ISSN: 1692-7257 - Volume 1 – Number 47 - 2026

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v1i47.4103

Blockchain oracle architecture with data validation based on machine learning processes

Arquitectura de oráculo blockchain con validación de datos basada en procesos de aprendizaje de maquina

PhD. Cristian Camilo Ordoñez Quintero[1], PhD. Hugo Armando Ordoñez Erazo[2], MSc. Juan Sebastián González Sanabria[3]

[1] ***Universidad de Nariño***, *Grupo de Investigación GREDIS, Pasto, Nariño-Colombia.*
[2] ***Universidad del Cauca***, *Grupo de Investigación GTI, Popayán, Cauca-Colombia*
[2] ***Universidad Pedagógica y Tecnológica de Colombia***, *Grupo de Investigación GIMI, Tunja, Boyacá-Colombia.*

Correspondence: *ccordonez@udenar.edu.co*

Received: *July 25, 2025.* ***Accepted:*** *December 20, 2025.* ***Published:*** *January 01, 2026.*

How to cite: *C. C. Ordoñez Quintero, H. A. Ordoñez Erazo y J. S. González Sanabria, "Arquitectura de oráculo blockchain con validación de datos basada en procesos de aprendizaje de maquina", RCTA, vol. 1, n.º. 47, pp. 72-82, Jan. 2026. Recuperado de https://ojs.unipamplona.edu.co/index.php/rcta/article/view/4103*

Abstract: The growing reliance on smart contracts in blockchain ecosystems has highlighted the need for reliable mechanisms to validate the data these contracts consume from external sources, commonly managed by oracles. This article presents a blockchain oracle architecture that incorporates a data validation system based on machine learning techniques, with the goal of enhancing data quality before it is incorporated into the blockchain. The solution integrates supervised and unsupervised models for anomaly detection, multi-label classification, time series analysis, and outlier detection. The validation process follows the CRISP-DM methodology and is complemented by a data integrity indicator based on descriptive statistics and majority voting mechanisms (hard voting), which allows for the automatic estimation of data acceptability. Experimental results, obtained in a testing environment using functional smart contracts, demonstrate improvements in the detection of inconsistencies and data manipulation, as well as in the reliability of automated decisions. This proposal provides a systematic and replicable strategy to mitigate risks associated with data consumption in blockchain applications.

Keywords: blockchain, machine learning, data processing, decision-making.

Resumen: La creciente dependencia de contratos inteligentes en ecosistemas blockchain ha puesto en evidencia la necesidad de contar con mecanismos confiables para validar los datos que estos consumen desde fuentes externas, comúnmente gestionadas por oráculos. Este artículo presenta una arquitectura de oráculo blockchain que incorpora un sistema de validación de datos basado en técnicas de Machine Learning, con el objetivo de fortalecer la calidad de los datos antes de su incorporación a la cadena de bloques. La solución integra modelos supervisados y no supervisados para la detección de anomalías, clasificación multietiqueta, análisis de series temporales y detección de valores atípicos. El proceso de validación sigue el enfoque CRISP-DM y se complementa con un indicador de integridad de datos basado en estadísticas descriptivas y mecanismos de votación por mayoría (hard

voting), que permite estimar automáticamente la aceptabilidad de los datos. Los resultados experimentales, obtenidos en un entorno de pruebas sobre contratos inteligentes funcionales, demuestran mejoras en la detección de inconsistencias y manipulación de datos, así como en la confiabilidad de las decisiones automatizadas. Esta propuesta aporta una estrategia sistemática y replicable para mitigar riesgos asociados al consumo de datos en aplicaciones blockchain.

Palabras clave: blockchain, aprendizaje de máquina, tratamiento de datos, toma de decisiones.

1. INTRODUCTION

The emergence of blockchain technology has profoundly transformed the way distributed information systems are conceived and executed [1]. Unlike traditional centralized architectures, blockchain proposes a decentralized model in which multiple nodes collaboratively participate to record, verify, and store transactions immutably [2]. This property has led to the development of smart contracts, computer programs that are executed automatically when certain pre-established conditions are met [3]. These contracts have gained relevance in a wide range of applications, from decentralized financial systems (DeFi) to supply chains, notarial systems, digital identity, and distributed governance [4].

However, despite their structural benefits in terms of transparency, decentralization, and auditability, smart contracts present a fundamental limitation: their inability to directly access data from outside the blockchain [5]. This is because the blockchain is designed as a closed and deterministic environment, in which all operations must be reproducible identically by all nodes. Consequently, smart contracts depend on intermediaries known as oracles, which are responsible for supplying the external information required for their execution [5]. This information can include asset prices, event outcomes, weather conditions, or any other data not originating directly within the blockchain network.

