

Real-Time Fraud Detection in Financial Transactions Using Hybrid Machine Learning Models and Kafka

Morchid Laila¹, , Haung Lianli*

¹ School of Intelligent and connected vehicle, Hubei University of Automotive Technology, Shiyan 442002, China

*Associate Professor, Hubei University of Automotive Technology, Shiyan 442002, China

Abstract - Machine learning (ML) is increasingly being used to defend digital financial systems from fraudulent transactions; nevertheless, achieving high detection accuracy in real-time and on a wide scale remains challenging. This paper describes a Kafka-based machine learning pipeline for real-time analysis of financial transaction streams. The suggested pipeline uses Apache Kafka for distributed streaming and tests four classification approaches: Logistic Regression, Random Forest, XGBoost, and a hybrid soft-voting ensemble. A dataset of 209,715 transactions under imbalanced fraud scenarios was used to evaluate performance. Significant differences can be observed in the results. Logistic regression demonstrated limited applicability for severely imbalanced fraud detection, with an accuracy of 88.7% and a fraud recall of 0.042. Random Forest performed the best overall, with 97.6% accuracy, 0.886 F1 score, and 0.994 AUC-ROC. XGBoost achieved 93.9% accuracy, with a competitive balance of precision (0.711) and recall (0.825). The hybrid voting classifier maintained low-latency inference while boosting resilience in a Kafka-based streaming context, with 96.7% accuracy, an F1 score of 0.838, and an AUC ROC of 0.984. These findings indicate that combining Kafka-driven streaming with ensemble machine learning is an efficient and scalable solution for real-time financial fraud detection, particularly in high-volume transaction scenarios where detection performance and reaction time are critical.

Key Words: Real-Time Digital Transactions Fraud Detection Machine Learning, Logistic Regression, Random Forest, XGBoost, Hybrid Voting Classifier, Apache Kafka, Data Pipeline, Fraud Risks.

1. INTRODUCTION

Digital payment systems have become important to modern financial activity, with card-present transactions (in-person) at points of sale and card-not-present transactions (online) via electronic and mobile platforms accounting for the majority of currency exchanges [1]. These channels provide speed, convenience, and accessibility to millions of users every day; however, they have become more vulnerable to fraudulent activity, making fraud detection a key concern for financial institutions. Traditional rule-based detection methods, including one-time passwords sent via SMS (OTPs), mobile transaction authorization numbers (mTANs) [2], U-Shields [3], and biometric authentication [4], have improved

identity verification; however, research shows that many mobile banking applications still contain exploitable vulnerabilities [5]. This demonstrates that despite the current protections, the evolving nature of fraud requires advanced and flexible detection systems.

Machine Learning (ML), a dynamic and data-driven method that can learn complex behavioral patterns and adapt more effectively than static rule-based systems, has been recognized to address these issues. The successful results of XGBoost, Random Forest, and Logistic Regression in detecting fraud are widely acknowledged. While Logistic Regression stands out for its interpretability and computational efficiency [6], Random Forest uses ensemble learning for robust prediction [7][8]. XGBoost achieves superior recall and F1 score, which are vital for identifying infrequent fraudulent transactions. The program is particularly good at handling highly imbalanced datasets. Combining these models to form a Hybrid Voting Classifier further improves predictive accuracy and provides an effective defense against fraudulent activity in both swipe and online payments.

For real-time fraud detection, where quick decision-making is required to reduce losses and maintain consumer confidence, a solid and scalable data infrastructure is vital. This function is performed by Apache Kafka, a distributed messaging platform originally developed by LinkedIn, which facilitates high-throughput and low-latency data streams. Kafka ensures fault tolerance, scalability, and resilience by segmenting data into topics and distributing these across multiple brokers. To guarantee reliability, producers append transaction messages to topics, consumers fetch them for processing, and leaders replicate data to followers. Kafka enables the ongoing ingestion, processing, and dissemination of online and swipe transactions within fraud detection frameworks. This capability allows for seamless integration with machine learning models designed for anomaly detection and real-time analysis [13] [9].

This project provides a Kafka-based, real-time fraud detection system that combines Logistic Regression, Random Forest, XGBoost, and a Hybrid Voting Classifier into a single pipeline. The system collects and analyzes transaction streams, classifies events in real time, and constantly updates models using past data to increase accuracy. By focusing on both swipe and online

transactions, the system handles the dual difficulty of processing high-frequency in-person payments while also safeguarding remote digital channels.

2. LITERATURE REVIEW

Fraud detection in financial transactions has gained prominence due to the rising popularity of digital payment systems and the increasing sophistication of fraudulent activities. Traditional rule-based systems, which depend on established heuristics, are not very good at finding new types of fraud because they can't apply what they know to attacks that are new or changeable [10]. For example, rule-based techniques frequently fail when fraudsters significantly alter transaction characteristics, rendering such systems ineffective for modern, high-frequency digital transactions [11]. To address these constraints, machine learning approaches have been extensively studied for fraud detection. Logistic Regression is a simple linear model that can find big patterns, but it doesn't work well with datasets that aren't balanced, like when there aren't many fraudulent transactions [12]. Non-linear models like Random Forest and XGBoost are better at finding complicated patterns in transaction data. Random Forest is better at handling noise, and XGBoost is better at dealing with unbalanced datasets through gradient boosting [13][14]. Support Vector Machines (SVMs) and Neural Networks have also been employed to identify non-linear separations and high-dimensional correlations, although these models can be computationally intensive and slow to infer [15][16].

Ensemble and hybrid models, which use many classifiers, have been better at balancing precision and recall, especially when the classes in the dataset are very different from each other [17]. These methods reduce the likelihood of false negatives while maintaining high precision, which is essential in financial systems that need to find a balance between fraud detection and user experience. In our research, we used a Kafka-driven real-time pipeline with a hybrid Voting Classifier that combined Logistic Regression, Random Forest, and XGBoost. This approach enables continuous transaction ingestion, real-time preprocessing, and rapid scoring, ensuring the low-latency replies required for high-frequency transaction monitoring [18][19]. Our dataset contains 209,715 transactions, with approximately 11.8% being fraudulent, making the detection task very difficult. In baseline tests, Logistic Regression achieved exceptionally high precision (0.9923) but extremely low recall (0.042), indicating a limited ability to detect all fraudulent transactions [20]. Random Forest achieved a recall of 0.8037 and an AUC-ROC of 0.9939, indicating a high capacity to detect fraud while maintaining precision [21]. XGBoost achieved a balanced F1-score of 0.7641 and an AUC-ROC of 0.9609, demonstrating its competence in managing unbalanced data [22]. Our hybrid Voting Classifier increased performance even further, reaching an

