National College of Ireland

Accurate and Efficient Short-Term Solar Power Forecasting Across Climatic Regions

MSc Research Project

MSc Data Analytics - C

NIHAL RAJ REDDY DANDA

Student ID: x23317612

School of Computing

National College of Ireland

Supervisor: Harshani Nagahamulla

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …NIHAL RAJ REDDY DANDA……

Student ID: …X23317612……

Programm : …DATA ANALYTICS…… **Year:** ……2023……..

Module: …Research Practicum……..…..

Supervisor: …Harshani Nagahamulla…… ……
Submission Due Date: ….15 september 2025……

Project Title: ….Accurate and Efficient Short-Term Solar Power Forecasting Across Climatic Regions……

Word Count: …553…… **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: …nihalreddy……..

Date: ..10/08/2025……

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):	

Accurate and Efficient Short-Term Solar Power Forecasting Across Climatic Regions

Nihal Raj Reddy Danda
Student ID: x23317612

System Infrastructure Requirements

- **Platform Compatibility**
    - Primary: Windows 10 or Windows 11 for maximum compatibility with tools and libraries
    - Alternative: Ubuntu 20.04 LTS or newer for open-source or server-oriented setups
- **Processing Framework**
    - Minimum: Multi-core CPU (Intel Core i5 / AMD Ryzen 5 or newer)
    - Recommended: Quad-core or higher for faster training and data processing
- **System Memory**
    - Minimum: 16 GB RAM for general execution and testing
    - Recommended: 32 GB RAM or more for large datasets and multiple experiments
- **Hardware Acceleration (Optional but Recommended)**
    - NVIDIA GPU with CUDA support for improved deep learning performance
- **Storage Requirements**
    - 15–20 GB free space to store datasets, model artifacts, embeddings, visualizations, and dependencies

Software Environment Setup

- **Programming Language**

- Python version 3.9 or higher preferred

- **Core Data Analysis Libraries**
  - pandas — data handling and transformations (pip install pandas)
  - numpy — numerical computations (pip install numpy)
  - matplotlib — static data visualizations (pip install matplotlib)
  - seaborn — statistical data visualizations (pip install seaborn)
  - plotly — interactive visualizations (optional) (pip install plotly)
  - scikit-learn — preprocessing, baseline models, evaluation (pip install scikitlearn)

- **Neural Network & Deep Learning Libraries**
  - torch (PyTorch) — model building and training (pip install torch)

Introduction

This configuration manual outlines the step-by-step process for setting up, training, validating, and deploying the **GRU + Attention** solar forecasting model described in the report. It is structured to ensure **reproducibility**, **consistency in model naming**, and **alignment with the experimental methodology** in the thesis.

Dataset Configuration

Data Source: Hourly solar power and environmental variables from Norway, Brazil, India, and Australia.
Data Format: CSV files, with the following required fields:

- **DC_Power** (Target Variable)
- AC_Power
- Ambient_Temperature
- Module_Temperature
- Irradiation
- Daily_Yield
- Total_Yield
- Country (categorical)
- Date (converted to datetime for feature extraction)

Preprocessing Steps:

- **Datetime Conversion:** Convert "Date" to datetime type, extract day, month, year, and hour.

- **Outlier Capping:** Apply **IQR-based capping** to Ambient_Temperature, Module_Temperature, and Daily_Yield.
- **Scaling:** Apply Min-Max normalization for all numerical features.
- **Feature Selection:** Retain only relevant columns as per feature importance ranking.

Model Configuration

Model Names for Consistency:

- **ANN** → Artificial Neural Network
- **GRU** → Gated Recurrent Unit
- **GRU+ATT** → Gated Recurrent Unit with Attention

Baseline Models:

- **XGBoost (XGB)**
- **Random Forest Regressor (RFR)**
- **LightGBM (LGBM)**

Random Forest

```
[ ] from sklearn.ensemble import RandomForestRegressor
    from sklearn.metrics import mean_absolute_error, mean_squared_error
    import matplotlib.pyplot as plt
    import numpy as np

    # Define estimator values
    estimator_values = [2, 8, 20, 40, 60]

    # Lists to hold errors
    mae_list = []
    mse_list = []
    rmse_list = []

    print("Random Forest Errors for Different n_estimators:\n")
    for n in estimator_values:
        rf = RandomForestRegressor(n_estimators=n, max_depth=2, random_state=42)
        rf.fit(X_train, y_train)
        y_pred_rf = rf.predict(X_test)

        mae = mean_absolute_error(y_test, y_pred_rf)
        mse = mean_squared_error(y_test, y_pred_rf)
        rmse = np.sqrt(mse)

        mae_list.append(mae)
        mse_list.append(mse)
        rmse_list.append(rmse)

        print(f"n_estimators = {n}: MSE = {mse:.4f}, MAE = {mae:.4f},   RMSE = {rmse:.4f}")
```

