Predicting Blockchain Application Performance with Machine Learning Techniques

Willian M. Rodrigues[1], Silvia D. Rissino[2], Karin S. Komati[1]

[1]*Graduate program in Applied Computing (PPComp)*
Instituto Federal do Espírito Santo (IFES), Campus Serra
Av. dos Sabiás, 330, Morada de Laranjeiras, 29166-630, Serra-ES, Brazil
willian15mr@gmail.com, kkomati@ifes.edu.br
[2]*Universidade Federal do Espírito Santo (UFES)*
Rodovia BR-101, Km 60, Bairro Litorâneo, 29932-540, São Mateus-ES, Brazil
silvia.rissino@ufes.br

Abstract. Blockchain technology is a distributed ledger designed to record all transactions within its network, characterized by its decentralized nature, resistance to tampering, and attributes such as consistency, anonymity, and traceability. However, evaluating blockchain applications' performance can be complex due to their intricate and distributed infrastructure. This research employs machine learning model-based methods to predict blockchain systems' performance using predetermined configuration parameters. The data used in this study is derived from a blockchain simulator, generating blockchain data to facilitate performance predictions. The simulation process involves using simulated data and configuration settings for each run, including parameters such as the number of nodes, the number of miners, consensus algorithm, maximum block size, and transaction quantities, among others. Output metrics such as the total number of blocks, transaction rate, block propagation time, and latency are utilized to assess network performance. The simulator was run 184 times with various configurations. Our findings indicate that the Random Forest model outperformed other models used in the experiments, achieving the highest R² scores for multiple metrics, such as 0.987 for total number of transactions and 0.765 for average block propagation time, while also demonstrating lower RMSE values, indicating more accurate predictions.

Keywords: Root Mean Square Error, R^2 Score, Blockchain simulator.

1 Introduction

Blockchain is a distributed ledger technology that enables secure and transparent transactions without the need for intermediaries [1]. Initially proposed as the foundation of Bitcoin, this technology has expanded to various applications, including finance, supply chains, healthcare, and governance, due to its decentralized and immutable nature [2]. With the continuous advancement of blockchain technologies, it is important to evaluate not only the structure and performance of networks but also to explore enhancement and prediction methods through advanced machine learning (ML) and optimization techniques.

Despite the success and increasing adoption of blockchain technology, significant challenges regarding its efficiency and scalability remain. The complex and distributed nature of blockchain networks makes it difficult to predict and optimize their performance [3]. Furthermore, the configuration of simulation parameters, such as the number of nodes, block size, number of transactions, and miner involvement, can drastically influence network behavior. These challenges create a gap in the literature where the application of ML and optimization techniques can offer innovative solutions. For instance, the work of Albshri et al. [4] focuses on the performance evaluation of blockchain-based applications using ML techniques to predict and enhance their functionality.

This work aims to continue the research of Albshri et al. [4] by incorporating previously used ML algorithms and introducing additional ones. Albshri et al. employed k-Nearest Neighbors (KNN) and Support Vector Machine (SVM). This study extends the analysis by including these algorithms and adding Random Forest (RF) [5], Linear Regression (LR) [6], XGBoost [7], CatBoost [8], and LightGBM [9]. Through these methods, it will be possible to anticipate network behaviors and trends, identifying patterns and relationships between input data and output metrics. This study also uses the same dataset as the work of Albshri et al. [4], which created a comprehensive dataset through a simulation environment, controlling nine configuration parameters to generate thirteen performance metrics. This dataset enables a detailed evaluation of blockchain applications by providing diverse data points for various configurations

The overall objective of this work is to provide an integrated approach that combines blockchain network simulation, ML, optimization, and data analysis, enabling a deeper understanding of network behavior and the development of strategies to improve its performance and efficiency in various scenarios. Specifically, we seek to apply ML techniques to identify and predict patterns in simulation data, use hyperparameter optimization algorithms to improve the performance of ML models, and analyze the results using performance metrics such as R² score and RMSE in order to provide a deeper understanding of network behavior and develop strategies to improve its performance and efficiency in different scenarios.

