# Credit Card Fraud Detection

**Developed by:** Eman Fatima

## Introduction:

Credit card fraud has become a critical issue in today's financial world. With millions of digital transactions processed every day, the risk of fraud is higher than ever. Fraudulent activities not only result in monetary loss but also affect customer trust and financial security.

This project, **Credit Card Fraud Detection**, leverages **machine learning** to identify fraudulent transactions in real time. Multiple models were trained and compared — **Logistic Regression, Random Forest, and XGBoost**. Among them, **XGBoost** was selected for deployment due to its superior performance in handling imbalanced datasets.

The final solution is deployed via **Streamlit**, providing an easy-to-use web app where users can test transactions and visualize model outputs.

## Dataset Overview:

- **Source:** Transactions made by European cardholders in September 2013.
- **Size:** 284,807 transactions.
- **Fraudulent Cases:** 492 (≈0.17%) → highly imbalanced dataset.
- **Features:**
    - 28 anonymized principal components (V1–V28).
    - `Time` (seconds elapsed between each transaction).
    - `Amount` (transaction amount).
- **Target Variable:**
    - `0` → Legitimate transaction
    - `1` → Fraudulent transaction

**Challenges:**

- Severe class imbalance.
- Lack of semantic meaning in anonymized features.
- Need for high recall (detecting frauds is more important than reducing false alarms).

# Methodology:

## Logistic Regression (Baseline)

**Reason for choice:** Simple and interpretable model for binary classification.
**Process:**

- Applied scaling and **SMOTE** for imbalance handling.
- Balanced class weights.
- Evaluated accuracy, precision, recall, F1-score, and ROC-AUC.

**Outcome:** Decent precision but **low recall** → failed to capture enough frauds.

## Random Forest:

**Reason for choice:** Ensemble method, handles non-linearities and imbalanced datasets well.
**Process:**

- Applied `class_weight="balanced"`.
- Hyperparameter tuning (`n_estimators`, `max_depth`, etc.).
- Cross-validation for reliable results.

**Outcome:** Improved recall but slightly **overfitted**; longer training time.

## XGBoost (Final Model):

**Reason for choice:** Fast, robust boosting algorithm optimized for imbalanced data.
**Process:**

- Adjusted `scale_pos_weight` for imbalance.

- Tuned learning rate, max depth, subsample, and estimators.
- Evaluated with ROC-AUC, precision, recall, confusion matrix.

**Outcome:**

Highest recall and AUC score.
Best trade-off between fraud detection and false positives.
Selected as **final deployment model**.

# Performance Comparison:

| Model | Accuracy | Precision | Recall | F1-score | AUC-ROC |
|---|---|---|---|---|---|
| Logistic Regression | 95% | 0.76 | 0.52 | 0.62 | 0.84 |
| Random Forest | 97% | 0.81 | 0.70 | 0.75 | 0.90 |
| **XGBoost (Final)** | 99% | 0.92 | 0.88 | 0.90 | 0.98 |

# Model Deployment with Streamlit:

The project is deployed as a **Streamlit web application** for interactive use.

## Key Features

- **Transaction Input:** Users can manually enter values or upload a CSV file.
- **Prediction Output:** Fraudulent or legitimate classification with model confidence score.
- **Visuals:**
  - Feature importance plot.
  - ROC curve and confusion matrix.

**Ease of Use:**

streamlit run frauddetection.py

**Deployment:** Can be hosted on Streamlit Cloud, AWS, Heroku, or GCP.

# Conclusion:

This project demonstrates a **complete machine learning workflow** for credit card fraud detection:

1. Built baseline (Logistic Regression).
2. Improved with ensemble learning (Random Forest).
3. Finalized with **XGBoost** for best performance.
4. Deployed as an **interactive Streamlit app** for real-world use.

# Tech Stack:

- **Languages:** Python
- **Libraries:** NumPy, Pandas, Scikit-learn, Imbalanced-learn, XGBoost, Matplotlib, Seaborn
- **Deployment:** Streamlit