**Fraud Detection System**

This project implements a **Fraud Detection System** using a pre-trained **LSTM model** for anomaly detection in financial transactions. The backend is built with **FastAPI** for handling predictions and serving a web-based UI for visualization.

**Features**

* Pre-trained LSTM model for fraud detection (created in Kaggle).
* REST API for predicting fraud status of transactions.
* Interactive charts and visualization for fraud analysis.
* UI integration using Jinja2 templates.

**Project Structure**

.

├── main.py # Main FastAPI application

├── model # Directory for ML models

│ └── Fraud\_detection\_model\_.h5

├── templates # Jinja2 HTML templates

│ ├── index.html

│ ├── chart.html

│ └── about.html

├── financial\_anomaly\_data.csv # Dataset used for analysis

├── README.md # Project documentation

└── requirements.txt # Python dependencies

**Requirements**

Ensure you have the following installed:

* Python 3.8+
* Libraries (Install using requirements.txt):
  + FastAPI
  + Uvicorn
  + TensorFlow
  + NumPy
  + Pandas
  + Scikit-learn
  + Jinja2

Install dependencies:

pip install -r requirements.txt

**How to Train the Model**

The LSTM model was trained in Kaggle using a financial transactions dataset. The steps include:

1. **Preprocess the data**:
   * Handle missing values and normalize features.
   * Create sequences for LSTM input.
2. **Train the LSTM model**:
   * Use an LSTM network to learn reconstruction errors from the data.
3. **Export the model**:
   * Save the trained model as Fraud\_detection\_model\_.h5 and place it in the model directory.

**How to Run thee Application Locally**

**Step 1: Verify files**

Ensure the following files are present:

* main.py
* model/Fraud\_detection\_model\_.h5
* financial\_anomaly\_data.csv
* templates/ directory with the required HTML files.

**Step 2: Install dependencies**

pip install -r requirements.txt

**Step 3: Run the application**

uvicorn main:app --reload

**Step 4: Access the application**

* Open your browser and navigate to: <http://127.0.0.1:8000>
* API documentation available at: http://127.0.0.1:8000/docs

**Step 5: Test the UI**

* **Main Dashboard**: Navigate to the root URL to see the transaction dashboard.
* **Chart Page**: Navigate to /chart to view fraud analysis visualizations.
* **About Page**: Navigate to /about for additional project information.

**Step 6: Test the API**

**Predict Endpoint**

curl -X POST "http://127.0.0.1:8000/predict/" \

-H "Content-Type: application/json" \

-d '{"TransactionID": "<Transaction ID>"}'

Response:

{

"TransactionID": "<Transaction ID>",

"Fraud": true/false

}

**API Endpoints**

**/**

* **Method**: GET
* **Description**: Displays the main dashboard.

**/predict/**

* **Method**: POST
* **Description**: Predicts whether a transaction is fraudulent.
* **Request Body**:

{

"TransactionID": "<Transaction ID>"

}

* **Response**:

{

"TransactionID": "<Transaction ID>",

"Fraud": true/false

}

**/chart**

* **Method**: GET
* **Description**: Displays fraud analysis charts.

**/about**

* **Method**: GET
* **Description**: Displays the "About Us" page.

**Dataset and Preprocessing**

* The dataset used is financial\_anomaly\_data.csv.
* Preprocessing steps include:
  + Filling missing values.
  + Normalizing the Amount column.
  + Creating sequences of 15 data points for LSTM input.

**Additional Notes**

1. **Error Handling**:
   * The application includes robust error handling for missing files, empty datasets, and invalid inputs.
2. **Threshold for Fraud Detection**:
   * Fraud is determined using a threshold based on reconstruction error percentile (95th percentile).
3. **Scalability**:
   * For large datasets, consider chunked processing or streaming.

**License**

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

**Acknowledgements**

* Model training: Kaggle
* Framework: FastAPI