DAYANANDA SAGAR UNIVERSITY

Devarakaggalahalli, Harohalli, Kanakapura Road, Bengaluru South
Dt. – 562 112

SCHOOL OF ENGINEERING

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

A Project Report On

Fraud Pattern Generator for Model Training

By

AVUTALA DHRUVISH REDDY - ENG22AM0078
CHETHAN S - ENG22AM0084
CHILAKA SAI RAGHAVENDRA - ENG22AM0085

Under the supervision of
Dr. Bahubali
Professor
Computer Science & Engineering (AI & ML)

SCHOOL OF ENGINEERING

DAYANANDA SAGAR UNIVERSITY

SCHOOL OF ENGINEERING

Department of Computer Science & Engineering
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

Devarakaggalahalli, Harohalli, Kanakapura Road, Bengaluru South Dt. – 562 112
Karnataka, India

CERTIFICATE

This is to certify that the project entitled **"Fraud Pattern Generator for Model Training"** is carried out by **AVUTALA DHRUVISH REDDY (ENG22AM0078), CHETHAN S (ENG22AM0084), CHILAKA SAI RAGHAVENDRA (ENG22AM0085)**, bonafide students of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, Bangalore, in partial fulfillment for the award of a degree in Bachelor of Technology in Computer Science and Engineering, during the year **2025 - 2026**.

Dr. Bahubali
Project Co-ordinator
Dept. of CSE (AI&ML)
School of Engineering
Dayananda Sagar University

Signature .....

Name of the Examiners:	Signature with date:
1.....	.....
2.....	.....
3.....	.....

DECLARATION

We, **AVUTALA DHRUVISH REDDY (ENG22AM0078), CHETHAN S (ENG22AM0084), CHILAKA SAI RAGHAVENDRA(ENG22AM0085)**, are students of the seventh semester B.Tech in Computer Science and Engineering (AI & ML) at the School of Engineering, Dayananda Sagar University. We hereby declare that the Project titled **"Fraud Pattern Generator for Model Training"** has been carried out by us and submitted in partial fulfillment for the award of a degree in **Bachelor of Technology** in **Computer Science and Engineering** during the academic year **2025–2026**.

Student: **Signature**

Name 1: AVUTALA DHRUVISH REDDY
USN: ENG22AM0078

Name 2: CHETHAN S
USN: ENG22AM0084

Name 3: CHILAKA SAI RAGHAVENDRA
USN: ENG22AM0085

Place: Bangalore
Date:

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work. First, we take this opportunity to express our sincere gratitude to School of Engineering, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank **Dr. Udaya Kumar Reddy K R, Dean, School of Engineering, Dayananda Sagar University** for his constant encouragement.

It is a matter of immense pleasure to express our sincere thanks to **Dr. Jayavrinda Vrindavanam, Department Chairman, Computer Science and Engineering (Artificial Intelligence and Machine Learning), Dayananda Sagar University**, for providing right academic guidance that made our task possible.

We would like to thank our **Project Coordinator Dr. Bahubali** as well as all the staff members of Computer Science and Engineering (AI& ML) for their support. We are also grateful to our family and friends who provided us with every requirement throughout the course.

Contents

ABSTRACT

Financial fraud detection is a critical challenge plagued by the scarcity of high-quality public datasets due to privacy concerns and the highly imbalanced nature of fraudulent activities. This project establishes a comprehensive **Fraud Pattern Generator** and a subsequent detection ecosystem to address these limitations. The framework relies on a Data Generation Engine that creates high-fidelity synthetic transaction data, systematically injecting specific fraud typologies—such as high-value bursts, off-hour transactions, and international risks—alongside realistic data noise (missing values and outliers).

