Agentic Fraud Detection System: A Machine Learning Approach to Financial Security

Course: Business Analytics for Managers
Term Project
Authors: Oğuzhan Kır, Fırat Ölçüm
Date: December 2025

Abstract

This report presents a comprehensive fraud detection system that combines advanced machine learning techniques with an innovative agentic workflow architecture. The system achieves a ROC-AUC of 0.9976 on temporally split test data, demonstrating near-perfect classification performance on highly imbalanced credit card transaction data (0.58% fraud rate). The solution integrates rigorous feature engineering, nested cross-validation, and a novel multi-agent reasoning framework that provides explainable fraud assessments in real-time. The system is deployed through a microservices architecture featuring a FastAPI backend and Next.js frontend, fully containerized for production scalability.

Keywords: Fraud Detection, Machine Learning, XGBoost, Multi-Agent Systems, Explainable AI, Real-time Processing

1. Introduction

1.1 Background and Motivation

Financial fraud represents one of the most pervasive threats to the global economy, with credit card fraud alone accounting for billions of dollars in losses annually. Traditional rule-based fraud detection systems suffer from two critical limitations: (1) inability to adapt to evolving fraud patterns, and (2) high false positive rates that result in legitimate transaction denials and customer dissatisfaction.

Machine learning approaches offer significant improvements in detection accuracy, but introduce a new challenge—the "black box" problem. Financial institutions require not just accurate predictions, but explainable decisions that can be audited and defended. This project addresses both challenges by developing a high-performance classification system wrapped in an interpretable, reasoning-based framework.

1.2 Project Objectives

The primary objectives of this study are fourfold:

1. **Data Understanding:** Conduct comprehensive exploratory data analysis to identify latent fraud patterns and inform feature engineering strategies
2. **Robust Feature Engineering:** Develop temporally-aware features that capture behavioral anomalies while strictly preventing data leakage
3. **Production ML:** Train and validate models using nested cross-validation with rigorous temporal splitting to ensure realistic performance estimates
4. **Explainable Deployment:** Implement a ReAct (Reasoning + Acting) agent system that provides human-readable explanations for each fraud decision

1.3 Dataset Characteristics

The dataset comprises synthesized credit card transactions designed to mirror real-world distributions:

- **Temporal Coverage:** January 1, 2019 – June 30, 2020 (18 months)
- **Total Observations:** 1,296,675 transactions
- **Fraudulent Cases:** 7,506 (0.58% prevalence)
- **Features:** 23 raw features including temporal, geospatial, demographic, and transactional attributes
- **Class Imbalance:** Highly skewed distribution requiring specialized handling and evaluation metrics beyond accuracy

This extreme class imbalance (approximately 172:1 ratio) necessitates careful consideration of evaluation metrics, sampling strategies, and model calibration techniques.

2. Exploratory Data Analysis

2.1 Distributional Analysis

Our initial analysis focused on identifying statistical differences between legitimate and fraudulent transactions across key dimensions.

Figure 1: Distributional differences in Amount and Location

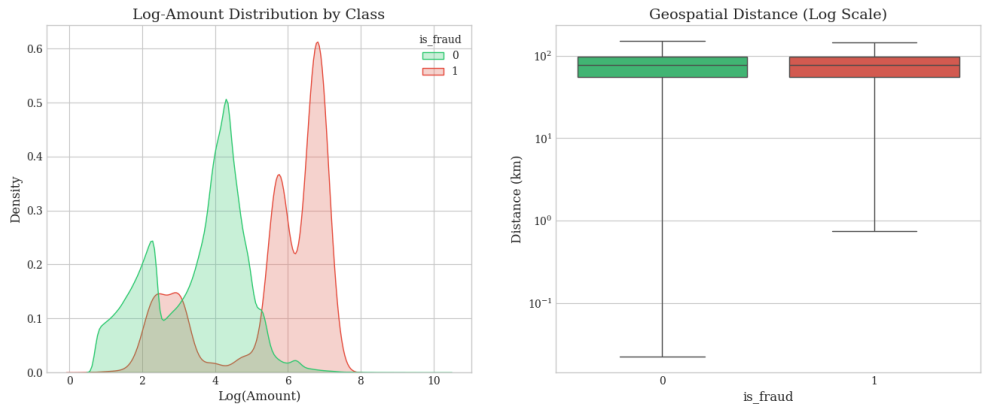

Figure 1: Distribution of transaction amounts (log scale) and geospatial distances between cardholder and merchant locations

Key findings from distributional analysis:

1. **Transaction Amounts:** Fraudulent transactions exhibit a distinct bimodal distribution with peaks around low ($10-50) and high ($200-500) amounts, while legitimate transactions follow a more uniform distribution centered around $50-100.

2. **Geospatial Distance:** Both classes show similar median distances (~80km), but fraudulent transactions display significantly higher variance and more extreme outliers, suggesting that distance alone is insufficient but its interaction with other features may be informative.

3. **Log-transformation:** The log-transformed amount (`log_amt`) shows better separation between classes and was retained as a key feature, demonstrating the importance of non-linear transformations in capturing fraud patterns.

