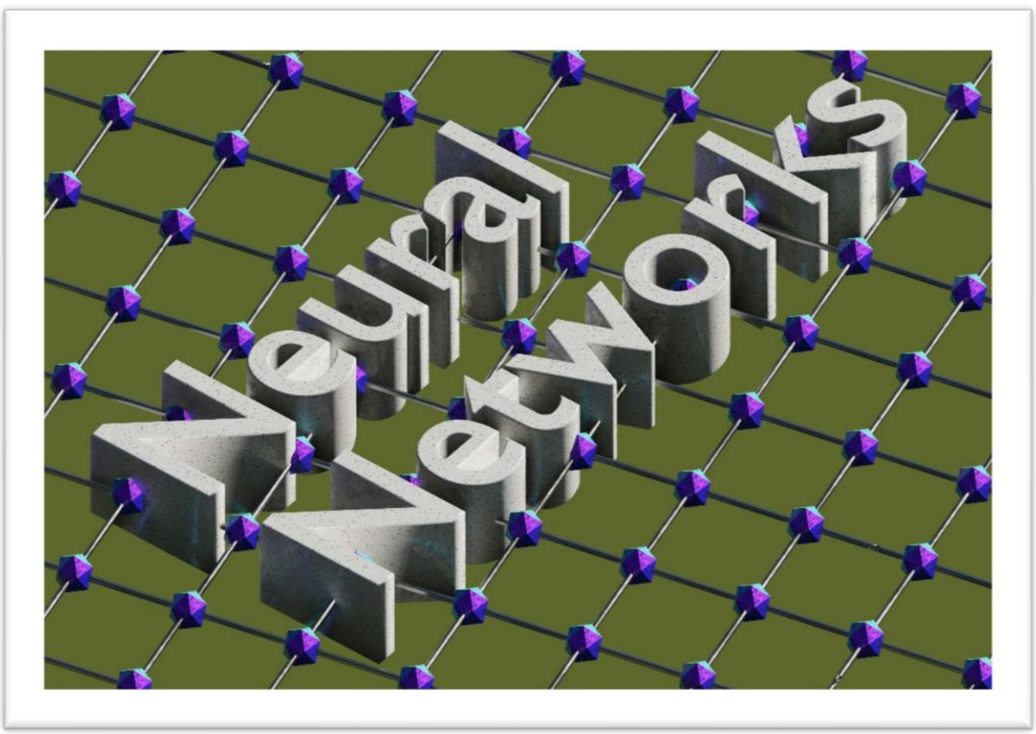
Multivariate Time-Series Prediction Using Deep Learning



I.V.R.S. Induruwa

Date: 2025/11/23

Tables of contents:

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).

