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

ORIGINAL

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

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

Cristian Camilo Ordoñez Quintero[1]
Hugo Armando Ordoñez Erazo[2]
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**
[3] **Universidad Pedagógica y Tecnológica de Colombia, Grupo de Investigación GIMI, Tunja, Boyacá, Colombia.**

Received: 25 July 2025. Revised: 25 September 2025. Accepted: 20 December 2025. Published: 01 January 2026.
Corresponding Author: Cristian Camilo Ordoñez Quintero

How to cite: C. C. Ordoñez Quintero, H. A. Ordoñez Erazo, and J. S. González Sanabria, "Arquitectura de oráculo blockchain con validación de datos basada en procesos de aprendizaje de maquina", *Revista Colombiana de Tecnologías de Avanzada (RCTA)*, vol. 1, no. 47, ene. 2026, art. no. 4103, doi: 10.24054/rcta.v1i47.4103

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

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.

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 participate collaboratively to record, verify, and store transactions immutably [2]. This property has given rise to smart contracts, computer programs that run automatically once certain previously established conditions are met [3]. These contracts have gained relevance across a broad spectrum of applications, from decentralized finance (DeFi) systems 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 the world outside the blockchain[5]. This is because the blockchain is designed as a closed and deterministic environment, in which all operations must be identically reproducible by all nodes. Consequently, smart contracts depend on intermediaries called oracles, responsible for supplying the external information required for their execution[5]. This information may include asset prices, event outcomes, weather conditions, or any other data not originating directly within the blockchain network.

Oracles are therefore a critical component of the blockchain ecosystem, acting as bridges between the off-chain and on-chain worlds[6]. Yet this same function makes them one of the most vulnerable links in the system. The blockchain is immutable and resistant to tampering thanks to consensus mechanisms such as Proof of Work or Proof of Stake, but 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 high-criticality environments[5].

This phenomenon, known as the “oracle problem”, has been identified as one of the main barriers to the mass adoption of blockchain solutions in contexts where data integrity is essential [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, and delegate this responsibility to external architectures that are poorly auditable or centralized.

Against this backdrop, new approaches are needed that allow explicit and replicable validations to be incorporated into the data entering the blockchain through oracles[6]. In particular, the field of Data Science and machine learning offers powerful tools for data analysis, anomaly detection and prediction based on historical behavior[9].

This article therefore proposes a blockchain oracle architecture with a data validation system based on machine learning processes. The proposal aims to improve the integrity of the external data used in smart contracts. To this end, a processing chain is implemented that includes supervised and unsupervised models, aimed at anomaly detection in the data sent by the oracle. This validation flow is structured according to the phases of the CRISP-DM (Cross-Industry Standard Process for Data Mining) model, which guarantees a systematic, replicable methodology aligned with data engineering best practices[10].

The architecture also includes a decision component based on a majority voting mechanism (hard voting)[11]. This approach allows not only anomalous or inconsistent data to be rejected, but also objective acceptability thresholds to be established that can be audited and traced in real time. In functional terms, 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 strengthens the security and reliability of the system.

This article is organized into the following sections. Section 2 presents the key concepts needed to contextualize the research problem, along with a brief but substantial review of the state of the art focused on oracles, smart contracts and data validation in blockchain environments. Section 3 then presents the methods used to develop the research, detailing the technical approach adopted, the data processing flow and the proposed architecture. Section 4 describes the implementation of the proposal and presents the results obtained after its implementation. Finally, Section 5 presents the conclusions of the work and outlines possible lines of future research.

RELATED WORKS

To identify the most relevant literature of the last five years, a systematic search was carried out in the Scopus and Web of Science (WOS) databases, using specific strings 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, showing how this integration has given rise to advanced architectures aimed at guaranteeing the integrity, reliability and traceability of off-chain data before their incorporation into smart contracts.

To introduce the topic, the authors[12]create a relevant approach: the TCO-DRL (Trust-Aware and Cost-Optimized Reinforcement Learning) model, which uses deep reinforcement learning techniques to dynamically select oracle nodes based on multiple reputation and cost dimensions. This model reduces assignments to malicious nodes and optimizes costs

without compromising data quality.

