Fraud Pattern Generator for Model Training

1st Dr. BAHUBALI SHIRAGAPUR
Department of CSE (AI&ML)
Dayananda Sagar University
Bengaluru, India
bahubali-aiml@dsu.edu.in

2nd Avutala Dhruvish Reddy
Department of CSE (AI&ML)
Dayananda Sagar University
Bengaluru, India
adr.meteor@gmail.com

3rd Chethan S
Department of CSE (AI&ML)
Dayananda Sagar University
Bengaluru, India
chethansharma29@gmail.com

4th Chilaka Sai Raghavendra
Department of CSE (AI&ML)
Dayananda Sagar University
Bengaluru, India
sairaghavendracsr@gmail.com

***Abstract*—The advancement of financial fraud detection is persistently hindered by the scarcity of public datasets, largely due to stringent privacy laws like GDPR and proprietary banking restrictions. To bridge this gap, this research proposes a Generative Augmented Fraud Detection Framework, a system designed to overcome data poverty through deep generative learning. Unlike traditional statistical simulations, this project integrates a Conditional Tabular GAN (CTGAN) to learn the high-dimensional joint probabilities of transaction features, synthesizing realistic, privacy-preserving fraud logs that capture complex non-linear correlations. To ensure the models are robust against production-grade imperfections, the pipeline incorporates a stochastic noise injection mechanism that introduces missing values and statistical outliers prior to training. The detection architecture employs a "Hybrid Intelligence" strategy, fusing deterministic domain heuristics—encoded as a dynamic "Rule Score"—with probabilistic machine learning. We benchmarked the performance of Logistic Regression, Random Forest, and Gradient Boosting classifiers, finding that a GAN-augmented Random Forest achieves superior recall when paired with an unsupervised Isolation Forest for anomaly discovery. The entire ecosystem is operationalized via a reactive Streamlit interface, enabling stakeholders to perform real-time sensitivity analysis and validate the efficacy of deep generative models in a self-contained, data-free environment.**

***Index Terms*—Anomaly Detection, Financial Fraud, Machine Learning, Random Forest, Streamlit, Synthetic Data Generation, Supervised Learning.**

I. Introduction

The rapid digitization of the global financial sector has precipitated a parallel rise in complex fraudulent activities. As digital payment ecosystems expand, so too do the vectors of attack utilized by malicious actors, ranging from identity theft and account takeovers to sophisticated synthetic identity fraud. Consequently, Financial Fraud Detection (FFD) has become a critical area of research within the domain of cybersecurity and computational intelligence. While Machine Learning (ML) algorithms—specifically Supervised and Unsupervised learning models—have demonstrated superior capability in identifying non-linear fraud patterns compared to traditional rule-based systems, their development is significantly hindered by a fundamental bottleneck: the scarcity of high-quality, publicly available training data.

Data privacy regulations, such as the General Data Protection Regulation (GDPR) in Europe and various Payment Card Industry Data Security Standards (PCI DSS), rightfully restrict the dissemination of sensitive financial records. This creates a "data vacuum" for researchers and independent developers who wish to innovate in the FFD space. Publicly available datasets are often heavily anonymized (e.g., using Principal Component Analysis transformations) or lack the temporal and behavioral granularity required to model complex fraud scenarios effectively. Furthermore, real-world data is rarely "clean"; it is characterized by missing values, system errors, and extreme class imbalance—challenges that pristine academic datasets often fail to replicate.

To address these limitations, this paper presents the design and implementation of a Generative Augmented Fraud Detection Framework. Unlike traditional studies that rely on static datasets or simple statistical sampling, this research establishes a self-contained "Generative AI Lab." The core contribution of this work is the integration of a Conditional Tabular Generative Adversarial Network (CTGAN). This Deep Learning module is capable of learning the high-dimensional joint probability distribution of transaction features, allowing the system to synthesize realistic, privacy-preserving transaction logs that capture complex non-linear correlations—such as the interplay between international status and transaction amount—better than traditional probabilistic methods.

The proposed system operates on a "Generate-Augment-Detect" paradigm. First, a statistical engine creates a baseline of user behaviors. Second, the Deep Generative Module (CTGAN) augments this baseline by "dreaming" novel fraud scenarios, enriching the dataset with synthetic samples that mimic the statistical properties of real fraud without exposing sensitive information. To ensure the robustness of downstream models, a Noise Injection Mechanism is integrated into the pipeline, artificially introducing missing values and statistical outliers to mimic the "dirty" nature of production data streams.

