

Configuration Manual

MSc Research Project
MSc Data Analytics

Jismol Thomas -
Student ID: 23414952

School of Computing
National College of Ireland

Supervisor: Harshani Nagahamulla

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Jismol Thomas -
Student ID:	23414952
Programme:	MSc Data Analytics
Year:	2025
Module:	MSc Research Project
Supervisor:	Harshani Nagahamulla
Submission Due Date:	28/01/2026
Project Title:	Multi-Source Deep Learning for TSO-Level Electricity Demand Forecasting
Word Count:	727
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:	<i>Jismol</i>
Date:	27th January 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 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

Jismol Thomas -
23414952

1 Introduction

The purpose of this configuration manual is to give all the directions required to replicate the Multi-Source Deep Learning for TSO-Level Electricity Demand Forecasting research.

2 Hardware Requirements

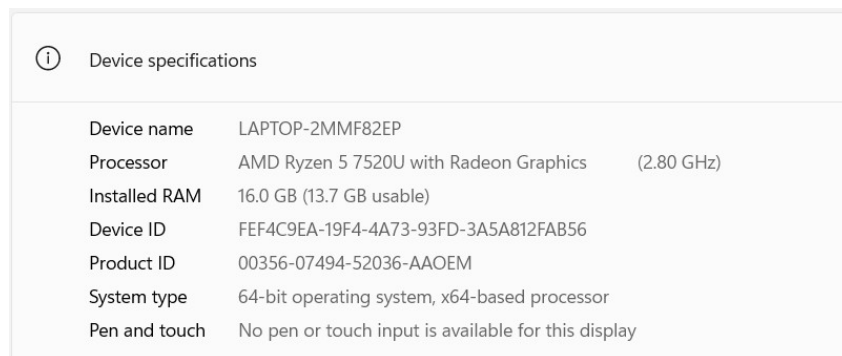
This research was implemented using the following hardware configuration.

2.1 Laptop Specifications

Laptop specifications (Device specifications & Windows Specifications) used in this research are described in Table 1. Figure 1 and Figure 2 to show the same.

Component	Specification
Processor	AMD Ryzen 5 7520U (2.80 GHz)
RAM	16 GB installed (13.7 GB usable)
System Type	64-bit Operating System, x64-based processor

Table 1: Laptop specifications



The image shows a Windows 'Device specifications' window. It contains the following information:

Device name	LAPTOP-2MMF82EP	
Processor	AMD Ryzen 5 7520U with Radeon Graphics	(2.80 GHz)
Installed RAM	16.0 GB (13.7 GB usable)	
Device ID	FEF4C9EA-19F4-4A73-93FD-3A5A812FAB56	
Product ID	00356-07494-52036-AAOEM	
System type	64-bit operating system, x64-based processor	
Pen and touch	No pen or touch input is available for this display	

Figure 1: Device Specifications

Windows specifications		Copy
Edition	Windows 11 Home	
Version	24H2	
Installed on	18/01/2025	
OS build	26100.7171	
Serial number	MP2Q6JV9	
Experience	Windows Feature Experience Pack 1000.26100.265.0	

Figure 2: Windows Specifications

3 Software Requirements

3.1 Jupyter Notebook Configuration

The stages like EDA, preprocessing, merging datasets, and creating the unified feature datasets were carried out using Jupyter Notebook installed on the local machine. The device specifications are shown in Table 2. Details of Jupyter are provided in Figures 3

Component	Configuration
Python Version	Python 3.10.18 (Conda-Forge build)
Platform	Windows 10 (Build 26100), 64-bit (AMD64)
Environment	Jupyter Notebook (local machine)
Usage	Preprocessing, EDA, dataset construction

Table 2: Configuration details of Jupyter Notebook

```
import sys
import platform

print("Python version:", sys.version)
print("Platform:", platform.platform())
```

Python version: 3.10.18 | packaged by conda-forge | (main, Jun 4 2025, 14:42:04) [MSC v.1943 64 bit (AMD64)]
Platform: Windows-10-10.0.26100-SP0

Figure 3: Python and platform information displayed in Jupyter Notebook

3.1.1 Libraries used in Jupyter Notebook

Libraries used include pandas, numpy, matplotlib, seaborn, plotly, scipy, statsmodels, requests, urllib, json, os, sys, glob, pathlib, datetime, joblib, geopandas, shapely, rapidfuzz, unicode, sklearn, tqdm, gc.