Oracles therefore represent a critical component within the blockchain ecosystem, acting as bridges between the off-chain and on-chain worlds [6]. However, this very function makes them one of the most vulnerable links in the system. Unlike the blockchain, which is immutable and resistant to manipulation thanks to consensus mechanisms such as Proof of Work or Proof of Stake, oracles introduce a point of trust that can be compromised [5]. An oracle that supplies incorrect, manipulated, or incomplete data can trigger the erroneous execution of smart contracts, with potentially catastrophic consequences in highly critical environments [5].

This phenomenon, known as the "oracle problem," has been identified as one of the main barriers to the massive adoption of blockchain solutions in contexts where data integrity is crucial [6], [7]. Despite advances in oracle decentralization techniques, such as Chainlink or Band Protocol, technical challenges persist regarding the verifiability, transparency, and quality of the data provided [8]. In many cases, current systems lack formal mechanisms to validate the consistency, accuracy, or reliability of external data, delegating this responsibility to external architectures that are often poorly auditable or centralized.

Against this backdrop, the need arises to develop new approaches that allow for explicit and replicable validations of the data entering the blockchain through oracles [6]. In particular, the fields of Data Science and Machine Learning offer powerful tools for data analysis, anomaly pattern detection, and prediction based on historical behavior [9].

Accordingly, this article proposes blockchain oracle architecture with a data validation system based on Machine Learning processes. The aim of the proposal is to enhance the integrity of external data used in smart contracts. To achieve this, a processing pipeline is implemented that includes supervised and unsupervised models oriented towards anomaly detection in data provided by oracles. This validation flow is structured according to the phases of the CRISP-DM (Cross-Industry Standard Process for Data Mining) model, ensuring a systematic, replicable methodology aligned with best practices in data engineering [10].

The architecture also includes a decision component based on a majority voting mechanism (hard voting) [11]. This approach allows not only the rejection of

anomalous or inconsistent data but also the establishment of objective acceptability thresholds that can be audited and traced in real time. Functionally, the solution is implemented as an intermediate layer between the external data source and the smart contract deployed on the blockchain, acting as an intelligent filter that reinforces the system's security and reliability.

This article is structured as follows. Section 2 presents the background required to contextualize the research problem, as well as a brief but substantial review of the state of the art, focusing on oracles, smart contracts, and data validation in blockchain environments. Section 3 then outlines the methods used for the development of the research, detailing the technical approach adopted, the data processing flow, and the proposed architecture. Sections 4 and 5 describe the implementation of the proposal and presents the results obtained after its deployment. Finally, Section 6 summarizes the conclusions of the study and proposes potential future research directions.

2. BACKGROUND

To identify the most relevant literature from the last five years, a systematic search was conducted in the Scopus and Web of Science (WOS) databases, using specific queries related to blockchain oracle architectures and the application of machine learning techniques. From this review, the most significant studies addressing the convergence between blockchain oracles and machine learning (ML) methods were selected, highlighting how this integration has given rise to advanced architectures aimed at ensuring the integrity, reliability, and traceability of off-chain data before its incorporation into smart contracts.

Zhang et al. [12] proposed a relevant approach, the TCO-DRL model (Trust-Aware and Cost-Optimized Reinforcement Learning), which uses deep reinforcement learning techniques to dynamically select oracle nodes based on multiple dimensions of reputation and cost. This model successfully reduces allocations to malicious nodes and optimizes costs without compromising data quality.

On the other hand, the DeepThought platform, introduced in 2022, incorporates a hybrid mechanism of verified voting and reputation, where users collaborate to validate information, and results are combined through a distributed reputation scheme that enhances resilience against human corruption [13].

Similarly, studies on blockchain for IIoT present a scheme for selecting oracle nodes based on Verifiable Random Functions (VRF) and reputation, combined with a sliding window data filtering algorithm. This approach improves quality of service, reduces variability, and increases accuracy in environments with malicious nodes [14].

Within the ML domain applied to blockchain, a recent systematic mapping analyzed 159 articles related to anomalies, classification, and on-chain data usage, identifying a growing trend in anomaly detection, challenges in standardization and scalability, as well as the limited interoperability between chains [14].

Authors such as Palaiokrassas et al. [15] have identified that a large portion of the research focuses on detecting anomalies or outlier patterns in blockchain data using supervised and unsupervised techniques, which is essential for validating data before it is used within an oracle system.

The literature also highlights the use of ML techniques to detect adversarial behavior or malicious manipulation, such as flash loan or Sybil attacks, through clustering, autoencoders, statistical filtering, and graph neural networks. Reinforcement learning agents have also been explored to dynamically adjust trust scores and reject erroneous data [16].

