Configuration Manual

MSc Research Project
MSc Artificial Intelligence for Business

Ana Paula Elizondo de la Garza
Student ID: 23140879

School of Computing
National College of Ireland

Supervisor: Victor del Rosal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ana Paula Elizondo de la Garza
Student ID:	23140879
Programme:	MSc A.I. for Business **Year:** 2025
Module:	Practicum
Lecturer:	Victor del Rosal
Submission Due Date:	August 11, 2025
Project Title:	Application of AI in E-Commerce for Startups with Limited Data: A Learn Startup Approach to Demand Forecasting and Inventory Optimization
Word Count:	1,189 **Page Count:** 10

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: Ana Paula Elizondo de la Garza

Date: August 10, 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

Ana Paula Elizondo de la Garza
Student ID: 23140879

1 Introduction

This report provides a descriptive overview of the related technical environment and implementation procedures used in the project: *"Application of AI in E-Commerce for Startups with Limited Data: A Lean Startup Approach to Demand Forecasting and Inventory Optimization"*. It highlights the datasets descriptions, hardware and software specifications and the step-by-step processes for implementing forecasting algorithms and developing a minimum viable product (MVP). It also outlines the development pipeline using Visual Studio code and Google Collab with multiple Python libraries.

2 Project Overview

This project revises the effectiveness of four different forecasting models: SARIMAX + Bootstrap, Baseline LSTM, LSTM + Transfer Learning + Data Augmentation and XGBoost for demand forecasting in e-commerce environments with data constraints. A comparative analysis was conducted using multiple Python libraries with emphasis on implementation accuracy, usability and performance evaluation. This configuration manual also describes the step by step of the development of minimum viable product (MVP), user interface, build in Visual Studio Code. Key phases included in this report are data preprocessing techniques, feature engineering, model training and evaluation to support reproducibility and clarity for future implementations.

3 System Requirements

3.1 Hardware Specifications

This project was developed fully and tested in the following hardware:

- **Device**: MacBook Air (2022)
- **Processor**: Apple M2 chip, 8-core CPU
- **RAM**: 16 GB unified memory
- **Storage**: 512 GB SSD
- **Graphics**: Integrated 10-core GPU (Apple Silicon)
- **Operating System**: macOS Ventura 13.x (64-bit)

Additionally, cloud computing resources in Google Collab were used for accelerated training of specific Python libraries and deep learning models.

3.2 Software Specifications

The following tools and libraries were used in this project for implementation, testing and analysis:

- **Programming Language**: Python 3.10

- **Development Environments**:
  - Visual Studio Code (VSC)
  - Google Collab Pro

- **Libraries and Frameworks**:
  - `pandas, numpy, scikit-learn, matplotlib, seaborn`
  - `statsmodels` (for SARIMAX)
  - `tensorflow` and `keras` (for LSTM models)
  - `xgboost`
  - `joblib` and `pickle` for model serialization
  - `Flask` for MVP interface deployment

- **Version Control**: GitHub

3.3 Datasets

The project used real datasets from a Mexican perfume e-commerce startup and a synthetic dataset extracted from Kaggle relevant to e-commerce demand forecasting. The key sources included:

- **Primary Dataset**: A time-series dataset of daily product sales was obtained from the e-commerce startup Shopify's API, then it was anonymized and separated in a CSV file for further analysis and evaluation (personal communication, May 21, 2025).

- **Data Format**: CSV, structured with fields including `Date`, `total_sales`, `total_sessions`, `inventory`, and `ingore` (stockout periods marked by the startup's founder).

- **Additional Synthetic Data**: "Perfume E-Commerce Dataset 2024" dataset was extracted from Kaggle to perform transfer learning and data augmentaiton techniques (Kanchana1990, 2024).

- **Preprocessing**: Handled missing values, normalization of numerical data, encoded categorical variables, and performed time-based feature engineering.

4 Environment Setup

This section highlights the local and cloud-based environments needed for the overall project execution.