The Jupyter Libraries with versions are shown in Figure 4

3.2 Google Colab Configuration

Google Collaboratory is used for the development of all models, hyperparameter tuning, training, and evaluation, because of the computational complexity of deep learning models. The device configuration provided of Colab is summarised in Figures 5 and 6.

```

File Edit View Run Kernel Settings Help Trusted
JupyterLab Python 3 (ipykernel)

requests: requests.__version__,
"sklearn": sklearn.__version__,
"geopandas": geopandas.__version__,
"shapely": shapely.__version__,
"rapidfuzz": rapidfuzz.__version__,
"tqdm": tqdm.__version__,
"python": sys.version
}

libs

[2]: {'pandas': '2.2.3',
      'numpy': '2.1.3',
      'matplotlib': '3.10.0',
      'seaborn': '0.13.2',
      'plotly': '5.24.1',
      'scipy': '1.15.3',
      'statsmodels': '0.14.4',
      'requests': '2.32.3',
      'sklearn': '1.6.1',
      'geopandas': '1.1.1',
      'shapely': '2.1.2',
      'rapidfuzz': '3.14.1',
      'tqdm': '4.67.1',
      'python': '3.13.5 | packaged by Anaconda, Inc. | (main, Jun 12 2025, 16:37:03) [MSC v.1929 64 bit (AMD64)]'}

```

Figure 4: Library versions used in Jupyter Notebook

```

!nvidia-smi

Mon Dec 8 12:56:57 2025

+-----+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
+-----+-----+
| GPU Name                               Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC | |
| Fan  Temp  Perf    Pwr:Usage/Cap       |              |         Memory Usage | GPU-Util  Compute M. |
|                                       |              |         MIG M.       |
+-----+-----+
| 0  NVIDIA A100-SXM4-40GB             Off          | 00000000:00:04:0 Off |             0         |
| N/A   29C   P0              45W / 400W |              | 0MiB / 40960MiB     |    0%      Default  |
+-----+-----+
|                                         |              |         Disabled     |
+-----+-----+

+-----+
| Processes: |
| GPU   GI   CI        PID   Type   Process name                        GPU Memory |
| ID    ID                                     |              | Usage      |
+-----+-----+
| No running processes found |
+-----+

```

Figure 5: Google colab pro+ configuration

```

!python --version
!uname -a

... Python 3.12.12
Linux b6ccce0c66fb 6.6.105+ #1 SMP Thu Oct 2 10:42:05 UTC 2025 x86_64 x86_64 x86_64 GNU/Linux

```

Figure 6: Python version in Google colab pro+

4 Dataset

Multiple sources of data are used in this research to construct a unified dataset for TSO-level electricity demand forecasting. The datasets include:

- TSO electricity consumption, generation and day ahead price data from SMARD.
- Open-meteo API weather data on parameters like temperature, relative humidity, wind speed, shortwave radiation, precipitation, surface pressure.
- Socio-economic indicators such as GDP, Population density from Eurostat.
- Mapping of districts to control zones of German Transmission System Operators (TSOs) dataset from Zenedo.

All datasets were downloaded separately and processed using Jupyter Notebook. The folder structure of the raw datasets is shown in Table 3. Screenshots of Jupyter details are provided in Figures 7 `Electricity_Demand/data`. Below are the files and folders inside the data folder.

