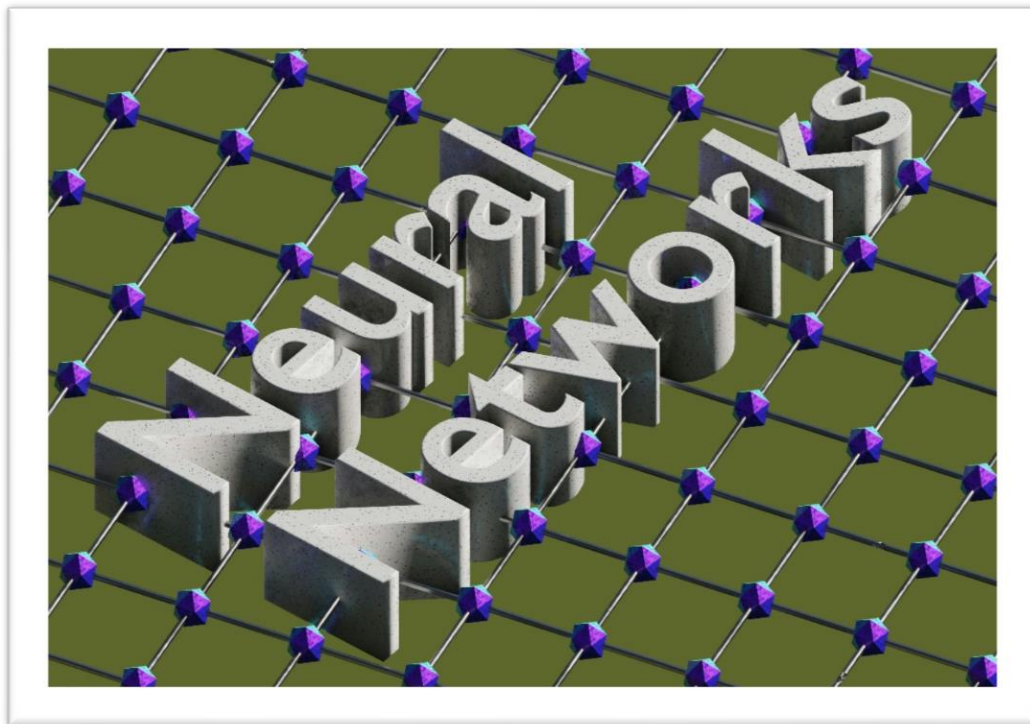


# Multivariate Time- Series Prediction Using Deep Learning



*I.V.R.S. Induruwa*

*Date: 2025/11/23*

# Tables of contents:

- 1) [Overview of Dataset](#)
- 2) [Exploratory Data Analysis](#)
- 3) [Feature Engineering](#)
- 4) [Data Preprocessing](#)
- 5) [Model Development](#)
- 6) [Model Training](#)
- 7) [Conclusion](#)
- 8) [References](#)

# Overview of Dataset

## Dataset Details

- Size: Approximately 20,000 records
- Interval: 10-minute intervals
- Duration Covered: Several months

## Key Features

- Date: Timestamp of each observation (YYYY-MM-DD HH:MM:SS).
- Appliances: Energy consumption in Wh (target variable).
- Lights: Energy consumption of lights in Wh.
- T1-T6: Indoor temperature readings from different areas of the building (in Celsius).
- RH\_1–RH\_6: Indoor humidity readings corresponding to temperature sensors.
- T\_out: Outdoor temperature in Celsius.
- RH\_out: Outdoor humidity percentage.
- Windspeed: Outdoor wind speed (m/s).
- Visibility: Outdoor visibility (km).
- Press\_mm\_hg: Atmospheric pressure outside the building (mmHg).
- NSM: Numerical step counter starting at midnight (seconds elapsed since 00:00:00).
- WeekStatus: Indicates whether a day is a Weekday or Weekend.
- Day\_of\_week: Day of the week as a number (e.g., 0 = Sunday, 6 = Saturday).

