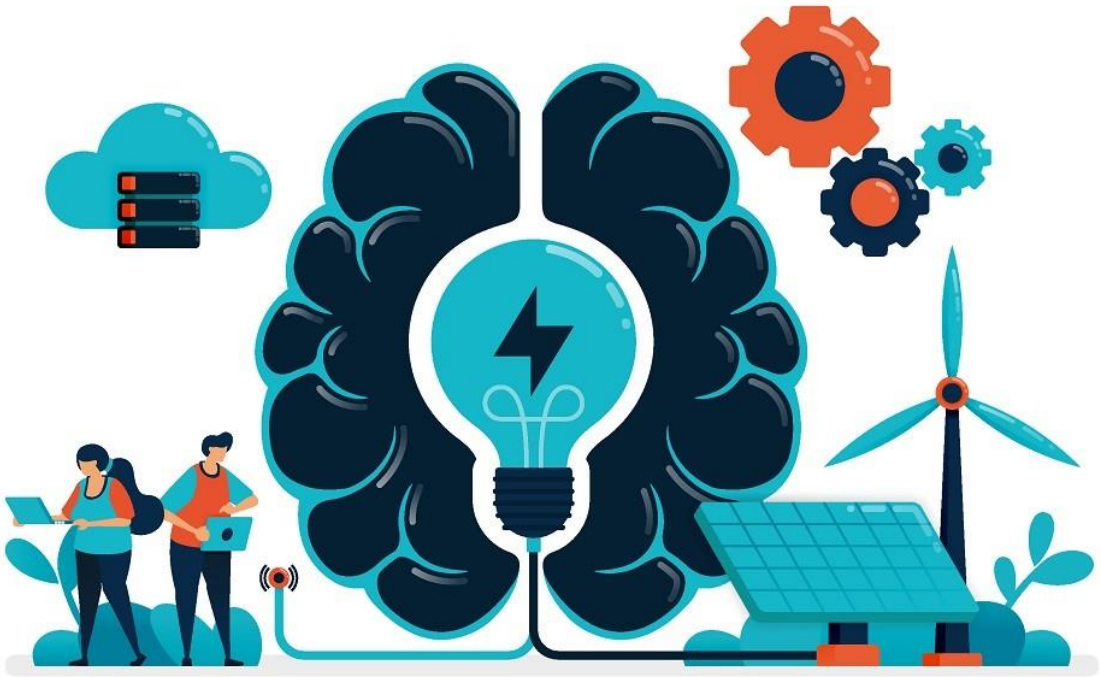


Appliance Energy Prediction Using the Energy Dataset

Divanka Samarasinghe

Date - 30.01.2025

Contents

1. Introduction

In the modern world, energy efficiency is essential, which makes realistic energy consumption predictions necessary. The goal of this evaluation is to use a multivariate time-series dataset to forecast appliance energy use. The dataset records energy-related, time-based, and environmental information at 10-minute intervals.

The goal is to use deep learning techniques to preprocess, analyze, and model this data. The objective is to create a predictive system that offers helpful knowledge and supports more intelligent energy management solutions by utilizing feature engineering and powerful architecture.

2. Data Insights

2.1. Dataset Overview

- Dataset Shape: With 19,735 rows and 29 columns, the dataset offers a significant amount of information for analysis.
- Null Values: The dataset was complete because no missing values were discovered.
- Duplicate Entries: The integrity of the data is confirmed by the absence of duplicate rows.

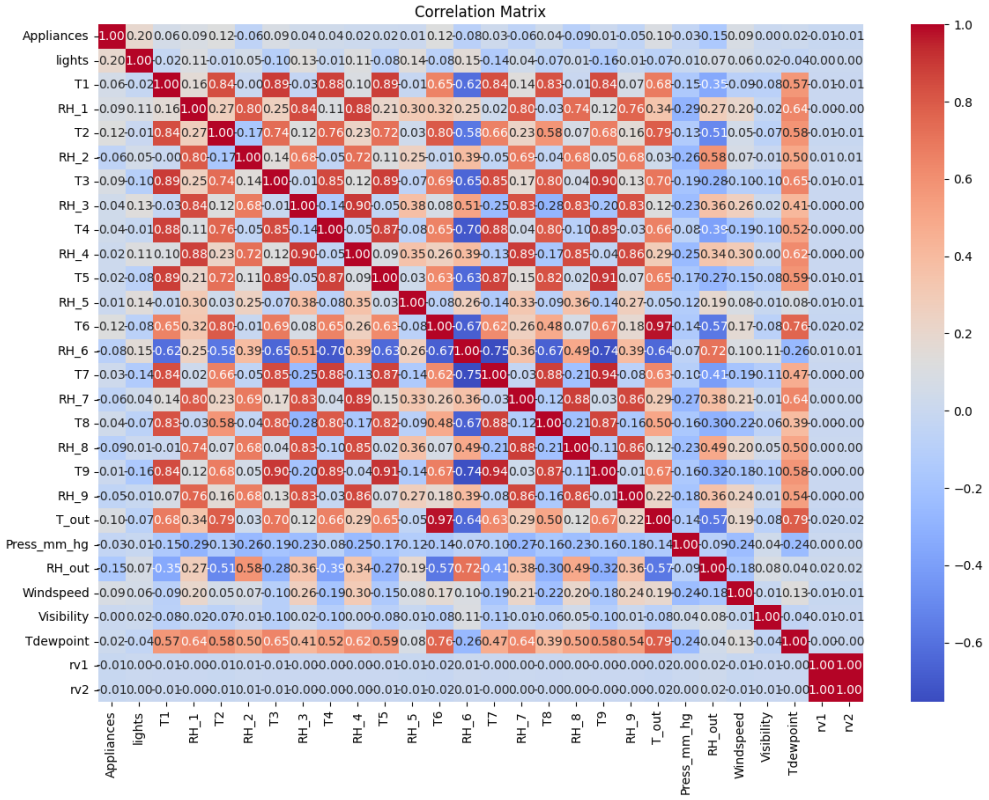
2.2. Summary Statistics

- The mean, median, and standard deviation of key features were analyzed to understand their distribution.

