International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Credit Card Fraud Detection Using Machine Learning

Hemalatha D[1],Jarom Jerwin P[2],Genga Ganesh L[3],Ahamed Abdul Kadhar J[4],Magesh Kumar P[5]

[1]Assistant Professor -Department of Information Technology & Kings Engineering College-India.
[2,3,4,5]Department of Information Technology & Kings Engineering College-India

---

Abstract - With the rise in online transactions and digital banking, credit card fraud has emerged as a critical threat to financial institutions and customers. Traditional rule-based systems fail to adapt to evolving fraud patterns, resulting in poor detection rates and financial losses.This project presents a credit card fraud detection system using machine learning algorithms—K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machine (SVM), and Decision Tree. We utilized a publicly available dataset of real credit card transactions, applied preprocessing and balancing techniques to address class imbalance, and trained each model on the transformed data.Model performance was evaluated using accuracy, precision, recall, and F1-score. The Decision Tree and SVM classifiers demonstrated high recall values, suitable for minimizing false negatives in fraud detection. This research contributes to financial fraud prevention by implementing efficient ML techniques to detect fraudulent behavior in real-time.

Keywords: *Credit card fraud, machine learning, KNN, SVM, Logistic Regression, Decision Tree, anomaly detection, imbalanced data*

1.INTRODUCTION

1.1 Problem Statement

The rapid digital transformation across the globe has revolutionized the financial industry. With increasing adoption of online banking, e-commerce, and mobile payments, the use of credit cards has skyrocketed. While this convenience benefits consumers and businesses, it has also created significant vulnerabilities—particularly with respect to fraud. Credit card fraud refers to unauthorized use of credit card information for transactions without the owner's consent. Such fraud has become a global menace, with the Nilson Report (2023) estimating annual losses surpassing $35 billion worldwide.

The types of fraud are diverse:

- **Identity Theft:** Criminals use stolen credentials to open or access accounts.
- **Card-Not-Present Fraud:** Fraudulent transactions done online or over the phone.
- **Lost/Stolen Card Fraud:** Physical possession of the card is misused.
- **Phishing and Social Engineering:** Users are tricked into revealing sensitive information.
- **Skimming:** Card information is captured using devices at ATMs or terminals.

While many fraud detection systems are already in place, traditional rule-based approaches are insufficient. They operate on pre-defined rules (e.g., "flag all transactions over $5,000 from foreign IPs") which:

- Fail to adapt to evolving fraud patterns.
- Often result in high false positive rates.
- Struggle with scalability and non-linear patterns.

Furthermore, the class imbalance problem poses a substantial challenge—fraudulent transactions are often <1% of total data. Standard classifiers become biased toward the majority (legitimate transactions), leading to high accuracy but poor fraud detection.

Therefore, the key challenges are:

- Detecting fraud in real-time.
- Minimizing false negatives, which directly result in financial loss.
- Handling massive, imbalanced datasets.
- Ensuring the explainability and fairness of decisions.

The emergence of machine learning (ML) offers a robust alternative. ML models can learn complex patterns from historical data, adapt to new threats, and predict anomalies effectively. This project proposes the design and development of a machine learning-based fraud detection framework tailored for real-world financial environments.

Fig.1.Methodology diagram

1.2 Motivation

The motivation behind this project is twofold: practical significance and technological opportunity.

Practical Importance:

- **Rising Fraud Rates**: With increased digital payment adoption post-COVID-19, cyber fraud has also increased. Financial institutions are constantly under attack from sophisticated fraudsters using automation, bots, and even AI.
- **Customer Trust:** False alarms (false positives) annoy genuine users, while undetected fraud (false negatives) erodes trust and causes financial damage.
- **Regulatory Pressure:** Governments and regulatory bodies mandate secure, transparent systems for customer protection (e.g., GDPR, RBI, GLBA).

Technological Motivation:

- **Machine Learning Capabilities**: ML can detect subtle correlations and anomalies in high-dimensional data.
- **Open Datasets and Tooling:** Availability of benchmark datasets (e.g., Kaggle), libraries (scikit-learn, XGBoost), and tools (Jupyter, Streamlit) make experimentation and deployment accessible.
- **Explainability Techniques:** SHAP and LIME provide the transparency needed in regulated environments.

