National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Olamide Abioro
Student ID: x23428902

School of Computing
National College of Ireland

Supervisor: Sallar Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	OLAMIDE ABIORO
Student ID:	X23428902
Programme:	MSCDAD_C_JAN25 **Year:** 2025
Module:	Research Practicum
Lecturer:	Sallar Khan
Submission Due Date:	January 28, 2026
Project Title:	Predictive and Comparative Analysis of Social Media Platform Performance for SMEs in Ireland and Nigeria
Word Count:	1,431 **Page Count:** 7

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: Olamide Abioro

Date: December 4, 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

Olamide Abioro
Student ID: x23428902

1 System overview and requirements

This configuration manual describes how to set up and run the social media analytics pipeline used in the thesis; *Predictive and Comparative Analysis of Social Media Platform Performance for SMEs in Ireland and Nigeria*. The system ingests public SME posts from Instagram, TikTok and Facebook into a PostgreSQL database, constructs features and labels in Python and then trains Random Forest and XGBoost classifiers to predict high engagement posts.

The configuration has three main layers:

1. a Python data science environment,
2. a PostgreSQL database and connection,
3. project specific scripts and notebooks for ingestion, feature engineering and modelling.

1.1 Hardware requirements

The experiments were run on a Macbook. The following minimum specifications are recommended:

- CPU: quad core processor,
- RAM: 16 GB (8 GB is possible, but processing will be slower),
- Storage: at least 20 GB free disk space for raw data, database files and intermediate artefacts.

A more powerful machine or a cloud instance will reduce training and processing time but is not strictly required.

1.2 Software requirements

The software stack consists of:

- Operating system: Windows 10 or 11, macOS, or a recent Linux distribution,
- Python: version 3.0 or later,
- Package manager: Conda (Anaconda or Miniconda) for environment isolation,

- Database: PostgreSQL version 14 or later (PostgreSQL Global Development Group 2023),
- Development environment: JupyterLab or Visual Studio Code with Python and Jupyter support.

Key Python libraries:

- `pandas` for tabular data manipulation (McKinney 2010),
- `numpy` for numerical computing (Harris et al. 2020),
- `scikit_learn` for Random Forest and evaluation (Pedregosa et al. 2011),
- `xgboost` for gradient boosted trees (Chen & Guestrin 2016),
- `sqlalchemy` and `psycopg2` for database access,
- `matplotlib` for visualisation (Hunter 2007),
- `jupyterlab` for notebook based workflows (Kluyver et al. 2016).

A typical Conda environment can be created as follows:

```
conda create -n sme-social python=3.10
conda activate sme-social
pip install pandas numpy scikit-learn xgboost sqlalchemy psycopg2-binary \
            matplotlib jupyterlab
```

You can adjust versions as needed, provided the libraries remain compatible.

2 Database and data ingestion configuration

This section describes how to configure PostgreSQL and connect it to the Python ingestion scripts that import Apify data.

2.1 PostgreSQL setup

1. **Install PostgreSQL**

   Install PostgreSQL from the official distribution. During installation, note the superuser name (usually `postgres`), password and port (usually `5432`).

2. **Create a database for the project**

   After installation, create a dedicated database, for example:

   ```
   CREATE DATABASE sme_social;
   ```

3. **Create a database user**

   Optionally create a separate user with limited privileges:

```
CREATE USER sme_user WITH PASSWORD 'your_password';
GRANT ALL PRIVILEGES ON DATABASE sme_social TO sme_user;
```

4. **Schema design**

   The project expects at least the following tables:

   - `accounts`

     Core fields:

     - `account_id` (primary key),
     - `platform` (Instagram, TikTok, Facebook),
     - `country` (IE, NG),
     - `sector` (fashion, beauty, retail),
     - `username`,
     - `max_followers`.

   - `posts`

     Core fields:

     - `post_id` (primary key),
     - `account_id` (foreign key to `accounts`),
     - `posted_at` (timestamp),
     - `content_type` (image, video, reel and similar),
     - `caption`,
     - engagement metrics, for example likes, comments, shares, views,
     - `followers_at_post`.

   - optional `apify_raw` table to store original JSON payloads for logging.

   The ingestion scripts can create these tables automatically if they are missing or they can be created using explicit `CREATE TABLE` statements.

2.2 Database connection configuration

Python scripts connect to PostgreSQL using a connection URL, for example:

```
postgresql+psycopg2://sme_user:your_password@localhost:5432/sme_social
```

To avoid hardcoding credentials, define environment variables:

- `SME_DB_USER`,
- `SME_DB_PASSWORD`,
- `SME_DB_HOST`,
- `SME_DB_PORT`,

- SME_DB_NAME.

A configuration module then constructs the URL, for example:

```
import os

DB_USER = os.getenv("SME_DB_USER")
DB_PASS = os.getenv("SME_DB_PASSWORD")
DB_HOST = os.getenv("SME_DB_HOST", "localhost")
DB_PORT = os.getenv("SME_DB_PORT", "5432")
DB_NAME = os.getenv("SME_DB_NAME", "sme_social")

SQLALCHEMY_URL = (
    f"postgresql+psycopg2://{DB_USER}:{DB_PASS}"
    f"@{DB_HOST}:{DB_PORT}/{DB_NAME}"
)
```

All scripts import and reuse this connection string.

