National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Lathifa Jaffer Diang'a
Student ID: x23346671

School of Computing
National College of Ireland

Supervisor: Teerath Kumar

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. ……LATHIFA JAFFER DIANG'A……

Student ID: ……X23346671…….……

Programme: …MSCDAD_B…… **Year:** ……2024/2025……..

Module: ……RESEARCH PRACTICUM…….……

Lecturer: ……TEERATH KUMAR…….……

Submission Due Date: ……15th September 2025…….……

Project Title: ……Configuration Manual…….……

Word Count: ……1717…… **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: ……LATHIFA JAFFER DIANG'A……

Date: ……11th September 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

Lathifa Jaffer Diang'a
Student ID: x23346671

1 Introduction

This configuration manual functions as a guide that details and documents all the necessary steps required to replicate the research project titled AI-based Solar Panel Fault Prediction and Anomaly Detection in Ireland from the environment setup to the preprocessing, model training, and evaluation steps required. The aim of the MSc project is to develop a functioning AI-based framework that is capable of detecting anomalies and predicting faults in solar panel systems through leveraging weather and sensor data and climate variables. The system utilizes several methods to achieve proactive maintenance insights, such as unsupervised learning techniques (Isolation Forest) and various supervised techniques using regression models (e.g., Random Forest, NuSVR, Gradient Boosting, ExtraTrees). Exploratory Data Analysis (EDA) and interpretation is supported through standard visualisation tools and the LazyPredict library is used for rapid benchmarking across all the applied models. Therefore, this guide ensures result reproducibility and research transparency as it enables the replication of all the necessary steps, as well as providing guidelines for risk handling and runtime expectations.

2 System Requirements

2.1 Hardware Requirements

Component	Specification
Operating System	macOS Monterey Version 12.7.6 and higher
Processor	Minimum multi-core processor – Intel Core (i5 minimum)
RAM	Minimum 8GB, 16GB+ recommended
Storage	At least 15GB of free disk space (for temporary files, datasets etc)
Internet	Required for downloading datasets and dependencies

2.2 Software Requirements

Tool	Version
Operating system	MacOS (Monterey 12.7.6), WindowsOS (Windows 11+), Ubuntu 22.04
Python	3.10 or higher
Anaconda Navigator	2023 or higher (Launch Jupyter Notebook included in Anaconda)
VS Code	Latest
Web Browser	Chrome, Mozilla or Safari

3 Installation and Setup

3.1 Install required libraries

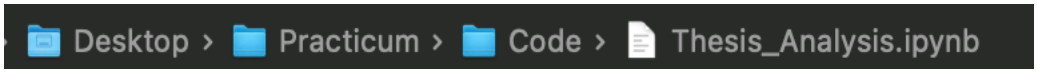
Python libraries are used for visualization, evaluation, EDA, model training and anomaly detection, some of these are listed with their functions:

1. Data manipulation and numerical operations – `pandas, numpy`
2. Visualization – `matplotlib, seaborn`
3. Machine learning models – `scikit-learn` (e.g. MinMaxScaler, IsolationForest, GradientBoostingRegressor, RandomForestRegressor)
4. Automated model evaluation – `lazypredict`

The required python libraries are installed using the terminal or on Anaconda prompt using '!pip install'.

3.2 Folder Organisation

The project follows the specific directory structure that is provided below:




```
Practicum/
├── Code/
│   ├── Thesis_Analysis.ipynb
│   └── Thesis_Analysis.html
├── Datasets/
│   ├── Dublin_Airport.json
│   ├── Portugal_Zenodo.csv
│   ├── MetEireann.csv
│   └── pvwatts_monthly.csv
└── Results Lazy Predict/
    ├── lazypredict_results_pvwatts.csv
    ├── lazypredict_results_portugal.csv
    └── lazypredict_results_meterirann.csv
```

The file paths used in the replicated analysis should be adjusted based on the local directory structure of the user. This ensures that there is a consistent environment for testing and executing the solar panel analysis models effectively. Thus:

1. Launch the Jupyter Notebook via Anaconda Navigator or Visual Studio code.
2. Navigate to the Code folder.
3. Open Thesis_Analysis.ipynb.
4. Run each cell sequentially from top to bottom.
5. Output plots and evaluation results will display inline and can be saved.