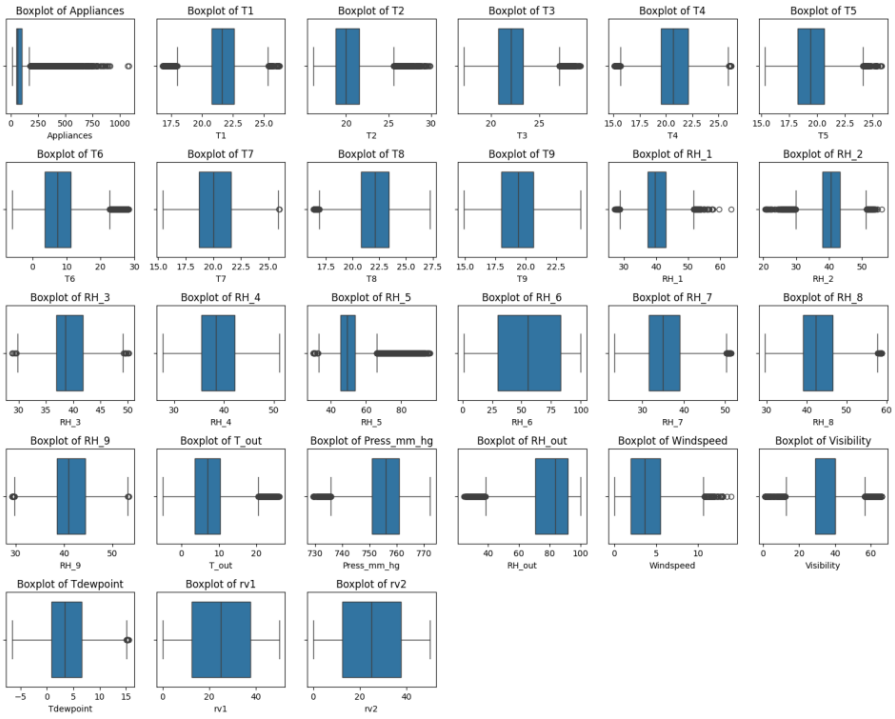
```
Summary Statistics:
         Appliances        lights            T1          RH_1            T2  \
count  19735.000000  19735.000000  19735.000000  19735.000000  19735.000000
mean      97.694958      3.801875     21.686571     40.259739     20.341219
std      102.524891      7.935988      1.606066      3.979299      2.192974
min       10.000000      0.000000     16.790000     27.023333     16.100000
25%       50.000000      0.000000     20.760000     37.333333     18.790000
50%       60.000000      0.000000     21.600000     39.656667     20.000000
75%      100.000000      0.000000     22.600000     43.066667     21.500000
max     1080.000000     70.000000     26.260000     63.360000     29.856667

              RH_2            T3          RH_3            T4          RH_4  \
count  19735.000000  19735.000000  19735.000000  19735.000000  19735.000000
mean      40.420420     22.267611     39.242500     20.855335     39.026904
std        4.069813      2.006111      3.254576      2.042884      4.341321
min       20.463333     17.200000     28.766667     15.100000     27.660000
25%       37.900000     20.790000     36.900000     19.530000     35.530000
50%       40.500000     22.100000     38.530000     20.666667     38.400000
75%       43.260000     23.290000     41.760000     22.100000     42.156667
max       56.026667     29.236000     50.163333     26.200000     51.090000

       ...            T9          RH_9         T_out   Press_mm_hg  \
count  ...  19735.000000  19735.000000  19735.000000  19735.000000
mean   ...     19.485828     41.552401      7.411665    755.522602
std    ...      2.014712      4.151497      5.317409      7.399441
min    ...     14.890000     29.166667     -5.000000    729.300000
25%    ...     18.000000     38.500000      3.666667    750.933333
50%    ...     19.390000     40.900000      6.916667    756.100000
75%    ...     20.600000     44.338095     10.408333    760.933333
max    ...     24.500000     53.326667     26.100000    772.300000

            RH_out     Windspeed    Visibility     Tdewpoint          rv1  \
count  19735.000000  19735.000000  19735.000000  19735.000000  19735.000000
mean      79.750418      4.039752     38.330834      3.760707     24.988033
std       14.901088      2.451221     11.794719      4.194648     14.496634
min       24.000000      0.000000      1.000000     -6.600000      0.005322
25%       70.333333      2.000000     29.000000      0.900000     12.497889
50%       83.666667      3.666667     40.000000      3.433333     24.897653
75%       91.666667      5.500000     40.000000      6.566667     37.583769
max      100.000000     14.000000     66.000000     15.500000     49.996530

               rv2
count  19735.000000
mean      24.988033
std       14.496634
min        0.005322
25%       12.497889
50%       24.897653
75%       37.583769
max       49.996530

[8 rows x 28 columns]
```

