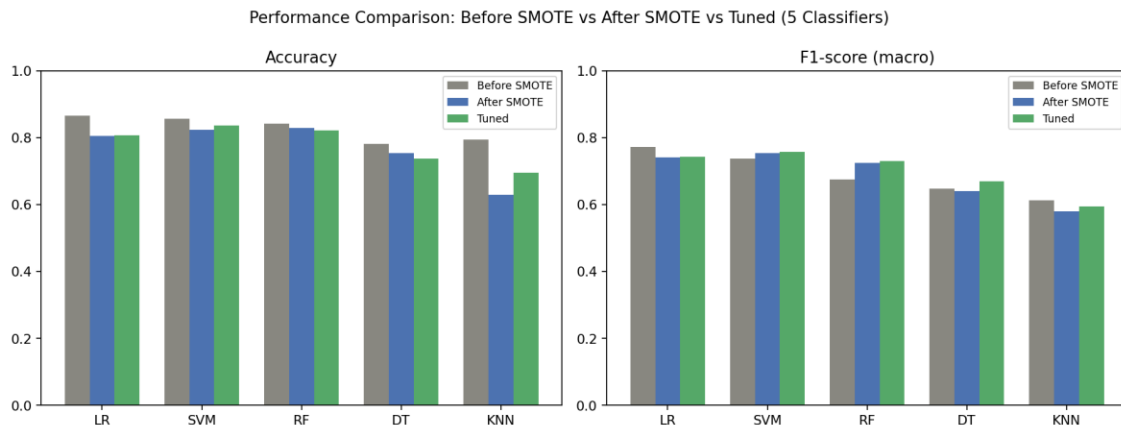


Figure 2. Performance Comparison before SMOTE, after SMOTE, and after tuning

3.4 Per-Class Performance

Table 6. Per-class precision (P), recall (R), and F1-score, tuned classifiers

Model	High: P / R / F1	Low: P / R / F1	Medium: P / R / F1
Logistic Regression	0.55 / 0.91 / 0.69	0.56 / 0.88 / 0.68	0.96 / 0.78 / 0.86
SVM	0.72 / 0.75 / 0.73	0.60 / 0.71 / 0.65	0.91 / 0.87 / 0.89
Random Forest	0.59 / 0.78 / 0.67	0.60 / 0.68 / 0.64	0.91 / 0.85 / 0.88

Model	High: P / R / F1	Low: P / R / F1	Medium: P / R / F1
Decision Tree	0.54 / 0.80 / 0.65	0.43 / 0.76 / 0.55	0.91 / 0.73 / 0.81
K-Nearest Neighbors	0.42 / 0.66 / 0.51	0.40 / 0.64 / 0.49	0.87 / 0.71 / 0.78

Table 6 shows that SVM achieves the most balanced per-class profile, with the highest Low-risk F1 (0.65) and Medium-risk F1 (0.89) among all five classifiers. LR achieves the highest High-risk recall (0.91), which matters when minimizing missed High-risk projects is the priority, at the cost of lower High-risk precision (0.55). KNN's per-class scores are uniformly the weakest, particularly for the two minority classes (High F1 = 0.51, Low F1 = 0.49), confirming its comparative disadvantage for this imbalanced, high-dimensional classification task.