File / Folder	Description
<code>weather_raw_tso</code>	Raw Open-Meteo weather data
<code>SMARD_DATA</code>	Consumption, generation, and day ahead price data in seperate folders
<code>demo_r_d3dens_page_linear_2.0.csv</code>	Eurostat population density dataset for NUTS3 regions
<code>nama_10r_3gdp_custom_18511051_linear_2.0.csv</code>	Eurostat GDP dataset for NUTS3 regions
<code>GER_NUTS3_TSOs.*</code>	NUTS3-to-TSO spatial mapping shapefiles

Table 3: Description of dataset files included in the project

After data cleaning and transformation, all processed datasets and final datasets are stored inside processed folder in data folder.

All code files are available in `Electricity_Demand/Code` folder as shown in Figure 8

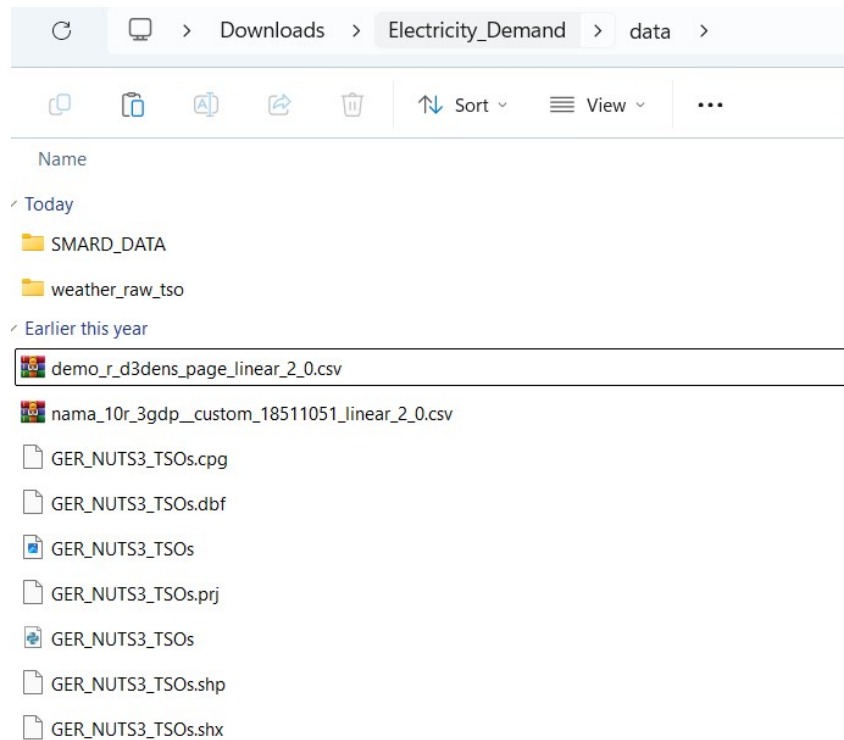


Figure 7: Structure of Data Folder

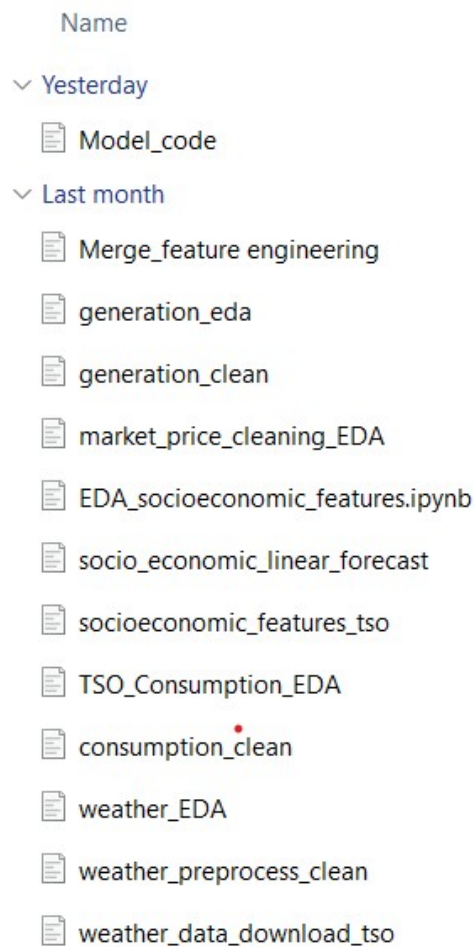


Figure 8: Structure of Code Folder

Using the notebook `Merge_feature_engineering.ipynb`, all processed datasets were merged into a final unified dataset:

This merged dataset is then uploaded to Google Drive to use in Google Colaboratory for modeling purpose.