The structure of the article is as follows: Section 2 explores related works. Section 3 describes the experimental methodology conducted. Section 4 details the results obtained and their analysis. Finally, Section 5 concludes the article with final considerations and suggestions for future work.

2 Related Work

Baliga et al. [10] conduct a detailed performance analysis of Quorum, an enterprise-focused blockchain platform derived from Ethereum. The study evaluates Quorum's performance in terms of transaction throughput, latency, and scalability under various configurations and workloads. The analysis reveals that Quorum can handle up to 1650 transactions per second (TPS) with a block time of 50 milliseconds in a network of 3 nodes. When using the Raft consensus mechanism, Quorum achieves better performance, with the capability to handle up to 1650 TPS and maintain lower latency compared to the Istanbul BFT mechanism, which provides slightly higher throughput only up to 1500 TPS but with significantly higher latency. The study also demonstrates that as the network size increases, the transaction throughput decreases and latency increases, highlighting the trade-offs between scalability and performance in Quorum. These quantitative results provide valuable insights into optimizing Quorum for enterprise blockchain applications, informing decisions on consensus mechanism selection and network configuration.

Woznica et al. [11] address security issues and challenges faced by public consensus networks, including scalability, throughput, and latency, along with essential considerations for enterprise blockchain networks. The work proposes a comprehensive performance evaluation of private blockchain implementations and algorithms, focusing on solutions like Hyperledger Fabric, Sawtooth, and Iroha. The evaluation uses metrics such as transaction latency, throughput, and network scalability, and tests various parameters including transaction sending rate, block size, and network traffic distribution. The results indicate that Hyperledger Fabric achieved a transaction throughput of up to 150 transactions per second (TPS) with a varying latency depending on network configuration. Sawtooth demonstrated a throughput of around 27.4 TPS with an average latency of 2.34 seconds for specific configurations, while Iroha showed varying performance based on network size and block size, with latency increasing significantly as network size grew. Additionally, the study found that increasing block size generally improved throughput but also increased latency, and higher transaction sending rates led to network congestion, impacting overall performance. These quantitative results provide valuable insights for optimizing private blockchain networks for enterprise use.

The article by Albshri et al. (2023) presents two primary contributions in the domain of blockchain performance evaluation. Firstly, the study employs KNN and SVM algorithms to predict the performance of blockchain applications based on various configuration parameters. The dataset used for training these models was generated through a simulation environment, controlling nine configuration parameters to produce thirteen performance metrics. The results demonstrated that the KNN model outperformed SVM, achieving an R² score of 0.92 compared to 0.89 for SVM, and a lower Root Mean Squared Error (RMSE) of 67.06 versus 72.44 for SVM. Secondly, the research introduces an Improved Salp Swarm Optimization (ISO) algorithm, which incorporates Rough Set Theory to handle uncertainties and optimize blockchain configuration parameters for desired performance levels. The ISO model showed a 4% reduction in RMSE compared to the standard Salp Optimization (SO) model and achieved an R² score closer to 0.83 for the target performance metric M13 (in this paper renamed as M12) of 1100 transactions per second, outperforming other optimization algorithms like Particle Swarm Optimization, Harris Hawk Optimization, Gray Wolf Optimization, Artificial Bee Colony, and Salp Optimization.

3 Materials and Methods

The experiments were conducted on a Windows 10 computer with an 11th Gen Intel Core i7-11800H processor (2.304 GHz), 15.75 GB of RAM, and an NVIDIA GeForce RTX 3060 Laptop GPU (6144 MB). The source code is available on GitHub[1].

[1]https://github.com/WillianMR/BlockchainOptimization

3.1 Dataset

The dataset utilized in this study corresponds to the dataset referenced in Albshri et al. [4], sourced from a Google Sheet named “BlockchainDataset”. This dataset comprises several sheets that delineate various facets of a simulated blockchain network, encompassing the Config, Results, Block, Transaction, Transaction Latency, and Transaction Pool sheets. Relevant data from these sheets were extracted and organized into pandas DataFrames to facilitate streamlined data manipulation and analysis. The simulated blockchain model underwent 184 iterations, each employing distinct parameter configurations detailed in Table 1. Conditional features were identified and subsequently used in conjunction with a defined set of performance metrics M (Table 2) to assess the blockchain’s operational efficacy.