4.1 Local Environment

The local environment of this project was configured in Visual Studio Code following this setup:

- **Operating System**: macOS Ventura 13.x
- **Python Version**: 3.10 (installed via Homebrew)
- **Environment Management**: `venv` (Python virtual environment)

- **Setup Step-by-Step:**
  - Install Homebrew
    - */bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"*
  - Install Python 3.10
    - *brew install python@3.10*
  - Create a Virtual Environment
    - *python3 -m venv venv*
    - *source venv/bin/activate*
  - Install the required libraries following requirements.txt (review Figure 1)
    - *pip install -r requirements.txt*
  - Launch the project in Visual Studio Code
    - *code .*

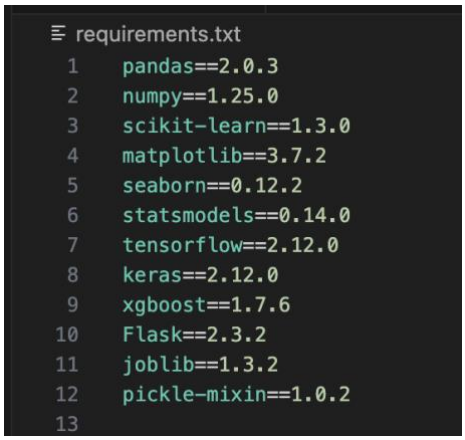

Figure 1. Requirements.txt

4.2 Cloud Environment (Google Collab)

Due to computational limitations to run certain Python libraries in the local device, Google Collab was used to leverage GPU acceleration and deep learning models development.

- **Steps:**
  - Upload the notebook and dataset to your Google Drive.
  - Open a Jupyter Notebook in your Collab.
  - Install any missing libraries in the first cell of your notebook (review Figure 2).
    - *!pip install xgboost tensorflow scikit-learn statsmodels*

```
# ── A) Libraries and Drive ──────────────────────────────────────────
import numpy as np
import pandas as pd
from scipy.interpolate import CubicSpline
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import (
    Input, LSTM, Dropout, RepeatVector, TimeDistributed, Dense
)
from tensorflow.keras.callbacks import EarlyStopping

from google.colab import drive
drive.mount('/content/drive', force_remount=True)

import matplotlib.pyplot as plt
```

Figure 2. Libraries and Drive requirements for Google Collab

4.3 Version Control

Version control and tracking was performed in GitHub. The steps are the following:

- Git initialized in the project folder
- Remote repository connected to GitHub
- Regular commits and pushes-maintained project history

5 Forecasting Algorithms

Four models were developed and evaluated to forecast daily-sales in limited data e-commerce environments. The implementation requirements are listed below.

5.1 SARIMAX + Bootstrap

- **Data Preparation**: Daily time series for `total_sales` constructed using forward filling.
- **Model Parameters**: SARIMA order was set as (1,1,1)(1,1,1,7) to identify weekly seasonality.
- **Forecasting**: 15-day forecast generated using `statsmodels`.

- **Bootstrap**: Confidence intervals estimated through built-in prediction intervals rather than manual resampling (review Figure 3 and 4).

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX

# 1. Prepare the historical series without the rows marked as ignore
df = combined_final.copy()
df.index = pd.to_datetime(df.index)
sales = df.loc[~df['ignore'], 'total_sales'].astype(float)
# Rellenar huecos si los hubiera (opcional)
sales = sales.asfreq('D').fillna(method='ffill')

# 2. Adjust your final SARIMAX over the entire series
model = SARIMAX(
    sales,
    order=(2,0,2),
    seasonal_order=(0,1,1,7),
    enforce_stationarity=False,
    enforce_invertibility=False
)
fit = model.fit(disp=False)

# 3. Average forecast for the next 15 days
horizon = 15
fc = fit.get_forecast(steps=horizon)
mean_fc = fc.predicted_mean

# 4. Bootstrap residuals to create uncertainty bands
resid = fit.resid.dropna().values
nsim  = 500

# We simulate nsim trajectories of length `horizon`
sims = np.zeros((horizon, nsim))
for i in range(nsim):
    eps = np.random.choice(resid, size=horizon, replace=True)
    sims[:, i] = mean_fc.values + eps
# prevent negative sales
sims = np.clip(sims, 0, None)

# 5. Calculate central and extreme percentiles
p20   = np.percentile(sims, 20, axis=1)
p80   = np.percentile(sims, 80, axis=1)
p3    = np.percentile(sims, 3,  axis=1)
p97   = np.percentile(sims, 97, axis=1)
```

Figure 3. SARIMAX + Bootstrap configuration part 1

