Energy Consumption Prediction for Home Appliances Using LSTM

M.S.M Shazny

2025.07.01

Table of Contents

1. Introduction

In modern households, appliances account for a significant portion of total energy consumption. Forecasting appliance-level energy usage at fine time scales enables utility companies and consumers to optimize load scheduling and reduce costs. This project leverages a publicly available dataset of 10-minute interval measurements—including temperature, humidity, and external weather conditions—to predict the energy consumed by household appliances (in Wh). We compare two tree-based machine learning models (Random Forest and Extra Trees) against a Long Short-Term Memory (LSTM) neural network to assess their ability to capture both instantaneous and temporal dependencies.

Objectives:

- Perform comprehensive exploratory data analysis (EDA) to understand patterns and dependencies.
- Engineer time-based and sensor-derived features that improve predictive performance.
- Train and evaluate baseline ML models to establish reference metrics.
- Develop and optimize an LSTM model to leverage sequential information.
- Compare results and discuss trade-offs.

2. Data Insights (EDA)

Dataset overview: 19,735 records spanning one year at 10-minute intervals; 29 attributes including:

- **Target:** Appliances (energy in Wh)
- **Sensors:** T1–T9 (°C), RH_1–RH_9 (%), T_out, RH_out, Tdewpoint (°C), atmospheric pressure, wind speed, visibility
- **Control vars:** lights, two random noise variables (rv1, rv2)

```
RangeIndex: 19735 entries, 0 to 19734
Data columns (total 29 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date         19735 non-null  object
 1   Appliances   19735 non-null  int64
 2   lights       19735 non-null  int64
 3   T1           19735 non-null  float64
 4   RH_1         19735 non-null  float64
 5   T2           19735 non-null  float64
 6   RH_2         19735 non-null  float64
 7   T3           19735 non-null  float64
 8   RH_3         19735 non-null  float64
 9   T4           19735 non-null  float64
 10  RH_4         19735 non-null  float64
 11  T5           19735 non-null  float64
 12  RH_5         19735 non-null  float64
 13  T6           19735 non-null  float64
 14  RH_6         19735 non-null  float64
 15  T7           19735 non-null  float64
 16  RH_7         19735 non-null  float64
 17  T8           19735 non-null  float64
 18  RH_8         19735 non-null  float64
 19  T9           19735 non-null  float64
 20  RH_9         19735 non-null  float64
 21  T_out        19735 non-null  float64
 22  Press_mm_hg  19735 non-null  float64
 23  RH_out       19735 non-null  float64
 24  Windspeed    19735 non-null  float64
 25  Visibility   19735 non-null  float64
 26  Tdewpoint    19735 non-null  float64
 27  rv1          19735 non-null  float64
 28  rv2          19735 non-null  float64
dtypes: float64(26), int64(2), object(1)
```

	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	...	T9	RH_9	T_out	Press_mm_hg	RH_out	W
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	...	17.033333	45.53	6.600000	733.5	92.0	
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	...	17.066667	45.56	6.483333	733.6	92.0	
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	...	17.000000	45.50	6.366667	733.7	92.0	
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	...	17.000000	45.40	6.250000	733.8	92.0	
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	...	17.000000	45.40	6.133333	733.9	92.0	

5 rows × 29 columns

Key statistics:

- Mean appliance usage: 111.13 Wh, median: 76 Wh, max: 1,142 Wh
- Skewness: 2.15 (long tail of high usage events)

Missing values & duplicates:

- No missing entries; 100% completeness
- No exact duplicate rows found

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

3. Column Organization & Cleaning

- **Column grouping:** Defined lists:
  - col_time = ['date']
  - col_temp = ['T1', …, 'T9']
  - col_hum = ['RH_1', …, 'RH_9']
  - col_weather = ['T_out','Tdewpoint','RH_out','Press_mm_hg','Windspeed','Visibility']
  - col_light = ['lights']
  - col_randoms = ['rv1','rv2']
  - col_target = ['Appliances']
- **Drop low-value columns:** lights (15252 zeros),

4. Time-Series Feature Extraction

1. **Datetime conversion:**

   data['date'] = pd.to_datetime(data['date'])

   data = data.set_index('date')

