# AI-Driven Fraud Detection Model

A comprehensive machine learning project for detecting fraudulent transactions using popular datasets from Kaggle.

## 🎯 Project Overview

This project implements a complete fraud detection pipeline including: - **Data Exploration**: Comprehensive analysis of transaction patterns - **Feature Engineering**: Advanced feature creation and selection - **Model Training**: Multiple ML algorithms with hyperparameter optimization - **Model Evaluation**: Detailed performance analysis and comparison - **Deployment Ready**: Production-ready model saving and loading

## 📊 Supported Datasets

The project is designed to work with popular fraud detection datasets:

1. **Credit Card Fraud Detection** (Recommended)
   * Source: [Kaggle Dataset](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)
   * Features: 30 features (28 anonymized + Amount + Time)
   * Fraud Rate: ~0.17%
   * Size: ~284K transactions
2. **IEEE-CIS Fraud Detection**
   * Source: [Kaggle Competition](https://www.kaggle.com/c/ieee-fraud-detection/data)
   * Features: 400+ features
   * Fraud Rate: ~3.5%
3. **Synthetic Financial Dataset (PaySim)**
   * Source: [Kaggle Dataset](https://www.kaggle.com/datasets/ealaxi/paysim1)
   * Features: 11 features
   * Fraud Rate: ~0.6%

## 🚀 Quick Start

### 1. Setup Environment

# Clone the repository  
git clone <repository-url>  
cd fraud\_modelling\_project  
  
# Install dependencies  
pip install -r requirements.txt

### 2. Download Dataset

# Option 1: Use the data downloader  
python src/data\_downloader.py  
  
# Option 2: Manual download  
# Download from Kaggle and place in data/raw/creditcard.csv

### 3. Run Data Exploration

python data\_exploration.py

### 4. Train Models

python train\_model.py

### 5. Interactive Analysis (Recommended)

# Start Jupyter notebook  
jupyter notebook notebooks/fraud\_detection\_workflow.ipynb

## 📁 Project Structure

fraud\_modelling\_project/  
├── data/  
│ ├── raw/ # Original datasets  
│ └── processed/ # Processed datasets  
├── src/  
│ ├── data\_downloader.py # Dataset download utilities  
│ └── feature\_engineering.py # Advanced feature creation  
├── notebooks/  
│ └── fraud\_detection\_workflow.ipynb # Complete workflow  
├── models/ # Trained models  
├── data\_exploration.py # Data analysis script  
├── train\_model.py # Model training script  
├── requirements.txt # Python dependencies  
└── README.md # This file

## 🔧 Features

### Data Exploration

* Automatic fraud column identification
* Comprehensive statistical analysis
* Visualization of fraud patterns
* Correlation analysis
* Time series analysis (if applicable)

### Feature Engineering

* **Time-based features**: Hour, day of week, business hours
* **Amount features**: Log, sqrt, squared, high-value flags
* **Statistical features**: Rolling statistics, z-scores, percentiles
* **Interaction features**: Feature combinations
* **Anomaly features**: Outlier detection, Mahalanobis distance
* **Dimensionality reduction**: PCA components

### Model Training

* **Multiple algorithms**: Logistic Regression, Random Forest, XGBoost, LightGBM
* **Class imbalance handling**: SMOTE, ADASYN, undersampling
* **Feature selection**: Statistical feature selection
* **Hyperparameter optimization**: Optuna-based optimization
* **Cross-validation**: Stratified k-fold validation

### Model Evaluation

* **Performance metrics**: AUC-ROC, Precision, Recall, F1-Score
* **Visualizations**: ROC curves, confusion matrices
* **Model comparison**: Side-by-side performance analysis

## 📈 Model Performance

Typical performance on Credit Card Fraud dataset: - **AUC-ROC**: 0.95+ (XGBoost/LightGBM) - **Precision**: 0.80+ (at 0.90 recall) - **Recall**: 0.90+ (for fraud detection)

## 🛠️ Usage Examples

### Basic Usage

from data\_exploration import FraudDataExplorer  
from train\_model import FraudModelTrainer  
  
# Data exploration  
explorer = FraudDataExplorer()  
df = explorer.load\_data("data/raw/creditcard.csv")  
explorer.generate\_report()  
  
# Model training  
trainer = FraudModelTrainer()  
trainer.load\_data("data/raw/creditcard.csv")  
results = trainer.train\_models()  
trainer.evaluate\_models(results)

### Advanced Feature Engineering

from src.feature\_engineering import FraudFeatureEngineer  
  
engineer = FraudFeatureEngineer()  
df\_enhanced = engineer.engineer\_all\_features(df, fraud\_column='Class')  
top\_features = engineer.select\_top\_features(df\_enhanced, 'Class', n\_features=50)