The DeepThought platform, proposed in 2022, incorporates a hybrid mechanism of verified voting and reputation, in which users collaborate to validate information and results are combined through a distributed reputation scheme that improves resistance to human corruption[13].

Studies such as the one on blockchain for IIoT are also considered; they present an oracle node selection scheme based on Verifiable Random Functions (VRF) and reputation, together 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 domain of ML applied to blockchain, a recent systematic mapping analyzes 159 articles related to anomalies, classification and the use of on-chain data, identifying a growing trend in anomaly detection, standardization and scalability challenges, as well as limited interoperability between chains[14].

Authors such as[15]have identified that much of the work focuses on anomaly detection or atypical patterns in blockchain data, using supervised and unsupervised techniques, which is essential for data validation before use within an oracle system.

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

Frameworks such as Oraichain also integrate AI-centric oracles that run AI off-chain and validate the models through verifiable execution tests; they inspire new trust layers to feed smart contracts with ML-processed data that have been previously verified[7].

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

PROPOSAL

The proposed architecture of a blockchain oracle with data validation based on machine learning techniques is grounded in an exhaustive review of recent literature, which supports the relevance and feasibility of each of its components; Figure 1 shows each of the components.

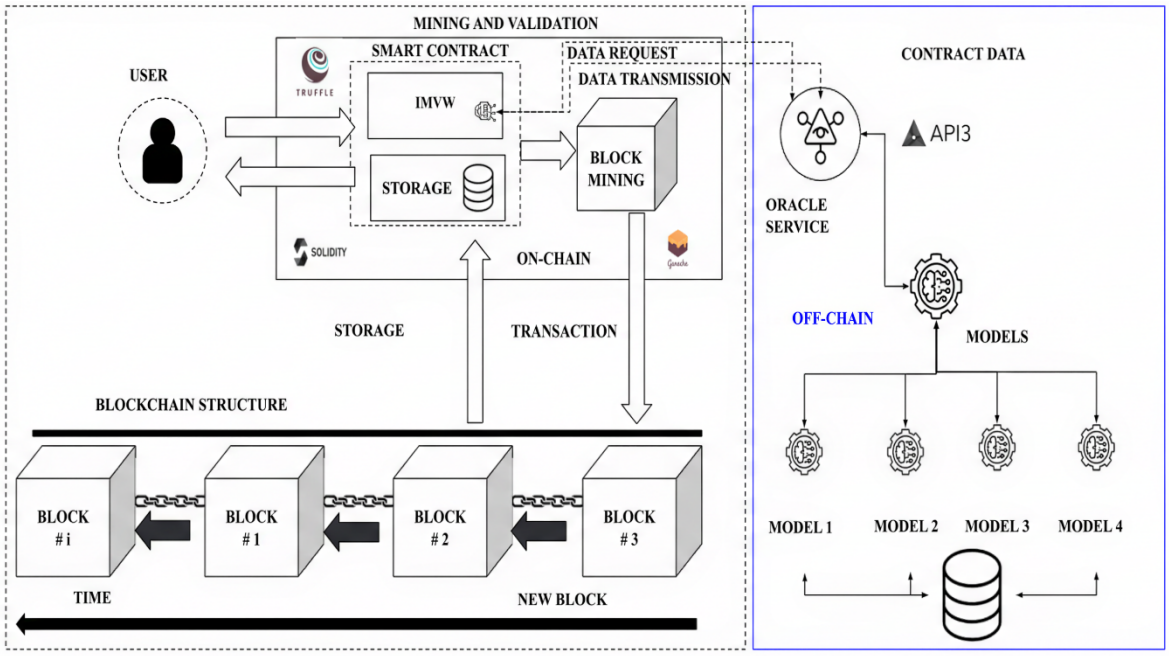

Fig. 1 Proposed architecture.
Source: own elaboration.