Assume a number of configuration parameters (input) and performance metrics (output) of a blockchain-based solution as follows:

1. Configuration Parameters in Config sheet, P: the set of Z parameters P = {P0, P1, . . . , PZ} represents the input configuration of the blockchain network. Table 1 provides information about the configuration columns used in the simulations, identification number, name, data type, minimum and maximum values. These parameters are critical for defining the initial setup and conditions under which the blockchain simulations were conducted. Notably, columns P1 (No. of Miner), P2 (Transactions), P3 (Consensus Algorithm), and P9 (Simulation Time) do not have any variation in their values, all being constant across the simulations. These parameters are critical for defining the initial setup and conditions under which the blockchain simulations were conducted.
2. Performance Metrics in Results sheet, M: the set of n metrics M = {M0, M1, . . . , Mn} represent the conditional outputs with respect to the given parameters P. Table 2 outlines the results dataset, identification number, name, data type, minimum and maximum values. Each column in the results data serves as an output metric used to evaluate the performance of the ML models in predicting and analyzing blockchain network behavior. Table 2 outlines the results dataset, identification number, name, and data type. It is important to note that the column M4 (Total No. of Blocks without Tx) does not have any variation in its values, being constant across the simulations. Therefore, this column was not used for training or testing the models. Each column in the results data serves as an output metric used to evaluate the performance of the ML models in predicting and analyzing blockchain network behavior.

Table 1. Config Columns Information

ID	Description	Data Type	Min	Max
P0	No. of Node	Integer	3.0	15.0
P1	No. of Miner	Integer	1.0	1.0
P2	Transactions	Integer	1.0	1.0
P3	Consensus Algorithm	Integer	1.0	1.0
P4	Total No of Transactions Per Sec	Float	9.0	1650.0
P5	Max Block Size	Integer	1.0	1.0
P6	Max Tx Size	Integer	0.064	0.064
P7	Min Tx Size	Integer	0.001	0.001
P8	Block Interval	Float	0.05	0.0999
P9	Simulation Time	Float	1.0	1.0

Table 2. Results Columns Information

ID	Description	Data Type	Min	Max
M0	Total No. of Blocks	Integer	9.0	43.0
M1	Total No. of Blocks include Tx	Integer	8.0	42.0
M2	Total No of Transactions	Integer	9.0	1203.0
M3	Total No. of Pending Tx	Integer	0.0	450.0
M4	Total No. of Blocks without Tx	Integer	1.0	1.0
M5	Avg. Block Size (MB)	Float	0.0303	0.9716
M6	Avg. No. of Tx per block	Integer	1.0	30.8462
M7	Avg. of Tx Inclusion Time (secs)	Float	0.4212	0.5854
M8	Avg. Tx Size (MB)	Float	0.0279	0.0373
M9	Avg. Block Propagation (secs)	Float	0.0209	0.0894
M10	Avg. Transaction Latency (secs)	Float	0.0160	0.2664
M11	Transactions execution (secs)	Float	0.8047	0.9995
M12	Transaction Throughput (Tx/secs)	Float	11.1841	1248.6553

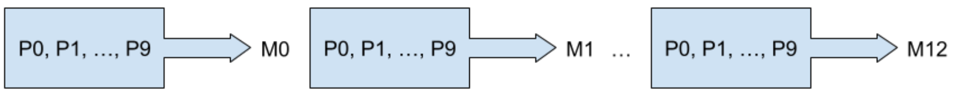

Figure 1. Training and testing flow with parameters and metrics

These tables are essential because they contain the configurations and results used to train the models, allowing us to evaluate performance based on each column of the results dataset. The configuration data provides the input parameters for the simulations, while the results data contains the corresponding output metrics. By analyzing these datasets, we can assess how effective different ML models are at predicting and optimizing various aspects of blockchain network performance.

