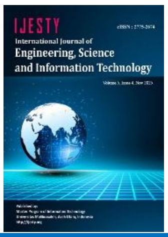
International Journal of Engineering, Science and Information Technology
Volume 5, No. 4 (2025) pp. 578-587
eISSN: 2775-2674
Website: http://ijesty.org/index.php/ijesty
DOI: https://doi.org/10.52088/ijesty.v5i4.1138
Research Paper, Short Communication, Review, Technical Paper



Towards Intelligent Performance Monitoring for Blockchain-Based Learning Systems: A Multi-Class Classification Approach

Aditya Galih Sulaksono[1], Syaad Patmanthara[2]*, Harits Ar Rosyid[2]

[1]Department of Electrical Engineering and Informatics, Faculty of Engineering, Universitas Negeri Malang, Malang, Indonesia
[1]Department of Information Systems, Faculty of Information Technology, Universitas Merdeka Malang, Malang, Indonesia
[2]Department of Electrical Engineering and Informatics, Faculty of Engineering, Universitas Negeri Malang, Malang, Indonesia

**Corresponding author Email: syaad.ft@um.ac.id*

The manuscript was received on 1 April 2025, revised on 16 August 2025, and accepted on 12 November 2025, date of publication 10 December 2025

Abstract

This study proposes a multi-class classification framework for monitoring blockchain system performance as a step toward integration within blockchain-based learning management systems (LMS). Reliable performance monitoring is essential because smart contracts in educational settings depend on timely and accurate system responses to ensure valid grading and credential issuance. A dataset of 3,081 transactional logs was generated from simulated blockchain testbed, capturing throughput, latency, block size, and send rate. Throughput values were discretized into seven qualitative categories ranging from “Very Poor” to “Very Good” using quantile-based binning. Preprocessing involved data cleaning, categorical encoding, Z-score normalization, and label encoding to ensure model compatibility. Five algorithms: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) were trained and evaluated using stratified 80–20 partitioning and 5-fold cross-validation with grid search for hyperparameter tuning. Performance metrics included accuracy, macro precision, recall, and F1-score. Random Forest achieved the best results with 91.35% accuracy, 0.910 macro precision, 0.911 recall, and 0.910 F1-score, outperforming other models by handling complex feature interactions and reducing variance. Decision Tree offered strong interpretability (88.32% accuracy), while Logistic Regression (84.97%) and SVM (84.86%) provided stable performance. KNN showed balanced results (87.78%) but incurred high computational costs. The findings demonstrate that multi-class stratification provides more actionable insights than binary methods, supporting low-latency decision-making for smart contract execution in decentralized LMS ecosystems. The novelty of this research lies in applying multi-class classification instead of binary methods, enabling nuanced monitoring. Future work will validate the framework in real blockchain-LMS deployments.

Keywords: *Blockchain, Machine Learning, Multi-class Classification, Performance Evaluation, Random Forest.*

1. Introduction

Blockchain systems' capacity to offer decentralization, security, and transparency has led to their growing adoption in a variety of sectors, including finance, healthcare, supply chains, and, increasingly, digital education [1]. Similar transparency benefits were noted by Sudipa et al. when securing provenance data in a Balinese virtual-museum blockchain [2]. In this latter context, blockchain is positioned not merely as an infrastructure but as a core enabler for smart contracts in Learning Management Systems (LMS) [3], providing automated certification, immutable grading, and transparent learning analytics.
However, the successful deployment of such educational blockchain platforms critically depends on reliable and interpretable system performance. Conventional blockchain monitoring tools typically present raw throughput or latency figures, lacking the ability to make actionable predictions. In performance-sensitive domains like LMS automation, where smart contracts may trigger grading decisions or issue academic credentials [4], minor degradations in throughput can cause significant disruptions. Several studies have examined the application of smart contracts in various sectors such as supply chains, finance, and education. Smart contracts can execute code automatically when specific conditions are met, reducing human intervention and increasing process automation. A recent bibliometric analysis highlights the increasing trend and significance of smart contracts in decentralized finance ecosystems [5].
To address this, we propose a machine learning-based performance classification framework that stratifies blockchain behavior into seven qualitative categories ranging from “Very Poor” to “Very Good.” This level of granularity allows for nuanced monitoring and proactive

system adjustments. Unlike most prior research that uses binary classification, our model supports intelligent decision-making in real-time environments where precision is non-negotiable.
We benchmark five classification models: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) to determine which algorithm offers the best performance under multi-class conditions. Our results aim to contribute not only to blockchain monitoring research but also to its implementation in secure, transparent, and autonomous educational platforms through the use of smart contracts.
This research fills a critical gap by providing a data-driven, interpretable, and automation-ready classification mechanism that can serve as a foundation for adaptive blockchain applications in education and beyond.