First, the smart contract is implemented in Solidity and tested with tools such as Truffle and Ganache, widely recognized in the literature as robust environments for the design, compilation, deployment and local verification of blockchain contracts[18].

This layer enables the automation of agreements through programmed conditions that require external inputs for their execution.

External data are acquired 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, which reduces the possibility of intermediate manipulation and improves the traceability of the source[19].

The proposal incorporates a validation module based on a multilevel weighted hard voting scheme (IMWV). It receives the predictions of multiple classification models and produces a consolidated decision, with weights assigned according to the historical performance of each model. This technique has proven to be highly effective in improving accuracy in multi-label classification and anomaly detection tasks on heterogeneous data[20].

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

The architecture also includes a continuous learning mechanism, through which the machine learning models are periodically updated from the data validated and incorporated into the chain. This feedback allows the models to adapt to new conditions and maintain a high level of predictive performance, in line with current trends in online learning and distributed federated learning.

IMPLEMENTATION

The proposal was constructed following the CRISP-DM (Cross Industry Standard Process for Data Mining) methodology, adapted to the context of data validation in blockchain oracles. The phases developed are detailed below, as guided in figure 2:

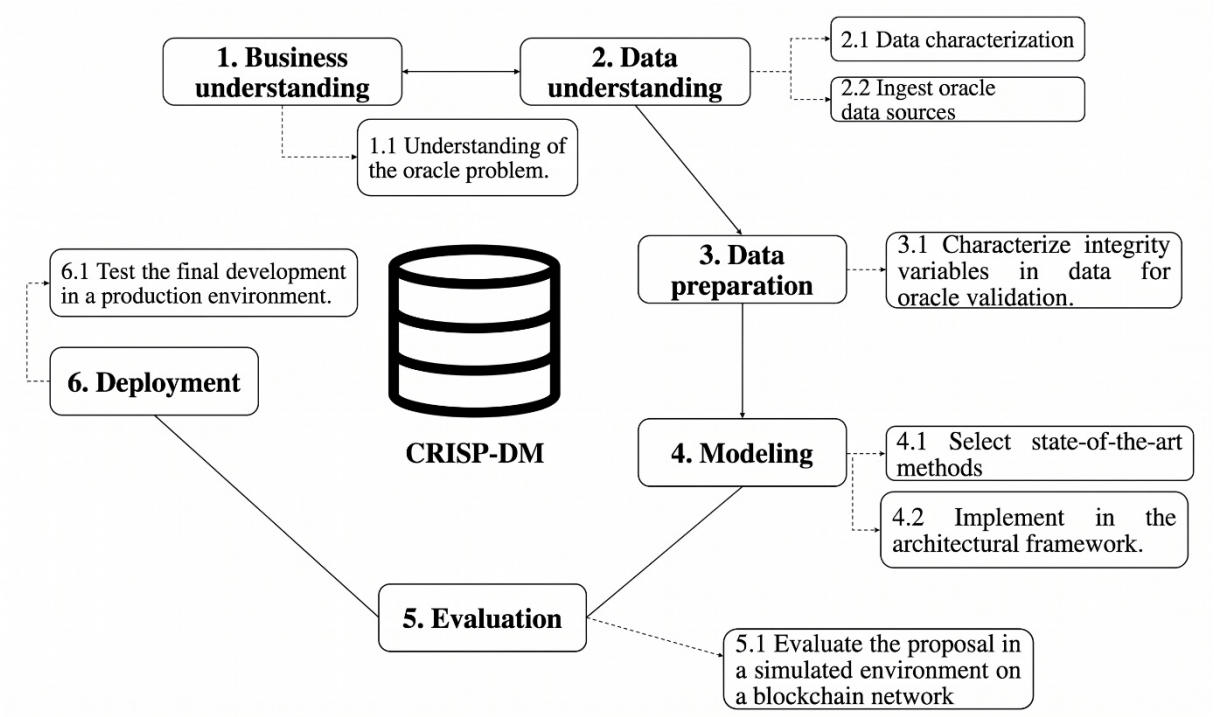

Fig. 2 Implementation of architecture and ml models for oracle data validation.
Source: own elaboration