4.1 Required Libraries

The following libraries must be installed in Colab is shown in Figures 11 and Figures 12

```
!pip install pandas numpy scikit-learn --quiet

!pip install "pytorch-forecasting>1.1.0" "lightning>2.2.0"

Requirement already satisfied: pytorch-forecasting>1.1.0 in /usr/local/lib/python3.12/dist-packages (1.5.0)
Requirement already satisfied: lightning>2.2.0 in /usr/local/lib/python3.12/dist-packages (2.6.0)
Requirement already satisfied: numpy<2.0.0 in /usr/local/lib/python3.12/dist-packages (from pytorch-forecasting>1.1.0) (2.0.2)
Requirement already satisfied: torch>2.0.1, <3.0.0, >=2.0.0 in /usr/local/lib/python3.12/dist-packages (from pytorch-forecasting>1.1.0) (2.0.0+cu126)
Requirement already satisfied: scipy>2.0.0, <1.0 in /usr/local/lib/python3.12/dist-packages (from pytorch-forecasting>1.1.0) (1.16.1)
Requirement already satisfied: pandas<3.0.0, >=1.3.0 in /usr/local/lib/python3.12/dist-packages (from pytorch-forecasting>1.1.0) (2.2.2)
Requirement already satisfied: scikit-learn<2.0.0, >=1.2 in /usr/local/lib/python3.12/dist-packages (from pytorch-forecasting>1.1.0) (1.6.1)
Requirement already satisfied: python-crontab>3.4 in /usr/local/lib/python3.12/dist-packages (from lightning>2.2.0) (6.0.3)
Requirement already satisfied: fsspec[http]>2022.5.0 in /usr/local/lib/python3.12/dist-packages (from lightning>2.2.0) (2025.3.0)
Requirement already satisfied: lightning-utilities<2.0.0, >=0.10.0 in /usr/local/lib/python3.12/dist-packages (from lightning>2.2.0) (0.15.2)
Requirement already satisfied: packaging>20.0 in /usr/local/lib/python3.12/dist-packages (from lightning>2.2.0) (25.0)
Requirement already satisfied: torchmetrics<3.0.0, >=0.7.0 in /usr/local/lib/python3.12/dist-packages (from lightning>2.2.0) (1.8.2)
Requirement already satisfied: tqdm>4.60.0, <4.60.0 in /usr/local/lib/python3.12/dist-packages (from lightning>2.2.0) (4.67.2)
Requirement already satisfied: typing-extensions<4.0.0, >=3.7.4 in /usr/local/lib/python3.12/dist-packages (from lightning>2.2.0) (4.15.0)
Requirement already satisfied: pytorch-lightning in /usr/local/lib/python3.12/dist-packages (from lightning>2.2.0) (2.6.0)
Requirement already satisfied: aiohttp<4.0.0, >=4.0.0a0 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]>2022.5.0->lightning>2.2.0) (3.13.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from lightning-utilities<2.0.0, >=0.10.0->lightning>2.2.0) (75.2.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas<3.0.0, >=1.3.0->pytorch-forecasting>1.1.0) (2.9.0.post0)
Requirement already satisfied: pytz>2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas<3.0.0, >=1.3.0->pytorch-forecasting>1.1.0) (2025.2)
Requirement already satisfied: tzdata>2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas<3.0.0, >=1.3.0->pytorch-forecasting>1.1.0) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=1.2->pytorch-forecasting>1.1.0) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=1.2->pytorch-forecasting>1.1.0) (3.6.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch>2.0.1, <3.0.0, >=2.0.0->pytorch-forecasting>1.1.0) (3.20.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>2.0.1, <3.0.0, >=2.0.0->pytorch-forecasting>1.1.0) (1.14.0)
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch>2.0.1, <3.0.0, >=2.0.0->pytorch-forecasting>1.1.0) (3.6)
```