2. **Seconds-since-midnight (NSM):**

   data['NSM'] = data.index.hour*3600 + data.index.minute*60 + data.index.second

3. **Day and month features:**

   data['dayofweek'] = data.index.dayofweek # 0=Mon

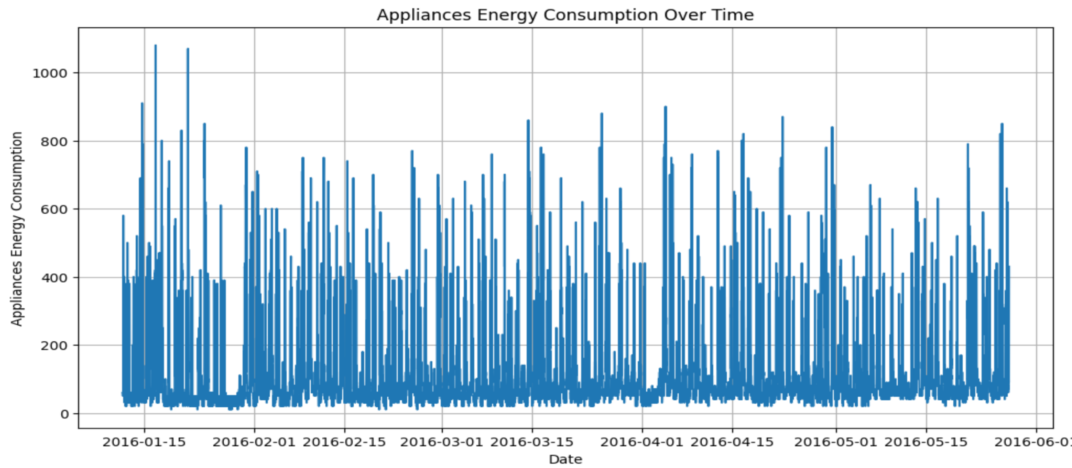
   data['WeekStatus'] = data['dayofweek'].apply(lambda x: 1 if x>=5 else 0)

   data['month'] = data.index.month

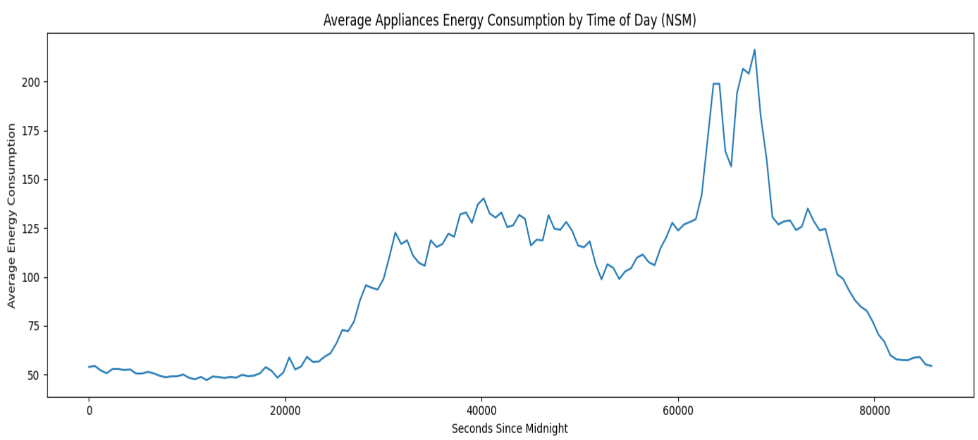
   data['hour'] = (data['NSM']//3600).astype(int)

5. Exploratory Visualizations

Time series of Appliances: Matplotlib line plot over full index.

