National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Etinosa Eghaghe
Student ID: x23138548

School of Computing
National College of Ireland

Supervisor: Naushad Alam

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Etinosa Eghaghe
Student ID:	x23138548
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Naushad Alam
Submission Due Date:	16/09/2024
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	6

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:	Etinosa Eghaghe
Date:	16th September 2024

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

Etinosa Eghaghe
x23138548

INTRODUCTION

This manual gives step-by-step instruction for setting up and running power consumption prediction models using various machine learning techniques. The model developed include Decision Tree(DT), Random Forest(RF), Long Short-Term Memory(LSTM), and XGBoost regressors. This manual is divided into four section which are: System Specification, Preparation and Preprocessing of the Data Used, Data Visualization Techniques, Model Training, Evaluation, and Feature Importance

1 System Specification

System Information	Details
OS Name	Microsoft Windows 10 Pro
Version	10.0.19045 Build 19045
OS Manufacturer	Microsoft Corporation

Table 1: Operating System Specifications

1.1 Hardware Specifications:

System Information	Details
System Manufacturer	HP
System Model	HP EliteBook 850 G3
System Type	x64-based PC
Processor	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 2496 Mhz, 2 Core(s), 4 Logical Processor(s)
Installed Physical Memory (RAM)	16.0 GB
Total Physical Memory	15.9 GB
Available Physical Memory	5.71 GB (at the time of recording, might fluctuate)
Total Virtual Memory	24.9 GB
Available Virtual Memory	10.3 GB (at the time of recording, might fluctuate)
Page File Space	9.00 GB

Table 2: System Specifications

1.2 Python/ Jupyter note book Version Used:

- Python version:

  import sys

  print(sys.version):

  3.11.0 (main, Oct 24 2022, 18:26:48)[MSC v.1933 64 bit (AMD64)]

- Jupyter notebook(!jupyter –version): Selected Jupyter core packages... IPython : 8.20.0 ipykernel : 6.29.0 ipywidgets : not installed jupyter_client : 8.6.0 jupyter_core : 5.7.1 jupyter_server : 2.12.5 jupyterlab : 4.0.11 nbclient : 0.9.0 nbconvert : 7.14.2 nbformat : 5.9.2 notebook : 7.0.7 qtconsole : not installed traitlets : 5.14.1

2 Preparation and Preprocessing of the Data Used

This is the beginning of the processes, it provides detail instruction for preparing and preprocessing the dataset. A good prepared data ensures the model will perform well.

2.1 Importing Libraries

Begin by importing necessary libraries needed. These include tools for data manipulation, visualization, machine learning and evaluation. libraries and tools used are in Figure 1 below:

```
# imptortation of neccessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import StratifiedShuffleSplit, cross_val_score, train_test_split,GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import statsmodels.api as sm
import matplotlib.dates as mdates
# For Decision tree
from sklearn.tree import DecisionTreeRegressor
# for Random Forest
from sklearn.ensemble import RandomForestRegressor
# FOR LSTM
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input
from tensorflow.keras.callbacks import EarlyStopping
 # FOR XGBoost
from xgboost import XGBRegressor
```

Figure 1: Necessary Libraries

2.2 Loading of the Dataset

Next, is to load the dataset. Make sure it's correctly loaded into a DataFrame for further processing. The dataset used is in CSV file, load from the device into DataFrame in jupyter notebook for further processing. By Data = pd.read_csv("power_consumption.csv")

2.3 Exploring the Data

Time is taking to explore the data. Look at its structure properly, data types and summary statistics. This step helps in understanding the dataset. The dataset is explored using print(Data.describe()),print(Data.shape)print(Data.dtypes) and print(Data.info())

2.4 Cleaning/Preprocessing of the Data

At this point the data is thoroughly cleaning. This include checking for missing values and handling them. However the data used in this study has no missing value. Used this to check for missing values: missing_counts = Data.isna().sum() and print()

Rename some columns for better readability.

```python
# create a dictionary to map old column names to new column names that you want
column_mapping = {
    'Zone 1 Power Consumption': 'Zone1_Consumption',
    'Zone 2  Power Consumption': 'Zone2_Consumption',
    'Zone 3  Power Consumption': 'Zone3_Consumption',
}

# To rename the columms use the rename() methon
Data = Data.rename(columns=column_mapping)

# print the data to verify the renaming of columns
print("Renamed columns:")
print(Data.columns)
```

Figure 2: columns Renamed

Convert dateTime column to appropriate datetime type.

Set the datetime columns as the index of the DataFrame.

Create additional features : hour, day and month from the datetime index. This process is called feature engineering.

```python
# Datetime column is converted to datetime type suitable for the analysis
Data['DateTime'] = pd.to_datetime(Data['DateTime'])
```

```python
# Set Datetime as the index
Data.set_index('DateTime', inplace=True)
```

```python
# Feature engineering: Convert DateTime to numerical features if necessary
Data['Hour'] = Data.index.hour
Data['Day'] = Data.index.day
Data['Month'] = Data.index.month
```

Figure 3: Cleaning/Preprocessing

3 The Data Visualization

Good visualization make it easier to understand the dataset. at this stage the cleaned data is visualized to identify patterns, trends and any anomalies. use Line charts plot, Histograms,Box plots, and Heatmaps.