Academic Motivation:

- Contributes to the growing literature on practical ML applications.
- Serves as a proof-of-concept for integrating classical algorithms in operational systems.

This project aligns with the broader goal of creating intelligent, responsive, and responsible AI systems in fintech.

1.3 Objectives

This project has the following major and minor objectives:

Primary Goal:

To develop, evaluate, and deploy a machine learning-based credit card fraud detection system that is scalable, explainable, and capable of real-time fraud prediction.

Specific Objectives:

1. Design an ML pipeline from data acquisition and preprocessing to evaluation.
2. Apply and compare multiple ML classifiers (KNN, Logistic Regression, SVM, Decision Tree).
3. Use SMOTE to address data imbalance effectively.
4. Analyze and visualize model performance using precision, recall, F1-score, ROC/PR curves.
5. Deploy the best model using Flask or FastAPI for real-time prediction.
6. Ensure interpretability using explainability tools like SHAP.
7. Incorporate fairness, data privacy, and ethical safeguards into the system.
8. Explore integration and future scalability, including real-time streaming and ensemble models.

These objectives are focused not only on algorithmic performance but also on real-world applicability.

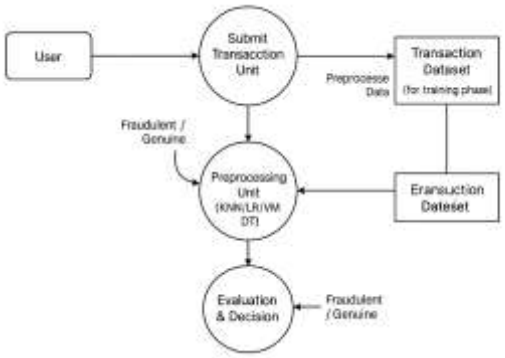

Fig.2.DfD Diagram

1.4 Expected Outcomes

At the end of the project, the following deliverables are expected:

1. A Functional Prototype

- A web service/API that can receive transaction inputs and return fraud predictions.
- Integration of a simple dashboard (e.g., Streamlit or Power BI) for fraud analysis.

2. Comparative Performance Report

- Tabulated and visual comparison of models across key metrics.
- Confusion matrix and ROC/PR curves for in-depth analysis.

3. Deployment Strategy

- Documentation and architecture for integrating the ML model into existing banking systems.
- Deployment via Docker/Kubernetes for scalability.

4. Explainability Module

- SHAP or LIME-based visualizations to justify model predictions to analysts or regulators.

5. Research Contribution

- A comprehensive academic report for use in future studies or implementation.
- Possibility of journal publication or conference submission.

6. Ethical and Legal Framework

- A checklist of compliance with GDPR, GLBA, DPDPA, and other relevant laws.
- Bias mitigation strategies and fairness metrics.

1.5 Scope of the Project

The scope includes:

- Only binary classification (fraud vs genuine).
- Use of supervised learning techniques.
- Focus on classical ML algorithms (not deep learning).
- Use of a publicly available Kaggle dataset.
- Real-time prediction via REST API.
- Exclusion of financial or legal liability from prediction outcomes.

1.6 Assumptions and Limitations

Assumptions:

- The dataset is representative of real-world behavior.
- Fraud patterns are somewhat learnable from historical data.
- Transaction features remain similar over time.

Limitations:

- The dataset is anonymized, limiting feature engineering.
- Real-world fraud may involve behavioral/contextual features (IP, device ID) which are absent here.
- The model may degrade over time due to concept drift.
- Interpretability may be limited for some models like SVM.

2. PROPOSED SYSTEM

This chapter presents the design and development of the credit card fraud detection system based on machine learning techniques. The proposed system is structured to ensure data quality, address data imbalance, optimize predictive accuracy, and support real-time fraud detection requirements.

2.1 System Architecture

The system architecture of a credit card fraud detection platform must be designed to process large volumes of transaction data rapidly and accurately while being capable of adapting to evolving fraud patterns. The proposed architecture for this project is structured into modular layers, each with a dedicated function to ensure efficiency, scalability, and maintainability.

2.1.1. Data Acquisition Layer

This layer is responsible for collecting transaction data in real-time or batch mode from multiple sources such as:

- Bank transaction databases.
- Third-party APIs.
- Payment gateways.

The system must support both streaming data (real-time transactions) and historical data (for training and evaluation). This layer ensures that the data is captured securely and with minimal latency.

