National College of Ireland

Configuration Manual

MSc Research Project
Msc in Data Analytics

Om Devlekar
Student ID: X23214945

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Om Sandeep Devlekar
Student ID:	X23214945
Programme:	Msc in Data Analytics **Year:** 2024-25
Module:	Research Practicum
Lecturer:	Hicham Rifai
Submission Due Date:	11/08/202
Project Title:	AI-Powered Anomaly Detection for Fraudulent Credit Card Transactions
Word Count:	1514 **Page Count: 11**

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: Om Devlekar

Date: 10/08/2025

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

Om Devlekar
Student ID: X23214945

1 Introduction

1.1 Project Overview

The "AI-Powered Anomaly Detection for Fraudulent Credit Card Transactions" project is a comprehensive machine learning system designed to identify and flag potentially fraudulent transactions from a large dataset. The system leverages advanced data analysis techniques and a variety of AI models, including Isolation Forest, Autoencoders, and Long Short-Term Memory (LSTM) networks, to distinguish between legitimate and fraudulent activities. The ultimate goal is to create a robust, accurate, and deployable model that can be integrated into a real-world financial security pipeline.

1.2 Purpose of this Manual

This manual provides all the necessary information to configure, set up, and run the project. It details the system requirements, project structure, step-by-step execution instructions, and an overview of the underlying code and models. Following this guide will enable a user to replicate the analysis, retrain the models, and deploy the prediction web application.

1.3 Target Audience

This document is intended for data scientists, machine learning engineers, and developers who wish to understand, replicate, or extend the project. A basic understanding of Python, machine learning concepts, and command-line operations is assumed.

2 System Requirements

2.1 Hardware Requirements

- **CPU**: Modern multi-core processor (Quad-core or higher recommended).
- **RAM**: 16 GB or more (due to the large dataset size, which consumes over 2.5 GB in memory).
- **Disk Space**: At least 10 GB of free space for the dataset, Python environment, libraries, and saved models.

2.2 Software Requirements

- **Operating System**: Windows, macOS, or Linux.
- **Python**: Version 3.8 or newer.
- **Python Libraries**: A full list of required packages is provided in requirements.txt (see Appendix 8.1). Key libraries include:
    - pandas, numpy
    - scikit-learn
    - tensorflow
    - plotly, matplotlib, seaborn
    - imbalanced-learn
    - flask

- joblib

```python
# Data manipulation and analysis
import pandas as pd
import numpy as np

# Visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots

# Suppress warnings
import warnings
warnings.filterwarnings('ignore')

# Machine Learning Libraries
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
from sklearn.preprocessing import StandardScaler, LabelEncoder, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import IsolationForest
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    roc_auc_score,
    roc_curve,
    precision_score,
    recall_score,
    f1_score,
    accuracy_score
)
from sklearn.model_selection import GridSearchCV
```

Figure 1: Libraries Import

3 Project Structure

3.1 Directory Layout

To ensure the code runs correctly, the project files should be organized as follows:

```
fraud-detection-project/
│
├── dataset/
│   ├── train_transaction.csv
│   └── train_identity.csv
│
├── saved_models/
│   ├── scaler.pkl
│   ├── pca_model.pkl
│   ├── best_isolation_forest.pkl
│   ├── autoencoder_model.h5
│   ├── autoencoder_threshold.pkl
│   ├── lstm_model.h5
│   ├── feature_columns.pkl
│   └── ... (other saved files)
│
├── templates/
│   ├── index.html
│   └── about.html
│
├── credit_fraud.ipynb        # The main analysis and training notebook
├── app.py                    # The Flask web application script
└── requirements.txt          # List of Python dependencies
```