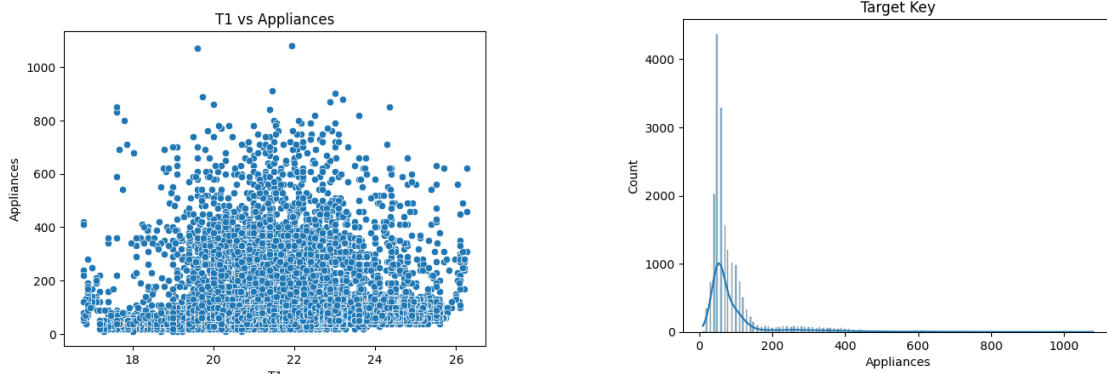
# Exploratory Data Analysis

## Overview

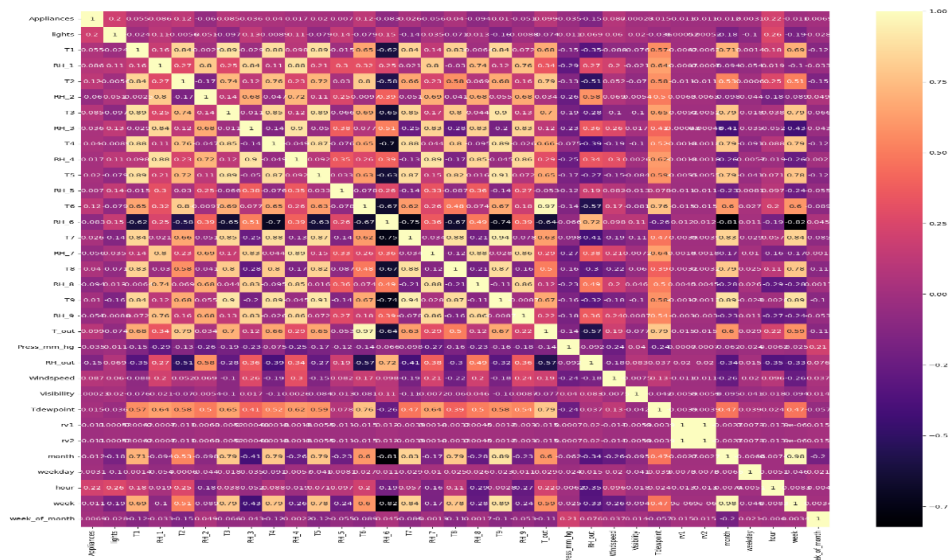
- Initial analysis was conducted to understand the distribution of energy consumption, identify seasonal patterns, and detect potential multicollinearity among sensor data.

## Visual Analysis

- **Target Distribution:** A histogram with a Kernel Density Estimate (KDE) revealed that the target variable, Appliances energy consumption, is heavily right-skewed. Most data points represent low energy usage, with occasional high-consumption spikes.



- **Correlation Analysis:** A heatmap was generated to visualize correlations between indoor or outdoor temperature (T1–T9, T\_out) and humidity (RH\_1–RH\_9). Strong correlations were observed between sensors in adjacent rooms, suggesting redundancy in the data.



- **Feature Relationships:** Scatter plots (T1 vs. Appliances) were used to identify linear or non-linear relationships between individual features and energy consumption.

**Multicollinearity Check (VIF)** To ensure statistical robustness, Variance Inflation Factor (VIF) analysis was performed on all numerical features.