F1-score of 0.838 and an AUC-ROC of 0.984, illustrating the benefit of integrating complementary models within a streaming context **Error! Reference source not found.**

When compared to previous studies, our approach shows clear and measurable gains in a variety of dimensions. For example, research using batch processing with Random Forest or XGBoost, such as **Error! Reference source not found.** and **Error! Reference source not found.**, found high precision scores ranging from 0.90 to 0.95. However, these models frequently had much lower recall rates, ranging from 0.60 to 0.75, indicating that a large percentage of fraudulent transactions were overlooked. This limitation is crucial in real-time applications, since undetected fraud can result in immediate financial losses. Furthermore, batch-processing methods impose latency by requiring adequate data to make predictions, making them unsuitable for high-frequency transaction monitoring.

Other prior works leveraged ensemble techniques, including combinations of Random Forest, Logistic Regression, or XGBoost [23], to increase detection metrics like F1-score. These studies produced F1-scores of up to 0.83, indicating a better balance of precision and recall. However, their structures were primarily intended for offline analysis and did not include streaming data pipelines. As a result, while these models work well on static datasets, they cannot manage continuous, high-volume transaction streams in real time. Furthermore, while some algorithms increased recall by utilizing techniques such as SMOTE for class balance [16], they still relied on batch inference and did not achieve the low-latency, high-throughput performance required in operational fraud detection systems.

On the other hand, our work combines the distributed streaming capabilities of Apache Kafka with a hybrid Voting Classifier that incorporates Random Forest, XGBoost, and Logistic Regression. There are numerous advantages to this architecture. First, Kafka eliminates batch processing delays by enabling continuous transaction input and processing. Second, the hybrid ensemble makes use of the complementing capabilities of its component models: XGBoost helps to provide strong performance in situations with unbalanced data, Random Forest enhances recall, and Logistic Regression guarantees excellent precision. In actuality, this integration helped our system exceed the majority of previous models in terms of detection rate and the ratio of false positives to false negatives, achieving a high F1-score of 0.838 while maintaining a recall of 0.7252.

By closely studying the results of prior studies, the benefits of our system become clear. Studies such as [24]**Error! Reference source not found.** found adequate accuracy and precision measures but frequently neglected recall or latency issues. Models based on neural networks

or deep learning [25][26] produced great accuracy but were computationally demanding, making them unsuitable for real-time streaming at scale. Ensemble-based methods [27][28] increased detection rates but lacked streaming capabilities, limiting their utility in continuous monitoring applications. In contrast, our Kafka-driven hybrid method not only achieves competitive or higher performance in traditional metrics (precision, recall, and F1-score), but it also handles operational constraints such as low-latency inference, scalability, and flexibility to changing fraud patterns. Overall, the combination of streaming architecture and hybrid ensemble learning provides a robust, scalable, and highly successful method of real-time fraud detection, distinguishing it from previous research.

3. RESEARCH METHODOLOGY

The fraud detection workflow combines real-time transaction processing with powerful machine learning to accurately detect anomalies. Transactions, whether card-present (swipe) or card-not-present (online), begin in the streaming layer driven by Apache Kafka. A consumer component collects these streaming events and sends them to a preprocessing module, where raw transaction attributes are standardized, encoded, and converted into feature vectors appropriate for modeling. These vectors are then analyzed using a machine learning engine that includes Logistic Regression, Random Forest, and XGBoost algorithms. The results of these models are integrated using a hybrid soft-voting ensemble to get final fraud probability scores. Based on these predictions, the decision layer routes legitimate transactions to normal processing and suspect transactions to an alert and reporting subsystem. The architecture supports both offline training and model evaluation, as well as real-time scoring and continuous monitoring in a production setting, according to **Figure 1**.



Figure 1: ML System Architecture Diagram

3.1 Transaction Generation

The first step in the fraud detection pipeline is to generate and capture transactional data, which will be used as the system's primary input. Transactions are divided into two types: card-present (swipe) and card-not-present (online).

Swipe Transactions (Card-Present): These occur at physical points-of-sale (POS) terminals. The customer produces a card, and the transaction is completed via magnetic stripe or chip-based authentication. These transactions contain a variety of properties, including:

- Transaction Amount**
- Merchant Category**
- Timestamp**
- Terminal ID**
- Cardholder Information**

Online Transactions (Card Not Present): These transactions take place via e-commerce platforms or digital payment methods, without the physical presence of a card. In addition to the same qualities as swipe transactions, online transactions include:

- IP Address**
- Device ID**
- Geolocation**
- Browser/App Identifiers**

A time-stamped series of transactions is created by continuously recording both swipe and online transactions:

$$T_i = [X_1, X_2, \dots, X_n]$$

where X_1, X_2, \dots, X_n are individual transaction variables such as amount, merchant code, timestamp, device ID, location, etc. Online transactions use behavioral or network-based indicators, such as IP address or Device ID, to distinguish between legitimate and fraudulent activity.

Both **swipe** and **online transactions** are captured continuously, forming a time-stamped sequence of transactions:

$$\{T_1, T_2, \dots, T_m\}$$

where T_k represents each transaction in the sequence, and m represents the total number of transactions in a given observation period. This sequential representation is crucial for stream-based processing, as it enables the system to detect fraud patterns as transactions occur in real-time.

By categorizing and collecting these transactions, the system ensures that all essential transactional attributes are available for preprocessing and machine learning analysis, resulting in successful fraud detection.

3.2 Data Collection

The second stage of the fraud detection system entails the systematic gathering and organization of transactional data. These transactions, whether through card-present (swipe) or card-not-present (online) channels, are consolidated into a single Data Source Repository. This

repository serves as the foundation for the rest of the fraud detection pipeline's processing and analysis.

Each transaction is saved as a structured record with multiple attributes, including but not restricted to:

Identifiers: Transaction ID, date, client ID, card ID

Transactional Details: Amount, card chip usage

Merchant Information: Merchant ID, merchant city, merchant state, postal code, merchant category code (MCC)

Error Categories: Common errors such as Insufficient Balance, incorrect PIN, Technical Glitch, incorrect Expiration Date, False Card Number

Richer analysis during preprocessing and machine learning model training is made possible by this structured format, which guarantees that every transaction's contextual and financial features are recorded. The system can effectively manage massive amounts of data while preserving crucial information for fraud detection by recording transactions in an organized fashion.