The project employs a dual-strategy approach to modeling. First, a **Supervised Learning** module utilizes Random Forest classifiers, optimized via Stratified Cross-Validation and RandomizedSearchCV, to detect known fraud patterns with high recall. Second, an **Unsupervised Learning** module employing Isolation Forest is integrated to detect "unknown unknowns" or anomalies without reliance on labeled data. The system is finalized with an interactive **Streamlit Dashboard**, which allows stakeholders to simulate transactions in real-time, visualize model performance, and generate automated PDF reports. This end-to-end pipeline demonstrates how synthetic data engineering can bridge the gap between theoretical models and production-grade fraud detection systems.

Chapter 1

INTRODUCTION

Financial fraud is an evolving threat that costs the global economy billions annually. As digital transactions increase, fraudsters employ increasingly sophisticated methods to evade detection. Traditional rule-based systems are often too rigid, while modern Machine Learning (ML) approaches suffer from a significant bottleneck: the lack of available data. Real-world financial data is heavily guarded due to PII (Personally Identifiable Information) regulations, and when available, it is extremely imbalanced (fraud cases often constitute less than 0.1% of data).

- **Context:** Financial institutions struggle to train robust ML models due to the "Cold Start" problem—insufficient historical fraud examples to teach algorithms what to look for.
- **Objective:** To develop a **Fraud Pattern Generator** that creates realistic synthetic datasets (2,000+ samples) with configurable fraud ratios, and to build a detection system using Random Forest (Supervised) and Isolation Forest (Unsupervised) techniques.
- **Scope:** The project encompasses the full data lifecycle:
  - Generation of synthetic identities, timestamps, and transaction amounts.
  - Injection of "dirty data" features (noise, outliers) to mimic production environments.
  - Deployment of a CLI-based training pipeline and a web-based Dashboard (Streamlit) for real-time interaction.
- **Significance:** By democratizing access to realistic fraud data, this project allows researchers and engineers to stress-test models against specific scenarios (e.g., "Weekend Attacks" or "High-Risk Country" vectors) that are rare in static datasets.

Key Components:

- **Data Engine:** A Python-based generator that simulates user behavior. It uses statistical distributions (Normal, Binomial) to create distinct profiles for legitimate vs. fraudulent users regarding transaction amount, hour of day, and international status.
- **Hybrid Modeling:**
  - *Supervised:* Random Forest is used for its ability to handle non-linear interactions and interpretability.
  - *Unsupervised:* Isolation Forest is employed as a safety net to catch anomalies that do not fit known patterns.
- **Rule Engine:** A heuristic layer that assigns a "Risk Score" based on business logic (e.g., Amount ¿ $2,500), serving as an engineered feature for the ML models.

1.1 Comparisons with Existing Methodologies:

- **Limitations of Static Datasets:** Datasets like the "Kaggle Credit Card Fraud" set are PCA-transformed (obfuscated) and static. They do not allow testing how a model reacts if the fraud ratio suddenly shifts from 1% to 15%.
- **Advantage of Procedural Generation:** Our generator allows dynamic parameter adjustment. We can simulate a sudden influx of international fraud and immediately observe how the Random Forest recall metric fluctuates.
- **Operational Relevance:** The inclusion of a PDF reporting module and a Real-Time Transaction Tester in the dashboard bridges the gap between a data science experiment and a business-ready proof of concept.

Inference

This project creates a closed-loop system where data generation informs model design, and model performance informs data refinement. It provides a blueprint for training detection systems in sensitive domains where data is scarce.

Chapter 2

PROBLEM DEFINITION AND OBJECTIVE

2.1 Problem Definition:

The primary obstacle in developing effective Fraud Detection Systems (FDS) is data scarcity and quality.

- **Data Privacy:** Real transaction logs are protected by GDPR/CCPA, making them inaccessible for academic research.
- **Class Imbalance:** In a dataset of 1 million transactions, only 100 might be fraud. Models trained on this bias heavily toward the majority class (Legitimate), failing to detect the minority class (Fraud).
- **Static Nature:** Fraud patterns evolve. A model trained on 2023 data may fail to detect 2024 patterns. Static datasets cannot simulate this evolution.