- **Method:** VIF scores were calculated for all sensor inputs (T1 to T9, RH\_1 to RH\_9, Weather data).
- **Finding:** High VIF scores were detected among several temperature and humidity sensors, confirming significant multicollinearity. This insight guided the decision to use non-linear models which are generally more robust to multicollinearity than simple linear regression.

# Feature Engineering

❖ **Time-Based Feature Extraction** Since energy consumption is highly dependent on human activity schedules, the date column was decomposed into granular temporal features:

- **Cyclical Features:** month, weekday, hour.
- **Categorical Features:** day\_of\_week, week\_of\_month, week.
- **Relevance:** These features allow the model to learn daily and weekly consumption patterns.

❖ **Feature Selection via Decision Tree** To identify the most predictive features and reduce noise, a tree-based feature importance analysis was conducted.

- **Method:** A DecisionTreeRegressor was trained on the processed dataset.
- **Outcome:** The feature\_importances\_ attribute was extracted to rank features.
- **Top Features Identified:** The model indicated that hour, T\_out, and RH\_out were among the strongest predictors of energy usage. This confirmed that time of day and weather conditions are the primary drivers of appliance energy consumption.

# Data Preprocessing

## Data Cleaning and Outlier Treatment

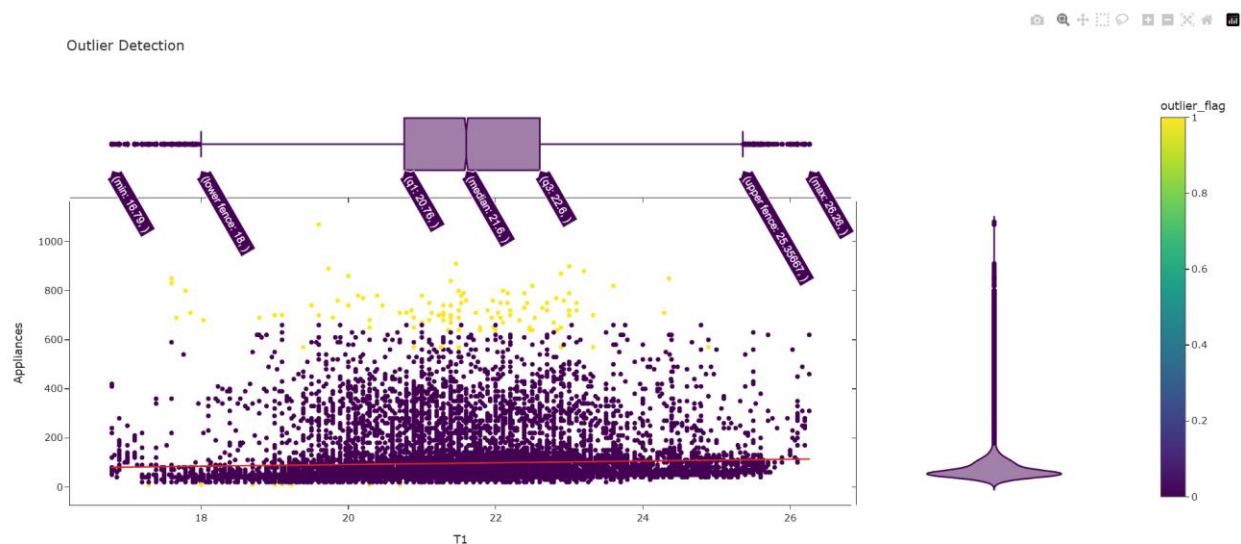
➤ A two-step approach was adopted to handle outliers effectively:

### 1. DBSCAN (Density-Based Spatial Clustering):

- **Method:** Used the DBSCAN algorithm ( $\text{eps}=0.5$ ,  $\text{min\_samples}=10$ ) to identify density-based clusters and isolate noise points.
- **Purpose:** This was primarily used to visualize complex outlier patterns that simple statistical thresholds might miss. A scatter plot with violin margins was generated to visualize these clusters against Temperature (T1).

### 2. IQR (Interquartile Range) Removal:

- **Method:** For the final cleaning pipeline, the robust IQR method was applied. Data points falling below  $Q1 - 1.5 \times \text{IQR}$  or above  $Q3 + 1.5 \times \text{IQR}$  in the Appliances column were removed.
- **Result:** This effectively removed extreme high-consumption spikes (likely sensor errors or rare events) that could skew model training, resulting in a cleaner dataset for the LSTM.