4 Dataset Sources and Preprocessing Summary

4.1 Dataset Overview

Dataset	Description	Train/Test Split
Met Éireann	Historical weather and solar radiation data	70% / 30%
PVWatts (NREL)	Local simulated solar energy output (Dublin)	70% / 30%
Portugal Solar (Zenodo)	IoT sensor and inverter solar panel data	70% / 30%
Dublin Airport	Daily solar irradiance, temperature and rainfall data	70% / 30%

4.2 Library Import Configuration

The following libraries need to be imported for data preprocessing.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
import lazypredict
from lazypredict.Supervised import LazyRegressor
import statsmodels.api as sm
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import IsolationForest
from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer
from pandas import json_normalize
from sklearn.ensemble import IsolationForest
import json
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, r2_score
```

4.3 Preprocessing Steps

Data preprocessing is crucial in the preparation of the datasets for analysis. Each of the datasets followed a standard preprocessing pipeline which included the following steps:

1. Loading: The data was loaded and imported using `pandas.read_csv()'
2. Cleaning: Irrelevant features and missing values are removed by imputing and dropping.
3. Date Parsing : All the timestamp columns are converted into datetime objects.
4. Normalization: StandardScaler and MinMaxScaler are applied for regression inputs.
5. Feature Engineering: This was applied to features such as temperature ratio, irradiance deltas, monthly aggregations etc.
6. Train-Test Split: Each dataset is split into a 70/30 ratio using the `train_test_split()'

4.4 Dataset Configuration preprocessing

1. **Met Éireann dataset:** The sunshine hours are converted to kWh/m²/day using division by 41.67.
2. **PVWatts dataset:** The missing values are handled using the forward fill method, the irradiance values are normalized for model compatibility, the efficiency ratios

between DC and AC output are calculated, and the irradiance values are normalized for model compatibility

3. **Portugal Dataset:** Sample 5% of data are sampled to ensure efficient processing as the dataset is large, forward fill is used to handle missing values for time series continuity, and feature engineering columns are created ('current_voltage_ratio' and 'temp_derating')

4.5 Exploratory Data Analysis (EDA)

EDA is performed separately for each dataset. The insights derived from the EDA guided fault detection design and feature selection. Moreover, the plots generated from the analysis include correlation heatmaps, distribution plots for the numerical columns, boxplots for outliers and seasonality insights, time series plots for irradiance and solar output as well as monthly trendlines and aggregations. A code snippet from the EDA performed on the PVWatts dataset is shown in the code below:

```
# Import necessary libraries
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

# Step 1: Define features and target
X = df[['POA_Irradiance']]  # Corrected column name
y = df['AC_Output']  # Target variable

# Step 2: Normalize the features using MinMaxScaler
scaler_X = MinMaxScaler()
X_normalized = scaler_X.fit_transform(X)

# Step 3: Normalize the target variable using MinMaxScaler
scaler_y = MinMaxScaler()
y_normalized = scaler_y.fit_transform(y.values.reshape(-1, 1)).ravel()

# Step 4: Split the normalized data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_normalized, y_normalized, test_size=0.3, random_state=42)

# Step 5: Use LazyRegressor
from lazypredict.Supervised import LazyRegressor
regressor = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric=None)
models, predictions = regressor.fit(X_train, X_test, y_train, y_test)

# Step 6: Display results
print(models)
```

5 Model Training Configuration

The model training configuration involves the setting up of several regression models as well as the evaluation of several performance metrics for a comprehensive analysis.

5.1 LazyPredict Benchmarking

The LazyPredict library is used to present the LazyRegressor class utilized to benchmark over forty regressor models on the Met Éireann, PVWatts and the Portugal solar dataset. Metrics recorded after the models are applied include the R^2 Score, Root Mean Squared Error (RMSE) and Training Time. Additionally, the top ten best performing models from each dataset are extracted and presented in tables and bar charts to guide model choice. Some of the applied models include – RandomForestRegressor, ExtraTreesRegressor., NuSVR, XGBoost, GradientBoostingRegressor.

```
import lazypredict

# Import necessary libraries
from lazypredict.Supervised import LazyRegressor
from sklearn.model_selection import train_test_split

# Step 3: Use the defined features and target
features = ['meant', 'maxtp', 'mintp', 'rain', 'wdsp']
X = df[features]
y = df['sun_kWh']  # Corrected target

# Step 4: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 5: Use LazyRegressor
regressor = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric=None)
models, predictions = regressor.fit(X_train, X_test, y_train, y_test)

# Step 6: Display results
print(models)
```

The test size is set to 0.3 for consistent training and testing splits, and a random state of 42 is used to present reproducible results.

5.2 Evaluation Metrics

Each of the regression model is evaluated on the RMSE (Root Mean Squared Error) and the R^2 Score. Other evaluation metrics applied included visualisations made such as Residual Distributions plots, Actual vs Predicted Plots, and Feature Importance plots. A visualisation example is shown below:

```
# Visualize the results using a bar chart
import matplotlib.pyplot as plt
import seaborn as sns

# Plot R-Squared values
plt.figure(figsize=(10, 6))
sns.barplot(x=models.index, y=models['R-Squared'], palette='viridis')
plt.xticks(rotation=90)
plt.title('R-Squared Values for Models')
plt.ylabel('R-Squared')
plt.xlabel('Models')
plt.tight_layout()
plt.show()