2.3 Apify configuration

Apify is used to scrape public SME posts from the three platforms.

1. **Apify account and token**

   Create an Apify account and obtain an API token. Set this as an environment variable, for example APIFY_TOKEN.

2. **Actors and inputs**

   The project uses actors that can retrieve posts from:

   - Instagram profiles,
   - TikTok accounts,
   - Facebook pages.

   Each actor requires:

   - a list of profile or page URLs or usernames for SME accounts,
   - a maximum number of posts per account,
   - optional date filters, for example from a given year onwards.

3. **Ingestion scripts**

   The following Python scripts orchestrate data collection and ingestion:

   - ingest_apify_instagram.py,
   - ingest_apify_tiktok.py,

- `ingest_apify_facebook.py`.

Each script:

- triggers the corresponding Apify actor with a JSON input,
- downloads the resulting dataset,
- normalises the JSON to the `accounts` and `posts` schema,
- inserts or updates records in PostgreSQL.

3 Running the analytics pipeline

This section explains how to move from a populated database to the final models and evaluation outputs.

3.1 Feature and label generation

After ingestion, feature and label generation is performed in Jupyter Notebooks, for example `EDA_and_ML_intagram.ipynb`, `EDA_and_ML_tiktok.ipynb` and `EDA_and_ML_facebook.ipynb`.

The typical procedure is:

1. Connect to PostgreSQL and load `accounts` and `posts` into pandas DataFrames.
2. Apply quality filters:
   - remove accounts with fewer than 300 followers,
   - drop posts with missing timestamps or follower counts,
   - exclude strongly promotional posts such as giveaways or heavy discount campaigns with regular expressions on captions.
3. Compute engagement metrics:
   - calculate engagement rate as interactions divided by follower count,
   - compute a log transformed engagement rate with a small constant for stability,
   - compute account level mean log rate and the demeaned log engagement.
4. Create features:
   - textual features such as caption length, hashtag count, mention count and call to action flags,
   - temporal features such as hour of day, day of week and time of day buckets,
   - historical features such as time since previous post, posts per week and rolling engagement,
   - account features such as follower counts and historical engagement.
5. Construct binary labels:

- rank posts by demeaned log engagement within each country, platform and sector cohort,
- assign high engagement labels based on top quartile or top decile thresholds.

6. Save the resulting feature label tables:
   - as serialised files such as parquet or pickle,
   - optionally in a dedicated schema in PostgreSQL.

3.2 Model training and evaluation

Model training and evaluation are performed in the same notebooks such as `EDA_and_ML_intagram.ipynb`, `EDA_and_ML_tiktok.ipynb` and `EDA_and_ML_facebook.ipynb`.

For each cohort:

1. Load the feature label table.
2. Sort posts chronologically and create a temporal train and test split.
3. Initialise:
   - a Random Forest classifier with suitable hyperparameters and class weights,
   - an XGBoost classifier with tuned depth, learning rate, number of trees and positive class scaling.
4. Fit both models on the training set.
5. Evaluate on the test set using:
   - area under the ROC curve (AUC),
   - average precision,
   - top decile lift.
6. Extract feature importance scores from each model.

Outputs include:

- metric tables per cohort and model,
- serialised model objects for reuse,
- plots such as:
  - AUC and top decile lift by sector and platform,
  - AUC by country and platform,
  - feature importance bar charts,
  - timing heatmaps and caption length plots.

3.3 Reproducing figures and tables

To reproduce thesis figures and tables:

1. Run the ingestion scripts to populate the `accounts` and `posts` tables.
2. Run the feature engineering notebooks to regenerate feature label tables.
3. Run the modelling notebooks to recompute metrics, feature importance scores and evaluation plots.

Each notebook is structured to save:

- tables that can be exported as LaTeX or pasted into a word processor,
- figures as image files with descriptive names, for example:
  - `auc_test_scores_by_sector_platform.png`,
  - `topdecilelift_by_sector_platform.png`,
  - `ie_ng_beauty_instagram_timing_heatmap.png`.

As long as the environment, database and Apify inputs match the configuration described above, another user should be able to reproduce the main analytical outputs.

References

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 785–794.

Harris, C. R. et al. (2020), 'Array programming with numpy', *Nature* **585**(7825), 357–362.

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', *Computing in Science & Engineering* **9**(3), 90–95.

Kluyver, T. et al. (2016), Jupyter notebooks: A publishing format for reproducible computational workflows, *in* 'Positioning and Power in Academic Publishing: Players, Agents and Agendas', IOS Press, pp. 87–90.

McKinney, W. (2010), 'Data structures for statistical computing in python', *Proceedings of the 9th Python in Science Conference* pp. 51–56.

Pedregosa, F. et al. (2011), 'Scikit-learn: Machine learning in python', *Journal of Machine Learning Research* **12**, 2825–2830.

PostgreSQL Global Development Group (2023), *PostgreSQL Documentation*, PostgreSQL Global Development Group.