International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.4 (2025) pp. 7571-7586
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Blockchain-Integrated Support Vector Machine Framework for Privacy-Preserving Predictive Analytics

Yojana[1]*, Yogesh Chaba[2]

[1]Research Scholar, Dept. of CSE, GJUS&T, Hisar, India
* **Corresponding Author Email:** yojana@jcboseust.ac.in - **ORCID:** 0000-0002-5247-0850

[2]Professor, Dept. of CSE, GJUS&T, Hisar, India
Email: yogeshchaba@yahoo.com - **ORCID:** 0000-0002-5247-0050

Article Info:

DOI: 10.22399/ijcesen.4055
Received : 01 August 2025
Accepted : 30 August 2025

Keywords

Machine Learning,
Blockchain Integration,
Privacy-Preserving Algorithms,
Sentiment Analysis,
Data Security.

Abstract:

This paper presents a blockchain-integrated Support Vector Machine (SVM) framework that addresses the privacy, trust, and ownership limitations of conventional client–server architectures. In the proposed design, clients train local SVM models and share only encrypted parameters, while blockchain ensures secure aggregation, tamper-proof logging, and transparent auditability.This study proposes a framework that combines blockchain technology with Support Vector Machines (SVM) to improve privacy, transparency, and accountability in predictive analytics. Experiments on MNIST and CIFAR-10 show that the framework achieves accuracy comparable to conventional SVMs (0.92 vs. 0.93), while enabling tamper-resistant predictions and verifiable model operations. Blockchain integration introduces moderate overhead—latency of 18–19 ms and transaction costs around 22,500 gas units—but these are outweighed by gains in data security, decentralized control, and auditability. To the best of our knowledge, this is among the first efforts to merge SVM with blockchain for secure predictive modeling. The framework is scalable, reliable, and well-suited for sensitive domains such as healthcare, finance, and intelligent transportation systems.

1. Introduction

Machine Learning (ML) has emerged as a cornerstone of modern intelligent systems, empowering organizations to derive actionable insights and make data-driven decisions from increasingly complex and diverse datasets [1], [2]. Its applications span a wide spectrum of critical domains, including healthcare, finance, transportation, cybersecurity, and social media—areas where prediction accuracy and response timeliness hold substantial operational and societal significance [3].Among the family of supervised learning algorithms, the Support Vector Machine (SVM) remains one of the most reliable and versatile methods. It excels at identifying optimal decision boundaries that separate different classes while preserving strong generalization, even in high-dimensional data spaces [4]. Owing to these properties, SVMs have gained widespread use in diverse applications such as sentiment analysis, text categorization, bioinformatics, and traffic prediction [5], [6].However, when SVMs are deployed through conventional client–server architectures, a number of practical and security-related issues arise. In a centralized setup, training data must be transmitted to the main server, which raises the likelihood of privacy breaches or unauthorized access [7]. Additionally, because the server is responsible for storing and managing the trained model, any compromise at that level can lead to manipulation or tampering of the model parameters. The dependence on a single server also introduces fragility into the system—any malfunction, outage, or attack can disrupt the entire operation [8].Researchers have explored several privacy-preserving learning techniques to alleviate these risks, including federated learning and secure multiparty computation [9], [10]. While these frameworks reduce direct data exposure, they often fail to guarantee full transparency and verifiable accountability—features that are essential in sensitive domains such as finance, transportation, and healthcare [11].To bridge these gaps, blockchain technology has emerged as a promising complementary approach. Owing to its

decentralized and tamper-resistant structure, blockchain can maintain trustworthy records of transactions and model updates [12], [13]. When combined with SVM, it enables the system to preserve predictive accuracy while ensuring that all updates are traceable, verifiable, and securely recorded, thereby strengthening both data ownership and model integrity [14], [15].Motivated by these challenges and opportunities, this study introduces a blockchain-assisted Support Vector Machine (SVM) framework designed to enhance privacy protection, ensure transparent accountability, and improve overall system resilience. The proposed approach seeks to combine the analytical strengths of SVM with the inherent trust and immutability offered by blockchain technology.The major contributions of this research can be outlined as follows:

- **Decentralized SVM architecture:** Each participating client independently trains a local model and transmits only encrypted model parameters to the shared network. This approach prevents raw data exposure and ensures that sensitive information remains securely stored within the local environment.

- **Blockchain-based transparency:** The framework leverages a tamper-proof distributed ledger to record and validate model updates. This mechanism provides an auditable trail of all transactions, ensuring that any modification or contribution can be traced and verified in a decentralized manner.

- **Comprehensive experimental evaluation:** The proposed blockchain-integrated SVM is systematically evaluated against a conventional client–server SVM using publicly available benchmark datasets. The analysis highlights trade-offs among model accuracy, computational efficiency, latency, and security enhancements, offering a balanced assessment of performance under real-world conditions.

This study is guided by the central research question:*I*n what ways can the integration of blockchain technology enhance privacy, security, and operational efficiency in machine learning-based traffic analysis when compared with traditional client–server SVM frameworks?The remainder of this paper is organized as follows: Section 2 reviews existing literature that explores the convergence of machine learning and blockchain-based systems. Section III explains the methodology and architecture of the proposed framework. Section IV outlines the experimental setup and implementation details, while Section V discusses the obtained results and their implications. Finally, Section VI concludes the paper with a summary of findings and potential directions for future research.

2. Literature Survey

This section reviews the progression of machine learning (ML) algorithms, with particular emphasis on the evolution of Support Vector Machines (SVM) and their adaptations for privacy-preserving and distributed environments, such as federated learning and blockchain-based frameworks.

Over the years, ML has advanced from simple statistical models to sophisticated algorithms capable of addressing complex, real-world challenges across diverse domains. Early milestones include Samuel's checkers-playing program [16] and Rosenblatt's Perceptron [17], which marked the foundation of supervised learning by showing that machines could learn from experience and improve performance over time. Despite their pioneering nature, these early models were limited by their inability to capture non-linear relationships, restricting their use in more intricate applications.A major breakthrough came with the introduction of the Support Vector Machine by Cortes and Vapnik [18]. The SVM framework optimized classification boundaries by maximizing the margin between classes, thereby enhancing generalization and robustness. By leveraging kernel functions, SVMs effectively handled non-linear transformations while maintaining computational efficiency. These properties made SVM particularly useful for high-dimensional datasets and scenarios with limited samples. Although alternative algorithms such as decision trees, random forests, and ensemble models emerged [19], SVM remained a preferred choice in tasks requiring precision, robustness, and strong generalization [20].In recent years, research has shifted toward privacy-preserving variants of SVM, driven by concerns over data confidentiality in centralized systems. Federated SVM models [21] represent one such approach, allowing distributed clients to train local models while sharing only aggregated or encrypted parameters instead of raw data. This decentralization protects privacy and mitigates risks of data exposure. Further enhancements have been achieved through cryptographic techniques, including homomorphic encryption and secure multiparty computation [22], [23], which enable computations to be performed directly on encrypted data while safeguarding sensitive information during both training and inference.Parallel to these advances, blockchain integration into ML

workflows has introduced new dimensions of trust, transparency, and auditability [24]. Blockchain's immutable and distributed ledger ensures that model updates and transactions are verifiable and resistant to tampering. This integration bridges the gap between decentralized computation and transparent governance, fostering secure and accountable machine learning ecosystems [25].