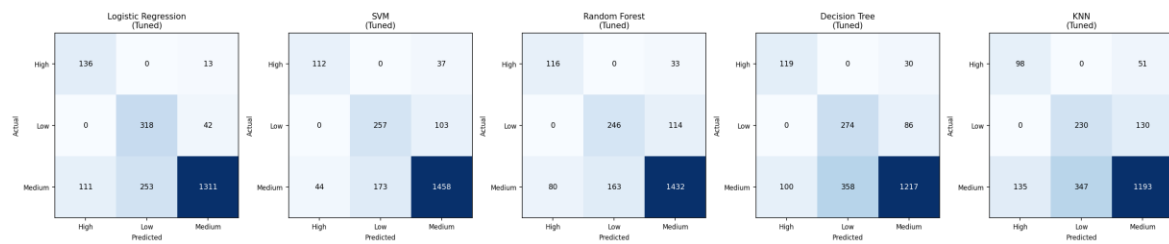


Figure 3. Confusion matrices, tuned classifiers

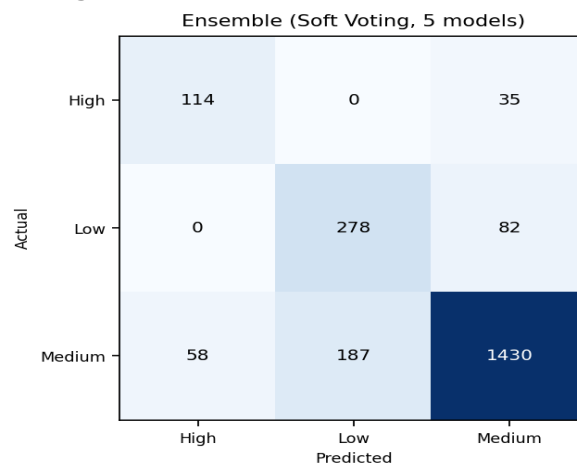


Figure 4. Confusion Matrix, Five-Model Soft-Voting Ensemble

3.5 Per-Class Ordinal-Consistency Analysis

Table 6 applies the ordinal-consistency decomposition to all sixteen model/stage configurations tested in this research: five classifiers at three stages (before SMOTE, after SMOTE, tuned), plus the soft-voting ensemble, which encompasses 7,165 aggregate misclassifications on the held-out test set.

Table 7. Per-class ordinal-consistency decomposition across all

Stage	Model	Total Errors	Adjacent Errors	Non-Adjacent Errors	OCR
Before SMOTE	Logistic Regression	291	291	0	1.0000
Before SMOTE	SVM	314	314	0	1.0000
Before SMOTE	Random Forest	344	344	0	1.0000
Before SMOTE	Decision Tree	475	475	0	1.0000
Before SMOTE	KNN	447	447	0	1.0000
After SMOTE	Logistic Regression	423	423	0	1.0000

Stage	Model	Total Errors	Adjacent Errors	Non-Adjacent Errors	OCR
After SMOTE	SVM	385	385	0	1.0000
After SMOTE	Random Forest	374	374	0	1.0000
After SMOTE	Decision Tree	537	537	0	1.0000
After SMOTE	KNN	810	810	0	1.0000
Tuned	Logistic Regression	419	419	0	1.0000
Tuned	SVM	357	357	0	1.0000
Tuned	Random Forest	390	390	0	1.0000
Tuned	Decision Tree	574	574	0	1.0000
Tuned	KNN	663	663	0	1.0000
Ensemble	Soft Voting (5 models)	362	362	0	1.0000

Table 7 shows that all sixteen configurations achieved a perfect Ordinal Consistency Rate of 1.0000. Across 7,165 aggregate misclassifications spanning five algorithmic paradigms (linear, margin-based, ensemble-tree, single-tree, and instance-based) and every experimental stage, no instance of a High-risk project being predicted as Low-risk, or vice versa, was observed. This holds even for the two weakest classifiers, DT and KNN, despite their substantially higher total error counts (up to 810 for KNN after SMOTE), indicating that classification weakness in this framework manifests as confusion with the adjacent Medium-risk category, never as a severity-jump error.