## **Handling Missing Values**

- The dataset was checked for null values. Missing data was handled using forward-fill and backward-fill methods to maintain temporal continuity essential for time-series forecasting.



# Model Development

## Baseline Models

- To establish performance benchmarks, two traditional algorithms were implemented:
  - **Linear Regression:** A simple baseline to test for linear relationships.
  - **Random Forest Regressor:** A non-linear ensemble method used to capture more complex interactions without deep learning.

## Deep Learning Architecture (LSTM)

- The primary model utilized a Long Short-Term Memory (LSTM) network, chosen for its ability to learn dependencies in time-series data.
  - **Input Layer:** Designed to accept a rolling window sequence of the past 24 hours
  - **LSTM Layer:** A layer with **64 units** was used to extract temporal features from the input sequence.
  - **Regularization:** A **Dropout layer (rate=0.2)** was inserted to randomly zero out inputs during training, preventing overfitting.
  - **Dense Layers:** The output was passed through a hidden Dense layer (**32 neurons**, ReLU activation) for further processing, followed by a single output neuron (Linear activation) to predict the continuous energy value.
  - **Compilation:** The model was compiled using the **Adam optimizer** (learning rate=0.001) and **Mean Squared Error (MSE)** as the loss function.

# Model Training

## Training Strategy

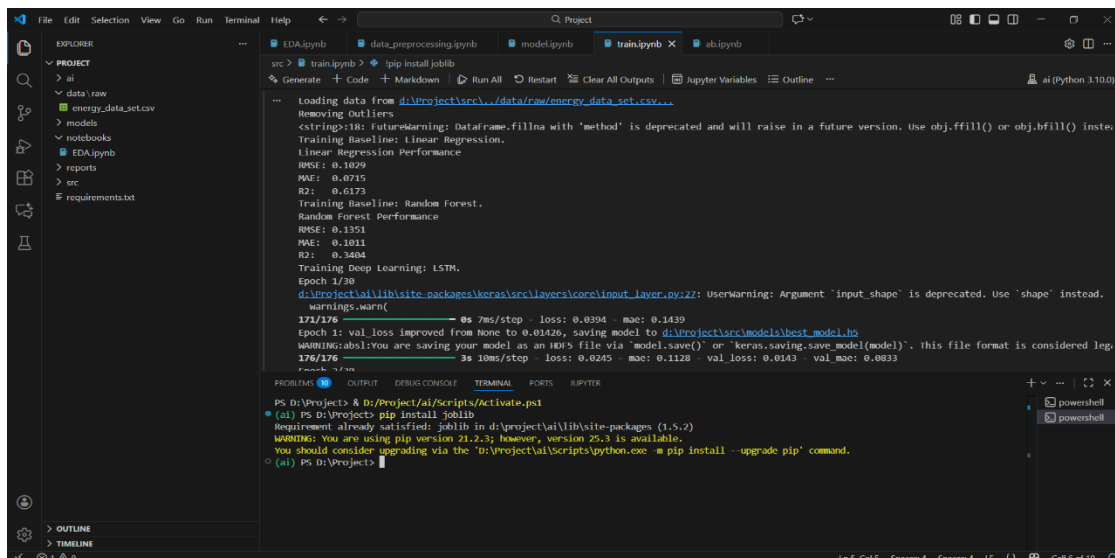
- The dataset was split into training (80%) and testing (20%) sets, strictly maintaining temporal order to prevent data leakage.

## Hyperparameters

- The LSTM model was trained with the following configuration:
  - **Batch Size:** 64 samples per gradient update.
  - **Epochs:** Set to 15 (with early stopping).
  - **Callbacks:**
    - **EarlyStopping:** Monitored validation loss with a patience of 3 epochs to automatically stop training when improvement plateaued.
    - **ModelCheckpoint:** Automatically saved the model weights that achieved the lowest validation loss during training.