In addition, frameworks such as Oraichain integrate AI-centric oracles that execute AI off-chain and validate models through verifiable execution tests, inspiring new layers of trust to feed smart contracts with ML-processed and pre-validated data [7].

Finally, critical reviews of the literature underline inherent challenges: latency in data acquisition, variable source quality, interoperability issues between protocols and chains, scalability of distributed ML models, and the need for privacy-preserving ML techniques such as federated learning or differential privacy-based SGD in permissioned environments [17].

3. PROPOSAL

The proposed blockchain oracle architecture with data validation based on machine learning techniques is grounded in an extensive review of

recent literature, which supports the relevance and feasibility of each of its components. Figure 1 illustrates the architecture, and its components are explained and described below.

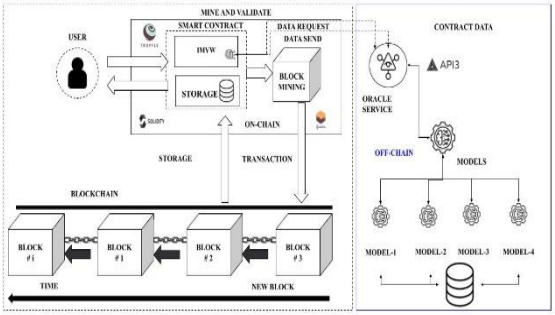

***Fig. 1**. Proposed blockchain oracle architecture.*
***Source:** Authors.*

First, smart contract implementation is developed in Solidity and tested using tools such as Truffle and Ganache, widely recognized in literature as robust environments for designing, compiling, deploying, and locally verifying blockchain contracts [18]. This layer enables the automation of agreements through programmed conditions that require external input for execution.

The acquisition of external data is carried out through decentralized oracles such as API3, which provide a secure and verifiable interface between the physical world and the blockchain. These oracles are operated directly by data providers, reducing the possibility of intermediate manipulation and improving source traceability [19].

The proposal incorporates a validation module based on a Multi-level Weighted hard Voting scheme (IMWV), which receives predictions from multiple classification models and produces a consolidated decision with weights assigned according to each model's historical performance. This technique has proven to be highly effective in improving accuracy in multi-label classification and anomaly detection tasks involving heterogeneous data [20].

Once validated, the data is integrated into the chain through a locally simulated mining process. This inclusion guarantees cryptographic integrity, traceability, and resistance to tampering—essential aspects in the context of smart contracts linked to external variables.

Additionally, architecture includes a continuous learning mechanism through which machine learning models are periodically updated based on validated data incorporated into the chain. This feedback loop allows the models to adapt to new conditions and maintain high predictive performance, aligning with current trends in online learning and distributed federated learning.

4. IMPLEMENTATION

The development of the proposed architecture was carried out following the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, adapted to the context of data validation in blockchain oracles. The main phases are described below, as illustrated in Figure 2.

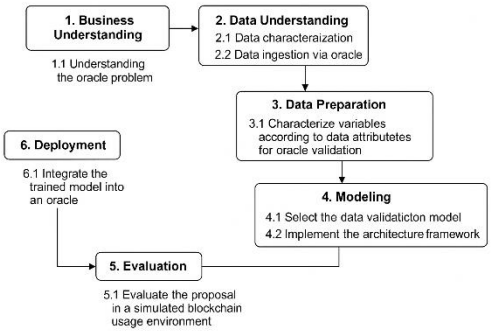

***Fig. 2**. Implementation of the architecture and ML models for oracle data validation. **Source:** Authors.*

4.1. Business Understanding

The first phase addressed the oracle problem, focusing on its critical role in the automatic execution of smart contracts that rely on real-world (off-chain) information. The main issue identified was the lack of effective mechanisms to validate the truthfulness, consistency, and completeness of the data provided by oracles, which can lead to erroneous or manipulated contractual decisions.

4.2. Data

This study focuses on a highly volatile smart contract, exemplified by coffee futures contracts. The selected variables—temperature, relative humidity, altitude, precipitation, variety, and coffee price—were included due to their direct relevance to the validation of external data used by smart contracts in agroclimatic futures domains [21]. These variables originate from real-world sources that commonly feed meteorological and agricultural information oracles; therefore, their role within architecture is grounded in the need to ensure coherence, internal correlation, and temporal

consistency before allowing their use within the smart contract.

Each variable serves a specific purpose in the validation process:

- Temperature, humidity, and precipitation enable the detection of abrupt meteorological anomalies or physically impossible values.
- Altitude acts as a spatial control variable in climatic analysis.
- Variety (agricultural category) allows validation of consistency between the crop type and its climatic response.