2.3. Insights from Visualizations

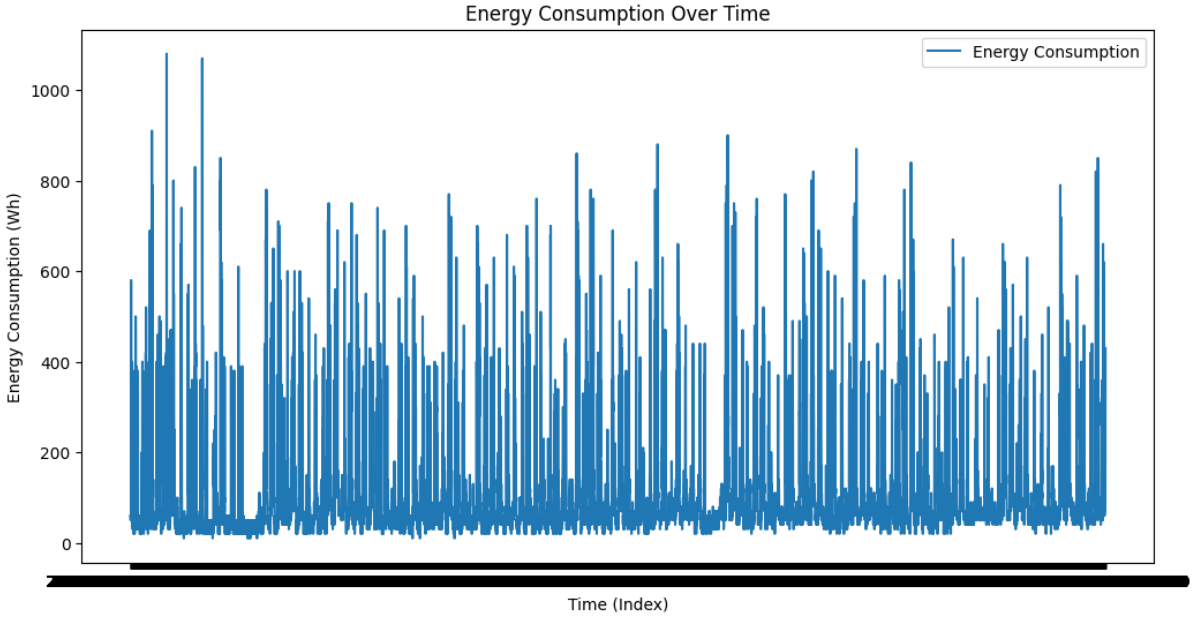
- **Correlation Heatmap**: Highlighted strong positive relationships between key variables, aiding in feature selection for modeling.



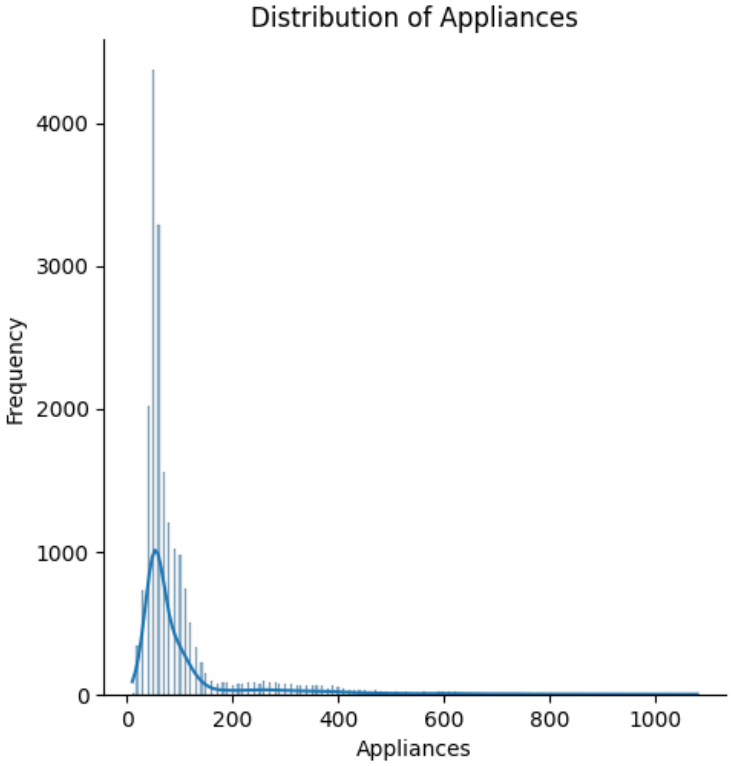
- **Boxplots**: Identified and visualized outliers in specific features before applying capping.



- **Line Chart**: Showcased the trends in appliance usage over time, helping to identify patterns and fluctuations in demand, which are useful for forecasting and analysis.



- **Displot**: Identified outliers, skewness, and the general distribution shape for improved data interpretation. It also visualized the distribution of appliance usage data, offering insights into the frequency and spread of usage.



3. Preprocessing

3.1. Handling Null Values

To deal with missing data, I first determined how many of each column, including important ones, were having missing values. But there were no missing values in the entire dataset.

```
Missing Values in Each Column:
 date           0
Appliances      0
lights          0
T1              0
RH_1            0
T2              0
RH_2            0
T3              0
RH_3            0
T4              0
RH_4            0
T5              0
RH_5            0
T6              0
RH_6            0
T7              0
RH_7            0
T8              0
RH_8            0
T9              0
RH_9            0
T_out           0
Press_mm_hg     0
RH_out          0
Windspeed       0
Visibility      0
Tdewpoint       0
rv1             0
rv2             0
dtype: int64
```

3.2. Data Cleaning

Analyzing the dataset to make sure it was consistent and complete was the next step. To avoid duplication in the analysis, I looked for and removed duplicate rows. But there were no duplicated records as well.

```
Duplicate Rows:
Empty DataFrame
Columns: [date, Appliances, lights, T1, RH_1, T2, RH_2, T3, RH_3, T4, RH_4, T5, RH_5, T6, RH_6, T7, RH_7, T8, RH_8, T9, RH_9, T_out, Press_mm_hg, RH_out, Windspeed, Visibility, Tdewpoint, rv1, rv2]
Index: []

[0 rows x 29 columns]
```

3.3. Outliers Detection and Treatment

A number of independent features had outliers, and in order to reduce their effect on model performance, the capping method was used, which substituted the threshold value for values that were above the threshold, keeping the data within a reasonable range without skewing the distribution as a whole.