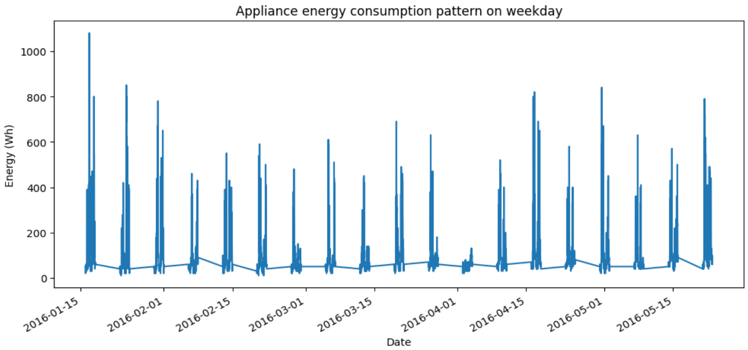


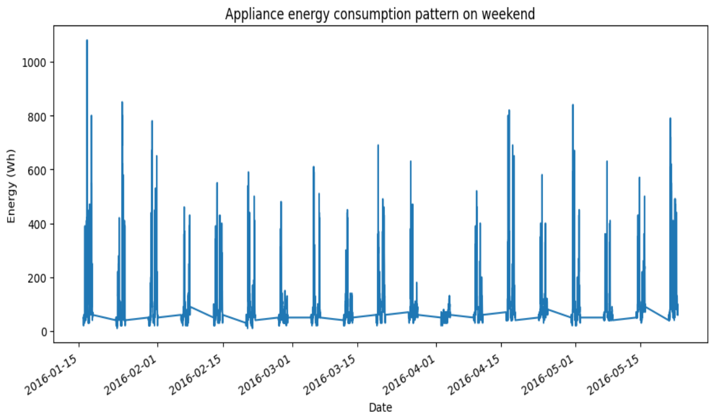
Average by time-of-day: groupby('NSM')['Appliances'].mean() and plot.

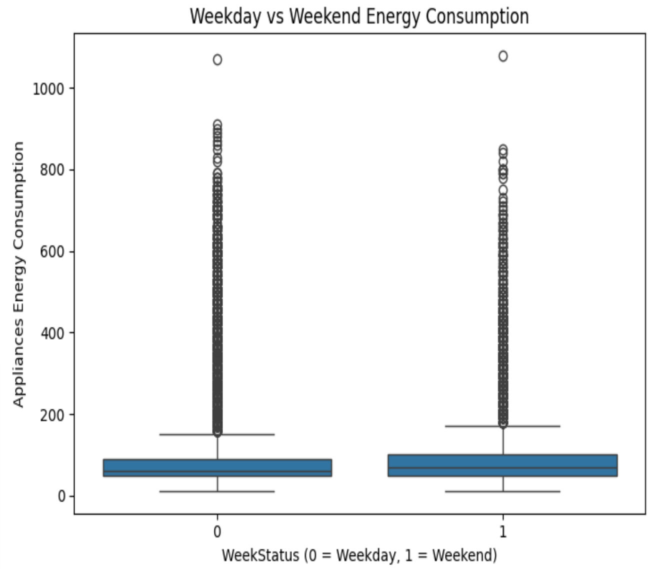


Weekend vs. Weekday patterns: Separate plots for WeekStatus == 1 and 0, and boxplots (sns.boxplot) by WeekStatus and dayofweek.