Subsequently, the detection architecture employs a Hybrid Intelligence Strategy. It combines domain-specific heuristics—implemented as a deterministic "Rule Score"—with advanced ML algorithms. We evaluate the performance of Logistic Regression, Gradient Boosting, and Random Forest classifiers, optimizing the latter via RandomizedSearchCV for maximum recall. To mitigate the risk of "unknown unknowns" (novel fraud attacks), the system augments supervised classification with an unsupervised IsolationForest model for geometric anomaly detection.

Finally, to bridge the gap between theoretical modeling and practical deployment, the framework is operationalized via an interactive web application built with Streamlit. This interface facilitates real-time sensitivity analysis, allowing stakeholders to visualize the impact of fraud ratios and anomaly contamination levels dynamically.

The remainder of this paper is structured to provide a comprehensive analysis of the proposed framework. Section II establishes the Theoretical Background, outlining the mathematical foundations of the Generative Adversarial Networks (GANs) and Supervised learning algorithms utilized. Section III details the Deep Generative Pipeline, elucidating the CT-GAN training process and noise injection protocols. Section IV describes the Machine Learning Methodology, focusing on the hybrid feature engineering approach and model comparison. Section V presents the Application Implementation, and Section VI discusses the experimental Results demonstrating the efficacy of Gen-AI augmentation.

II. Theoretical Background

The detection of fraudulent patterns in financial systems is fundamentally a classification problem, complicated by extreme class imbalance and the evolving nature of adversarial attacks. This research leverages a multi-faceted approach, employing Supervised Learning for pattern recognition, Unsupervised Learning for anomaly detection, Stochastic Simulation for baseline data synthesis, and Deep Generative Learning for data augmentation. This section delineates the mathematical foundations of the algorithms utilizing in the framework.

A. Supervised Learning Framework

Supervised learning algorithms aim to learn a mapping function $f : X \to Y$, where input vectors X (transaction features) are mapped to discrete output labels $Y \in \{0, 1\}$ (Legit, Fraud).

1) *Random Forest Classifier:* The primary engine of our detection pipeline is the Random Forest (RF) algorithm. RF is an ensemble learning method that constructs a multitude of Decision Trees at training time. It operates on the principle of Bootstrap Aggregating (Bagging), which reduces the variance of individual decision trees and mitigates overfitting—a critical requirement when dealing with noisy financial data.
   For a given feature set, the Random Forest selects a random subset of features at each candidate split. The quality of a split is typically measured using Gini Impurity. For a node t with samples belonging to C classes, the Gini Impurity $I_G(t)$ is defined as:
$$I_G(t) = 1 - \sum_{i=1}^{C} p(i|t)^2$$
   Where $p(i|t)$ is the probability of a randomly selected sample belonging to class i at node t. The algorithm seeks to minimize this impurity. The final classification for a new instance is determined by the majority vote of the ensemble trees:
$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_N(x)\}$$
   Where $h_k(x)$ is the prediction of the k-th tree in the forest.
2) *Gradient Boosting Classifier:* To provide a comparative baseline, we employ Gradient Boosting. Unlike Random Forest, which builds trees in parallel, Gradient Boosting builds trees sequentially. Each new tree $h_{t+1}(x)$ attempts to correct the residual errors of the previous ensemble $F_t(x)$. The model minimizes a loss function $L(y, F(x))$ (typically Log Loss for classification) using gradient descent:
$$F_{t+1}(x) = F_t(x) + \gamma h_{t+1}(x)$$
   While powerful, Gradient Boosting is often more sensitive to noise and outliers than Random Forest, making it an ideal candidate for comparison in our "dirty data" environment.
3) *Logistic Regression:* As a baseline linear model, Logistic Regression models the probability that a given input X belongs to the fraud class ($Y = 1$) using the sigmoid function:
$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta^T X)}}$$
   This model serves as a benchmark to demonstrate the necessity of non-linear models (like RF) in capturing complex fraud patterns such as the interaction between "Time of Day" and "Transaction Amount".

B. Unsupervised Anomaly Detection

Supervised models are limited to detecting known fraud patterns. To identify novel or "zero-day" attacks, we incorporate the Isolation Forest (iForest) algorithm. Unlike distance-based methods (e.g., K-Means) which are computationally expensive, iForest relies on the principle that anomalies are "few and different." It isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

The core logic is that anomalies are easier to isolate (require fewer splits) than normal points. The anomaly score $s(x, n)$ for an instance x is defined as:
$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$
Where:

- $h(x)$ is the path length (number of edges) from the root to the leaf node for sample x.
- $E(h(x))$ is the average path length over a collection of isolation trees.
- $c(n)$ is the average path length of an unsuccessful search in a Binary Search Tree (BST), used as a normalization factor.