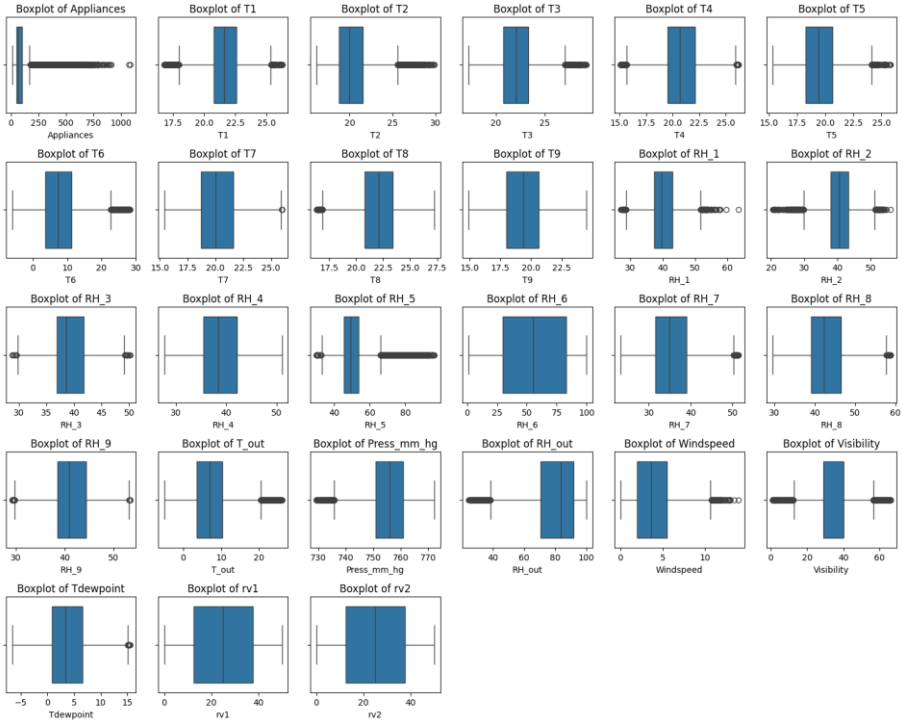


Figure 1: Before Applying Capping (With Outliers)

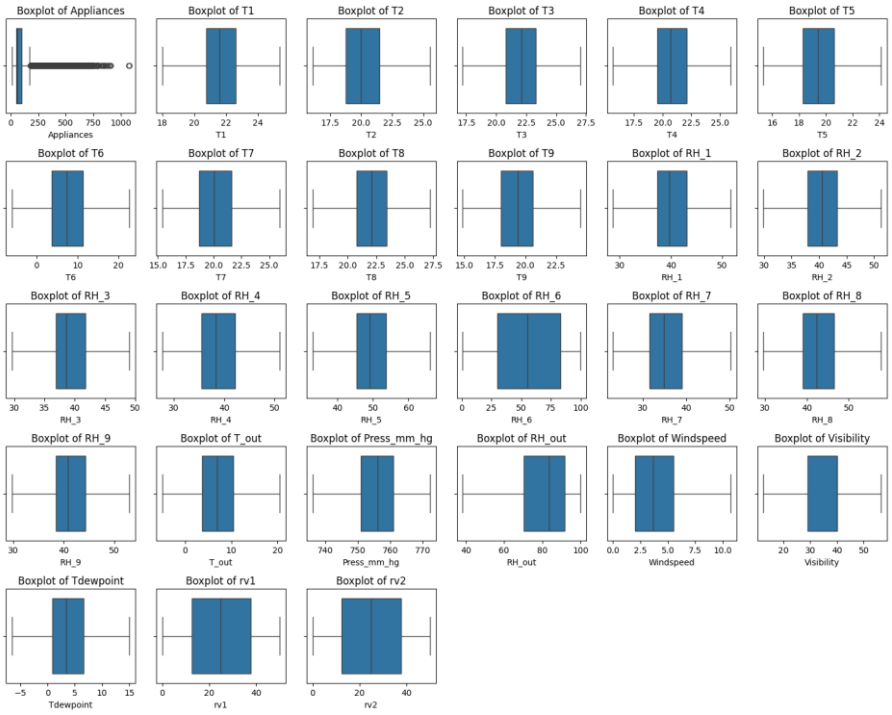


Figure 2: After Applying Capping (Without Outliers)

3.4. Feature Scaling

Standardization was selected for feature scaling due to the dataset's characteristics and the deep learning model that will be employed. Using this method, the features are changed to have a mean of 0 and a standard deviation of 1. For this kind of data, standardization is especially appropriate since it guarantees that every feature makes an equal contribution to the model's performance and helps keep any one feature from controlling the learning process because of scale discrepancies.
The deep learning model can learn more effectively by standardizing the features, which will result in quicker training convergence and more precise predictions.

4. Feature Engineering

4.1. Features created and selected

4.1.1. Time-Based Features

- Hour: Derived the timestamp's hour of the day.
  - ✓ Relevance: Records daily consumption trends, which can change dramatically during the day.
- Day of week: Indicated by the date (Monday = 0, Sunday = 6)
  - ✓ Relevance: Helpful in determining trends in the data between weekdays and weekends.
- Weekend: A binary characteristic that indicates if it's a weekend or not.
  - ✓ Relevance: Records the changes in appliance usage patterns between workdays and weekends.

4.1.2. Lagged Features

- Appliances Lag 10, 30, 60: Target variable lagged versions for 10, 30, and 60 minutes were created.
  - ✓ Relevance: Helps in the model's learning of autocorrelations and temporal dependencies in appliance usage.

4.1.3. Statistical Rolling Features

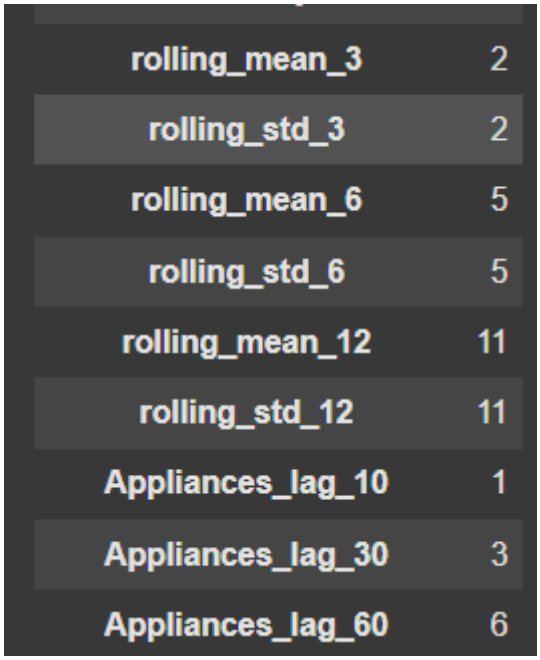
- Rolling Mean & Standard Deviation (3, 6, 12): This function calculates the rolling mean and standard deviation over 3, 6, and 12 intervals.
  - ✓ Relevance: It helps to smooth out noise and discover patterns by capturing local trends and variability in the time series.

