Assessment Multivariate Time-Series Prediction Using Deep Learning

R.M.A.Madushan

(28/05/2025)

Table of Contents

Introduction

Accurate energy consumption prediction in smart homes is a growing challenge due to the complex and time-dependent nature of appliance usage patterns. Traditional methods often fail to capture these temporal dynamics effectively. This project aims to develop a deep learning-based time series model using the Appliance Energy Prediction dataset, which contains over 20,000 records of multivariate sensor data, including temperature, humidity, and appliance usage. The objective is to build and evaluate a model that can predict future appliance energy consumption with high accuracy by leveraging temporal relationships in the data. (Tzelepi, 2023) (geeksforgeeks, 2024)

EDA Analysis

1. Strong Hourly and Daily Patterns

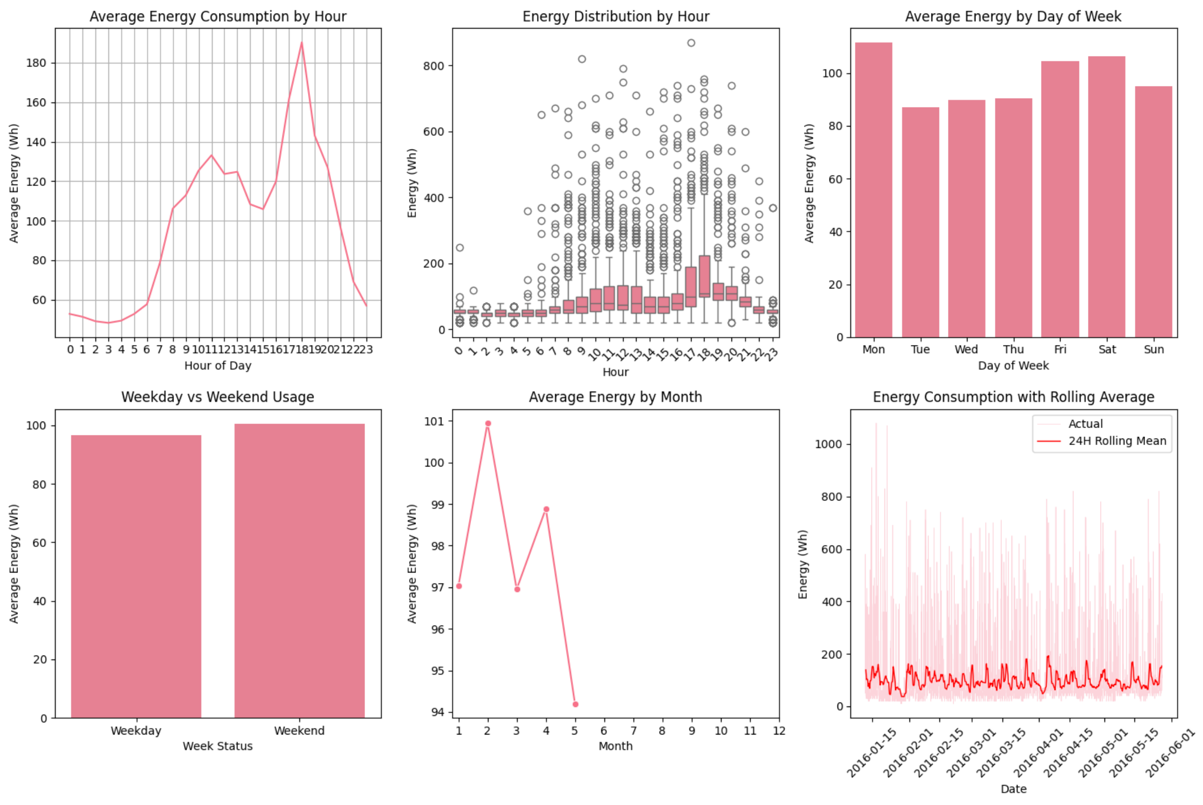


Figure 1

- Energy consumption peaks consistently around 6 PM, and is lowest during early morning hours (2–5 AM), indicating daily cyclic behavior.

- Weekday vs. Weekend consumption shows subtle differences, with weekends slightly higher, possibly due to occupants being home.
- Hourly trends display strong periodicity, as confirmed by rolling average plots.
- These insights suggest that the following engineered time features will enhance prediction:
    - hour, day_of_week, is_weekend, month
- Time-based lags and rolling statistics can also help model short-term dependencies (e.g., past 3-hour average consumption).

2. Temperature Features Offer Moderate Predictive Power

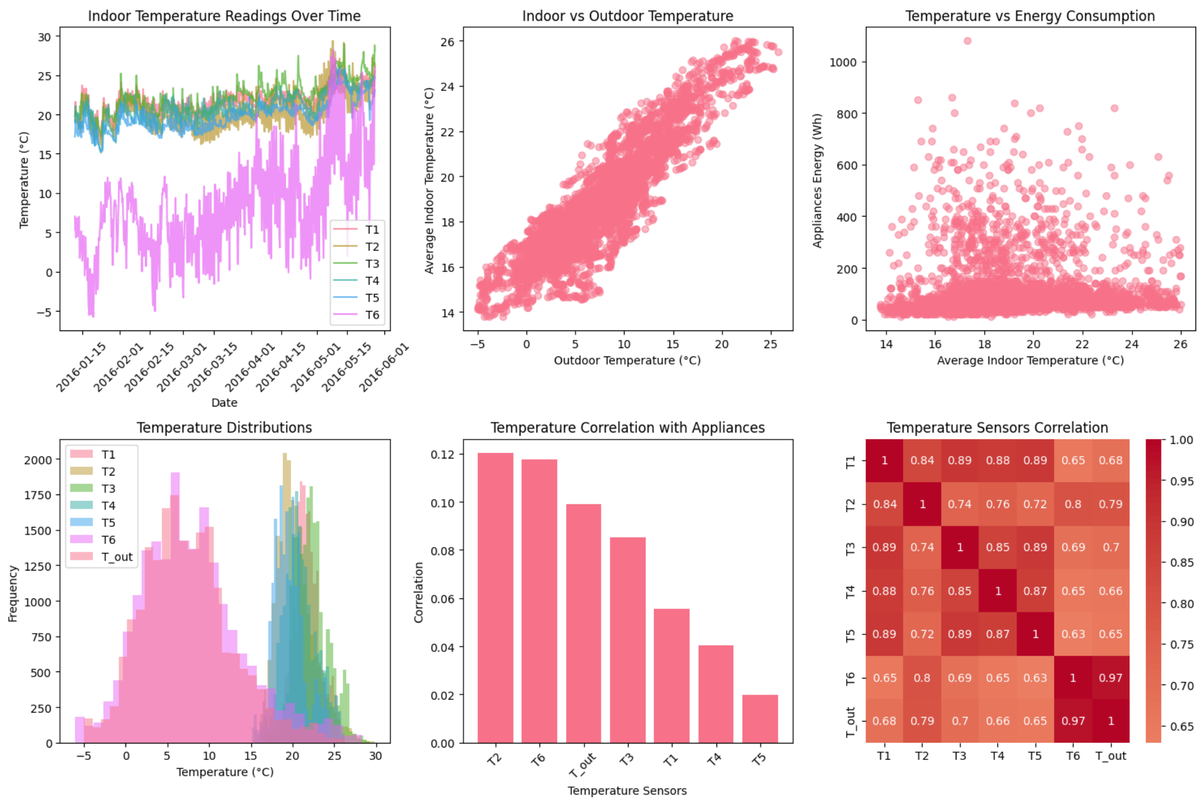
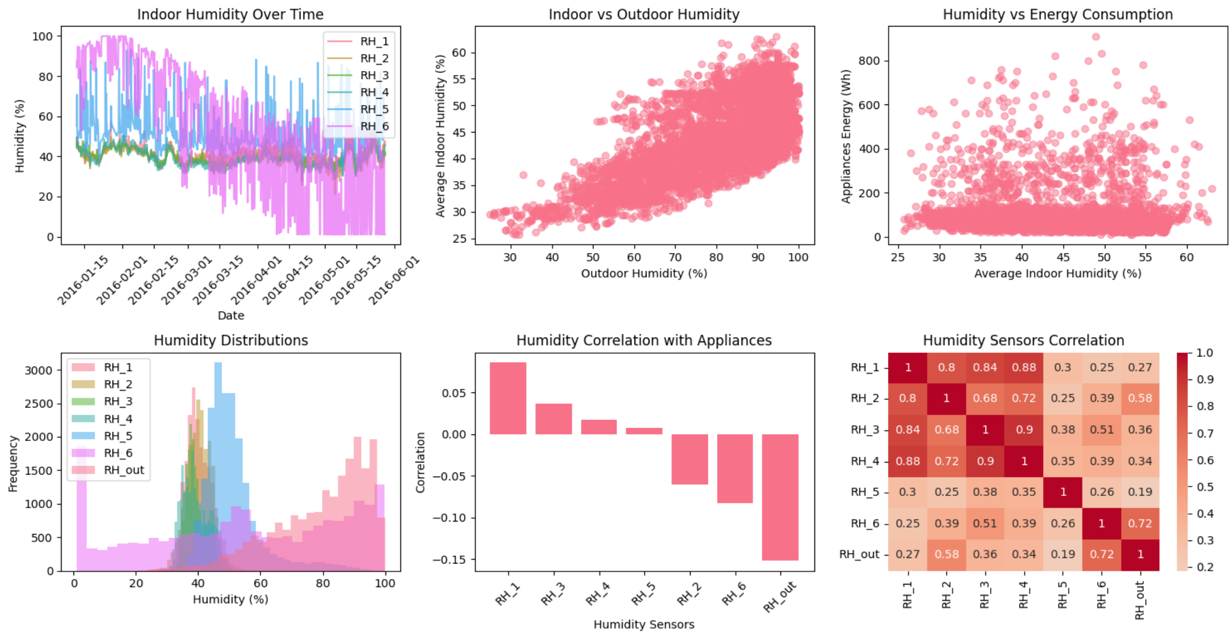