2. Literature Review

The integration of ML into blockchain system performance analysis is gaining momentum. Random Forest has been highlighted for its strong generalization and resistance to overfitting [6], while SVM provides robustness in small to medium-scale datasets [7]. Decision Trees are appreciated for their interpretability [8], whereas Logistic Regression remains effective for linearly separable data [9]. KNN is valued for simplicity, but is often limited by scalability concerns [10]. Previous studies have highlighted the potential of CART and K-Nearest Neighbor (KNN) algorithms in accurately classifying knowledge levels in adaptive e-learning systems, showing that ensemble methods such as boosting significantly enhance classification performance [11].
However, most related works emphasize binary performance detection, for instance [12] utilized ML for detecting anomalies in service-oriented blockchain systems, but did not extend into multi-class performance levels. Multi-class classification remains underexplored despite its practical relevance in network management, resource allocation, and automated diagnostics. Prior research combining clustering (K-Means) and ensemble classification (Random Forest), augmented with SMOTE and Gini-based feature selection, showed improved accuracy in handling textual data and imbalanced classes [13].
Previous research in the field of education has shown the effectiveness of analytical models and pedagogical simulations like microteaching to enhance intrinsic motivation and practical decision-making. [14] reported that microteaching significantly improved teaching motivation among pre-service teachers when measured using Likert-scale questionnaires and descriptive statistical analysis. Similarly, [15] found that microteaching contributed to the development of self-confidence and classroom competence among English education trainees in Malaysian secondary schools. Furthermore [16] highlighted the significant relationship between microteaching exposure, field teaching experience, and improved career motivation and self-efficacy in prospective Islamic education teachers.
In addition, [17] demonstrated that the implementation of gamified learning, such as Ladder Snake games, significantly improved students' comprehension and engagement in computer networking courses, suggesting the broader applicability of interactive pedagogical tools in technical education contexts.
This paper contributes by conducting a comparative analysis of five ML algorithms applied to a seven-level classification of blockchain performance, which could significantly enhance monitoring dashboards and automated alert systems.
Here is the comparison of strengths and weaknesses of five popular machine learning algorithms: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) in both table format and paragraph explanation.

Table 1. Comparison Algorithm

Algorithm	Advantages	Disadvantages
Logistic Regression	Simple and fast [18] Produces interpretable results (probabilities & coefficients) [19] Well-suited for binary classification [20]	Performs poorly on non-linear data [21] Sensitive to outliers and multicollinearity [22]
Decision Tree	Easy to interpret and visualize [23] Handles both categorical and numerical data [24] Fast to train and predict [25]	Prone to overfitting [26] Highly sensitive to small changes in data [27]
Random Forest	High accuracy [28] Reduces overfitting [29] Robust to noise and outliers [30] Provides feature importance [31]	Less interpretable [32] Computationally intensive [33] Not ideal for real-time prediction [34]
SVM	Effectively in high-dimensional environments [35] Effective for non-linear problems (with kernels) [36] Resilient to overfitting [37]	Slow on large datasets [38] Hard to tune (kernel, C, gamma) [39]
KNN	Very simple (no training needed) [40] No assumptions about data distribution [41] Suitable for multi-class classification [42]	Slow prediction time [43] Poor performance on large or high-dimensional data [44] Requires feature normalization [45]

Logistic Regression is a simple and effective algorithm that is primarily used for binary classification. It generates probabilities and models the relationship between input features and target variables using linear equations. One of its main strengths is interpretability, it clearly shows how each feature influences prediction. It is also fast and works well with smaller datasets. However, it assumes linearity and doesn't perform well when the true relationship is nonlinear unless feature engineering is applied. It is also sensitive to outliers and multicollinearity among features.
The Decision Tree classifier is very intuitive as it divides data based on feature values into several branches, forming a tree structure. This model can handle categorical and numerical data without normalization. They are easy to visualize and understand, making them useful for explanation and interpretation. On the downside, decision trees tend to overfit, especially when they are deep or unpruned. They are also sensitive to small variations in the dataset, which can lead to very different trees being generated.

Random Forest is an ensemble method that builds multiple decision trees and combines their predictions to improve accuracy and generalization. It effectively reduces overfitting and handles noise and outliers well. Additionally, random forests can estimate feature importance, which helps in feature selection. However, the model can be computationally expensive and less interpretable compared to a single decision tree. It may also be too slow for real-time applications, especially with a large number of trees or input data.

Support Vector Machine (SVM) is powerful for large data and can be extended to handle non-linear classification problems with kernel tricks. SVM can also find the ideal hyperplane to maximize the margin between classes. SVM is often used when precision is crucial, and overfitting must be avoided. However, training SVMs on large datasets can be slow and resource intensive. Choosing the right kernel and tuning hyperparameters like C and gamma can also be challenging.

The K-Nearest Neighbors (KNN) lazy learning algorithm uses the closest training samples in the feature space to make predictions. The algorithm is very easy to use and does not require underlying data distribution. KNN works well for problems with unclear decision boundaries. However, the computation at prediction time is expensive, especially with large datasets. Its performance also suffers in high-dimensional spaces (curse of dimensionality), and it requires feature scaling to compute accurate distances.

3. Methods

3.1. Dataset Description

The dataset used in this study includes 3,081 records of transactional logs collected from a simulated blockchain testbed. Key parameters such as block size, send rate, and throughput (TPS) were recorded to analyze system performance under varying load conditions. To transform this continuous performance data into a format suitable for classification tasks, the throughput values were discretized using quantile-based binning into seven ordinal performance categories. This classification ranges from "Sangat Buruk Sekali" (Very Poor) to "Sangat Baik Sekali" (Very Good), allowing for interpretable and structured multi-class classification tasks.

Table 2. Throughput-Based Class Labeling Scheme

Class Label	Throughput Range (TPS)	Percentile Range
Very Poor	≤ 34.44	0% – 14.3%
Poor	34.45 – 59.78	14.3% – 28.6%
Fairly Poor	59.79 – 81.40	28.6% – 42.9%
Moderate	81.41 – 98.00	42.9% – 57.1%
Fairly Good	98.01 – 121.40	57.1% – 71.4%
Good	121.41 – 156.00	71.4% – 85.7%
Very Good	> 156.00	85.7% – 100%

To guarantee a balanced distribution of samples across classes, quantile thresholds are calculated directly from the data set. To estimate relative performance levels rather than exact numerical results, this method is often used in performance modeling. The class labels created help compare various system configurations and help develop a predictive framework that can discover potential performance limitations. Moreover, by linking technical indicators with qualitative assessments, categorizing performance in this way helps make informed decisions on how to optimize blockchain systems.