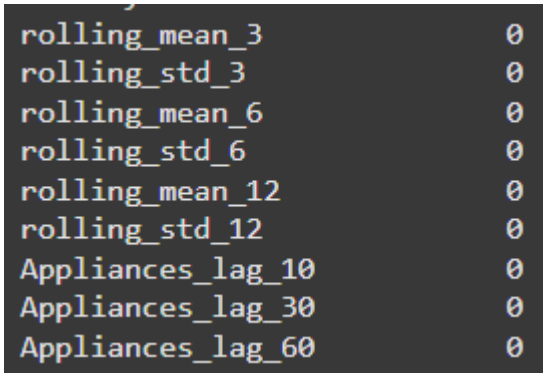
4.1.4. Interaction Features

- T1_RH_1 Interaction: Indoor temperature (T1) and indoor humidity (RH_1) are multiplied.
- T_out_RH_out Interaction: Outdoor temperature (T_out) and outdoor humidity (RH_out) are multiplied.
- T1_T2 Interaction: Relationship between T1 and T2 indoor temperatures.
- T2_RH_2 Interaction: Relationship between humidity (RH_2) and temperature (T2).
  - ✓ Relevance: These characteristics capture intricate connections that could affect energy usage.

4.2. Handling Null Values

After implementing new features, it was found that some columns contained null values, which had been created during feature engineering. The median was used to fill these null values because the features were skewed. The median was chosen because it is more resilient to outliers and better preserves the distribution of skewed attributes.





4.3. Feature Selection

4.3.1. Methods used

1. Recursive Feature Elimination (RFE):
   Random Forest was used for feature selection, which involved recursively deleting less significant features. RFE assesses the value of each trait and eliminates those with poor significance, allowing the most influential predictors to remain.

2. Correlation Analysis:
   High correlation features were found and removed in order to reduce multicollinearity. Features with correlation coefficients greater than a predefined threshold were eliminated to prevent overfitting owing to predictor redundancy.

3. Tree-Based Importance:
   Once again, Random Forest was used to calculate feature importance. This strategy assists in determining the most important features based on their contribution to the model's decision-making process. Features of greater relevance were maintained.

4.3.2. Final Feature Selection

By merging the results of the three approaches discussed above, the final set of features was chosen by taking the intersection of the features indicated as significant in all three ways. This guarantees that only the most significant and non-redundant features are included in the final dataset, hence improving the model's performance and understanding.

4.3.3. Scaling of Newly Created Features

After selecting the final features, scaling was applied to some newly constructed columns, particularly those with varied scales that could impair model performance. Standardization (Z-score scaling) was employed to convert these attributes into a mean of 0 and a standard deviation of 1. This allows the deep learning model to train more rapidly while also ensuring that all characteristics contribute equally to the prediction process.

The scaled features were saved in a new data frame and used for additional model training.

5. Model Design

In this case study, two baseline models—Linear Regression and Random Forest Regressor—were tested, followed by three deep learning models—LSTM, CNN-LSTM, and GRU—to forecast energy consumption (Appliances) using the available dataset.

5.1. Baseline Models

5.1.1. Linear Regression Model

Linear regression is a straightforward and understandable technique that is appropriate for evaluating the linear correlations between data and the target variable (energy consumption). The performance of the linear regression model was assessed using the **Mean Squared Error (MSE)** and **R-squared (R2)** metrics.

5.1.2. Random Forest Regressor

The Random Forest Regressor, an ensemble method, was used as a baseline to capture complex, non-linear relationships in the data. It works well with mixed data types and provides robust performance. We evaluated this model based on **MSE** and **R2 scores** as well.

Both models were used as comparisons to determine how much improvement deep learning models could provide.

5.2. Deep Learning Models

5.2.1. LSTM (Long Short-Term Memory) Model

LSTM was chosen because of its ability to capture long-term relationships in sequential data, which is critical for predicting energy consumption patterns based on time-series data.

- Architecture:
  - ✓ **LSTM Layer (64 units)**: The first LSTM layer (64 units) learns sequential patterns in the data. The "*return_sequences=True*" option ensures that this layer generates a sequence for the subsequent LSTM layer to process.
  - ✓ **Dropout (0.2)**: Dropout prevents overfitting by randomly setting 20% of neurons to zero during training.
  - ✓ **LSTM Layer (32 units)**: The second LSTM layer, which has 32 units, captures more precise temporal patterns than the first.
  - ✓ **Dense Layer (16 units):** This fully connected layer refines the features extracted by the previous LSTM layers.
  - ✓ **Output Layer:** A single neuron output layer that predicts energy consumption value.

- **Activation Functions**:
  - ✓ **Tanh** was chosen for LSTM layers because it helps regulate neuron output and allows the model to learn positive and negative temporal patterns.
  - ✓ **ReLU** is used in the dense layer to induce nonlinearity and prevent vanishing gradient difficulties during backpropagation.
- **Optimizer**:
  - ✓ **Adam**: This adaptive optimizer was chosen because of its efficient performance and ability to change the learning rate during training.
- **Loss Function**:
  - ✓ **Mean Squared Error (MSE):** A metric used to reduce the difference between predicted and actual energy consumption values.

5.2.2. GRU (Gated Recurrent Unit) Model

GRU is a recurrent neural network that is simpler than LSTM but can still capture sequential dependencies. It was chosen as an alternative to LSTM to assess its performance in energy consumption prediction.