Figure 9: Pytorch package in Colab

```
!pip install optuna

Requirement already satisfied: optuna in /usr/local/lib/python3.12/dist-packages (4.6.0)
Requirement already satisfied: alembic>1.5.0 in /usr/local/lib/python3.12/dist-packages (from optuna) (1.17.2)
Requirement already satisfied: colorlog in /usr/local/lib/python3.12/dist-packages (from optuna) (6.10.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from optuna) (2.0.2)
Requirement already satisfied: sqlalchemy>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from optuna) (25.0)
Requirement already satisfied: sqlalchemy-chem>=1.4.2 in /usr/local/lib/python3.12/dist-packages (from optuna) (2.0.44)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from optuna) (4.67.1)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.12/dist-packages (from optuna) (6.0.3)
Requirement already satisfied: Mako in /usr/local/lib/python3.12/dist-packages (from alembic>1.5.0->optuna) (1.3.10)
Requirement already satisfied: typing-extensions>=4.12 in /usr/local/lib/python3.12/dist-packages (from alembic>1.5.0->optuna) (4.15.0)
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-packages (from sqlalchemy-chem>=1.4.2->optuna) (3.2.4)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.12/dist-packages (from Mako->alembic>1.5.0->optuna) (3.0.3)
```

Figure 10: Optuna package installed in Colab

```
#Import Libraries
import numpy as np
import pandas as pd
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import torch
from torch.utils.data import Dataset, DataLoader
import torch.nn as nn
import time
from pytorch_forecasting import TimeSeriesDataSet
import lightning.pytorch as pl
from lightning.pytorch.callbacks import EarlyStopping, ModelCheckpoint
from pytorch_forecasting import TemporalFusionTransformer
from pytorch_forecasting.metrics import QuantileLoss
import warnings
```

Figure 11: Libraries needed in Colab

4.2 Colab GPU Configuration

Enable GPU before training:

Runtime → Change runtime type → Hardware accelerator → GPU

```
import optuna
from google.colab import drive
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use("default")
sns.set(style="whitegrid")
```

Figure 12: Libraries needed in Colab

Change runtime type

Runtime type

Python 3 ▼

Hardware accelerator ⓘ

☐ CPU ☒ A100 GPU ☐ L4 GPU ☐ T4 GPU

☐ v6e-1 TPU ☐ v5e-1 TPU

High-RAM ☐

Runtime version ⓘ

Latest (recommended) ▼

Cancel Save

Figure 13: Change Runtime to A100

4.3 Mount Google Drive

The Dataset needs to be uploaded in Google drive. Then mount the google drive with Google colab as shown in Figure 14.

```
from google.colab import drive
```

```
# Mount Google Drive
drive.mount('/content/drive')
```

Mounted at /content/drive

Figure 14: Mount Colab

4.4 Loading the Dataset in Colab

The dataset is loaded in Google Colab as shown in Figure 15.

```
# Define file path
file_path = "/content/drive/MyDrive/Final_Tso_Dataset_Updated.csv"
```

```
# Load dataset
data = pd.read_csv(file_path)
```

```
#Print basic info
print("Shape:", data.shape)
print("Columns:", data.columns.tolist())
data.head()
```

```
Shape: (201596, 65)
Columns: ['consumption_mwh', 'tso', 'timestamp_utc', 'total_generation_mwh', 'total_renewable_mwh', 'total_conventional_mwh', 'biomass_mwh', 'hydropower_mwh', 'wind_offshore_mw']
```

	consumption_mwh	tso	timestamp_utc	total_generation_mwh	total_renewable_mwh	total_conventional_mwh	biomass_mwh	hydropower_mwh	wind_offshore_mw
0	9307.50	50 Hertz	2020-01-01 00:00:00+00:00	13654.75	5432.25	8222.50	1143.00	133.75	639.00
1	9089.50	50 Hertz	2020-01-01 01:00:00+00:00	15117.00	5237.25	9879.75	1149.00	134.25	728.00