Figure 2

- Among temperature variables, T2 (Kitchen temp) and T6 (Outer wall temp) show the highest correlation with appliance usage, suggesting their importance.
- Outdoor temperature (T_out) also has moderate influence, aligning with potential HVAC usage patterns.
- Some sensors (T1, T3, T5) are highly correlated, indicating redundancy.
    - Consider dropping or combining these to reduce multicollinearity.
- Suggests that temperature dynamics inside and outside impact appliance usage, particularly heating/cooling loads.

3. Humidity and Weather Have Limited Predictive Value



	RH_1	RH_2	RH_3	RH_4	RH_5	RH_6	RH_out
RH_1	1	0.8	0.84	0.88	0.3	0.25	0.27
RH_2	0.8	1	0.68	0.72	0.25	0.39	0.58
RH_3	0.84	0.68	1	0.9	0.38	0.51	0.36
RH_4	0.88	0.72	0.9	1	0.35	0.39	0.34
RH_5	0.3	0.25	0.38	0.35	1	0.26	0.19
RH_6	0.25	0.39	0.51	0.39	0.26	1	0.72
RH_out	0.27	0.58	0.36	0.34	0.19	0.72	1

Figure 3

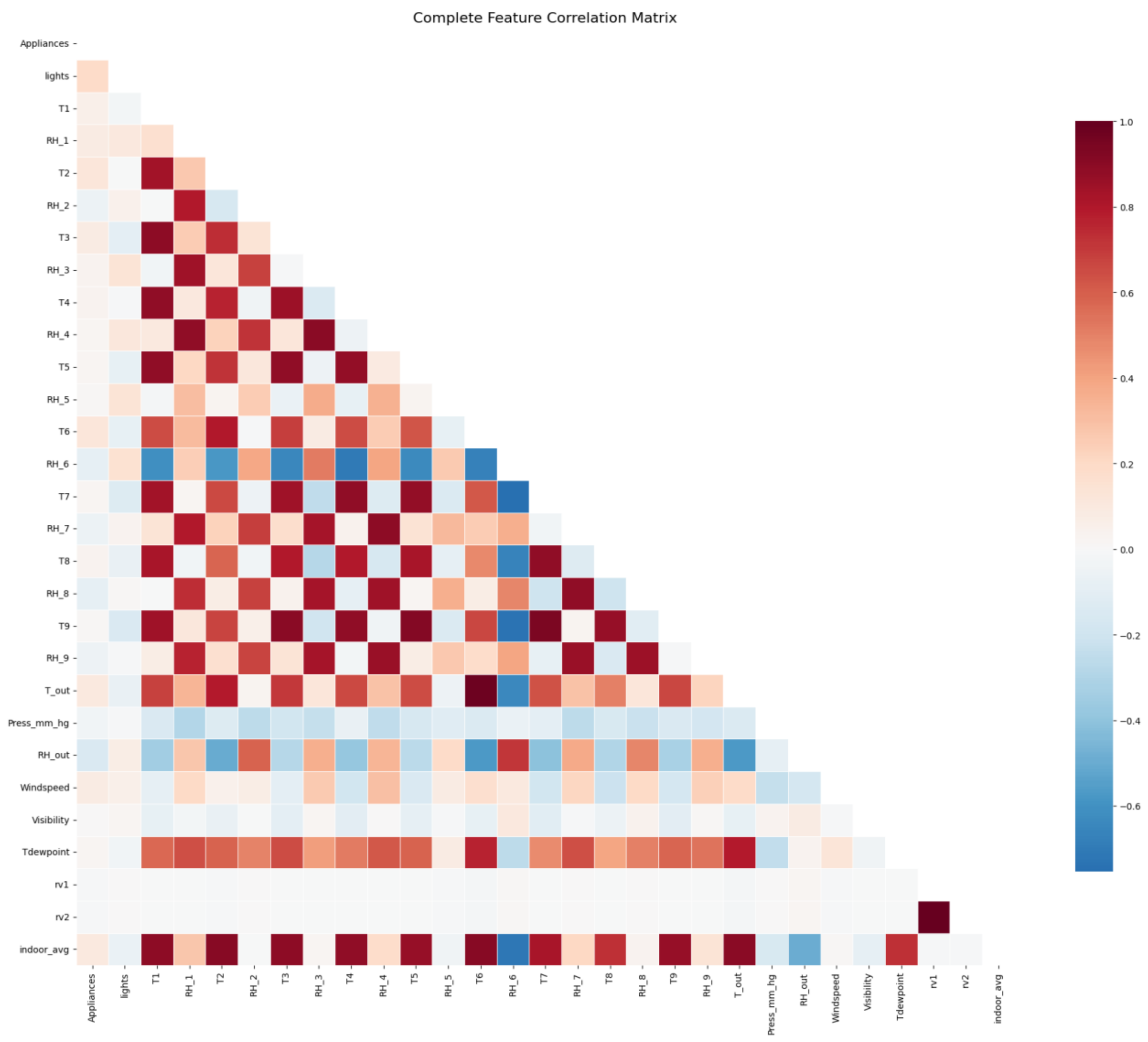


Figure 4

- Most humidity features show weak correlation with energy consumption.
  - Exception: RH_6 (Ironing room), which shows a moderate negative correlation — may reflect room-specific appliance activity (Figure 4).
- Outdoor humidity is consistently high and negatively correlated with usage.
- Weather features (e.g., windspeed, visibility, pressure) show very weak or no direct relationship with appliance usage.
  - Might be included for completeness but should not be heavily weighted in model training unless advanced modelling finds non-linear effects.

4. Many Features Are Correlated → Feature Selection / Reduction Is Necessary

- Strong multicollinearity exists among:
  - Temperature sensors: T1–T3–T5, T6–T_out
  - Humidity sensors: RH_1–RH_3–RH_4
- This could lead to model instability, inflated importance, and overfitting in regression models. (Chandra, 2021)

- Recommended techniques:
  - Correlation thresholding (e.g., remove one of any pair with |r| > 0.9)
  - Principal Component Analysis (PCA) or Recursive Feature Elimination (RFE) for dimensionality reduction
  - Lasso regularization to shrink unimportant features