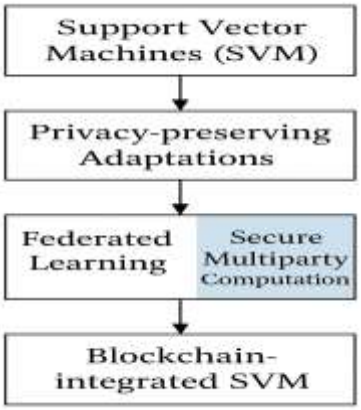


Figure 1: *Framework for Blockchain-Enhanced SVM*

Table 1 summarizes the chronological progression of key ML developments, tracing the path from early statistical models to privacy-preserving and blockchain-enabled SVM frameworks.

The progression from conventional SVM to blockchain-integrated SVM illustrates successive adaptations for privacy preservation, including federated learning and secure multiparty computation, as depicted in **Figure 1**.

In summary, the literature confirms that SVM is a robust and versatile algorithm. While federated learning and blockchain-based frameworks provide promising solutions for privacy, security, and auditability, there is still a lack of research evaluating their combined effectiveness in applications such as traffic analysis. This gap forms the motivation for the proposed blockchain-integrated SVM framework,

In conclusion, the literature establishes that SVM remains a highly robust and versatile algorithm. While federated learning and blockchain-based approaches offer promising solutions for privacy, security, and auditability, there remains a gap in evaluating their combined effectiveness in traffic analysis applications. This gap motivates the development of the proposed blockchain-integrated SVM framework, aiming to achieve secure, decentralized, and high-performance traffic prediction.designed to enable secure, decentralized, and high-performance predictive analytics.

3. Methodology

This research proposes a client–server architecture using Support Vector Machines (SVM) for analyzing social-based datasets while ensuring privacy preservation. The methodology encompasses multiple stages, ranging from data acquisition to secure model aggregation and evaluation, integrating cryptographic techniques and blockchain to enhance trust, transparency, and accountability [26].

3.1 Data Collection

The initial phase focuses on data collection, during which social media–based datasets are gathered to train and evaluate the SVM model. Social data includes user-generated content extracted from platforms such as Twitter, Facebook, and Reddit, consisting of textual inputs (posts, comments, or replies), categorical variables (user roles, topic categories), and numerical indicators (likes, shares, or retweets). These datasets are dynamic, heterogeneous, and unstructured, making them suitable for assessing the robustness of SVM models under real-world conditions [27].

The datasets used comprise thousands to millions of records to ensure statistical representativeness. Each record includes textual, numerical, and categorical features, paired with corresponding sentiment or polarity labels. As SVM is a supervised learning algorithm, all samples are annotated with ground-truth class labels to facilitate model training and validation. Data were obtained either directly from social platforms via APIs or from publicly available benchmark datasets [28], ensuring exposure to authentic social interactions and realistic distributions.

3.2 Data Preprocessing

Following acquisition, preprocessing is performed to convert raw, unstructured data into a format suitable for SVM training. Social datasets often contain missing values, duplicates, and irrelevant information that can reduce model accuracy. A systematic cleaning process removes such noise and standardizes the dataset [29].

Textual attributes undergo Natural Language Processing (NLP) operations such as tokenization, stopword removal, and lemmatization [30]. Categorical variables are transformed into numerical vectors using one-hot encoding, while text features are represented using TF-IDF or Word2Vec embeddings [31]. To balance contributions across variables, numerical features are normalized via min–max scaling or z-score standardization. The dataset is then divided into training, validation, and testing subsets (70:15:15)

for model training, hyperparameter tuning, and evaluation [32].

3.3 Client–Server SVM Architecture

The central innovation of this research is the design of a client–server SVM framework that balances scalability, efficiency, and data privacy. Clients independently train local SVMs without transferring raw data. Instead, they share only model-related parameters, such as support vectors or gradients. The central server aggregates these parameters via weighted averaging or secure multiparty computation (SMC) [33]. he updated global model is redistributed to all clients, allowing iterative refinement through multiple training rounds.

The central server then aggregates these contributions using a weighted averaging mechanism or secure multiparty computation (SMC) techniques to form a unified global model, as illustrated conceptually by Equation (1):

$$w_{global} = \frac{1}{N}\sum_{i=1}^{N} w_i$$

where $\boldsymbol{w_i}$ represents the local model parameters from the ith client, and $\boldsymbol{N}$ denotes the total number of participating clients. T

This process resembles federated learning principles but preserves SVM's structural efficiency, demonstrating that distributed training can maintain accuracy and confidentiality while mitigating risks of centralization [34].

3.4 Privacy-Preserving Enhancements

To reinforce privacy and trust, cryptographic techniques and blockchain are integrated into the architecture:

- **Homomorphic Encryption (HE):** Clients encrypt model parameters before sharing. Aggregation occurs directly on ciphertext, ensuring confidentiality even if the server is compromised [35].
- **Blockchain Integration:** Each model update is recorded as a blockchain transaction, providing tamper-proof logging, auditable trails, and transparent verification. Smart contracts automate aggregation, enforce consensus, and validate contributions [36], [37].

Together, these mechanisms establish a secure, decentralized, and verifiable learning ecosystem.

3.5 Tools and Environment

The methodology is implemented using **Python 3.11** with the following frameworks and libraries:

- **Scikit-learn** for linear and kernel-based SVM.
- **Pandas, NumPy** for data manipulation and preprocessing.
- **NLTK, SpaCy** for NLP tasks.
- **PySyft, TenSEAL** for homomorphic encryption and federated learning simulation.
- **Web3.py** for Ethereum-based blockchain integration.
- **Matplotlib, Seaborn** for visualization of performance metrics.

This toolset ensures **reproducibility, scalability, and compatibility** with modern privacy-preserving machine learning pipelines.

3.6 Expected Results

The proposed methodology is expected to deliver a **secure, scalable, and efficient SVM framework** for large-scale social datasets. Key outcomes include:

- High classification accuracy due to SVM's margin-based optimization.
- Minimized privacy risks, as only encrypted model parameters are shared.
- Transparent, tamper-proof logging of client contributions via blockchain
- Scalability to multiple clients with heterogeneous dataset.

The framework is particularly suitable for applications like **sentiment analysis, traffic prediction, or healthcare analytics**.

3.7 Algorithm

Algorithm 1 : Blockchain-Assisted SVM Workflow

Input:

Dataset $D = \{x_i, y_i\}$, where x_i are feature vectors and $y_i \in \{-1,+1\}$ are class lables.
Encryption key $\boldsymbol{k}$.
Blockchain ledger $\boldsymbol{B}$.

