National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Kyaw May Pyone Shinn
Student ID: 24123455

School of Computing
National College of Ireland

Supervisor: Cristina Hava Muntean

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Kyaw May Pyone Shinn

Student ID: 24123455

Programme: MSc in Data Analysis Year: 2025

Module: MSc Research Practicum

Lecturer: Cristina Hava Muntean

Submission Due Date:

Project Title:

Page Count:

Word Count:

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: Shinn

Date: 11th December 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

Kyaw May Pyone Shinn
Student ID: x24123455

SME Sales Forecasting and Drift Detection Using Neural Networks and Public Macroeconomic Indicators

National College of Ireland
MSc in Data Analytics

1. Introduction

This configuration manual describes the technical environment required to reproduce the full data-processing and modelling workflow developed in this project. The system consists of four sequential Jupyter notebooks DataPull3years, CPI, SyntheticData, and Final_Project which collectively perform macroeconomic data extraction, inflation processing, synthetic SME dataset construction, feature engineering, forecasting model training, anomaly detection, and dashboard output generation.

The notebooks were executed using the Anaconda distribution of Python, which provides a stable package management environment and simplifies dependency installation. The entire pipeline is designed to run on a standard personal computer without specialised hardware, while still supporting optional acceleration through GPUs for deep learning training if available.

This section outlines the hardware and software requirements necessary to execute all four notebooks successfully.

2. Hardware Requirements

The project is designed to run on typical consumer-grade hardware. The following specifications were used during development and are recommended for smooth execution:

Minimum Requirements

- Processor: Dual-core CPU (Intel i3 / AMD equivalent)
- RAM: 8 GB
- Storage: 5 GB of free disk space
- Graphics: Integrated graphics (no GPU required)

Recommended Requirements

- Processor: Quad-core CPU (Intel i5/i7 or AMD Ryzen 5/7)
- RAM: 16 GB or more
- Storage: SSD with 10 GB free space
- Optional GPU: NVIDIA GPU with CUDA support for accelerated TensorFlow training

The LSTM model and autoencoder run efficiently on CPU, but training speed improves significantly if a compatible GPU is available.

3. Software Requirements

The following software components are required to run the notebooks in the project:

3.1 Anaconda Distribution

The entire project was executed within the Anaconda Python environment, which provides:

- A reliable package manager (conda)
- Pre-compiled numerical libraries
- Easy environment isolation
- Built-in support for Jupyter Notebook

Recommended version:

- Anaconda 2023.x or later
- Python 3.9–3.11

3.2 Core Python Libraries

The four notebooks use the following essential libraries:

Required Python Libraries (Table Format)

Category	Libraries
Core	pandas, numpy, os, glob, datetime, json, time, warnings
Data Access	requests, python-dotenv
Visualisation	matplotlib, seaborn, matplotlib.dates, plotly
Statistics	statsmodels (ADF test), scipy.stats (skew, kurtosis)
Machine Learning	scikit-learn (scalers, train_test_split, LinearRegression, LOF, metrics)
Deep Learning	tensorflow / keras (Sequential, LSTM, Dense, Dropout, EarlyStopping)

Installation Command

Run once inside the chosen Conda environment:

```
pip install pandas numpy requests matplotlib seaborn scikit-learn
tensorflow streamlit plotly python-dotenv
```

Execution Note

All notebooks must be opened from within the extracted Final_Project directory so that file paths resolve correctly.

3.3 Streamlit (for dashboard execution)

Although not part of the notebooks themselves, the final outputs generated in Final_Project.ipynb are used by a Streamlit dashboard.
Streamlit must be installed to run:

```
streamlit run app.py
```

4. Jupyter Notebook Environment

All notebooks were executed using:

- Jupyter Notebook (bundled with Anaconda)
- Running under the base Anaconda environment

To launch:

```
jupyter notebook
```

Each notebook can then be opened and executed sequentially.

Dataset Description

The project integrates real macroeconomic data with a simulated SME dataset to create a unified modelling dataset used for forecasting and anomaly detection. The dataset creation process follows three stages: (1) macroeconomic feature extraction, (2) inflation integration, and (3) synthetic SME simulation merged with macro drivers.

1. Macroeconomic Dataset (DataPull3years Notebook)

Source: Alpha Vantage
URL: https://www.alphavantage.co/

Two datasets are produced from three years of SPY and EUR/USD data:

1.1 Features Dataset – features_2021_2023.csv

Contains cleaned and engineered macro features:

- SPY closing price
- FX closing price
- SPY and FX daily returns
- 5-day rolling mean of SPY returns
- 5-day rolling standard deviation (volatility)
- Lagged return values
- Continuous date index

Shape: (747 rows, 9 columns)

1.2 Merged Market Dataset – merged_2021_2023.csv

Contains raw merged SPY and FX series before inflation is added.