Scores close to 1 indicate high certainty of an anomaly (fraud), while scores significantly smaller than 0.5 indicate normal instances.

C. Stochastic Data Simulation Theory

To synthesize realistic baseline data, we model human behavior using fundamental probability distributions rather than uniform randomness. This allows us to encode "signals" into the dataset that the ML models can learn.

- *Normal Distribution ($\mathcal{N}$):* Transaction amounts are modeled using a Gaussian distribution, defined by a mean (μ) and variance (σ^2).

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

  We use separate parameters for legit users (μ_{legit}) and fraudsters (μ_{fraud}) to create a distinguishable decision boundary.
- *Binomial Distribution (B):* Binary behavioral features, such as is_international (International Transaction), are modeled as Bernoulli trials where the probability of success p varies significantly between classes ($p_{fraud} \gg p_{legit}$).

D. Generative Adversarial Networks (Mathematical Formulation)

To augment the stochastic baseline and overcome data scarcity, we employ a **Conditional Tabular GAN (CTGAN)**. The core generative component is based on the Wasserstein GAN with Gradient Penalty (WGAN-GP) optimization objective. The objective function is defined as a min-max game where the Generator G tries to minimize the distance between the real data distribution $\mathbb{P}_r$ and the generated distribution $\mathbb{P}_g$, while the Critic D tries to maximize it:

$$\mathcal{L}_{WGAN} = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] + \lambda \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

Where the term $\lambda(\dots)^2$ represents the **Gradient Penalty**, which enforces the Lipschitz-1 constraint, preventing the "Mode Collapse" common in financial data synthesis.

Since transaction amounts follow a multi-modal distribution (peaks at \$500 and \$3500), CTGAN uses a **Variational Gaussian Mixture (VGM)** model. A continuous value c_i is represented as:

$$c_i = \alpha_i \oplus \beta_i$$

Where α_i is a one-hot vector indicating the selected mode (cluster), and β_i is a scalar representing the value within that mode. This ensures the GAN can generate "Whale" transactions (outliers) accurately without smoothing over the distinct clusters of legitimate and fraudulent behavior.

III. System Architecture and Data Engineering

To ensure scalability and reproducibility, the proposed framework is architected as a modular, three-tier system: the **Generative Synthesis Layer**, the **Analytical Engine**, and the **Interactive Presentation Layer**. This modularity allows for individual components—such as the Deep Learning generator or the classification model—to be updated without disrupting the broader pipeline.

A. High-Level Data Flow

The system operates on a linear execution pipeline designed to simulate the full lifecycle of a Generative AI project, from raw data creation to production inference. The data flow follows five distinct stages:

1) *Statistical Initialization:* The generate_fraud_data module creates a baseline of pristine transaction logs based on probabilistic user profiles.
2) *Deep Generative Augmentation:* The system passes the baseline data into the **CTGAN Module**. This Deep Learning component learns the latent feature space and synthesizes novel, complex fraud samples that augment the original dataset.
3) *Corruption & Noise Injection:* The `inject_noise` module acts as a "chaos monkey," intentionally degrading data quality to mimic production environments.
4) *Feature Enrichment:* The raw data is passed to the `add_rule_score` processor, which appends heuristic risk scores alongside standard features.
5) *Model Training & Inference:* The data is split into Stratified K-Folds. The system trains supervised models (Random Forest) and unsupervised models (Isolation Forest) in parallel, creating a serialized pipeline served via the Streamlit web interface.

B. Deep Generative Augmentation (CTGAN)

To augment the baseline statistical data, the system integrates the **Conditional Tabular GAN (CTGAN)** architecture. Unlike standard GANs which struggle with discrete columns, CTGAN employs **Mode-Specific Normalization** to model continuous variables and a **Conditional Generator** to handle class imbalances.

The architecture consists of two adversarial neural networks:

1) **The Generator (G):** A deep residual network that transforms a random latent noise vector z into a synthetic transaction row x_{syn}.
2) **The Critic (D):** A discriminator network trained to distinguish between the generated transactions and the original "seed" data.

The system utilizes this GAN to synthesize minority-class samples (fraud cases), thereby enriching the training dataset with novel, non-linear fraud signatures that were not explicitly programmed in the rule-based engine.