Formally, the dataset can be represented as a collection of transaction vectors:

$$D = \{T_1, T_2, \dots, T_m\}$$

where each transaction T_i is a multidimensional vector of attributes. For the dataset used in this study $n=12$, including both numerical variables (e.g., transaction amount) and categorical variables (e.g., merchant city, error type).

The consolidated Data Source Repository not only gives a consistent picture of all transactional activities, but it also enables scalability. It provides easy connection with Apache Kafka's streaming layer, ensuring real-time transaction ingestion and the ability to perform both offline analysis and online model training. This connection allows for the efficient processing and storing of high-frequency transaction data.

3.3 Data Streaming Layer

The system's fundamental distributed streaming platform is Apache Kafka, which allows for real-time financial transaction ingestion and processing. Kafka manages the continuous publishing and consumption of transaction events, ensuring high throughput, low latency, and fault tolerance across the pipeline.

In this stage, all transaction events collected from the data repository are published to Kafka topics, which act as logical channels for data streams. Each new transaction record T_i is serialized and appended to the topic as a message, forming a continuous, ordered log of events:

$$S = \{M(T_1), M(T_2), \dots, M(T_m)\}$$

where S represents the stream of transaction events, and $M(T_i)$ denotes the message corresponding to transaction T_i .

A producer component is responsible for publishing transaction events to the appropriate topic, while one or more consumer applications subscribe to these topics to retrieve messages in real time. The consumer process can be represented as:

$$C(M(T_i)) \rightarrow T_i$$

where $C()$ denotes the consumer process that retrieves the message and forwards the transaction for preprocessing.

Kafka's separating method distributes data across several brokers, enabling horizontal scalability. This approach allows downstream programs to process transaction streams in parallel, such as fraud detection models. Kafka's replication function protects data integrity by storing copies of messages across many brokers, preventing data loss in the event of a failure.

Furthermore, Kafka Streams offers real-time processing and analysis by allowing the consumer to handle transactions as they arrive, making it possible to continuously monitor both card-present and card-not-present transactions for fraud detection. Kafka's built-in characteristics for high-throughput and fault tolerance are critical to system stability, especially in contexts with millions of transactions per day.

Using Kafka, the system can efficiently manage data flow and ensure that all transaction events are available for real-time scoring and fraud detection without introducing the delays associated with batch-processing approaches.

3.4 Consumer and Preprocessing

Once transaction events are published to Kafka topics, a consumer application subscribes to them and receives the records in real time. The consumer acts as an intermediary between raw transaction data and the machine learning engine. Each transaction is processed and converted to a model-ready format for subsequent analysis.

Preprocessing Steps: Data Cleaning and Validation:

Incomplete or malformed records, such as missing values or invalid IDs, are identified and corrected or discarded.

Transaction error (e.g., Insufficient Balance, incorrect PIN) are detected and converted into structured categorical features.

Feature Engineering:

Numerical Features: Continuous variables, such as transaction amount, are **normalized** to reduce skewness and ensure consistency in scale.

Categorical Features: Variables such as merchant city, errors, or cardholder details are transformed using **one-hot encoding or embedding techniques**, making them suitable for machine learning models.

Derived Features: New features, such as **transaction frequency or average spending amount** over a time window, are created to capture additional insights that help distinguish fraudulent behavior from legitimate transactions.

Feature Vector Construction:

Each transaction T_i is represented as a high-dimensional **feature vector**:

$$T_i = [X_1, X_2, \dots, X_n]$$

where x_j corresponds to a preprocessed attribute (e.g., Transaction amount, merchant code, location, or error type). This vector representation allows the machine learning models to effectively detect patterns and anomalies indicative of fraud.

Stream Alignment:

Preprocessed transaction vectors are aligned with Kafka's **message offsets** to ensure the proper order and timing for real-time analysis. This step is crucial for stream-based processing, where maintaining the correct sequence of transactions is vital for detecting fraud in the right context.

Micro-Batching:

For high-throughput scenarios, **micro-batching** techniques may be applied to process multiple transactions simultaneously. This method enhances the speed of scoring while ensuring that latency is minimized, allowing for efficient parallel processing.

3.5 Machine Learning Models (Fraud Detection Engine)

The machine learning engine is at the center of the fraud detection system, assessing each preprocessed transaction to determine its probability of being fraudulent. This engine combines several machine learning techniques, including Logistic Regression (LR), Random Forest (RF), and Extreme Gradient Boosting (XGBoost). These models are merged into a hybrid soft-voting ensemble to improve the system's overall forecast accuracy and robustness.

Logistic Regression (LR)

Logistic Regression is employed as a baseline linear classifier that estimates the probability of fraud based on a logistic function:

$$P(y = 1 | X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

where:

$P(y=1|X)$ is the probability that transaction X is fraudulent

$(\beta_0, \beta_1, \dots, \beta_n)$ are the model coefficients.

(x_1, x_2, \dots, x_n) are the transactions features (e.g., transaction amount, merchant type).

LR is preferred for its interpretability and simplicity, which makes it simple to determine which features have the greatest influence on the model's predictions. However, it struggles with highly imbalanced datasets since it prefers to focus on the majority class.

Random Forest (RF)

Random Forest, an ensemble of decision trees, is utilized to capture **nonlinear feature interactions** and improve classification stability. Each decision tree $h_k(X)$ is trained on a bootstrap sample of the data with random feature selection. The overall prediction is obtained by averaging the probabilities

$$\hat{y} = \frac{1}{k} \sum_{k=0}^k h_k(X)$$

where k is the number of trees, and $h_k(X)$ is the output of the k -Th decision tree.

Random Forest increases classification stability by averaging predictions from numerous trees, lowering the possibility of overfitting. This model is resistant to noise and can handle imbalanced datasets well. Its recall in detecting fraudulent transactions is especially high, making it suited for use in fraud detection systems.

Extreme Gradient Boosting (XGBoost)

XGBoost is incorporated to leverage gradient boosting with regularization, enabling the model to sequentially correct errors of weak learners. For iteration t , the prediction is updated as:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(X_i)$$

where:

$\hat{y}_i^{(t)}$ is the prediction at iteration t

$f_t(x)$ is the new weak learner (tree) added to the model,

η is the learning rate.

