# AI-Driven Fraud Detection System - Complete Documentation

## 🎯 Project Overview

This comprehensive fraud detection system combines machine learning algorithms with a modern web dashboard for real-time fraud monitoring. The system is designed for bank-level constraints with small datasets and domain-specific features.

## 📊 System Architecture

### Core Components

1. **Bank Fraud Detector** (src/bank\_fraud\_detector.py)
   * Main fraud detection engine
   * Multiple ML algorithms (Logistic Regression, Random Forest, Isolation Forest)
   * Real-time transaction scoring
   * Customer risk profiling
2. **Streamlit Dashboard** (src/dashboard.py)
   * Real-time monitoring interface
   * Transaction analysis and visualization
   * Alert system with color-coded notifications
   * Performance metrics display
3. **FastAPI Server** (src/api\_server.py)
   * RESTful API endpoints
   * Real-time prediction API
   * Model management endpoints
   * Health monitoring
4. **Multi-Agent Pipeline** (src/multi\_agent\_pipeline.py)
   * Advanced fraud detection using multiple AI agents
   * LangGraph-based workflow
   * Enriched transaction analysis

## 🚀 Quick Start Guide

### 1. Local Development Setup

# Install dependencies  
pip install -r requirements.txt  
  
# Run the dashboard  
python3 -m streamlit run src/dashboard.py --server.port 8501  
  
# Run the API server (optional)  
python3 -m uvicorn src.api\_server:app --host 0.0.0.0 --port 8000 --reload

### 2. Dashboard Access

* **Local URL**: http://localhost:8501
* **Network URL**: http://192.168.68.106:8501 (for sharing on local network)

### 3. API Endpoints

* **Health Check**: http://localhost:8000/health
* **Predict Transaction**: http://localhost:8000/predict
* **Model Info**: http://localhost:8000/model-info

## 📈 Dashboard Features

### Real-Time Monitoring

* **Transaction Feed**: Live transaction monitoring
* **Risk Scoring**: Real-time fraud probability calculation
* **Alert System**: Color-coded alerts (High/Medium/Low Risk)
* **Performance Metrics**: Model accuracy and performance indicators

### Analytics & Visualization

* **Transaction Distribution**: Amount and time-based analysis
* **Fraud Patterns**: Historical fraud trend analysis
* **Customer Profiles**: Risk categorization and behavior patterns
* **Model Performance**: ROC curves, confusion matrices, feature importance

### Alert System

* **High Risk Alerts**: Red cards for immediate attention
* **Medium Risk Alerts**: Orange cards for monitoring
* **Low Risk Alerts**: Yellow cards for awareness
* **Log Display**: Detailed transaction logs with timestamps

## 🔧 Technical Implementation

### Bank Fraud Detector Class

class BankFraudDetector:  
 def \_\_init\_\_(self):  
 self.models = {}  
 self.scaler = None  
 self.imputer = None  
 self.feature\_columns = None  
 self.customer\_profiles = {}  
 self.risk\_thresholds = {}

**Key Methods:** - load\_bank\_data(): Load and validate transaction data - engineer\_bank\_features(): Create domain-specific features - train\_bank\_models(): Train multiple ML algorithms - predict\_transaction\_risk(): Real-time risk scoring - create\_customer\_profiles(): Customer risk categorization

### Feature Engineering

**Time-based Features:** - Weekend transactions - Night-time transactions - Business hours analysis

**Amount-based Features:** - Log-transformed amounts - High-value transaction flags - Amount percentiles

**Transaction Features:** - Online vs ATM transactions - International transaction flags - Card-present vs card-not-present

**Customer Behavior Features:** - Average transaction amounts - Transaction frequency - Historical fraud rates - Risk scores

### Model Training

**Supported Algorithms:** 1. **Logistic Regression**: Baseline model with interpretability 2. **Random Forest**: Robust ensemble method 3. **Isolation Forest**: Anomaly detection approach

**Training Process:** - Feature scaling with RobustScaler - Missing value imputation - Stratified train-test split - Cross-validation - Hyperparameter optimization

## 📊 Performance Metrics

### Model Evaluation

* **AUC-ROC Score**: Area under the ROC curve
* **Precision**: Accuracy of positive predictions
* **Recall**: Sensitivity to fraud detection
* **F1-Score**: Harmonic mean of precision and recall

### Risk Thresholds

* **High Risk**: Top 5% of risk scores
* **Medium Risk**: Top 15% of risk scores
* **Low Risk**: Top 30% of risk scores
* **Safe**: Below 30th percentile

## 🌐 Deployment Options

### 1. Local Development

# Start dashboard only  
python3 -m streamlit run src/dashboard.py --server.port 8501  
  
# Start complete system  
python3 start\_dashboard.py

### 2. Network Sharing

# Make dashboard accessible on local network  
python3 -m streamlit run src/dashboard.py --server.port 8501 --server.address 0.0.0.0

### 3. Streamlit Cloud Deployment

1. Push code to GitHub (public repository)
2. Connect to Streamlit Cloud
3. Deploy with main file: src/dashboard.py
4. Get public URL for sharing

### 4. Docker Deployment

# Build and run with Docker  
docker build -t fraud-detection-dashboard .  
docker run -p 8501:8501 fraud-detection-dashboard

## 🔍 API Documentation

### Endpoints

#### Health Check

GET /health

Returns system health status and model information.

