National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Lakshmi Meena Manivannan
Student ID: x23426918

School of Computing
National College of Ireland

Supervisor: Prof Sallar Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Lakshmi Meena Manivannan

Student ID: x23426918

Programme: MSc Data Analytics **Year:** 2025 - 2026

Module: MSc (Research) Practicum/Internship

Submission Due Date: 28/01/2026

Project Title: Green AI: Quantifying the Carbon Footprint of Machine Learning Models Through Multi-Model Energy Benchmarking and Emissions Analysis

Word Count: 1648 **Page Count: 9**

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: Lakshmi Meena Manivannan

Date: 27/01/2026

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 a 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

Lakshmi Meena Manivannan
Student ID: x23426918

1 Abstract

This installation guide gives a detailed instruction on how to install, configure, and run the GreenAI project to Measuring the Carbon Footprint of Machine Learning Models. The project compares various machine learning algorithms based on their energy usage, run time and their approximate carbon footprint during training and tuning. To benchmark the various supervised and unsupervised models, two datasets are involved, namely, Air Quality and Electricity Consumption, and sustainability is assessed with the aid of energy and carbon tracking tools. The following manual describes every hardware and software environment, data set preparation, environment configuration, feature engineering, code, and instructions to execute the model and reproduce the experimental results. It will be designed to assist the users, researchers and the reviewers to effectively execute the project and learn the operational environments required to ensure the consistency and reproduction of results.

2 System Requirements

- Operating System: Windows, Mac, or Linux.
- Processor: Intel Core i5 8th Gen
- RAM: At least 8GB (16GB preferred).
- Disk Space: Free space that is retrievable and easily accessible, and this should be a minimum of 5GB.
- Internet Connection: During downloading of pre-trained models and datasets.

3 Software Requirements

- Python Version: Python 3.10 or higher.
- Jupyter Notebook or Google colab.
- Libraries: Pandas, Numpy, Scikit-Learn, XGBoost, Plotly and CodeCarbon
- Tools: StreamLit
- Dataset Requirement:

  The datasets used in this project consist of two real-world environmental datasets that support the analysis of energy-aware machine-learning models.

  Air Quality Dataset (UCI Repository): This dataset contains hourly measurements from chemical sensors used to predict air-quality metrics.

Dataset link:
https://archive.ics.uci.edu/ml/datasets/Air+Quality

Electricity Consumption Dataset: This dataset contains electricity-meter readings used for time-series forecasting and regression modelling.
Dataset link (Kaggle):
https://www.kaggle.com/datasets/robbiemccullough/electricity-consumption-data

Both datasets were downloaded in CSV format and imported into Python for preprocessing, feature engineering, model training, carbon-emissions tracking, and evaluation.

4 Installation Guide

4.1 Python Installation

Install Python (via Anaconda or Python.org) using the official installer suitable for your operating system.

```
!python --version
```

4.2 Required Package Installation

Install the necessary Python packages for running the GreenAI project.

```
!pip install numpy pandas scikit-learn xgboost plotly codecarbon streamlit
```

4.3 Additional Libraries

All remaining libraries used in the project can be installed using the following command:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, mean_absolute_error

from codecarbon import EmissionsTracker
```

Figure 1: Import Commands

4.4 Launching Jupyter Notebook

Start Jupyter Notebook from the configured environment to run the project code.

```
jupyter notebook
```

4.5 Running the Streamlit Dashboard

Execute the dashboard file to launch the web interface.