5. Outlier Treatment is Essential for Reliable Model Performance

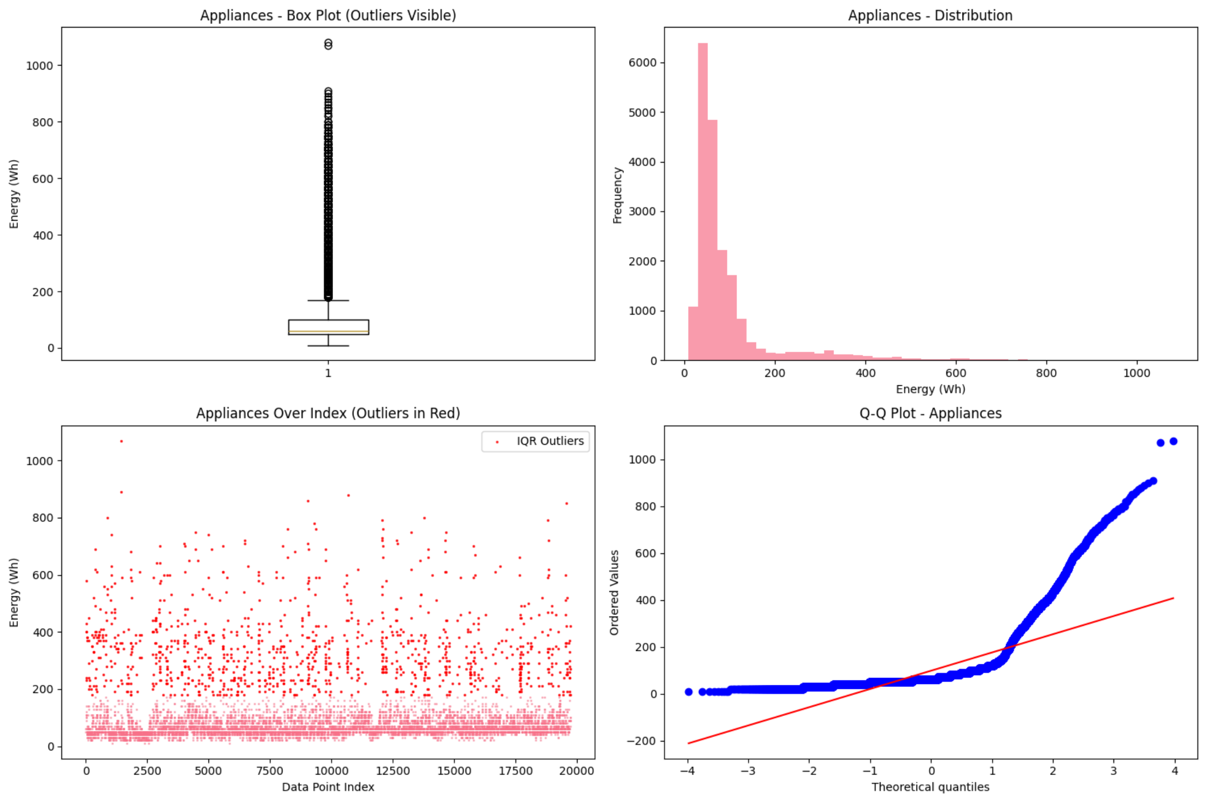


Figure 5

- Energy consumption is right-skewed with many extreme values (as seen in histograms, box plots, and Q-Q plots).
- IQR method found ~11% of data as outliers; Z-score method found ~3%.
- Outliers may be caused by:
  - Sudden heating/cooling demands
  - Appliance malfunctions or simultaneous use of multiple devices
- Why it matters:
  - Outliers can skew model training, especially linear models or MSE-based loss functions.
- Recommended treatment strategies:
  - capping extreme values
  - Log-transformation to reduce skewness
  - Use of robust models (like Random Forest or Huber loss in regression)

6. Feature Irrelevance and Noise → Drop Non-informative Features

- Features rv1 and rv2 show no meaningful correlation with any variable, including the target — likely random noise(Figure 4).
- Removing such features can improve model efficiency and generalization.

Preprocessing

1. Data Cleaning and Preparation

The dataset consisted of 19,735 records collected at 10-minute intervals over a period of 137 days, with no missing values detected. This ensured a solid foundation for consistent and reliable model training.

To enhance predictive performance, several new features were engineered from the datetime column, including:

- hour, minute, day, month, year
- day_of_week (0 = Sunday, 6 = Saturday)
- WeekStatus (Weekday or Weekend)
- NSM (Number of Seconds elapsed in a day)

These temporal features are essential for capturing daily and weekly usage patterns of energy.

2. Handling of Missing Values

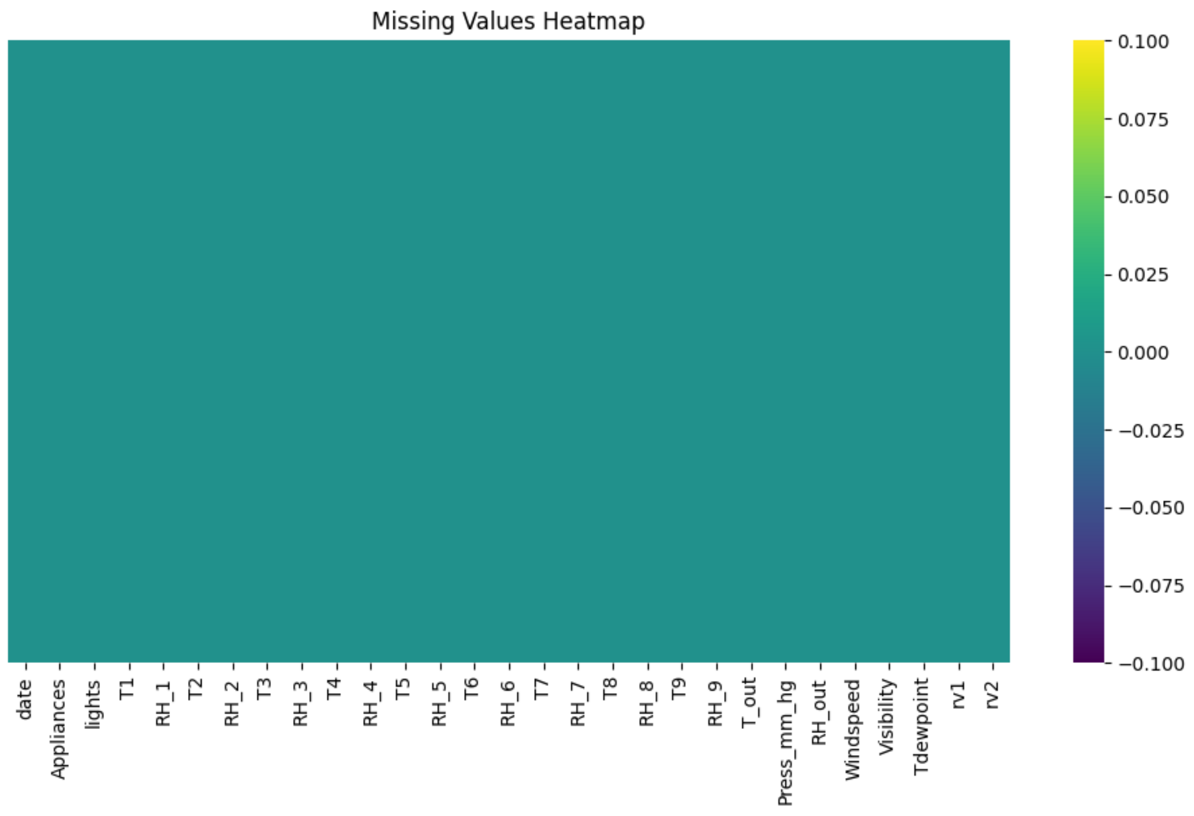


Figure 6

No missing values were found in the dataset, eliminating the need for imputation techniques. All columns were complete and consistent across the dataset.

3. Outlier Detection and Treatment

Outlier Analysis:

Two methods were applied to detect outliers in the target variable:

- IQR method: Detected 2138 outliers (10.83%)
- Z-score method: Detected 648 outliers (3.28%)

Outliers had a significant impact on the distribution and could potentially bias the model.

Outlier Treatment Strategy;

To address outliers without losing valuable data:

- Capping was applied at the 1st and 99th percentiles of the target variable distribution.

This approach preserved the entire dataset while minimizing the influence of extreme values. Post-treatment, the max value was reduced from 1080 Wh to 576.6 Wh, improving stability during training.

4. Data Scaling and Normalization

To prepare the data for neural networks, z-score normalization (StandardScaler) was applied to 27 continuous features, including:

- Temperature features (T1–T9)
- Humidity features (RH_1–RH_9)
- Weather indicators (Windspeed, Visibility, Press_mm_hg)
- Random variables (rv1, rv2)

Scaling was necessary due to differing units and ranges across features, which could otherwise bias the model.

5. Temporal Data Splitting

A chronological split was used to avoid data leakage:

- Training set: 15,788 records
  (2016-01-11 to 2016-04-30)
- Testing set: 3,947 records
  (2016-04-30 to 2016-05-27)

This method ensures that the model generalizes to unseen future data rather than memorizing patterns.

6.Preprocessed datasets saved :

- train_data_preprocessed.csv
- test_data_preprocessed.csv

Feature Engineering

1. Enhanced Time-Based Features

To capture cyclical and behavioural time patterns influencing energy consumption, we engineered 13 new time-based features:

- Cyclical Features: Transformed hour, day, and month into sine and cosine components to represent their periodic nature.
- Time of Day Binning: Segmented the day into Morning, Afternoon, Evening, and Night, followed by one-hot encoding.
- Business Logic Indicators:
  - is_working_hours (9 AM–5 PM)
  - is_peak_energy_hour (5 PM–7 PM)
  - weekend_evening_flag (Weekends, 6 PM–8 PM)

Justification: These features help the model learn time-based human behaviour, working schedules, and daily routines, which are critical drivers of residential energy consumption.