Coffee price is incorporated as a variable susceptible to manipulation in financial oracles, making it critical to prevent erroneous execution of futures contracts.

Subsequently, the available data were characterized (2.1), drawing from environmental sensors, meteorological stations, commercial APIs, and historical agricultural records. In addition, a data ingestion mechanism from external sources via oracles was developed (2.2), integrating services such as API3 to provide decentralized and verifiable data.

4.3. Data Preparation

Once the nature of the data sources was understood, the data preparation phase was initiated (3.1). In this stage, criteria were defined to identify data integrity attributes, including internal consistency, timeliness, completeness, and source reliability.

To construct the training dataset for the models, a binary labeling process was implemented to classify each event as valid or invalid. Labels were assigned based on three criteria:

- Internal consistency: coherence among meteorological and agricultural variables.
- Temporal coherence: direct comparison with historically recorded behavior using time-series techniques.
- Threshold verification: ranges provided by official sources or external reference data (API3 and climatic stations).

The attributes presented in Table 1 were used as inputs to train models that validate data quality prior to their incorporation into the blockchain.

Table 1: *Oracle Data Labeling*

Altitude	Temperature	Humidity	Precision	Precision by Altitude	Temperature Consistency	Irrigation Consistency	Completeness	Reliability
1424.7	19.57	73.58	1	1	1	1	1	1
1770.4	19.48	77.08	1	1	1	1	1	1
1639.2	23.44	76.40	1	1	0	1	1	1
1559.2	19.50	67.31	1	1	0	1	0	1
1293.6	19.63	67.24	1	1	0	0	1	1

4.4. Modeling

During the modeling phase, state-of-the-art machine learning methods were selected (4.1). The models employed—Random Forest, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Extra Trees—were chosen due to their reported effectiveness in multi-label classification tasks, anomaly detection, and heterogeneous data validation. Each algorithm was trained on a previously cleaned and labeled dataset, following a binary procedure that distinguishes between valid and invalid events based on criteria of internal consistency, temporal coherence, and threshold-based technical verification.

To ensure stability and comparability across models, the dataset was split using a stratified partitioning scheme: 70% for training, 15% for validation, and 15% for testing, preserving class proportions. Hyperparameter tuning was performed using Grid Search [22], optimizing metrics such as precision, recall, and F1-score for each algorithm. The best configurations identified were subsequently used in the aggregation process.

Model integration within the system (4.2), was carried out using the IMWV (Incremental Multi-Level Weighted Hard Voting) scheme. Under this mechanism, each model generates an individual prediction and contributes a weighted vote based on its historical performance on the selected metrics. The system consolidates these predictions through hard voting that considers both the binary decision and the assigned weight, resulting in a final evaluation that is more stable and representative than that produced by any single model independently. This architecture also facilitates early detection of inconsistencies, as any significant disagreement among models can be leveraged as an additional anomaly signal within the oracle architecture.

4.5. Evaluation

The resulting architecture was subjected to an experimental evaluation within a simulated blockchain environment (5.1).

The experimental architecture was defined and documented prior to the results phase to ensure full traceability and reproducibility of the study. For simulating the blockchain environment, Ganache was used, enabling the creation of controlled and replicable local nodes. The compilation, deployment, and verification of smart contracts were carried out using Truffle, while programmatic interaction with the network was performed through Web3.py.

All experiments were executed on a Windows 11 Pro operating system, running on an ASUS TUF F15 laptop equipped with a 12th-generation Intel Core i5 processor, 8 GB of DDR4 RAM, and an NVMe SSD for storage. The machine learning–based validation module was implemented in Python 3.X, leveraging specialized libraries such as Scikit-learn, Pandas, NumPy, and Matplotlib for data processing, model training, evaluation, and visualization.

This entire configuration was explicitly integrated into the methodology section in order to provide a clear reference framework that enables replication of the experiment under the same technical conditions.

To evaluate the effectiveness of the proposed architecture, an experiment consisting of 60 sequential tests was conducted, each representing a complete validation cycle. Each cycle included the following stages:

- Reception of external data from the simulated oracle.
- Validation through the ML + IMWV system, where individual models perform classification and the weighted hard voting mechanism determines the final decision.
- Transmission of the validated data to the smart contract deployed on the simulated local network.
- Automatic logging of the execution status, labeled as success or failure according to system behavior.

All executions were recorded and documented, enabling precise identification of the events that triggered the early failures observed during the initial tests. These failures were associated with synchronization conditions and the integrity hash validation process. The consistency of the environment and the experimental configuration made it possible to reproduce these events in a controlled manner, thereby confirming the stability of the system once the initial adjustment phase was completed.