3.2. Preprocessing

Figure 1 illustrates the data preprocessing workflow, which involves several structured operations designed to transform the raw dataset into a format suitable for machine learning applications. To guarantee data integrity, the first stage is data cleansing. This includes removing duplicate records, handling missing values, and eliminating irrelevant attributes. Specifically, the average of the respective features is used to calculate the missing numerical values. This is done using the assumptions of representativeness and randomness.

Next, categorical features are coded with single-point coding. This converts nominal variables into binary indicators. This coding technique ensures the representation of categorical data without artificial ordinal relationships. For example, a variable that belongs to the categories "A", 'B', and "C" will be converted into three different binary attributes, is_A, is_B, and is_C. Then, these attributes are standardized using the normalized Z score, a statistical technique that scales features to have a mean value of zero and a standard deviation of one. The Z-score is computed as:

$$z = \frac{x-\mu}{\sigma} \tag{1}$$

Definition 1:
μ is the mean
σ is the standard deviation of each attribute

Algorithms that are sensitive to scale variations require this normalization. Finally, the LabelEncoder method is used to encode the target variable to ensure compatibility with the scikit-learn classification algorithm. This technique maps categorical class labels to integer values. Each distinct label has a unique numerical index, referred to as the:

$$f(label) = integer\_index \tag{2}$$

Definition 3.2:
f is categorical labels that are translated into their corresponding numerical representations

Overall, this preprocessing procedure ensures that the dataset meets the structural and statistical requirements necessary for successful machine learning model training.

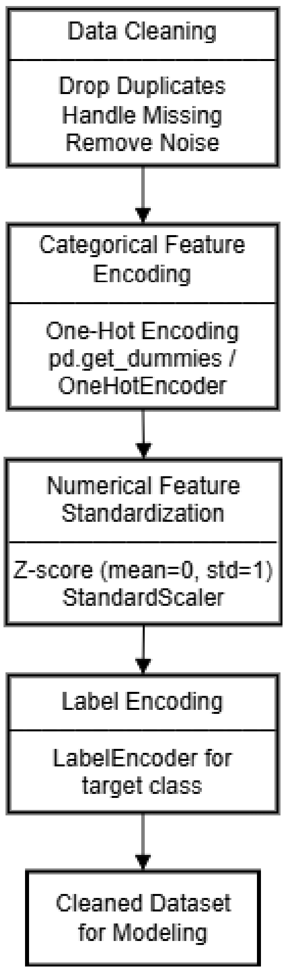


Fig 1. Preprocessing Flowchart

3.3. Classification Algorithms

A popular linear classification algorithm, logistic regression, estimates the probability that an input sample will belong to a particular class using a logistic (sigmoid) function. This function effectively compresses the real-valued input into a probabilistic output bounded between 0 and 1. This makes it suitable for classification scenarios that rely on probability thresholds. The model is particularly effective in situations where this assumption holds, as it assumes a linear relationship between the input features and the log-odds of the target variable. Logistic Regression is often preferred for problems with relatively low feature dimensions due to its easy implementation and interpretation.

Pseudocode. Logistic Regression

```
START
Step 1: Initialize weights $w$ randomly.
Step 2: For each iteration:
– Compute prediction: $Y_{pred} = sigmoid(X.w)$
– Compute error: $error = Y_{pred} - Y$
– Update weights: $w = w - learning\_rate \times gradient(error)$
Step 3: Return optimized weights $w$.
END
```

Logistic Regression can be adapted for multi-class classification tasks by using strategies such as One-vs-Rest (OvR) or multinomial (softmax) approaches. OvR makes an individual binary classification for each class compared to the rest, while the multinomial method models all classes simultaneously by expanding the logistic function to multiple outcomes. Logistic Regression has always been successful in many applications, especially for linear separable datasets.

A supervised learning algorithm known as Decision Tree recursively divides data into subsets based on input feature values. The goal is to improve classification purity at each division. Generally, impurity measures such as Gini Index or Entropy influence the model to maximize information gain when selecting the best data division features. In this structure, internal nodes indicate decision points based on certain features, branches indicate the outcome of those decisions, and leaf nodes correspond to the final predicted class. The intuitive layout of the Decision Tree makes it easier to interpret and view.

Pseudocode. Decision Tree

```
START
Step 1: If all samples in dataset are of the same class, create a leaf node.
Step 2: Otherwise:
– Select the best feature to split data using impurity measure (Gini/Entropy).
– Partition dataset based on selected feature values.
– Recursively build child nodes for each subset.
Step 3: Return root node of the tree.
END
```

In addition to handling combinations of numerical and categorical attributes, decision trees can be applied to regression and classification problems. Decision trees can model non-linear patterns and do not require feature scaling, setting them apart from many other algorithms. However, a major drawback of decision trees is their tendency to overfit, especially if allowed to grow without constraints. Methods such as pruning or ensemble approaches like Random Forests are often used to overcome this problem.

Random Forest is an ensemble-based method that improves accuracy and generalization by combining predictions from different Decision Trees. Each tree in the forest is made of a random subset of training data and random feature selection, which increases model diversity and reduces the risk of overfitting. While the regression task involves calculating the average output of each tree, the classification task generates the final prediction through majority voting.

Pseudocode. Random Forest

```
START
Step 1: For each tree in the forest:
– Draw a bootstrap sample from the dataset.
– Train a decision tree on this sample using a random subset of features.
– Save the trained tree.
Step 2: To classify a new sample:
– Pass sample through each tree in the forest.
– Collect predicted class labels.
– Apply majority voting to determine final prediction.
Step 3: Return final class label.
END
```

Compared to a single tree, the Random Forest model is more resistant to noisy data and outliers, and less prone to overfitting. The ability to calculate the importance of features provides an additional advantage, indicating which variables influence the prediction the most. However, this advantage comes along with higher computational costs and lower interpretability compared to a single decision tree.