C. The "Hybrid Intelligence" Design

The architecture explicitly rejects the "Black Box" approach common in deep learning applications. By inserting a Rule-Based Heuristic Layer between the Generative Data and the Machine Learning models, the system creates a "Hybrid Intelligence" architecture composed of three intelligence layers:

1) *Layer 1 (Deterministic Logic):* This layer applies hard-coded business rules extracted from domain expertise. For example, the system flags any transaction exceeding a predefined threshold (e.g., $2500) or occurring during high-risk hours (00:00–06:00). These rules provide an immediate, explainable baseline score (`rule_score`).
2) *Layer 2 (Probabilistic Classification):* The Random Forest model consumes both the raw features (e.g., amount, hour) and the Layer 1 score. This allows the model to learn non-linear nuances—such as a user who transacts at 3:00 AM but has a history of legitimate low-value purchases—which a simple rule would misclassify.
3) *Layer 3 (Geometric Anomaly Detection):* Running in parallel, the `IsolationForest` algorithm ignores labels entirely. It detects outliers that bypass Layer 1 and Layer 2 by analyzing the geometric density of the feature space. This serves as a safety net for "zero-day" fraud patterns that do not match historical signatures.

D. Implementation Stack

The system is implemented entirely in Python 3.9+, leveraging a specific stack of libraries chosen for their performance in vectorized operations and interactive capabilities:

- *SDV (Synthetic Data Vault):* Provides the PyTorch-based implementation of the CTGAN architecture used for the Generative AI augmentation.
- *NumPy & Pandas:* Employed for vectorized data manipulation and statistical initialization.
- *Scikit-Learn:* Provides the algorithmic backbone for the Discriminator/Classifier models. We utilize `Pipeline` objects to ensure that data transformations (like scaling) are encapsulated, preventing data leakage during the Cross-Validation phase.
- *Streamlit:* This framework serves as the application server. It utilizes a reactive programming model where the entire script re-runs upon widget interaction, ensuring that the "Custom Transaction Tester" always utilizes the most current state of the model hyperparameters.

IV. Data Engineering and Generation Strategy

The core innovation of this framework is the Synthetic Data Engine, designed to overcome the "Cold Start" problem in fraud detection where labeled training data is unavailable or legally restricted. Unlike simple random number generation, which fails to capture the subtle correlations found in financial logs, our engine utilizes a Behavior-Based Stochastic Model. This module is responsible for synthesizing user identities, merchant interactions, and complex transactional metadata.

A. Entity and Attribute Synthesis

The generation process begins with the initialization of the simulation environment. The system accepts a set of configuration hyperparameters: total sample size (N), fraud ratio (ρ), and a random state seed (θ) for reproducibility.

1) *Identity Generation:* To mimic a realistic payment network, the system generates discrete sets of Users (U) and Merchants (M).
   - *User Population:* The number of unique users is derived dynamically as $N_{users} = \max(50, N/20)$. This ensures a realistic density of transactions per user, allowing the model to potentially learn user-specific history rather than treating every row as independent.
   - *Merchant Ecosystem:* Similarly, merchant density is calculated as $N_{merchants} = \max(20, N/50)$, reflecting the real-world reality that users transact with a finite set of vendors.
2) *Class Balancing Strategy:* The dataset is stratified into two distinct classes: Legitimate (L) and Fraudulent (F). The number of fraud samples is strictly controlled by the fraud ratio ρ:

$$N_{fraud} = \lfloor N \times \rho \rfloor$$

$$N_{legit} = N - N_{fraud}$$

This strict stratification allows us to test the model's performance under varying degrees of class imbalance. The system defaults to $\rho = 0.15$ (15% fraud), but the Streamlit interface allows this to be adjusted dynamically between 5% and 25% to simulate different threat environments.

B. Behavioral Modeling and Feature Distributions

The engine creates features by sampling from distinct statistical distributions for legitimate and fraudulent classes. This creates a decision boundary that is learnable but non-trivial.

1) *Temporal Patterns:* Time is a critical indicator in fraud detection. Legitimate human behavior follows a circadian rhythm, while automated fraud scripts or stolen card misuse can occur at any hour.
   - *Legitimate Behavior:* Modeled using a uniform distribution restricted to waking hours, specifically $t \in [7, 23]$ (7:00 AM to 11:00 PM).
   - *Fraudulent Behavior:* Modeled as a uniform distribution across the full 24-hour cycle $t \in [0, 24]$. This probabilistic overlap creates a "Risk Zone" between 00:00 and 06:00 where the probability of fraud $P(Fraud|Hour < 7)$ increases significantly. The model must learn that while legitimate transactions can happen at 2:00 AM, they are statistically rare.
2) *Financial Transaction Values:* Transaction amounts are modeled using independent Gaussian distributions. We

assume fraud transactions aim for higher value extraction.