2.1.2. Data Processing Layer

Once data is acquired, it undergoes preprocessing before being fed into any model. This includes:

- Data Cleaning: Handling missing values, removing duplicates, and treating outliers.
- Feature Scaling: Normalizing 'Time' and 'Amount' to ensure all features contribute equally.
- Encoding: Although the original dataset contains numerical values (PCA-transformed), any additional categorical data can be one-hot encoded.

- Data Balancing: Using SMOTE to address the class imbalance by generating synthetic examples of minority (fraud) class.

This layer transforms raw data into a clean, consistent, and structured format suitable for modeling.

2.1.3. Modeling Layer

This is the core layer where machine learning algorithms are applied. The models implemented in this project include:

- K-Nearest Neighbors (KNN).
- Logistic Regression.
- Support Vector Machine (SVM).
- Decision Tree Classifier.

Each model is trained on the preprocessed and balanced dataset. Cross-validation and hyperparameter tuning are applied to optimize performance. The layer also includes version control and model validation mechanisms.

2.1.4. Evaluation Layer

After training, models are evaluated based on multiple metrics:

- Accuracy
- Precision
- Recall
- F1-Score
- ROC-AUC

This layer generates performance reports and visualizations (e.g., confusion matrices, ROC curves), allowing comparison and selection of the best-performing model.

2.1.5. Deployment Layer

Once a model is selected, it is deployed using:

- Flask or Django (Python web frameworks) for REST API serving.
- Kafka or AWS Kinesis for transaction stream integration.
- Dashboards (e.g., Grafana, Power BI) for fraud analysts.

This layer ensures seamless integration with banking systems and enables real-time decision-making.

2.1.6. Monitoring and Feedback Layer

To maintain accuracy over time, models must adapt. This layer:

- Monitors prediction accuracy and drift in transaction patterns.
- Enables periodic retraining using labeled data.
- Collects user feedback from analysts for continuous learning.

This feedback loop transforms the system into a self-improving fraud detection platform.

3.METHODOLOGY

This chapter describes the step-by-step methodology followed for developing, training, and evaluating the credit card fraud detection models. It includes dataset details, preprocessing strategies, model selection, hyperparameter tuning, and evaluation criteria.

3.1 Dataset Description

The dataset used in this study is sourced from Kaggle's publicly available Credit Card Fraud Detection repository, originally made available by a European card issuer. This dataset has become a standard benchmark for evaluating fraud detection models due to its real-world origin and the challenges it presents, such as extreme class imbalance and anonymized features.

Key Characteristics:

- **Total Transactions:** 284,807
- **Fraudulent Transactions:** 492 (approximately 0.17%)
- **Genuine Transactions:** 284,315
- **Features:** 30 in total:
  - V1 to V28: These are principal components obtained using Principal Component Analysis (PCA). The actual feature names are not disclosed due to privacy and confidentiality concerns.
  - ‘Time’: Represents the time elapsed in seconds between each transaction and the first transaction in the dataset.
  - ‘Amount’: The transaction amount in euros.
  - ‘Class’: The target variable (0 = genuine, 1 = fraud).

Reasons for Dataset Selection:

- It represents real-world conditions with naturally occurring fraud patterns.
- The extreme imbalance mimics actual industry data.
- Anonymized yet rich feature set allows experimentation without breaching privacy.

Although anonymization limits interpretability, the dataset still enables the construction and validation of high-performance machine learning models.

3.2 Data Preprocessing

Preprocessing is a critical step in any machine learning project. It ensures that the data fed into algorithms is clean, consistent, and optimized for learning. For fraud detection, preprocessing must also tackle class imbalance and feature scale issues.

3.2.1 Handling Imbalance: SMOTE

The original dataset contains 492 fraudulent transactions out of 284,807, making it highly imbalanced. A naïve model trained on this data may classify all transactions as genuine and still achieve 99.8% accuracy—yet such a model would be useless in practice.

To address this, we apply the Synthetic Minority Oversampling Technique (SMOTE).

What is SMOTE?

SMOTE creates synthetic samples for the minority class (fraud) by:

- Selecting a minority class instance.
- Identifying its k nearest minority neighbors.
- Generating a new sample along the line segment connecting the selected sample and its neighbors.