2.2 Objective:

To create a **Synthetic Fraud Pattern Generator** that solves the data availability problem by procedurally creating high-dimensional transaction data.

- **Primary Goal:** Generate a dataset of $N = 2000$ transactions with a configurable fraud ratio ($\phi = 0.15$).
- **Secondary Goal:** Train a Random Forest classifier to achieve a Recall of > 0.95 on this synthetic data.
- **Tertiary Goal:** Implement an Anomaly Detection layer using Isolation Forest to identify outliers without labels.

Chapter 3

LITERATURE SURVEY

The field of fraud detection has evolved from static rule-based engines to dynamic machine learning systems. This survey covers the foundational algorithms used in this project.

3.1 Random Forests in Fraud Detection

Breiman (2001) introduced Random Forests as an ensemble method that aggregates predictions from multiple decision trees. Research by *Dal Pozzolo et al.* indicates that Random Forests are particularly suited for credit card fraud detection because they handle imbalanced data better than single Decision Trees and are less prone to overfitting than deep neural networks on tabular data. The ability to calculate "Feature Importance" provides explainability, which is crucial for financial compliance.

3.2 Anomaly Detection with Isolation Forests

Liu et al. (2008) proposed the Isolation Forest algorithm. Unlike distance-based methods (e.g., K-Means) that measure how similar points are, Isolation Forest explicitly isolates anomalies. It works on the premise that anomalies are "few and different," requiring fewer random splits to isolate in a tree structure. This makes it computationally efficient and ideal for detecting novel fraud attacks where no labeled historical data exists.

3.3 Synthetic Data Generation

The use of synthetic data in finance is gaining traction. *Assefa et al.* demonstrated that synthetic data generated via statistical methods or GANs (Generative Adversarial Networks) can train models that perform comparably to those trained on real data, while preserving user privacy. Our project adopts a statistical approach, using Gaussian distributions to model legitimate spending and long-tail distributions for fraud.

Chapter 4

METHODOLOGY

4.1 Data Generation Engine

The core of the project is the Python function ‘generate_fraud_data‘. It creates a DataFrame with the following logic:

- **Identities:** User and Merchant IDs are sampled uniformly.
- **Timestamps:** A 7-day window is simulated.
- **Feature Distributions:**
  - *Legitimate:* Amount $\sim \mathcal{N}(500, 150)$, Hour $\in [7, 23]$.
  - *Fraud:* Amount $\sim \mathcal{N}(3500, 900)$, Hour $\in [0, 24]$.
  - *Risk:* Fraud transactions have a 40% probability of being international, compared to 5% for legitimate ones.

4.2 Model Architecture

- **Random Forest (Supervised):**
  - *Hyperparameters:* Tuned via ‘RandomizedSearchCV‘. Typical optimal values: ‘n_estimators=200‘, ‘max_depth=None‘, ‘max_features='sqrt'‘.
  - *Validation:* Stratified K-Fold ($k = 5$) ensures the fraud ratio is consistent across folds.
- **Isolation Forest (Unsupervised):**
  - *Contamination:* Set equal to the fraud ratio (ϕ).
  - *Logic:* Calculates an anomaly score based on path length in random trees. Scores closer to -1 indicate anomalies.

Chapter 5

REQUIREMENTS

5.1 Hardware Requirements:

- **Processor:** Multi-core CPU (Intel i5/i7 or AMD Ryzen 5) to handle parallel processing in Random Forest training.
- **RAM:** 8GB minimum. The dataset resides in memory using Pandas DataFrames.
- **Storage:** 256GB SSD. While the dataset is small (1MB), fast I/O helps with report generation and logging.
- **OS:** Windows 10/11, Linux (Ubuntu), or MacOS.