Shape: (753 rows, 4 columns)

2. Inflation Dataset (CPI Notebook)

Source: Federal Reserve Bank of St. Louis (FRED)
URL: https://fred.stlouisfed.org/series/CPALTT01USM657N

The monthly CPI series is:

- Loaded
- Cleaned
- Converted to daily frequency
- Merged with the features dataset

2.1 Final Macroeconomic + CPI Dataset – merged_us_macro_dataset.csv

This file integrates all macroeconomic variables:

- Market close
- FX close
- Daily returns

- Rolling features
- CPI
- Daily inflation index

Shape: (747 rows, 10 columns)

This dataset becomes the macroeconomic base for SME simulation.

3. Synthetic SME Dataset (SyntheticData Notebook)

Source: Internally generated using controlled Python simulation logic
Merged With: merged_us_macro_dataset.csv

The synthetic SME simulation produces firm-level time-series data for 50 firms, each observed across 747 days.
Variables include:

- customer_traffic
- ad_spend
- sales_revenue
- operating_costs
- cash_on_hand

Each firm has unique behavioural parameters (conversion rate, demand elasticity, inflation sensitivity, FX sensitivity, baseline traffic, promotional effects).
Macro variables influence demand, costs, and spending patterns.

3.1 Synthetic SME Dataset – synthetic_sme_dataset.csv

Contains granular SME operation records prior to macro merge.

3.2 Final Merged SME + Macro Dataset – final_dataset.csv

The final dataset is created by merging the synthetic SME panel with the macroeconomic dataset on the date column.
It contains:

- 5 SME KPI fields
- 9 macro features
- 1 identifier (firm_id)

Total Columns: 15
Total Rows: 37,350
Shape: (37,350 rows, 15 columns)

This dataset is used as the primary modelling input for:

- LSTM forecasting
- Autoencoder anomaly detection
- Dashboard output generation

Model Configuration

The project trains three models inside the *Final_Project.ipynb* notebook:

1. Multiple Linear Regression (Baseline Model)
   Used to establish benchmark forecasting performance and generate sensitivity coefficients for macro-scenario analysis.

Model Training and Performance Evaluation (Linear Regression) ¶

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

scaler = StandardScaler().fit(train_X)
train_X_scaled = scaler.transform(train_X)
val_X_scaled   = scaler.transform(val_X)
test_X_scaled  = scaler.transform(test_X)

model = LinearRegression().fit(train_X_scaled, train_y)
val_pred  = model.predict(val_X_scaled)
test_pred = model.predict(test_X_scaled)

print("Val RMSE:", np.sqrt(mean_squared_error(val_y, val_pred)))
print("Val R²:",  r2_score(val_y, val_pred))
print("Test RMSE:", np.sqrt(mean_squared_error(test_y, test_pred)))
print("Test R²:",  r2_score(test_y, test_pred))
```

```
Val RMSE: 172.85232073320662
Val R²: 0.8515546284358313
Test RMSE: 180.14452048169684
Test R²: 0.8546670687552708
```

2. LSTM Forecasting Model
   Trained on 30-day sequences of SME and macroeconomic variables to produce next-day sales forecasts.
   Its predictions are saved to lstm_predictions.csv and consumed by the Streamlit dashboard.

Implementing LSTM forecasting model

```
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score

# 1. Select features and target
features = ['sales_revenue','cash_on_hand','operating_costs','ad_spend','customer_traffic',
            'market_close','fx_close','inflation_index']
target = 'sales_revenue_next'

# scale values
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df[features])
scaled_df = pd.DataFrame(scaled, columns=features)
scaled_df['sales_revenue_next'] = df[target].values
scaled_df['date'] = df['date']

# 2. Create sequences (e.g., 30-day window)
def create_sequences(data, seq_length=30):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data.iloc[i:i+seq_length][features].values)
        y.append(data.iloc[i+seq_length][target])
    return np.array(X), np.array(y)

seq_length = 30
X, y = create_sequences(scaled_df, seq_length)

# 3. Train/validation/test split by date (time-based)
train_size = int(len(X)*0.7)
val_size   = int(len(X)*0.15)
X_train, y_train = X[:train_size], y[:train_size]
X_val, y_val = X[train_size:train_size+val_size], y[train_size:train_size+val_size]
X_test, y_test = X[train_size+val_size:], y[train_size+val_size:]

# 4. Build LSTM model
model = Sequential([
    LSTM(64, return_sequences=True, input_shape=(seq_length, len(features))),
    Dropout(0.2),
    LSTM(32),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
history = model.fit(X_train, y_train, epochs=30, batch_size=32,
                    validation_data=(X_val, y_val), verbose=1)
```