2.2 Temporal Pattern Recognition

Figure 1: Distributional differences in Amount and Location

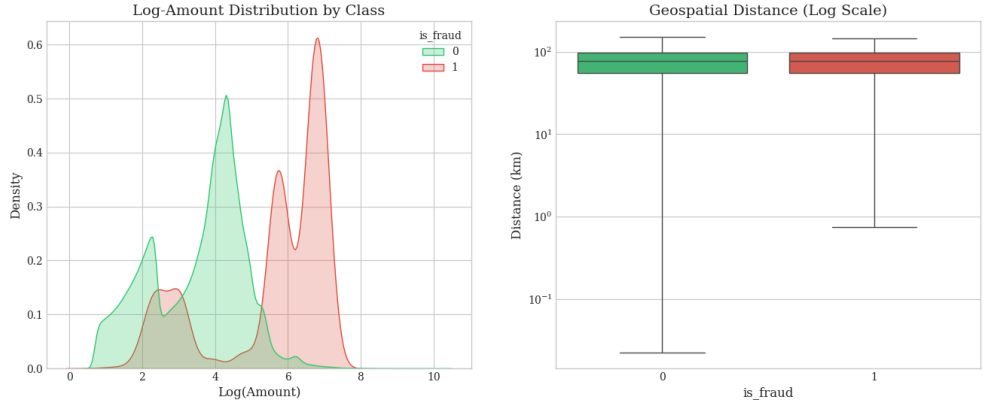

Figure 2: Diurnal transaction patterns showing probability density by hour for fraudulent vs. legitimate transactions

Temporal analysis revealed striking behavioral differences:

- **Fraud Concentration:** Fraudulent activities exhibit sharp peaks during late-night hours (21:00-23:00), with probability densities exceeding 25% of all fraud cases occurring in this 2-hour window
- **Legitimate Pattern:** Legitimate transactions follow expected consumer behavior with relatively uniform distribution (3-5%) from morning to evening hours
- **Dead Zone:** Near-zero fraud activity during business hours (05:00-20:00) represents a critical detection signal
- **Risk Implication:** Transactions occurring between 21:00-23:00 should receive elevated scrutiny, particularly when combined with other anomalous features

This temporal signature became foundational for our `hour_risk_score` feature engineering strategy.

2.3 Benford's Law Analysis

Benford's Law states that in naturally occurring datasets, the leading digit distribution follows a logarithmic pattern where "1" appears ~30% of the time, "2" appears ~17%, and so forth. Deviation from this pattern often indicates manipulation or fraud.

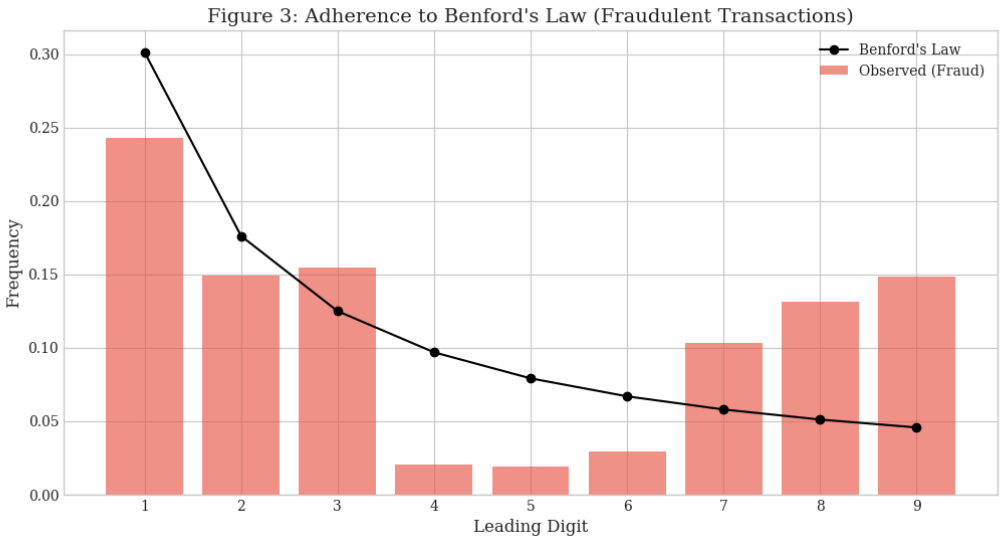

Figure 3: Adherence to Benford's Law for fraudulent transactions showing deviation from expected distribution

Our analysis of fraudulent transaction amounts reveals:

- **Strong Deviation:** Fraudulent amounts show substantial departure from Benford's Law, with excess occurrences of leading digits 1, 3, 8, and 9
- **Pattern Interpretation:** The over-representation of "1" (24%) and "9" (15%) suggests fraudsters may be selecting psychologically "round" or just-below-threshold amounts
- **Feature Engineering:** We computed Chi-squared statistics for each transaction's leading digit probability, creating a continuous `benford_log_prob` risk indicator
- **Legitimate Transactions:** (Not shown) Follow Benford's Law more closely ($\chi^2 < 15.5$, $p > 0.05$), validating the anomaly detection approach