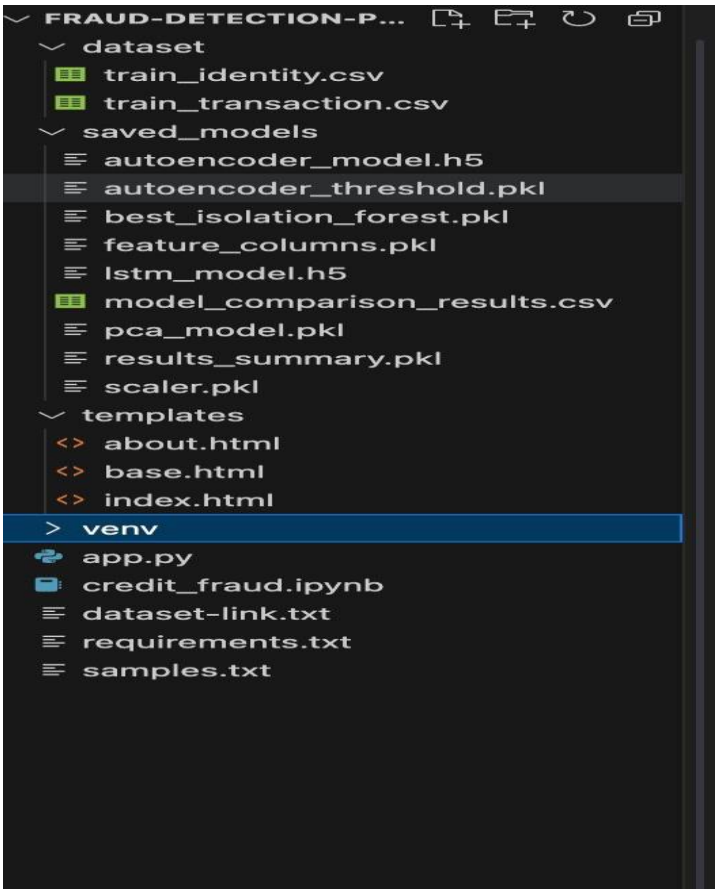

Figure 2: Directory Layout

4 Configuration and Setup

4.1 Step 1: Obtain Project Files

Download or clone all project files, including the Jupyter Notebook (Untitled.ipynb), the Flask application script (app.py), and the templates folder, into the root fraud-detection-project/ directory.

4.2 Step 2: Set Up Python Environment

It is highly recommended to use a virtual environment to avoid conflicts with other Python projects.

```
# Navigate to the project directory
cd fraud-detection-project

# Create a virtual environment
python -m venv .venv

# Activate the virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate
```

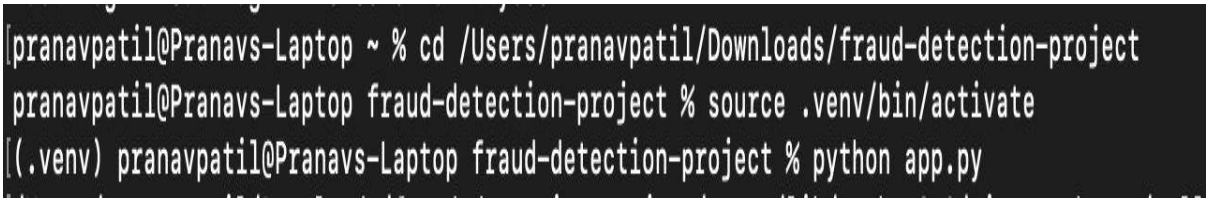

Figure 3: Activating Virtual Environment

4.3 Step 3: Install Dependencies

Install all required libraries using the requirements.txt file.
pip install -r requirements.txt

4.4 Step 4: Place the Dataset

Download the IEEE-CIS Fraud Detection dataset from Kaggle or another source. Place the two primary CSV files into the dataset/ directory:

- train_transaction.csv
- train_identity.csv

The application will not run without these files in the correct location.

5 Execution Guide

5.1 Part A: Running the Analysis and Model Training Notebook

The Untitled.ipynb notebook contains the end-to-end pipeline for data analysis, preprocessing, model training, and evaluation.