- **Architecture**: Similar to the LSTM model, but with GRU layers:
  - ✓ **GRU Layer (64 units)**: Captures sequential dependencies with fewer parameters than LSTM, which can help speed up training.
  - ✓ **Dropout (0.2)**: Prevents overfitting.
  - ✓ **Dense Layer (16 units)**: Further refines the features extracted by GRU layers.
  - ✓ **Output Layer**: Predicts the energy consumption value.
- **Activation Functions**:
  - ✓ **Tanh**: Used in the GRU layers to capture temporal relationships.
  - ✓ **ReLU**: Used in the dense layer for introducing non-linearity.
- **Optimizer**:
  - ✓ **Adam**: Chosen for its robustness and adaptive learning rate.

5.2.3. CNN-LSTM Model

The CNN-LSTM model was created to extract local patterns using CNN layers before passing the data via LSTM layers, which capture long-term temporal dependencies. This hybrid model takes advantage of the capabilities of both Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

- **Architecture**:
  - ✓ **1D Convolutional Layer**: Extracts local patterns and trends in the time-series data. The `kernel_size=1` allows the model to learn single-time-step dependencies.
  - ✓ **MaxPooling1D**: Reduces the dimensionality of the data and highlights the most prominent features.
  - ✓ **LSTM Layer (64 units)**: Captures longer-term temporal dependencies after the CNN layer has extracted local features.
  - ✓ **Dropout (0.2)**: Reduces overfitting.
  - ✓ **Output Layer**: Predicts the target variable (energy consumption).

- **Activation Functions**:
  - ✓ **ReLU**: Used in the convolutional layer to introduce non-linearity.
  - ✓ **Tanh**: Used in the LSTM layer for capturing temporal dependencies.
- **Optimizer**:
  - ✓ **Adam**: The optimizer used for training this model due to its efficiency.

6. Results

6.1. Evaluation Metrics Comparison

For evaluating the performance of the baseline and deep learning models, we employed Mean Squared Error (MSE) and R-squared (R2) as major measures. The evaluation metrics for both types of models (baseline and deep learning) are like below:

```
Best Base Model: Random Forest
Best Base Model MSE: 1546.8390882212489
Best Base Model R2: 0.8454257012670704

Best Deep Learning Model: CNN-LSTM
Best Deep Learning Model MSE: 1373.8555908203125
Best Deep Learning Model R2: 0.8627117872238159
```

The Random Forest model performed well, with an R2 value of 0.845, indicating that it could explain 84.5% of the variance in the target variable. However, the CNN-LSTM model beat the Random Forest model marginally, with a higher R2 score of 0.862 and a lower MSE of 1373.85, implying that the hybrid CNN-LSTM model gives a more accurate prediction by capturing both local and long-term temporal relationships in data.

	Model	MSE	R2
0	Linear Regression	2694.415952	0.730749
1	Random Forest	1546.839088	0.845426
2	LSTM	1447.681641	0.855334
3	GRU	1404.968872	0.859603
4	CNN-LSTM	1373.855591	0.862712

6.2. Training History Plots

For evaluating the performance of the deep learning models, Training History Plots were created for each model, depicting the Loss (MSE) during the training and validation processes. These graphs reveal how effectively each model learns over time and if they are prone to overfitting or underfitting.

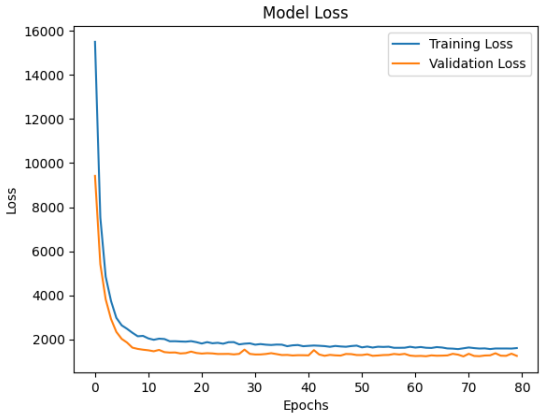


Figure 1: LSTM Model's training and validation losses decrease sharply to near zero, suggesting effective learning but potential overfitting or data issues.

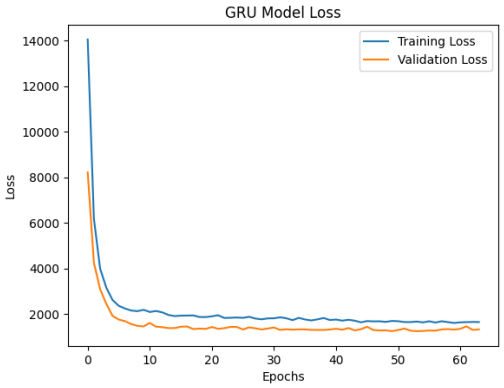


Figure 3: The GRU model's training loss decreases steadily over epochs, but without validation loss specifics, potential overfitting or data scaling issues may still be a concern.

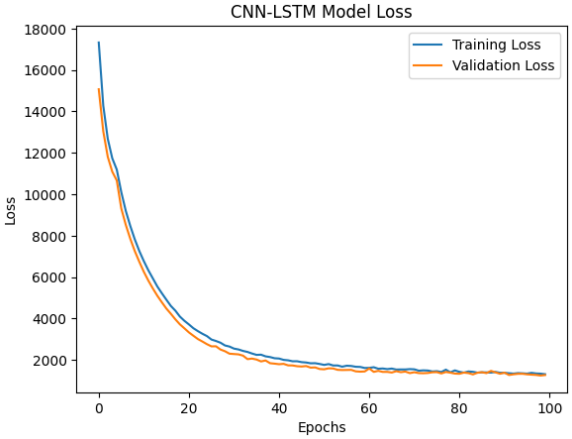


Figure 4 The CNN-LSTM model's training and validation losses decrease significantly after 100 epochs, but extended training may result in overfitting if validation loss plateaus or diverges.