```python
# 6. Assemble a DataFrame with the forecast
future_dates = pd.date_range(start=sales.index[-1] + pd.Timedelta(days=1),
                             periods=horizon, freq='D')
forecast_df = pd.DataFrame({
    'mean_fc':   mean_fc.values,
    'p20':       p20,
    'p80':       p80,
    'p3':        p3,
    'p97':       p97
}, index=future_dates)

# 7. History graph + forecast + bands
plt.figure(figsize=(12,6))
# Historic
plt.plot(sales.index, sales, color='C0', label='Histórico')

# Average forecast
plt.plot(future_dates, forecast_df['mean_fc'], color='C1', lw=2, label='Pronóstico (media)')

#  20–80%
plt.fill_between(future_dates,
                 forecast_df['p20'],
                 forecast_df['p80'],
                 color='C2', alpha=0.3, label='20–80% PI')

#  3–97%
plt.fill_between(future_dates,
                 forecast_df['p3'],
                 forecast_df['p97'],
                 color='C1', alpha=0.1, label='3–97% PI')

plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('15-day forecast with SARIMAX + Residual Bootstrap')
plt.legend()
plt.tight_layout()
plt.show()

# 8. Display the forecast DataFrame
print(forecast_df)
```

Figure 4. SARIMAX + Bootstrap configuration part 2

5.2 Baseline LSTM

- **Architecture**: Sequence-to-sequence model with two LSTM layers, a RepeatVector, and TimeDistributed dense layers (review Figure 5).
- **Preprocessing**: Normalization with MinMaxScaler; sliding window sequences (30 timesteps) prepared as inputs.
- **Training**: 100 epochs with early stopping; evaluation using RMSE and MAE.
- **Output**: 7-day ahead forecasts for both sales and sessions.

```
# ── 9) Define the seq2seq model "With calendar" ──────────────────────
model = Sequential([
    Input(shape=(window_size, n_f)),
    LSTM(128, return_sequences=True),
    LSTM(64),
    Dropout(0.1),
    RepeatVector(horizon),
    LSTM(64, return_sequences=True),
    LSTM(32, return_sequences=True),
    TimeDistributed(Dense(n_f))
])
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.summary()


# ── 10) Train hard (up to 100 epochs, patience=5) ────────────────────
es = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100,
    batch_size=32,
    callbacks=[es]
)


# ── 11) Learning Curve ──────────────────────────────────────────────
plt.figure(figsize=(8,4))
plt.plot(history.history['loss'],  label='train_loss')
plt.plot(history.history['val_loss'],label='val_loss')
plt.xlabel('Época'); plt.ylabel('MSE')
plt.legend(); plt.show()
```

Figure 5. LSTM training and learning curve

5.3 LSTM + Transfer Learning + Data Augmentation

- **Transfer Learning**: Pretrained on the Kaggle perfume dataset, then fine-tuned on the main dataset (Kanchana1990, 2024).
- **Augmentation Techniques**: Jittering, scaling, and permutation applied via `tsaug` to expand training data.

```python
# 2) Imports and routes
import os, numpy as np, pandas as pd, matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from math import sqrt

DATA_DIR  = '/content/drive/MyDrive/ecommerce_dashboard/data'
MODEL_DIR = '/content/drive/MyDrive/ecommerce_dashboard/models'
os.makedirs(MODEL_DIR, exist_ok=True)
```

Figure 6. LSTM + DA + TL Datasets and libraries

```python
# 7) Building and training the LSTM
model_pre = Sequential([
    LSTM(64, input_shape=(KAG_WINDOW,1), return_sequences=True),
    Dropout(0.2),
    LSTM(32),
    Dense(1)
])
model_pre.compile(optimizer=Adam(LR_PRE), loss='mean_squared_error')

cp = ModelCheckpoint(f"{MODEL_DIR}/pre_lstm.h5", save_best_only=True, monitor='val_loss')
history_pre = model_pre.fit(
    Xk, yk,
    epochs=EPOCHS_PRE,
    batch_size=bs_pre,
    validation_split=0.2,
    callbacks=[cp],
    verbose=2
)
```

Figure 7. Kaggle Dataset Transfer Learning LSTM

5.4 XGBoost

- **Features**: Lag features, rolling means, and calendar encodings (e.g., day of week, month) engineered.
- **Model**: MultiOutputRegressor wrapper used for 7-day forecasting across targets.
- **Evaluation**: Same metrics (RMSE, MAE) used to benchmark performance against the other three selected models.