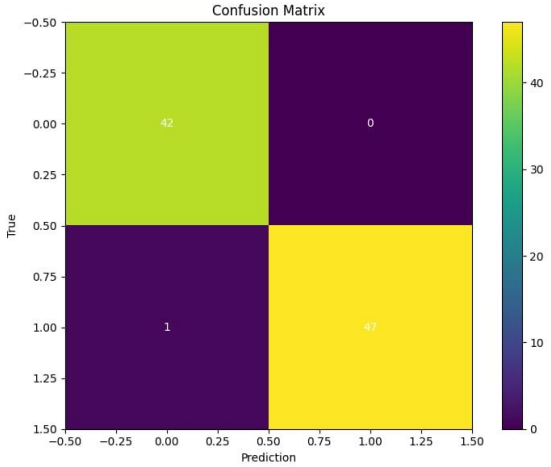

Fig. 3. *Confusion matrix of actual and predicted labels of the proposed approach.* ***Source:*** *Authors*

Figure 3 presents the confusion matrix corresponding to the performance of the implemented classifier. This representation shows that the model correctly identified 42 negative instances (TN) and 47 positive instances (TP). The absence of false positives (FP = 0) indicates that the system did not produce incorrect positive predictions, which translates into perfect specificity. On the other hand, only one false negative (FN = 1) was recorded, indicating a minimal margin of error in identifying the positive class.

This distribution demonstrates that the model exhibits outstanding capability in distinguishing between valid and invalid data, maintaining an appropriate balance between sensitivity and specificity. The complete elimination of false positives is particularly relevant in data validation systems; additionally, the very limited presence of false negatives suggests high sensitivity, ensuring that most valid data are correctly identified. These results are reflected in an overall accuracy of 1.00, evidencing the classifier's excellent performance in this scenario.

Figure 4 shows the ROC curve of the model along with its corresponding AUC value. The curve rises almost vertically toward the upper-left corner of the plot, describing the characteristic behavior of a

highly discriminative classifier. The area under the curve (AUC = 1.00) confirms that the model achieves complete separation between positive and negative instances.

An AUC equal to 1.00 implies that there exists at least one decision threshold at which the true positive rate (TPR) reaches its maximum possible value while the false positive rate (FPR) remains at zero. This means that the model assigns higher probabilities to positive cases than to negative ones in 100% of the possible comparisons.

Such behavior evidences an ideal discriminative capacity, confirming that the decision boundaries generated by the classifier fully separate both classes without overlap, demonstrating an accurate and effective understanding of the data**.**

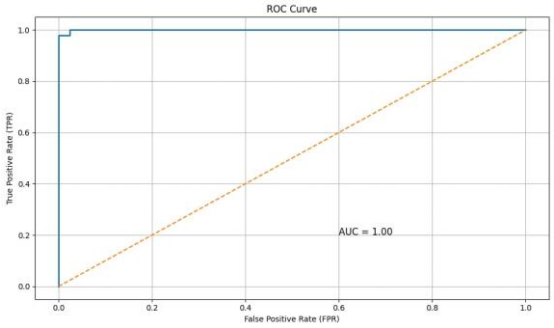

Fig. 4. *True Positive Rate of the Model*
Source: *Authors*

4.6. Deployment

Finally, the system was deployed in a production-like environment and tested with real data in a local blockchain network. This step validated the scalability and adaptability of the system to new contexts, ensuring that the architecture can be reused in other contractual domains requiring automated validation of external data.

5. RESULTS

To validate the effectiveness and stability of the proposed approach, experimental architecture simulating a smart contract operating environment with data validation through distributed oracles was designed and executed. This architecture underwent a testing protocol consisting of 60 sequential, controlled, and monitored executions to evaluate its performance and ability to handle potential failures, as shown in Figure 5.

Fig. 5*. Evaluation process.* ***Source:*** *Authors.*

5.1. Architecture Configuration

The proposed architecture consists of the following functional components:

- Main smart contract: Responsible for receiving and storing data from oracles.
- External oracles: Provide input based on simulated or real data regarding agricultural conditions such as temperature, humidity, and precipitation.
- Majority voting module (hard voting): Consolidates decisions from multiple oracles on the validity of a data set.
- Logging and monitoring subsystem: Records the result of each execution in real time, labeling events as “Successful” or “Failed”.