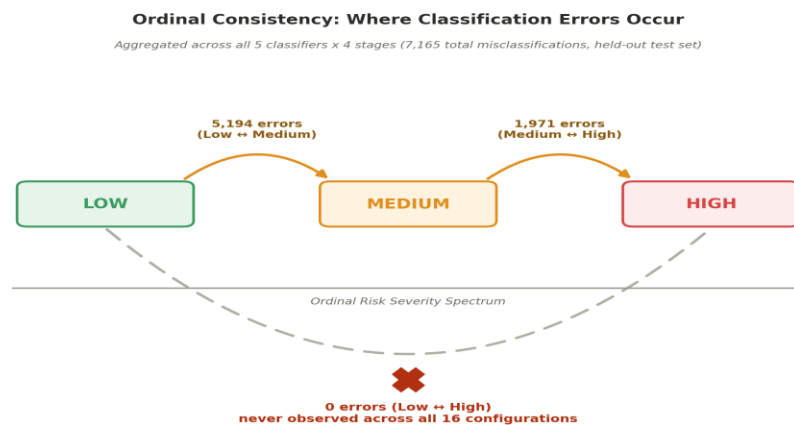


Figure 5. Conceptualizing the Ordinal Consistency

Figure 5 shows that all 7,165 aggregate misclassifications occurred between adjacent risk categories: 5,194 errors between Low and Medium and 1,971 errors between Medium and High, while no direct Low–High misclassifications were observed across all 16 model configurations. This indicates that classification errors generally followed the ordinal structure of risk levels, with predictions shifting toward neighboring categories rather than jumping between extreme risk levels.

3.6 Interpretation of the Ordinal-Consistency Finding

A plausible explanation lies in the underlying data-generating structure: because Risk_Status was found to be threshold-derived from a latent, continuous risk signal, the retained predictor features which correlate with that same underlying signal even after the leaked variable's removal appear to preserve enough of its ordinal structure that classifiers rarely, if ever, place a project at the opposite extreme of the risk spectrum. This interpretation is the most parsimonious explanation consistent with the leakage-audit findings, rather than evidence of an intrinsic ordinal-awareness capability in the classifiers themselves, none of which was trained with an ordinal-specific loss function. Irrespective of mechanism, the practical implication is directly relevant to the framework's intended use case: a

decision-support tool for project risk triage that never mistakes a High-risk project for a Low-risk one (or the reverse) even when using a comparatively weak classifier such as KNN offers a categorically different reliability profile than aggregate accuracy alone would suggest. This distinction is recoverable only through the per-class, ordinal-aware decomposition proposed here [34].

3.7 Comparison with Related Work

Table 8. Comparison with representative related work

Research	Algorithm(s)	Risk Classes	Best Acc.	Best F1
[18], 2025	ANN Framework	IT project risk	0.65	0.67
[11], 2024	GBM + t-SNE	Multi-level risk	0.85	0.80
[14], 2021	LR, RF, DT, SVM	Risk percentage	0.89	0.87
[16], 2024	ML ensemble	Investment risk levels	0.87	0.85
[22], 2026	Hybrid ML	Risk & resource	0.92	0.91
This research (SVM, tuned)	LR, SVM, RF, DT, KNN	Low/Medium/High Risk	0.8365	0.7587
This research (Ensemble)	5-model soft voting	Low/Medium/High Risk	0.8342	0.7573

Table 8 shows that the leakage-audited results presented here (accuracy 0.78-0.86 at baseline, macro F1-score up to 0.759 after tuning) sit within the range reported by comparable, methodologically transparent studies in this literature [15], [16], [17]. Consider this alignment a positive validity signal: near-ceiling performance (>99%) on a task of this nature would be implausible absent target leakage, and its absence here is consistent with the explicit leakage-screening step performed.

3.8 Practical Implications

SVM and the five-model ensemble are recommended as primary classifiers for deployment, given their leading macro F1-scores and the most balanced per-class profiles. LR remains the preferred classifier where interpretability is a governance requirement, given its coefficient-level transparency and the highest observed High-risk recall. DT, despite weaker aggregate performance, retains practical value where rule-based transparency is prioritized over raw accuracy. The Ordinal Consistency Rate finding supports a specific operational recommendation: because no classifier in this framework, including the weakest, KNN, produced a severity-jump error, a two-stage triage protocol (automated classification followed by mandatory human review only for projects flagged near the High-Medium or Medium-Low boundary) can be implemented with confidence that the automated stage will not silently misroute a High-risk project into the Low-risk queue.