```
# ── 4) Scaler ──────────────────────────
scX = MinMaxScaler().fit(X)
Xs  = scX.transform(X)

scY = MinMaxScaler().fit(Y)
Ys  = scY.transform(Y)


# ── 5) Split  Train / Val ──────────────
n = len(Xs)
cut = int(n * 0.8)

Xtr, Xva = Xs[:cut], Xs[cut:]
Ytr, Yva = Ys[:cut], Ys[cut:]
date_va  = date_ends[cut:]


# ── 6) Train XGBoost multi-output ──────
xgb_model = MultiOutputRegressor(
    xgb.XGBRegressor(
        n_estimators=300,
        learning_rate=0.05,
        max_depth=4,
        subsample=0.8,
        random_state=42,
        verbosity=0
    )
)
xgb_model.fit(Xtr, Ytr)
```

Figure 8. XGBoost Training

6 Minimum Valuable Product (MVP)

6.1 Scope

Create a lightweight, interactive web interface designed for e-commerce startups to centralize their critical operations and AI-driven forecasts to increment productivity and certainty in their decisions and next steps. The interface was developed following the Lean Startup methodology (Ries, 2011) and validated by a real startup founder. It provides:

- 7-day demand forecasts based on SARIMAX model with three configurations: conservative, neutral and optimistic.
- Visualizations of forecast trends and inventory status
- An input interface to adjust parameters (e.g., forecast type, date range)
- Downloadable reports for business use

6.2 Technology Used

- **Frontend**: HTML, CS and JavaScript
- **Backend**: Flask (Python)
- **Visualization**: Matplotlib and Plotly for forecast charts
- **Deployment**: Local Flask server (tested on macOS Ventura)

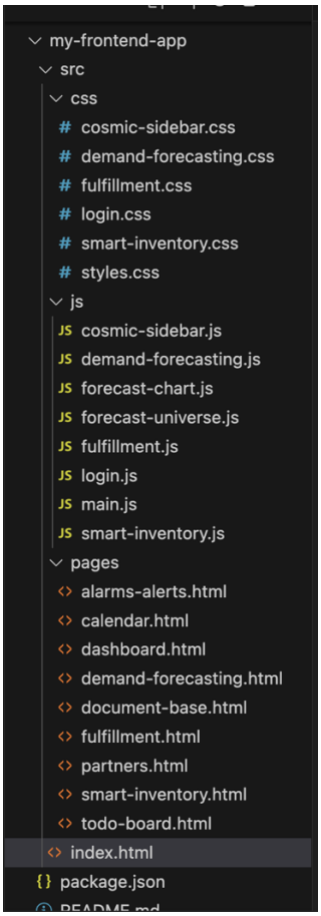

Figure 9. MVP Project in VSC documents

6.3 Limitations and Reproducibility

The MVP can be visualized by running `Python -m HTTP.server` within the configured virtual environment. The required dependencies are listed in the document called `requirements.txt`.

Limitations:

- The MVP is local-only (no cloud hosting).
- The dashboard is the only section fully functional, other sections are represented visually for further development in next MVP iterations (review Figure 10).
- It assumes structured input format and may need enhancements for commercial deployment such as user authentication, file uploads and mobile-first design.

Figure 10. Dashboard Visualization

Note: You can access to the front end thorugh this link.

7 Conclusion

The configuration manual shown in this report provides the necessary steps to replicate the technical setup and implementation of the MSc in AI for business project on AI-driven demand forecasting for e-commerce startups for limited data. It covers 6 sections related to the environmental setup, tested forecasting models (SARIMAX, two LSTM variants, XGBoost) and the development of a first MVP with validated learning from a real e-commerce startup. This documentation supports reproducibility and offers a foundation for further development.

References

Kanchana1990. (2024). *Perfume e-commerce dataset 2024* [Data set]. Kaggle. https://www.kaggle.com/datasets/kanchana1990/perfume-e-commerce-dataset-2024

Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Business.