Business understanding

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

Data

This study focuses on a highly volatile smart contract, an example being coffee futures contracts. The selected variables

temperature, relative humidity, altitude, rainfall, variety and coffee price were included because of their direct relevance to the data validation of external data used by smart contracts in agroclimatic domains of futures contracts[21]. These variables correspond to real sources that commonly feed weather and agricultural oracles, so their role within the architecture rests on the need to guarantee coherence, internal correlation and temporal consistency before their use within the smart contract is allowed.

Each variable fulfills a specific purpose within the data validation process:

• Temperatura, humedad y precipitaciones permiten detectar anomalías meteorológicas abruptas o valores imposibles.

• Altitud actúa como variable de control espacial en el análisis climático.

• Variedad (categoría agrícola) permite validar consistencia entre el tipo de cultivo y su respuesta climática.

Coffee price is incorporated as a variable that can be manipulated in financial oracles, being critical to avoid erroneous executions of futures contracts.

Subsequently, the available data were characterized (2.1), originating from environmental sensors, weather stations, commercial APIs and historical agricultural records. A data ingestion mechanism from external sources through oracles was also developed (2.2), connecting to services such as API3 to provide data in a decentralized and verifiable manner.

Data preparation

Once the nature of the sources was understood, the data preparation phase began (3.1), in which criteria were designed to identify data integrity attributes, such as internal consistency, timeliness, completeness and reliability of origin.

To build the training set for the models, a binary labeling process was implemented that classifies each event as valid or invalid. The label is assigned considering three criteria:

• Internal consistency: coherence between meteorological and agricultural variables.
• Temporal coherence: direct comparison with the historical behavior recorded using time series techniques.
• Threshold verification: ranges provided by official sources or external reference data (API3 and weather stations).
These attributes were used as inputs to train models that validate data quality before their incorporation into the blockchain.

Table 1: Oracle data labeling

Altitude	Temperature	Humidity	Precision	Precision by altitude	Consistency by temperature	Consistency by irrigation	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

Modeling

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

To ensure stability and comparability among the models, the dataset was split using a stratified partition of 70% for training, 15% for validation and 15% for testing, maintaining the proportion between classes. Hyperparameter tuning was carried out using Grid Search [22], optimizing metrics such as precision, recall and F1 score for each algorithm. The best configurations found were subsequently used in the aggregation process.

The integration of the models within the system was carried out through the IMWV (Incremental Multi-Level Weighted Hard Voting) scheme. In this mechanism, each model generates its individual prediction and contributes a weighted vote according to 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, allowing a final evaluation that is more stable and representative than that produced by any model independently. This architecture also facilitates the early detection of inconsistencies, since any significant disagreement among the models can be used as an additional signal of anomaly, also implemented within the oracle architecture.

Evaluation

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

The experimental architecture was defined and documented before the results phase, in order to guarantee full traceability and reproducibility of the study. Ganache was used to simulate the blockchain environment, allowing the creation of controlled and replicable local nodes. Smart contracts were compiled, deployed and verified using Truffle, while programmatic interaction with the network was carried out using Web3.py.

The tests were run on a Windows 11 Pro operating system, on an ASUS TUF F15 computer equipped with a 12th-generation Intel Core i5 processor, 8 GB of DDR4 RAM and an NVMe SSD storage drive. The machine learning-based validation module was implemented in Python 3.X, using specialized libraries such as Scikit-learn, Pandas, NumPy and Matplotlib for the processing, training, evaluation and visualization of the models.

All of this configuration was explicitly integrated into the methodological section with the aim of providing a clear reference framework that allows the experiment to be replicated under the same technical conditions.

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

- Reception of the external data from the simulated oracle.
- Validation by means of the ML + IMWV system, where the models perform the individual 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 recording of the execution status, labeled as success or failure according to the system's behavior.