3.1 Line charts plot

This helps to identify seasonal variations and anomalies in the data. create line chart to show trends in temperature, humidity, wind speed, and power consumption over time. using:

```python
plt.figure(figsize=(14, 7))
plt.plot(Data.index, Data['Temperature'], label='Temperature')
plt.plot(Data.index, Data['Humidity'], label='Humidity')
plt.plot(Data.index, Data['Wind Speed'], label='Wind Speed')
plt.title('Temperature, Humidity, and Wind Speed Over Time')

# Use AutoDateLocator and DateFormatter for the x-axis
locator = mdates.AutoDateLocator()
formatter = mdates.DateFormatter('%Y-%m-%d')  # Customize the format to suit the plot

plt.gca().xaxis.set_major_locator(locator)
plt.gca().xaxis.set_major_formatter(formatter)

plt.xlabel('DateTime')
plt.ylabel('Values')
plt.legend()
plt.gcf().autofmt_xdate()  # Rotate and align the date labels
plt.show()
```

Figure 4: Line Chart

3.2 Histograms

Use the histagram to display the distribution of the dataset variables. The purpose of this is to understand the range and frequency of these variables, indicating whether the data is skewed, normally distributed, or if there is any outlier.

3.3 Box plots

This is used to visualize the spread and central tendency of the variable in the dataset. This can also by use to highlight any potential outliers.

3.4 Heatmaps

This is generated to display the correlation between different variables in the dataset. This plot help in identifying strong and weak correlations between variables, such as the environmental factors and the power consumption.

4 The Model Training, Evaluation and Feature Importance

This stage details the process of training, evaluating, and understanding the importance of various features. It covers the machine learning algorithms used and their evaluation metrics.

4.1 Decision Tree

Train a Decision Tree Regressor for predicting power consumption in different zones. Evaluate the model using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R2 Score, and Mean Absolute Error (MAE).

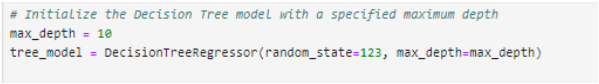

Figure 5: Decision Tree Model

4.2 Random Forest

Train a Random Forest Regressor, which uses multiple decision trees to improve prediction accuracy. Evaluate its performance using similar metrics.

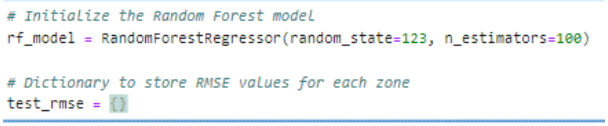

Figure 6: Random Forest Model

4.3 Long Short-Term Memory

Train a Long Short-Term Memory (LSTM) neural network, suitable for time series prediction. Perform feature scaling and reshape the input data appropriately for the LSTM model. Use early stopping to prevent overfitting and evaluate the model's performance on the test set.

```
# Building the LSTM model
model = Sequential()
model.add(Input(shape=(X_train_lstm.shape[1], X_train_lstm.shape[2])))
model.add(LSTM(units=units, return_sequences=True))
model.add(Dropout(dropout_rate))
model.add(LSTM(units=units))
model.add(Dropout(dropout_rate))
model.add(Dense(units=len(target_variables)))

# To Compile the model with mean_squared_error loss for regression tasks
model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
```

Figure 7: LSTM Model

4.4 XGBoost

Train an XGBoost Regressor, which is a powerful boosting algorithm for regression tasks. Perform hyperparameter tuning using GridSearchCV to find the best model parameters. Evaluate the model's performance on the test set.

```
# Function to tune hyperparameters
def tune_hyperparameters(X_train, y_train):
    param_grid = {
        'n_estimators': [100, 200],
        'learning_rate': [0.01, 0.1],
        'max_depth': [5, 7, 9],
        'min_child_weight': [3, 5, 7],
        'subsample': [0.8, 1.0],
        'colsample_bytree': [0.8, 1.0],
    }
    xgb_reg = XGBRegressor()
    grid_search = GridSearchCV(estimator=xgb_reg, param_grid=param_grid, scoring='neg_root_mean_squared_error', cv=5, verbose=1, n_jobs=-1)
    grid_search.fit(X_train, y_train)
    return grid_search.best_estimator_
```

Figure 8: XGBoost

4.5 Feature Importance

Plot the feature importance for each model to understand which features contribute most to the predictions. This helps in interpreting the model and identifying key factors affecting power consumption.

```
# Function to plot feature importance
def plot_feature_importance(importance, names, model_type, save_path):
    # Create a dataframe for plotting
    feature_importance = np.array(importance)
    feature_names = np.array(names)
    data = {'Feature Names': feature_names, 'Importance': feature_importance}
    fi_df = pd.DataFrame(data)

    # Sort the dataframe
    fi_df.sort_values(by=['Importance'], ascending=False, inplace=True)

    # Define size of bar plot
    plt.figure(figsize=(10, 8))
    # Plot the feature importance
    sns.barplot(x=fi_df['Importance'], y=fi_df['Feature Names'])
    plt.title(f'Feature Importance for {model_type}')
    plt.xlabel('Feature Importance')
    plt.ylabel('Feature Names')
    plt.tight_layout()
    plt.savefig(f'{save_path}/{model_type}_feature_importance.png')
    plt.show()

# Plot feature importance for Decision Tree Regressor
for target in target_variables:
    tree_model.fit(X_train, y_train[target])
    plot_feature_importance(tree_model.feature_importances_, X.columns, f'Decision Tree - {target}', save_path)
```

Figure 9: Function for Feature Importance

4.6 Saving and Loading the Models

Save the trained models to disk for future use and load them as needed. This ensures that you can reuse the models without retraining them every time.

4.7 Conclusion

This section provides a comprehensive summary of the steps and techniques used for setting up and running various machine learning models for power consumption prediction. Follow the guidelines carefully to ensure correct implementation and achieve optimal results.