- $Amount_{legit} \sim \mathcal{N}(\mu = 500, \sigma = 150)$
- $Amount_{fraud} \sim \mathcal{N}(\mu = 3500, \sigma = 900)$ The significant separation in means (μ) provides a strong signal, but the overlap in standard deviations (σ) ensures that high-value legitimate purchases are occasionally conflated with low-value fraud.

3) *Geospatial Risk Indicators:* We simulate geographical risk via the is_international and country_risk flags.
   - *International Transactions:* Modeled as a Bernoulli process.
     $P(Intl=1 — Legit) = 0.05
     $P(Intl=1 — Fraud) = 0.40
   - *Country Risk Score:* A categorical variable $\{0, 1, 2\}$ representing Low, Medium, and High-risk jurisdictions. The generator utilizes a weighted choice function where legitimate transactions prefer Low Risk ($p = [0.7, 0.3]$ for 0/1), whereas fraud skews heavily toward High Risk ($p = [0.3, 0.7]$ for 1/2).

C. Noise Injection and Data Corrupting

A perfectly clean dataset creates "Lab Models" that fail in production. To enhance robustness, the inject_noise module introduces controlled entropy into the dataset before training.

1) *Missing Value Simulation:* In real-world streaming data, fields often drop due to latency or parsing errors. We randomly select a fraction $\alpha_{miss} = 0.02$ (2%) of the rows and forcibly set the amount to NaN. This forces the pipeline to implement an imputation strategy (Median Imputation) rather than crashing.
2) *Outlier Injection:* To stress-test the linear models (like Logistic Regression), we introduce extreme outliers. A subset of rows $\alpha_{out} = 0.01$ (1%) is selected, and their transaction amounts are multiplied by a uniform random factor $k \sim U(3, 8)$.

$$Amount_{new} = Amount_{original} \times k$$

This results in transactions ranging from 3x to 8x their normal value, creating "Whale" transactions that can skew the decision boundary of non-robust algorithms.

V. Machine Learning Methodology

The detection framework utilizes a hybrid architecture that integrates deterministic heuristic scoring with probabilistic classification models. This approach ensures that the system is robust against both obvious, high-value fraud (caught by rules) and subtle behavioral anomalies (caught by ML).

A. Hybrid Feature Engineering: The Rule-Based Heuristic

To accelerate model convergence and provide interpretability, we engineer a synthetic feature known as the Rule Score ($\mathcal{S}_{rule}$). This score encodes domain-specific risk thresholds directly into the feature space.Let x be a transaction vector. The score is calculated as a weighted summation of indicator functions $\mathbb{I}(\cdot)$, defined formally as:

$$\mathcal{S}_{rule}(x) = \sum_{k=1}^{K} w_k \cdot \mathbb{I}(C_k(x))$$

Based on the logic implemented in the add_rule_score function, the specific formulation used is:

$$\mathcal{S}_{rule} = \mathbb{I}(Amt > 2500) + \mathbb{I}(Hr < 6) + 2 \cdot \mathbb{I}(Intl) + \mathbb{I}(Risk_{ctry} = High)$$

- High Value Weight ($w = 1$): Flags transactions exceeding $2500.
- Temporal Risk ($w = 1$): Captures the "vampire window" (00:00–06:00).
- International Weight ($w = 2$): Assigns double variance to cross-border transactions, reflecting their higher base probability of fraud.
- Jurisdictional Risk ($w = 1$): Flags high-risk country codes.

This heuristic creates a "warm start" for the downstream classifiers, allowing them to differentiate between high-risk legitimate users (e.g., business travelers) and actual fraudsters.

B. Preprocessing and Imputation

Before training, the raw feature matrix X undergoes a preprocessing pipeline to handle the noise injected during the data engineering phase.

1) *Imputation:* The randomly injected NaN values in the amount column are imputed using the Median strategy. The median is selected over the mean to resist the influence of the extreme outliers (3x–8x multipliers) introduced by the noise engine.
2) *Scaling:* While Tree-based models (Random Forest) are generally invariant to scaling, the inclusion of Logistic Regression for baseline comparison necessitates standardization (Zero Mean, Unit Variance) for convergence.

C. Model Selection and Configuration

We evaluate three distinct supervised learning algorithms to determine the optimal boundary for the synthetic dataset:

1) *Logistic Regression:* Serves as the linear baseline. It is configured with a maximum iteration limit of 1000 to ensure convergence on the unscaled "dirty" data.
2) *Gradient Boosting Classifier:* A boosting ensemble that minimizes bias.
3) *Random Forest Classifier:* A bagging ensemble that minimizes variance.

