National College of Ireland

A Comparative Analysis of Tree-Based, Deep Learning, and Ensemble Models for Building Energy Forecasting

MSc Practicum
MSc Top-up Artificial Intelligence

Dan Alexandru Bujoreanu
Student ID: x23309903

School of Computing
National College of Ireland

Supervisor: Faithful Chiagoziem ONWUEGBUCHE

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Dan Alexandru Bujoreanu
Student ID:	x23309903
Programme:	MSc Top-up Artificial Intelligence
Year:	2025
Module:	MSc Practicum
Supervisor:	Faithful Chiagoziem ONWUEGBUCHE
Submission Due Date:	11/08/2025
Project Title:	A Comparative Analysis of Tree-Based, Deep Learning, and Ensemble Models for Building Energy Forecasting
Word Count:	n/a
Page Count:	n/a

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.

Signature:	
Date:	11th August 2025

1 Introduction

This configuration manual documents how to replicate the experimental setup/ Jupyter Notebooks reported in the thesis. It covers: hardware/software requirements; environment setup; execution order of notebooks; expected intermediate artefacts; and scripts to regenerate the thesis figures (MAE bar chart, MAE–time trade-off, actual vs. predicted). The pipeline is four-phase and "feature-engineering-first": (i) EDA & consolidation, (ii) feature engineering & selection, (iii) model training & ensembling, and (iv) evaluation & figure export.

2 Hardware Requirements

- **CPU/GPU:** Apple Silicon (M1/M2/M3) with Metal (MPS) acceleration *or* x86_64 with CUDA-capable GPU. CPU-only is acceptable; deep models will train slower.
- **Memory:** $\geq$ 16 GB RAM recommended (dataset $\sim$1.6M rows after merges).
- **Storage:** $\geq$ 10 GB free for intermediate CSVs & model artefacts.

3 Software and Library Dependencies

3.1 Core stack

- **Python** 3.10+ (tested with 3.11).
- **JupyterLab/Notebook** for running the three project notebooks.
- **Data/ML:** `pandas`, `numpy`, `scikit-learn`, `xgboost`, `lightgbm`.
- **Deep Learning:** `tensorflow/keras` for LSTM & CNN–LSTM; `pytorch`, `pytorch-lightning`, `pytorch-forecasting` for TFT.
- **Plotting:** `matplotlib`.

3.2 Environment creation

```
python3 -m venv stbelf_env
source stbelf_env/bin/activate     # Windows: .\stbelf_env\Scripts\activate
pip install -U pip wheel setuptools

# Core libraries
pip install pandas numpy scikit-learn matplotlib
pip install xgboost lightgbm

# Deep learning
pip install tensorflow              # or tensorflow-macos on Apple Silicon
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl
pip install pytorch-lightning pytorch-forecasting
```

Apple Silicon (optional). For MPS acceleration, use `tensorflow-macos` and the official Apple Metal builds for PyTorch per Apple/PyTorch docs.

4 Repository Layout and Figures

Assume the working directory contains:

- `1. Drammen EDA_after merge changes_FINAL_v1.ipynb`
- `2. Drammen_Feature_Engineering_Final.ipynb`
- `3. Drammen_Model_Training_Final_Ensemble.ipynb`
- `figures/` (for all exported PNGs used in the thesis)

5 End-to-End Setup and Execution

5.1 Phase 1 — EDA and Consolidation (Notebook 1)

Open `1. Drammen EDA_after merge changes_FINAL_v1.ipynb` and set the data folder:

```
LOCAL_DATA_FOLDER = "/path/to/Cofactor_Drammen_Buildings_45_v3/"
```

Run all cells. Outputs created:

- `consolidated_building_metadata.csv` (45 rows static metadata).
- `drammen_model_ready_data.csv` (hourly timeseries for all buildings).

