# *Credit Card Fraud Detection System: An End-to-End Machine Learning Pipeline with FastAPI Deployment*

Submitted by: Ankan Kisku (231DS004),P Jyothir Aditya(231DS023), Chandhu HP(231DS006)

Department: Mathematical and Computational Sciences

Institute: NITK Surathkal

Course: Practical Training

Academic Year: 2025–2026

## 1. Abstract

The exponential growth of digital financial transactions has catalyzed a parallel escalation in sophisticated cybercrime, positioning credit card fraud as a systemic threat to global economic stability and consumer confidence. This project engineers a robust, end-to-end machine learning pipeline capable of identifying fraudulent activities within milliseconds, addressing the specific constraints of real-time financial monitoring. The study utilizes the European Credit Card Fraud dataset (ULB), comprising 284,807 transactions over a two-day period. A defining characteristic of this dataset is its extreme class imbalance, where positive fraud instances account for only 0.172% of all records, rendering standard accuracy metrics deceptive and necessitating specialized statistical handling.

The methodological framework employs a rigorous data science lifecycle, beginning with a granular Exploratory Data Analysis (EDA) to characterize the distribution of Principal Component Analysis (PCA) transformed features (V1–V28) alongside the non-transformed 'Time' and 'Amount' variables. To mitigate the impact of extreme outliers inherent in financial data, the `RobustScaler` algorithm—which scales data according to the Interquartile Range (IQR)—was applied to the 'Amount' feature. To

scientifically rectify the class disproportionality, the Synthetic Minority Over-sampling Technique (SMOTE) was implemented during the training phase. This technique synthesizes new minority instances via interpolation rather than simple duplication, preventing model overfitting while correcting the bias toward the majority class. Great care was taken to apply resampling strictly within the training folds to prevent data leakage into the validation sets.

A comparative analysis was conducted across a diverse spectrum of algorithms to identify the optimal decision boundary for this high-dimensional space. This included linear baselines (Logistic Regression), non-linear deep learning architectures (Artificial Neural Networks), and unsupervised anomaly detection methods (Autoencoders) designed to flag transactions based on reconstruction error. These were benchmarked against ensemble tree-based gradient boosting frameworks, specifically XGBoost and LightGBM. The models were evaluated using fraud-centric metrics, prioritizing the Area Under the Precision-Recall Curve (AUPRC) and F1-Score over raw accuracy.

The experimental results conclusively identified a hyperparameter-tuned XGBoost Classifier as the superior model. Optimized via `RandomizedSearchCV`, this model effectively captured non-linear feature interactions, achieving a test-set F1-Score of 0.80. This score represents a critical operational balance: maximizing the detection of illicit transactions (Recall) while minimizing the friction of false positives for legitimate users (Precision). Finally, the project transitions from theoretical modeling to production utility through the development of a RESTful API using the FastAPI framework. This microservice encapsulates the preprocessing logic and the serialized XGBoost model, providing a scalable, low-latency interface for real-time inference in a live banking environment.

---

## 2. Introduction

Financial fraud is an adversarial problem that evolves continuously. As detection systems improve, fraudsters adapt their tactics, moving from simple theft to complex synthetic identity fraud and account takeovers. Traditional rule-based systems (e.g., "flag all transactions over $10,000") are static, brittle, and generate high false-positive rates that frictionize the user experience. In contrast, Machine Learning (ML) offers a dynamic approach, learning non-linear patterns from historical data to predict the probability of fraud in real-time.

However, applying ML to fraud detection is fraught with specific challenges:

1. The "Needle in a Haystack" Problem: With fraud rates often below 0.2%, a model can achieve 99.8% accuracy by simply predicting "legitimate" for every transaction. This "accuracy paradox" renders standard metrics useless and necessitates a focus on Precision, Recall, and the Area Under the Precision-Recall Curve (AUPRC).
2. Data Privacy and Anonymity: Financial data is highly sensitive. Features are often anonymized (e.g., via PCA), stripping away semantic context and forcing the data scientist to rely purely on statistical signal processing.
3. Real-Time Latency Requirements: A fraud detection system is only valuable if it can intercede before a transaction is finalized. This imposes strict latency constraints (typically < 50ms) on the inference engine.