All executions were recorded and documented, making it possible to precisely identify the events that originated the early failures observed in the initial tests, which were associated with synchronization conditions and with the integrity hash validation process. The consistency of the environment and of the experimental configuration made it possible to reproduce these events in a controlled manner, confirming the stability of the system once the initial adjustment phase had been overcome.

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

This distribution shows that the model is notably able to distinguish between valid and invalid data while maintaining an adequate balance between sensitivity and specificity. The complete elimination of false positives is especially relevant in data validation systems; in a complementary way, the low presence of false negatives suggests high sensitivity, ensuring that most valid data are identified as such. These results are reflected in an overall accuracy of 1.00, which shows the classifier's performance in this scenario.

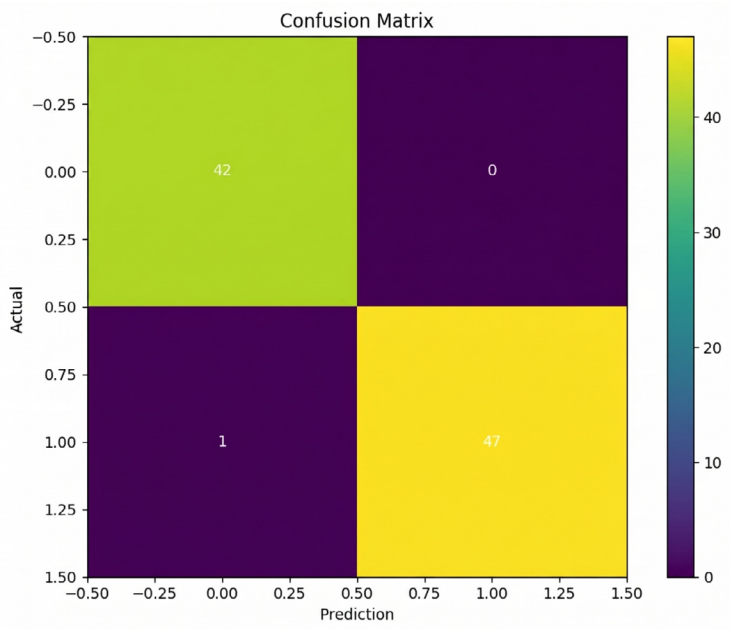

Fig. 3 Confusion matrix of actual and predicted labels of the proposal.
Source: own elaboration

Figure 4 shows the ROC curve of the model together with its corresponding AUC value. The curve rises almost vertically until it reaches 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 a separation between positive and negative instances.

An AUC equal to 1.00 implies that there is at least one decision threshold at which the true positive rate (TPR) reaches the 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 shows an ideal discriminative capacity, which confirms that the decision boundaries generated by the classifier completely separate both classes without overlap, interpreting the data adequately.

Deployment

Finally, work advanced on deploying the system in a production environment (6.1), testing the solution with real data on a local blockchain network. This step made it possible to validate the scalability and adaptability of the system to new contexts, ensuring that the architecture can be reused in other contractual domains that require automated validation of external data.