- **LightGBM (LGBM)**

lightgbm

```
[ ]  from lightgbm import LGBMRegressor
     from sklearn.metrics import mean_absolute_error, mean_squared_error
     import matplotlib.pyplot as plt
     import numpy as np
     import warnings

     warnings.simplefilter(action='ignore')

     # Define estimator values
     estimator_values = [40, 60, 80, 100, 120, 130, 140]

     # Lists to store metrics
     mae_list = []
     mse_list = []
     rmse_list = []

     # Training and collecting metrics
     for n in estimator_values:
         lgbm = LGBMRegressor(
             n_estimators=n,
             learning_rate=0.01,
             max_depth=2,
             random_state=42
         )
         lgbm.fit(X_train, y_train)
         y_pred = lgbm.predict(X_test)

         mae = mean_absolute_error(y_test, y_pred)
         mse = mean_squared_error(y_test, y_pred)
         rmse = np.sqrt(mse)
```

Model Architecture – GRU + Attention

```
# --- Imports ---
import os
import random
import numpy as np
import tensorflow as tf
from sklearn.metrics import mean_squared_error, mean_absolute_error
from tensorflow.keras.layers import Input, GRU, Attention, Dense, GlobalAveragePooling1D
from tensorflow.keras.models import Model

# --- Set seeds for reproducibility ---
seed = 42
os.environ['PYTHONHASHSEED'] = str(seed)
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)
os.environ['TF_DETERMINISTIC_OPS'] = '1'

# --- Build GRU + Attention Model ---
inputs = Input(shape=(1, X_train_scaled.shape[1]))
gru_out = GRU(64, return_sequences=True)(inputs)
attention = Attention()([gru_out, gru_out])
context_vector = GlobalAveragePooling1D()(attention)
output = Dense(1)(context_vector)

attn_model = Model(inputs=inputs, outputs=output)
attn_model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# --- Train the Model ---
ahistory = attn_model.fit(
    X_train_seq,
    y_train_scaled,
    epochs=20,
    batch_size=32,
    validation_split=0.2,
    verbose=1)
```

- **Layer 1:** GRU Layer ○ Units: 64 ○ Activation: tanh
  - ○ Return sequences: True

- **Layer 2:** Attention Layer ○ Mechanism: Context-based dot-product attention ○ Output: Weighted sequence representation
- **Layer 3:** Dense Layer (Fully Connected) ○ Units: 1 (for DC Power prediction) ○ Activation: linear

Training and Validation

```
GRU + Attention Model Performance:
Mean Squared Error (MSE): 0.5387
Mean Absolute Error (MAE): 0.5379
Root Mean Squared Error (RMSE): 0.7340
```

- **Training-Validation Split:** 80%-20% • **Evaluation Metrics:** ○ MAE ○ MSE ○ RMSE
- **Best Model** ○ GRU+ATT → MAE: **0.53**, MSE: **0.53**, RMSE: **0.73**

References

Core Libraries & Frameworks

- **Python Documentation** — https://docs.python.org/3/
- **NumPy Reference Guide** — https://numpy.org/doc/
- **Pandas Documentation** — https://pandas.pydata.org/docs/
- **Scikit-learn User Guide** — https://scikit-learn.org/stable/documentation.html

Deep Learning Frameworks

- **PyTorch API Reference** — https://pytorch.org/docs/stable/index.html
- **TensorFlow (Keras API)** — https://www.tensorflow.org/api_docs/python/tf/keras

Ensemble & Gradient Boosting

- **LightGBM Official Docs** — https://lightgbm.readthedocs.io/en/latest/
- **XGBoost Documentation** — https://xgboost.readthedocs.io/en/stable/

Visualization Tools

- **Matplotlib User Guide** — https://matplotlib.org/stable/users/index.html
- **Seaborn Documentation** — https://seaborn.pydata.org/

Development & Experimentation

- **Jupyter Notebook Docs** — https://jupyter.org/documentation