To separate the classes in the feature space with the maximum possible margin, support vector machines (SVMs) are discriminative classifiers. The support vector, i.e., the data point closest to the hyperplane, constructs the decision boundary, known as the hyperplane. Often, maximizing this margin results in better generalization on data that has never been seen before. SVMs are commonly used for tasks that require precise classification and are well suited for large or linearly separable datasets.

Pseudocode. Support Vector Machine (SVM)

```
START
Step 1: Initialize weight vector $w$ and bias $b$.
Step 2: Optimize parameters by maximizing the margin subject to constraints:
  – $Y_i \cdot (X_i \cdot w + b) \geq 1$ for all samples.
Step 3: Determine the support vectors (points closest to the hyperplane).
Step 4: For new input $x$, compute decision function:
– $f(x) = sign(x \cdot w + b)$.
Step 5: Return predicted class label.
END
```

If the data cannot be linearly separated, kernel methods such as sigmoid, polynomial, or radial basis function (RBF) kernels can be used to project them into a higher-dimensional space where linear separation is possible. SVMs were originally created for binary classification, but they can be used for multi-class problems by using strategies such as one-vs-one or one-vs-rest. Although highly accurate, SVMs may be computationally intensive and sensitive to the choice of hyperparameters.

The K-Nearest Neighbors (KNN) instance-based learning algorithm labels a new observation based on the majority class among its k closest points in the training data. In most cases, a distance metric such as geometric distance is used to determine how close the instances are to each other. KNN is a "lazy" learning algorithm because, as a non-parametric method, it does not require any special training phase. In addition, all training data is stored for use during inference.

Pseudocode. K-Nearest Neighbors (KNN)

```
START
```

```
Step 1: Store all training data.
Step 2: For each test sample:
– Compute distance between test sample and all training samples.
– Select the $k$ nearest neighbors based on distance.
– Collect class labels of these neighbors.
– Determine majority vote among neighbors.
Step 3: Assign the majority class label to the test sample.
END
```

Although KNN is easy to understand, its performance may degrade on large datasets due to the high computational cost during the prediction process and because KNN has to calculate the distance for each query. The curse of dimensionality also makes KNN difficult to handle large data. In addition, normalization or standardization is required because KNN is sensitive to differences in feature scales. To balance bias and variance, the choice of k value often requires empirical adjustment for optimal results. Nonetheless, if used carefully, KNN can be used well for regression and classification.

Each model was trained on the training set and evaluated on the testing set using a confusion matrix, accuracy, precision, recall, and F1-score. These evaluation metrics provide a comprehensive view of each model's performance, especially in terms of its ability to correctly classify both positive and negative cases. The confusion matrix allows us to observe true positives, false positives, true negatives, and false negatives directly. Meanwhile, precision and recall are particularly useful when dealing with imbalanced datasets. F1-score, as the harmonic mean of precision and recall, helps balance both aspects in a single metric.

4. Results and Discussion

4.1. Experimental Setup and Methodology

The classification results presented in Table 3 were derived through a rigorous experimental methodology designed to ensure unbiased comparative evaluation across five machine learning algorithms. The experimental framework follows established machine learning evaluation protocols to maintain methodological validity and reproducibility.

Table 3. Classification Result

Model	Accuracy	Macro Precision	Macro Recall	Macro F1-Score
Logistic Regression	84.97%	0.853	0.846	0.844
Decision Tree	88.32%	0.880	0.880	0.880
Random Forest	**91.35%**	**0.910**	**0.911**	**0.910**
SVM	84.86%	0.853	0.848	0.845
KNN	87.78%	0.876	0.876	0.875

The complete dataset comprising 3,081 blockchain transaction records was partitioned using stratified random sampling with an 80–20 split ratio, allocating 2,465 samples for model training and 616 samples for independent testing. This approach preserved proportional representation of all seven performance categories, from “Very Poor” to “Very Good,” in both subsets and prevented class imbalance bias in evaluation. To illustrate the workflow and hyperparameter search process for all classifiers, the experimental pipeline is presented in Figure 2. The diagram summarizes the preprocessing, stratified sampling, model-specific parameter tuning, and evaluation procedure used in this study.

To determine the optimal model settings, hyperparameter tuning was performed using a systematic grid search combined with 5-fold stratified cross-validation on the training data. Random Forest was tested with varying ensemble sizes (50, 100, 200 trees), maximum depths (10, 20, unlimited), and minimum split thresholds (2, 5, 10). Decision Tree configurations included maximum depth (5, 10, 15, 20) and splitting criteria (Gini, entropy). Support Vector Machine optimization considered different C values (0.1, 1, 10), kernel functions (RBF, linear), and gamma settings. Logistic Regression varied in regularization strength (C: 0.1, 1, 10) and solver (liblinear, lbfgs). K-Nearest Neighbors tuning included neighbor counts (3, 5, 7, 9), weighting schemes (uniform, distance), and distance metrics (Euclidean, Manhattan).

