# **🧾 Final Project Report: Financial Fraud Detection Using Machine Learning**

## **🔍 Project Overview**

This project focuses on detecting fraudulent transactions in a financial dataset with over **6.3 million rows** and **10 columns**. The primary goal is to build robust machine-learning models that can distinguish fraudulent transactions from genuine ones. We approached this task with meticulous steps covering data exploration, cleaning, feature engineering, model training, evaluation, and interpretability.

## **📂 Dataset Description**

The dataset included transactional attributes such as:

* step (hour of transaction),
* amount,
* type (e.g., PAYMENT, CASH\_OUT),
* account balances (oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest),
* and binary labels like isFraud and isFlaggedFraud.

## **✅ Data Cleaning & Preprocessing**

### **Steps Completed:**

* **Checked data types:** All variables were in expected formats.
* **Missing values:** Verified using df.isnull().sum() – no missing values were found.
* **Removed irrelevant columns:** Dropped nameOrig and nameDest as they were anonymized and had no predictive value.
* **Outlier detection:** Detected using IQR; however, we **retained outliers** since they might indicate fraud. This was clearly documented:  
   *“Outliers were retained for analysis as they may indicate fraudulent transactions.”*
* **Log transformation:** Applied on amount and other skewed columns to reduce the effect of extreme values and improve interpretability.  
   *“Most extreme values in the amount column were non-fraudulent and may represent noise or unusual but genuine activity. Log transformation was applied to reduce skewness while retaining meaningful fraud data.”*

## **📊 Exploratory Data Analysis (EDA)**

### **Insights:**

* **Class imbalance:** isFraud had significantly fewer 1s compared to 0s → **dataset is imbalanced.**
* **Transaction types:** CASH\_OUT and PAYMENT dominated the dataset; DEBIT was rare.
* **New features:** Created:  
  + balanceDiffOrig = oldbalanceOrg - newbalanceOrig
  + balanceDiffDest = newbalanceDest - oldbalanceDest
  + amountToBalanceRatio = amount / oldbalanceOrg
* **Correlation analysis:** Only balanceDiffOrig had moderate correlation with fraud (≈ 0.36), others were weak → simple correlation insufficient to detect fraud.
* **Visualizations:** Included:  
  + Histograms, bar plots, box plots
  + Violin plots comparing fraud vs. non-fraud
  + Class distributions
  + Correlation heatmaps

## **🏗️ Feature Engineering**

* Created **domain-relevant features** as mentioned above.
* **Log transformed** skewed numeric features.
* **No categorical encoding** needed as type was the only categorical column and it was already numeric.
* **Normalization** was handled implicitly through log transformation.

## **📉 Feature Selection**

* Used **correlation matrix** and **model-based importance scores** to retain top features.
* Avoided highly correlated or redundant features.
* Decision trees and XGBoost inherently performed feature selection.

## **🤖 Models Implemented**

### **1. Logistic Regression**

* Baseline model
* Accuracy: ~96%
* Strength: Interpretable
* Weakness: Not enough for highly imbalanced data

### **2. Decision Tree Classifier**

* Achieved **100% accuracy**, but likely **overfitted**
* Noted in report:  
   *“Decision Tree achieved perfect accuracy on the balanced dataset, which may indicate overfitting.”*

### **3. XGBoost Classifier**

* Also reached 100% accuracy (same concern about overfitting)
* More robust than a simple tree

### **4. MLPClassifier (Neural Network)**

* Accuracy: ~95%
* Took longer to run
* Strong precision and recall

### **5. Isolation Forest**

* Unsupervised anomaly detection
* Used as an exploratory tool to identify potential frauds

## **📚 Model Training Strategy**

* **Train-test split:** Used a typical 80-20 strategy.
* **No cross-validation** was done (due to size & time constraints).
* **No hyperparameter tuning** (e.g., GridSearchCV/RandomizedSearch) was applied due to resource/time limits.
* **Bagging and boosting:**
  + Boosting: XGBoost was used.
  + Bagging: Not used explicitly (e.g., Random Forest was not run).

## **🧪 Model Evaluation**

* **Evaluation metrics used:**
  + Accuracy
  + Precision, Recall, F1-score
  + Confusion matrix
  + ROC-AUC curve and Precision-Recall curve
* High-performing models showed excellent scores but lacked external validation to confirm generalizability.

## **⚖️ Constraints & Decisions**

* **Overfitting Risk:** Noted with Decision Tree and XGBoost; no pruning or regularization applied.
* **Cross-validation & tuning:** Skipped due to dataset size and time.
* **Dropped plan to use:** Random Forest and complex hyperparameter tuning (due to time & compute limits).

## **🧠 Model Explainability**

* **SHAP or LIME was not applied** (due to time/resource constraints), but was considered for future interpretability.

## **📈 Key Insights & Patterns**

* Fraudulent transactions often had:  
  + Large withdrawal amounts (especially in CASH\_OUT)
  + Discrepancies in account balances
  + Unique balance ratios

## **🛡️ Business Recommendations**

* Use real-time anomaly detection systems.
* Monitor high-amount and high-frequency transactions.
* Apply AI-based dynamic thresholding.
* Focus fraud detection especially on CASH\_OUT and TRANSFER types.

## **🔄 Future Work**

* Apply cross-validation to validate models properly.
* Add SHAP/LIME for explainability.
* Explore deep learning (RNNs) or streaming models.
* Build a real-time dashboard or deployment pipeline (e.g., using Flask or Streamlit).
* Consider concept drift detection techniques.

## **🧰 Tools & Libraries Used**

* **Language:** Python
* **Libraries:** pandas, numpy, seaborn, matplotlib, scikit-learn, xgboost
* **IDE:** Visual Studio Code
* **Notebook Conversion:** .ipynb to .html via nbconvert

## **📌 Conclusion**

This project successfully demonstrated how a combination of careful EDA, thoughtful feature engineering, and multiple machine learning algorithms can help detect fraudulent transactions. Despite not including cross-validation and tuning, our models achieved high performance. We also ensured that outliers were interpreted correctly and new insights were drawn from engineered features.

While some aspects like SHAP analysis, Random Forest, and hyperparameter tuning were skipped due to time/resource constraints, the core objective—building and evaluating a fraud detection model pipeline—was completed successfully.