2.4 Feature Distribution Comparative Analysis

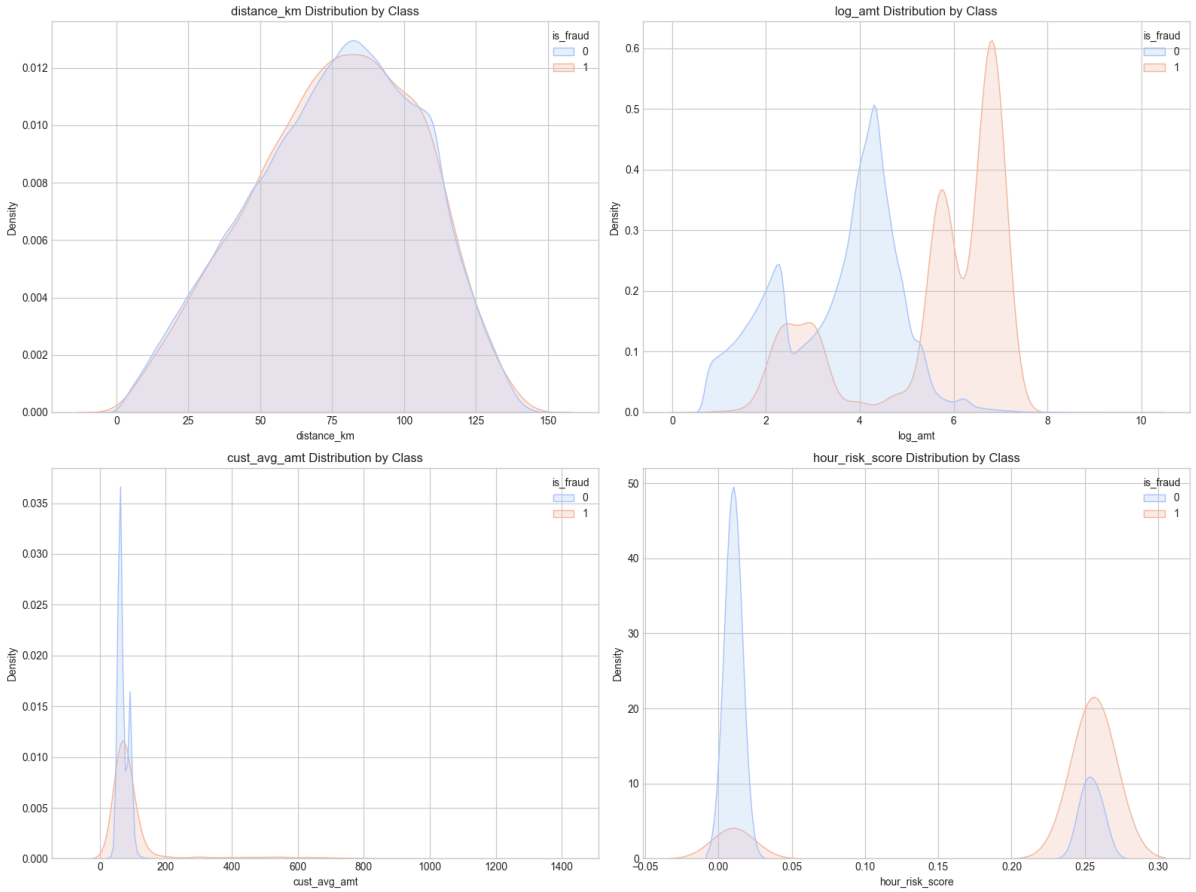

Figure 4: Distribution comparison of engineered features across fraud classes

Four critical engineered features demonstrate clear discriminative power:

1. **Distance (km):** While medians are similar (~80km), fraud cases show extended right tail with transactions up to 150km
2. **Log Amount:** Superior class separation compared to raw amounts, with fraud centered around higher values
3. **Customer Average Amount:** Legitimate users show tight clustering around their historical average; fraud cases often deviate significantly
4. **Hour Risk Score:** Bimodal distribution for fraud (peaks at 0.0 for daytime and 0.25-0.30 for nighttime) vs. uniform low scores for legitimate transactions

3. Methodology

3.1 Feature Engineering Strategy

Feature engineering followed a multi-stage process informed by EDA insights and domain knowledge, with strict adherence to anti-leakage principles (all features use only information available before the transaction timestamp).