This project addresses these challenges through a systematic "Data Science Lifecycle" approach. We begin with a deep statistical dive into the data, proceed to engineer features that maximize separability, and employ state-of-the-art resampling techniques to handle imbalance. We then train and rigorously validate a suite of models, selecting the best performer for hyperparameter optimization. Finally, we encapsulate the logic into a microservice using FastAPI, demonstrating a complete path from raw data to a deployed software product.

---

## 3. Dataset Description & Statistical Properties

The dataset underpinning this research is the European Credit Card Fraud Dataset (sourced from the Machine Learning Group at ULB). It contains transactions made by credit cards in September 2013 by European cardholders.

- Temporal Scope: Transactions occurring over two days.
- Total Transactions: 284,807
- Fraudulent Transactions: 492
- Imbalance Ratio: 0.172% (Positive Class) vs 99.828% (Negative Class).

Feature Analysis:

- Time (Numeric): Represents the seconds elapsed between the current transaction and the first transaction in the dataset. This feature captures cyclical patterns in spending behavior (e.g., high volume during the day, low volume at night).

- Amount (Numeric): The transaction value. This feature is heavily right-skewed, with the vast majority of transactions being small amounts, but a long tail of high-value purchases. This skewness necessitates robust scaling techniques.
- V1 through V28 (Numeric): These features are the result of a Principal Component Analysis (PCA) transformation. PCA was applied for confidentiality reasons to mask original features like User ID, Merchant Location, etc.
    - *Statistical Implication:* These features are orthogonal (uncorrelated) to each other. They have a mean of 0 but varying standard deviations.
- Class (Target): A binary variable where 1 indicates Fraud and 0 indicates Genuine.

## 4. Exploratory Data Analysis (EDA)

A thorough EDA was conducted to understand the underlying structure of the data and inform preprocessing strategies.

### 4.1 Class Imbalance Visualization

Visualizing the target variable Class revealed the severity of the imbalance. A standard bar plot shows the "Fraud" class as barely visible compared to the "Genuine" class. This confirmed that accuracy would be a deceptive metric and that resampling (SMOTE) would be a mandatory step in the pipeline.

### 4.2 Transaction Amount and Time Distributions

- Amount: Density plots for Amount showed distinct distributions for fraud vs. normal transactions. Fraudulent transactions tended to be clustered in specific lower-value ranges (likely to avoid triggering rule-based alerts) but also included high-value outliers. This "heavy-tailed" distribution confirmed the need for log-transformation or robust scaling.
- Time: The Time feature revealed a bimodal distribution for normal transactions (corresponding to day/night cycles). Interestingly, fraudulent transactions did not follow this circadian rhythm as closely, suggesting that fraud occurs at all hours, potentially automated by bots or originating from different time zones.

### 4.3 Correlation and Feature Separation

We analyzed the correlation matrix of the features. While V1-V28 are uncorrelated with each other (due to PCA), their correlation with the Target (Class) varied significantly.

- Negative Correlation: Features V10, V12, V14, and V17 showed strong negative correlations with Class, meaning lower values in these dimensions indicate a higher probability of fraud.
- Positive Correlation: Features V2, V4, and V11 showed positive correlations.
- Visual Separation: Boxplots of features like V14 showed a dramatic separation between classes. For genuine transactions, V14 is centered around 0. For fraud, V14 creates a distinct cluster around -15. This separation makes V14 a "super-feature" for decision tree-based algorithms.

---

## 5. Data Preprocessing & Feature Engineering

The preprocessing phase transforms raw, noisy data into a clean, normalized format suitable for high-performance algorithms.

### 5.1 Outlier Management with RobustScaler

Standardization (subtracting mean, dividing by standard deviation) is sensitive to outliers. Since fraud datasets naturally contain extreme outliers (e.g., a $15,000 purchase among $50 ones), we utilized RobustScaler.