1. **Launch Jupyter:** Open a terminal, activate your virtual environment, and run:

jupyter lab
or
jupyter notebook

2. **Open the Notebook:** In the Jupyter interface, navigate to and open Untitled.ipynb.
3. **Run the Cells:** To execute the entire pipeline, click Kernel > Restart & Run All. This will:
   - Load and merge the datasets.

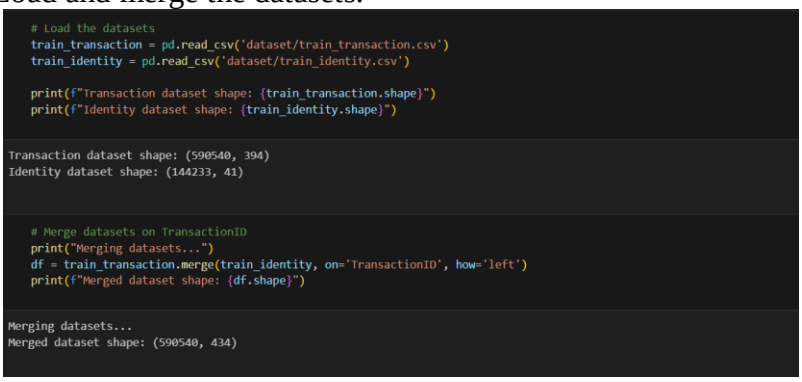

Figure 4: Loading and Merging of Data

   - Perform data cleaning and EDA.

Figure 5: EDA

   - Engineer new features.

```python
# ==============================
# FEATURE ENGINEERING
# ==============================

print("\n=== FEATURE ENGINEERING ===")

# Create new features
df_features = df.copy()

# Time-based features
if 'TransactionDT' in df_features.columns:
    df_features['TransactionHour'] = (df_features['TransactionDT'] / 3600) % 24
    df_features['TransactionDay'] = (df_features['TransactionDT'] / (3600 * 24)) % 7
    df_features['IsWeekend'] = df_features['TransactionDay'].isin([5, 6]).astype(int)
```

Figure 6: Feature Engineering

- Apply PCA for dimensionality reduction.

```python
# ================================
# DIMENSIONALITY REDUCTION (PCA)
# ================================

print("\n=== DIMENSIONALITY REDUCTION WITH PCA ===")

# Apply PCA to reduce dimensionality
pca = PCA(n_components=0.95, random_state=42)  # Keep 95% of variance
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print(f"Original features: {X_train_scaled.shape[1]}")
print(f"PCA components: {X_train_pca.shape[1]}")
print(f"Explained variance ratio: {pca.explained_variance_ratio_.sum():.4f}")
```

Figure 7: PCA

- Train the Isolation Forest, Autoencoder, and LSTM models.

```python
print("\n--- Training Isolation Forest ---")

# Isolation Forest works with unlabeled data for anomaly detection
iso_forest = IsolationForest(contamination=0.1, random_state=42, n_jobs=-1)
iso_forest.fit(X_train_pca)

# Predict anomalies (-1 for anomaly, 1 for normal)
iso_pred_train = iso_forest.predict(X_train_pca)
iso_pred_test = iso_forest.predict(X_test_pca)

# Convert to binary classification (1 for fraud, 0 for normal)
iso_pred_train_binary = (iso_pred_train == -1).astype(int)
iso_pred_test_binary = (iso_pred_test == -1).astype(int)

# Evaluate Isolation Forest
iso_accuracy = accuracy_score(y_test, iso_pred_test_binary)
iso_precision = precision_score(y_test, iso_pred_test_binary)
iso_recall = recall_score(y_test, iso_pred_test_binary)
iso_f1 = f1_score(y_test, iso_pred_test_binary)

models_results['Isolation Forest'] = {
    'accuracy': iso_accuracy,
    'precision': iso_precision,
    'recall': iso_recall,
    'f1_score': iso_f1,
    'model': iso_forest,
    'predictions': iso_pred_test_binary
}
```