# Plot RMSE values
plt.figure(figsize=(10, 6))
sns.barplot(x=models.index, y=models['RMSE'], palette='magma')
plt.xticks(rotation=90)
plt.title('RMSE Values for Models')
plt.ylabel('RMSE')
plt.xlabel('Models')
plt.tight_layout()
plt.show()
```

5.3 Anomaly Detection

Isolation Forest is used to flag data points significantly deviating from patterns (Portugal, Dublin datasets). Scaled features are fitted using Isolation Forest, then data points are classified using anomaly flags (-1 = anomaly) – thus anomalous months like December 2024 and April 2025 are logged in the reports. The code snippet is shown below:

```
from sklearn.ensemble import IsolationForest

clf = IsolationForest(contamination=0.25)  # Expect ~25% anomalies
df['Anomaly_Score'] = clf.fit_predict(df[['AC_Output', 'POA_Irradiance']])
df['Anomaly_Flag'] = (df['Anomaly_Score'] == -1).astype(int)
```

5.4 Visualisation Configuration

It is important to configure the visualisation settings for a professional and consistent output across all the analysis – hence, matplotlib and seaborn are used.

5.5 Threshold Configuration

Seasonal thresholds are configured as follows:

```
# Dynamic thresholds by season
def flag_issues(row):
    if row['date'].month in [12, 1, 2]:  # Winter
        return int(row['solar_daily'] < 1.5 or row['total_rainfall'] > 80)
    elif row['date'].month in [6, 7, 8]:  # Summer
        return int(row['performance_ratio'] < 0.6)
    else:
        return int((row['solar_daily'] < 2) & (row['mean_temperature'] < 5))
```

These thresholds when reached trigger an alert system for maintenance, that is also visualised as a scatter plot for easier understanding.

```
weather_df['maintenance_alert'] = weather_df.apply(flag_issues, axis=1)

# Displaying the first few rows of the DataFrame
print(weather_df[['date', 'solar_daily', 'total_rainfall', 'performance_ratio', 'mean_temperature', 'maintenance_alert']].head())


# Extracting the month from the date column
weather_df['month'] = weather_df['date'].dt.month


# Scatter plot of solar_daily vs total_rainfall, colored by maintenance_alert
plt.figure(figsize=(10, 6))
plt.scatter(weather_df['solar_daily'], weather_df['total_rainfall'], c=weather_df['maintenance_alert'], cmap='coolwarm', s=50)
plt.title('Maintenance Alerts (Colored by Alert Status)')
plt.xlabel('Solar Daily (kWh)')
plt.ylabel('Total Rainfall (mm)')
plt.colorbar(label='Maintenance Alert (1 = Alert, 0 = No Alert)')
plt.show()
```

6 Output Configuration

To ensure consistency it is important to consider small tips that come handy such as – maintaining all the datasets in one folder and running the cells in the correct order as presented in the notebook. Thus, this configuration manual provides the requirements and the complete setup for replicating the analysis, which ensures consistent results across different environments.

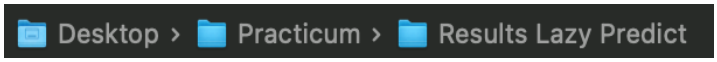
6.1 CSV Output Configuration

The model results are saved into a CSV file as shown:

```
# Saving results into a CSV file
models.to_csv('/Users/Latifa/Downloads/lazypredict_results.csv', index=True)
```

6.2 Results Directory Structure

To ensure consistency it is important to consider small tips that come handy such as – maintaining all the datasets in one folder and running the cells in the correct order as presented in the notebook.



Results Lazy Predict/
├── lazypredict_results_pvwatts.csv
├── lazypredict_results_portugal.csv
└── lazypredict_results_meterirann.csv

6.3 Risk and Rollback

If any of the datasets fails to validate or the metrics degrade by 20% or more, rerun the code using the last saved results in 'Results Lazy Predict/' and flag in the run log.

Risk	Impact	Mitigation	Rollback Strategy
Missing dataset	Notebook halts its run on the 'read_csv/json' calls	Ensure the datasets are stored in '/Practicum/Datasets/' with their correct names	Redownload the dataset
Path misconfiguration	FileNotFoundError during data load	Update the file paths in the first code cell to match the local system paths	Reset the paths to the default structure suggested in the manual
Large dataset runtime (Portugal dataset)	Longer LazyPredict execution	Run on a computer with 16GB RAM or more Test with smaller subsets	Restart Kernel Rerun only the final cells
Kernel interruption	Notebook crashes while running	Constantly save your work and run the code section by section	Reload the last checkpoint Restart kernel

6.4 Runtime Profiles

The datasets should take no longer than 5 minutes to run – Met Éireann and PVWatts take between a few seconds to a couple of minutes to run on laptop CPU, while the larger Portugal dataset should take from between a couple of minutes to 5 minutes to run for the full LazyPredict sweep.

References