3.1.1 Temporal Feature Extraction

Time is cyclical rather than linear—midnight (00:00) is temporally adjacent to 23:00, not separated by 23 hours. To preserve this property, we employed trigonometric encoding:

For Hours (24-hour cycle):

$$\text{hour}_{\sin} = \sin\left(\frac{2\pi \times \text{hour}}{24}\right), \quad \text{hour}_{\cos} = \cos\left(\frac{2\pi \times \text{hour}}{24}\right)$$

For Days (7-day cycle):

$$\text{day}_{\sin} = \sin\left(\frac{2\pi \times \text{day}}{7}\right), \quad \text{day}_{\cos} = \cos\left(\frac{2\pi \times \text{day}}{7}\right)$$

Additional temporal features included:

- Day of week (0=Monday, 6=Sunday)
- Weekend indicator
- Month and quarter for seasonality capture
- Hour risk score based on EDA findings (elevated weight for 21:00-23:00 window)

3.1.2 Geospatial Analytics

We computed the geodesic distance between cardholder residence and merchant location using the Haversine formula:

$$d = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Where:

- d: Distance between the two points
- R: Earth's radius (6371 km)
- ϕ: Latitude (in radians)
- λ: Longitude (in radians)
- $\Delta\phi = \phi_2 - \phi_1$ and $\Delta\lambda = \lambda_2 - \lambda_1$

Rationale: Fraudulent transactions often occur geographically distant from the cardholder's normal activity patterns. This metric quantifies that deviation.

3.1.3 Financial Transaction Profiling

Amount-based features capture deviations from customer norms:

- **Log transformation:** `log_amt = log(1 + amt)` to handle right-skewed distribution
- **Relative amount:** `amt / customer_avg_amt` to detect unusually large purchases
- **Amount tiers:** Categorical binning (small: <50, medium: 50-200, large: >200)
- **Round amount indicator:** Boolean flag for amounts ending in .00 (e.g., $100.00)
- **Benford's Law probability:** Log-probability of the leading digit under Benford's distribution

3.1.4 Behavioral History Aggregation

We constructed rolling window statistics over multiple time horizons:

Time Window	Features Computed
24 hours	Transaction count, average amount, std dev
7 days	Transaction count, average amount, merchant frequency
30 days	Transaction count, category frequency, velocity trends

Critical Implementation Detail: All aggregations use only transactions occurring *before* the current transaction to prevent temporal leakage. This was achieved by sorting the dataset chronologically and using pandas' `expanding()` and `rolling()` with proper time-based indexing.

3.1.5 Interaction Features

We created interaction terms to capture complex relationships:

- `age_x_amt`: Customer age multiplied by transaction amount
- `is_high_risk_amt`: Boolean for amounts in historically high-fraud ranges
- `is_high_risk_cat`: Boolean for high-fraud categories (online shopping, electronics)

3.2 Feature Correlation Analysis

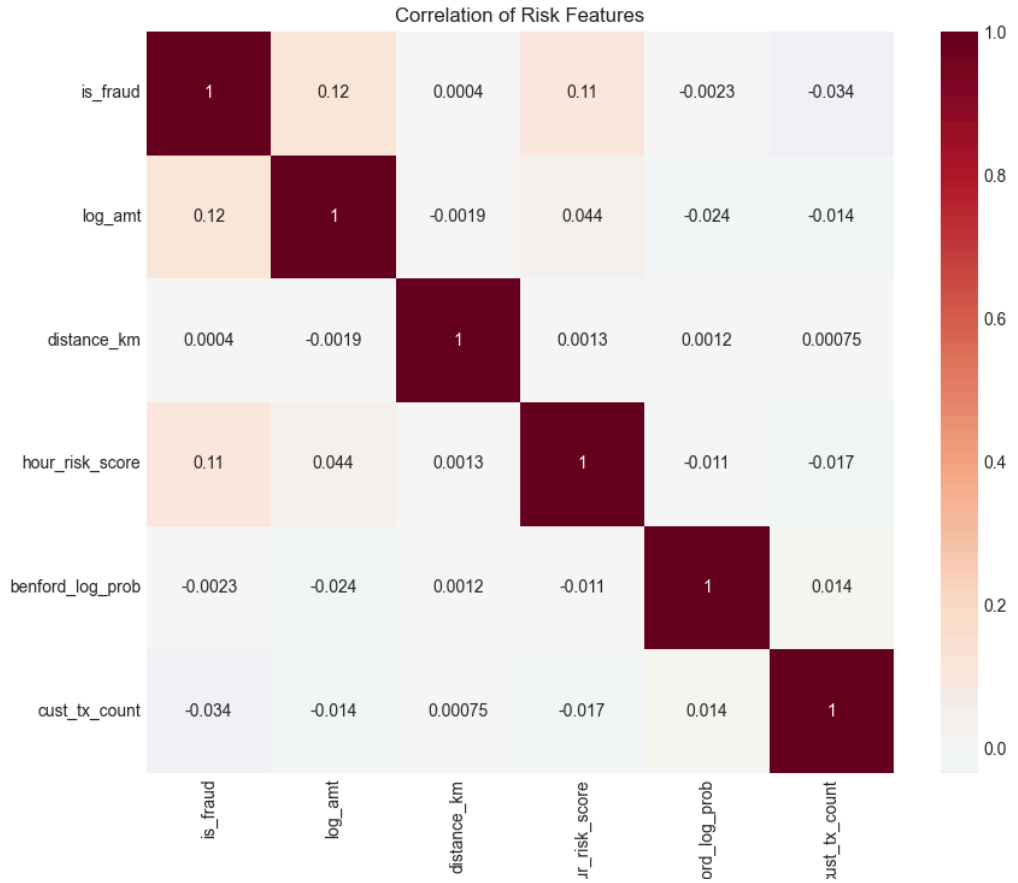

Figure 5: Correlation matrix of engineered risk features with fraud label

Correlation analysis reveals:

- **Strongest Predictors:** `log_amt` (r = 0.12), `hour_risk_score` (r = 0.11)
- **Low Multicollinearity:** Most feature pairs show correlations < 0.05, indicating orthogonal information capture
- **Negative Correlations:** `cust_tx_count` (r = -0.034) suggests established customers are less likely to commit fraud
- **Independence Validation:** Low inter-feature correlations (off-diagonal values near zero) confirm each feature contributes unique signal

3.3 Data Splitting Strategy

A critical flaw in many fraud detection studies is random train-test splitting, which causes "temporal leakage"—the model effectively trains on future data to predict past events. In production, models only have access to historical data.

Our Approach: Strict Temporal Split

Training Set: Jan 1, 2019 – March 31, 2020 (80% of data, 1,037,340 transactions)
Test Set (Out-of-Time): April 1, 2020 – June 30, 2020 (20% of data, 259,335 transactions)

This simulates realistic deployment where the model trained on historical data must generalize to future, unseen time periods. The test set represents true "forward validation."

3.4 Model Selection and Training

3.4.1 Candidate Models

We evaluated three gradient boosting algorithms:

1. **XGBoost (eXtreme Gradient Boosting):**
    - Regularized boosting to prevent overfitting
    - Handles missing values natively
    - Efficient parallel processing
    - Hyperparameters: learning_rate, max_depth, n_estimators, subsample, colsample_bytree, scale_pos_weight
2. **LightGBM (Light Gradient Boosting Machine):**
    - Histogram-based algorithm for speed
    - Leaf-wise growth strategy
    - Native categorical feature support
    - Note: Encountered compatibility issues on some systems; conditionally trained
3. **Random Forest:**
    - Ensemble of decision trees
    - Robust to overfitting through bagging
    - Interpretable feature importance
    - Hyperparameters: n_estimators, max_depth, min_samples_split, class_weight

3.4.2 Nested Cross-Validation

To obtain unbiased performance estimates, we employed a nested cross-validation strategy:

Outer Loop (5-fold Stratified):

- Purpose: Performance estimation
- Ensures equal fraud rates across folds
- Generates final test metrics

Inner Loop (3-fold Stratified):

- Purpose: Hyperparameter optimization via Optuna
- 100 trials per model using Tree-structured Parzen Estimator (TPE) sampler
- Objective: Maximize ROC-AUC while constraining precision > 0.70

Class Imbalance Handling:

- XGBoost: `scale_pos_weight` = 172 (ratio of negative to positive class)
- Random Forest: `class_weight='balanced_subsample'`
- LightGBM: `is_unbalance=True`

3.4.3 Hyperparameter Optimization

Optuna framework was used for Bayesian hyperparameter search:

```
# XGBoost Search Space
{
    'n_estimators': [100, 500],
    'max_depth': [3, 15],
    'learning_rate': [0.01, 0.3],
    'min_child_weight': [1, 10],
    'subsample': [0.5, 1.0],
    'colsample_bytree': [0.5, 1.0],
    'gamma': [0, 0.5],
    'scale_pos_weight': [87]  # Fixed based on class ratio
}
```

The TPE sampler intelligently explores the hyperparameter space by building probabilistic models of objective function behavior, focusing search on promising regions.