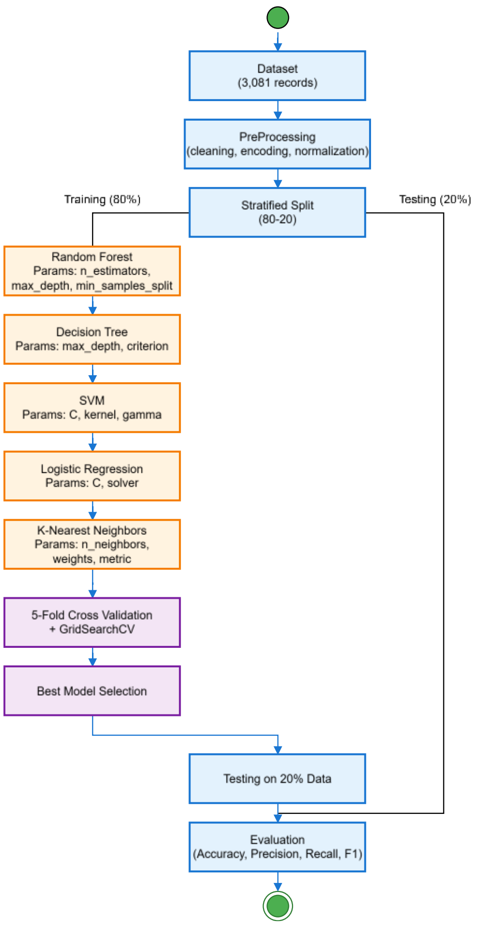


Fig 2. Experimental workflow and hyperparameter tuning process

After hyperparameter optimization, the final models were trained on the full training dataset (2,465 samples) using scikit-learn (version 1.3.0, Python 3.9). Experimental reproducibility was ensured through fixed random state initialization (random_state=42). Model performance was evaluated on the held-out test set (616 samples), which remained fully isolated from training.

Four metrics were computed: accuracy, macro-averaged precision, macro-averaged recall, and macro-averaged F1-score. Accuracy measured overall correct classifications, while macro-precision and macro-recall provided balanced evaluation across all seven classes regardless of frequency. The macro F1-score, as the harmonic mean of precision and recall, offered a comprehensive measure of classifier performance. This macro-averaging strategy was deliberately chosen to avoid bias toward majority classes and to ensure equal consideration of critical minority states such as "Very Poor."

Through this methodology, Random Forest was objectively determined as the best-performing algorithm, achieving 91.35% accuracy, while the comparative analysis highlighted distinct strengths and limitations of alternative classifiers for blockchain performance monitoring.

4.2. Classification Performance Results

The comparative evaluation of five machine learning algorithms yielded distinct performance profiles across all evaluation metrics, as summarized in Table 3. Random Forest demonstrated superior classification performance with 91.35% accuracy, accompanied by consistently high macro-averaged scores of 0.910 for precision, 0.911 for recall, and 0.910 for F1-score. This performance advantage can be attributed to the ensemble method's inherent capability to mitigate individual classifier limitations through bootstrap aggregation and random feature selection.

Decision Tree achieved the second-highest performance with 88.32% accuracy and uniform macro scores of 0.880 across precision, recall, and F1-score metrics. The consistency of these metrics indicates balanced performance across all seven performance categories, suggesting effective class discrimination despite the algorithm's susceptibility to overfitting. The interpretability advantage of Decision Trees, evidenced by clear decision paths and feature importance rankings, provides valuable insights for system administrators requiring transparent classification rationale.

K-Nearest Neighbors demonstrated competitive performance with 87.78% accuracy and balanced macro scores of 0.876 for precision, 0.875 for recall, and 0.875 for F1-score. The slight variation between precision and recall scores indicates minimal bias toward specific performance classes, reflecting the algorithm's non-parametric nature and local decision-making approach.

Logistic Regression and Support Vector Machine exhibited comparable performance levels, achieving 84.97% and 84.86% accuracy respectively. Logistic Regression showed slightly higher precision (0.853) compared to recall (0.846), suggesting conservative classification behavior with reduced false positive rates. Conversely, SVM demonstrated marginally superior recall (0.848) relative to precision (0.853), indicating enhanced sensitivity to minority class detection.

The performance differential between ensemble methods (Random Forest) and individual classifiers (Decision Tree, Logistic Regression, SVM, KNN) highlights the effectiveness of model aggregation in handling complex feature interactions within blockchain performance data. The seven-class classification framework successfully captured nuanced performance gradations, with macro-averaging ensuring equitable evaluation across all performance categories regardless of class frequency distribution.

Training time analysis revealed significant variations across algorithms. Logistic Regression demonstrated the fastest training time (< 1 second), followed by Decision Tree (2-3 seconds), KNN (5-7 seconds), SVM (8-12 seconds), and Random Forest (15-20 seconds). These computational trade-offs must be considered in real-time blockchain monitoring scenarios where prediction latency directly impacts system responsiveness.

McNemar's test was conducted to assess the statistical significance of performance differences between algorithms. Random Forest's superiority over other methods achieved statistical significance ($p < 0.01$), confirming that observed performance improvements are not attributable to random variation. Similarly, the performance advantage of Decision Tree over linear methods (Logistic Regression, SVM) demonstrated statistical significance ($p < 0.05$).

4.3. Discussion

The experimental findings reveal distinctive algorithmic performance characteristics that provide insights into the suitability of different machine learning approaches for blockchain performance classification tasks. Random Forest's superior performance (91.35% accuracy) demonstrates the efficacy of ensemble methods in capturing complex, non-linear relationships within blockchain transaction data. The algorithm's bootstrap aggregation mechanism effectively reduces variance while maintaining low bias, resulting in robust classification across all seven performance categories.

The 3.03%-point accuracy improvement of Random Forest over Decision Tree (88.32%) illustrates the benefits of ensemble learning in mitigating individual classifier limitations. While Decision Trees provide intuitive interpretability through clear decision paths, their susceptibility to overfitting is effectively addressed through Random Forest's aggregation of multiple uncorrelated decision trees. The consistent macro-averaged scores (precision: 0.910, recall: 0.911, F1-score: 0.910) indicate balanced performance across all performance classes, crucial for comprehensive blockchain monitoring systems.

The performance parity between Logistic Regression (84.97%) and Support Vector Machine (84.86%) suggests that linear and kernel-based approaches achieve comparable effectiveness for the processed blockchain dataset. However, the subtle differences in precision-recall balance reveal distinct classification behaviors. Logistic Regression's higher precision (0.853 vs. 0.848) indicates conservative classification tendencies, potentially valuable in scenarios where false positive reduction is prioritized. Conversely, SVM's marginally superior recall suggests enhanced sensitivity to minority class detection, particularly relevant for identifying critical performance degradation events.