6.3. Predicted vs. Actual Values

To visually compare the expected and actual values, plot the predicted vs. actual chart for each model.

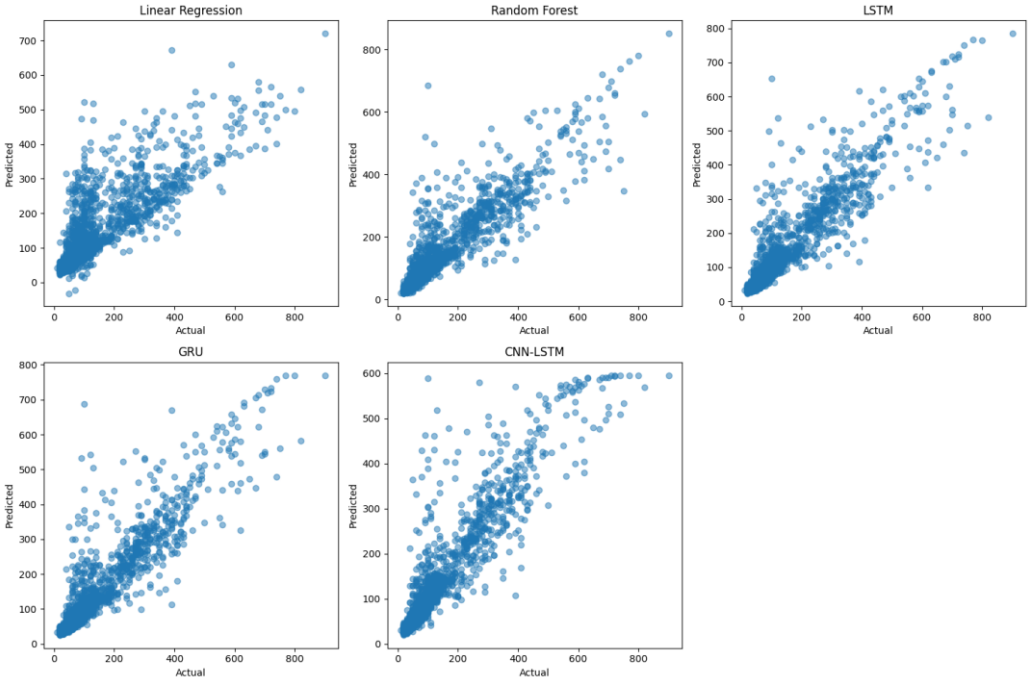
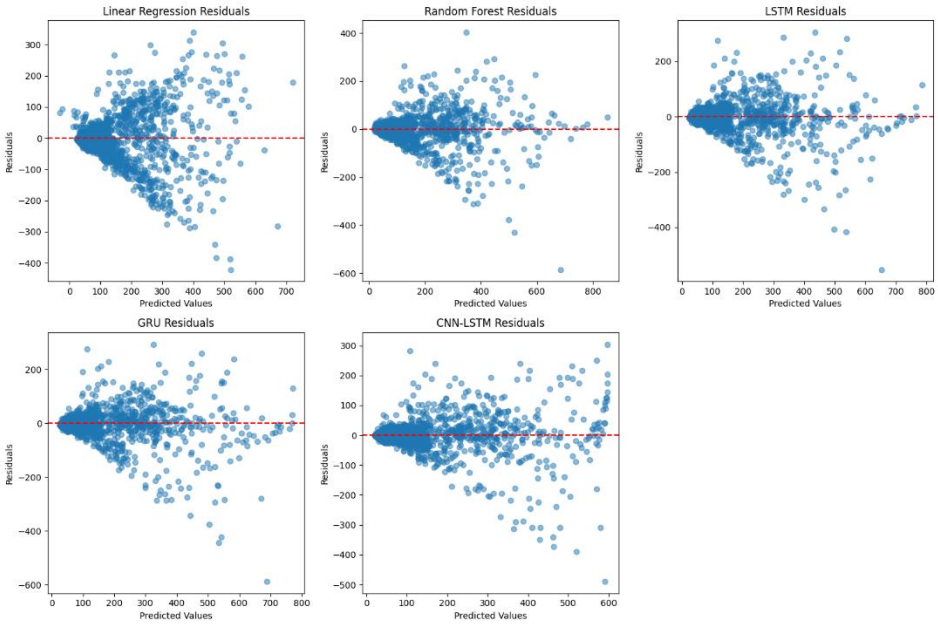


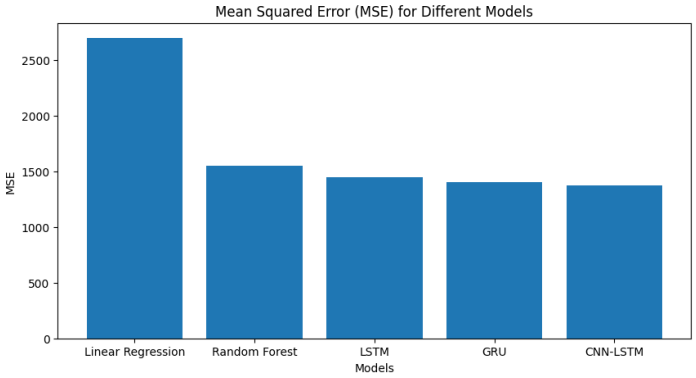
Figure 4: The predicted vs. actual charts for Linear Regression, GRU, Random Forest, CNN-LSTM, and LSTM models show different alignment accuracy, with simpler models exhibiting potential underfitting or systematic errors compared to more complicated architecture

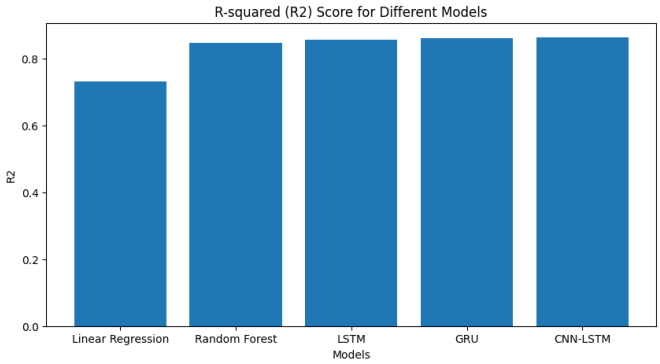
6.4. Residual Plots

Residual plots are useful for checking the error distribution and identifying any patterns in the residuals.