5.2. Execution Sequence

Each test consisted of simulating the transmission of data from multiple oracles to the smart contract. The complete sequence included 60 consecutive runs, numbered from 1 to 60. The results of each execution were classified as:

- Success (1): When data were correctly validated and the smart contract accepted and recorded the event.
- Failure (0): When an error occurs in validation, synchronization, or data transmission.
- Latency evaluation in 60 executions
- Average model runtime evaluation

5.3. Behavior Visualization

To illustrate the system’s behavior over the 60 executions, a scatter plot was generated where the horizontal axis represents the execution number and the vertical axis represents the result status (0 for failures and 1 for successes). Two critical zones were highlighted:

- Initial calibration zone: Corresponding to executions 1 to 10, where some instability was expected due to initial adjustments and system load. This area was highlighted in yellow.
- Error zone: Representing failure events (result = 0), shaded in pink to indicate unsuccessful transactions.s.

5.4. Behavior Analysis

During the calibration phase, two failures were recorded in executions 10 and 12. These early failures were linked to synchronization conditions in the oracle network and the response time for validating the integrity hash. From execution 13 onwards, no further failures were observed, indicating that the environment and architecture stabilized after the initial adjustments.

5.5. Results Overview

The evaluation procedure achieved a 96.66% success rate (58 out of 60 executions), with only two failures occurring during the early calibration phase (Figure 6).

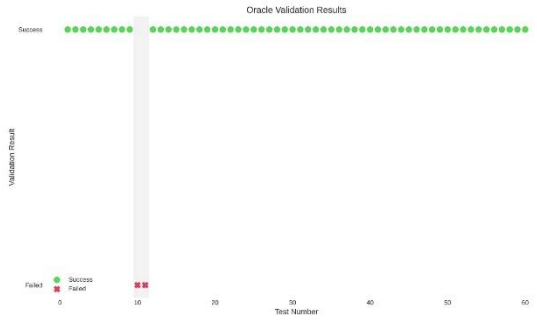

Fig. 6*. Test execution using the proposed architecture.*
Source: *Authors.*

The evaluation procedure of the proposed approach was carried out through 60 consecutive tests on the designed architecture, using Ganache as a local simulation environment, Truffle for smart contract management and deployment, and Web3.py for programmatic interaction with the blockchain network. In each attempt, both the test number and its result (success or failure) were recorded to validate the system's behavior against simulated data integrity events.

The system achieved a 96.66% success rate (58 out of 60), with only two failures detected in tests 10 and 12, both occurring during the initial phase of the experiment. Visual segmentation facilitated the interpretation of the system's performance, demonstrating that incorporating calibration phases and continuous monitoring was key to detecting and correcting early errors.

Overall, the results showed that the proposed architecture maintained stable and reliable behavior, reinforcing its applicability in contexts requiring automated data validation over blockchain infrastructure.

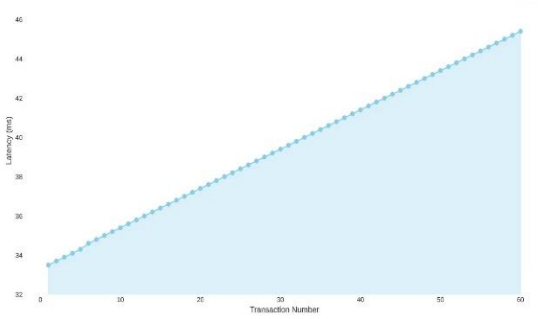

Fig. 7. *Transaction latency analysis.*
Source: *Authors.*

In Figure 7, the behavior of latency during the execution of 60 consecutive transactions can be observed. The test was performed in a controlled environment without parallel high-consumption processes, ensuring minimal system load during the evaluation.

The graph clearly illustrates an upward trend in latency measured in milliseconds (ms), with values initially ranging from 33.5 ms and reaching up to 46.0 ms by the end of the transaction set. The curve shows a stepped, progressive evolution, suggesting that the system's response times increase cumulatively as the transactional load advances.

This pattern may be attributed to various internal factors of the testing environment, such as progressive saturation of network buffers, increasing system resource consumption, or the management of internal queues in the oracle logic or the smart contract associated with the experiment.

Moreover, the data represented as orange points reinforce the experiment's consistency, indicating no losses, outliers, or abrupt performance drops, which is a positive sign for scenarios requiring stability, such as smart contracts in financial applications.

Overall, the observed behavior can be considered acceptable within a local simulation environment but also highlights the need for additional testing on distributed networks (testnet/mainnet) to validate the impact of decentralized network infrastructure and the number of nodes on overall latency.

In Figure 8, the relationship between the number of oracles used as input samples and the average training time of the IMWV model—designed and implemented to validate the integrity of data sent to the network in the context of smart contracts [20]—is shown.

The experiment considers five input configurations, with sample sizes equivalent to 2, 5, 10, 15, and 20 oracles, and demonstrates how the computational time required to adjust the model varies as the amount of data received from the oracles increases.