```
streamlit run app.py
```

5 Data Preparation

5.1 Data Collection and Storage

The project uses two primary datasets: the Air Quality dataset and the Electricity Consumption dataset, both stored in CSV format. A custom function (load_raw_any) is implemented to safely load files with different encodings, separators, or mis-labelled extensions. The datasets are stored locally within the project directory under the /data/ folder for easy access during preprocessing and modeling.

5.2 Exploratory Data Analysis (EDA)

Initial exploration includes checking dataset shapes, missing values, summary statistics, and distributions of key numerical variables. Visual inspection is performed using Seaborn and Matplotlib, including histograms, pairplots, and correlation heatmaps. This step helps understand data quality issues, detect outliers, and identify early feature patterns relevant to emissions and energy modelling. A custom Carbon Tracker utility is implemented using CodeCarbon's EmissionsTracker to measure CO_2 emissions and runtime for any model training or evaluation block. The project defines wrapper functions (carbon_tracker and run_with_carbon) that automatically log emissions, CPU/GPU energy usage, and execution time for each experiment. All carbon metrics are saved into the /results/ directory, enabling consistent comparison of model sustainability throughout the pipeline.

```python
@contextmanager
def carbon_tracker(run_name: str, output_dir: str = "results"):
    """
    Track carbon emissions for any code block.
    Example:
        with carbon_tracker("train_model"):
            model.fit(X, y)
    """
    tracker = EmissionsTracker(project_name=run_name, output_dir=output_dir, save_to_file=True)
    tracker.start()
    try:
        yield
    finally:
        emissions = tracker.stop()
        print(f"[CodeCarbon] {run_name} CO2(kg): {emissions}")


def run_with_carbon(run_name: str, func, *args, **kwargs):
    """
    Run a function while measuring CO2 emissions and time.
    Returns (result, co2_kg, seconds)
    """
    tracker = EmissionsTracker(project_name=run_name, output_dir="results", save_to_file=True)
    tracker.start()
    t0 = time.time()
    try:
        result = func(*args, **kwargs)
    finally:
        co2 = tracker.stop()
    secs = time.time() - t0
    print(f"[CodeCarbon] {run_name} CO2(kg): {co2}, time(s): {secs:.3f}")
    return result, (co2 if co2 is not None else 0.0), secs
```

Figure 2: Utility Carbon Tracker

5.3 Data Pre-processing and Feature Engineering

Data preprocessing involves handling missing values, correcting encodings, and standardising formats using Pandas utilities. Time-related fields are parsed to extract meaningful components, and numeric features are normalised for model stability. Advanced feature engineering is applied separately for each dataset, including lag features (electricity), sensor-level cleaning (air quality), and creation of task-specific target variables. A unified structure is produced for model training and carbon tracking.

```python
def load_raw_any(path):
    """
    Load AirQualityUCI.csv safely as:
    - Excel (.xlsx disguised as .csv)
    - CSV with unknown encoding
    - CSV with wrong separators
    """
    path = Path(path)

    try:
        df = pd.read_excel(path, engine="openpyxl")
        print("Loaded as EXCEL file")
        return df
    except:
        pass

    try:
        df = pd.read_csv(path, sep=None, engine="python", encoding="latin1")
        print("Loaded as CSV (auto-separator, latin1)")
        return df
    except:
        pass

    attempts = [
        ("utf-8", ";"),
        ("utf-8", ","),
        ("latin1", ";"),
        ("latin1", ","),
        ("utf-8-sig", ";"),
        ("utf-16", None),
        ("utf-16", ";"),
    ]