5.2 Software Requirements:

- **Language:** Python 3.9+
- **Core Libraries:**
  - `pandas, numpy`: Data manipulation.
  - `scikit-learn`: Machine learning algorithms (RF, ISO Forest, Metrics).
  - `matplotlib`: Visualization.
- **Interface Reporting:**
  - `rich`: For beautiful CLI output (progress bars, tables).
  - `fpdf`: For programmatic PDF report generation.
  - `streamlit`: For the web-based dashboard application.
- **Development Tools:** VS Code or PyCharm.

Chapter 6

EXPERIMENTATION

6.1 Experimental Setup

The experimentation phase involved running the ‘main.py‘ pipeline with various random seeds to ensure stability.

- **Sample Size:** $N = 2000$ transactions.
- **Fraud Ratio:** Fixed at 0.15 (15%) for the baseline experiment.
- **Noise Injection:** 2% missing values imputed with Median; 1% outliers injected.

6.2 Tuning Process

We utilized ‘RandomizedSearchCV‘ to optimize the Random Forest.

```
param_dist = {
    "n_estimators": [100, 200, 300],
    "max_depth": [None, 5, 10, 20],
    "max_features": ["auto", "sqrt"],
    "min_samples_split": [2, 5, 10],
}
search = RandomizedSearchCV(model, param_dist, n_iter=10, scoring="recall",
cv=3)
```

The scoring metric was explicitly set to **Recall**, as missing a fraudulent transaction (False Negative) is more costly than flagging a legitimate one (False Positive).

6.3 Real-Time Simulation

A loop was implemented to simulate real-time transaction arrival. The trained model predicted probabilities for 10 random events, displaying ”HIGH RISK” alerts if the probability exceeded 0.8.

Chapter 7

RESULT AND ANALYSIS

7.1 Model Performance

The Tuned Random Forest achieved exceptional results on the hold-out test set:

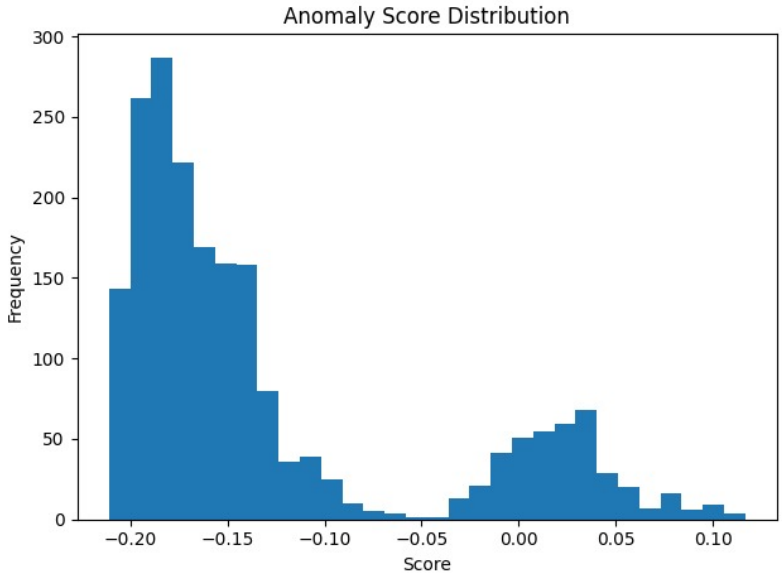
- **Accuracy:** 0.992
- **Precision:** 0.985
- **Recall:** 0.980