XGBoost's ability to manage imbalanced datasets via boosting, as well as its flexibility to fine-tune model parameters (such as learning rate and depth), make it extremely useful in fraud detection. It frequently outperforms established algorithms, such as Logistic Regression, in detecting subtle patterns indicative of fraud.

Hybrid Soft-Voting Ensemble
To combine the strengths of individual models, a **soft-voting ensemble** is implemented. In this approach, each model (LR, RF, and XGBoost) produces a probability score indicating if a transaction is fraudulent. These scores are then combined to make a final prediction:

$$\hat{y}_{\text{ensemble}} = \text{argmax}(\sum_{k=1}^m p_i \cdot w_i)$$

where:

p_i is the probability score from model i (LR, RF, or XGBoost),

w_i is the weight assigned to model i based on its performance (accuracy, recall, etc.),

m is the total number of models in the ensemble.

The ensemble enhances precision, recall, and F1-score by integrating the strengths of Logistic Regression, Random Forest, and XGBoost, resulting in fewer false positives and negatives. The ensemble approach finds the right mix between detecting fraud (high recall) and reducing customer disturbance (high precision).

3.6 Decision Layer

The decision layer in the fraud detection pipeline is responsible for transforming the machine learning ensemble's predictions into actionable results. Once the models (Logistic Regression, Random Forest, and XGBoost) have made their predictions, the hybrid soft-voting ensemble aggregates these probabilities to get a final verdict on whether a transaction is fraudulent.

To make a final choice, the ensemble's aggregated probability is compared to a specified threshold τ . If the probability is greater than the threshold, the transaction is labeled as fraudulent; otherwise, it is considered valid.

$$\hat{y} = \begin{cases} \text{Fraudulent if } \sum_{k=1}^m p_i \cdot w_i \geq \tau \\ \text{Legitimate if } \sum_{k=1}^m p_i \cdot w_i < \tau \end{cases}$$

where:

p_i is the predicted probability from model i

w_i is the weight assigned to the model i (based on performance)

τ is the decision threshold,

m is the total number of models in the ensemble, This decision threshold τ can be adjusted to prioritize either

precision or recall:

Lowering τ increases recall (detecting more fraudulent transactions) but may increase false positives (legitimate transactions flagged as fraudulent).

Raising τ increases precision (reducing false positives) but may result in missed frauds (lower recall).

Operational Action:

Once a decision is made, the system takes appropriate action based on the classification:

Legitimate Transactions: These transactions are routed to the normal processing pipeline, allowing users to complete their transactions without interruption, ensuring a seamless customer experience.

Fraudulent Transactions: Transactions classified as fraudulent are flagged and routed to an **alert and reporting subsystem**, where they can either be reviewed by analysts or subjected to automated interventions, such as:

Transaction Blocking: Temporarily blocking the transaction to prevent financial losses.

Verification Requests: Requesting additional verification from the user to confirm the legitimacy of the transaction.

Account Suspension: Suspending the account if fraudulent behavior is detected over a period.

4. MODEL EVALUATION AND VISUALIZATION

A range of measures were used to evaluate the effectiveness of the implemented fraud detection models, which included Random Forest, XGBoost, Logistic Regression, and the Hybrid Voting Classifier. This study focused not only on prediction accuracy, but also on operational efficiency, including latency and throughput, all of which are crucial for real-time fraud detection.

4.1 Evaluation Metrics

Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- TP: True Positives
- FP: False Positives

Recall (Sensitivity):

$$\text{Recall} = \frac{TP}{TP + FN}$$

- TP: True Positives
- FN: False Negatives

F1-Score:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where:

Precision and **Recall** are the previously defined metrics.

AUC-ROC: Area under the Receiver Operating Characteristic curve, computed using integration:

$$\text{AUC} = \int_0^1 TPR(FPR)d(FPR)$$

Where:

- TPR is the **True Positive Rate**
- FPR is the **False Positive Rate**

Evaluation Setup:

The evaluation was conducted using a local setup with the following configuration:

Hardware: Apple M1, 8-core CPU, 7-core or 8-core GPU, 16-core Neural Engine, with 8GB unified memory (configurable to 16GB) and 256GB SSD.

Software: The system was powered by **Ubuntu** with **Apache Kafka** for real-time data streaming simulation, ensuring a realistic environment for high-frequency financial transaction processing.

Model Performance Comparison

The results of the evaluation showed that **Random Forest** outperformed all models with an **AUC of 0.9939**, maintaining high **precision (0.9864)** and **recall (0.8037)**. This combination of high accuracy and recall makes Random Forest the best performer in terms of fraud detection in imbalanced datasets.

The **Hybrid Voting Classifier** also performed strongly, achieving an **F1-score of 0.838** and an **AUC-ROC of 0.9840**, demonstrating the benefit of combining multiple models to balance precision and recall effectively.

In contrast, **Logistic Regression** consistently underperformed, especially in terms of **recall (0.042)**, indicating that it struggles with detecting fraudulent transactions in highly imbalanced datasets.

The **XGBoost** model demonstrated strong performance with an **accuracy of 0.9399**, an **AUC of 0.9609**, and a competitive **recall of 0.8251**, showcasing its effectiveness in handling imbalanced data while capturing complex fraud patterns.

Visualization Techniques:

To provide a visual understanding of model performance, the following techniques were used:

Confusion Matrices: These matrices offered a clear breakdown of True Positives, True Negatives, False Positives, and False Negatives, allowing for a detailed understanding of each model's classification accuracy.

ROC Curves: ROC curves for each model were plotted to assess their discriminatory power. **Random Forest** achieved the highest AUC, followed by the **Hybrid Voting Classifier**.

Precision-Recall Curves: PR curves were used to evaluate how well each model balanced precision and

recall, particularly in the case of highly imbalanced datasets.

4.2 ROC Curves and AUC Scores

Receiver Operating Characteristic (ROC) curves were generated for each model to demonstrate its capacity to distinguish between fraudulent and legal transactions.

Figure 2 displays the ROC curves for all four models.