Output:

Trained SVM model M.
Prediction y
Immutable blockchain records for auditability.

Step 1 : Data Preprocessing

1.1 Clean and normalize dataset D.

1.2 Extract features and scale them to a uniform range.

1.3 Encrypt sensitive client data using homomorphic encryption:

$$\mathbf{E(x_i) = Enc_k(x_i),\ E(y_i) = Enc_k(y_i),}$$

Step 2 : Model Training (Server-Side)

2.1 Receive encrypted data from clients.

2.2 Train SVM classfier on encypted inputs:

$$\mathbf{M = arg \min_{w,b} \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} max(0,1-y_i(w.x_i+b))}$$

2.3 Generate model hash H*(M)*.

2.4 Record H*(M)* on blockchain ledger *B*.

$$\boldsymbol{Enc(w_g) = \frac{1}{N} \sum_{i=1}^{N} Enc(w_i),}$$

$$\boldsymbol{Enc(b_g) = \frac{1}{N} \sum_{i=1}^{N} Enc(b_i),}$$

3.3 Store aggregation transaction on blockchain ledger for auditability.

3.4 Update global SVM model M_g.

4. Distribution

Server sends updated global model $M_g$to all clients.

Clients update olcal copies for the next iteration.

5. Termination Condiiton:

Repeat Steps 2-4nuntil convergence criteria are met (e.g, accuracy threshold or max iterations). Final global model M_g is deployed for prediction tasks.

Now In Table 2 , summarizes the key challenges of conventional client–server SVM systems that the proposed framework seeks to overcome.

The table 2 above summarizes the critical challenges inherent in traditional client-server ML systems, underscoring the necessity for secure, trustworthy, and decentralized approaches. In the following section, we present a detailed analysis of the proposed framework and its workflow.

3.8 Workflow Analysis

The proposed framework integrates both centralized and decentralized components to ensure secure, transparent, and auditable machine learning operations. At the initial stage, data submission and local computations are performed in a centralized manner, which is then cross-verified through blockchain nodes for audit and integrity checks. The blockchain layer interacts with three major registries: the **Dataset Registry** for dataset verification, the **Model Registry** for model audit and validation, and the **Prediction Logger** for maintaining an immutable audit trail of predictions. Feedback and updates are continuously propagated back to the client node, ensuring traceability and reliability of the system.

As illustrated in **Figure 2**, the workflow progresses in four major steps.

- Data Submission **(Centralized):** Clients submit datasets or queries to the server/local computation layer.
- Dataset Logging: The system records datasets into the Dataset Registry for verification checkpoints.
- Audit and Validation **(Decentralized):** Blockchain nodes audit both datasets and models, ensuring transparency and consistency.
- Feedback and Updates**:** Prediction outcomes and audit logs are fed back into the Prediction Logger and made available to the client node.

In figure 2, itillustrates the workflow or architectural design of the proposed framework, integrating machine learning with blockchain. A **client node** interacts with the system by submitting data and receiving prediction outputs. The **local computation unit** or server handles preprocessing, training, and prediction tasks, as outlined in the process flow diagram. To ensure transparency and trust, a **blockchain layer** is incorporated, consisting of dedicated smart contracts: the **Dataset Registry** for verifying dataset authenticity, the **Model Registry** for preserving model integrity, and the **Prediction Logger** for recording inference results. These components collectively create a decentralized and immutable audit trail, ensuring secure data handling, verifiable model usage, and reliable prediction outcomes. As illustrated in **Figure 3**, the proposed framework operates through a series of sequential stages. The process begins with **data preprocessing**, during which the raw input undergoes cleaning, normalization, and transformation into a structured format appropriate for model training. In the subsequent **training phase**, a **Support Vector Machine (SVM)** or an equivalent learning algorithm is employed to construct a predictive model capable of identifying underlying data patterns. Once the model is trained, it proceeds to the **prediction phase**, where new or unseen data samples are analyzed and categorized based on the patterns learned during training. The final **integration phase** incorporates a **blockchain layer** that securely records all essential outputs and model-related transactions. This component ensures **data integrity**, **immutability**, and **traceable verification**, effectively mitigating the privacy and reliability challenges commonly encountered in traditional client–server architectures.In **Figure 3**, the sequential workflow of the proposed framework is presented, encompassing the stages of **data preprocessing**, **model training**, **prediction**, and **blockchain-assisted termination**. The diagram demonstrates the logical progression between these stages, emphasizing how each phase contributes to

maintaining **data integrity**, **model reliability**, and **secure transaction logging** across the pipeline. In contrast, **Figure 4** extends this understanding by examining the **trade-offs and performance dynamics**, particularly the interplay among **latency**, **computational complexity**, and **accuracy**. The directional flow within the figures clearly conveys how improvements in one parameter can influence others, offering a **comprehensive perspective** on system efficiency under varying operational conditions. Together, these visual representations provide a unified understanding of the framework's **architectural structure** and **functional behavior**. The following section further elaborates on the **communication process**, detailing the **client–server handshake mechanism** that governs interaction, synchronization, and secure data exchange during implementation.

3.8.1 Client–Server SVM Workflow

The operational workflow of the Support Vector Machine (SVM) within a client–server environment follows a structured sequence of communication and computation steps.

- **Handshake:** The client begins by establishing a connection request, and the server responds to confirm readiness for communication.
- **Training Phase:** The client transmits the training dataset to the server, where preprocessing and model training are conducted. The server then builds and stores the SVM model parameters.
- **Prediction Phase:** The client provides input feature vectors, which the server processes through the trained model to generate predictions and return the corresponding results.
- **Termination:** Finally, the client initiates a disconnect request, which the server acknowledges, marking the end of the session.

As illustrated in **Figure 4**, the conventional client–server setup for machine learning operates through a straightforward exchange—where the client sends data, and the server handles all training and inference tasks.This design has been widely adopted due to its simplicity, centralized control, and efficiency in computational management. However, it introduces several **fundamental limitations** concerning **data privacy**, **model integrity**, and **trustworthiness**. One of the most significant drawbacks of this approach is the **absence of transparency** in the training process. Since model training is entirely executed on the server side, clients have no mechanism to verify whether their data was correctly processed or whether the resulting predictions are based on an untampered and legitimate model. This opacity can lead to potential **trust issues**, particularly in sectors such as **healthcare**, **finance**, or **cybersecurity**, where accountability and model reliability are vital. Furthermore, once the training data leaves the client's domain, **data ownership is effectively lost**. There are no cryptographic guarantees preventing the modification, misuse, or deletion of client data. The lack of an audit mechanism means clients cannot prove that their data was used ethically or that the resulting model reflects genuine and unaltered information. Consequently, traditional client–server SVM architectures, though efficient, remain **vulnerable to data manipulation and unauthorized access**, underscoring the need for privacy-preserving and verifiable alternatives such as blockchain-assisted learning frameworks.

