Configuration Manual

MSc Research Project
Data Analytics

Anurag Jaipal Rathore
Student ID: x23271302

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Anurag Jaipal Rathore

Student ID: X23271302

Programme: Data Analytics **Year:** 2025

Module: MSc Research Project

Lecturer: Abubakr Siddig

Submission Due Date: 11/08/2025

Project Title: Securing Online Payments: AI-based fraud detection for e-commerce platforms

Word Count: 1612 **Page Count:** 5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Anurag Jaipal Rathore

Date: 11/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Anurag Jaipal Rathore
X23271302

1.1 Introduction

Configuration manual for the AI-Based Fraud Detection System. This document provides all the necessary information to set up, configure, and run the fraud detection application for an e-commerce platform.

The system is designed to analyze financial transactions in real-time and classify them as either "Fraudulent" or "Not Fraudulent". It leverages a machine learning model trained on a large dataset of transactional data. The core of the system is a Flask web application that serves the model via a simple web interface, making it easy to integrate into existing e-commerce workflows.

This manual is intended for developers, system administrators, and data scientists responsible for deploying and maintaining the application.

2 System Architecture Overview

The system consists of three main components that work together:

1. **Flask Web Application (app.py):** A lightweight Python web server that provides a user interface and an API endpoint. It receives transaction data, processes it, and returns the fraud prediction.
2. **Preprocessing Pipeline (scaler.pkl, pca.pkl):**
   - **StandardScaler (scaler.pkl):** Normalizes the input features to have a mean of 0 and a standard deviation of 1. This is essential for the model's performance.
   - **PCA (pca.pkl):** Principal Component Analysis is used to reduce the dimensionality of the feature space, which helps in reducing noise and improving model efficiency.
3. **Machine Learning Model (best_xgboost_model.pkl):** The trained model that performs the fraud classification. Based on the training notebook, this is a highly optimized **Random Forest classifier** (despite the filename) that predicts the probability of a transaction being fraudulent.

3 Prerequisites

Before you begin, ensure you have the following installed on your system:

- Python 3.8 or higher
- pip (Python package installer)
- git (for cloning the repository)

4 Installation and Setup

Follow these steps to get the application running on your local machine.

4.1 File Structure

Your project directory should be organized as follows for the application to work correctly:

/fraud-detection-system/

```
├── app.py              # The main Flask application file
├── models/
│   ├── best_xgboost_model.pkl # The trained classification model
│   ├── scaler.pkl          # The saved StandardScaler object
│   └── pca.pkl             # The saved PCA object
├── templates/
│   └── index.html          # The HTML template for the user interface
├── dataset/
│   ├── dataset.csv         # Original raw dataset (for reference/retraining)
│   └── balanced_dataset.csv# Balanced dataset used for training
├── Untitled.ipynb          # Jupyter notebook with the full analysis and training workflow
└── requirements.txt        # File listing all Python dependencies
```

4.2 Environment Setup

It is highly recommended to use a virtual environment to manage project dependencies.

```
# 1. Clone your project repository (if it's in git)
# git clone <your-repo-url>
# cd fraud-detection-system

# 2. Create a Python virtual environment
python -m venv venv

# 3. Activate the virtual environment
# On Windows
venv\Scripts\activate

# On macOS/Linux
source venv/bin/activate
```

4.3 Installing Dependencies

Create a requirements.txt file in your root directory with the following content:

requirements.txt

```
Flask
numpy
scikit-learn
joblib
pandas
xgboost
imblearn-learn
seaborn
matplotlib
plotly
```

Now, install these dependencies using pip:

```
pip install -r requirements.txt
```

5 Configuration

The primary configuration is done within the app.py file.

5.1 Application Configuration

In app.py, locate the app.run() command at the bottom of the file:

codePython

```
if __name__ == '__main__':
    app.run(debug=True)
```

- **debug=True**: This is ideal for development as it provides detailed error pages and automatically reloads the server when you make code changes.
- **For Production**: You must change this. Set debug=False and use a production-grade WSGI server like Gunicorn or uWSGI to run the application.