5.2 Phase 2 — Feature Engineering & Selection (Notebook 2)

Open `2. Drammen_Feature_Engineering_Final.ipynb`. Confirm:

- `MODEL_READY_DATA_FILE = "drammen_model_ready_data.csv"`
- `TARGET_VARIABLE = "Electricity_Imported_Total_kWh"`

Run all cells. Outputs saved under `phase2_output_data/`:

- `X_train_fs.csv`, `y_train_fs.csv`, `X_val_fs.csv`, `y_val_fs.csv`, `X_test_fs.csv`, `y_test_fs.csv`
- `scaler_fs.pkl` (for DL models)

5.3 Phase 3 — Model Training, Ensembles, and Exports (Notebook 3)

Open `3. Drammen_Model_Training_Final_Ensemble.ipynb`. Confirm the configuration at the top (paths, seeds). Run all cells. This notebook:

- Trains baselines, RF/XGB/LGBM, LSTM, CNN–LSTM, TFT on the final feature set.
- Trains validation-set stacks (ridge/LGBM) using aligned meta-features.
- Exports final metrics and figures to `figures/` (see below).

6 Random Seeds and Splits

Seeds are set for `numpy/scikit-learn/TensorFlow/PyTorch` (seed = 42). Splits are chronological:

- Train $\leq$ 2020-12-31; Validation $\leq$ 2021-06-30; Test $>$ 2021-06-30.

All engineered features are computed per-building and aligned to $t-1$ to prevent look-ahead.

7 Regenerating Thesis Figures

7.1 Sequence window ($72 \rightarrow 24$) illustration

Exported from Notebook 3 to `figures/seq_window_72_24.png`. If needed, any simple schematic is acceptable.

7.2 CNN–LSTM architecture diagram

If the model object is available in your Python session:

```
from tensorflow.keras.utils import plot_model
# model = ...  # your compiled CNN{LSTM model
plot_model(model,
           to_file="figures/cnn_lstm_arch.png",
           show_shapes=True,
           show_layer_names=True,
           dpi=200)
```

7.3 RF actual vs. predicted (two-week slice)

If `Actual_Predicted_RF.png` wasn't saved by Notebook 3, you can generate it with a short cell:

```
# In Notebook 3, after you have rf_model, X_test, y_test (aligned index):
import pandas as pd, numpy as np, matplotlib.pyplot as plt
pred = np.maximum(rf_model.predict(X_test), 0)
df = pd.DataFrame({"y": y_test, "yhat": pred}).dropna()
# pick one building and a 14-day window in the test set
bid = df.index.get_level_values(0)[0]
two_weeks = df.xs(bid, level=0).iloc[:24*14]
two_weeks.plot(figsize=(12,5))
plt.title("Random Forest: Actual vs. Predicted (two-week slice)")
plt.xlabel("Timestamp"); plt.ylabel("kWh"); plt.tight_layout()
plt.savefig("figures/Actual_Predicted_RF.png", dpi=200)
```

7.4 MAE by model & MAE vs. time (one cell, two PNGs)

If Notebook 3 saved the table `stbelf_model_metrics.csv` (or you've created it from your printed results), use this one-cell helper:

```
import pandas as pd, numpy as np, matplotlib.pyplot as plt

df = pd.read_csv("stbelf_model_metrics.csv")   # columns: Model, MAE_kWh, Train_s
df = df.dropna(subset=["MAE_kWh", "Train_s"])

# --- Figure A: MAE bar chart ---
plt.figure()
df_sorted = df.sort_values("MAE_kWh")
plt.bar(df_sorted["Model"], df_sorted["MAE_kWh"])
plt.ylabel("MAE (kWh)"); plt.title("MAE by Model (Test Set)")
plt.xticks(rotation=45, ha="right")
plt.tight_layout(); plt.savefig("figures/mae_by_model.png", dpi=200)

# --- Figure B: MAE vs Train Time (log scale) ---
plt.figure()
x = np.log10(df["Train_s"].clip(lower=1e-3))
plt.scatter(x, df["MAE_kWh"])
for i, row in df.iterrows():
    plt.annotate(row["Model"], (np.log10(max(row["Train_s"],1e-3)), row["MAE_kWh"]),
                 xytext=(3,3), textcoords="offset points", fontsize=8)
plt.xlabel("log10(Train time [s])"); plt.ylabel("MAE (kWh)")
plt.title("Accuracy vs. Training Time")
plt.tight_layout(); plt.savefig("figures/mae_vs_time.png", dpi=200)
```

8 Troubleshooting

Kernel restarts / memory. Close other notebooks; restart kernel; re-run Phase 2 to regenerate the `phase2_output_data/` panel if shapes mismatch.