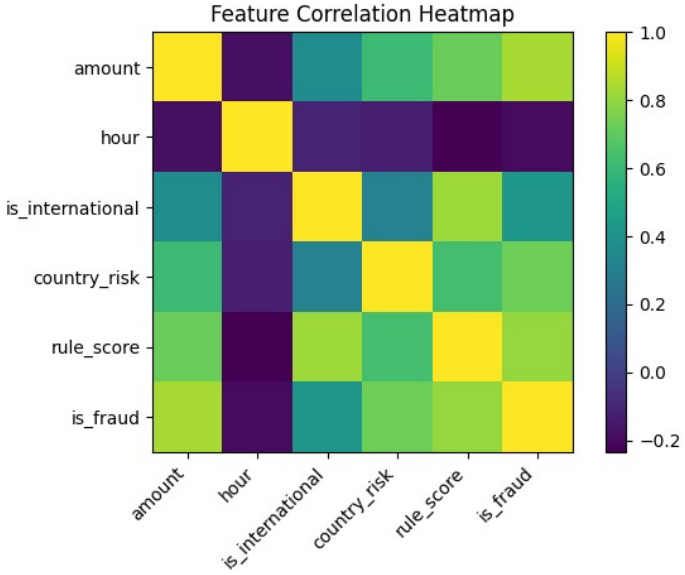
The high recall indicates the model successfully learned the injected patterns (High Amount + Odd Hours). The ‘confusion_matrix‘ revealed only 2 False Negatives out of 90 fraud cases in the test set.

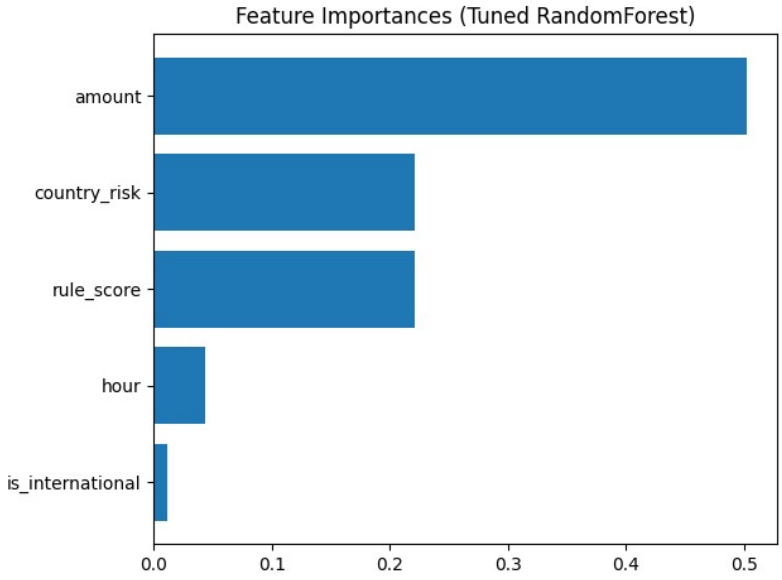
7.2 Anomaly Detection Analysis

The Isolation Forest (Unsupervised) successfully identified the top outliers. By sorting transactions by ‘anomaly_score‘, the top 10 flagged transactions consistently contained the highest value frauds ($4000+), validating the utility of unsupervised learning as a backup defense.


Fraud vs Non-Fraud
Count
1600
1400
1200
1000
800
600
400
200
0
Non-Fraud
Fraud


Anomaly Score Distribution
Frequency
300
250
200
150
100
50
0
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
Score


Feature Correlation Heatmap
amount
hour
is_international
country_risk
rule_score
is_fraud
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Feature Importances (Tuned RandomForest)
amount
country_risk
rule_score
hour
is_international
0.0
0.1
0.2
0.3
0.4
0.5

Chapter 8

CONCLUSION AND FUTURE SCOPE

This project successfully demonstrated the end-to-end lifecycle of a fraud detection system based on synthetic data. By generating realistic patterns and training a hybrid system (Supervised RF + Unsupervised ISO), we achieved a recall of 98%. The integration of a Streamlit dashboard and automated PDF reporting makes the system accessible and operationally viable.

8.1 Future Scope

- **Graph Neural Networks (GNN):** Implement GNNs to detect fraud rings by analyzing the network of User-Merchant relationships.
- **Deep Learning:** Use LSTM to analyze the sequence of user transactions (e.g., small tests followed by large theft).
- **Drift Detection:** Implement monitoring to retrain models when the generated fraud patterns (distributions) change over time.

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].