Hyperparameter Optimization: The Random Forest model is subjected to a rigorous tuning process using RandomizedSearchCV to optimize for Recall. The search space Θ is defined as follows:

- *Number of Estimators (N_{est}):* $\{100, 200, 300\}$
- *Maximum Depth (D_{max}):* $\{None, 5, 10, 20\}$
- *Split Criterion:* Gini Impurity vs. Entropy
- *Min Samples Split:* $\{2, 5, 10\}$

This tuning process prevents the model from overfitting to the specific noise patterns while maximizing the detection of the minority fraud class.

D. Unsupervised Anomaly Detection

To detect novel fraud patterns that do not match historical signatures (i.e., "unknown unknowns"), the system employs an Isolation Forest.The model is configured with a contamination parameter equal to the expected fraud ratio ($\rho \approx 0.15$).

$$AnomalyScore(x) = -\mathbb{E}[h(x)]$$

The system calculates the decision function for every transaction, where a lower path length $h(x)$ corresponds to a higher anomaly score. In the application layer, this score is normalized and presented alongside the supervised probability to provide a "second opinion" on ambiguous transactions.

VI. Application Implementation and User Interface

To bridge the gap between theoretical modeling and practical deployment, the framework is operationalized via a web-based dashboard built on the Streamlit library. This interface serves two critical functions: it allows stakeholders to visualize the impact of class imbalance on model performance in real-time, and it provides a "sandbox" environment for testing specific transaction scenarios against the trained model.

A. SReactive Architecture and State Management

The application follows a reactive programming paradigm. Unlike traditional request-response web servers (e.g., Flask or Django), the Streamlit application re-executes the entire script logic whenever a user interacts with a widget. This ensures that the model state is always consistent with the user's input parameters.

To manage the computational overhead of re-training models on every interaction, we rely on the efficient in-memory data structures of pandas. The data generation and training pipeline is encapsulated within the run_app() function, which acts as the main entry point. The separation of concerns is maintained by importing the core logic engines (generate_fraud_data, inject_noise) directly from the backend script, ensuring that the CLI and Web versions share the exact same mathematical foundation.

B. Dynamic Sensitivity Analysis

The application Sidebar serves as the control center for the simulation environment. It allows users to manipulate the core hyperparameters of the data engine dynamically:

1) *Fraud Scenario Toggle:* Users can select between "Low Fraud" (%), "Medium Fraud" (5%), and "High Fraud" (5%). This immediately triggers a regeneration of the synthetic dataset, allowing researchers to observe how the Recall metric fluctuates as the dataset becomes more or less balanced.
2) *Anomaly Contamination Slider:* This widget controls the sensitivity of the IsolationForest. By adjusting the contamination parameter (ranging from 0.01 to 0.30), users can visually inspect the trade-off between false positives (legitimate transactions flagged as anomalies) and detection coverage.

C. The Custom Inference Engine

A distinct feature of the application is the Custom Transaction Tester. This module simulates a "Human-in-the-Loop" workflow where an analyst manually validates a suspicious transaction.

The interface provides a form for inputting raw features: Amount, Hour, International Status, and Country Risk. Upon submission, the system performs On-the-Fly Feature Engineering:

1) It accepts the raw inputs.
2) It calculates the heuristic rule_score in real-time using the same logic as the training pipeline (e.g., adding +2 for International).
3) It constructs a single-row DataFrame and feeds it into the pre-trained Random Forest model.

The result is displayed not just as a binary label, but as a probabilistic risk assessment (e.g., "Fraud Probability: 0.85 $\rightarrow$ HIGH RISK"). This immediate feedback loop is essential for explaining the model's decision-making process to non-technical stakeholders.

VII. Experimental Results and Performance Analysis

The experimental evaluation focuses on quantifying the trade-off between false positives (legitimate transactions flagged as fraud) and false negatives (missed fraud). Given the financial context, **Recall** is prioritized over Accuracy, as the cost of missing a fraudulent transaction significantly outweighs the administrative cost of verifying a suspicious one.

A. Comparative Analysis of Classifiers

We evaluated three supervised learning algorithms on a synthetic test set of $N_{test} = 600$ transactions (30% of the generated 2000 samples). The models were trained on the remaining 70% with a fraud ratio $\rho \approx 0.15$. We conducted an ablation study to measure the impact of Gen AI augmentation.