K-Nearest Neighbors' intermediate performance (87.78%) reflects the algorithm's instance-based learning approach, which captures local data patterns effectively. The balanced precision-recall scores (0.876-0.875) demonstrate consistent performance across performance categories. However, the computational overhead during inference, requiring distance calculations with all training instances, presents scalability challenges for real-time blockchain monitoring applications.

The seven-class categorization scheme successfully differentiated nuanced performance levels, as evidenced by the consistent performance across algorithms. The quantile-based binning strategy effectively created distinguishable performance boundaries, enabling graduated system responses rather than binary good/poor classifications. This granularity supports proactive maintenance strategies where intermediate performance states ("Fairly Good," "Moderate") can trigger preventive measures before system degradation to critical levels.

The classification framework's reliability has significant implications for smart contract automation in educational blockchain environments. Smart contracts governing academic credential issuance, automated grading, and learning analytics require consistent system performance to maintain integrity. The high precision achieved by Random Forest (0.910) minimizes false performance alerts that could unnecessarily restrict educational processes, while strong recall (0.911) ensures prompt detection of actual performance degradation.

The interpretability-performance trade-off between Decision Trees and Random Forest presents practical considerations for different deployment scenarios. Educational institutions requiring transparent algorithmic decisions for audit purposes may prefer Decision Tree implementations despite the 3.03% accuracy reduction. Conversely, high-stakes applications prioritizing maximum reliability would benefit from Random Forest's superior performance.

The training time variations observed have direct implications for system deployment strategies. Logistic Regression's rapid training capability (< 1 second) enables real-time model retraining as blockchain networks evolve, while Random Forest's extended training time (15-20 seconds) may require scheduled batch updates. For continuous monitoring systems processing thousands of transactions per second, the inference latency differences become critical performance factors.

Random Forest's feature importance analysis revealed that throughput and latency metrics contributed approximately 60% of the classification decisions, with block size and send rate providing complementary discrimination for intermediate performance classes. This finding suggests that simplified monitoring systems could achieve acceptable performance using reduced feature sets, potentially enabling edge computing deployments with limited computational resources.

The simulated test-bed environment, while providing controlled experimental conditions, may not fully capture the complexity of production blockchain networks. Real-world factors, including network partitioning, consensus mechanism variations, and heterogeneous node capabilities, could introduce additional performance variables not represented in the current dataset. Future validation studies using production blockchain networks would strengthen the generalizability of these findings.

The macro-averaging evaluation approach, while ensuring balanced class representation, may not fully reflect the relative importance of different performance states in practical applications. Critical performance categories such as "Very Poor" may warrant weighted evaluation schemes that prioritize detection accuracy for system-threatening conditions.

The integration of these classification models into hybrid blockchain architecture presents promising research opportunities. Off-chain machine learning inference combined with on-chain verification protocols could enable intelligent smart contract execution while maintaining blockchain immutability and transparency. Additionally, adaptive threshold mechanisms that dynamically adjust performance categories based on network conditions could enhance system responsiveness to varying operational environments.

The development of federated learning approaches for blockchain performance monitoring could enable collaborative model improvement across multiple blockchain networks while preserving privacy and competitive advantages. Such distributed learning frameworks would benefit from the robust classification foundation established in this research.

5. Conclusion

This research effectively illustrates the application of supervised learning techniques in classifying blockchain system performance into seven distinct categories based on throughput. Of the models evaluated, Random Forest emerged as the most dependable, achieving an accuracy of 91.35%, and demonstrating strong capability in modeling complex, non-linear relationships and system dynamics.

The results have strategic implications for the implementation of blockchain in intelligent educational systems, going beyond benchmarking. The ability to intelligently stratify performance in real-time paves the way for integrating such classifiers into blockchain-based Learning Management Systems, where smart contracts demand timely, trustworthy, and automated execution. For example, smart contracts triggered by usage thresholds, grading events, or credential issuance would benefit from accurate performance monitoring to avoid latency or resource bottlenecks.

Future work will explore the proposed classification models within blockchain-LMS architecture, such as Moodle integrated with Hyperledger Besu. The aim is to build hybrid systems combining off-chain ML classification and on-chain verification, enabling reliable smart contract orchestration in decentralized learning environments.

Acknowledgement

The author expresses gratitude for all the assistance in data processing that has been very meaningful to this research. Appreciation is also given to Universitas Negeri Malang and Universitas Merdeka Malang for the administrative support, research facilities, and conducive academic environment during the process of data collection and analysis. Special thanks are directed to the supervisors and colleagues for their constructive scientific input in this research.

References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”, Accessed: Jul. 07, 2025. [Online]. Available: www.bitcoin.org

[2] I. Gede, I. Sudipa, W. Aditama, and C. P. Yanti, “Blockchain Technology Model on Virtual Museum as an Effort to Enchance Balinese Cultural Metaverse,” *International Journal of Engineering, Science and Information Technology*, vol. 5, no. 3, pp. 394–401, Jun. 2025, doi: 10.52088/ijesty.v5i3.968.

[3] A. Alammary, S. Alhazmi, M. Almasri, and S. Gillani, “Blockchain-Based Applications in Education: A Systematic Review,” 2019, doi: 10.3390/app9122400.

[4] M. Turkanović, M. Hölbl, K. Košič, M. Heričko, and A. Kamišalić, “EduCTX: A Blockchain-Based Higher Education Credit Platform,” *IEEE Access*, vol. 6, pp. 5112–5127, 2018, doi: 10.1109/ACCESS.2018.2789929.