## Evaluation

- Post-training, the model's performance was evaluated on the unseen test set using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared to quantify prediction accuracy and variance explanation.



The screenshot displays a Jupyter Notebook with three tabs: 'EDA.ipynb', 'data\_preprocessing.ipynb', and 'train.ipynb'. The 'train.ipynb' tab is active, showing the following output:

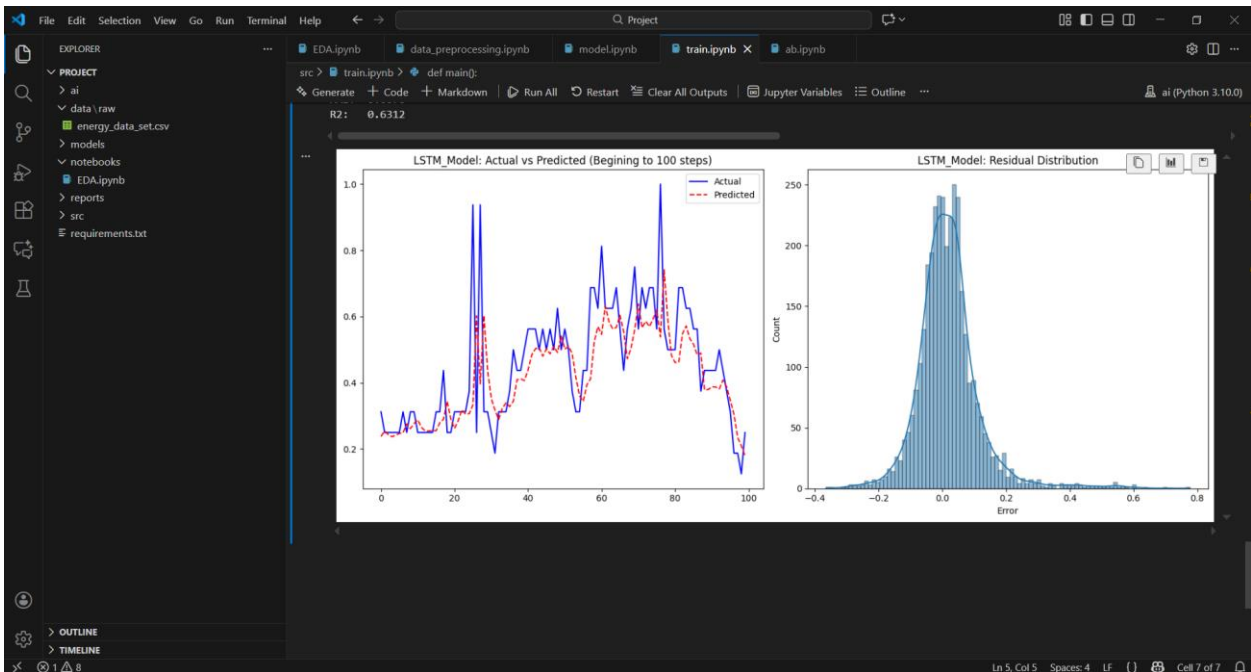
```
src> train.ipynb > pip install joblib
Loading data from d:\Project\src\...\data\raw\energy_data_set.csv...
Removing Outliers
cstrings:18: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version, use obj.ffill() or obj.bfill() inste...
Training Baseline: Linear Regression.
Linear Regression Performance
RMSE: 0.1029
MAE: 0.0715
R2: 0.6173
Training Baseline: Random Forest.
Random Forest Performance
RMSE: 0.1351
MAE: 0.1011
R2: 0.3404
Training Deep Learning: LSTM.
Epoch 1/10
d:\Project\ai\lib\site-packages\keras\layers\core\input_layer.py:27: UserWarning: Argument 'input_shape' is deprecated. Use 'shape' instead.
warnings.warn(
171/176 ----- 0s 7ms/step - loss: 0.0394 - mae: 0.1439
Epoch 1: val_loss improved from None to 0.01426, saving model to d:\Project\src\models\best_model.h5
WARNING:absl:You are saving your model as an H5 file via 'model.save()' or 'keras.saving.save_model(model)', this file format is considered leg...
176/176 ----- 3s 10ms/step - loss: 0.0245 - mae: 0.1128 - val_loss: 0.0143 - val_mae: 0.0833
Epoch 2/10
```

The bottom of the notebook shows the terminal output of the command `pip install joblib`, which is successful. The status bar at the bottom indicates the file is `Ln 5, Col 5`, with `Spaces: 4` and `LF`.