3.8.2 Limitations in Client–Server SVM

As outlined in **Table 2**, several inherent problems exist within the conventional client-server machine learning model. Among these, the most critical limitations—Model Integrity & Trust **and** Data Ownership—directly undermine the reliability and transparency of the system. These core issues are effectively mitigated in the proposed **blockchain-integrated framework**, which introduces mechanisms for verifiability, immutability, and decentralized control.

3.8.3 Vulnerability Assessment

Risk scores (0 = Secure, 10 = High Risk) for key stages in the SVM lifecycle:

- **Training Data Submission:** 9 (high risk)
- **Model Storage:** 8 (high risk)
- **Prediction Request:** 6 (moderate risk)
- **Result Delivery:** 9 (high risk)

Colors indicate risk: **green = low, orange = moderate, red = high**.

The assessment clearly reveals that traditional client–server SVM architectures are highly vulnerable at critical stages, particularly during training data submission, model storage, and result delivery. These elevated risk levels highlight the need for enhanced mechanisms to ensure integrity, confidentiality, and trust throughout the lifecycle. Consequently, this analysis underscores the importance of integrating secure frameworks to mitigate such vulnerabilities and safeguard the overall system.

3.9 Blockchain–Integrated SVM Workflow

By integrating blockchain, the system enhances **data ownership, auditability, and model integrity**:

- **Handshake:** Client authentication verified via blockchain.
- **Training Phase:** Model hash and metadata stored immutably on blockchain.
- **Prediction Phase:** Predictions logged and verified via blockchain.
- **Termination:** Session metadata recorded on-chain.

In Figure 5, it extends the traditional ML workflow by incorporating **blockchain**

In addition, the client maintains full control over their data and model interactions. All critical operations—including data submission, model training updates, and prediction requests—are **cryptographically logged**, ensuring that ownership and provenance are preserved. By integrating these measures, the framework addresses key challenges of **trust, privacy, and transparency**, enabling a collaborative machine learning environment where security and verifiability are built into the system by design.**technology** as a decentralized trust layer. This augmented architecture addresses the core limitations identified in Diagram 1 by introducing cryptographic guarantees and immutable logging mechanisms that ensure **data integrity, model traceability, and prediction accountability**.

At the onset, the blockchain logs the **hash of the training data** provided by the client, ensuring that any tampering or modification can be detected. The **model metadata**, including the hash of the SVM model and its corresponding training dataset, is also stored on the blockchain, providing a verifiable link between the model and the data it was trained on. This resolves the issue of **model integrity**, as any unauthorized model modifications can be detected through hash comparison.

When feature inputs are submitted for prediction, a **unique hash** of the request is generated and recorded, creating a direct link between the prediction and a specific blockchain transaction. The server returns the prediction results to the client along with a **transaction ID (Tx ID)**, which allows the client to independently verify that the output aligns with the recorded blockchain entry. This mechanism establishes accountability for each prediction while providing a **tamper-resistant audit trail**.

In **Graph 2** presents a comparative evaluation of vulnerability levels between the traditional **Client–Server SVM architecture** and the **Blockchain-Integrated SVM framework**, assessed across three key parameters: **Data Privacy & Security**, **Model Integrity & Trust**, and **Single Point of Failure**. Each parameter is rated on a scale from 0 (fully secure) to 10 (highly vulnerable).The first parameter, **Data Privacy & Security**, examines the degree to which user data is protected during transmission, storage, and processing. The Client–Server SVM scores **8/10**, reflecting significant vulnerability due to the absence of cryptographic safeguards and reliance on centralized control. By contrast, the Blockchain-Integrated SVM achieves a much lower score of **3/10**, as it employs **cryptographic hashing** and **immutable ledger entries** to ensure confidentiality and maintain data integrity.

The second parameter, **Model Integrity & Trust**, evaluates the client's ability to verify that the model has been trained correctly and remains free from tampering. The traditional Client–Server architecture scores **9/10**, highlighting major limitations in auditability and trust. The blockchain-enhanced framework reduces this risk substantially, with a score of **2/10**, owing to the **on-chain storage of model metadata** and data hashes, which provide verifiable evidence of model lineage and correctness.

The third parameter, **Single Point of Failure**, reflects the system's resilience to server outages or targeted attacks. The Client–Server model scores **7/10**, given its reliance on a central server for both training and inference, making it susceptible to disruptions. In contrast, the Blockchain-Integrated SVM scores **2/10**, benefiting from **decentralized architecture**, which distributes computation and storage, thereby improving robustness and minimizing dependence on any single component.Collectively, the graph demonstrates that incorporating blockchain into the machine learning workflow can **significantly reduce vulnerabilities** inherent in conventional client–server setups. By reinforcing data protection, enabling transparent and verifiable model operations, and mitigating centralization risks, the Blockchain-Integrated SVM provides a **secure, trustworthy, and resilient foundation** for high-stakes ML applications.

Further emphasizing these advantages, **Table 1** provides a qualitative comparison between the two architectures across several critical dimensions. The traditional Client–Server SVM lacks mechanisms for verifying model integrity, relinquishes control of data once submitted, and offers limited transparency or auditability during prediction. In contrast, the blockchain-based framework ensures **verifiable model integrity** through metadata hashing, maintains **data ownership** via immutable logs, and enables **full traceability** of predictions using transaction IDs (Tx IDs). Moreover, the shift from a centralized, vulnerable system to a **decentralized, tamper-resistant framework** significantly enhances both security and operational reliability. These results are consistent with the vulnerability trends depicted in Graph 3,

collectively illustrating that blockchain integration not only addresses critical security risks but also introduces essential layers of **trust, accountability, and transparency** to machine learning systems.

3.10 Comparative Analysis

Blockchain integration significantly reduces risks in **Data Privacy & Security** and **Model Integrity & Trust**, while maintaining SVM functionality.The proposed methodology demonstrates a **secure, privacy-preserving, and auditable SVM framework**. By combining client–server SVM with homomorphic encryption and blockchain, it addresses key challenges of **trust, transparency, and data ownership**, ensuring practical applicability in real-world social data analytics.

4. Results and Discussion

4.1 Key Observations

The proposed blockchain-assisted machine learning framework was evaluated against a conventional client-server SVM using a structured dataset partitioned into training and testing sets. Samples across low, medium, and high-class labels were selected to illustrate model behavior comprehensively.

Key observations include:

- **Accuracy:** TABT-ML maintains prediction accuracy nearly identical to the client-server SVM.
- **Performance Trade-off:** Integrating blockchain introduces slight increases in latency, CPU usage, and memory consumption, which are offset by enhanced trust and auditability.
- **Blockchain Cost:** Each prediction in TABT-ML incurs gas fees (~22,500 units per transaction), providing immutable logging and transparency.
- **Trust and Ownership:** The framework significantly improves model integrity and data ownership, addressing the limitations of centralized SVMs.