As illustrated in Figure 1, the models are trained and tested using a combination of parameters P0, P1, ..., P9. During each training session, the model focuses on predicting a single column M0, M1, ..., M12 separately, ensuring that the result of one column does not influence the result of another. This method allows for an isolated evaluation of each performance metric, providing a clear understanding of how each parameter affects different aspects of the blockchain network.

3.2 Models

Supervised learning algorithms encompass a diverse array of models tailored for classification, regression, and other predictive tasks. The fundamental concept of SVM involves identifying a hyperplane that optimally separates the dataset into classes [12]. The KNN method is a non-parametric approach wherein a sample is classified based on the majority vote of its nearest neighbors, assigning the sample to the most prevalent class among its K nearest neighbors; for regression tasks, the predicted value is derived by averaging the values of its K nearest neighbors [13]. Logistic Regression is a model employed to predict the probability of a categorical dependent variable, estimating the likelihood that an observation belongs to a particular category, and is particularly advantageous when the output variable is binary [6]. RF is an ensemble learning technique that constructs multiple decision trees during training and makes predictions by aggregating the outcomes of all trees, thereby enhancing classification accuracy, mitigating overfitting, and maintaining high generalization capacity [5].

Advanced boosting frameworks like XGBoost, LightGBM, and CatBoost have been developed to enhance computational efficiency and predictive accuracy. XGBoost (Extreme Gradient Boosting) is an optimized framework for decision trees that employs boosting, enhancing computational efficiency and predictive accuracy through extensive regularization and pruning techniques to avoid overfitting [7]. LightGBM is a boosting framework that uses gradient boosting to efficiently construct decision trees, noted for its ability to handle large datasets and its speed and efficiency by employing binning techniques to convert continuous data into discrete bins, thus accelerating the training process [9]. Finally, CatBoost is a gradient boosting algorithm that can handle categorical variables without extensive preprocessing, designed to provide optimal results with minimal configurations, especially useful in datasets with many categorical variables [8].

For model selection and hyperparameter optimization, three distinct methods were employed: Exhaustive Grid Search, Randomized Parameter Optimization, and Optuna. Exhaustive Grid Search operates by evaluating multiple combinations of parameter values, cross-validating each combination to determine the optimal parameters for a model. Despite being computationally intensive, it is widely used due to its thoroughness in identifying the best hyperparameters for ML models [14]. Randomized Parameter Optimization is a hyperparameter tuning technique that randomly samples a fixed number of parameter combinations from a specified distribution, rather than exhaustively searching all possible combinations. This approach reduces computational cost and time compared to exhaustive grid search by focusing on a random subset of the hyperparameter space, which can still effectively identify suitable parameter values [15]. Optuna is an open-source hyperparameter optimization framework designed to automate the process of tuning hyperparameters for ML models. It employs efficient sampling algorithms, such as the Tree-structured Parzen Estimator (TPE) and the Asynchronous Successive Halving Algorithm (ASHA), to explore the hyperparameter space more effectively than traditional methods like grid search or random search [16].

Each model was evaluated using a variety of performance metrics with an 80-20 hold-out train-test split. The code was implemented in Python, leveraging data science libraries extensively: pandas for data manipulation, NumPy for numerical operations, and scikit-learn for building ML pipelines and model evaluation. Additionally, the libraries Optuna, joblib, matplotlib, and seaborn were utilized for hyperparameter optimization, model saving and loading, and data visualization, respectively.

3.3 Evaluation Metrics

The performance of ML models in this work is quantified using two key metrics: the R^2 Score and the Root Mean Squared Error (RMSE). The choice of these metrics is further motivated by their ability to provide a clear, interpretable evaluation of model performance in the context of blockchain network simulations. These metrics were selected based on their prevalence in the literature, specifically in the context of blockchain performance evaluation, as highlighted by Albshri et al. (2023) [4]. Both R^2 and RMSE provide comprehensive insights into the accuracy and reliability of the predictions made by the models used in this study, such as KNN and SVM, which have been effectively applied in prior research.