Figure 8: Training Models

- Perform hyperparameter tuning and cross-validation.

```
# ================================
# CROSS VALIDATION
# ================================

print("\n=== CROSS VALIDATION ===")

# Cross validation for Isolation Forest
cv_scores_iso = []
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

for train_idx, val_idx in kfold.split(X_train_pca, y_train):
    X_train_fold, X_val_fold = X_train_pca[train_idx], X_train_pca[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    iso_cv = IsolationForest(contamination=0.1, random_state=42)
    iso_cv.fit(X_train_fold)

    val_pred = iso_cv.predict(X_val_fold)
    val_pred_binary = (val_pred == -1).astype(int)

    cv_score = f1_score(y_val_fold, val_pred_binary)
    cv_scores_iso.append(cv_score)

print(f"Isolation Forest CV F1-Score: {np.mean(cv_scores_iso):.4f} (+/- {np.std(cv_scores_iso)*2:.4f})")
```

Figure 9: Cross Validation

- Evaluate and compare the models.

```
# ================================
# MODEL COMPARISON
# ================================

print("\n=== MODEL COMPARISON ===")

# Create comparison dataframe
comparison_data = []
for model_name, results in models_results.items():
    comparison_data.append({
        'Model': model_name,
        'Accuracy': results['accuracy'],
        'Precision': results['precision'],
        'Recall': results['recall'],
        'F1-Score': results['f1_score']
    })

comparison_df = pd.DataFrame(comparison_data)
print(comparison_df)

# Find best model based on F1-score
best_model_name = comparison_df.loc[comparison_df['F1-Score'].idxmax(), 'Model']
print(f"\nBest Model: {best_model_name}")
```

Figure 10: Evaluation of Model

- Save the trained models, scaler, PCA object, and other components to the saved_models/ directory.

```
# ================================
# SAVE MODELS AND COMPONENTS
# ================================

print("\n=== SAVING MODELS AND COMPONENTS ===")

# Create directory for saved models
os.makedirs('saved_models', exist_ok=True)

# Save scaler
joblib.dump(scaler, 'saved_models/scaler.pkl')
print("Scaler saved successfully")

# Save PCA
joblib.dump(pca, 'saved_models/pca_model.pkl')
print("PCA model saved successfully")

# Save best Isolation Forest model
if 'Isolation Forest (Tuned)' in models_results:
    joblib.dump(models_results['Isolation Forest (Tuned)']['model'],
                'saved_models/best_isolation_forest.pkl')
    print("Best Isolation Forest model saved successfully")

# Save Autoencoder
models_results['Autoencoder']['model'].save('saved_models/autoencoder_model.h5')
print("Autoencoder model saved successfully")

# Save autoencoder threshold
with open('saved_models/autoencoder_threshold.pkl', 'wb') as f:
    pickle.dump(models_results['Autoencoder']['threshold'], f)
print("Autoencoder threshold saved successfully")
```

Figure 11: Model Saving

5.2 Part B: Running the Flask Web Application for Live Predictions

The app.py script launches a web server with an interface for making real-time fraud predictions using the best-trained model (LSTM).

1. **Ensure Models are Saved:** Make sure you have successfully run the Jupyter Notebook (Part A) and the saved_models/ directory is populated.
2. **Start the Flask App:** In your terminal, with the virtual environment activated, run the following command:

python app.py

3. **Access the Application:** Open a web browser and navigate to:

http://127.0.0.1:5001

4. **Make a Prediction:** Fill in the transaction details in the web form and click "Predict". The application will process the input, run it through the saved LSTM model, and return a prediction indicating the probability of fraud and a corresponding risk level.

6 Code and Model Overview

6.1 Data Loading and Preprocessing

The system begins by loading train_transaction.csv and train_identity.csv and merging them into a single DataFrame. It then performs extensive cleaning by dropping columns with high-missing value percentages and imputing the remaining missing values (median for numerical, mode for categorical).