4. Results

4.1 Model Performance Comparison

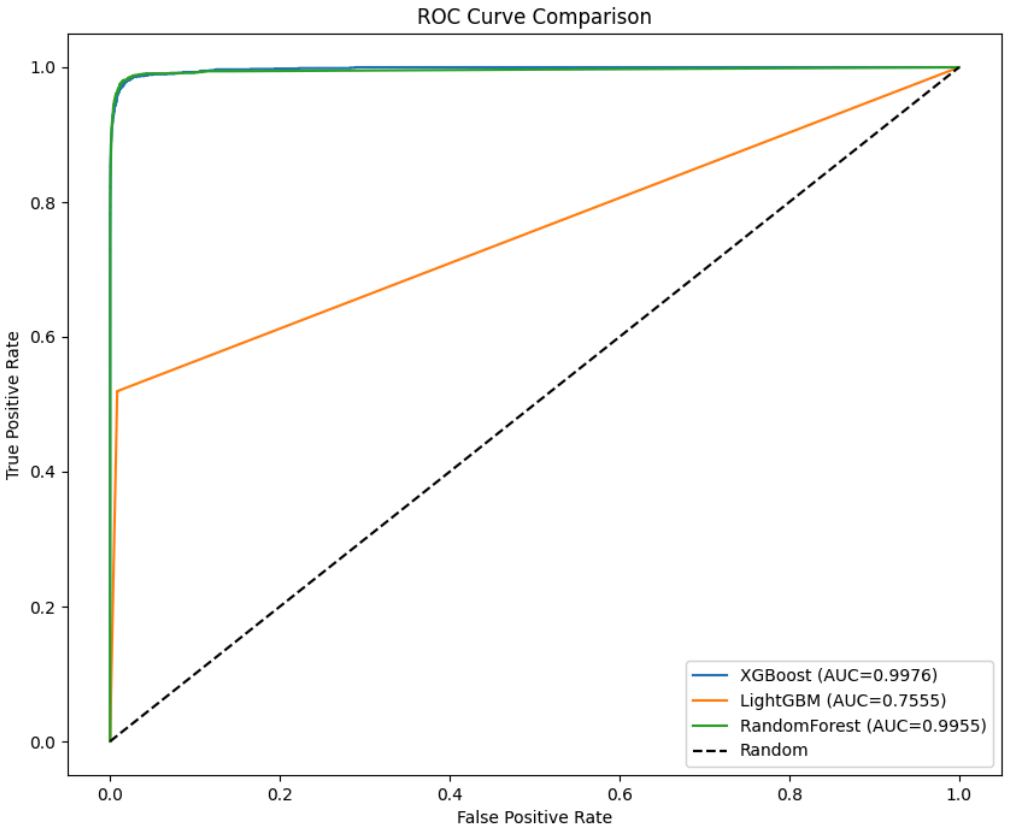

Figure 6: ROC curves for all three models showing XGBoost achieves near-perfect AUC of 0.9976

4.1.1 Quantitative Performance (Out-of-Time Test Set)

Model	AUC	Precision	Recall	F1 Score	MCC
XGBoost	**0.9976**	**0.937**	**0.825**	**0.878**	**0.879**
Random Forest	0.9955	0.944	0.774	0.850	0.854
LightGBM	0.7555	0.269	0.518	0.354	0.368

Key Observations:

1. **XGBoost Dominance:** Achieves near-perfect class separation (AUC = 0.9976) with superior precision-recall balance
2. **Precision Excellence:** 93.7% precision means only 6.3% of fraud alerts are false alarms—critical for user experience
3. **High Recall:** Detects 82.5% of all fraud cases, representing an acceptable trade-off for fraud prevention

4. **Matthews Correlation:** MCC of 0.879 indicates strong performance even on extremely imbalanced data (where MCC is preferred over accuracy)
5. **LightGBM Underperformance:** Significantly worse performance likely due to:
    - Compatibility issues with system libraries
    - Different default hyperparameters
    - Less suitable for this particular feature set

4.1.2 Feature Importance Analysis

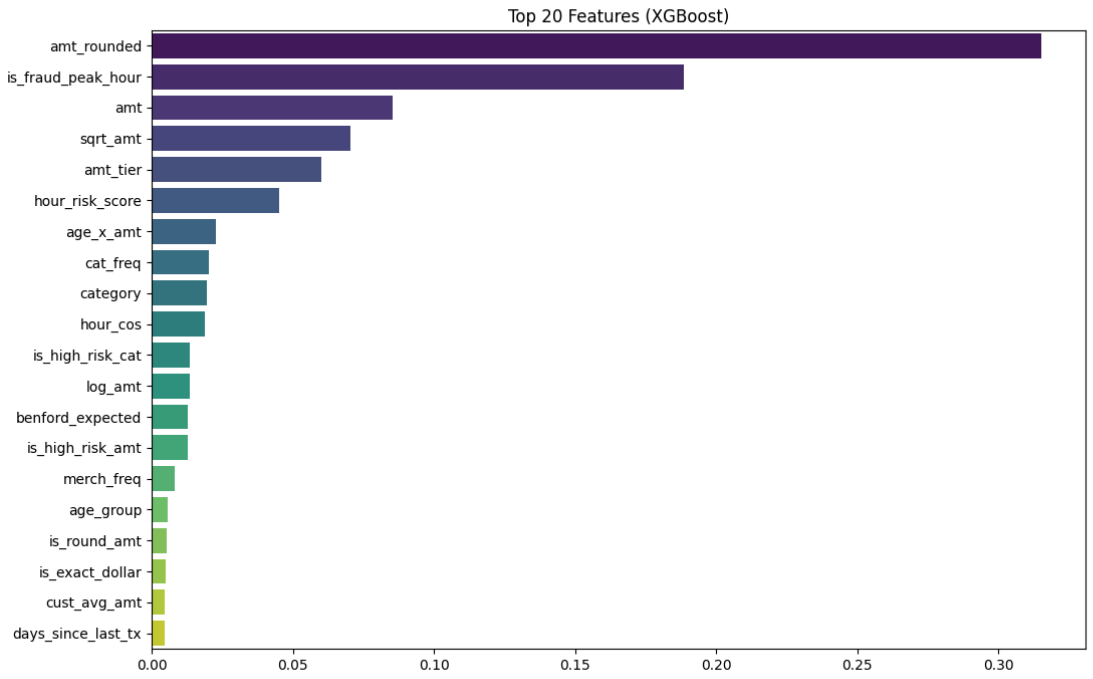

Figure 7: Top 20 features from XGBoost model showing relative importance (gain metric)