The R^2 Score, also known as the coefficient of determination, measures how well the predicted values approximate the real data points. It captures the proportion of variance in the dependent variable that is predictable from the independent variables, making it a useful metric for understanding the effectiveness of the models in capturing complex relationships in blockchain performance metrics [17].

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{1}$$

where SS_{res} is the sum of squares of residuals and SS_{tot} is the total sum of squares.

RMSE measures the average magnitude of the errors between the predicted and actual values. It is the square root of the average of squared differences between predictions and actual observations [18]. RMSE is a widely used metric in predictive modeling and is particularly useful for assessing the accuracy of models when errors in prediction carry significant implications.

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

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of observations.

4 Results

Table 3 provides information on the best performing model for each column {M0, M1, ... , M12} except M4, including the R^2 score, RMSE, training time, and prediction time. The R^2 score and RMSE values are inversely correlated for the same model and output M. However, each row in Table 3 was trained independently for a specific performance metric, making each row independent of the others. This explains why similar R^2 values can correspond to significantly different RMSE values, such as an R^2 of 0.782 with an RMSE of 27.782 for M0 and an R^2 of 0.765 with an RMSE of 0.005 for M9.

The RF model achieved the highest R^2 scores for columns M0, M2, M3, M6, M9, and M11. This performance can be attributed to its ensemble nature, which aggregates multiple decision trees to enhance predictive performance. RF effectively handles feature interactions and variances, resulting in higher accuracy in predicting complex blockchain behaviors. The model showed moderate performance in column M11.

Table 3. Best Model for Each Column

Column	**Description**	**Model**	R^2	**RMSE**	**Train Time (s)**	**Predict Time (s)**
M0	Total No. of Blocks	**Random Forest**	0.782	27.782	0.022	0.022
M1	Total No. of Blocks include Tx	**LightGBM**	0.813	36.493	0.019	0.019
M2	Total No of Transactions	**Random Forest**	0.987	27.782	0.020	0.020
M3	Total No. of Pending Tx	**Random Forest**	0.767	33.539	0.028	0.028
M5	Avg. Block Size (MB)	**XGBoost**	0.965	0.060	0.033	0.033
M6	Avg. No. of Tx per block	**Random Forest**	0.959	1.819	0.033	0.033
M7	Avg. of Tx Inclusion Time (secs)	**Linear Regression**	0.479	0.022	0.016	0.016
M8	Avg. Tx Size (MB)	**KNN**	0.190	0.001	0.003	0.003
M9	Avg. Block Propagation (secs)	**Random Forest**	0.765	0.005	0.017	0.017
M10	Avg. Transaction Latency (secs)	**LightGBM**	0.834	0.015	0.016	0.016
M11	Transactions execution (secs)	**Random Forest**	0.603	0.457	0.014	0.014
M12	Transaction Throughput (Tx/secs)	**XGBoost**	**0.989**	26.929	0.019	0.019

The LightGBM model performed best for columns M1 and M10, leveraging its gradient boosting capabilities. LightGBM's strength lies in its ability to build and combine multiple weak learners, making it adept at capturing subtle patterns in the data. The XGBoost model excelled in columns M5 and M12, due to its regularization techniques that prevent overfitting and improve generalization. This characteristic is particularly beneficial in scenarios with diverse transaction sizes and network conditions, typical in blockchain environments.

The LR model showed competitive performance in column M7, indicating that simpler linear relationships were predominant for that specific performance metric. LR's straightforward approach is sometimes advantageous when the underlying data structure is not overly complex. However, the overall performance in column M7 was relatively low. This could be due to the high variability and noise present in the blockchain network data, which Linear egression might not handle well due to its simplicity and assumption of linear relationships. The KNN model performed best for column M8, benefiting from its non-parametric nature, which allows it to adapt based on the proximity of data points. This method is useful when transaction data points are naturally clustered. Despite this, the performance in column M8 was notably low. This can be attributed to the curse of dimensionality, where KNN's effectiveness diminishes as the number of features increases. Additionally, the clustering of data points might not be as clear or distinct, making it challenging for KNN to find optimal neighbors for accurate predictions.