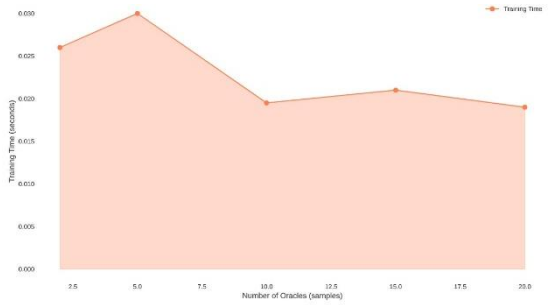

Fig. 8*. Average model training time.*
Source: *Authors.*

The obtained values show an initial fluctuation, peaking at 0.030 seconds when 5 oracles are used, and then progressively decreasing until stabilizing around 0.0195 seconds for configurations ranging from 10 to 20 oracles.

This behavior suggests that the model exhibits efficient adaptability beyond a certain input threshold. Initially, the computational cost may be influenced by overfitting on small samples or the need to readjust voting weights. However, as the number of oracles increases, training becomes more stable, with times close to 20 milliseconds, which represents a significant advantage for real-time systems requiring data validation without sacrificing performance.

These results confirm the feasibility of the proposed approach in distributed scenarios, where the model can be dynamically trained with multiple input sources without representing a considerable computational load.

6. CONCLUSIONS

The results obtained from the experimental process allow us to conclude that the methodological proposal designed to validate data from oracles demonstrates highly efficient computational performance. In particular, it was shown that despite the progressive increase in the number of oracles used to send data to the network, both latency and average training time remained within acceptable and stable ranges.

The latency test over 60 transactions revealed a growing but controlled pattern, with times ranging between 33 and 46 milliseconds, indicating that the system can sustain continuous operations without significantly affecting performance. This trend suggests that the flow of external data to smart contracts is processed consistently, even under moderate load scenarios.

On the other hand, the analysis of training time as a function of the number of oracles demonstrated that the proposal can scale without incurring critical computational penalties. In configurations ranging from 2 to 20 oracles, processing time did not exceed 30 milliseconds, remaining stable around 20 milliseconds beyond a certain threshold. This confirms that the system is suitable for implementation in environments requiring near real-time processing, such as agricultural applications, supply chains, or early warning systems.

The overall results indicate that the proposed architecture adapts well to increasing volumes of data from multiple oracles without compromising the system's operational efficiency. This feature is particularly valuable in contexts where data validation must be performed in a decentralized, reliable manner within reduced timeframes.

For future work, it is proposed to validate the architecture on real distributed blockchain networks, incorporate data from physical sensors or IoT devices, and conduct stress tests simulating critical scenarios with multiple simultaneous oracles. Additionally, optimization techniques for input flow will be explored to reduce latency, along with deployment strategies for resource-constrained devices, enabling implementation in edge environments. Finally, the integration of this solution with self-executing smart contracts is planned, aiming to automate processes based on pre-validated data.

Authors' Contributions:

CO: Conceptualization, Methodology, Validation, Writing – original draft, Investigation, Resources, Visualization. **HO:** Funding acquisition, Investigation, Methodology, Project administration, Resources, Validation, Writing – review and editing. **JSGS:** Investigation, Methodology, Validation, Writing – review and editing.

Funding: This work was carried out within the framework of the research project "Increasing the Supply of Pre-Commercial Technological Prototypes Derived from R&D Results to Strengthen the Agricultural Sector in the

Department of Cauca" (BPIN code 2020000100098), funded by the General System of Royalties (SGR) of Colombia.

Acknowledgments: The authors would like to thank the GTI Research Group and the GREDIS Research Group for the time, guidance, and support provided during the development of this research.

REFERENCES

[1] C. Contini, F. Boncinelli, G. Piracci, G. Scozzafava, and L. Casini, "Can blockchain technology strengthen consumer preferences for credence attributes?," *Agric. Food Econ.*, vol. 11, no. 1, 2023, doi: 10.1186/s40100-023-00270-x.

[2] M. Enayati *et al.*, "Blockchain-Based Location Sharing in 5G Open RAN Infrastructure for Sustainable Communities," *Lect. Notes Networks Syst.*, vol. 333, pp. 571 – 585, 2022, doi: 10.1007/978-981-16-6309-3_54.

[3] C. C. Ordoñez, M. M. Organero, G. Ramirez-Gonzalez, and J. C. Corrales, "Smart Contracts as a Tool to Support the Challenges of Buying and Selling Coffee Futures Contracts in Colombia," *Agric.*, vol. 14, no. 6, pp. 1–19, 2024, doi: 10.3390/agriculture14060845.