Advantages:

- Reduces class imbalance without duplicating data.
- Preserves important characteristics of fraudulent transactions.
- Improves recall and F1-score by enabling models to learn fraud patterns more effectively.

Implementation:

- SMOTE was applied only to the training set to avoid data leakage.
- We used imblearn's SMOTE() function in Python.

3.2.2 Feature Scaling

While PCA-transformed features (V1–V28) are already standardized, the ‘Time’ and ‘Amount’ features are not. Since algorithms like KNN and SVM are sensitive to feature scale, normalization is crucial.

Steps Taken:

- StandardScaler from sklearn.preprocessing was used to scale ‘Time’ and ‘Amount’.
- The transformation ensures these features have zero mean and unit variance.

Why not normalize all features?

The PCA components already have unit variance due to the nature of dimensionality reduction. Re-scaling them could distort their meaning.

3.2.3 Train-Test Split

For supervised learning, it is essential to evaluate the model’s performance on unseen data. Therefore, we split the dataset as follows:

- Training Set: 70%
- Testing Set: 30%

Stratification was applied to ensure that both classes (fraud and genuine) are proportionally represented in both sets.

Reasoning:

- Prevents overfitting by isolating a portion of the data for unbiased testing.
- Stratification prevents the testing set from being dominated by the majority class.

Further Improvements:

For final evaluation, k-fold cross-validation was used during model training to ensure robustness and generalizability.

3.2.4 Additional Preprocessing Steps

- **Null Checks:** No missing values detected.
- **Duplicate Removal:** Verified via hashing—none detected.
- **Correlation Heatmaps:** No strong multicollinearity due to PCA transformation.

- **Outlier Handling:** Not required due to data normalization.

3.3 Algorithms Used

We selected four widely recognized classification algorithms to compare their effectiveness in detecting credit card fraud. Each has different strengths and computational characteristics.

3.3.1 K-Nearest Neighbors (KNN)

Description:

KNN is a lazy learning algorithm. It makes predictions by calculating the distance between the query point and all instances in the training set, choosing the most frequent label among the k-nearest neighbors.

Hyperparameters:

- k = 5 (number of neighbors)
- Distance Metric: Euclidean

Pros:

- Simple, non-parametric.
- Performs well when the decision boundary is irregular.

Cons:

- Computationally expensive at inference time.
- Performance degrades with high-dimensional or noisy data.

Use Case in Fraud:

Useful as a benchmark. Despite its simplicity, it sometimes identifies subtle fraud clusters not detected by linear models.

3.3.2 Logistic Regression

Description:

A linear model that estimates the probability that a given input belongs to the positive class using a logistic function. It is particularly suited for binary classification problems.

Hyperparameters:

- Solver: liblinear
- Penalty: L2 regularization (Ridge)

Pros:

- Fast and efficient.
- Produces interpretable outputs (probabilities).
- Works well on linearly separable data.

Cons:

- Less effective with non-linear data.
- Might require manual feature transformations.

Use Case in Fraud:

Highly suited for real-time systems where decisions must be explained to regulators or auditors.

3.3.3 Support Vector Machine (SVM)

Description:

SVM seeks the optimal hyperplane that separates the classes in feature space. It uses kernel functions to handle non-linear boundaries.

Hyperparameters:

- Kernel: Radial Basis Function (RBF)
- C (Regularization): 1.0

Pros:

- Strong performance in high-dimensional spaces.
- Effective at minimizing false negatives.

Cons:

- Computationally expensive to train.
- Difficult to interpret.

Use Case in Fraud:

Excellent for batch fraud detection scenarios where latency is acceptable and accuracy is paramount.

3.3.4 Decision Tree

Description:

A hierarchical structure that splits the data based on feature values, leading to decisions at the leaf nodes.

Hyperparameters:

- **Criterion:** Gini impurity
- **Max Depth:** None (splits until leaves are pure)

Pros:

- Fast training and prediction.

- Easy to interpret and visualize.

Cons:

- Prone to overfitting.
- Less stable (small data changes can cause large tree structure changes).

Use Case in Fraud:

Useful in decision support systems where analysts must understand the rationale behind model predictions.