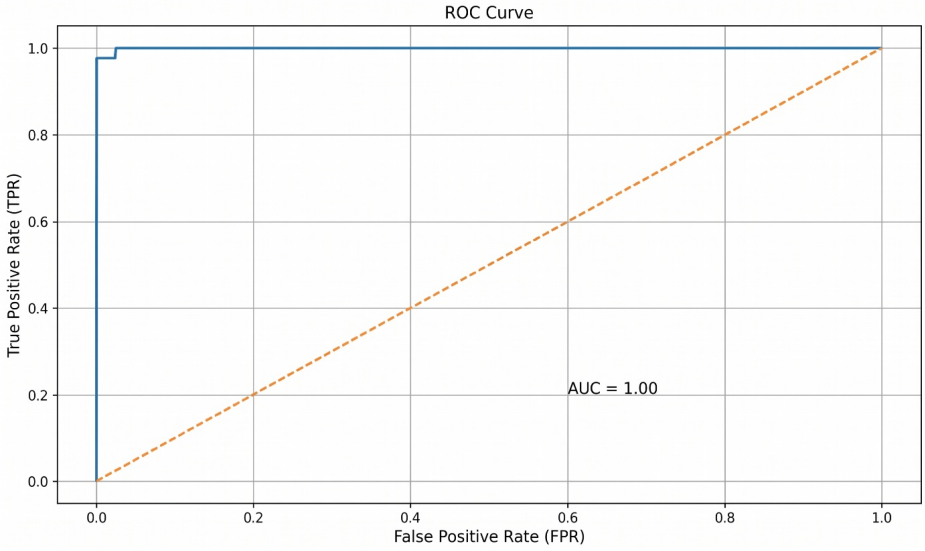

Fig. 4 True positive rate of the model
Source: own elaboration

RESULTS

To validate the effectiveness and stability of the proposed approach, an experimental architecture was designed and executed that simulates a smart contract operating environment in which data submission validation processes are applied through distributed oracles. This architecture was subjected to a test protocol based on 60 sequential, controlled and monitored executions, in order to verify its performance and its capacity to respond to possible failures, as observed in the following image.

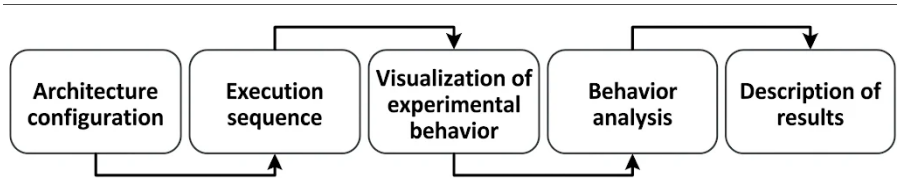

Fig. 4 True positive rate of the model
Source: own elaboration

Architecture configuration

The proposed architecture is composed of the following functional components:

- Main smart contract, responsible for receiving and storing data from oracles.
- External oracles, which provide inputs from simulated or real data on agricultural conditions (such as temperature, humidity,

and precipitation).
- Majority voting module (hard voting), which consolidates the decisions of multiple oracles on the validity of a data set.
- Logging and monitoring subsystem, which records the result of each execution in real time, labeling the events as "Successful" or "Failed".

Sequence of executions

Each execution consists of a simulation of data input from multiple oracles to the smart contract. The complete sequence is composed of 60 consecutive executions numbered from 1 to 60. The result of each execution is classified binarily as:

- Success(1): When the data are correctly validated and the smart contract responds with the acceptance and recording of the event.
- Failure (0): When an error occurs in the validation, synchronization, or transmission of data.
- Latency evaluation in 60 executions
- Average time evaluation of models

Behavior visualization

To illustrate the behavior of the system during the 60 executions, a scatter plot was constructed, where the horizontal axis represents the execution number and the vertical axis represents the status of the result (0 for failures and 1 for successes). At the visual level, two key zones were implemented:

- Initial calibration zone: Corresponds to executions 1 to 10, where a possible instability of the system was expected due to adjustments and initial load. This zone was highlighted in yellow to facilitate its identification.
- Error zone: Represents the failure area (result equal to 0) and is shaded in pink to visually indicate the occurrence of unsuccessful events.

Behavior analysis

During the calibration phase, two failures were recorded in executions 10 and 12. These early failures were analyzed and associated with synchronization conditions in the oracle network and with response time in the integrity hash validation. From execution number 13 through 60, no further failures occurred, which shows that the environment and architecture conditions stabilized correctly after the initial adjustments.