6.2 Exploratory Data Analysis (EDA)

Interactive visualizations are used to understand the data's characteristics, including the target variable distribution, transaction amounts, product codes, and time-based fraud patterns.

6.3 Feature Engineering

New, more informative features are created from existing data to improve model performance. This includes time-based features (hour, day), amount-based features (log-transformed amount), and encoded categorical features.

6.4 Dimensionality Reduction (PCA)

Principal Component Analysis (PCA) is applied to the scaled numerical features to reduce the high dimensionality of the dataset. The number of components is chosen to retain 95% of the variance, significantly reducing computational complexity while preserving most of the information.

6.5 Handling Class Imbalance (SMOTE)

The dataset is highly imbalanced, with far more legitimate transactions than fraudulent ones. The Synthetic Minority Over-sampling Technique (SMOTE) is used on the training data to create synthetic fraudulent samples, balancing the class distribution and helping the models learn the minority class patterns more effectively.

6.6 Model Architecture and Training

Three different anomaly detection models are trained and evaluated:

- **Isolation Forest:** An unsupervised model that works by isolating anomalies instead of profiling normal data points. It is tuned for optimal performance.
- **Autoencoder:** A deep neural network trained to reconstruct normal transactions. Transactions with high reconstruction errors are flagged as anomalies.

- **LSTM Network:** A recurrent neural network that processes data as sequences. This is the primary supervised model, trained on the balanced dataset to classify transactions as fraudulent or legitimate.

6.7 Model Evaluation and Persistence

Models are evaluated on a held-out test set using standard classification metrics (Accuracy, Precision, Recall, F1-Score). The best-performing model (LSTM) and all necessary preprocessing components (scaler, PCA) are saved to disk for later use in the Flask application.

7 Conclusion

7.1 Project Summary

This project successfully implemented a robust pipeline for detecting fraudulent credit card transactions. It encompassed all stages of a machine learning project, from data cleaning and exploratory analysis to advanced feature engineering, model training, and deployment via a web interface. By comparing multiple AI-powered anomaly detection techniques, we identified the most effective approach for this specific problem.

7.2 Key Findings

The comprehensive evaluation of the three models revealed that the **LSTM network was the best-performing model**, achieving the highest F1-Score of **0.5312**. This score indicates a strong balance between precision (correctly identifying fraud) and recall (capturing a high proportion of all actual fraud cases). While the unsupervised models (Isolation Forest and Autoencoder) provided a valuable baseline, the supervised LSTM model, trained on a balanced dataset, demonstrated superior predictive power for this classification task.

7.3 Future Enhancements

While the current system is effective, several avenues for future improvement exist:

- **Advanced Feature Engineering:** Incorporate more complex interaction features or features based on user transaction history over time.
- **Real-Time Data Streaming:** Integrate the system with a real-time data pipeline (e.g., using Apache Kafka) to analyze transactions as they occur.
- **Model Explainability:** Implement tools like SHAP or LIME to provide clear explanations for why a specific transaction was flagged as fraudulent.
- **Ensemble Modeling:** Combine the predictions of all three models into an ensemble to potentially achieve higher accuracy and robustness.

8 Appendix

8.1 requirements.txt

```
pandas
numpy
scikit-learn
tensorflow
plotly
matplotlib
seaborn
imbalanced-learn
flask
```

joblib

8.2 Web Application Interface

The Flask application provides a user-friendly form with the following fields:

- Transaction Amount
- Product Code
- Card1 through Card6 details
- Address 1 and Address 2
- Distance 1 and Distance 2
- Purchaser and Recipient Email Domains

Upon submission, it returns:

- **Fraud Probability:** A score from 0 to 1.
- **Is Fraud:** A binary decision (0 for No, 1 for Yes).
- **Risk Level:** A qualitative assessment (Low, Medium, High).
- **Confidence Score:** How confident the model is in its prediction.
- **Contributing Risk Factors:** A breakdown of rules-based checks that contribute to the risk score.