2. Rolling Averages and Moving Windows

We created 25 rolling features based on hourly statistics of energy and environmental variables:

- Rolling Mean/Std/Min/Max for various windows (1h, 3h, 6h, 12h, 24h)
- Applied across energy consumption (Appliances) and external variables (temperature, humidity, etc.)

Justification: These features smooth noise, reveal temporal consumption trends, and capture fluctuation levels, improving temporal pattern recognition in the model.

3. Autocorrelation Analysis and Lag Feature Selection

A detailed autocorrelation analysis on Appliances energy data revealed 129 significant lags, with the top 20 lags ranging from 10 minutes to 3 hours and 20 minutes showing correlations ≥ 0.35.

Strong autocorrelations indicate that recent energy usage significantly influences current consumption. This insight guided lag-based feature engineering.

4. Lagged and Derived Features

We constructed 21 lag and derived features:

- Lag Features: Appliances_lag_1 to Appliances_lag_15
- Difference Features: Changes over key time windows (diff_1, diff_6, diff_144)
- Ratio Features: Normalized energy shifts (ratio_1h, ratio_24h)

Justification: These features capture temporal dependencies, short-term trends, and energy usage volatility, supporting better forecasting.

5. Interaction Features Engineering

A total of 11 interaction features were created to capture cross-variable relationships:

- temp_humidity_T1_RH1 – Proxy for living room thermal comfort
- temp_humidity_T2_RH2 – Proxy for kitchen thermal comfort
- temp_humidity_T3_RH3 – Proxy for laundry room thermal comfort
- temp_humidity_T4_RH4 – Proxy for office room thermal comfort
- temp_humidity_T7_RH7 – Proxy for ironing room thermal comfort
- temp_humidity_T9_RH9 – Proxy for parent room thermal comfort
- outdoor_discomfort – Proxy for outdoor weather discomfort
- avg_indoor_temp – Mean of all indoor temperatures
- avg_indoor_humidity – Mean of all indoor relative humidities

- temp_diff_indoor_outdoor – Difference between average indoor temperature and outdoor temperature
- humidity_diff_indoor_outdoor – Difference between average indoor humidity and outdoor humidity
- lights_work_hour – Light usage during typical working hours (8 AM – 5 PM)
- lights_peak_hour – Light usage during energy peak hours (6 PM – 10 PM)
- weather_discomfort – Composite index of outdoor temperature, humidity, and visibility
- month_temp_interaction – Interaction between month (cyclical encoding) and outdoor temperature
- temp_efficiency – Ratio of average indoor temperature to appliance energy usage

Justification: These interactions reveal hidden patterns and dependencies that individual variables alone may not capture, such as temperature-humidity synergy or seasonal behavioural effects.

6. Domain-Specific Energy Features

Incorporating 13 domain-informed features helped improve the model's contextual awareness by embedding real-world energy usage patterns and seasonality. Here's a breakdown of the categories

Load Categories

These categorize the appliance energy consumption into discrete levels based on thresholds (e.g., quartiles or domain-specific cutoffs):

- load_Low – Binary flag indicating energy usage is below a defined low threshold.
- load_Medium – Indicates medium-range energy consumption.
- load_High – Indicates high energy usage periods.
- load_Very_High – Flags very high energy draw situations (e.g., multiple appliances running).

Purpose: Helps the model differentiate energy usage patterns during normal vs. peak operations.

HVAC Demand

Derived based on indoor and outdoor temperature differences:

- heating_demand – Estimated heating need (e.g., when indoor temp < set threshold and outdoor temp is low).
- cooling_demand – Estimated cooling need (e.g., when indoor temp > comfortable threshold and outdoor temp is high).

Purpose: Captures HVAC-related energy behavior, which can be a major load factor.

Energy Behavior Metrics

These features break down energy usage dynamics:

- base_load – Minimum appliance load over a time window (e.g., night usage), indicating always-on devices.
- variable_load – Difference between max and min usage, reflecting dynamic device behavior.
- energy_momentum – Short-term trend of energy usage (e.g., rolling mean or gradient).
- is_standby_power – Flag for low, non-zero usage possibly due to standby devices.

Purpose: Highlights predictable vs. fluctuating energy patterns.

Seasonal Flags

Capture the effect of climate and seasonality:

- is_winter_months – Flag for winter months (e.g., Nov–Feb), where heating needs may be higher.
- is_summer_months – Flag for summer months (e.g., May–Aug), associated with cooling demand.

Purpose: Embeds seasonal behavior patterns into the model.

Efficiency Measure

- appliance_efficiency – Ratio of actual energy usage to expected usage based on external and internal conditions (e.g., weather, occupancy).

Purpose: Identifies whether current energy use is efficient compared to norm or if anomalies exist.

7. Feature Selection via Correlation Analysis

- Initial Features: 118
- After Filtering (correlation ≥ 0.01): 103
- Top Correlated Features (with target Appliances):
    - variable_load (0.997)
    - load_Very_High (0.883)
    - Appliances_lag_1 (0.768)
    - Appliances_rolling_max_1h (0.761)
- Multicollinearity Check: 11 pairs found
- Final Features Selected: 93

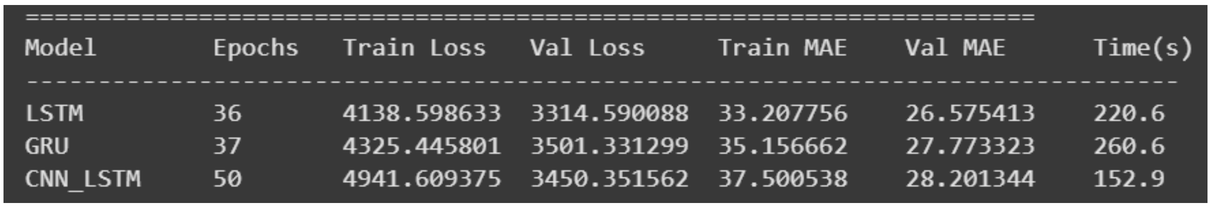
Justification: Correlation filtering ensured that only informative features were retained. Removing multicollinear features helped reduce overfitting and improve model interpretability.

Model Design:

Model Choice: **LSTM**

Reasons for Choosing LSTM:

- Best Validation Performance: Lowest validation loss (3314.59) and MAE (26.58), indicating better generalization.(Figure 7)



```
=====================================================================
Model         Epochs    Train Loss   Val Loss     Train MAE    Val MAE      Time(s)
--------------------------------------------------------------------------------
LSTM          36        4138.598633  3314.590088  33.207756    26.575413    220.6
GRU           37        4325.445801  3501.331299  35.156662    27.773323    260.6
CNN_LSTM      50        4941.609375  3450.351562  37.500538    28.201344    152.9
```

Figure 7

- Stable Training: Required fewer epochs (36) to converge than GRU or CNN-LSTM.
- Balanced Complexity: Two-layer LSTM with dropout provides enough capacity to learn without overfitting.
- Simpler Data Handling: Requires minimal reshaping (timestep = 1), avoiding data loss from sequence generation.
- Familiarity and Proven Success: Builds upon previous successful models and practices, increasing confidence.
- Interpretability and Flexibility: Easier to understand and extend than hybrid models like CNN-LSTM.

LSTM Model Architecture

The model is a simple LSTM-based neural network designed for regression, with each input sample treated as a single timestep.

- Input Layer:
  Input shape is (1, num_features), preserving all engineered features.
- LSTM Layer 1 (64 units):
  Uses tanh activation and sigmoid for internal gates. return_sequences=True to pass outputs to the next LSTM layer.
- Dropout Layer (rate = 0.2):
  Reduces overfitting by randomly deactivating 20% of neurons during training.
- LSTM Layer 2 (32 units):
  Also uses tanh activation. return_sequences=False to output the final state only.

- Dropout Layer (rate = 0.2):
  Further regularization.
- Dense Output Layer (1 unit):
  No activation (linear), suitable for regression output.

Design Justifications

LSTM Layers:

- LSTM(64) → LSTM(32):
  - Progressive reduction of units enables the model to compress learned patterns into a more compact latent space.
  - Helps in reducing overfitting by lowering model complexity.
- Two LSTM layers:
  - Captures both shallow and deeper temporal patterns without introducing unnecessary depth.
  - Deeper models showed no performance improvement but increased overfitting risk.

Dropout Layers:

- Dropout rate of 0.2 used after each LSTM layer:
  - Helps prevent overfitting by randomly deactivating neurons during training.
  - Maintains generalization ability on unseen test data.

Dense Output Layer:

- Single neuron output (Dense(1)):
  - Suitable for regression problems (predicting continuous values).
  - Direct mapping from LSTM representations to predicted energy consumption.

