# 💳 Credit Card Fraud Detection — Exploratory & Predictive Analysis

### 🎯 Project Overview

This project analyzes and models \*\*credit card transaction data\*\* to detect fraudulent activities.

It focuses on \*\*Exploratory Data Analysis (EDA)\*\*, \*\*feature engineering\*\*, and \*\*model evaluation\*\*, providing a strong analytical baseline before experimenting with advanced machine learning methods.

## 🧠 Objective

To build a \*\*data-driven fraud detection baseline\*\* through exploratory analysis and logistic regression modeling,

and later extend to \*\*tree-based ensemble models\*\* (Random Forest, XGBoost).

## 🧾 Dataset Overview

The dataset used in this project is the public \*\*Credit Card Fraud Detection\*\* dataset

from [Kaggle](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud),

containing anonymized credit card transactions made by European cardholders in \*\*September 2013\*\*.

---

### 📋 Basic Information

| Property | Description |

|-----------|--------------|

| \*\*Rows\*\* | 284,807 transactions |

| \*\*Fraud Cases\*\* | 492 (≈ 0.17%) → highly imbalanced |

| \*\*Columns\*\* | 31 |

| \*\*Feature Types\*\* | Numerical only (no categorical variables) |

| \*\*Target Variable\*\* | `Class` — 1 = Fraud, 0 = Normal |

\*\*Feature Details:\*\*

- \*\*V1–V28:\*\* anonymized numerical features generated via \*\*PCA (Principal Component Analysis)\*\*

— the original sensitive variables (e.g., cardholder info, merchant type) were transformed for confidentiality.

- \*\*Time:\*\* seconds elapsed since the first transaction in the dataset.

- \*\*Amount:\*\* transaction amount.

- \*\*Class:\*\* binary label indicating fraud status.

---

### 📊 Data Notes

- PCA ensures privacy by removing identifiable information while preserving statistical structure.

- Because of PCA, `V1–V28` lack direct interpretability,

so the analysis focuses on \*\*distribution shapes\*\*, \*\*outliers\*\*, and \*\*correlations\*\* with `Class`.

- The dataset is \*\*heavily imbalanced\*\*, meaning traditional accuracy metrics can be misleading.

Therefore, metrics such as \*\*Precision\*\*, \*\*Recall\*\*, \*\*F1-score\*\*, and \*\*PR-AUC\*\* are prioritized.

---

### 🔍 Feature Selection Strategy

Since PCA already standardized and decorrelated most features,

feature selection focused on identifying those most correlated with fraudulent behavior.

\*\*Steps:\*\*

1. \*\*Correlation with `Class`:\*\* Found strongest relationships in `V11`, `V4`, `V12`, `V14`, `V17`.

2. \*\*KDE Distribution Comparison:\*\* Compared fraud vs. non-fraud distributions for each feature.

3. \*\*Skewness & Scaling:\*\*

- Log-transformed `Amount` → `log\_amount`

- Standardized `Time` → `Time\_scaled`

4. \*\*Final Feature Set for Baseline Model:\*\*

```python

['V4', 'V10', 'V11', 'V12', 'V14', 'V17', 'log\_amount', 'Time\_scaled']---

## 📊 Exploratory Data Analysis (EDA)

> Understand data distribution, detect imbalance, and visualize relationships.

### Key Steps:

- Checked missing values, outliers, and feature distributions.

- Visualized PCA-based features (V1–V28) and their relationship to `Class` (fraud vs. non-fraud).

- Identified severe \*\*class imbalance\*\* (fraud ≈ 0.17%).

- Scaled numeric variables (`Time`, `Amount`) for model stability.

### Example Plots:

- KDE plots comparing fraud vs. non-fraud transactions.

- Correlation heatmap among PCA features.

- Boxplots showing skewness reduction after log transformation.

---

## ⚙️ Feature Engineering

> Transform and enhance data for better model learning.

- \*\*Scaling:\*\* Standardized `Time` and `Amount` using `StandardScaler`.

- \*\*Transformation:\*\* Log-transformed `Amount` to reduce skewness.

- \*\*Encoding:\*\* Not applicable (all numeric features).

- \*\*Imbalance handling:\*\* Prepared baseline before applying SMOTE or undersampling.

---

## 🧮 Baseline Modeling — Logistic Regression

> Build an interpretable benchmark before complex models.

- \*\*Model:\*\* Logistic Regression (L2 regularization, tuned via GridSearchCV).

- \*\*Evaluation Metrics:\*\*

- ROC-AUC = \*\*0.97\*\*

- PR-AUC = \*\*0.74\*\*

- F1-score (fraud) = \*\*0.63\*\*

- \*\*Conclusion:\*\*

The model generalizes well (train ROC-AUC 0.98 vs test 0.97) and shows minimal overfitting.

Despite moderate recall (0.49), it effectively captures fraud probability patterns.

---

## 🌲 Advanced Modeling (Future Work)

> Move beyond linear models to capture nonlinear patterns.

### Planned Extensions:

- \*\*Tree-based models:\*\* Random Forest and XGBoost to capture feature interactions.

- \*\*Imbalance handling:\*\*

- `class\_weight='balanced'`

- Oversampling (SMOTE) / Undersampling

- \*\*Evaluation:\*\* Compare F1, ROC-AUC, and PR-AUC across models.

---

## 📈 Key Takeaways

- EDA and Logistic Regression together form a \*\*strong baseline\*\* for fraud detection.

- \*\*PR-AUC\*\* is prioritized over accuracy and ROC-AUC due to severe class imbalance.

- Ensemble models (e.g., XGBoost) can potentially improve recall while maintaining precision.

---

## 🧰 Tech Stack

\*\*Languages:\*\* Python (3.10)

\*\*Libraries:\*\* pandas, numpy, matplotlib, seaborn, scikit-learn, imbalanced-learn, xgboost

\*\*Environment:\*\* Jupyter Notebook

---

## 📚 Learnings

This project strengthened my understanding of:

- End-to-end data preprocessing and feature engineering workflows.

- Evaluating models on imbalanced data using PR-AUC and F1-score.

- Interpreting regularization (C, L1/L2) and its effect on logistic regression.

---

## 🤖 Author

\*\*Wen Zhang

---

## 💬 Optional: Project Motivation

Credit card fraud detection is one of the most critical challenges in modern finance.

Even a small improvement in recall can save millions of dollars annually.

This project explores how statistical modeling and modern machine learning can be combined

to identify fraud more effectively while keeping false alarms low.