Figure 2(a) shows scatter plots of R^2 scores for each model across all columns. These visualizations highlight the variability in model performance, with some models consistently outperforming others in specific columns. Notably, the scatter plots for M2, M5, M6, and M12 present better R^2 values across all models, indicating these metrics are generally well-predicted by the models. In contrast, M8 shows poor R^2 values for all models, suggesting that this metric is more challenging to predict accurately. The variability in the results for M0, M1, M3, M9, and M10 indicates that the models' performance changes significantly depending on the target variable.

The box plots in Figure 2(b) provide additional insights into the distribution of R^2 scores for each model. These figures illustrate the overall performance trends and the variability in prediction accuracy. The box plots summarize the central tendency and spread of the R^2 scores, showing the median, quartiles, and potential outliers. This helps to identify models that not only perform well on average but also have consistent performance across different metrics. For example, RF shows a higher median R^2 score and less variability, indicating more reliable performance. On the other hand, models like KNN and SVM exhibit a wider range of R^2 scores, suggesting they may be more sensitive to specific configurations or metrics.

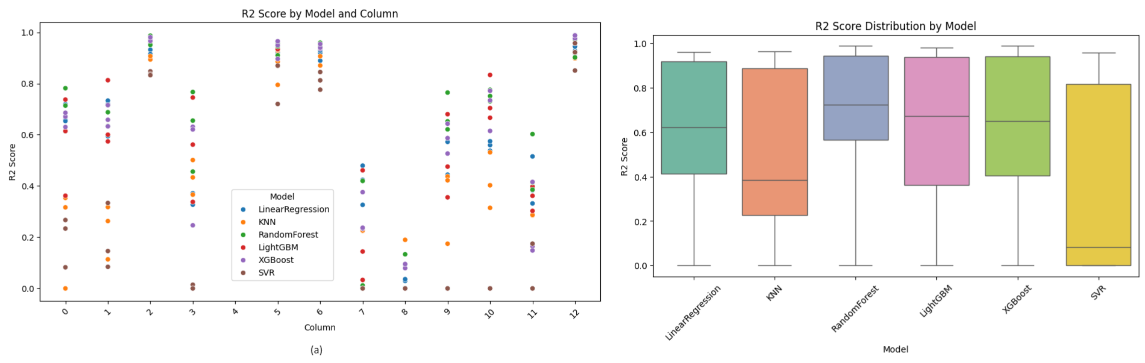

Figure 2. (a) Scatter plot of R^2 scores by model and M (column). (b) Box plot of R^2 score distribution by model.

The results of Albshri et al. [4] demonstrate that KNN, used as a regression tool to predict the overall performance of the blockchain in terms of metrics such as throughput, latency, and transaction success, achieved superior accuracy compared to SVM. The optimal k value was determined to be 5, resulting in an RMSE of 67.06 and an accuracy of 92%, while SVM, used as a classification algorithm, achieved an accuracy of 89%. Comparing these results with the models discussed in this paper, such as RF, which showed higher R^2 scores and lower RMSEs across various columns, highlights that the choice of ML model is highly dependent on the nature of the data and the characteristics of the predictive variables. While the main article focused on predicting overall performance using a single set of parameters, the models in the provided text were evaluated across different columns with varied metrics, highlighting the importance of a detailed and segmented analysis to capture the complexities of blockchain behavior. This underscores the difference in approach, where a column-by-column analysis can reveal specific nuances that a global assessment might miss, providing a more comprehensive and precise understanding of the system's performance.

The practical implications of our results are substantial, particularly for enterprises and organizations deploying blockchain solutions. Models like **RandomForest** and **XGBoost** provide reliable predictive power across a range of performance metrics, which can aid in optimizing network configurations, improving transaction throughput, and minimizing latency. These models, due to their scalability and ability to handle large datasets, are well-suited for real-world applications where blockchain systems are characterized by high variability and demand high accuracy.