Compilation Choices

Optimizer – Adam:

- Why Adam?
  - Combines the benefits of AdaGrad (adaptive learning rate) and RMSprop (efficient in online settings).
  - Includes momentum, which speeds up convergence and avoids local minima.
  - Proven effective for training deep networks, especially LSTMs.
- Learning Rate = 0.001:
  - A safe default that works well in most deep learning scenarios.
  - Provides a balance between convergence speed and stability.

Loss Function – Mean Squared Error (MSE):

- Why MSE?
  - Measures the average squared difference between predicted and actual values.
  - Penalizes larger errors more strongly, encouraging the model to focus on minimizing big prediction mistakes.

- Common choice for regression tasks.

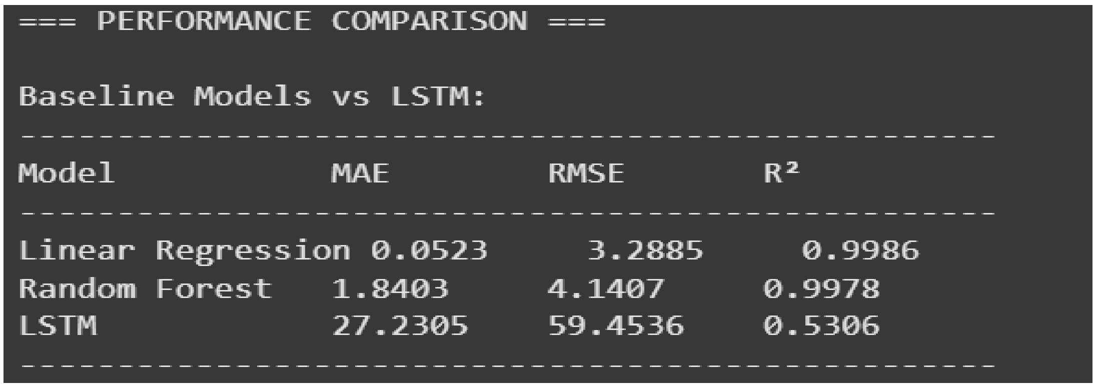
Evaluation Metric – Mean Absolute Error (MAE):

- Why MAE?
    - Measures average absolute error in the same unit as the target (energy usage).
    - Less sensitive to outliers compared to MSE.
    - Provides an intuitive, interpretable performance indicator for model evaluation.

Results:

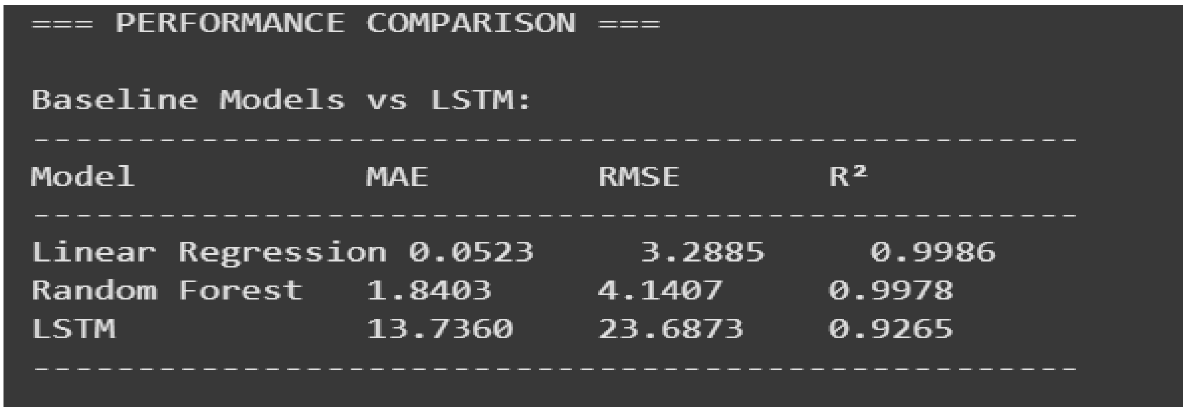
Results Tables

First Model



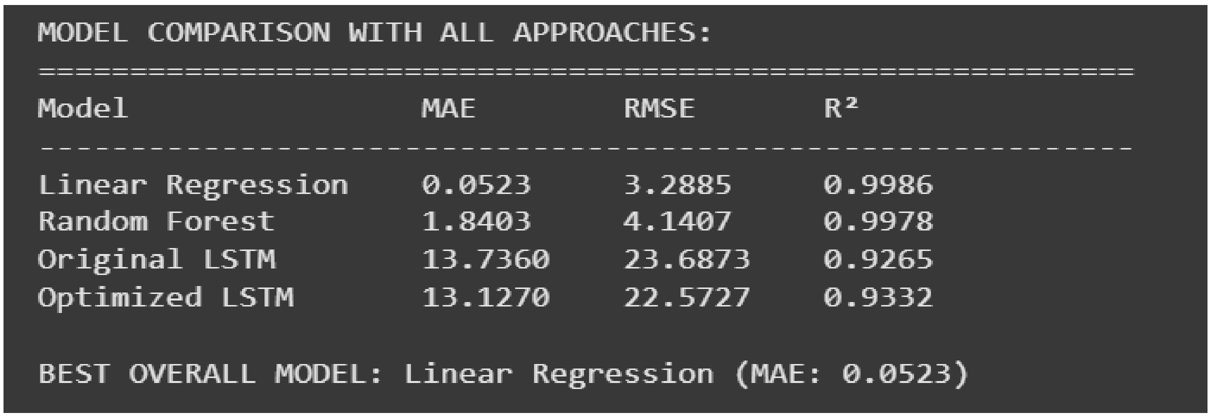
```
=== PERFORMANCE COMPARISON ===

Baseline Models vs LSTM:
--------------------------------------------------
Model           MAE        RMSE       R²
--------------------------------------------------
Linear Regression 0.0523     3.2885     0.9986
Random Forest   1.8403     4.1407     0.9978
LSTM            27.2305    59.4536    0.5306
--------------------------------------------------
```

Second Model



```
=== PERFORMANCE COMPARISON ===

Baseline Models vs LSTM:
--------------------------------------------------
Model           MAE        RMSE       R²
--------------------------------------------------
Linear Regression 0.0523     3.2885     0.9986
Random Forest   1.8403     4.1407     0.9978
LSTM            13.7360    23.6873    0.9265
--------------------------------------------------
```