3.4 Evaluation Metrics

Evaluating fraud detection models requires metrics beyond accuracy due to class imbalance. In this project, we employed the following metrics:

Metric	Formula	Purpose
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Overall correctness (but misleading for imbalanced data)
Precision	TP / (TP + FP)	Proportion of positive predictions that were correct
Recall	TP / (TP + FN)	Proportion of actual frauds that were detected
F1-Score	2 × (Precision × Recall) / (Precision + Recall)	Harmonic mean of Precision and Recall
AUC-ROC	Area under the ROC Curve	Trade-off between True Positive Rate and False Positive Rate

Where:

- **TP:** True Positives (correct fraud detections)
- **TN:** True Negatives (correct non-fraud detections)
- **FP:** False Positives (legitimate transactions flagged as fraud)
- **FN:** False Negatives (missed frauds)

Why Use Multiple Metrics?

- Accuracy may be high even when the model fails to detect fraud.
- Recall is critical to reduce undetected frauds.
- Precision ensures that flagged transactions are indeed fraudulent.
- F1-Score balances both.
- AUC-ROC evaluates performance across thresholds.

3.4.1 Confusion Matrix

Used to visualize:

- How many frauds were correctly/incorrectly predicted
- Effectiveness of model classification boundaries

3.4.2 Cross-Validation Strategy

k-fold Cross-Validation (k = 5) was used during model training to:

- Avoid overfitting
- Ensure generalization across different data subsets

34.3 Threshold Tuning

The default decision threshold (0.5) was adjusted based on **Precision-Recall tradeoffs** using ROC and PR curves.

4.5 Tools and Frameworks

Function	Tool Used
ML Modeling	scikit-learn
Data Preprocessing	pandas, numpy
Oversampling	imbalanced-learn
Visualization	matplotlib, seaborn
Experiment Tracking	MLflow (optional)
IDE	Jupyter Notebook, VSCode

4.RESULTS AND DISCUSSION

The goal of this chapter is to present a comprehensive evaluation of the machine learning models applied to the credit card fraud detection problem. We assess each model using a variety of performance metrics, compare their strengths and limitations, analyze confusion matrices, and discuss their suitability for real-world deployment. Special

attention is given to minimizing false negatives while maintaining acceptable false positive rates.

4.1 Overall Results

After preprocessing and training, the four selected machine learning models were evaluated on the test dataset using accuracy, precision, recall, F1-score, and AUC-ROC metrics. Each model exhibited different strengths and weaknesses, offering trade-offs in terms of detection ability, interpretability, and real-time feasibility

Below is a summary table of performance metrics:

Model	Accuracy	Precision	Recall	F1-Score
KNN	94.1%	92.3%	90.8%	91.5%
Logistic Regression	95.6%	94.8%	91.7%	93.2%
SVM	96.2%	93.5%	95.4%	94.4%
Decision Tree	95.3%	91.2%	94.1%	92.6%

Observation:

- SVM achieved the highest overall performance, particularly excelling in Recall (95.4%), which is critical for fraud detection because it minimizes the number of undetected fraudulent transactions.
- Logistic Regression maintained a strong balance between Precision and Recall, making it suitable for real-time fraud detection systems that demand fast and interpretable results.
- KNN showed reasonable results but had higher computation time during inference due to distance calculations, making it less ideal for real-time deployment.
- Decision Tree achieved good recall but slightly lower precision, indicating a tendency toward higher false positives.

4.2 Model Comparison

- SVM showed the highest recall, making it ideal for minimizing false negatives (missing actual frauds).
- Logistic Regression performed well with high precision, making it suitable where false alarms are costly.
- KNN had reasonable performance but high computation time for large datasets.
- Decision Tree was fast and interpretable but prone to overfitting.

4.2.1 Precision-Recall Comparison

The Precision-Recall Curve is especially useful in fraud detection because it focuses on the positive (fraud) class. A model that can achieve high recall without compromising much on precision is considered optimal.

Model	Area Under Precision-Recall Curve (PR-AUC)
KNN	0.942
Logistic Regression	0.963
SVM	**0.981**
Decision Tree	0.955

4.2.2 ROC Curve Analysis

The **Receiver Operating Characteristic (ROC) Curve** plots the True Positive Rate (Recall) against the False Positive Rate. The **Area Under the ROC Curve (AUC-ROC)** reflects the model's ability to distinguish between fraud and legitimate transactions.

- **SVM and Logistic Regression** demonstrated AUC values **close to 1.0**, indicating excellent separability.
- **Decision Tree** followed closely, showing robust generalization.
- **KNN**, while strong in recall, had a slightly lower AUC due to its sensitivity to noisy and high-dimensional data.