5 Conclusions

The findings of this study underscore the importance of selecting appropriate ML models for predicting blockchain performance metrics. Our results demonstrated that the RF model performed better than the other models used in the experiments, managing intricate feature interactions and variances, and achieving superior R^2 scores across multiple variables. In contrast, models such as SVM displayed elevated RMSE values, indicative of less reliable predictions. The use of KNN and SVM as baseline models provided a comparative benchmark, highlighting the potential improvements offered by advanced models.

Future research will concentrate on refining these models further and exploring additional ML techniques to bolster predictive accuracy and computational efficiency. The generation of new simulations to augment dataset volume will play a pivotal role in enhancing model resilience and applicability. Moreover, a critical avenue for future exploration involves the evaluation of alternative performance metrics. For instance, Mean Absolute Error (MAE) provides a more interpretable error measure in terms of actual units, which may be more practical for real-time blockchain performance evaluations. On the other hand, Mean Absolute Percentage Error (MAPE) offers insights into relative prediction errors, but can introduce bias when dealing with very small true values.

Furthermore, hybrid model exploration and real-time data integration represent developmental avenues to address the dynamic attributes of blockchain systems. These endeavors will incorporate advanced ensemble methods and deep learning strategies to capture intricate data patterns and interdependencies effectively. Additionally, exploring feature engineering techniques to better encapsulate temporal and network-related facets of blockchain data will be imperative. Collectively, these advancements aim to foster the development of more precise, efficient, and dependable predictive models for evaluating blockchain performance, while ensuring that the choice of evaluation metrics supports both accuracy and interpretability within the context of blockchain systems.

Acknowledgements. The authors would like to thank FAPES/UnAC (No. FAPES 1228/2022 P 2022-CD0RQ, No. SIAFEM 2022-CD0RQ) for the financial support provided through the Sistema UniversidaES. Professor Komati thanks CNPq for the DT-2 grant (nº 302726/2023-3) and project nº407742/2022-0, also thanks FAPES for project nº 1023/2022 P:2022-8TZV6.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
[2] M. Pilkington. Blockchain technology: Principles and applications. In M. Oliver and R. Smith, eds, *Research Handbook on Digital Transformations*, pp. 225–253. Edward Elgar Publishing, 2016.
[3] I. S. Rao, M. M. Kiah, M. M. Hameed, and Z. A. Memon. Scalability of blockchain: a comprehensive review and future research direction. *Cluster Computing*, pp. 1–24, 2024.
[4] A. Albshri, A. Alzubaidi, and E. Solaiman. A model-based machine learning approach for assessing the performance of blockchain applications. In *2023 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pp. 46–55. IEEE, 2023.
[5] L. Breiman. Random forests. *Machine learning*, vol. 45, n. 1, pp. 5–32, 2001.
[6] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied logistic regression*. John Wiley & Sons, 2013.
[7] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.
[8] L. Prokhorenkova, A. Gulin, A. Dobrynin, and P. Tokmakov. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, vol. 31, 2018.
[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pp. 3146–3154, 2017.
[10] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee. Performance evaluation of the quorum blockchain platform. *arXiv preprint arXiv:1809.03421*, 2018.
[11] A. Woznica and M. Kedziora. Performance and scalability evaluation of a permissioned blockchain based on the hyperledger fabric, sawtooth and iroha. *Computer Science and Information Systems*, vol. 19, n. 2, pp. 659–678, 2022.
[12] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, vol. 20, n. 3, pp. 273–297, 1995.
[13] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, vol. 46, n. 3, pp. 175–185, 1992.
[14] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pp. 2546–2554, Red Hook, NY, USA. Curran Associates Inc., 2011.
[15] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, vol. 13, n. Feb, pp. 281–305, 2012.
[16] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631. ACM, 2019.
[17] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.
[18] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, vol. 22, n. 4, pp. 679–688, 2006.