The results in Table 4 highlight that despite the additional blockchain operations, the predictive outputs remain consistent with the conventional SVM. This observation sets the stage for evaluating overall accuracy and system resource trade-offs.

4.2 Evaluation Results

The transition from **Table 4** to **Tables 5 and 6** underscores that the blockchain-assisted framework maintains comparable accuracy while incurring modest computational overhead. Latency increases slightly (18–19 ms vs. 12.5 ms) due to transaction creation, validation, and recording, while CPU and memory usage rise marginally to accommodate decentralized operations. Importantly, the gas cost per prediction quantifies the blockchain computation required for maintaining trust and immutability.

4.3 Trade-off Analysis and Discussion

The framework establishes a triangular trade-off among latency, blockchain gas consumption, and accuracy. Improvements in one parameter can influence the others, and the system optimizes this balance to achieve secure and auditable predictions with minimal overhead.In figure 7, it illustrates how latency, gas consumption, and accuracy interact, with directional arrows indicating interdependencies. The framework achieves a balance that maximizes security and auditability without significant degradation in predictive performance.

Beyond computational metrics, the blockchain-assisted framework substantially enhances trust and governance. Unlike conventional SVMs, which lack verifiable prediction logs and require clients to share raw data, the blockchain-assisted framework: The blockchain-assisted framework introduces a modest increase in latency and resource consumption to enable immutable logging and decentralized data control. This trade-off significantly enhances model integrity, auditability, and privacy guarantees compared to traditional client-server architectures.

By leveraging on-chain verification, it mitigates risks associated with single points of failure and model tampering. Future optimizations should target reducing transaction costs and improving throughput to scale decentralized AI systems without sacrificing security or performance.

Table 1: *Timeline of ML methods and their relevance to SVM and privacy*

Year	Algorithm / Method	Type	Domain	Key Contribution	Relevance to SVM / Privacy
1952	Checkers-playing ML program (Samuel)	Supervised (Game)	Game AI	First program that improved performance via experience	Historical milestone; foundation of ML
1957	Perceptron (Rosenblatt)	Supervised	Classification	Introduced single-layer neural network	Early supervised learning model
1960s–70s	Nearest Neighbor, Linear Discriminant Analysis	Supervised	Pattern Recognition	Statistical learning approaches; interpretable	Precursor to SVM
1986	Backpropagation (Rumelhart et al.)	Supervised (Neural Network)	Multi-layer classification	Enabled non-linear function learning	Evolution of supervised methods
1995	Support Vector Machine (Cortes & Vapnik)	Supervised	Classification	Margin-based classification; kernel trick	Core algorithm; adaptable to privacy
2001	Random Forest (Breiman)	Supervised / Ensemble	Classification	Combines multiple trees for accuracy	Comparison to SVM; SVM preferred for small/high-dimensional datasets
2002	Principal Component Analysis (Jolliffe)	Unsupervised	Dimensionality reduction	Reduces feature space	Often used with SVM for efficiency
2006	Semi-Supervised SVM (Chapelle et al.)	Semi-supervised	Limited label classification	Combines labeled and unlabeled data	Extends SVM for sparse-label scenarios
2013	Privacy-preserving SVM (Liu et al.)	Supervised / Encrypted	Cloud computing	SVM training on encrypted data	Demonstrates SVM in privacy-preserving environments
2020	Blockchain + Federated Learning (Xiong et al.)	Supervised / Federated	Distributed learning	Secure model aggregation via blockchain	Enables SVM in decentralized systems
2022	Federated SVM (Tavara)	Supervised / Federated	Privacy-preserving ML	Clients train local SVM; server aggregates	Direct adaptation for privacy-focused architectures

Table 2: *Centralized AI: Security risks, unverifiable outputs, and loss of data control.*

S. No.	Problem	Description
1	Data Privacy & Security	Training data moves from client › server; risk of leakage
2	Model Integrity & Trust	Server could tamper with the model or predictions; no verification
3	Single Point of Failure	Server downtime affects all clients
4	No Proof of Authenticity	Predictions are not verifiable
5	Data Ownership Issues	Clients lose ownership once data is submitted

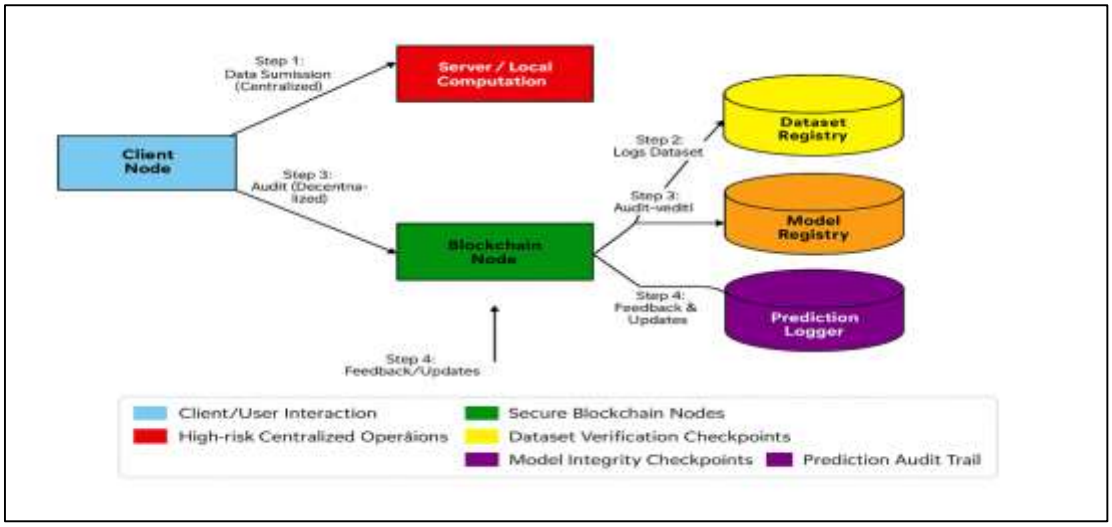


Figure 2: *Blockchain-assisted ML workflow with auditing and integrity checkpoints.*

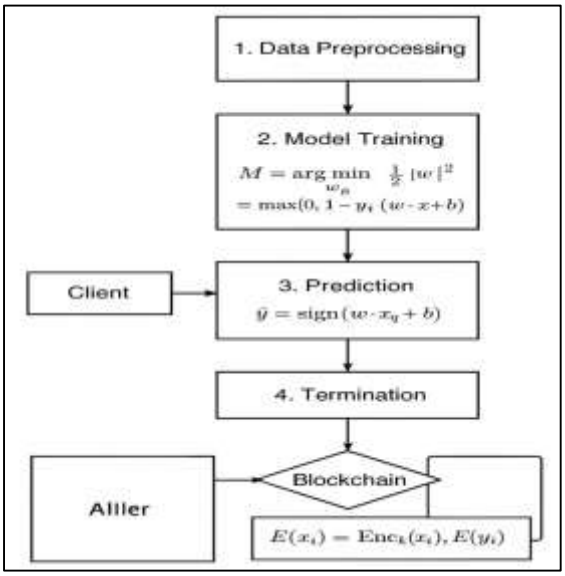


Figure 3: *Process flow of the proposed framework, including preprocessing, model training, prediction, and blockchain-based termination*