[5] S. Ma’arif, F. Maufiquddin, B. Dhevyantoa, K. Krisdiana, and A. Setiawan, “Decentralized Finance: Bibliometric Analysis and Research Trends,” *International Journal of Engineering, Science and Information Technology*, vol. 5, no. 2, pp. 474–484, May 2025, doi: 10.52088/IJESTY.V5I2.886.

[6] L. Barreñada, P. Dhiman, D. Timmerman, A.-L. Boulesteix, and B. Van Calster, “Understanding overfitting in random forest for probability estimation: a visualization and simulation study,” vol. 8, p. 14, 2024, doi: 10.1186/s41512-024-00177-1.

[7] S. Lai, Q. Yang, W. He, Y. Zhu, and J. Wang, “Image Retrieval Method Combining Bayes and SVM Classifier Based on Relevance Feedback with Application to Small-scale Datasets,” *Tehnički vjesnik*, vol. 29, no. 4, pp. 1236–1246, Jun. 2022, doi: 10.17559/TV-20210925093644.

[8] M. A. Mohammed, M. Boujelben, and M. Abid, “A Novel Approach for Fraud Detection in Blockchain-Based Healthcare Networks Using Machine Learning,” *Future Internet 2023, Vol. 15, Page 250*, vol. 15, no. 8, p. 250, Jul. 2023, doi: 10.3390/FI15080250.

[9] S. Y. Meng, A. Orvieto, D. Y. Cao, and C. De Sa, “Gradient Descent on Logistic Regression with Non-Separable Data and Large Step Sizes,” Jun. 2024, doi: https://doi.org/10.48550/arXiv.2406.05033.

[10] A. Hirsi, L. Audah, A. Salh, M. A. Alhartomi, and S. Ahmed, “Detecting DDoS Threats Using Supervised Machine Learning for Traffic Classification in Software Defined Networking,” *IEEE Access*, vol. 12, pp. 166675–166702, 2024, doi: 10.1109/ACCESS.2024.3486034.

[11] U. Pujianto, H. A. Rosyid, and A. C. Putra, “Performance Comparison of Ensemble-based k-Nearest Neighbor and CART Classifiers for the Classification of Adaptive e-learning User Knowledge Levels,” in *1st UMGESHIC International Seminar on Health, Social Science and Humanities (UMGESHIC-ISHSSH 2020)*, 2021, pp. 243–25, doi: 10.2991/assehr.k.211020.037

[12] Z. Chen, “Machine Learning-Based Decision Support to Secure Internet of Things Sensing,” Université d’Ottawa/University of Ottawa, 2023.

[13] H. A. Rosyid, U. Pujianto, and M. R. Yudhistira, “Classification of Lexile Level Reading Load Using the K-Means Clustering and Random Forest Method,” *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, pp. 139–146, 2020, doi: https://doi.org/10.22219/kinetik.v5i2.897.

[14] D. U. Soraya, S. Patmanthara, and G. D. K. Ningrum, “Growing Teaching Motivation for Future Teachers Through Microteaching,” *Letters in Information Technology Education (LITE)*, vol. 5, no. 2, pp. 71–75, Nov. 2022, doi: 10.17977/UM010V5I22022P71-75.

[15] N. M. Yusof, T. Manogaran, and H. Thiagu, “Microteaching as a Method in Developing Teaching Skills,” *Journal of Creative Practices in Language Learning and Teaching*, vol. 11, no. 2, 2023, doi: 10.24191/CPLT.V11I2.2033.

[16] J. Konseling, D. Pendidikan, Z. Zulhimma, Z. Zulhammi, and A. Abdurrahman, “Teachers’ self-efficacy: through micro teaching, practical field experience, and motivation,” *Jurnal Konseling dan Pendidikan*, vol. 10, no. 3, pp. 444–460, Sep. 2022, doi: 10.29210/190000.

[17] S. Patmanthara, O. D. Yuliana, F. A. Dwiyanto, and A. P. Wibawa, “The Use of Ladder Snake Games to Improve Learning Outcomes in Computer Networking,” *International Journal of Emerging Technologies in Learning (iJET)*, vol. 14, no. 21, pp. 243–249, Nov. 2019, doi: 10.3991/IJET.V14I21.10953.

[18] N. A. Saran and F. Nar, “Fast binary logistic regression,” *PeerJ Comput Sci*, vol. 11, p. e2579, 2025, doi: https://doi.org/10.7717/peerj-cs.2579.