```
Epoch 1/30
C:\Users\kmpsh\anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
r. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
808/808 ━━━━━━━━━━━━━━━━━━━━ 12s 12ms/step - loss: 1377429.8750 - val_loss: 1579742.1250
Epoch 2/30
808/808 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - loss: 1320659.1250 - val_loss: 1522207.3750
Epoch 3/30
808/808 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 1266823.1250 - val_loss: 1466499.7500
Epoch 4/30
808/808 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - loss: 1214724.7500 - val_loss: 1412504.2500
Epoch 5/30
808/808 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - loss: 1164192.8750 - val_loss: 1359979.7500
Epoch 6/30
```

3. Autoencoder Drift-Detection Model
   An unsupervised model trained on normal historical patterns to identify deviations in SME behaviour.
   Its outputs are saved to ae_anomaly_flags.csv and used in the Anomaly Summary dashboard.

Build & train the autoencoder (Dense)

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

tf.random.set_seed(42)

n_in = len(FEATS_AE)

ae = Sequential([
    Dense(64, activation='relu', input_shape=(n_in,)),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),          # bottleneck
    Dense(32, activation='relu'),
    Dense(64, activation='relu'),
    Dense(n_in, activation='linear')
])

ae.compile(optimizer='adam', loss='mse')

es  = EarlyStopping(monitor='val_loss', patience=6, restore_best_weights=True)
rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, verbose=1)

hist = ae.fit(
    train_X, train_X,              # reconstruct inputs
    validation_data=(val_X, val_X),
    epochs=80, batch_size=256,
    shuffle=True, callbacks=[es, rlr], verbose=1
)
```

```
Epoch 1/80
C:\Users\kmpsh\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:92: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a lay
er. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.4415 - val_loss: 0.0935 - learning_rate: 0.0010
Epoch 2/80
96/96 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0319 - val_loss: 0.0194 - learning_rate: 0.0010
Epoch 3/80
96/96 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0106 - val_loss: 0.0131 - learning_rate: 0.0010
Epoch 4/80
96/96 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0076 - val_loss: 0.0110 - learning_rate: 0.0010
Epoch 5/80
96/96 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0061 - val_loss: 0.0088 - learning_rate: 0.0010
Epoch 6/80
96/96 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0051 - val_loss: 0.0078 - learning_rate: 0.0010
Epoch 7/80
96/96 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0045 - val_loss: 0.0071 - learning_rate: 0.0010
Epoch 8/80
```

All models are trained in a single pipeline. The dashboard does not retrain models; it only loads the exported prediction and anomaly files. Therefore, *Final_Project.ipynb* must be executed before running the dashboard.

Model Overview

Three models are trained in the project: a Multiple Linear Regression model used as the baseline, an LSTM model for sequence-based forecasting, and an Autoencoder used for drift and anomaly detection. Each model contributes a different output to the system, and their performance metrics confirm that the pipeline is functioning correctly, with the regression model providing the strongest baseline, the LSTM offering temporal forecasting capability, and the Autoencoder supplying drift-detection signals for the dashboard.

Model Accuracy Summary

Model	Metric	Score
Multiple Linear Regression	R^2	0.85
	RMSE	≈180
LSTM Forecasting Model	R^2	0.83
	RMSE	≈199
Autoencoder (Drift Detection)	Recall	≈0.60

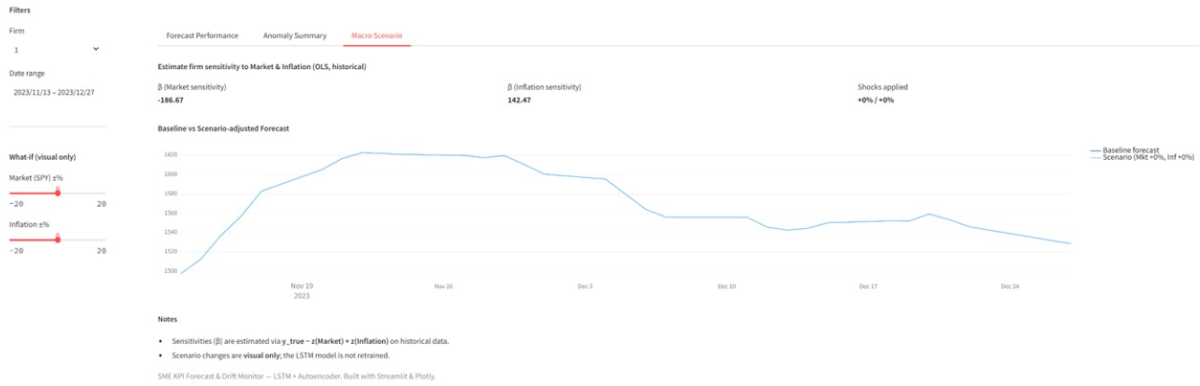
Dashboard Execution (Streamlit)