### Model Prediction

# Load trained model  
trainer.load\_model('XGBoost', 'models/xgboost\_model.pkl')  
  
# Make predictions on new data  
predictions, probabilities = trainer.predict\_new\_data('XGBoost', new\_data)

## 🔍 Data Exploration Features

The data exploration module provides:

1. **Automatic Analysis**:
   * Dataset shape and memory usage
   * Data types and missing values
   * Fraud distribution analysis
2. **Visualizations**:
   * Transaction distribution plots
   * Feature correlation heatmaps
   * Time series analysis (if applicable)
3. **Statistical Insights**:
   * Feature importance ranking
   * Correlation with fraud
   * Distribution comparisons

## 🎛️ Model Training Features

The training module includes:

1. **Preprocessing**:
   * Automatic fraud column detection
   * Feature scaling and selection
   * Missing value handling
2. **Class Imbalance**:
   * SMOTE oversampling
   * ADASYN adaptive sampling
   * Random undersampling
3. **Model Selection**:
   * Multiple algorithm comparison
   * Hyperparameter optimization
   * Cross-validation

## 📊 Model Evaluation

Comprehensive evaluation including:

1. **Performance Metrics**:
   * AUC-ROC score
   * Precision, Recall, F1-Score
   * Confusion matrix
2. **Visualizations**:
   * ROC curves comparison
   * Precision-Recall curves
   * Feature importance plots
3. **Model Comparison**:
   * Side-by-side performance
   * Statistical significance testing
   * Best model selection

## 🚀 Deployment

### Model Saving

# Save best model  
best\_model = max(results.keys(), key=lambda x: results[x]['auc'])  
trainer.save\_model(best\_model, f"models/{best\_model.lower().replace(' ', '\_')}\_model.pkl")

### Model Loading

# Load saved model  
trainer.load\_model('XGBoost', 'models/xgboost\_model.pkl')

### Production Prediction

# Make predictions on new transactions  
predictions, probabilities = trainer.predict\_new\_data('XGBoost', new\_transactions)

## 🔧 Configuration

### Environment Variables

* Set KAGGLE\_USERNAME and KAGGLE\_KEY for automatic dataset download
* Configure model parameters in train\_model.py

### Model Parameters

* Adjust hyperparameter optimization trials
* Modify feature selection criteria
* Change class imbalance handling methods

## 📚 Dependencies

Key packages used: - **Data Processing**: pandas, numpy - **Machine Learning**: scikit-learn, xgboost, lightgbm - **Visualization**: matplotlib, seaborn, plotly - **Optimization**: optuna - **Imbalanced Learning**: imbalanced-learn

## 🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Add tests if applicable
5. Submit a pull request

## 📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

## 🙏 Acknowledgments

* Kaggle for providing the datasets
* The open-source ML community for the excellent libraries
* Contributors and maintainers of the used packages

## 📞 Support

For questions or issues: 1. Check the documentation 2. Review the example notebooks 3. Open an issue on GitHub

**Happy Fraud Detection! 🕵️‍♂️**

## Bank Fraud Detection System: Production Checklist

### Data & Feature Engineering

* Consistent feature engineering for both training and inference
* Handle missing values robustly
* Validate input data schema before processing
* Use advanced feature engineering (see feature\_engineering.py)

### Model Training & Evaluation

* Train multiple models and compare performance (AUC, F1, etc.)
* Use stratified splits for imbalanced data
* Save model, scaler, imputer, and feature list for reproducibility
* Evaluate on realistic, small, and noisy datasets

### Real-Time Scoring

* Apply the same preprocessing pipeline to new transactions
* Add error handling for unexpected input
* Log predictions and errors for monitoring

### Customer Profiling & Risk

* Aggregate customer behavior and risk scores
* Update profiles regularly with new data
* Use risk thresholds for actionable alerts

### Deployment & Monitoring

* Use Python logging instead of print statements
* Add unit and integration tests for all modules
* Monitor model drift and retrain as needed
* Document all code and data flows

### Security & Compliance

* Secure sensitive data (PII, account info)
* Audit access to models and data
* Ensure compliance with relevant regulations (e.g., GDPR)

## Planned Enhancement: LangGraph AI Agent Integration

* Add [LangGraph](https://github.com/langchain-ai/langgraph) framework to requirements
* Design an AI agent workflow for transaction enrichment, anomaly explanation, and human-in-the-loop review
* Integrate LangGraph agent with the fraud detection pipeline for:
  + Automated enrichment of suspicious transactions
  + Contextual explanations for alerts
  + Escalation to human analysts when needed
* Add tests and documentation for the LangGraph agent

*This checklist will help ensure your bank fraud detection system is robust, production-ready, and enhanced with next-generation AI agent capabilities.*