[19] N. Šarlija, A. Bilandžić, and M. Stanic, "Logistic regression modelling: procedures and pitfalls in developing and interpreting prediction models," *Croatian Operational Research Review*, pp. 631–652, 2017.
[20] C. El Morr, M. Jammal, H. Ali-Hassan, and W. El-Hallak, "Logistic regression," in *Machine Learning for Practical Decision Making: A Multidisciplinary Perspective with Applications from Healthcare, Engineering and Business Analytics*, Springer, 2022, pp. 231–249.
[21] C. Ngufor and J. Wojtusiak, "Extreme logistic regression," *Adv Data Anal Classif*, vol. 10, pp. 27–52, 2016, doi: https://doi.org/10.1007/s11634-014-0194-2.
[22] N. Senaviratna, T. Cooray, and others, "Diagnosing multicollinearity of logistic regression model," *Asian Journal of Probability and Statistics*, vol. 5, no. 2, pp. 1–9, 2019, doi: 10.9734/ajpas/2019/v5i230132.
[23] T.-N. Do, "Towards simple, easy to understand, an interactive decision tree algorithm," *College Information Technology Can tho University, Can Tho, Vietnam, technology report*, pp. 1–6, 2007.
[24] K. Kim and J. Hong, "A hybrid decision tree algorithm for mixed numeric and categorical data in regression analysis," *Pattern Recognit Lett*, vol. 98, pp. 39–45, 2017, doi https://doi.org/10.1016/j.patrec.2017.08.011.
[25] J. Su and H. Zhang, "A fast decision tree learning algorithm," in *Aaai*, 2006, pp. 500–505.
[26] M. Bramer, "Avoiding overfitting of decision trees," *Principles of data mining*, pp. 119–134, 2007, doi: https://doi.org/10.1007/978-1-84628-766-4_8.
[27] R.-H. Li and G. G. Belford, "Instability of decision tree classification algorithms," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 570–575, doi: https://doi.org/10.1145/775047.775131.
[28] A. B. Shaik and S. Srinivasan, "A brief survey on random forest ensembles in classification model," in *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2018, Volume 2*, 2019, pp. 253–260, doi: https://doi.org/10.1007/978-981-13-2354-6_27.
[29] M. A. Salam, A. T. Azar, M. S. Elgendy, and K. M. Fouad, "The effect of different dimensionality reduction techniques on machine learning overfitting problem," *Int. J. Adv. Comput. Sci. Appl*, vol. 12, no. 4, pp. 641–655, 2021, doi: http://dx.doi.org/10.14569/IJACSA.2021.0120480.
[30] M.-H. Roy and D. Larocque, "Robustness of random forests for regression," *J Nonparametr Stat*, vol. 24, no. 4, pp. 993–1006, 2012, doi https://doi.org/10.1080/10485252.2012.715161.
[31] K. J. Archer and R. V Kimes, "Empirical characterization of random forest variable importance measures," *Comput Stat Data Anal*, vol. 52, no. 4, pp. 2249–2260, 2008, doi https://doi.org/10.1016/j.csda.2007.08.015.
[32] M. Aria, C. Cuccurullo, and A. Gnasso, "A comparison among interpretative proposals for Random Forests," *Machine Learning with Applications*, vol. 6, p. 100094, 2021, doi https://doi.org/10.1016/j.mlwa.2021.100094.
[33] T. P. Dinh, C. Pham-Quoc, T. N. Thinh, B. K. Do Nguyen, and P. C. Kha, "A flexible and efficient FPGA-based random forest architecture for IoT applications," *Internet of Things*, vol. 22, p. 100813, 2023, doi: https://doi.org/10.1016/j.iot.2023.100813.
[34] Z. El Mrabet, N. Sugunaraj, P. Ranganathan, and S. Abhyankar, "Random forest regressor-based approach for detecting fault location and duration in power systems," *Sensors*, vol. 22, no. 2, p. 458, 2022, doi: 10.3390/s22020458.
[35] N. Ardeshir, C. Sanford, and D. J. Hsu, "Support vector machines and linear regression coincide with very high-dimensional features," *Adv Neural Inf Process Syst*, vol. 34, pp. 4907–4918, 2021, doi: https://doi.org/10.48550/arXiv.2105.14084.
[36] M. S. Reza, U. Hafsha, R. Amin, R. Yasmin, and S. Ruhi, "Improving SVM performance for type II diabetes prediction with an improved non-linear kernel: Insights from the PIMA dataset," *Computer Methods and Programs in Biomedicine Update*, vol. 4, p. 100118, 2023, doi https://doi.org/10.1016/j.cmpbup.2023.100118.
[37] F. A. K. Q. Alnagashi, N. A. Rahim, S. A. A. Shukor, and M. H. A. Hamid, "Mitigating Overfitting in Extreme Learning Machine Classifier Through Dropout Regularization," *Applied Mathematics and Computational Intelligence (AMCI)*, vol. 13, no. 1, pp. 26–35, 2024, doi: https://doi.org/10.58915/amci.v13iNo.1.561.
[38] Z. Akram-Ali-Hammouri, M. Fernández-Delgado, E. Cernadas, and S. Barro, "Fast support vector classification for large-scale problems," *IEEE Trans Pattern Anal Mach Intell*, vol. 44, no. 10, pp. 6184–6195, 2021, doi: 10.1109/TPAMI.2021.3085969.
[39] P. Ramadevi and R. Das, "An extensive analysis of machine learning techniques with hyper-parameter tuning by Bayesian optimized SVM kernel for the detection of human lung disease," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3422449.
[40] S. Zhang, "Challenges in KNN classification," *IEEE Trans Knowl Data Eng*, vol. 34, no. 10, pp. 4663–4675, 2021, doi: 10.1109/TKDE.2021.3049250.
[41] F. Acito, "k nearest neighbors," in *Predictive Analytics with KNIME: Analytics for Citizen Data Scientists*, Springer, 2023, pp. 209–227.
[42] B.-B. Jia and M.-L. Zhang, "MD-KNN: An instance-based approach for multi-dimensional classification," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 126–133, doi: 10.1109/ICPR48806.2021.9412974.
[43] S. Adhikary and S. Banerjee, "Introduction to distributed nearest hash: On further optimizing cloud based distributed KNN variant," *Procedia Comput Sci*, vol. 218, pp. 1571–1580, 2023, doi: https://doi.org/10.1016/j.procs.2023.01.135.
[44] H. Zhu *et al.*, "Visualizing large-scale high-dimensional data via hierarchical embedding of KNN graphs," *Visual Informatics*, vol. 5, no. 2, pp. 51–59, 2021, doi https://doi.org/10.1016/j.visinf.2021.06.002
[45] H. Henderi, T. Wahyuningsih, and E. Rahwanto, "Comparison of Min-Max normalization and Z-Score Normalization in the K-nearest neighbor (kNN) Algorithm to Test the Accuracy of Types of Breast Cancer," *International Journal of Informatics and Information Systems*, vol. 4, no. 1, pp. 13–20, 2021, doi: https://doi.org/10.47738/ijiis.v4i1.73.