Missing figures. Re-run the "Regenerating Thesis Figures" cells above, which only require the saved CSVs and/or trained model in memory.

No MPS/CUDA. All models fall back to CPU; deep models will train slower. Trees (RF/XGB/LGBM) remain fast.

9 Provenance and Reproducibility Checklist

- Unique key: `(building_id, TimeStamp)`; timezone-aware; hourly alignment before merges.
- Chronological split: Train $\leq$2020-12-31, Val $\leq$2021-06-30, Test $>$2021-06-30.
- Final predictors: 35 selected features shared across all model families (trees consumed full engineered set; DL consumed scaled tensors).

- Seeds fixed: Python, NumPy, scikit-learn, TensorFlow, PyTorch (42).
- Ensembling: validation-set stacking (ridge/LGBM meta-learner); OOF omitted on compute grounds.

10 Key Artefacts and Where They Come From

Artefact	Created by	Notes
`consolidated_building_metadata.csv`	Notebook 1	Static per-building metadata.
`drammen_model_ready_data.csv`	Notebook 1	Hourly target and weather aligned.
`phase2_output_data/.csv, pkl`	Notebook 2	Final 35-feature panel (train/val/test) & scaler.
`stbelf_model_metrics.csv`	Notebook 3	Test metrics + train times (export step).
`figures/Actual_Predicted_RF.png`	Notebook 3	Actual vs. predicted (RF).
`figures/mae_by_model.png`	Notebook 3/ (helper)	MAE by model (bar).
`figures/mae_vs_time.png`	Notebook 3/ (helper)	MAE vs. time scatter (log x).
`figures/cnn_lstm_arch.png`	Notebook 3/ (plot_model)	CNN–LSTM block diagram.
`figures/seq_window_72_24.png`	Notebook 3	72→24 sliding window illustration.

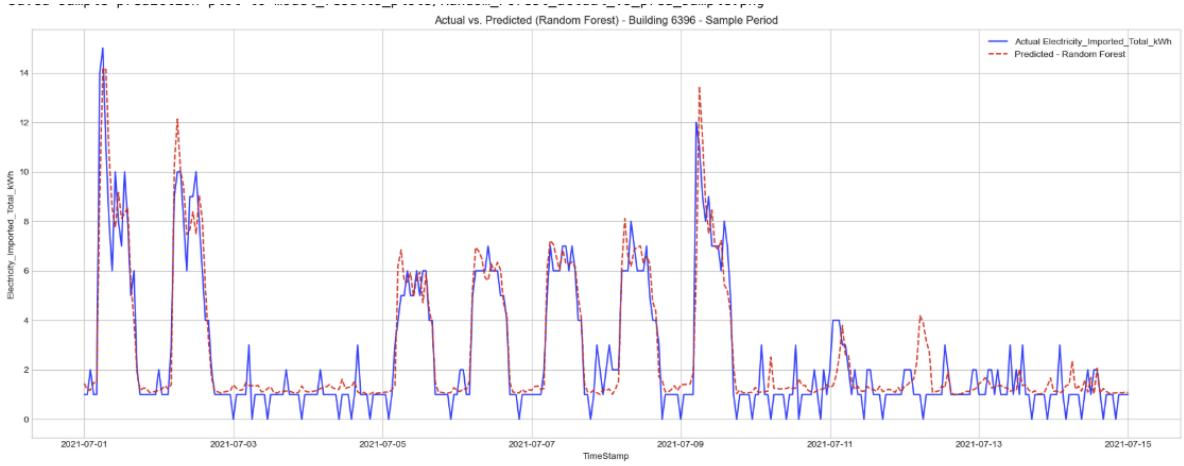

Figure 1: Actual vs. predicted load for Random Forest on a representative building (two-week slice, test period).