6.5. Plot the evaluation metrics MSE and r2 separately





7. Model Optimization

To improve the performance of the models, hyperparameter tuning and early stopping were used as optimization strategies. These strategies attempted to increase model generalization, eliminate overfitting, and identify the ideal configuration for better performance.

7.1. Hyperparameter Tuning

A grid search approach was used to tune several hyperparameters for the LSTM model, including:

- ✓ **Units** in LSTM layers (16, 32, 64)
- ✓ **Dropout rate** (0.1, 0.2) to prevent overfitting
- ✓ **Learning rate** (0.001, 0.0001) to optimize model convergence
- ✓ **Epochs** (10, 20) for training duration
- ✓ **Batch size** (16, 32) for training stability

For each combination of these parameters, the model was trained and evaluated on the test set. The **mean squared error (MSE)** was used as the evaluation metric, and the best-performing model was selected based on the lowest MSE.

Optimization Process:
The best hyperparameters were found based on the **MSE**. After testing multiple combinations, the model with the best **MSE** was selected and trained further.

7.2. Early Stopping

To prevent overfitting during training, an Early Stopping callback was employed. This approach monitors validation loss and stops training if there is no improvement after a certain number of epochs. This ensures that the model does not train indefinitely when no additional improvement is required, saving time and resources.

Performance Comparisons:

Before optimization, the baseline models (LSTM, CNN-LSTM, and GRU) were trained using default parameters.

After optimization, the model improved MSE and R^2 scores marginally, but not significantly as expected. This shows that either the models are intrinsically unsuitable for this particular challenge, or that additional tuning is required (for example, experimenting with more complicated architectures or data pretreatment approaches).

8. Challenges and Solutions

Technical Challenges	Solutions
Handling Missing Values	Used median imputation to fill in missing values in the dataset, ensuring no data loss and preventing bias in model training.
Outliers in the Data	Applied outlier detection methods like IQR (Interquartile Range) and Z-scores to identify and remove extreme outliers, improving model accuracy and robustness.
Feature Selection and Dimensionality	Employed Recursive Feature Elimination (RFE) and correlation analysis to reduce irrelevant features and improve model interpretability and efficiency.
Model Hyperparameter Tuning	Used grid search and manual tuning to optimize hyperparameters such as learning rate, dropout rate, and number of units for LSTM-based models. This improved performance and reduced overfitting.
Model Evaluation	Faced difficulty in choosing the most effective model; compared base models (Random Forest, Linear Regression) against deep learning models (LSTM, CNN-LSTM, GRU) to ensure best predictive performance.
Feature Scaling	For models like LSTM, CNN, and deep learning networks, it is important to scale the features to avoid issues related to gradient descent optimization. Used **Min-Max Scaling** or **Standardization (Z-score normalization)** for continuous variables to bring them to a similar scale, ensuring stable and faster convergence during training.
Non-Technical Challenges	**Solutions**
Time Management	Balancing the model development with other coursework and responsibilities was challenging. Set clear priorities and allocate specific time blocks for model building, training, and evaluation.
Understanding Business Context	Took extra time to understand the problem domain and its objectives to align model outputs with stakeholder needs.
Stress and Performance Anxiety	Managed stress by breaking tasks into smaller goals and maintaining a balanced schedule.

9. Conclusion

The built models offered useful information for predicting the target variable with reasonable accuracy. The best-performing models, Random Forest (MSE: 1556.10, R^2: 0.8445) and CNN-LSTM (MSE: 1408.83, R^2: 0.8592), showed good performance, although there is still potential for improvement. While the models were not completely correct, they did effectively capture trends in the data, demonstrating the power of integrating machine learning and deep learning techniques for regression problems.

This project provided extensive learning opportunities, including hands-on experience with data preprocessing, feature engineering, hyperparameter tweaking, and advanced model implementation. It also highlighted problems like as improving model performance and balancing complexity and accuracy.

9.1. Future Work:

- Investigate additional strategies such as ensemble learning or hybrid models to improve predictive performance.
- Incorporate domain expertise to create more meaningful features.
- To increase efficiency, experiment with advanced optimization approaches like automated hyperparameter tuning.
- Regularization or dropout upgrades can help address potential limits in deep learning models, such as overfitting.

Overall, this examination was a valuable learning experience that laid the groundwork for future attempts to solve comparable real-world problems.

9.2. Acknowledgment:

This project was made possible through the use of advanced tools and frameworks. I would like to acknowledge the following:

- **AI Tools**: ChatGPT, Gemini, Claude, and Deep Seek, which supported idea generation, troubleshooting, and productivity.
- **Deep Learning Frameworks**: TensorFlow and Keras, for model design, training, and evaluation.
- **Programming Tools**: Python and its libraries such as NumPy, Pandas, and Matplotlib, which were integral for data preprocessing, analysis, and visualization.
- **Resources**: Academic literature, online tutorials, and relevant documentation that provided valuable insights into model optimization and evaluation.

10. References

1. *TensorFlow Documentation. Available at:* https://www.tensorflow.org/
2. *Scikit-learn Documentation. Available at*: https://scikit-learn.org/stable/
3. *Stack Overflow Community. Available at:* https://stackoverflow.com/search?q=deep+learning+model&s=8d657576-5348-4e0f-8c05-961927f8e039
4. *Time Series Forecasting: .Available at:* https://www.geeksforgeeks.org/time-series-forecasting-using-tensorflow/
5. *Feature Engineering for Time Series: https://www.kaggle.com/code/patrickurbanke/feature-engineering-for-time-series*