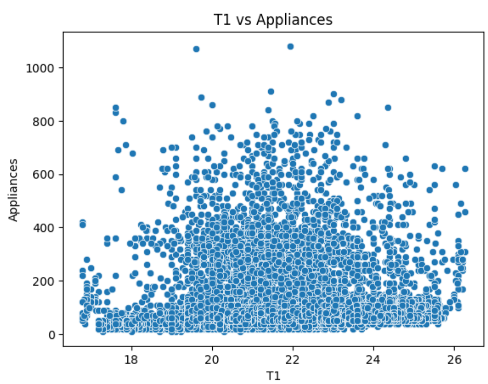
Exploratory Data Analysis

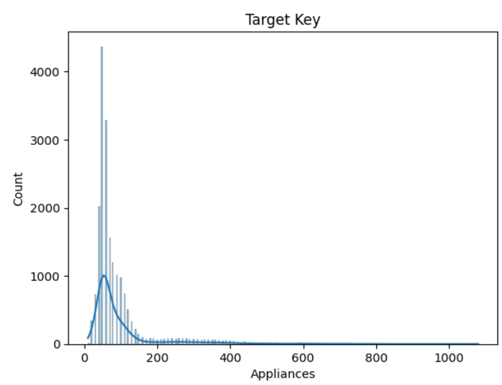
Overview

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

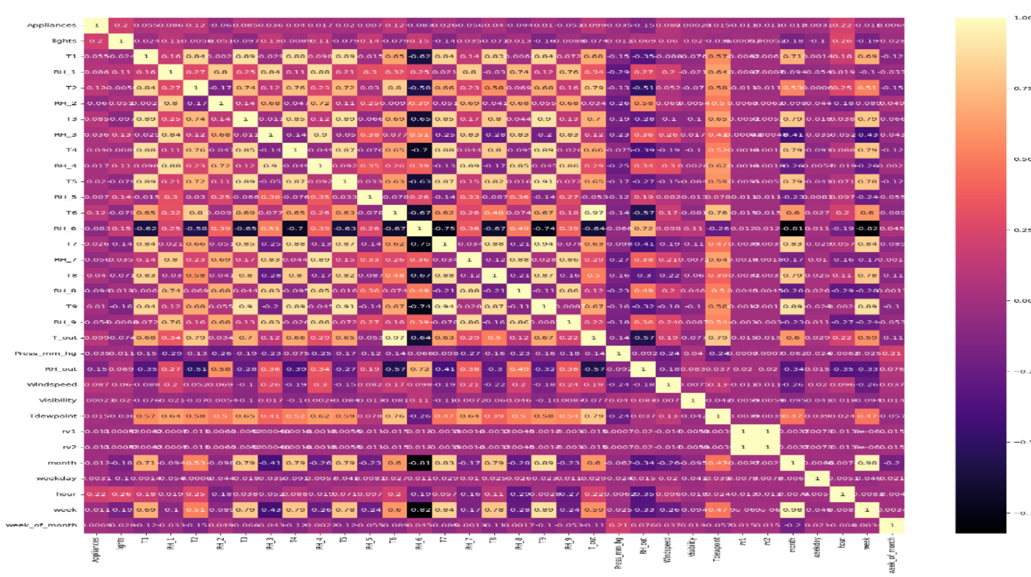
Visual Analysis

- **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 (`eps=0.5`, `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`).
    - *[Insert Figure: DBSCAN Outlier Detection Plot]*

2. **IQR (Interquartile Range) Removal:**
    - **Method:** For the final cleaning pipeline, the robust IQR method was applied. Data points falling below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$ 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.

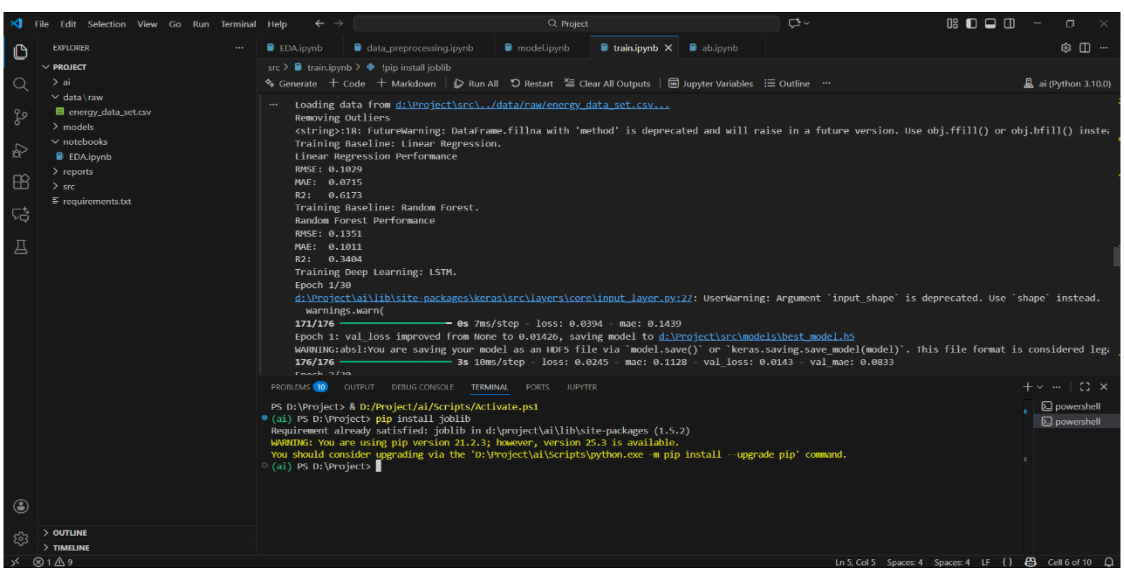
Hyperparameters

➢ The LSTM model was trained with the following configuration:

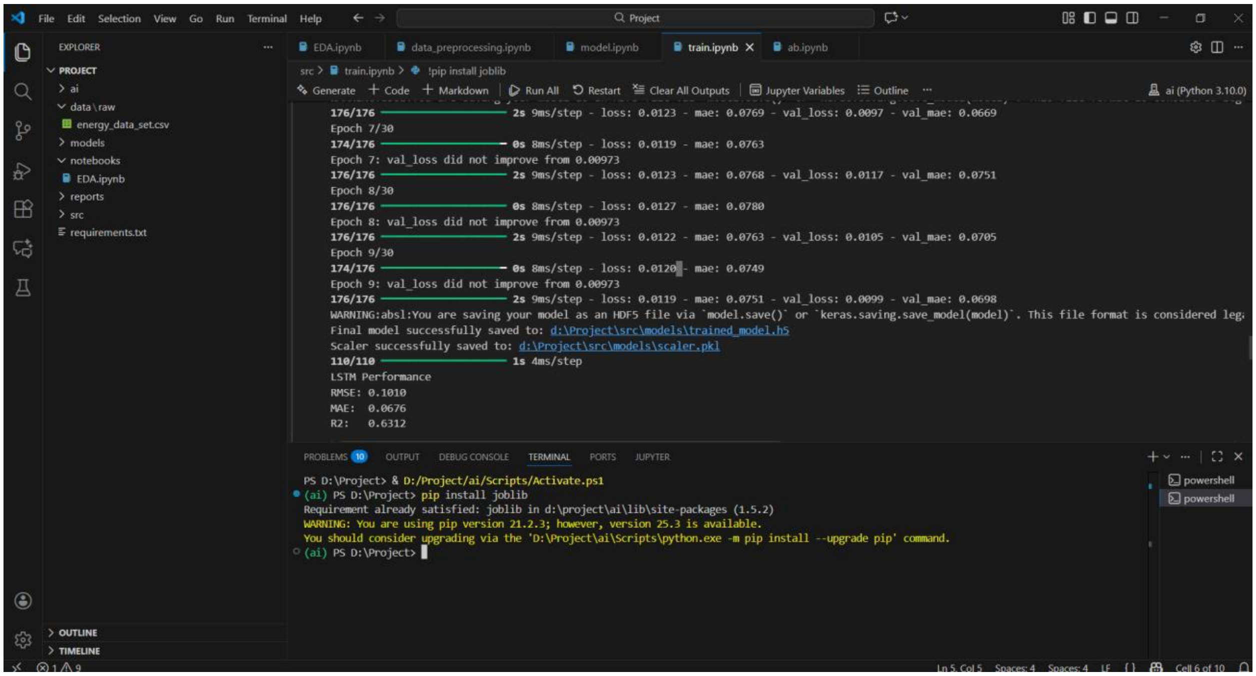
- **Batch Size:** 64 samples per gradient update.
- **Epochs:** Set to 15 (with early stopping).
- **Callbacks:**
    - **EarlyStopping:** Monitored validation loss with a patience of 3 epochs to automatically stop training when improvement plateaued.
    - **ModelCheckpoint:** Automatically saved the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

Evaluation

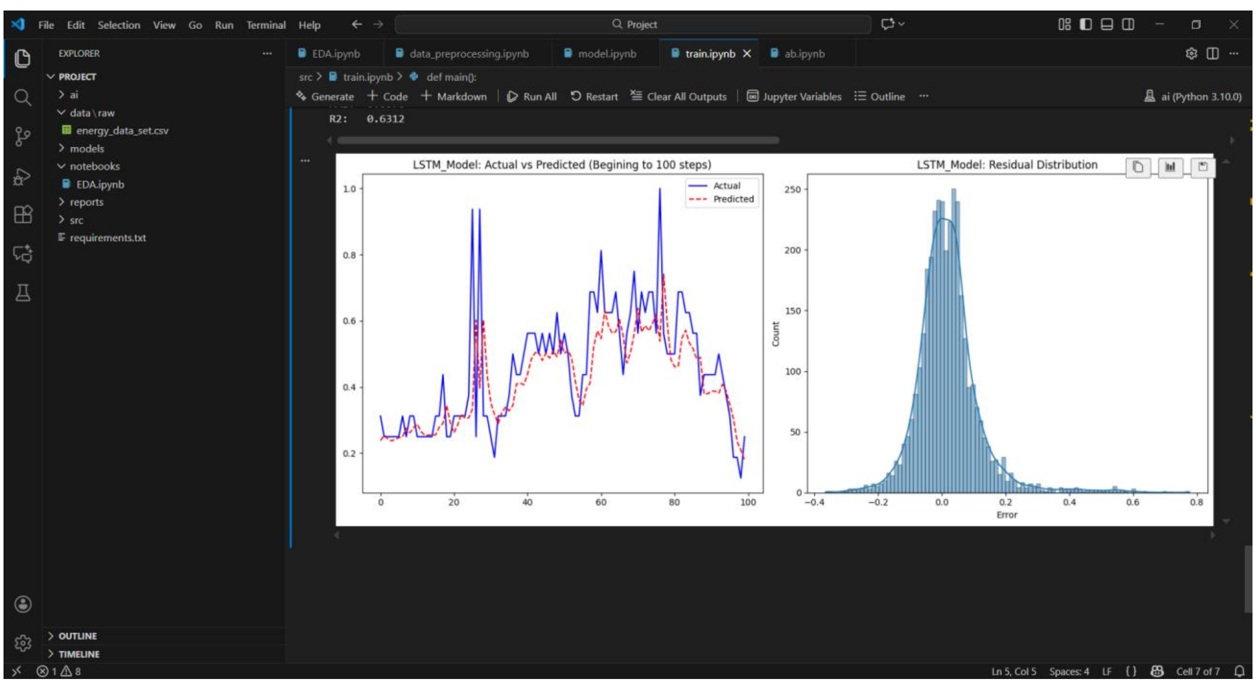
➢ Post-training, the model's performance was evaluated on the unseen test set using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared to quantify prediction accuracy and variance explanation.



LINEAR REGRESSION AND RANDOM FOREST PERFORMENCES



LSTM PERFORMENCES



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]**

- ✓ **Telegrame Channel – [Data Science]**

- ✓ **Researchgate – [Click Here]**