Description of results

This procedure allowed the reliability of the proposed approach to be evaluated in a controlled manner, highlighting the following, as observed in Figure 6:

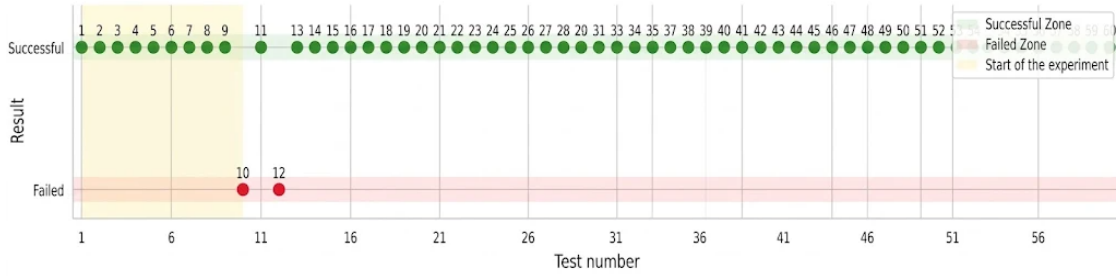

Fig. 6 Execution of tests using the proposed architecture

The evaluation procedure for the proposed approach was carried out by running 60 consecutive tests on the designed architecture, using Ganache as the local simulation environment, Truffle for the management and deployment of smart contracts, 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, with the purpose of validating the system's behavior against simulated data integrity events.

The system achieved a success rate of 96.66% (58 of 60), with only two failures detected in tests 10 and 12, both occurring in the initial phase of the experiment. The visual segmentation facilitated the interpretation of the system's performance, showing that the incorporation of calibration and continuous monitoring phases was key to detecting and correcting early errors. Overall, the results showed that the proposed architecture maintained stable and reliable behavior, reaffirming its applicability in contexts that require automated data validation on blockchain infrastructure.

Figure 7 shows the latency behavior during the execution of 60 consecutive transactions. The test was run in a controlled environment without high-consumption parallel processes, which guarantees a minimal system load during the evaluation.

The graph clearly illustrates an upward trend in latency measured in milliseconds (ms), with values that initially range from 33.5 ms and reach up to 46.0 ms at the end of the transaction set. The curve shows a stepped and progressive evolution, which

suggests that the system's response times increase cumulatively as the transactional load advances.

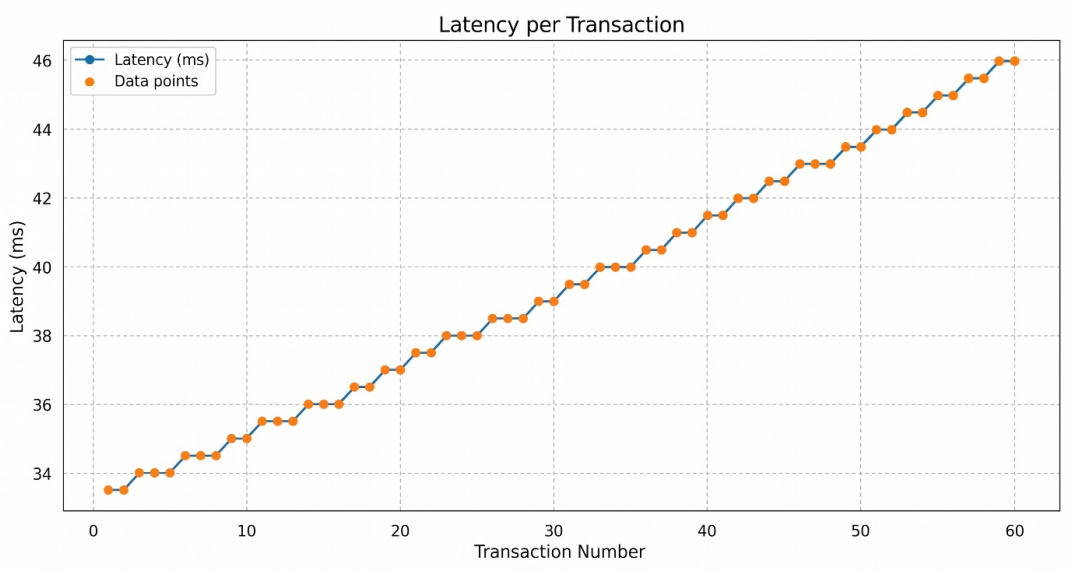

Fig. 7 Latency analysis per transaction
Source: own elaboration

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

In addition, the data represented as orange dots reinforce the consistency of the experiment, indicating that there were no losses, outliers, or abrupt performance drops, which is positive for scenarios that require stability, such as smart contracts in financial applications.