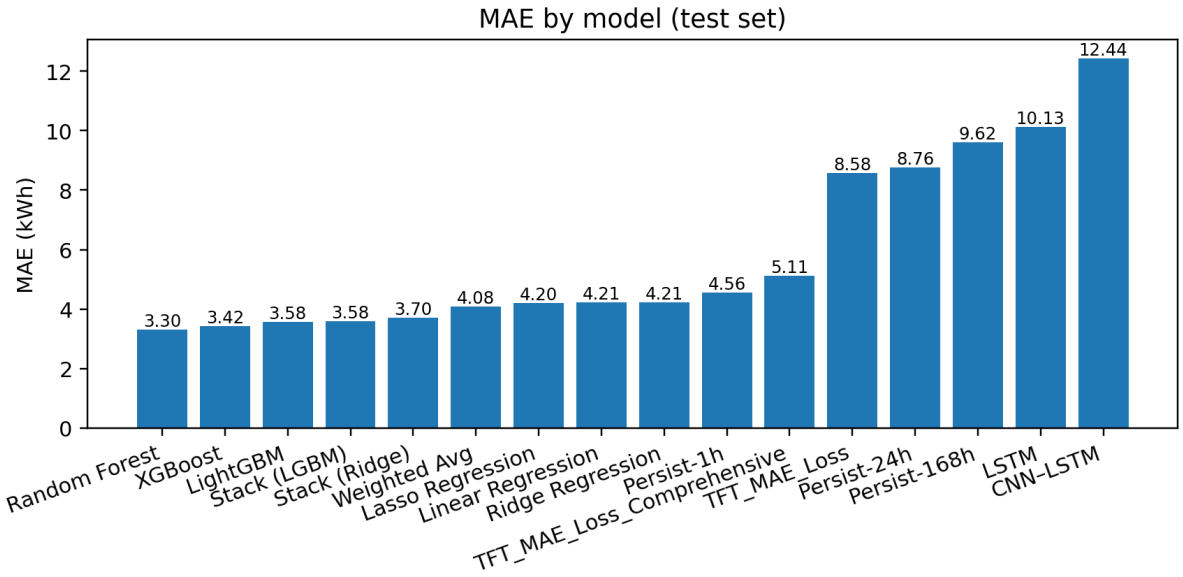

Figure 2: MAE by model.