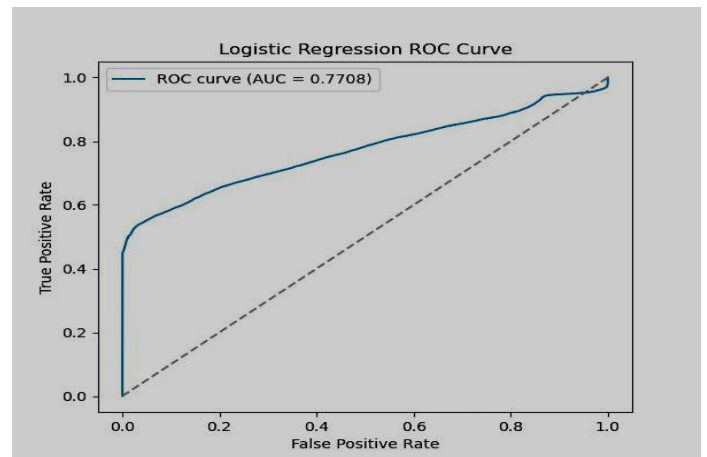
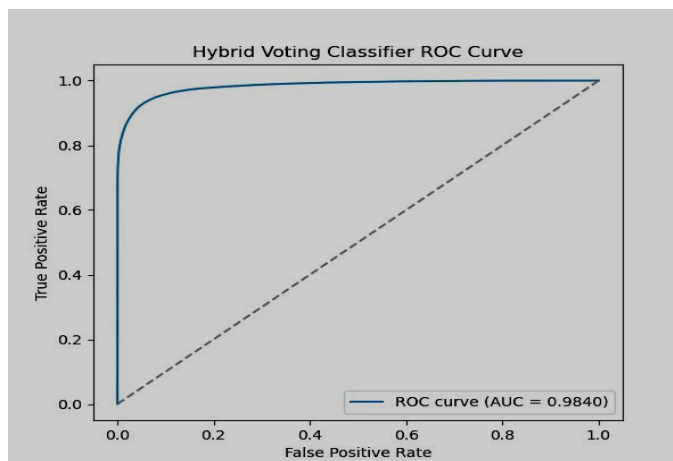
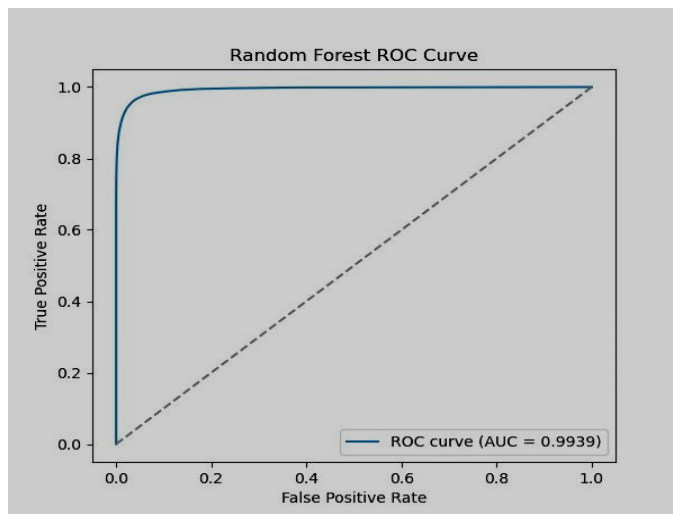
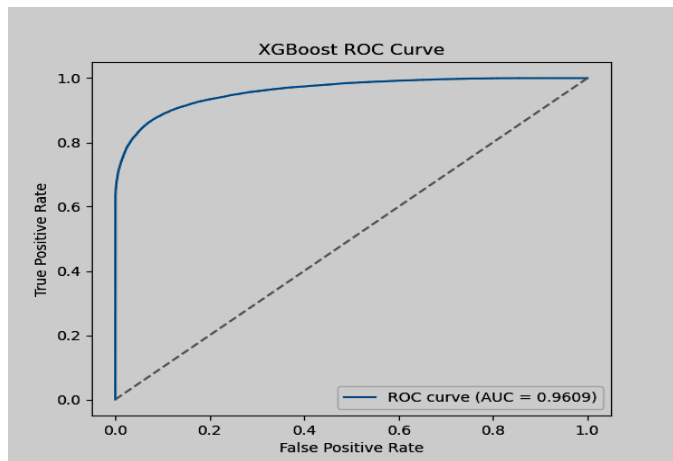
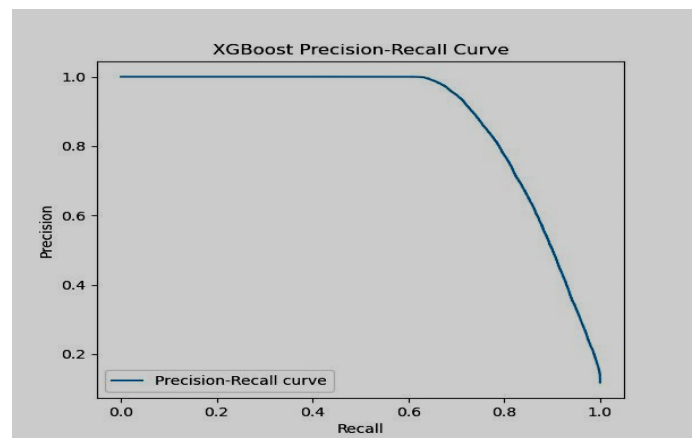


Figure 2. ROC Curves of Random Forest, XGBoost, Logistic Regression, and Hybrid Voting Classifier

The Random Forest model had the highest Area Under the Curve (AUC) score of 0.9939, showing almost flawless classification between classes. The Hybrid Voting Classifier came in close second with an AUC of 0.9840, indicating strong ensemble performance. XGBoost also fared well, with an AUC of 0.9609, whilst Logistic Regression lagged with an AUC of 0.7708, indicating lesser classification performance. These charts illustrate the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR), emphasizing the superior performance of ensemble-based and tree-based models in this real-time fraud detection system.

4.3 Precision-Recall Curves

To further understand model performance in scenarios with class imbalance, Precision-Recall (PR) curves were examined. Figure 3 presents the PR curves for all models. These curves plot Precision against Recall, emphasizing the models' ability to correctly identify fraudulent transactions while minimizing false alarms.