[4] S. P. Tan *et al.*, "A review on post-COVID-19 impacts and opportunities of agri-food supply chain in Malaysia," *PeerJ*, vol. 11, 2023, doi: 10.7717/peerj.15228.

[5] K. Almi'ani, Y. C. Lee, T. Alrawashdeh, and A. Pasdar, "Graph-Based Profiling of Blockchain Oracles," *IEEE Access*, vol. 11, no. March, pp. 24995–25007, 2023, doi: 10.1109/ACCESS.2023.3254535.

[6] G. Caldarelli, "Before Ethereum. The Origin and Evolution of Blockchain Oracles," *IEEE Access*, vol. 11, no. April, pp. 50899–50917, 2023, doi: 10.1109/ACCESS.2023.3279106.

[7] A. Beniiche, "A Study of Blockchain Oracles," pp. 1–9, 2020, [Online]. Available: http://arxiv.org/abs/2004.07140.

[8] S. Ellis, A. Juels, and S. Nazarov, "ChainLink: A Decentralized Oracle Network," 2017. [Online]. Available: https://link.smartcontract.com/whitepaper.

[9] Valencia-Payan, "Smart Contract to Traceability of Food Social Selling," *Comput. Mater. Contin.*, vol. 74, no. 3, pp. 4703 – 4728, 2023, doi: 10.32604/cmc.2023.031554.

[10] M. Sharifi, T. Khatibi, M. H. Emamian, S. Sadat, H. Hashemi, and A. Fotouhi, "Development of glaucoma predictive model and risk factors assessment based on supervised models," *BioData Min.*, vol. 14, no. 1, 2021, doi: 10.1186/s13040-021-00281-8.

[11] N. Peppes, E. Daskalakis, T. Alexakis, E. Adamopoulou, and K. Demestichas, "Performance of Machine Learning-Based Multi-Model Voting Ensemble Methods for Network Threat Detection in Agriculture 4.0," *Sensors*, vol. 21, no. 22, 2021, doi: 10.3390/s21227475.

[12] H. Zhang, S. Li, H. Bao, S. Wu, and J. Li, "A Trust-Aware and Cost-Optimized Blockchain Oracle Selection Model with Deep Reinforcement Learning." 2025, doi: 10.48550/arXiv.2502.16133.

[13] M. Di Gennaro, L. Italiano, G. Meroni, and G. Quattrocchi, "DeepThought: A Reputation and Voting-Based Blockchain Oracle," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 13740 LNCS, pp. 369–383, 2022, doi: 10.1007/978-3-031-20984-0_26.

[14] P. Liu, Y. Xian, C. Yao, P. Wang, L. Wang, and X. Li, "A Trustworthy and Consistent Blockchain Oracle Scheme for Industrial Internet of Things," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 5, pp. 5135–5148, 2024, doi: 10.1109/TNSM.2024.3399837.

[15] G. Palaiokrassas, S. Bouraga, and L. Tassiulas, "Machine Learning on Blockchain Data: A Systematic Mapping Study," *Elsevier BV*, 2024, [Online]. Available: http://arxiv.org/abs/2403.17081.

[16] S. Woo, J. Song, and S. Park, "A distributed oracle using intel SGX for blockchain-based IoT applications," *Sensors (Switzerland)*, vol. 20, no. 9, 2020, doi: 10.3390/s20092725.

[17] H. Taherdoost, "Blockchain and Machine Learning: A Critical Review on Security," *Information*, vol. 14, no. 5, 2023, doi: 10.3390/info14050295.

[18] C. Connors and D. Sarkar, "Survey of prominent blockchain development platforms," *J. Netw. Comput. Appl.*, vol. 216, p. 103650, 2023, doi:

https://doi.org/10.1016/j.jnca.2023.103650.

[19] H. Xiong, M. Chen, C. Wu, Y. Zhao, and W. Yi, "Research on Progress of Blockchain Consensus Algorithm: A Review on Recent Progress of Blockchain Consensus Algorithms," *Futur. Internet*, vol. 14, no. 2, 2022, doi: 10.3390/fi14020047.

[20] C. C. Ordoñez, G. Ramirez-Gonzalez, and J. C. Corrales, "Enhancing Data Integrity in Blockchain Oracles Through Multi-Label Analysis," *Appl. Sci.*, vol. 15, no. 5, 2025, doi: 10.3390/app15052379.

[21] C. De and G. Barrios-puente, "Cobertura de precios para el café, utilizando el mercado de futuro," *Rev. Mex. Ciencias Agrícolas*, vol. 13, no. 6, pp. 1147–1154, 2022.

[22] B. H. Shekar and G. Dagnew, "Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, 2019, pp. 1–8, doi: 10.1109/ICACCP.2019.8882943.