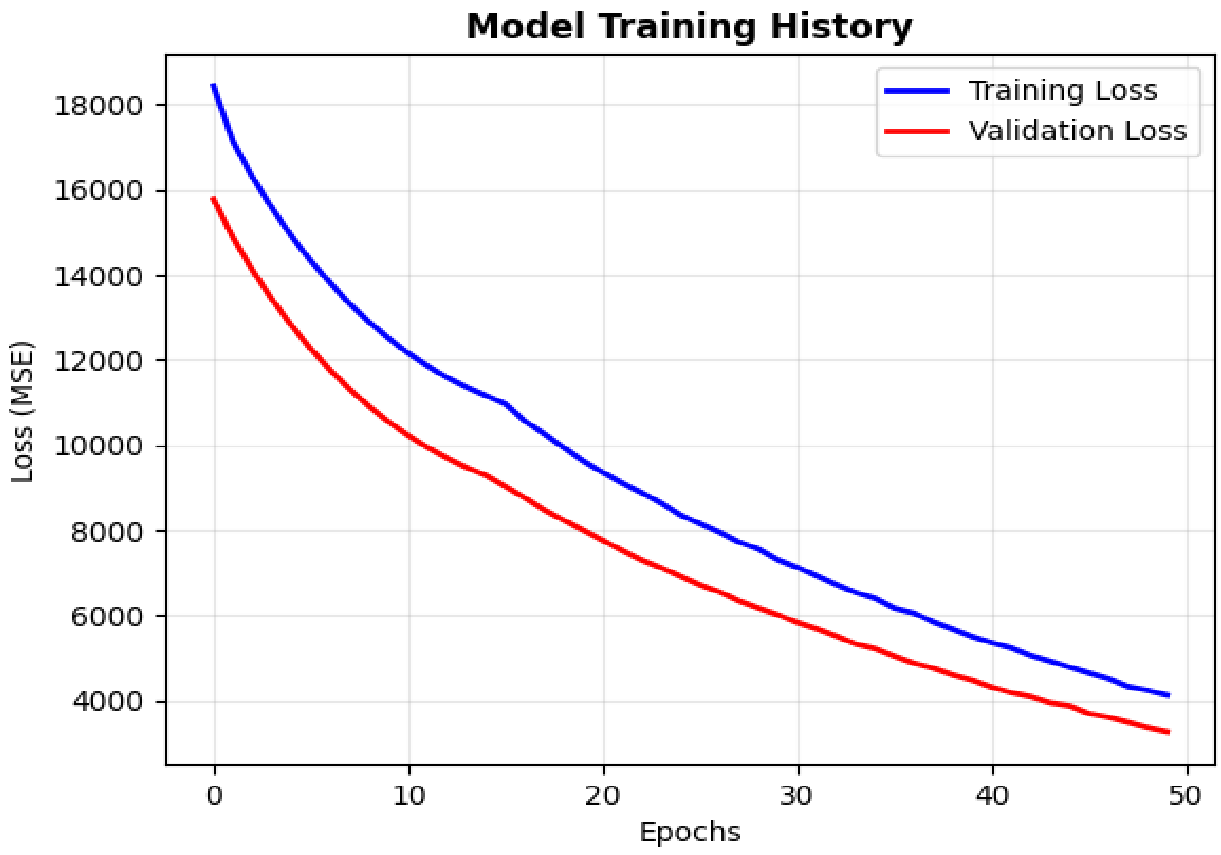
Final Model



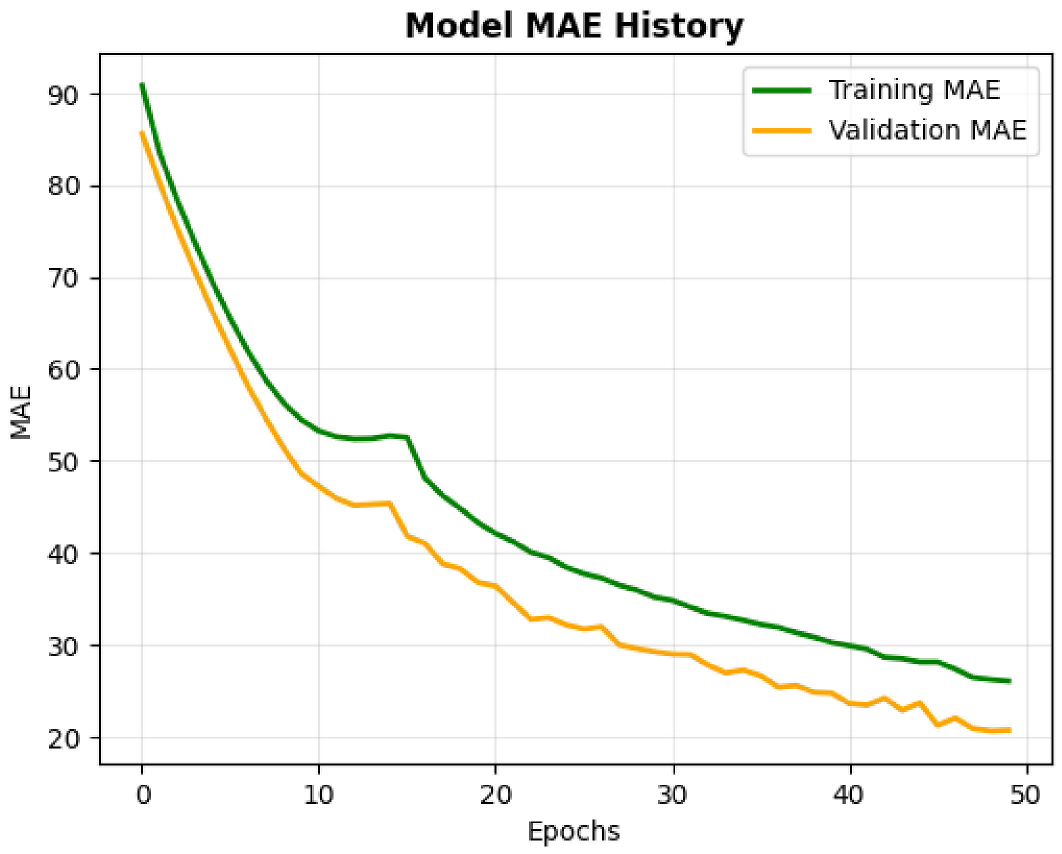
```
MODEL COMPARISON WITH ALL APPROACHES:
============================================================
Model                    MAE        RMSE       R²
------------------------------------------------------------
Linear Regression        0.0523     3.2885     0.9986
Random Forest            1.8403     4.1407     0.9978
Original LSTM            13.7360    23.6873    0.9265
Optimized LSTM           13.1270    22.5727    0.9332

BEST OVERALL MODEL: Linear Regression (MAE: 0.0523)
```