Model Configuration	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.895	0.820	0.580	0.679
Gradient Boosting	0.940	0.890	0.860	0.875
Random Forest (Statistical Only)	0.965	0.910	0.940	0.925
Random Forest (Gen AI + GAN)	**0.972**	**0.915**	**0.962**	**0.938**

TABLE I
Performance Metrics Comparison: Impact of Gen AI Augmentation

The **Gen AI Augmented** model achieved the highest detection capability. Specific observations include:

1) *Logistic Regression (Baseline):* The linear model achieved high accuracy but demonstrated poor recall. This underperformance is attributed to its inability to capture the non-linear "Vampire Effect" (interaction

between late hours and high amounts) without explicit polynomial feature engineering.

2) *Random Forest (Champion Model):* The Random Forest Classifier consistently outperformed the baseline. After hyperparameter tuning via `RandomizedSearchCV` and augmentation via **CTGAN**, the model achieved a Recall of approximately 0.96. The ensemble nature of the algorithm allowed it to robustly handle the injected noise (outliers) and effectively utilize the categorical `rule_score` feature.

B. Feature Importance Analysis

To understand the decision-making process of the "Black Box" Random Forest, we extracted the Gini Importance scores. The analysis reveals that the engineered Rule Score was the dominant predictor, confirming the hypothesis that hybrid intelligence outperforms raw features.

- *Amount ($\approx 50\%$):* The most critical feature, driving detection of high-value theft.
- *Rule Score ($\approx 25\%$):* The single most predictive feature, validating the effectiveness of the heuristic layer.
- *Country_Risk ($\approx 5.0\%$):* A supplementary indicator providing granular jurisdictional risk. Its lower importance reflects its partial redundancy with the 'Is International' flag and the aggregated Rule Score.
- *Hour ($\approx 5\%$):* Useful for distinguishing temporal anomalies but less decisive than the rule score.
- *Is International ($\approx 2\%$):* A strong indicator, correlating heavily with the fraud class.

C. Anomaly Detection Distribution

The `IsolationForest` provided a complementary risk assessment. The distribution of anomaly scores (visualized in the Streamlit dashboard) followed a bimodal distribution. Legitimate transactions clustered around a score of 0.6, while fraudulent entities produced negative scores or scores close to -0.8. This clear separation indicates that the geometric properties of the synthetic fraud data are distinct enough for unsupervised detection to function as an effective safety net.

VIII. Conclusion and Future Scalability

This research presented the design and implementation of a **Generative Augmented Fraud Detection Framework**. By synthesizing a behavior-based dataset and augmenting it via **Conditional Tabular GANs (CTGAN)**, we successfully created a robust laboratory environment for training fraud detection models without compromising user privacy.

The experimental results demonstrate that a **Hybrid Intelligence** approach—fusing deterministic domain rules, Deep Generative Learning, and a probabilistic Random Forest classifier—significantly outperforms standalone baselines. The GAN-augmented model achieved a Recall of over 96%, proving that financial fraud detection systems can effectively "dream" new attack vectors using Generative AI. Furthermore, the integration of the `IsolationForest` provided a necessary layer of defense against "unknown unknowns," validating the utility of unsupervised learning in this domain.

The operationalization of this pipeline via the Streamlit web application bridges the gap between theoretical data science and practical utility. It allows stakeholders to move beyond static metrics and engage with the model dynamically, testing hypotheses and visualizing risk in real-time.

A. Future Scalability Plans

While the current framework is a self-contained prototype, the architecture is designed for modular scalability. Future iterations of this system could expand in three key directions:

1) *Transformer-Based Sequence Learning:* While the current CTGAN handles tabular data effectively, replacing the Random Forest with Transformer architectures (such as TAB-BERT) would allow the model to learn sequential temporal patterns (e.g., a series of small test transactions followed by a large theft) more effectively than tree-based models.
2) *API Microservices:* The current monolithic Streamlit application can be decoupled. The inference engine could be deployed as a REST API (using FastAPI), allowing external banking systems to query the `predict_proba` endpoint programmatically.
3) *Graph Neural Networks (GNNs):* Moving from flat CSV files to a Graph Database (e.g., Neo4j) would allow for the detection of "Fraud Rings"—clusters of users sharing the same device or IP address—which is the next frontier in advanced fraud detection.

References

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

[2] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation Forest," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[3] Scikit-learn developers, "Scikit-learn: Machine Learning in Python," *JMLR*, 2011.

[4] N. V. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique," *JAIR*, 2002.

[5] Streamlit Inc., "Streamlit Documentation," [Online]. Available: https://docs.streamlit.io. [Accessed: 2025].