- Mechanism: It removes the median and scales the data according to the Interquartile Range (IQR: the range between the 25th and 75th quantile).
- Application: This was applied to Time and Amount, converting them into scaled_time and scaled_amount.

### 5.2 Stratified Data Splitting

To prevent "data leakage" and ensure statistical representativeness, the data was split into training (70%) and testing (30%) sets using Stratified Sampling. Stratification guarantees that the 0.17% fraud ratio is perfectly preserved in both the train and test sets, preventing the scenario where the test set accidentally contains zero fraud cases.

### 5.3 Synthetic Minority Over-sampling Technique (SMOTE)

Training a model on the raw imbalanced data biases it toward the majority class. We employed SMOTE to balance the training set.

- Process: SMOTE does not simply duplicate minority samples (which causes overfitting). Instead, it selects a fraud instance, identifies its $k$-nearest neighbors in the feature space, and interpolates new synthetic points along the vectors connecting them.

- Result: The training set was balanced to a 50/50 ratio (approx. 199,000 Genuine / 199,000 Fraud).
- Constraint: SMOTE was applied only to the training data. The test set remained in its original imbalanced state to reflect the reality of production data.

---

## 6.PROJECT PIPELINE

```
RAW DATA SOURCE
284,807 Transactions
|
V
PREPROCESSING ENGINE
Data Cleaning Check for Nulls Dupes
Robust Scaling Transform Time and Amount
Using IQR to handle extreme financial outliers
|
V
STRATIFIED SPLITTING
70 Percent Training Set
30 Percent Unseen Test Set
Maintains 0.17 Percent fraud ratio
Left raw for fair evaluation
|
V
IMBALANCE HANDLING
Training Only
SMOTE Algorithm
Synthesize Minority Class to 50 50
|
V
MODEL TRAINING
Logistic Regression
Neural Network
LightGBM
XGBoost Selected
|
V
OPTIMIZATION
RandomizedSearchCV
Tuning Learning
Rate Max Depth
Estimators
|
V
```

FINAL EVALUATION
Validated against the Unseen Test Set
Metric F1 Score Precision Recall Balance
|
V
ARTIFACT SERIALIZATION
Output fraud model joblib
|
V
PRODUCTION DEPLOYMENT
FastAPI
Server Uvicorn ASGI
Interface POST predict
Validation Pydantic Schema

INPUT JSON INFERENCE JSON
Transaction XGBoost Fraud

---

## 7. Model Architectures & Selection

We trained and evaluated a diverse spectrum of algorithms to determine the best fit for this specific problem space.

### 7.1 Logistic Regression (Baseline)

A statistical baseline used to establish the minimum acceptable performance. While computationally efficient, it struggled to capture the complex, non-linear boundaries between fraud and normal transactions, yielding high recall but unacceptably low precision.

### 7.2 Deep Neural Network (ANN)

A feed-forward Artificial Neural Network was constructed using Keras/TensorFlow.

- Architecture: Input Layer (30 nodes) → Dense (32, ReLU) → Dense (16, ReLU) → Dropout (0.5) → Output (1, Sigmoid).
- Performance: The ANN showed promise but required significantly more training time and data to converge compared to tree-based models. It also lacked the interpretability of decision trees.

### 7.3 Autoencoder (Anomaly Detection)

An unsupervised approach where the model is trained *only* on genuine transactions. It learns to compress and reconstruct normal data. When a fraud transaction is introduced, the model fails to reconstruct it accurately, resulting in a high "Reconstruction Error." This error serves as the anomaly score. While theoretically sound, the Autoencoder found it difficult to set a universal threshold that didn't flag too many legitimate outliers.

### 7.4 Ensemble Methods: LightGBM & XGBoost

Gradient Boosting Decision Trees (GBDTs) are the state-of-the-art for tabular data.