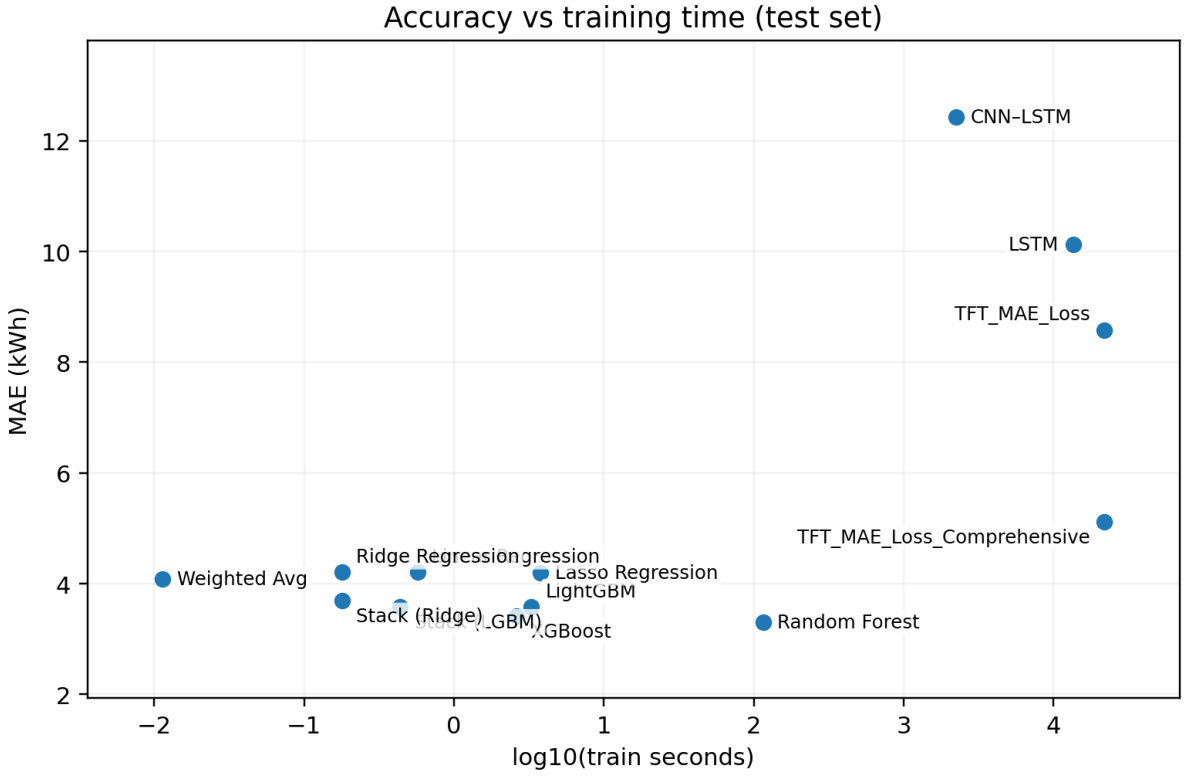

Figure 3: MAE vs time to train.

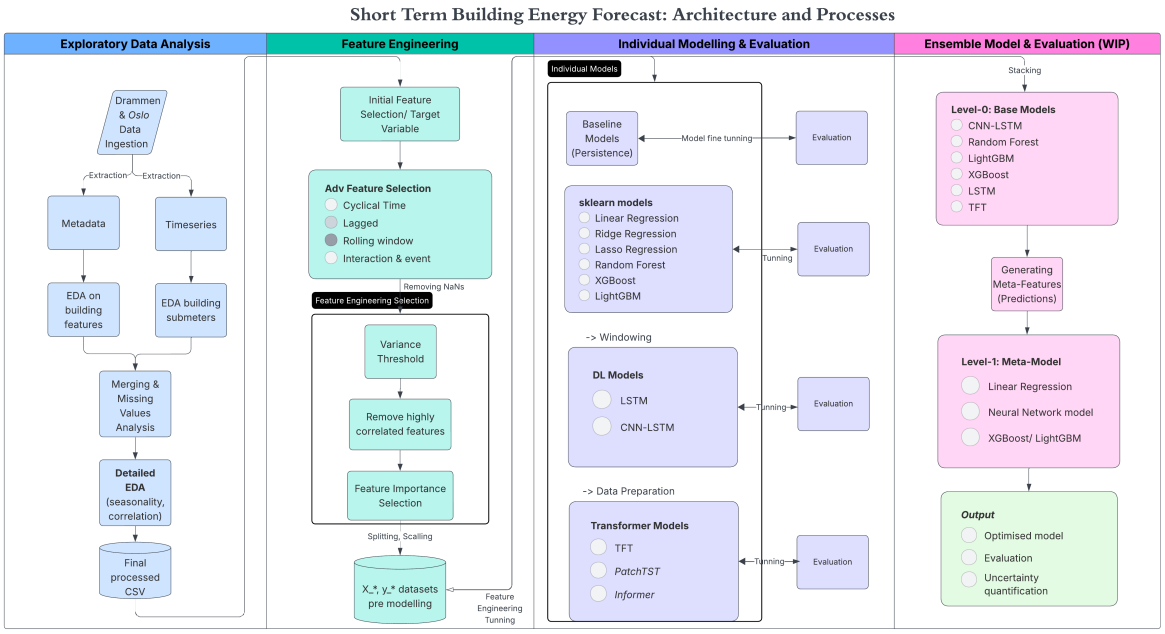

Figure 4: Architecture.