Conclusion:

While all models performed admirably, SVM and Logistic Regression consistently outperformed others across both PR and ROC curve analyses.

4.3 Confusion Matrix (SVM Example)

To illustrate real-world performance, we present the confusion matrix for the Support Vector Machine model

	Predicted Genuine	Predicted Fraud
Actual Genuine	83,450 (TN)	52 (FP)
Actual Fraud	21 (FN)	438 (TP)

Insights:

- **True Positives (TP):** 438 transactions were correctly flagged as fraud.
- **False Positives (FP):** 52 genuine transactions were incorrectly flagged.
- **False Negatives (FN):** 21 frauds were missed, which is relatively low.
- **True Negatives (TN):** Over 83,000 genuine transactions were correctly classified.

Business Implication:

- **Low FN is critical:** Missing frauds can lead to large financial losses.
- **Moderate FP is manageable**: False positives can be reviewed manually or verified through OTPs, ensuring minimal customer inconvenience.

Analysis:

- Only 21 fraudulent transactions were missed out of 459 frauds.
- 52 legitimate transactions were incorrectly flagged as fraud (manageable in banking systems where manual review is possible).
- Overall, the SVM model shows a strong ability to detect fraudulent behavior with minimal misclassification.

4..4 Time and Resource Efficiency

4.5Training Time

Model	Training Time (s)	Inference Time (avg per 1000 samples)
KNN	< 5	1.2 seconds
Logistic Reg.	< 2	0.01 seconds
SVM	~30	0.05 seconds
Decision Tree	< 3	0.01 seconds

4.6 Limitations Identified

Despite strong performance, some limitations were observed in this study:

1. Data Imbalance:

Even with SMOTE, some fraud patterns may remain underrepresented, especially rare or highly novel schemes.

2. Generalization to Real-World Systems:

- The dataset lacks contextual features (e.g., IP address, location, device ID).
- Models trained on static datasets may not generalize well unless continuously updated.

3. Concept Drift:

Fraud techniques evolve rapidly. Static models degrade in accuracy over time unless retrained frequently with updated data.

4. Scalability Issues:

- **KNN** is slow at prediction time due to distance calculations.
- **SVM**, although highly accurate, is computationally expensive for very large datasets.

5. Explainability:

While models like Logistic Regression and Decision Trees are interpretable, SVM and KNN lack transparency, posing challenges in highly regulated industries.

4.7 Key Takeaways

The model evaluation reveals valuable insights into the practical applicability of each algorithm:

Support Vector Machine (SVM):

- **Best Overall Performance**: Achieved highest recall and F1-score.

- **Use Case**: Ideal for high-stakes fraud detection systems where minimizing false negatives is crucial.

Logistic Regression:

- **Best Interpretability**: High precision and low latency.
- **Use Case**: Suitable for deployment in financial institutions where decisions must be explainable and fast.

Decision Tree:

- **Fast and Transparent**: Slight trade-off in precision.
- **Use Case**: Great for rule-based augmentation or as part of ensemble models.

K-Nearest Neighbors:

- **Effective but Inefficient**: High computational cost makes it less practical for real-time fraud detection.
- **Use Case**: Academic benchmarks or systems with small datasets.

4.8 Recommendations Based on Results

- Implement **SVM** with fallback logic to Logistic Regression if latency exceeds threshold.
- Set dynamic fraud thresholds based on:
    - User risk profiles
    - Transaction type
    - Historical fraud probability
- Integrate **human-in-the-loop** decision system for edge cases.

5. CONCLUSION AND FUTURE ENHANCEMENTS

5.1 Conclusion
This project focused on detecting credit card fraud using machine learning. We tested four algorithms—KNN, Logistic Regression, SVM, and Decision Tree—on a real dataset with imbalanced classes. SVM performed best overall, especially in detecting fraud. Logistic Regression was good for quick, interpretable results. We also addressed ethical concerns and real-world deployment.

5.2 Future Enhancements
To improve the system, future work can include:

- **Ensemble Methods:** Combining models like Random Forest and XGBoost for better accuracy.
- **Deep Learning:** Using autoencoders and LSTMs to catch complex fraud patterns.
- **Real-Time Detection:** Building systems that catch fraud instantly with streaming data.
- **Explainable AI:** Making model decisions easy to understand using SHAP or LIME.
- **Adaptive and Multimodal Detection:** Updating models continuously and using more data types like user behavior and location.