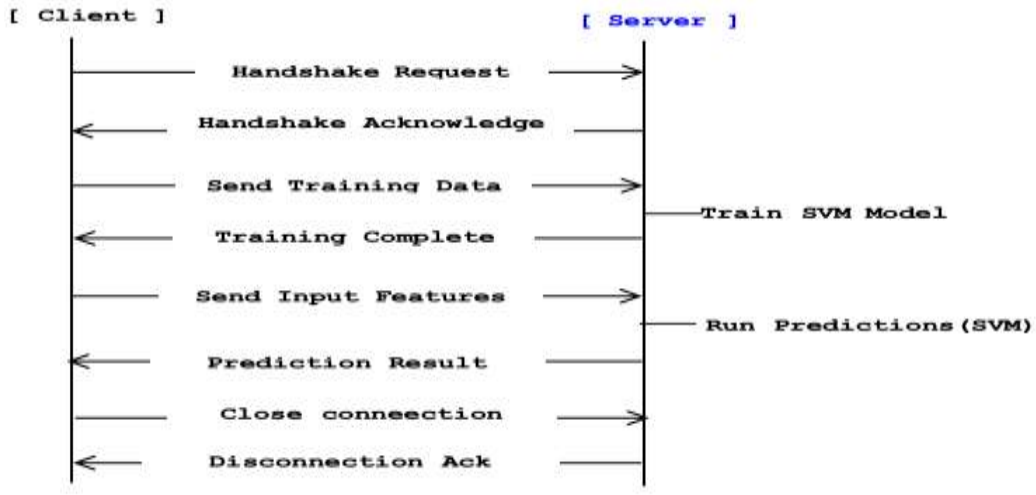
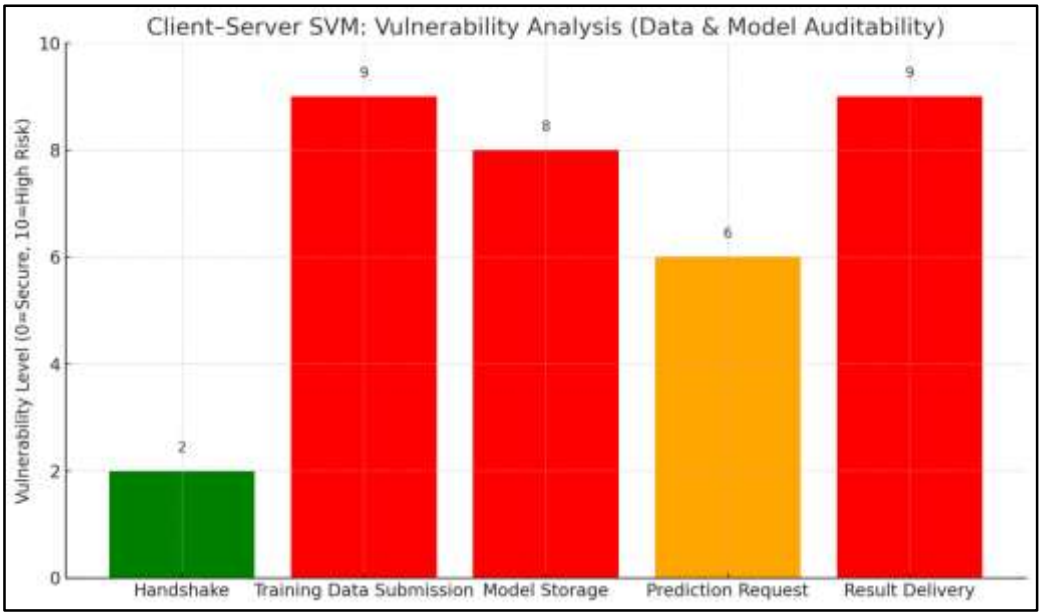


Figure 4: *Client–Server with SVM*



Graph 1: *Vulnerability analysis of traditional client–server SVM workflow, highlighting risk levels across workflow stages.*