Plots

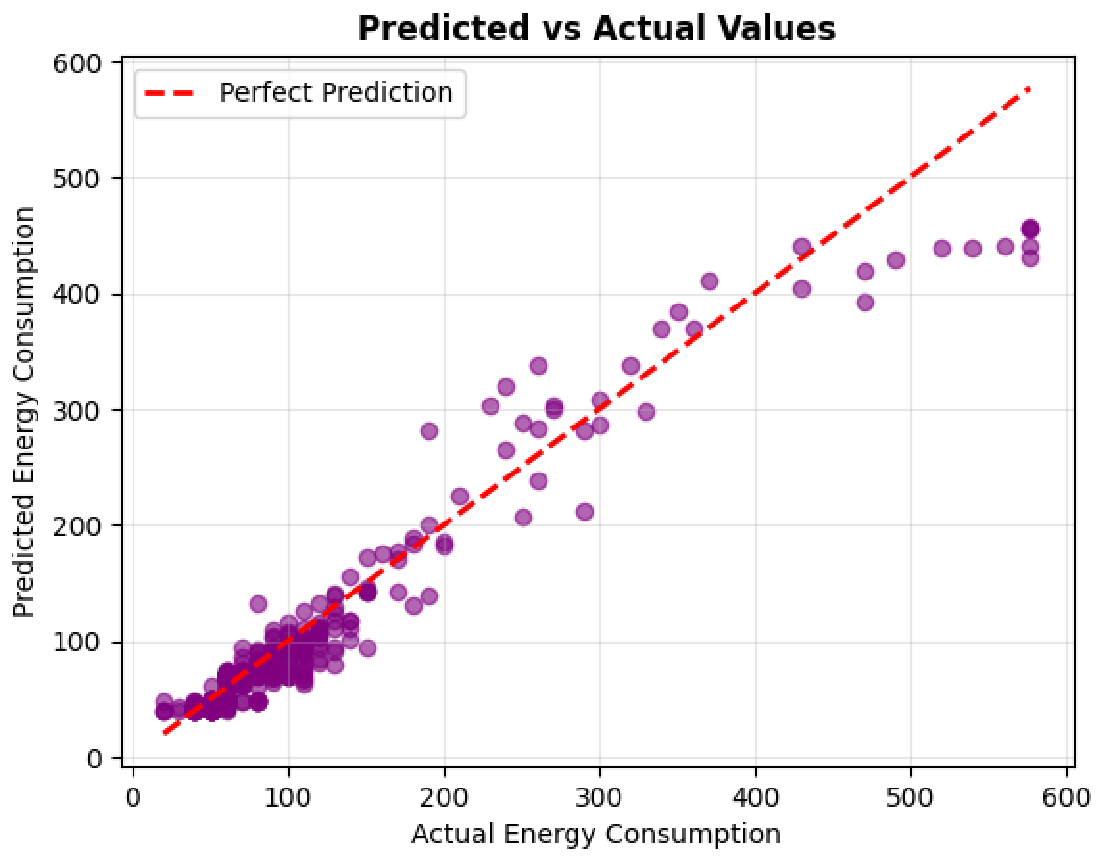
Training History - Loss Curves



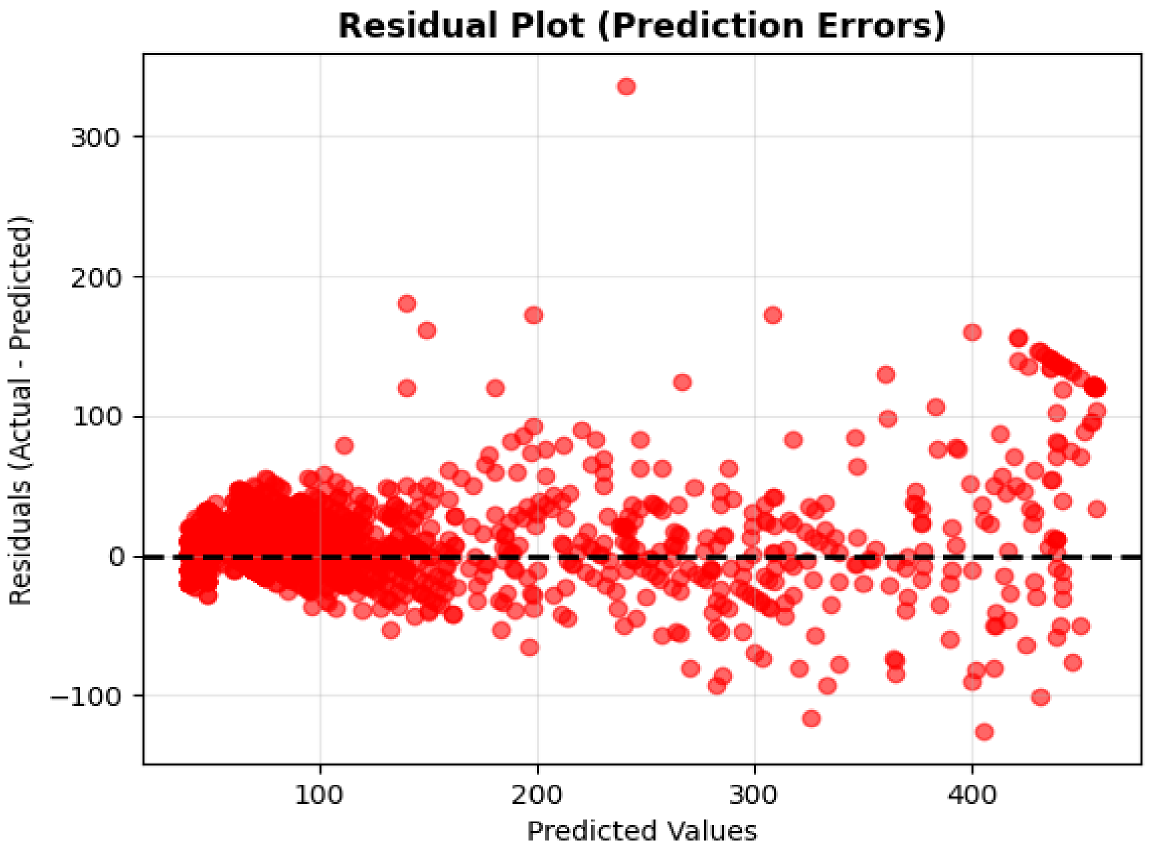
Training History – MAE



Predicted vs Actual Values



Residual Plot



Model Optimization:

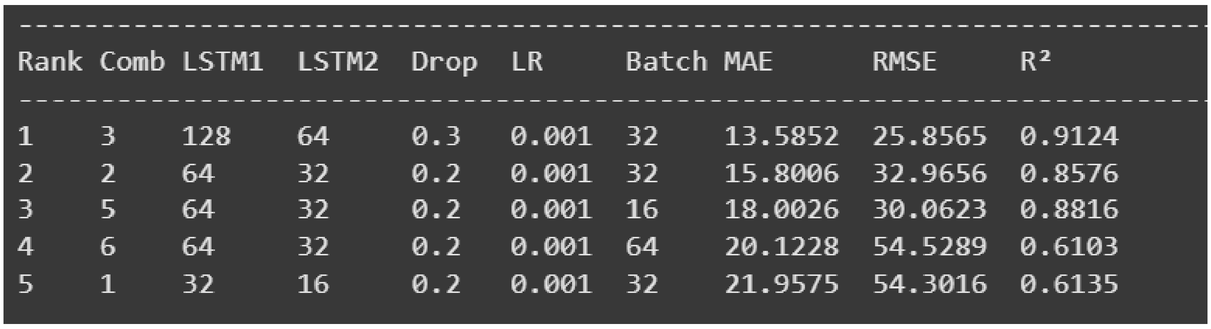
This project explores two model optimization approaches: Hyperparameter Search and Advanced Regularization Techniques. Both methods were implemented and compared to evaluate their effectiveness in improving model performance and generalization.

Smart Hyperparameter Search

Method Overview

Instead of testing all possible combinations (which is time-consuming and computationally expensive), this approach:

- Selects 6 promising hyperparameter combinations based on practical experience and common tuning strategies.
- Trains and evaluates each combination to find the best model in terms of MAE, RMSE, and R^2.
- Uses a fixed number of epochs (50) for comparability and efficiency.
- Captures training time and performance metrics for fair comparison.



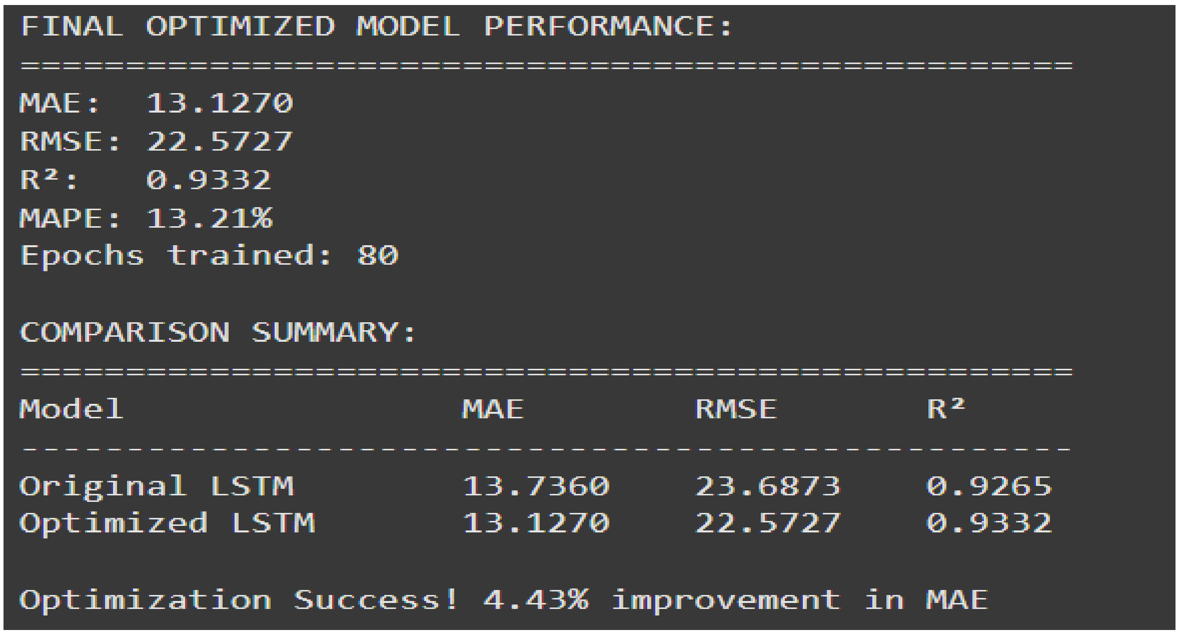
```
------------------------------------------------------------------------------------
Rank Comb LSTM1  LSTM2  Drop  LR     Batch MAE      RMSE     R²
------------------------------------------------------------------------------------
1    3    128    64     0.3   0.001  32    13.5852  25.8565  0.9124
2    2    64     32     0.2   0.001  32    15.8006  32.9656  0.8576
3    5    64     32     0.2   0.001  16    18.0026  30.0623  0.8816
4    6    64     32     0.2   0.001  64    20.1228  54.5289  0.6103
5    1    32     16     0.2   0.001  32    21.9575  54.3016  0.6135
```

Figure 8

Best Model: Combination 3

- Architecture:
  - LSTM Layer 1 Units: 128
  - LSTM Layer 2 Units: 64
  - Dropout Rate: 0.3
- Training Settings:
  - Learning Rate: 0.001
  - Batch Size: 32
  - Epochs: 50
- Performance:
  - MAE: 13.5852
  - RMSE: 25.8565
  - R^2: 0.9124
  - Training Time: 149.3 seconds

Final Optimized Model Performance (using 100 epochs)



```
FINAL OPTIMIZED MODEL PERFORMANCE:
==================================================
MAE:  13.1270
RMSE: 22.5727
R²:   0.9332
MAPE: 13.21%
Epochs trained: 80

COMPARISON SUMMARY:
==================================================
Model                MAE        RMSE       R²
--------------------------------------------------
Original LSTM        13.7360    23.6873    0.9265
Optimized LSTM       13.1270    22.5727    0.9332

Optimization Success! 4.43% improvement in MAE
```

Advanced Regularization Techniques (Talukdar, 2025)

This section investigates whether applying different dropout rates to the optimized LSTM model could further improve performance by reducing overfitting and enhancing generalization. Dropout is a regularization technique that randomly "drops out" units during training to prevent the model from becoming too reliant on any one neuron.