#### Predict Transaction

POST /predict  
Content-Type: application/json  
  
{  
 "transaction\_id": "12345",  
 "amount": 150.00,  
 "customer\_id": "CUST001",  
 "transaction\_type": "ONLINE",  
 "hour": 14,  
 "day\_of\_week": 2  
}

#### Model Information

GET /model-info

Returns detailed model performance metrics and configuration.

## 📁 Project Structure

fraud\_modelling\_project/  
├── src/  
│ ├── bank\_fraud\_detector.py # Main fraud detection engine  
│ ├── dashboard.py # Streamlit dashboard  
│ ├── api\_server.py # FastAPI server  
│ ├── multi\_agent\_pipeline.py # Advanced AI pipeline  
│ ├── feature\_engineering.py # Feature creation utilities  
│ └── data\_downloader.py # Dataset management  
├── models/  
│ └── bank\_fraud\_detector.pkl # Trained model  
├── data/  
│ ├── raw/ # Original datasets  
│ └── processed/ # Processed datasets  
├── notebooks/  
│ └── fraud\_detection\_workflow.ipynb  
├── requirements.txt # Python dependencies  
├── start\_dashboard.py # System startup script  
└── README.md # Main documentation

## 🛠️ Customization

### Adding New Features

1. Extend engineer\_bank\_features() method
2. Update feature selection in train\_bank\_models()
3. Modify dashboard visualizations

### Adding New Models

1. Import new algorithm in train\_bank\_models()
2. Add to models dictionary
3. Update prediction logic in predict\_transaction\_risk()

### Customizing Risk Thresholds

# Modify in set\_risk\_thresholds() method  
thresholds = {  
 'high\_risk': np.percentile(probabilities, 95),  
 'medium\_risk': np.percentile(probabilities, 85),  
 'low\_risk': np.percentile(probabilities, 70)  
}

## 🔒 Security Considerations

### Data Privacy

* No sensitive customer data stored
* Transaction IDs are anonymized
* Model doesn’t require PII

### Access Control

* Local deployment for sensitive environments
* Network-level security for shared access
* API authentication for production use

### Model Security

* Model files are serialized securely
* Input validation on all endpoints
* Error handling prevents information leakage

## 📈 Future Enhancements

### Planned Features

1. **Real-time Data Integration**: Connect to live transaction feeds
2. **Advanced Analytics**: Deep learning models and neural networks
3. **Multi-language Support**: Internationalization for global deployment
4. **Mobile Dashboard**: Responsive design for mobile devices
5. **Advanced Reporting**: Automated report generation

### Scalability Improvements

1. **Database Integration**: PostgreSQL/MongoDB for data persistence
2. **Message Queues**: Redis/RabbitMQ for high-throughput processing
3. **Microservices**: Containerized deployment with Kubernetes
4. **Load Balancing**: Multiple dashboard instances

## 🆘 Troubleshooting

### Common Issues

#### Dashboard Not Loading

# Check if port is available  
lsof -i :8501  
  
# Try different port  
python3 -m streamlit run src/dashboard.py --server.port 8502

#### Model Loading Errors

# Rebuild model  
python3 -c "from src.bank\_fraud\_detector import BankFraudDetector; detector = BankFraudDetector(); detector.save\_bank\_model('models/bank\_fraud\_detector.pkl')"

#### Missing Dependencies

# Install requirements  
pip install -r requirements.txt  
  
# For specific packages  
pip install streamlit fastapi uvicorn

### Performance Optimization

1. **Reduce Data Size**: Use smaller sample datasets for testing
2. **Optimize Features**: Select only essential features
3. **Caching**: Enable Streamlit caching for repeated computations
4. **Background Processing**: Use async processing for heavy computations

## 📞 Support

For technical support or questions: 1. Check the troubleshooting section above 2. Review the code documentation in source files 3. Examine the Jupyter notebook for detailed examples 4. Check GitHub issues for known problems

**Last Updated**: December 2024 **Version**: 1.0.0 **Author**: AI Assistant **License**: MIT