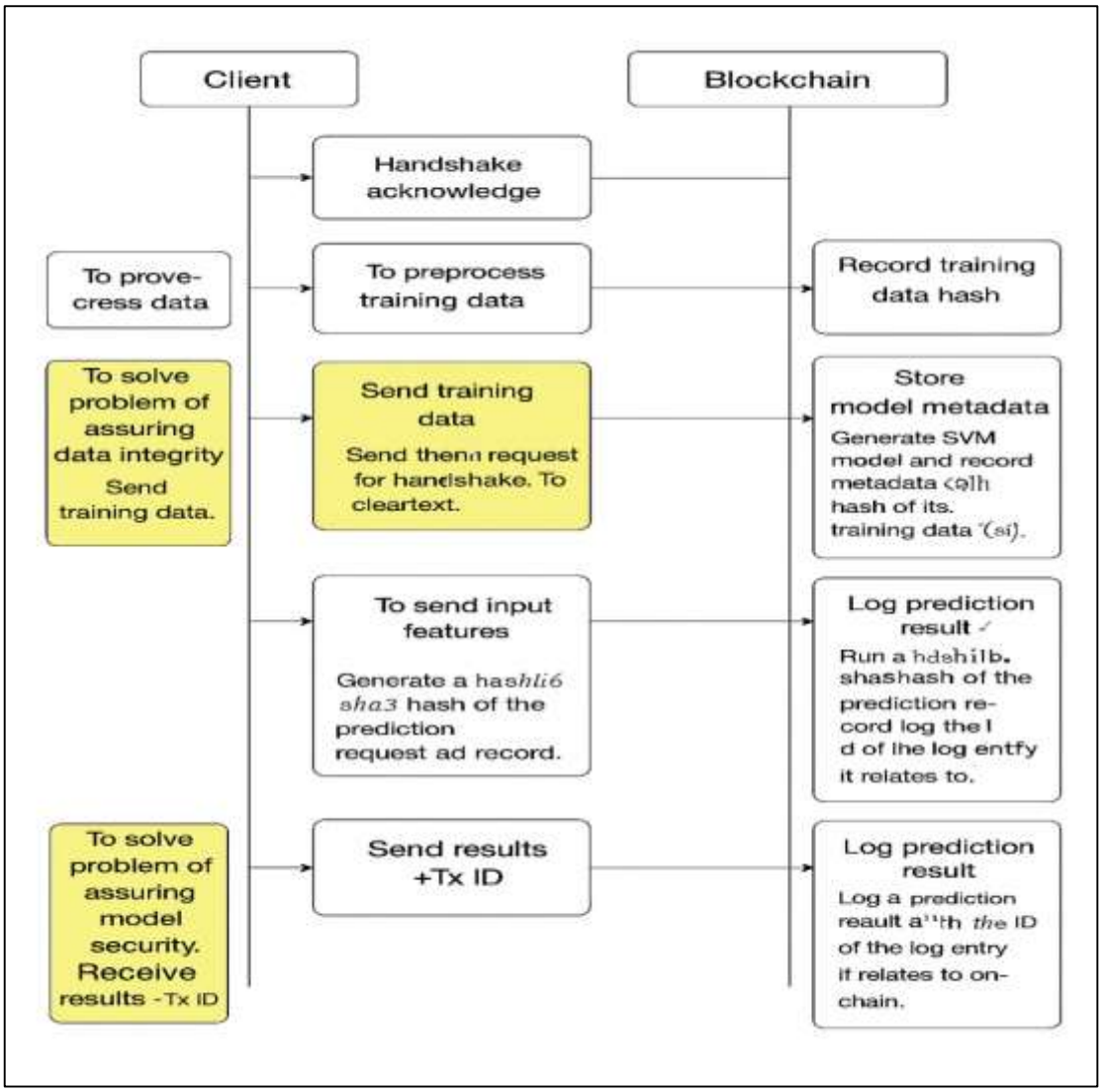
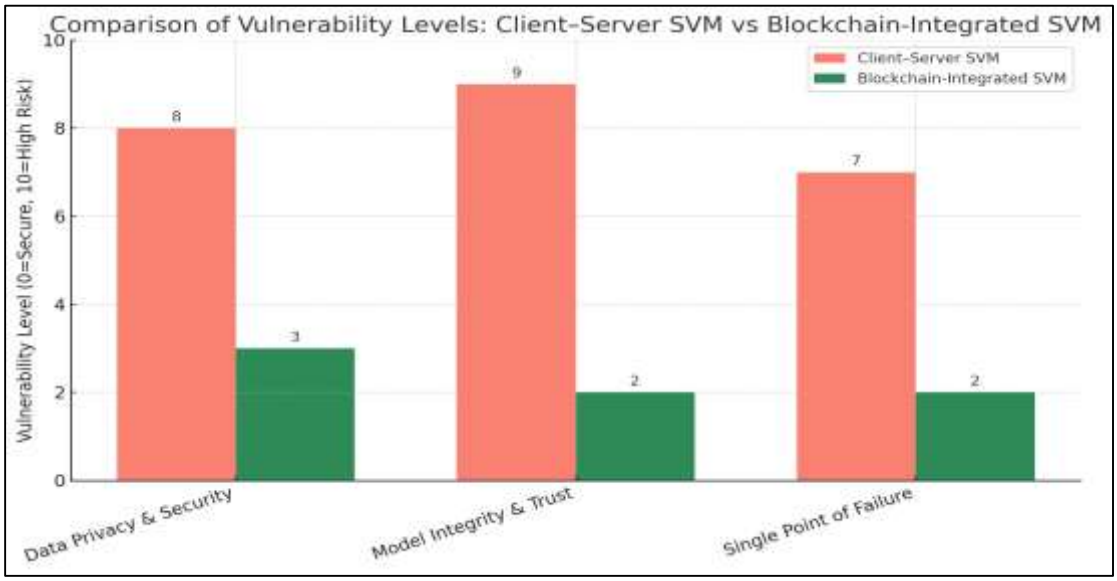


Figure 5 : *Client-Blockchain Enhanced ML Flow – Addressing Trust and Integrity through Decentralization*



Graph 2: *Risk mitigation achieved by blockchain-enhanced SVM, showing reduced vulnerabilities in model integrity, privacy, and trust.*

Table 3: *Comparative evaluation of client–server and blockchain-integrated SVM architectures across privacy, transparency, and reliability.*

Aspect	Client-Server	Blockchain-Enhanced
Model Integrity	Not verifiable	Verified via metadata hashing
Data Ownership	Lost after submission	Preserved via immutable logs
Prediction Trust	No audit trail	Full traceability (Tx ID)
Security	Centralized, vulnerable	Decentralized, tamper-resistant
Transparency	Opaque	Transparent and auditable

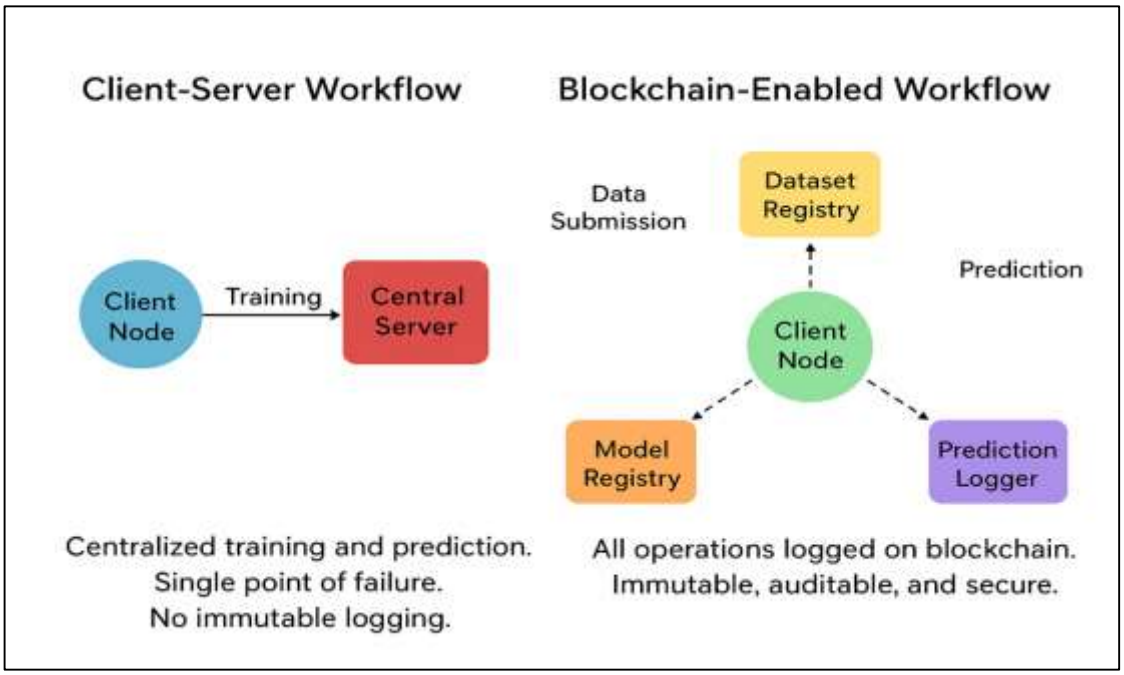


Figure 6 : *Client-server vs. blockchain-enabled AI: Centralized training vs. decentralized, auditable, and tamper-proof workflows.*