What Happened

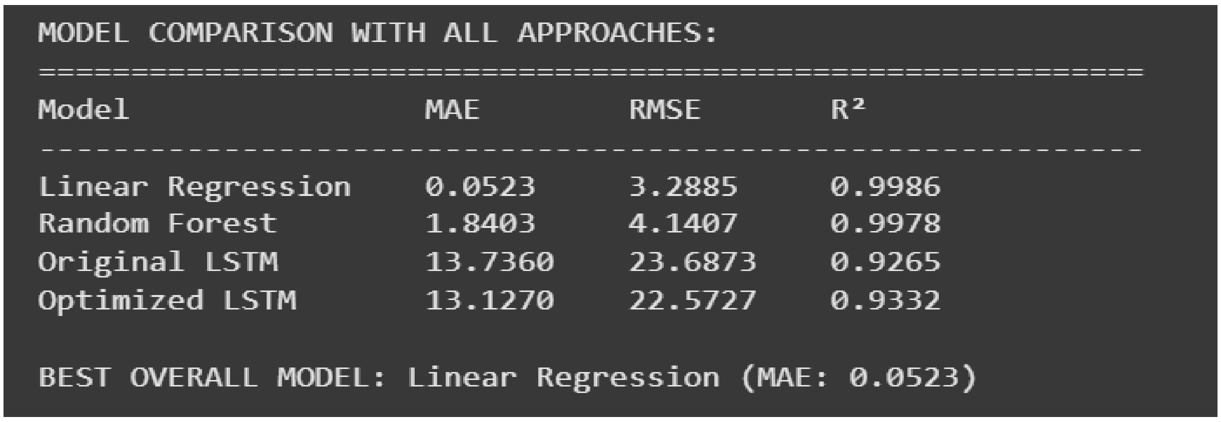
- The code first checked if a previously optimized model exists.
- A range of dropout rates [0.1, 0.2, 0.3, 0.4, 0.5] were tested.
- For each dropout rate:
    - A new LSTM model was built using the best-found hyperparameters.
    - The model was trained for 50 epochs.
    - The model's Mean Absolute Error (MAE) on the test set was recorded.
- After testing all dropout values, the one with the lowest MAE was selected as the best dropout rate.
- The best result was compared with the original optimized model to check for performance improvement.

Results

- Best dropout rate: 0.1
- Best MAE from dropout testing: 13.1517
- Original optimized model MAE: 13.1270
- Since the MAE with dropout rate 0.1 was slightly worse than the original model's MAE, the original dropout setting was retained.

Insights from Regularization Testing

- Dropout rates 0.3 to 0.5 slightly worsened performance, possibly due to underfitting.
- Lower dropout rates (0.1 – 0.2) performed better for this dataset, indicating that light regularization is more suitable.
- No further improvement was achieved through additional dropout tuning; however, this test validated the robustness of the original configuration.



```
MODEL COMPARISON WITH ALL APPROACHES:
============================================================
Model                    MAE         RMSE        R²
------------------------------------------------------------
Linear Regression        0.0523      3.2885      0.9986
Random Forest            1.8403      4.1407      0.9978
Original LSTM            13.7360     23.6873     0.9265
Optimized LSTM           13.1270     22.5727     0.9332

BEST OVERALL MODEL: Linear Regression (MAE: 0.0523)
```

Discuss improvements

At the beginning of the project, I started with building a baseline LSTM model to predict appliance energy consumption using multivariate time-series data. The initial model had the following performance:

- MAE: 13.7360
- RMSE: 23.6873
- R^2: 0.9265

This result showed that while the LSTM could model the sequence of energy consumption to some extent, the error was still relatively high.

To improve the model, I applied hyperparameter tuning by adjusting key parameters like the number of LSTM units, dropout rate, learning rate, and batch size. After tuning, I retrained the model and achieved better performance:

- MAE: 13.1270
- RMSE: 22.5727
- R^2: 0.9332

This was a 4.43% improvement in MAE, and the model showed better generalization. The final optimized parameters were:

- lstm_units_1: 128
- lstm_units_2: 64
- dropout_rate: 0.3
- learning_rate: 0.001
- batch_size: 32

Challenges and Solutions:

1. Capturing Complex Temporal Patterns

- Challenge: Traditional models fail to capture hourly, daily, and weekly patterns in appliance usage.
- Solution: Engineered cyclical time-based features (e.g., sine/cosine of hour, day, month) and domain-specific temporal indicators like is_working_hours, is_peak_energy_hour.

2. Multicollinearity Among Features

- Challenge: Many features, especially temperature and humidity sensors, were highly correlated, risking model instability and overfitting.
- Solution: Conducted correlation analysis, identified redundant features (e.g., T1–T3–T5), and proposed feature reduction methods like correlation thresholding and PCA.

3. Presence of Outliers

- Challenge: Extreme values in appliance energy consumption could skew model training and evaluation.
- Solution: Applied IQR and Z-score methods to detect outliers; implemented capping at the 1st and 99th percentiles to retain data while reducing skewness.

4. Irrelevant and Noisy Features

- Challenge: Features like rv1 and rv2 were found to be irrelevant (random noise).
- Solution: Removed non-informative features to improve model efficiency and prevent noise interference.

5. Feature Scaling for Neural Networks

- Challenge: Features had different units and scales, which could bias deep learning models.
- Solution: Applied Z-score normalization using StandardScaler to all continuous numerical features.

6. Enhancing Predictive Power Through Feature Engineering

- Challenge: Raw features were insufficient to capture complex user behavior and environmental effects.
- Solution:
    - Created rolling window features (mean, std, min, max over 1h–24h).
    - Generated lag and difference features for autocorrelation exploitation.
    - Added interaction features between temperature, humidity, and time.
    - Designed domain-informed features for HVAC demand and load categorization.

7. Weak Predictive Power of Humidity and Weather

- Challenge: Most humidity and weather features showed low correlation with energy usage.
- Solution: Kept only moderately useful features, deprioritized weak features during modeling.

8. Encoding Human Behavioral Patterns

- Challenge: Appliance usage is influenced by human behavior (e.g., weekends, evenings).
- Solution: Engineered features like:
    - weekend_evening_flag
    - is_weekend
    - time_of_day bins (morning, afternoon, evening, night)

9. Model Input Optimization

- Challenge: Raw data alone doesn't capture complex consumption trends.
- Solution: Transformed dataset into a feature-rich, clean, normalized, and behavior-aware format ready for deep learning models.

10. Autocorrelation and Lag Analysis

- Challenge: Need to capture temporal dependencies.
- Solution: Conducted autocorrelation analysis and created significant lag-based features and energy ratios over past time windows.

11. Model Evaluation and Generalization

- Challenge: Ensuring the model performs well on unseen data.
- Solution: Evaluated using MAE, RMSE, and R^2; also compared against baseline models (Linear Regression, Random Forest).

12. Computational Resource Management

- Challenge: Training deep learning models on large data required efficient usage of memory and time.
- Solution: Used batch processing, efficient data loading from Google Drive, and reduced feature dimensionality.

Conclusion:

Key Findings

- The Original LSTM model achieved good performance (MAE: 13.736, R^2: 0.927), but was outperformed by traditional models.
- Hyperparameter tuning improved the LSTM model by 4.43% (MAE reduced to 13.127, R^2 increased to 0.933), showing effective optimization.
- Despite improvements, Linear Regression was the best overall model with the lowest MAE (0.0523) and highest R^2 (0.9986), indicating simpler models may suffice for this task.
- Random Forest also performed well (MAE: 1.8403, R^2: 0.9978), validating tree-based approaches for energy prediction.
- The results suggest that while deep learning (LSTM) can learn temporal patterns, classical models still provide strong baseline predictions in this dataset.

Potential Areas For Future Work

- Investigate feature engineering and selection techniques to enhance model input quality and improve predictions.
- Implement ensemble methods combining classical and deep learning models to leverage their strengths.
- Test the models on larger and more diverse datasets for better generalization.
- Incorporate real-time data streaming and online learning for dynamic appliance energy prediction.

References

geeksforgeeks. (2024, 05 26). *geeksforgeeks*. Retrieved from https://www.geeksforgeeks.org/time-series-forecasting-using-tensorflow/

geeksforgeeks. (2025, 05 10). *geeksforgeeks*. Retrieved from https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/

Talukdar, S. (2025, 03 19). *Appliance Energy Prediction using Time Series Forecasting: A Comparative Analysis of Different Machine Learning Algorithms*. Retrieved from researchgate.net: https://www.researchgate.net/publication/390155934_Appliance_Energy_Prediction_using_Time_Series_Forecasting_A_Comparative_Analysis_of_Different_Machine_Learning_Algorithms

Tzelepi, M. (2023, 06 15). *Deep Learning for Energy Time-Series Analysis and Forecasting*. Retrieved from https://www.researchgate.net/publication/371605583_Deep_Learning_for_Energy_Time-Series_Analysis_and_Forecasting