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

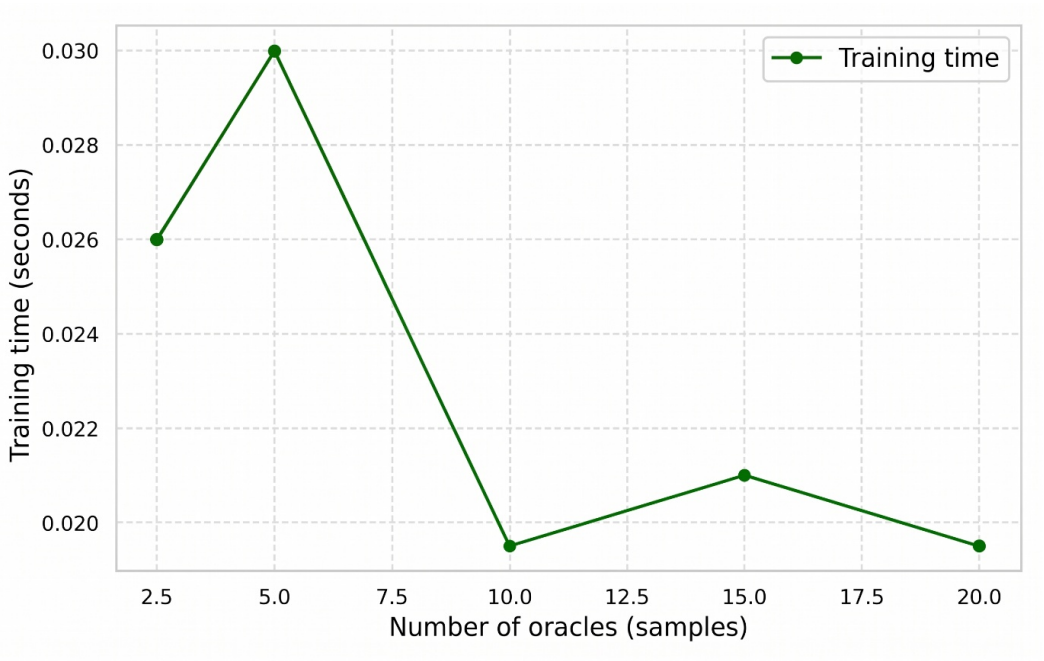

Fig. 8 Average model training time
Source: own elaboration

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

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

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

This behavior suggests that the model presents efficient adaptability from a certain input threshold onward. Initially, the computational cost may be affected by overfitting on small samples or by the need to readjust the 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 that require data validation without sacrificing performance.

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

CONCLUSIONS

The results obtained in the experimental process allow us to conclude that the methodological proposal designed to validate data from oracles shows highly efficient behavior from the computational point of view. In particular, it was evidenced 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 on 60 transactions revealed a growing but controlled pattern, with times ranging between 33 and 46 milliseconds, which indicates 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 showed that the proposal can scale without incurring critical computational penalties. In configurations ranging from 2.5 to 20 oracles, the processing time did not exceed 30 milliseconds, remaining stable at around 20 milliseconds from a certain threshold onward. This confirms that the system is suitable for implementation in environments that require near-real-time processing, such as agricultural applications, supply chains or early warning systems.

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.

ACKNOWLEDGMENTS

The authors thank the GTI research group and the GREDIS Research Group for the time, guidance, and support provided during the development of this research.

FUNDING

This work was carried out within the framework of the research project "Incremento de la Oferta de Prototipos Tecnológicos en Estado Pre-Comercial Derivados de Resultados de I+D para el Fortalecimiento del Sector Agropecuario en el Departamento del Cauca" (code BPIN 2020000100098), funded by the General Royalties System (SGR) of Colombia.

CONFLICT OF INTEREST

None

AUTHORSHIP 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.