The dashboard is launched after model outputs are generated. Streamlit loads the exported files and displays the three dashboard tabs:

- Forecast Performance: actual vs forecast, RMSE, R^2, anomaly markers

- Anomaly Summary: reconstruction error, threshold, top drifted firms, heatmap

- Macro Scenario: baseline vs scenario-adjusted forecast using market and inflation shocks



Step-by-Step Workflow to Run the Entire Project (ZIP Download Version)

This guide explains how to run the project from scratch after downloading the ZIP file.
A user does not need to understand your code — they only follow the steps below.

1. Download and Extract the ZIP File

1. Download the project ZIP file.
2. Extract it anywhere on your computer (e.g., Desktop or Documents).
3. After extraction, you will have a folder named:

Final_Project

Inside this folder, you will see:

- Final_Project.ipynb
- DataPull3years.ipynb
- SyntheticData.ipynb
- CPI.ipynb
- A Datasets/ folder
- A Dashboard/ folder (contains app.py)

This is the correct structure.

2. Open Jupyter Notebook Using Anaconda

1. Open Anaconda Navigator or Anaconda Prompt.
2. Choose the base environment or create a clean one if needed.
3. Launch Jupyter Notebook.

When Jupyter opens in the browser:

4. Navigate to the extracted folder:

Final_Project/

This folder must contain the four notebooks.

3. Install Required Python Packages

Before running any notebook, install the required libraries.

In Anaconda Prompt:

```
conda activate base
```

```
cd path/to/Final_Project
```

```
pip install pandas numpy requests matplotlib seaborn scikit-learn
tensorflow streamlit plotly python-dotenv
```

If the user uses a separate conda environment, they just activate that environment first.

This must be done once.

4. Run the Notebooks in the Correct Order

Each notebook depends on files produced by the previous one.
The user must run all notebooks in this order:

Step 4.1 Run DataPull3years.ipynb

Purpose: Pull historical macro data (SPY, FX) and create engineered features.

1. Open DataPull3years.ipynb
2. Click Run → Run All
3. This notebook will create:
    - features_2021_2023.csv.xlsx
    - merged_2021_2023.csv.xlsx

Both files will appear inside the Final_Project folder.

Step 4.2 Run CPI.ipynb

Purpose: Pull CPI from FRED, convert to daily, and merge with macro data.

1. Open CPI.ipynb
2. Run all cells
3. This notebook will create:
    - CPALTT01USM657N.csv.xlsx
    - merged_us_macro_dataset.csv.xlsx

Saved into Final_Project.

Step 4.3 Run SyntheticData.ipynb

Purpose: Generate synthetic SME dataset and merge with macro dataset.

1. Open SyntheticData.ipynb
2. Run all cells
3. This notebook will create:
    - synthetic_sme_dataset.csv.xlsx
    - final_dataset.csv.xlsx (37,350 rows, 15 columns)

Saved into Final_Project.

This is the final dataset used for modelling.

Step 4.4 Run Final_Project.ipynb

Purpose: Train all models and export results for the dashboard.

Models trained:

- Multiple Linear Regression
- LSTM
- Autoencoder

1. Open Final_Project.ipynb
2. Run all cells
3. It will produce three required output files:

- lstm_predictions.csv
- ae_anomaly_flags.csv
- macro_daily.csv

These appear in the Final_Project/ folder (same level as the notebooks).

These files are required for the dashboard to function.

5. Run the Streamlit Dashboard

After all notebooks are executed successfully:

1. Open Anaconda Prompt
2. Navigate to the project folder:

```
cd path/to/Final_Project
```

3. Run the dashboard:

```
streamlit run app.py
```

This opens a web app with:

- Forecast Performance
- Anomaly Summary
- Macro Scenario

These tabs visualize the outputs generated by the LSTM and Autoencoder.

Important Notes for Users

- Always run the project inside the original extracted Final_Project directory. All notebooks are written to load files using direct filenames (e.g., "merged_us_macro_dataset.csv.xlsx"). This works only when the dataset files remain in their expected locations.
- If a notebook fails to load a file (e.g., FileNotFoundError), this usually means the dataset has been moved. Ensure that all data files are stored in the correct folder:

Final_Project/Datasets/

- When importing files manually or using a custom path, remember to include the folder name. For example:

pd.read_excel("Datasets/merged_us_macro_dataset.csv.xlsx")

- Do not rename dataset files. The notebooks expect the original filenames exactly as provided.
- The notebooks must be executed in order, because each one generates new datasets required by the next step. If any notebook is skipped, later notebooks will not run correctly.