Figure 15: File Loaded in Colab

5 Execution Steps

1. Unzip the ICT Solution Artefact-23414952 Folder and extract the Electricity_Demand folder.
2. Run all notebooks **except** `Model_code.ipynb` in the `code` folder (inside the `Electricity_Demand` directory) directly in Jupyter Notebook by clicking *Restart Kernel and Run All*.
3. Run the files in the following order:
 - (a) `weather_data_download_tso.ipynb`
 - (b) `consumption_clean.ipynb`
 - (c) `generation_clean.ipynb`
 - (d) `weather_preprocess_clean.ipynb`
 - (e) `socioeconomic_features_tso.ipynb`
 - (f) `socio_economic_linear_forecast.ipynb`

- (g) `TSO_Consumption_EDA.ipynb`
 - (h) `generation_eda.ipynb`
 - (i) `weather_EDA.ipynb`
 - (j) `market_price_cleaning_EDA.ipynb`
 - (k) `EDA_socioeconomic_features.ipynb`
 - (l) `Merge_feature_engineering.ipynb`
 - (m) `Model_code.ipynb` (Google Colab Pro+)
4. Execute `Model_code.ipynb` in Google Colab Pro+. Prior to the execution of the code, update dataset path to match your Google Drive directory.
 5. Ensure GPU runtime is enabled to avoid slow training.
 6. Click *Run all*.

After this, open the `research_dashboard.pbix` in Power BI. Then upload the combined result CSV file generated from Collab, then click Home → Refresh. Figure 16 is the Power BI Desktop Version used in this research.

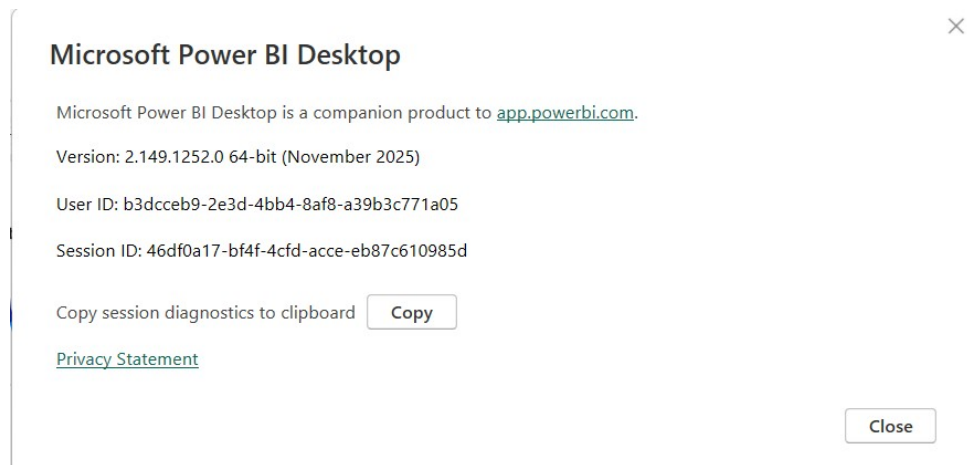


Figure 16: PowerBI Version

These steps will make sure that the entire process of working with raw data until the final model prediction can be reproduced successfully.

References

- SMARD Electricity Data: <https://www.smard.de/en>
 Open -meteo: <https://open-meteo.com/>
 Eurostat Socio-Economic Indicators: <https://ec.europa.eu/eurostat>
 Frysztacki, M.M. (2023) “Mapping of districts to control zones of German Transmission System Operators (TSOs)”. Zenodo. doi:10.5281/zenodo.7530196.