    for enc, sep in attempts:
        try:
            if sep is None:
                df = pd.read_csv(path, encoding=enc, sep=None, engine="python")
            else:
                df = pd.read_csv(path, encoding=enc, sep=sep)
            print(f"Loaded as CSV (encoding={enc}, sep='{sep}')")
            return df
        except:
            continue
```

Figure 3: Data Preprocessing

6 Model Implementation

The implementation phase of the model implied the execution of a sequence of supervised, unsupervised, and regression models on the made datasets to assess the predictive power, efficiency and carbon performance. In the case of Air Quality Classification data, five supervised learning models, which included Logistic Regression, Random Forest, SVM (RBF), MLP Neural Network, and XGBoost models, were used. Each model was trained on cleaned and feature-engineered data, and training was tracked with the help of the CarbonTracker utility to record the execution time, CPU/GPU consumption, and approximated emissions of CO2.

In the case of the Electricity Consumption Regression dataset, Random Forest Regressor, MLP Regressor, and XGBoost Regressor were fitted with lag features on a monthly-aggregated time-series dataset. The guide to hyperparameter tuning was used selectively to enhance performance, but this did not increase the computational load with a significant constraint

The Air Quality dataset was also subject to unsupervised models, such as K-Means, Gaussian Mixture Models (GMM), and Agglomerative Clustering, to discover latent models of pollutants by grouping at pollutant level and patterns of environmental behaviour. It was implemented based on a common pipeline with Python (pandas, numpy, scikit-learn, xgboost) and custom wrappers in Carbon Tracking, to guarantee consistency between measurements.

7 Model Evaluation

The comparison of the adopted models was conducted both in the air-quality categorization task and the electricity consumption prediction task, and outcomes were directly obtained based on the notebook results that were run. In the case of the classification dataset, the Accuracy and F1-score were used to evaluate five supervised algorithms, namely Logistic Regression, Random Forest, SVM (RBF), MLP Neural Network, and XGBoost. In general, each model performed well with the highest predictive power demonstrated by the XGBoost, closely followed by the Logistic Regression and Random Forest models, and with a lower score and a few convergence alerts demonstrated by the MLP model. In the case of the electricity regression data, the performance of the models was measured by the Mean Absolute Error (MAE) with the lowest error being that of the Random Forest Regressor, which means that it had the best predictive power compared to the MLP and XGBoost regressors, whose residual errors were considerably high. K-Means, GMM, and Agglomerative Clustering, as unsupervised models, were also applied to air-quality data, giving results of significant natural clusters of pollutant patterns, and supporting the exploratory interpretation of environmental data. Combined with this evaluation, outcomes prove that tree-based techniques, specifically Random Forest and XGBoost, provide the best trade-off between predictive performance and run time on both datasets in this project.

	Model	Accuracy	F1
0	Logistic Regression	0.896902	0.844480
1	Random Forest	0.933761	0.904025
2	XGBoost	0.933226	0.903772
3	SVM	0.913996	0.873128
4	MLP Neural Network	0.922009	0.884311

Figure 4: Accuracy Analysis

8 Execution of the Code

- Download the project folder containing all scripts and datasets
- Unzip the files into a working directory
- Open the main Python file or Jupyter Notebook (GreenAI.ipynb)
- Update the file paths to match the local dataset locations
- Run each code cell sequentially to generate model outputs and sustainability metrics.

9 Conclusion

This study concludes that integrating GreenAI principles into machine learning workflows provides meaningful insights into both model performance and environmental sustainability. Across all experiments, tree-based models such as Random Forest and XGBoost consistently delivered the strongest predictive results for both air-quality classification and electricity-consumption forecasting. The analysis further demonstrates that several high-accuracy models also incur substantial carbon and energy costs, highlighting the importance of evaluating computational impact alongside predictive accuracy. The findings suggest that adopting energy-efficient modelling practices - supported by tools such as CodeCarbon—can reduce emissions while maintaining comparable performance. This supports the broader goal of developing responsible, low-carbon AI systems. Future work could incorporate real-time monitoring, optimize hyperparameter tuning strategies for energy efficiency, and extend the sustainability analysis to larger and more complex datasets.

References

Kaggle Electricity Consumption Dataset (2025) *Kaggle*. Available at: https://www.kaggle.com/datasets/robbiemccullough/electricity-consumption-data (Accessed: 12 December 2025).

UCI Air Quality Dataset (2004) De Vito, S. et al. *Air Quality Dataset*. UCI Machine Learning Repository. Available at: https://archive.ics.uci.edu/ml/datasets/Air+Quality (Accessed: 12 December 2025).

Strubell, E., Ganesh, A. and McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 3645–3650.

Lacoste, A., Luccioni, A., Schmidt, V. and Dandres, T. (2019). Quantifying the Carbon Emissions of Machine Learning. NeurIPS Workshop on Tackling Climate Change with Machine Learning.