3.7 Limitations and Future Directions

The dataset, while large (10,919 records) and reflective of realistic class-imbalance proportions, has characteristics consistent with rule-based or threshold-derived labeling; validation on empirically labeled, organization-specific project datasets is necessary to establish distributional generalizability. The SVM and Random Forest hyperparameter search spaces were computationally constrained; a broader search, particularly for SVM, may further improve tuned performance. The ordinal-consistency finding is best interpreted as a property of this dataset's latent structure rather than a guaranteed property of these classifiers in general; replication on independent, non-threshold-derived project-risk datasets is needed to establish external validity. This framework was not compared against models trained with explicit ordinal-aware loss functions (e.g., cumulative-link or rank-consistent architectures); such a comparison would clarify whether the observed ordinal consistency reflects dataset structure, algorithmic behavior, or both. Future work includes SHAP-based explainability for per-project risk explanations, ensemble stacking with a learned meta-learner, real-time integration with project management information systems, and cross-organizational validation studies [32][14] [33][34].

4. CONCLUSION

This research presented a multi-classifier machine learning framework for automated IT project risk level identification across 10,919 IT project records, classified into Low, Medium, and High Risk

categories using five algorithms (LR, SVM, RF, DT, KNN) under a systematic, auditable preprocessing protocol. A target-leakage screening step identified and removed a variable that deterministically encoded the risk label, preventing spurious near-perfect accuracy that arises when a feature is built from the same components used to define the target class. After a controlled before/after SMOTE comparison and hyperparameter optimization, the tuned classifiers achieved the following test-set accuracy (macro F1-score): SVM 83.65% (0.759), RF 82.14% (0.731), LR 80.82% (0.744), DT 73.72% (0.669), and KNN 69.64% (0.596), while the five-model soft-voting ensemble achieved 83.42% (0.757). SVM therefore gave the best single-model result, closely followed by the ensemble, whereas DT and KNN consistently underperformed the other classifiers despite computational efficiency. Before SMOTE, some models achieved higher accuracy (up to 86.68% for LR), but macro recall was markedly lower (0.57-0.75), confirming that raw accuracy overstates performance on the minority High and Low classes.

The central contribution of this research is a per-class ordinal-consistency evaluation, introduced to decompose classification errors according to the ordinal structure of the risk categories. Across all sixteen model configurations tested and 7,165 aggregate misclassifications, every configuration achieved a perfect Ordinal Consistency Rate of 1.0000: no classifier, at any stage, ever confused a High-risk project with a Low-risk project. This finding, undetectable through aggregate accuracy or macro F1-score alone, demonstrates the practical value of systematic, leakage-audited preprocessing combined with per-class, structure-aware evaluation for ordinally organized IT project risk classification, and is proposed as a reporting standard for future work in this domain.

ACKNOWLEDGEMENTS

The author would like to thank Universitas Ahmad Dahlan and Universitas Putera Batam for their support for this research. Without their help and strong commitment, this research would not have been successfully implemented. In addition, the author would like to thank all parties who contributed, directly or indirectly, to the research process.