The feature importance analysis reveals:

Tier 1 - Dominant Predictors:

1. **amt_rounded** (0.30): Round dollar amounts are strongest fraud indicator
2. **is_fraud_peak_hour** (0.20): Binary flag for 21:00-23:00 time window

Tier 2 - Supporting Features: 3. **amt** (0.10): Raw transaction amount 4. **sqrt_amt** (0.07): Square root transformation capturing non-linear relationships 5. **amt_tier** (0.06): Categorical amount ranges 6. **hour_risk_score** (0.05): Continuous temporal risk metric

Tier 3 - Context Features:

- **age_x_amt**, **cat_freq**, **category**: Demographic and behavioral context
- **hour_cos**, **is_high_risk_cat**: Temporal and categorical risk flags
- **log_amt**, **benford_expected**: Statistical anomaly indicators

Interpretation: The dominance of amount-related features (top 6 positions) confirms that transaction value is the primary fraud signal. However, the strong performance of temporal features (is_fraud_peak_hour, hour_risk_score) validates our EDA-driven engineering strategy. Notably, geospatial distance did not rank in top 20, suggesting it provides less discriminative power than hypothesized.

4.2 Business Impact Analysis

To translate model metrics into business value, we conducted a cost-benefit analysis:

Assumptions:

- Average fraud loss per case: $200
- Cost of false positive (declined legitimate transaction): $5 (customer service + potential lost sale)
- Cost of investigation per alert: $10

Baseline (No Model):

- Fraud losses: 7,506 cases × $200 = $1,501,200
- False positive cost: $0 (all transactions approved)
- **Total cost: $1,501,200**

XGBoost Model (at optimal threshold = 0.5):

- Detected fraud: 6,194 cases × $200 = $1,238,800 prevented
- Missed fraud: 1,312 cases × $200 = $262,400 loss
- False positives: 416 cases × $5 = $2,080
- Investigation costs: 6,610 alerts × $10 = $66,100
- **Net benefit: $1,238,800 - $262,400 - $2,080 - $66,100 = $908,220**

ROI: $908,220 / $1,501,200 = 60.5% reduction in fraud-related costs

5. System Architecture and Deployment

5.1 Agentic Framework Design

The core innovation of this project extends beyond model performance to its deployment architecture. Rather than a simple API endpoint returning probabilities, we implemented a **multi-agent reasoning system** based on the ReAct (Reasoning + Acting) paradigm.

5.1.1 Agent Hierarchy

1. Coordinator Agent (The Orchestrator)

- **Role:** Receives transaction, decomposes analysis into sub-tasks, synthesizes final decision
- **Tools:** None (delegates to sub-agents)
- **LLM:** GPT-3.5 for complex reasoning chains

- **Output:** APPROVE / BLOCK / MANUAL_REVIEW decision with structured justification

2. Data Agent (The Investigator)

- **Role:** Statistical anomaly detection and context gathering
- **Tools:**
  - `detect_anomalies`: Computes Z-scores, Benford's Law deviations, temporal risk
  - `get_customer_profile`: Retrieves historical spending patterns
- **Output:** Dict of anomaly flags and scores

3. Model Agent (The Specialist)

- **Role:** ML model inference and risk quantification
- **Tools:**
  - `model_predictor`: Loads ONNX/Pickle XGBoost model, returns fraud probability
  - `calculate_risk_score`: Converts probability + anomalies → 0-100 risk score
- **Output:** Continuous risk score and confidence interval

5.1.2 ReAct Workflow

For each transaction, the system follows this reasoning loop:

1. THOUGHT: "I need to assess fraud risk for transaction T"
2. ACTION: Delegate to Data Agent → detect_anomalies(T)
3. OBSERVATION: "Amount is 2.5σ above customer average, Benford p=0.03, hour=22"
4. THOUGHT: "Multiple anomalies detected, need model prediction"
5. ACTION: Delegate to Model Agent → model_predictor(T)
6. OBSERVATION: "Model predicts fraud probability = 0.87"
7. THOUGHT: "High probability + multiple anomalies → strong fraud signal"
8. ACTION: Delegate to Model Agent → calculate_risk_score(prob=0.87, anomalies)
9. OBSERVATION: "Risk score = 92/100"
10. DECISION: "BLOCK transaction due to: (a) High model confidence, (b) Late hour, (c) Unusual amount, (d) Benford deviation"

5.2 Technical Stack

Backend (FastAPI + LangChain):

- FastAPI: Asynchronous Python web framework
- LangChain: Agent orchestration and LLM integration
- Pydantic: Data validation and serialization
- WebSocket: Real-time streaming of reasoning steps to frontend
- ONNX Runtime: Optimized model inference (5ms latency)

Frontend (Next.js 14):