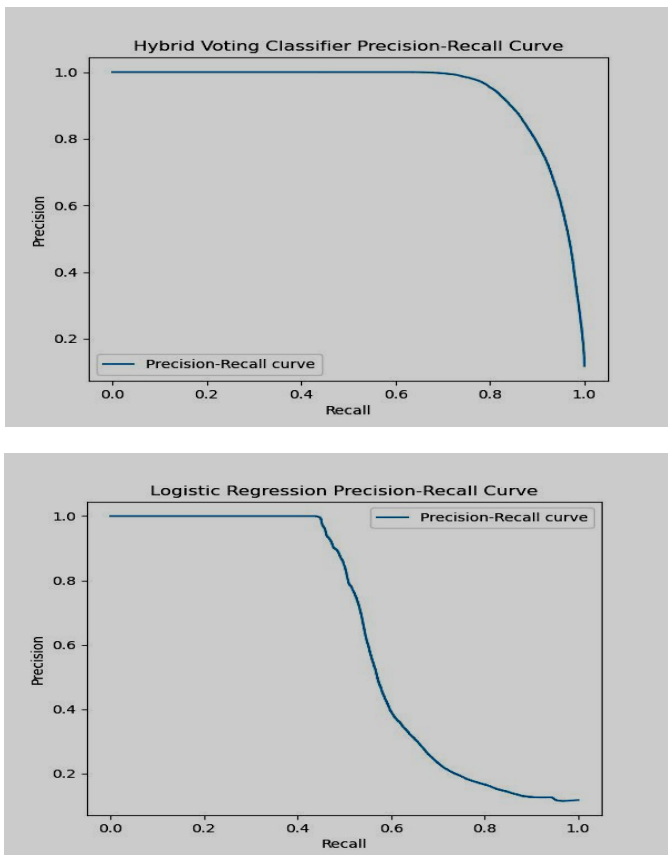


Figure 3. Precision-Recall Curves of Random Forest,

XGBoost, Logistic Regression, and Hybrid Voting Classifier. The Random Forest and Hybrid Voting Classifier maintained high Precision across varying Recall values, illustrating their robustness in handling the imbalanced dataset, where fraud constitutes only a small fraction of total transactions. Logistic Regression, however, showed a steeper drop in Precision at higher Recall values, suggesting limited reliability in capturing all fraudulent instances without incurring more false positives.

4.4 Confusion Matrices

Confusion matrices were generated to provide a detailed breakdown of model predictions, offering a direct view of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

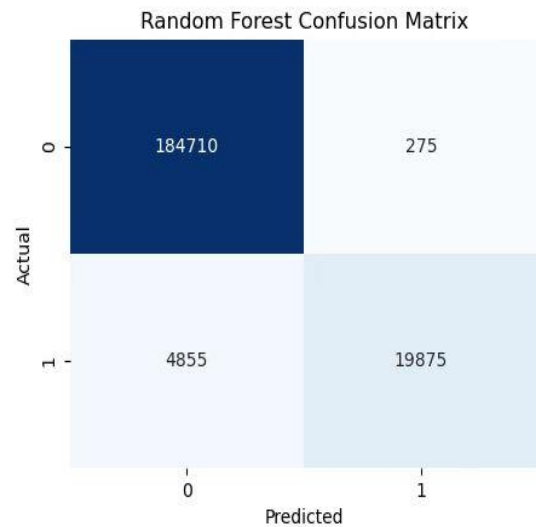


Figure 4. confusion Matrix of Random Forest Model

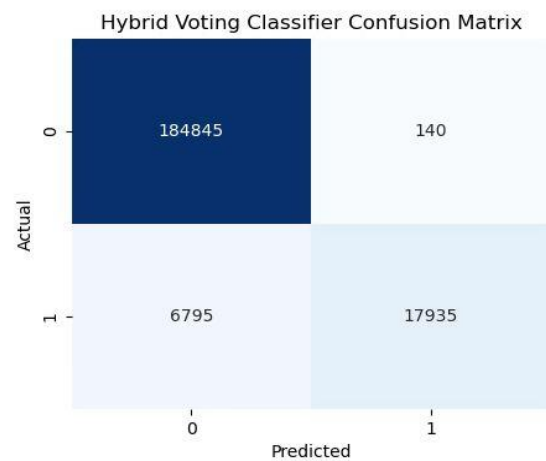
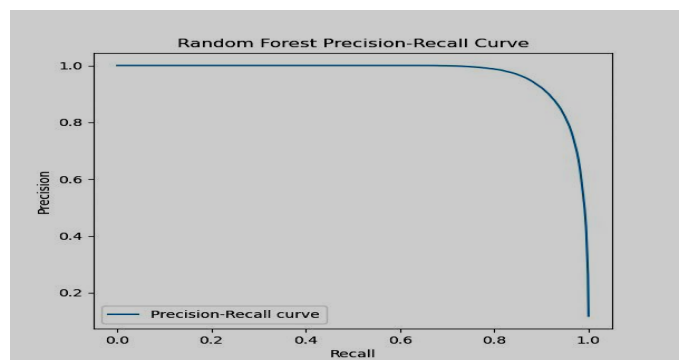


Figure 5. Confusion Matrix of Hybrid Voting Classifier



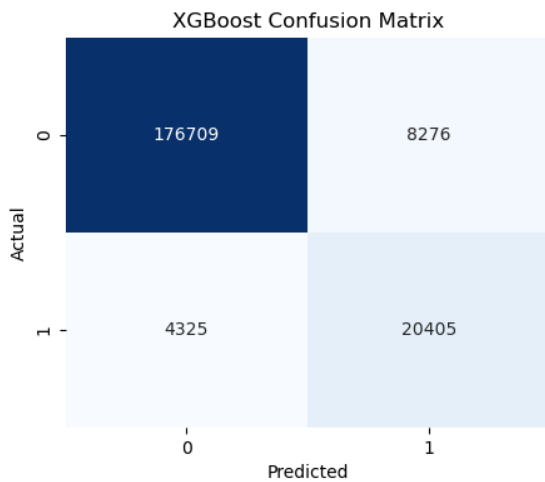


Figure 6. Confusion Matrix of XGBoost

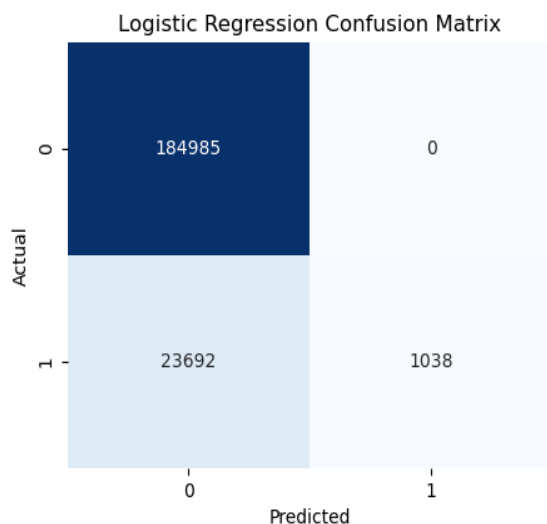


Figure 7. Confusion Matrix of Logistic Regression

Random Forest Confusion Matrix

- True Positives (TP): 19,875
- True Negatives (TN): 184,710
- False Positives (FP): 275
- False Negatives (FN): 4,855

The **Random Forest** model achieved strong performance with minimal misclassification, maintaining a low **false positive rate** and **false negative rate**.