PaperRater

Abstract—The advancement of financial fraud detection is persistently hindered by the scarcity of public datasets, largely due to stringent privacy laws like GDPR and proprietary banking restrictions. To bridge this gap, this research proposes a Generative Augmented Fraud Detection Framework, a system designed to overcome data poverty through deep generative learning. Unlike traditional statistical simulations, this project integrates a Conditional Tabular GAN (CTGAN) to learn the high-dimensional joint probabilities of transaction features, syn thesizing realistic, privacy-preserving fraud logs that capture complex non-linear correlations. To ensure the models are robust against production-grade imperfections, the pipeline incorporates a... (only first 800 chars shown)

Analysis complete. Our feedback is listed below in printable form. Some of the items have been truncated or removed to provide better print compatibility.

Plagiarism Detection

Original Work

Originality: 90%

It looks like this paper is original, but check that un-original text is cited accordingly.

The following web pages may contain content matching this document:
https://www.etudier.com/dissertations/Marchand-...
https://arxiv.org/abs/1505.00389
https://www.bartleby.com/essay/Report-On-The-Di...
https://arxiv.org/abs/2012.11774
https://arxiv.org/abs/2004.00491
https://arxiv.org/abs/2012.13796
http://www.mdpi.com/2071-1050/11/5/1439
https://arxiv.org/abs/1911.10728
https://www.linkedin.com/pulse/article-authenti...
https://www.bartleby.com/essay/Information-Syst...
View Matching Text
Click the button above to see which portions of your text match which sources. This is a premium-only feature.

More info on our originality scoring process.

Word Choice

Usage of Bad Phrases

Bad Phrase Score: 0.37 (lower is better)
The Bad Phrase Score is based on the quality and quantity of trite or inappropriate words, phrases, egregious misspellings, and cliches found in your paper. You did equal or better than **100%** of the people in your grade.

Outstanding use of quality phrases! Your score is significantly above average.

Style

Usage of Transitional Phrases

Transitional Words Score: 48
This score is based on quality of transitional phrases used within your paper. You did equal or better than **22%** of the people in your grade.

Your usage of transitional phrases is below average. Please review the writing tips below.

One sign of an excellent writer is the use of transitional phrases (e.g. therefore, consequently, furthermore). Transitional words and phrases contribute to the cohesiveness of a text and allow the sentences to flow smoothly. Without transitional phrases, a text will often seem disorganized and will most likely be difficult to understand. When these special words are used, they provide organization within a text and lead to greater understanding and enjoyment on the part of the reader.

The following transitional phrases were found in your document:
specifically, and, finally, furthermore, subsequently, consequently

Consider using additional transitions where appropriate:
- moreover
- nevertheless
- notwithstanding
- accordingly
- conversely
- ordinarily

Transitional phrases may be used in various places in a text:
- between paragraphs
- between sentences
- between sentence parts
- within sentence parts

Consider this example:
She is one of the most kind persons I have ever known. **Moreover**, she is generous, patient and possesses a magnificent sense of humor.

The word **'moreover'** contributes to greater unity or cohesion between sentences and allows the text to flow more smoothly.

Style

Sentence Length Info

Total Sentences: 39
Avg. Length: 18.0 words
Short Sentences (< 17 words): 20 (51%)
Long Sentences (> 35 words): 2 (5%)
Sentence Variation: 10.3 words (std deviation)

Your average sentence length is within an acceptable range, but consider that effective use of sentence length cannot be easily measured.

Line chart of the length of each sentence (first 50 sentences). A jagged chart indicates variation.

Helpful Resources:
- Effective Use of Sentence Length

Vocabulary Words

Usage of Academic Vocabulary

Vocabulary Score: 1047.59
This score is based on the quantity and quality of scholarly vocab words found in the text. You did equal or better than **100%** of the people in your grade.

Vocabulary Word Count: 157
Percentage of Vocab Words: 26.53%
Vocab Words in this Paper (top 20):
interplay, stringent, augmented, framework, integrates, employs, probabilistic, operationalized, stakeholders, efficacy, precipitated, dissemination, temporal, implementation, adversarial, augment, paradigm, augments, subsequently, mitigate

Excellent work! Your usage of sophisticated words is on par with other well-written papers! Nevertheless, you may still wish to use our Vocab Builder to maintain your edge.

Tips
Whether you are writing for a school assignment or professionally, it is imperative that you have a vocabulary that will provide for clear communication of your ideas and thoughts. You need to know the type and level of your audience and adjust your vocabulary accordingly. It is worthwhile to constantly work at improving your knowledge of words. To help with this task, please consider using our Vocabulary Builder to improve your comprehension and usage of words.