REFERENCES

- [1] J. Menezes, C. Gusmão, and H. Moura, "Risk factors in software development projects: a systematic literature review," *Software Quality Journal*, vol. 27, no. 3, pp. 1149–1174, 2019, doi: 10.1007/S11219-018-9427-5.
- [2] M. N. Alatawi *et al.*, "A Data-Driven Artificial Neural Network Approach to Software Project Risk Assessment," *IET Software*, vol. 2023, no. 1, Jan. 2023, doi: 10.1049/2023/4324783.
- [3] V. Nikolaenko and A. Sidorov, "Analysis of 105 IT Project Risks," *Journal of Risk and Financial Management*, vol. 16, no. 1, p. 33, Jan. 2023, doi: 10.3390/jrfm16010033.
- [4] J. Masso, J. Masso, F. J. Pino, C. Pardo, F. García, and M. Piattini, "Risk management in the software life cycle: A systematic literature review," *Comput. Stand. Interfaces*, vol. 71, p. 103431, 2022, doi: 10.1016/J.CSI.2020.103431.
- [5] K. Buganová and J. Šimíčková, "Risk management in traditional and agile project management," *Transportation Research Procedia*, vol. 40, pp. 986–993, 2019, doi: 10.1016/J.TRPRO.2019.07.138.
- [6] R. Al Mnaseer, "Ensemble machine learning for risk assessment in construction project management: applications in infrastructure development," *Asian Journal of Civil Engineering*, vol. 27, no. 2, pp. 485–497, Feb. 2026, doi: 10.1007/s42107-025-01515-4.
- [7] A. Cucus, W. P. Sari, A. Aminuddin, D. I. S. Saputra, W. A. Hammood, and R. F. A. Aziza, "Systematic Review of IT Risk Management Using a Scientometric Approach," *Systematic Review of IT Risk Management Using a Scientometric Approach*, pp. 126–131, 2023, doi: 10.1109/icoiact59844.2023.10455938.
- [8] S. Di Zio, M. Bolzan, M. Marozzi, and M. Scioni, "Delphi-based scenarios and risk management: A parallelism between paths destined to meet," *Socioecon. Plann. Sci.*, vol. 92, p. 101832, Apr. 2024, doi: 10.1016/j.seps.2024.101832.
- [9] M. Khalilzadeh, S. A. Banihashemi, and D. Božanić, "A Step-By-Step Hybrid Approach to Project Risk Management," *Decision Making: Applications in Management and Engineering*, vol. 7, no. 1, pp. 442–472, Jan. 2024, doi: 10.31181/dmame712024884.
- [10] B. G. Tavares, C. E. S. da Silva, and A. D. de Souza, "Risk management analysis in Scrum software projects," *International Transactions in Operational Research*, vol. 26, no. 5, pp. 1884–1905, Sep. 2019, doi: 10.1111/itor.12401.
- [11] Q. Hammouri *et al.*, "Influence of information technology on project risk management: The mediating role of risk identification," *Journal of Project Management*, vol. 10, no. 1, pp. 143–150, 2025, doi: 10.5267/jjpm.2024.10.001.
- [12] M. Z. Nabeel, "AI-Enhanced Project Management Systems for Optimizing Resource Allocation and Risk Mitigation," *Asian Journal of Multidisciplinary Research & Review*, vol. 5, no. 5, pp. 53–91, 2024, doi: 10.55662/ajmrr.2024.5502.