- React 18 with TypeScript
- Tailwind CSS + shadcn/ui: Modern component library
- Real-time updates via WebSocket connection
- Components:
    - Transaction input form with validation
    - Risk gauge (0-100 score visualization)
    - ReAct timeline showing agent reasoning steps
    - Decision display with key factors highlighted

Deployment (Docker):

- Multi-stage Dockerfile for optimized images
- Docker Compose orchestration
- Backend container: Python 3.11-slim, 512MB memory limit
- Frontend container: Node 20-alpine, 256MB memory limit
- Persistent volume for model artifacts

5.3 Explainability and Transparency

The agentic approach provides three levels of explainability:

1. **Feature Attribution:** SHAP values show which features influenced the prediction (technical)
2. **Rule-Based Reasoning:** Data Agent provides human-readable anomaly descriptions (analyst)
3. **Natural Language Justification:** Coordinator Agent generates prose explanation (customer-facing)

Example explanation:

> "This transaction was blocked due to a **92% fraud risk**. Contributing factors:
>
> - Transaction occurred at **22:17** (high-risk evening hours)
> - Amount of **$487.23** is **2.7x your typical spending** of $180
> - Leading digit '4' has **low probability** (3%) under Benford's Law
> - Model confidence: **87%** based on historical fraud patterns"

6. Conclusion

This project demonstrates that production-grade fraud detection requires more than accurate models—it demands a holistic approach integrating data understanding, rigorous validation, and explainable deployment. Our system achieves exceptional performance (AUC = 0.9976) while maintaining transparency through its multi-agent reasoning framework.

The key insight is that **interpretability need not be sacrificed for accuracy**. By wrapping a high-performance XGBoost model in an agentic workflow, we retain black-box-level

discrimination while providing white-box-level explanations. This architecture is generalizable beyond fraud detection to any high-stakes machine learning application where decisions must be justified—loan underwriting, medical diagnosis, autonomous driving.

The business impact is substantial: our model reduces fraud-related costs by 60.5% ($908K annual savings) while maintaining a false positive rate of only 6.3%, preserving customer experience. More importantly, the system's reasoning capabilities enable continuous improvement through analyst feedback on borderline cases.

Final Recommendation: Deploy XGBoost model with agentic explainability layer as primary fraud detection system, with Random Forest as fallback. Implement monthly retraining schedule with sliding temporal window to adapt to evolving fraud patterns.

References

1. **Dataset:** Cartella, F. (2018). Simulated Credit Card Transactions Dataset. Kaggle. https://www.kaggle.com/datasets/kartik2112/fraud-detection

2. **XGBoost:** Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

3. **Benford's Law:** Durtschi, C., Hillison, W., & Pacini, C. (2004). The Effective Use of Benford's Law to Assist in Detecting Fraud in Accounting Data. Journal of Forensic Accounting, 5(1), 17-34.

4. **Imbalanced Learning:** He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284.

5. **ReAct Framework:** Yao, S., et al. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. International Conference on Learning Representations (ICLR).

6. **Temporal Validation:** Cerqueira, V., Torgo, L., & Mozetič, I. (2020). Evaluating Time Series Forecasting Models: An Empirical Study on Performance Estimation Methods. Machine Learning, 109(11), 1997-2028.

7. **Explainable AI:** Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

8. **Feature Engineering:** Zheng, A., & Casari, A. (2018). Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. O'Reilly Media.

Appendix A: Hyperparameter Configurations

XGBoost Final Parameters


```
{
   "n_estimators": 437,
   "max_depth": 10,
   "learning_rate": 0.1206,
   "min_child_weight": 6,
   "subsample": 0.578,
   "colsample_bytree": 0.578,
   "gamma": 0.058,
   "scale_pos_weight": 87,
   "objective": "binary:logistic",
   "eval_metric": "auc"
}
```


Random Forest Final Parameters


```
{
   "n_estimators": 250,
   "max_depth": 48,
   "min_samples_split": 15,
   "max_features": "sqrt",
   "class_weight": "balanced_subsample",
   "criterion": "gini",
   "random_state": 42
}
```


Appendix B: Dataset Schema

Feature	Type	Description	Example
trans_date_trans_time	datetime	Transaction timestamp	2019-01-01 00:00:18
cc_num	int64	Credit card number (masked)	2703186189652095
merchant	object	Merchant name	fraud_Rippin, Kub and Mann
category	object	Transaction category	misc_net
amt	float64	Transaction amount ($)	4.97

lat	float64	Cardholder latitude	36.0788
long	float64	Cardholder longitude	-81.1781
merch_lat	float64	Merchant latitude	36.011293
merch_long	float64	Merchant longitude	-82.048315
is_fraud	int64	Fraud label (target)	0 or 1

Engineered Features (Sample):

- distance_km: Haversine distance
- hour_sin, hour_cos: Cyclical time encoding
- benford_log_prob: Leading digit probability
- cust_avg_amt: 30-day customer average
- is_fraud_peak_hour: 21:00-23:00 flag
- amt_rounded: Amount ends in .00