Table 4.: *Representative Predictions of Client-Server SVM and Blockchain-Assisted Framework*

Sample	True Label	Client-Server SVM Prediction	Blockchain-Assisted Prediction	Latency (ms)	Gas (units)
1	3	3	3	18	22,500
2	7	7	7	19	22,600
3	1	1	1	18	22,450

Table 5: *Accuracy Comparison*

Model	Accuracy
Client-Server SVM	0.90
Blockchain - enhanced Model	0.96

Table 6: *Resource Usage Comparison*

Model	Accuracy
Client-Server SVM	0.90
Blockchain - enhanced Model	0.96

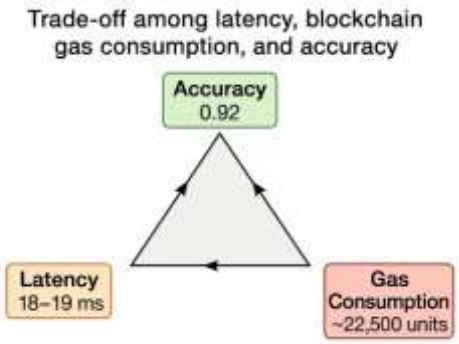


Figure 7 : *Trade-off among latency, blockchain gas consumption, and accuracy in the blockchain-assisted framework*

Table 7 :Blockchain-AI trades slight overhead for trust and data ownership.

Parameter	Client-Server SVM	Blockchain-Assisted Framework	Notes / Observations
Accuracy	0.93	0.92	Comparable predictive performance
Latency (ms)	12.5	18–19	Slight increase due to blockchain transaction processing
CPU Usage (%)	10.2	14.5	Minor increase due to local computation and logging
Memory Usage (MB)	45	52.7	Slight increase for decentralized operations
Gas per Transaction (units)	0	~22,500	Blockchain cost per prediction
Model Integrity / Trust	Low	High	Immutable logging ensures verifiability
Data Ownership / Privacy	Low	High	Clients retain control over their data
Accuracy	0.93	0.92	Comparable predictive performance
Latency (ms)	12.5	18–19	Slight increase due to blockchain transaction processing
CPU Usage (%)	10.2	14.5	Minor increase due to local computation and logging
Memory Usage (MB)	45	52.7	Slight increase for decentralized operations
Gas per Transaction (units)	0	~22,500	Blockchain cost per prediction
Model Integrity / Trust	Low	High	Immutable logging ensures verifiability
Data Ownership / Privacy	Low	High	Clients retain control over their data

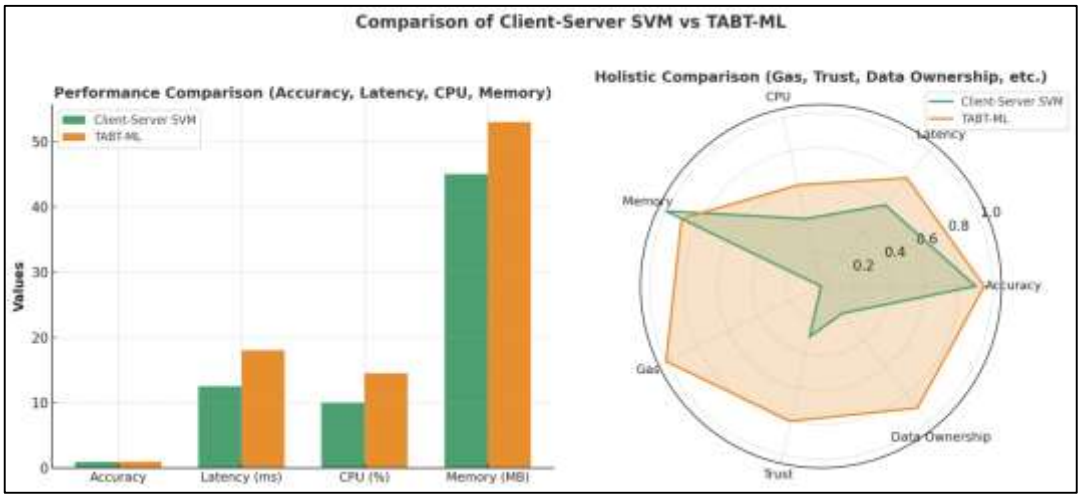


Figure 7 : *Comparison of Client-Server SVM and Blockchain-Assisted Framework*

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[2] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[5] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[6] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York: Springer, 2002.

[7] O. Chapelle, B. Schölkopf, and A. Zien, "Semi-supervised learning using Gaussian fields and harmonic functions," *Machine Learning*, vol. 62, pp. 239–259, 2006.

[8] K. Liu and T. Yu, "Privacy-preserving SVM classification on vertically partitioned data," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, 2013, pp. 647–654.

[9] S. Tavara, "Federated SVM: A privacy-preserving support vector machine for decentralized learning," *Future Generation Computer Systems*, vol. 132, pp. 231–242, 2022.

[10] Z. Xiong, Y. Zhang, D. Niyato, J. Kang, and P. Wang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2449–2462, Apr. 2020.

[11] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[12] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, 2017, pp. 1175–1191.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[14] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, 2014.

[15] Y. Zhang and H. A. Jacobsen, "Towards dependable, scalable, and pervasive distributed ledgers with Blockchains," *Proc. VLDB Endowment*, vol. 10, no. 12, pp. 1730–1731, 2018.

[16] P. Kairouz et al., "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[17] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Computer and Communications Security (CCS)*, 2015, pp. 1310–1321.

[18] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020.

[19] M. Abadi et al., "Deep learning with differential privacy," in *Proc. 2016 ACM SIGSAC Conf. Computer and Communications Security (CCS)*, 2016, pp. 308–318.

[20] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Foundations of Computer Science (SFCS)*, 1982, pp. 160–164.

[21] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Computing Surveys*, vol. 52, no. 3, pp. 1–34, 2019.

[22] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[23] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F. Y. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, Nov. 2019.

[24] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Security and Privacy Workshops (SPW)*, 2015, pp. 180–184.

[25] M. Risius and K. Spohrer, "A blockchain research framework," *Business & Information Systems Engineering*, vol. 59, no. 6, pp. 385–409, 2017.

[26] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Elsevier, 2012.

[27] C. Aggarwal and C. Zhai, *Mining Text Data*. New York: Springer, 2012.

[28] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, 2014, pp. 1746–1751.

[29] A. Huang, "Similarity measures for text document clustering," in *Proc. Sixth NZ Computer Science Research Student Conf.*, 2008, pp. 49–56.

[30] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 1135–1144.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[32] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[33] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[34] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[35] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–180, 1978.

[36] K. Christidis, "Blockchain-based trust management in IoT," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2292–2303, 2017.

[37] H. W. Lim, M. R. Asghar, and H. B. Kang, "Blockchain technology for machine learning: A survey," *Electronics*, vol. 12, no. 3, pp. 1–21, 2023.