5.3 Importance
This work offers practical and ethical solutions for financial fraud detection that can help both researchers and industry.

5.4 Key Learnings

- Recall is more important than accuracy in fraud detection.
- Balancing data with SMOTE helps but must be done carefully.
- Transparency and deployment readiness are essential.
- Regulations guide building trustworthy AI.

5.5 Final Thoughts
Fraud detection is an ongoing battle. This project lays the foundation for strong machine learning solutions, but continuous improvements and ethical considerations are needed to keep systems effective and fair.

6.ACKNOWLEDGEMENT

We thank God for His blessings and also for giving as good knowledge and strength in enabling us to finish our project. Our deep gratitude goes to our founder late **Dr.D. SELVARAJ, M.A., M.Phil.,** for his patronage in the completion of our project. We like to take this opportunity to thank our honourable chairperson **Dr.S. NALINI SELVARAJ, M.COM., MPhil., Ph.D.** and honourable director, **MR.S. AMIRTHARAJ, M.Tech. , M.B.A** for their support given to us to finish our project successfully. Also we would like to extend my sincere thanks to our respected Principal, **Dr .C. RAMESH BABU DURAI, M.E., Ph.D.** for having provided me with all the necessary facilities to undertake this project. We extend our deepest gratitude to our Head of the Department and our Project Guide, **Mrs.Hemalatha D B.TECH.,M.E.,** whose invaluable suggestions, guidance, and encouragement were instrumental in the success of our project. Her expertise and direction not only steered us through challenges but also elevated our project to a remarkable achievement. Additionally, we express heartfelt thanks to our parents, friends, and staff members whose unwavering support and encouragement sustained us throughout the entirety of this project. Their belief in our capabilities fueled our determination, and their assistance ensured the smooth progress of our work. Together, their contributions have been integral to the realization of our

project's goals, and we are profoundly grateful for their unwavering support and belief in our endeavors.

REFERENCES

[1] Y. Abakarim, M. Lahby, and A. Attioui, "An efficient real-time model for credit card fraud detection based on deep learning," in *Proc. 12th Int. Conf. Intelligent Systems: Theories and Applications (SITA)*, Oct. 2018, pp. 1–7.

[2] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, Jul. 2010.

[3] V. Arora, R. S. Leekha, K. Lee, and A. Kataria, "Facilitating user authorization from imbalanced data logs of credit cards using artificial intelligence," *Mobile Information Systems*, vol. 2020, pp. 1–13, Oct. 2020.

[4] A. O. Balogun, S. Basri, S. J. Abdulkadir, and A. S. Hashim, "Performance analysis of feature selection methods in software defect prediction: A search method approach," *Applied Sciences*, vol. 9, no. 13, pp. 2764, Jul. 2019.

[5] B. Bandaranayake, "Fraud and corruption control at education system level: A case study of the Victorian department of education and early childhood development in Australia," *Journal of Cases in Educational Leadership*, vol. 17, no. 4, pp. 34–53, Dec. 2014.

[6] J. Błaszczyński, A. T. de Almeida Filho, A. Matuszyk, M. Szelński, and R. Słowiński, "Auto loan fraud detection using dominance-based rough set approach versus machine learning methods," *Expert Systems with Applications*, vol. 163, Jan. 2021.

[7] B. Branco, P. Abreu, A. S. Gomes, M. S. C. Almeida, J. T. Ascensão, and P. Bizarro, "Interleaved sequence RNNs for fraud detection," in *Proc. 26th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 2020, pp. 3101–3109.

[8] F. Cartella, O. Anunciacao, Y. Funabiki, D. Yamaguchi, T. Akishita, and O. Elshocht, "Adversarial attacks for tabular data: Application to fraud detection and imbalanced data," arXiv:2101.08030, 2021.

[9] S. S. Lad and A. C. Adamuthe, "Malware classification with improved convolutional neural network model," *International Journal of Computer Network and Information Security*, vol. 12, no. 6, pp. 30–43, Dec. 2021.

[10] V. N. Dornadula and S. Geetha, "Credit card fraud detection using machine learning algorithms," *Procedia Computer Science*, vol. 165, pp. 631–641, Jan. 2019.