- LightGBM: Optimized for speed and memory efficiency, using leaf-wise tree growth.
- XGBoost: Uses level-wise growth and advanced regularization (L1/L2) to prevent overfitting.
- Verdict: XGBoost outperformed all other models. Its ability to handle non-linear feature interactions and its robustness to noise made it the ideal candidate for the final model.

---

## 8. Hyperparameter Optimization

To squeeze maximum performance from the XGBoost model, we moved beyond default settings using RandomizedSearchCV. This method explores a defined grid of parameters to find the optimal configuration.

Hyperparameter Space:

- n_estimators (Number of trees): [100, 200, 300]
- max_depth (Tree complexity): [3, 5, 7]
- learning_rate (Step size): [0.05, 0.1, 0.2]
- subsample (Data sampling): [0.7, 1.0]
- gamma (Regularization): [0.0, 0.1, 0.2]

Optimal Configuration:

The search converged on a configuration with max_depth=7, learning_rate=0.2, and n_estimators=200. This setup allowed the model to learn deep, complex patterns without overfitting, thanks to the gamma regularization.

---

## 9. Results and Discussion

The models were evaluated on the Unseen Test Set (30% of original data), which preserved the natural 0.17% fraud ratio.

Performance Metrics:

| Algorithm | Precision (Fraud) | Recall (Fraud) | F1-Score (Fraud) |
|---|---|---|---|
| Logistic Regression | 0.06 | 0.88 | 0.12 |
| Neural Network | 0.46 | 0.80 | 0.58 |
| LightGBM | 0.57 | 0.82 | 0.67 |
| XGBoost (Tuned) | 0.80 | 0.79 | 0.80 |

Analysis:

The Tuned XGBoost model achieved an F1-Score of 0.80, a massive improvement over the baseline.

- Recall of 79%: The model successfully detects nearly 8 out of every 10 fraud attempts.
- Precision of 80%: When the model flags a transaction, it is correct 80% of the time. This is crucial for reducing operational costs (fewer manual reviews) and maintaining customer trust (fewer declined genuine cards).

## 10. Deployment: The FastAPI Microservice

The final stage of the project involved operationalizing the model. We encapsulated the trained XGBoost model into a REST API using FastAPI.

Technical Specification:

- Framework: FastAPI (Python)
- Server: Uvicorn (ASGI server for high concurrency)
- Input Schema: Validated using Pydantic. The API enforces strict type checking, ensuring that all 30 required features (V1-V28, Time, Amount) are present and are valid floats.
- Serialization: The trained pipeline was serialized using joblib for efficient loading.

API Endpoints:

1. GET / (Health Check): Confirms the API is live and the model is loaded.
2. POST /predict (Inference):
    - *Input:* JSON payload containing transaction features.
    - *Processing:* Converts JSON to DataFrame, scales Time/Amount using the saved scaler, and feeds data to XGBoost.

*Output:*

JSON

```
{
  "prediction": 1,
  "status": "Transaction BLOCKED (Fraud Detected)",
  "fraud_probability": 0.92
}
```

○

This deployment proves the model's capability to function in a production environment, providing millisecond-level responses required for real-time transaction approval.

---

## 11. Conclusion and Future Scope

Conclusion:

This project successfully developed a high-fidelity fraud detection system capable of operating in a highly imbalanced environment. By leveraging SMOTE for training stability and XGBoost for predictive power, we achieved an F1-score of 0.80, effectively balancing risk mitigation with user experience. The FastAPI deployment demonstrates the system's readiness for real-world integration.

Future Work:

- Model Monitoring: Implementing tools like Prometheus or Grafana to monitor "Concept Drift" (how fraud patterns change over time) and trigger automated retraining.
- Explainability (XAI): Integrating SHAP (SHapley Additive exPlanations) into the API response to provide reasons for declines (e.g., "Declined due to unusually high Amount at 3 AM").
- Sequence Modeling: Utilizing Long Short-Term Memory (LSTM) networks to analyze the *sequence* of transactions for a user, rather than treating each transaction in isolation.