# Kaggle Dataset Guide for Fraud Detection

This guide will help you download and use real fraud detection datasets from Kaggle to test the fraud modeling code.

## 🎯 Available Datasets

### 1. Credit Card Fraud Detection (Recommended)

* **URL**: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
* **Size**: ~150 MB
* **Features**: 30 (28 anonymized + Amount + Time)
* **Fraud Rate**: ~0.17%
* **Transactions**: ~284K
* **Best for**: Credit card transaction fraud

### 2. IEEE-CIS Fraud Detection

* **URL**: https://www.kaggle.com/c/ieee-fraud-detection/data
* **Size**: ~1.5 GB
* **Features**: 400+
* **Fraud Rate**: ~3.5%
* **Best for**: Large-scale fraud detection

### 3. Synthetic Financial Dataset (PaySim)

* **URL**: https://www.kaggle.com/datasets/ealaxi/paysim1
* **Size**: ~200 MB
* **Features**: 11
* **Fraud Rate**: ~0.6%
* **Best for**: Mobile money fraud

## 📥 Download Methods

### Method 1: Kaggle API (Recommended)

1. **Create Kaggle Account**
   * Go to https://www.kaggle.com
   * Sign up for a free account
2. **Get API Credentials**
   * Go to https://www.kaggle.com/settings/account
   * Scroll to “API” section
   * Click “Create New API Token”
   * Download kaggle.json
3. **Setup API Credentials**

* # Create kaggle directory  
  mkdir -p ~/.kaggle  
    
  # Copy kaggle.json to the directory  
  cp path/to/downloaded/kaggle.json ~/.kaggle/  
    
  # Set correct permissions  
  chmod 600 ~/.kaggle/kaggle.json

1. **Download Dataset**

* # Download Credit Card Fraud dataset  
  kaggle datasets download mlg-ulb/creditcardfraud -p data/raw --unzip  
    
  # Or use the provided script  
  python3 download\_kaggle\_dataset.py

### Method 2: Manual Download

1. **Visit Dataset Page**
   * Go to https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
   * Click “Download” button
2. **Extract and Place**

* # Extract the downloaded zip file  
  unzip creditcardfraud.zip  
    
  # Move to project directory  
  mv creditcard.csv data/raw/

1. **Verify Download**

* # Check file exists  
  ls -la data/raw/creditcard.csv  
    
  # Verify with Python  
  python3 -c "import pandas as pd; df=pd.read\_csv('data/raw/creditcard.csv'); print(f'Shape: {df.shape}, Fraud rate: {df.Class.mean():.4f}')"

## 🧪 Testing with Real Data

### Quick Test

# Run the comprehensive test  
python3 test\_fraud\_modeling.py

### Full Pipeline

# 1. Data exploration  
python3 data\_exploration.py  
  
# 2. Model training  
python3 train\_model.py  
  
# 3. Interactive analysis  
jupyter notebook notebooks/fraud\_detection\_workflow.ipynb

## 📊 Expected Results with Real Data

### Credit Card Fraud Dataset

* **Dataset Size**: ~284K transactions
* **Fraud Rate**: ~0.17% (492 fraudulent out of 284,807 total)
* **Expected AUC**: 0.95+ (with proper feature engineering)
* **Training Time**: 2-5 minutes (depending on hardware)

### Performance Comparison

| Model | AUC Score | Precision | Recall | F1-Score |
| --- | --- | --- | --- | --- |
| Logistic Regression | 0.85-0.90 | 0.70-0.80 | 0.80-0.90 | 0.75-0.85 |
| Random Forest | 0.90-0.95 | 0.80-0.90 | 0.85-0.95 | 0.85-0.90 |
| XGBoost | 0.95-0.98 | 0.85-0.95 | 0.90-0.98 | 0.90-0.95 |
| LightGBM | 0.95-0.98 | 0.85-0.95 | 0.90-0.98 | 0.90-0.95 |

## 🔧 Customization for Different Datasets

### For IEEE-CIS Dataset

# Update data paths  
data\_path = "data/raw/train\_transaction.csv"  
fraud\_column = "isFraud"  
  
# Handle categorical features  
categorical\_features = ['ProductCD', 'card4', 'card6', 'P\_emaildomain']

### For PaySim Dataset

# Update data paths  
data\_path = "data/raw/PS\_20174392719\_1491204439457\_log.csv"  
fraud\_column = "isFraud"  
  
# Handle different feature names  
amount\_column = "amount"

## 🚀 Advanced Usage

### 1. Feature Engineering

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

### 2. Hyperparameter Optimization

from train\_model import FraudModelTrainer  
  
trainer = FraudModelTrainer()  
best\_params = trainer.hyperparameter\_optimization('XGBoost', n\_trials=100)

### 3. Model Deployment

# Save best model  
trainer.save\_model('XGBoost', 'models/best\_fraud\_model.pkl')  
  
# Load for predictions  
trainer.load\_model('XGBoost', 'models/best\_fraud\_model.pkl')  
predictions, probabilities = trainer.predict\_new\_data('XGBoost', new\_transactions)

## 📈 Performance Monitoring

### Key Metrics to Track

* **AUC-ROC**: Overall model performance
* **Precision**: Accuracy of fraud predictions
* **Recall**: Ability to catch all fraud
* **F1-Score**: Balanced measure
* **False Positive Rate**: Cost of false alarms

### Visualization Files Generated

* feature\_correlations.png: Feature importance
* roc\_curves.png: Model comparison
* confusion\_matrices.png: Prediction accuracy
* feature\_importance.png: Random Forest insights

## 🛠️ Troubleshooting

### Common Issues

1. **Kaggle API Errors**

* # Check credentials  
  ls -la ~/.kaggle/kaggle.json  
    
  # Test API  
  kaggle datasets list --limit 1

1. **Memory Issues**

* # Use smaller sample for testing  
  df\_sample = df.sample(n=10000, random\_state=42)

1. **Package Installation**

* # Install missing packages  
  pip3 install -r requirements.txt  
    
  # For XGBoost on Mac  
  brew install libomp

### Performance Tips

1. **Use Sample Data for Development**

* # Use 10% of data for quick testing  
  df\_dev = df.sample(frac=0.1, random\_state=42)

1. **Optimize Memory Usage**

* # Use appropriate dtypes  
  df['Amount'] = df['Amount'].astype('float32')

1. **Parallel Processing**

* # Use all CPU cores  
  model = RandomForestClassifier(n\_jobs=-1)

## 📚 Additional Resources

* **Kaggle Notebooks**: Search for “credit card fraud” on Kaggle
* **Research Papers**: IEEE-CIS fraud detection competition papers
* **Community**: Kaggle forums and discussions

## 🎉 Success Indicators

You’ll know everything is working when you see: - ✅ Dataset loaded successfully - ✅ AUC scores > 0.90 - ✅ Generated visualization files - ✅ Model saved to models/ directory - ✅ No errors in the pipeline

**Happy Fraud Detection! 🕵️‍♂️**

For questions or issues, check the main README.md or open an issue in the repository.