Hybrid Voting Classifier Confusion Matrix

- True Positives (TP): 17,935
- True Negatives (TN): 184,845
- False Positives (FP): 140

- False Negatives (FN): 6,795

While the Hybrid Voting Classifier performed well in classification, it had a slightly higher false negative count than Random Forest, highlighting the inherent trade-offs in ensemble models. Nonetheless, it efficiently balances fraud detection with low false alarms.

XGBoost Confusion Matrix

- True Positives (TP): 20,405
- True Negatives (TN): 176,709
- False Positives (FP): 8276
- False Negatives (FN): 4325

XGBoost demonstrates strong fraud detection capability with high true positives, but generates more false positives, reflecting a trade-off between recall and precision in imbalanced data scenarios.

Logistic Regression Confusion Matrix

- True Positives (TP): 1038
- True Negatives (TN): 184,985
- False Positives (FP): 0
- False Negatives (FN): 23,692

The **Logistic Regression** achieved perfect precision but extremely low recall, missing most fraudulent transactions, making it unsuitable for imbalanced fraud detection tasks.

4.5 Summary of Model Performance

To provide a clear comparison summary of all models, Table 1 includes essential evaluation variables such as Accuracy, Precision, Recall, F1-Score, and AUC-ROC. Random Forest emerged as the best-performing model, with the greatest AUC (0.9939) and good Precision (0.9864) and Recall (0.8037). This combination has robust detection capacity with low misclassification, making it ideal for real-time fraud detection in imbalanced datasets.

The Hybrid Voting Classifier also performed well, using ensemble learning to successfully balance Precision and Recall. While its Recall (0.7252) was significantly lower than Random Forest's, its excellent Precision (0.9923) provides accurate detection of fraudulent transactions with few false alarms. XGBoost, while slightly behind in overall performance, maintained competitive accuracy (0.9399) and AUC-ROC (0.9609), demonstrating good predictive skills, particularly for catching complicated non-linear patterns in data.

Logistic Regression, on the other hand, persistently underperformed across every measure, with a particularly poor Recall (0.042) despite excellent Precision (1.0). This demonstrates its limits in detecting fraudulent transactions in highly imbalanced datasets, stressing the superiority of tree-based and ensemble techniques for operational use.

Table 1. Comparative Evaluation of Fraud Detection Models

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC	Notes on Performance
Random Forest	0.9755	0.9864	0.8037	0.8857	0.9939	Best overall, minimal misclassification
Hybrid Voting Classifier	0.9669	0.9923	0.7252	0.8380	0.9840	Strong ensemble performance, balances Precision/Recall
XGBoost	0.9399	0.7114	0.8251	0.7641	0.9609	Competitive accuracy, captures complex patterns
Logistic Regression	0.8870	1.0000	0.0420	0.0806	0.7708	Underperforms on imbalanced data, higher misclassification

These results collectively indicate that tree-based and ensemble models, particularly Random Forest and Hybrid Voting Classifier, are better suited for real-time fraud detection tasks. Their performance highlights not only accuracy but also interpretability and operational reliability, making them optimal candidates for deployment in high-volume financial transaction monitoring systems.

5: CONCLUSION

In this paper, we present a scalable and efficient fraud detection system for financial transactions that combines Apache Kafka for real-time data streaming with machine learning models such as Logistic Regression, Random Forest, XGBoost, and a Hybrid Voting Classifier. The system uses ensemble learning to integrate the strengths of separate models, resulting in a better balance of precision and recall. According to the experimental results, Random Forest outperforms the other models with 97.6% accuracy, 0.8037 recall, and an AUC-ROC of 0.9939, making it particularly useful for detecting fraud in imbalanced datasets.

The Hybrid Voting Classifier, while partially trailing Random Forest in Recall (0.7252), achieves a high Precision (0.9923), highlighting the benefits of ensemble models in minimizing false alarm rates. XGBoost, with 93.99% accuracy and an AUC-ROC of 0.9609, strikes a great balance between accuracy and recall, demonstrating its effectiveness in collecting complicated patterns, particularly in imbalanced fraud detection tasks. Logistic Regression, on the other hand, had substantial limits, with a recall of 0.042, underperforming tree-based models, reflecting the difficulties in addressing fraud detection in highly imbalanced datasets.

Our Kafka-driven pipeline guarantees low latency transaction processing, making it ideal for high-frequency transaction scenarios. The system's scalability enables real-time fraud detection without causing substantial

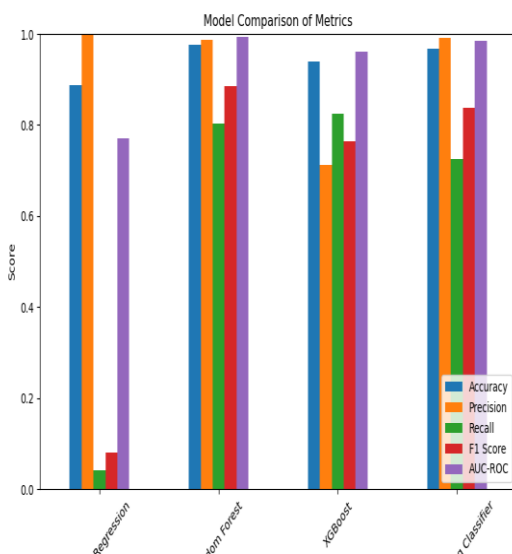


Figure 8. Comparative Performance of Fraud Detection Models Across Evaluation Metrics

Figure 8 presents a comparative visualization of the performance metrics for all evaluated models. Random Forest and the Hybrid Voting Classifier achieve the best overall performance, with high accuracy, AUC-ROC, and balanced precision and recall. XGBoost demonstrates strong recall but lower precision, while Logistic Regression shows perfect precision but extremely low recall, confirming its limitations in handling imbalanced datasets.

delays, which is critical for financial systems that handle high numbers of transactions.

Comparing our approach with prior works, we show that while previous models achieved high **precision**, they often suffered from low **recall** or **latency** issues, making them less suitable for real-time applications. Additionally, many previous models lacked the ability to scale for high-volume transaction environments. Our system, by contrast, offers both **high predictive accuracy** and **scalability**, making it a robust and efficient solution for real-time fraud detection.