LINEAR REGRESSION AND RANDOM FOREST PERFORMANCES

```
File Edit Selection View Go Run Terminal Help
train.ipynb
src > train.ipynb > !pip install joblib
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
176/176 2s 9ms/step - loss: 0.0123 - mae: 0.0769 - val_loss: 0.0097 - val_mae: 0.0669
Epoch 7/30
174/176 0s 8ms/step - loss: 0.0119 - mae: 0.0763
Epoch 7: val_loss did not improve from 0.00973
176/176 2s 9ms/step - loss: 0.0123 - mae: 0.0768 - val_loss: 0.0117 - val_mae: 0.0751
Epoch 8/30
176/176 0s 8ms/step - loss: 0.0127 - mae: 0.0780
Epoch 8: val_loss did not improve from 0.00973
176/176 2s 9ms/step - loss: 0.0122 - mae: 0.0763 - val_loss: 0.0105 - val_mae: 0.0705
Epoch 9/30
174/176 0s 8ms/step - loss: 0.0120 - mae: 0.0749
Epoch 9: val_loss did not improve from 0.00973
176/176 2s 9ms/step - loss: 0.0119 - mae: 0.0751 - val_loss: 0.0099 - val_mae: 0.0698
WARNING:absl:You are saving your model as an HDF5 file via 'model.save()' or 'keras.saving.save_model(model)'. This file format is considered leg.
Final model successfully saved to: d:\Project\src\models\trained_model.h5
Scaler successfully saved to: d:\Project\src\models\scaler.pkl
110/110 1s 4ms/step
LSTM Performance
RMSE: 0.1010
MAE: 0.0676
R2: 0.6312
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
PS D:\Project> & D:\Project\ai\scripts\activate.ps1
(ai) PS D:\Project> pip install joblib
Requirement already satisfied: joblib in d:\project\ai\lib\site-packages (1.5.2)
WARNING: You are using pip version 21.2.3; however, version 25.3 is available.
You should consider upgrading via the 'D:\Project\ai\scripts\python.exe -m pip install --upgrade pip' command.
(ai) PS D:\Project>
```

## LSTM PERFORMANCES



## FINAL PREDICTIONS WITH ACTUAL VALUES

# Conclusion

## Tips to Improve Model Performances

### Hyperparameter Tuning

- **Batch Size:**
  - Lowering batch size to 32.
- **Units (Neurons):**
  - Increasing LSTM units from 64 to 128 or 256. this increases training time and risk of overfitting (make sure you keep Dropout).

### Data & Features

- **Increase Window Size:** Currently, looking back **24 hours**. Energy consumption often has weekly patterns.
  - Change WINDOW = 168 (24 hours \* 7 days).

### Training Strategy

- **Increase Epochs & Patience:** Instead of just increasing epochs to 100, must also relax the "Early Stopping." If model needs time to learn complex patterns, a patience of 3 might be too strict.
  - *Change:* Set epochs=100 and patience=10.
- **Learning Rate Scheduler:** Sometimes the model gets "stuck" because the learning rate is too high to find the exact bottom of the error valley. A scheduler lowers the rate when progress stalls.

### Model Architecture Changes

- **Stacked LSTM:** One LSTM layer might not be enough to capture complex relationships.
  - The first LSTM layer must have return\_sequences=True to pass the sequence to the second layer.
- **Bidirectional LSTM:** This allows the model to learn from the sequence in both forward and backward directions. It is often more powerful than standard LSTM.

# My Referance

- ✓ YouTube Video - [Multivariate Time Series Forecasting Using LSTM, GRU & 1d CNNs](#)
- ✓ Telegram Channel – [Data Science](#)
- ✓ ResearchGate – [Click Here](#)