- [13] A. Valencia-Arias *et al.*, "Machine learning applications in risk management: Trends and research agenda," *F1000Res.*, vol. 14, p. 233, Apr. 2025, doi: 10.12688/f1000research.161993.2.
- [14] M. Banda, E. K. Ngassam, and E. Mnkandla, "Enhancing Classification and Prediction through the Application of Hybrid Machine Learning Models," in *2024 IST-Africa Conference (IST-Africa)*, IEEE, May 2024, pp. 1–12. doi: 10.23919/IST-Africa63983.2024.10569590.
- [15] G. P. and S. Sankaranarayanan, "Prediction of Risk Percentage in Software Projects by Training Machine Learning Classifiers," *Computers & Electrical Engineering*, vol. 94, p. 107362, Sep. 2021, doi: 10.1016/j.compeleceng.2021.107362.
- [16] S. R. Bauskar, C. R. Madhavaram, E. P. Galla, J. R. Sunkara, H. K. Gollangi, and S. K. Rajaram, "Predictive Analytics for Project Risk Management Using Machine Learning," *Journal of Data Analysis and Information Processing*, vol. 12, pp. 566–580, 2024, doi: 10.4236/jdaip.2024.124030.
- [17] Z. Diao, "Research on investment project risk prediction and management based on machine learning," *Applied and Computational Engineering*, vol. 87, no. 1, pp. 142–147, Jul. 2024, doi: 10.54254/2755-2721/87/20241561.
- [18] A. N. Ismawan and S. Wahyu, "Hybrid Machine Learning Predictive Model for Resource Allocation Optimization and Project Risk Management," *Sinkron*, vol. 10, no. 1, pp. 713–724, Jan. 2026, doi: 10.33395/sinkron.v10i1.15773.
- [19] R. Yang, L. Shang, and H. Zhang, "Risk Factor Identification of the Information Technology Project based on the DEMATEL-ISM Model," *Journal of Engineering Science and Technology Review*, vol. 15, no. 4, pp. 53–59, 2022, doi: 10.25103/jestr.154.08.
- [20] A. Khodabakhshian, T. Puolitaival, and L. Kestle, "Deterministic and Probabilistic Risk Management Approaches in Construction Projects: A Systematic Literature Review and Comparative Analysis," *Buildings*, vol. 13, no. 5, p. 1312, May 2023, doi: 10.3390/buildings13051312.
- [21] I. Hadi, A. Ghozali, and A. R. Handoko, "Exploration of the Project Risk Management Framework for Information Technology Companies," *Cogito Smart Journal*, vol. 9, no. 2, pp. 266–279, 2023.
- [22] V. Wijaya and N. Rachmat, "Comparison of SVM, Random Forest, and Logistic Regression Performance in Student Mental Health Screening," *JEECS*, vol. 9, no. 2, pp. 173–184, Dec. 2024, doi: 10.54732/jeecs.v9i2.9.
- [23] Z. M. Yaseen, Z. H. Ali, S. Q. Salih, and N. Al-Ansari, "Prediction of Risk Delay in Construction Projects Using a Hybrid Artificial Intelligence Model," *Sustainability*, vol. 12, no. 4, p. 1514, Feb. 2020, doi: 10.3390/su12041514.
- [24] L. K. Aggabou, B. Lakehal, and M. Mouda, "An Artificial Neural Network Approach for Construction Project Risk Management," *International Journal of Safety and Security Engineering*, vol. 14, no. 2, pp. 553–561, Apr. 2024, doi: 10.18280/ijssse.140222.
- [25] A. Iftikhar, M. Alam, R. Ahmed, S. Musa, and M. M. Su'ud, "Risk Prediction Using Artificial Neural Network in Global Software Development," *Comput. Intell. Neurosci.*, vol. 2021, Jan. 2021, doi: 10.1155/2021/2922728.
- [26] A. S. Filippetto, R. Lima, and J. L. V. Barbosa, "A risk prediction model for software project management based on similarity analysis of context histories," *Inf. Softw. Technol.*, vol. 131, p. 106497, Mar. 2021, doi: 10.1016/j.infsof.2020.106497.
- [27] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 4, pp. 1–21, 2012, doi: 10.1145/2382577.2382579.
- [28] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002, doi: 10.1613/jair.953.
- [29] R. van den Goorbergh, M. van Smeden, D. Timmerman, and B. Van Calster, "The harm of class imbalance corrections for risk prediction models: illustration and simulation using logistic regression," *Journal of the American Medical Informatics Association*, vol. 29, no. 9, pp. 1525–1534, Aug. 2022, doi: 10.1093/jamia/ocac093.
- [30] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967, doi: 10.1109/TIT.1967.1053964.
- [31] A. Yudhana, S. Sunardi, and A. J. S. Hartanta, "Application of the K-Nearest Neighbor Algorithm Using Euclidean Distance for Predicting Sengon Wood Sawing Outcomes," *Transmisi*, vol. 22, no. 4, pp. 123–129, Nov. 2020, doi: 10.14710/transmisi.22.4.123-129.
- [32] L. I. Kuncheva, "Combining Pattern Classifiers: Methods and Algorithms," Wiley, 2014, doi: 10.1002/9781118914564.
- [33] S. Srivastava and D. Mishra, "Severity of error in hierarchical datasets," *Scientific Reports*, vol. 13, no. 1, p. 21903, 2023, doi: 10.1038/s41598-023-49185-z.
- [34] S. Zhu, K. Wang, and C. Li, "Crash Injury Severity Prediction Using an Ordinal Classification Machine Learning Approach," *International Journal of Environmental Research and Public Health*, vol. 18, no. 21, p. 11564, 2021, doi: 10.3390/ijerph182111564.