5.2 Model and Preprocessor Configuration

The paths to the model files are hardcoded in app.py. If you retrain the model or change the filenames, you must update these lines:

```
# Load saved model, scaler, and PCA
model = joblib.load('models/best_xgboost_model.pkl')
scaler = joblib.load('models/scaler.pkl')
pca = joblib.load('models/pca.pkl')
```

The FEATURES list is critical. It defines the exact order and names of the input fields the model expects. **Do not change this list unless you retrain the model with a different set of features.**

```
# Feature columns expected by the model
FEATURES = [
    'step', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest',
    'isFlaggedFraud', 'type_CASH_IN', 'type_CASH_OUT', 'type_DEBIT', 'type_PAYMENT', 'type_TRANSFER',
    'balance_change_org', 'balance_change_dest', 'amount_to_oldbalanceOrg_ratio', 'amount_to_oldbalanceDest_ratio'
]
```

6 Running the Application

Once the setup is complete, you can start the Flask server with one command from your terminal:

```
python app.py
```

You will see output similar to this, indicating the server is running:

```
* Serving Flask app 'app'
 * Debug mode: on
 * Running on http://127.0.0.1:5000
Press CTRL+C to exit
```

You can now access the application by navigating to http://127.0.0.1:5000 in your web browser.

7 Usage and API Endpoint

The application provides a web form at the root URL (/) for manual entry and testing. The endpoint handles both GET (to display the page) and POST (to submit data for prediction) requests.

7.1 Input Parameters

To get a prediction, you must provide values for all the features listed in the FEATURES array. The form fields correspond to these features. Note that the type_* features are one-hot encoded; for a single transaction, only one of them should be 1 and the rest 0.

Feature Name	Type	Description
step	float	Represents a unit of time in the real world (e.g., 1 step

		is 1 hour).
amount	float	The amount of the transaction.
oldbalanceOrg	float	Initial balance of the originator's account before the transaction.
newbalanceOrig	float	Final balance of the originator's account after the transaction.
oldbalanceDest	float	Initial balance of the recipient's account.
newbalanceDest	float	Final balance of the recipient's account.
isFlaggedFraud	float	1 if the system flagged the transaction, 0 otherwise.
type_CASH_IN	float	1 for CASH_IN type, 0 otherwise.
type_CASH_OUT	float	1 for CASH_OUT type, 0 otherwise.
type_DEBIT	float	1 for DEBIT type, 0 otherwise.
type_PAYMENT	float	1 for PAYMENT type, 0 otherwise.
type_TRANSFER	float	1 for TRANSFER type, 0 otherwise.
balance_change_org	float	Engineered feature: newbalanceOrig - oldbalanceOrg.
balance_change_dest	float	Engineered feature: newbalanceDest - oldbalanceDest.
amount_to_oldbalanceOrg_ratio	float	Engineered feature: amount / (oldbalanceOrg + 1).
amount_to_oldbalanceDest_ratio	float	Engineered feature: amount / (oldbalanceDest + 1).

7.2 Output

After submitting the form, the page will reload and display the prediction results:

- **Prediction:** Either "Fraudulent" or "Not Fraudulent".
- **Probability (proba):** A value between 0 and 1 representing the model's confidence that the transaction is fraudulent. A higher value indicates a higher likelihood of fraud.

8 Troubleshooting

- **FileNotFoundError: [Errno 2] No such file or directory: 'models/...'**: Ensure the models directory exists and contains all three .pkl files. Also, check that you are running python app.py from the root directory of the project.
- **ModuleNotFoundError: No module named 'flask' (or other package)**: Your virtual environment is not activated, or the dependencies were not installed correctly. Activate the environment and run pip install -r requirements.txt again.
- **Incorrect Predictions**: Verify that the input data sent to the model matches the format and order of the FEATURES list in app.py. Incorrect data types or missing values can lead to poor results.

9 Model Retraining (Advanced)

The Untitled.ipynb notebook contains the complete workflow for retraining the model. To create new model artifacts, follow these general steps within the notebook:

1. **Load Data**: Update the path to your new dataset if necessary.
2. **Preprocessing**: The notebook handles data cleaning, feature engineering, one-hot encoding, and data balancing (SMOTE).
3. **Scaling and PCA**: The StandardScaler and PCA objects are fitted on the new training data.
4. **Train-Test Split**: The data is split for training and validation.
5. **Model Training**: The notebook trains multiple models. The XGBoost model performed best.
6. **Save Artifacts**: The final step saves the best model, the scaler, and the PCA object. **Crucially, ensure the filenames you save match the ones being loaded in app.py**.

  - **Note**: The notebook saves the best Random Forest (best_rf) model with the filename best_xgboost_model.pkl. This is potentially confusing. For clarity, it is recommended to save it as best_random_forest_model.pkl and update app.py accordingly.

10 Conclusion

This AI-based Fraud Detection System provides a powerful and scalable solution for protecting e-commerce platforms from fraudulent transactions. By following this manual, you can successfully deploy, configure, and maintain the application. The system's architecture is modular, allowing for future updates, such as retraining the model with new data to adapt to evolving fraud patterns, ensuring long-term effectiveness.