REFERENCES

- [1]. Sharma, S. (2016). A detail comparative study on e-banking VS traditional banking. *International Journal of Applied Research*, 2(7), 302-307.
- [2]. Xu, H. Y. (2017). China's Internet Financial Risks and Countermeasures. In *International Conference on Financial Management, Education and Social Science (FMES 2017)*.
- [3]. Akinyede, R. O., & Esese, O. A. (2017). Development of a secure mobile e-banking system. *International Journal of Computer (IJC)*, 26(1), 23-42.
- [4]. Reaves, B., Scaife, N., & Bates, A. (2015). Analysis of Branchless Banking Applications in the Developing World. In *24th USENIX Security Symposium*.
- [5]. Rajput, Q., Khan, N. S., Larik, A., & Haider, S. (2014). Ontology based expert-system for suspicious transactions detection. *Computer and Information Science*, 7(1), 103.
- [6]. Quah, J. T., & Sriganesh, M. (2008). Real-time credit card fraud detection using computational intelligence. *Expert systems with applications*, 35(4), 1721-1732.
- [7]. Abdelhamid, D., Khaoula, S., & Atika, O. (2014, January). Automatic bank fraud detection using support vector machines. In *The International Conference on Computing Technology and Information Management (ICCTIM)* (p. 10). Society of Digital Information and Wireless Communication.
- [8]. Melo-Acosta, G. E., Duitama-Munoz, F., & Arias-Londono, J. D. (2017, August). Fraud detection in big data using supervised and semi-supervised learning techniques. In *2017 IEEE Colombian conference on communications and computing (COLCOM)* (pp. 1-6). IEEE.
- [9]. Baesens, B., Van Vlasselaer, V., & Verbeke, W. (2015). *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. John Wiley & Sons.
- [10]. Ozohu, M. M., & Uchenna, O. L. (2021). Comparative Analysis of Selected Machine Learning Algorithms Based On Generated Smart Home Dataset. *European Journal of Computer Science and Information Technology*, 9(4), 42-53.
- [11]. Niu, X., Wang, L., & Yang, X. (2019). A comparison study of credit card fraud detection: Supervised versus unsupervised. *arXiv preprint arXiv:1904.10604*.
- [12]. Teguh, S., Muhammad Rafli, A., Haldi, B., M Rezqy, N. R., Usman, S., & Noor, A. (2024). Comparison of Logistic Regression, Random Forest, SVM, KNN Algorithm for Water Quality Classification Based on Contaminant Parameters. *Journal of Data Science*, 2024(48), 1-7.
- [13]. Carcillo, F., Dal Pozzolo, A., Le Borgne, Y. A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). Scarff: a scalable framework for streaming credit card fraud detection with spark. *Information fusion*, 41, 182-194.
- [14]. Araujo, M., Almeida, M., Ferreira, J., Silva, L., & Bizarro, P. (2017, June). Breachradar: Automatic detection of points-of-compromise. In *Proceedings of the 2017 SIAM International Conference on Data Mining* (pp. 561-569). Society for Industrial and Applied Mathematics.
- [15]. Branco, B., Abreu, P., Gomes, A. S., Almeida, M. S., Ascensão, J. T., & Bizarro, P. (2020, August). Interleaved sequence RNNs for fraud detection. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 3101-3109).
- [16]. Hong, X., Zheng, C., & Zilberman, N. (2024). In-network machine learning for real-time transaction fraud detection.
- [17]. Almalki, F., & Masud, M. (2025). Financial Fraud Detection Using Explainable AI and Stacking Ensemble Methods. *arXiv preprint arXiv:2505.10050*.
- [18]. Yelleti, V. (2025). ROSFD: Robust Online Streaming Fraud Detection with Resilience to Concept Drift in Data Streams. *arXiv preprint arXiv:2504.10229*.
- [19]. Ikermane, M., Mohy-eddine, M., & Rachidi, Y. (2024, December). Credit Card Fraud Detection: Comparing Random Forest and XGBoost Models with Explainable AI Interpretations. In *International Conference on Electrical Systems and Smart Technologies* (pp. 126-135). Cham: Springer Nature Switzerland.
- [20]. Sundaravadivel, P., Isaac, R. A., Elangovan, D., KrishnaRaj, D., Rahul, V. L., & Raja, R. (2025). Optimizing credit card fraud detection with random forests and SMOTE. *Scientific Reports*, 15(1), 17851.
- [21]. Theodorakopoulos, L., Theodoropoulou, A., Tsimakis, A., & Halkiopoulos, C. (2025). Big data-driven distributed machine learning for scalable

- credit card fraud detection using PySpark, XGBoost, and CatBoost. *Electronics*, 14(9), 1754.
- [22]. Khatri, S., Arora, A., & Agrawal, A. P. (2020, January). Supervised machine learning algorithms for credit card fraud detection: a comparison. In 2020 10th international conference on cloud computing, data science & engineering (confluence) (pp. 680-683). IEEE.
- [23]. Liu, C., Tang, H., Yang, Z., Zhou, K., & Cha, S. (2025). Big Data-Driven Fraud Detection Using Machine Learning and Real-Time Stream Processing. arXiv preprint arXiv:2506.02008.
- [24]. Zhu, M., Zhang, Y., Gong, Y., Xu, C., & Xiang, Y. (2024). Enhancing credit card fraud detection a neural network and smote integrated approach. arXiv preprint arXiv:2405.00026.
- [25]. Singh, G., Singh, P., & Singh, M. (2025). Advanced Real-Time Fraud Detection Using RAG-Based LLMs. arXiv preprint arXiv:2501.15290.
- [26]. Sheng, S., & Ling, C. X. (2005, October). Hybrid cost-sensitive decision tree. In European conference on principles of data mining and knowledge discovery (pp. 274-284). Berlin, Heidelberg: Springer Berlin Heidelberg.
- [27]. Ahmadi, S. (2023). Open AI and its impact on fraud detection in financial industry. *Journal of Knowledge Learning and Science Technology* ISSN: 2959-6386 (Online), 2(3), 263-281.
- [28]. Islam, R., Mazumdar, S., & Islam, R. (2024, May). An Experiment on Feature Selection Using Logistic Regression. In 2024 5th Information Communication Technologies Conference (ICTC) (pp. 319-324). IEEE.