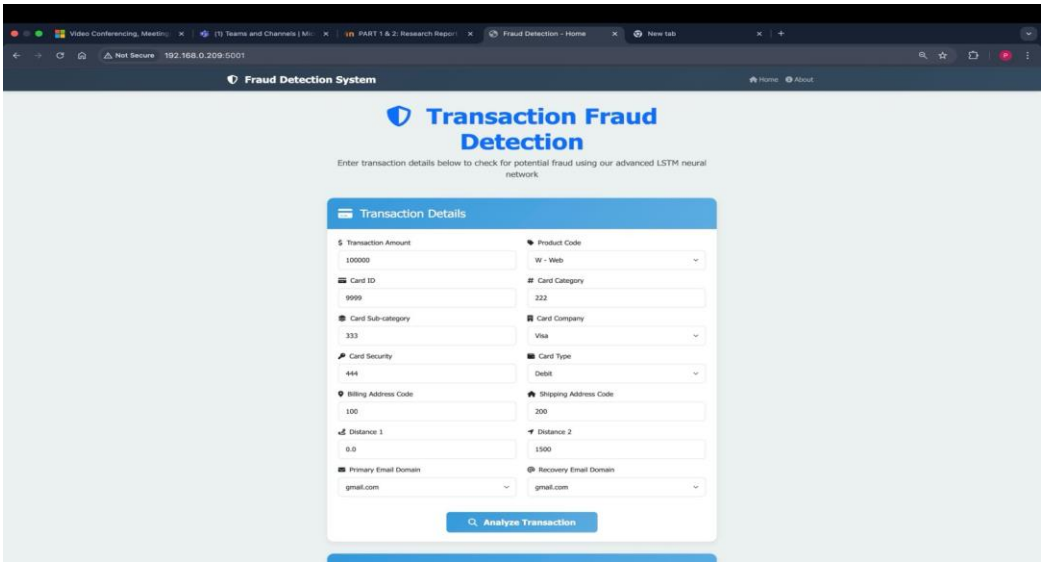

Figure 12: Web Application Interface Form

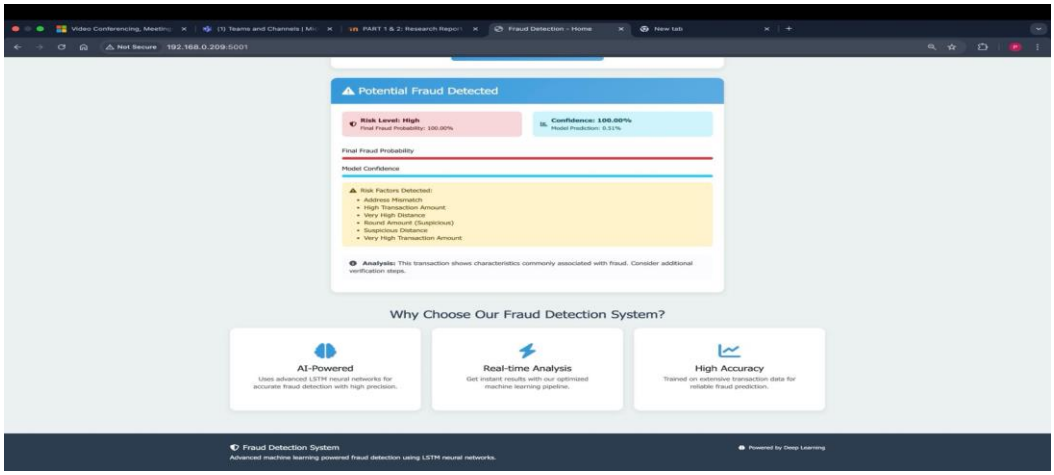

Figure 13: Web Application Interface Output