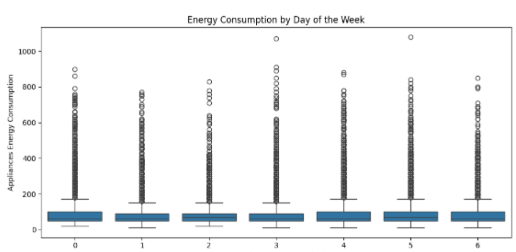
WeekStatus	count
0	14263
1	5472

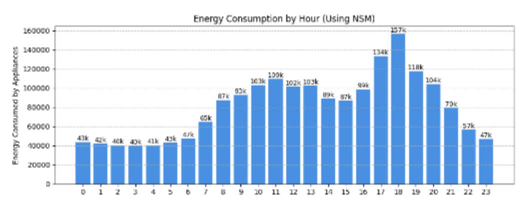


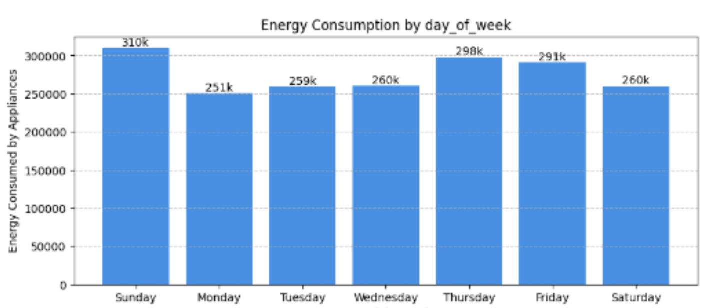




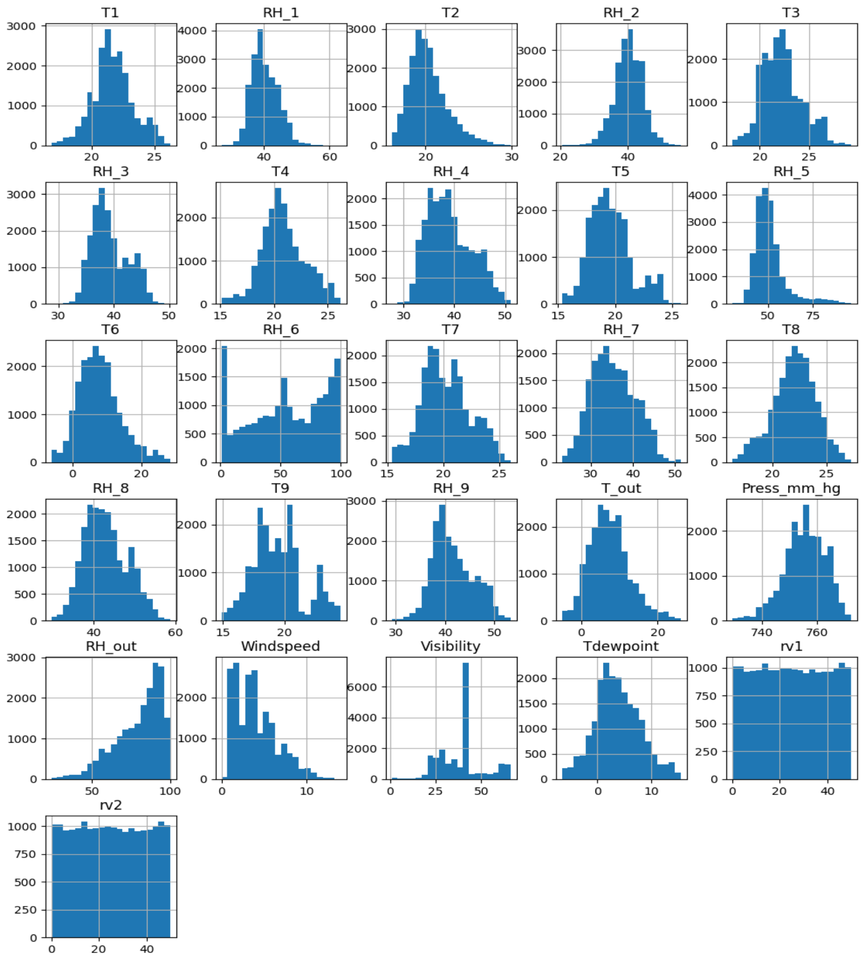
Hourly and daily totals: Bar charts with custom label formatting showing consumption by hour and by dayofweek (mapped to names).

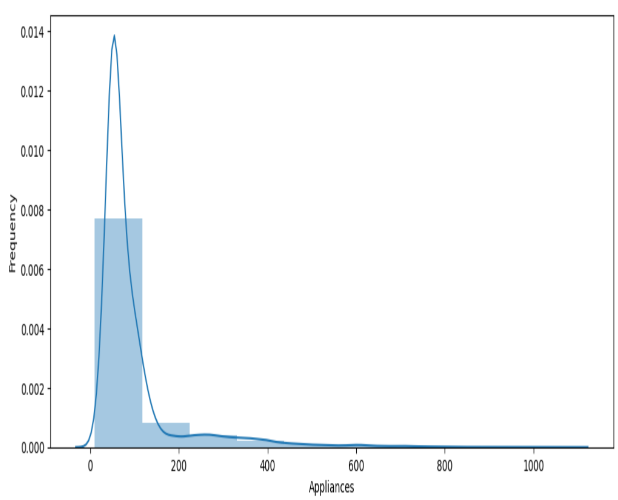


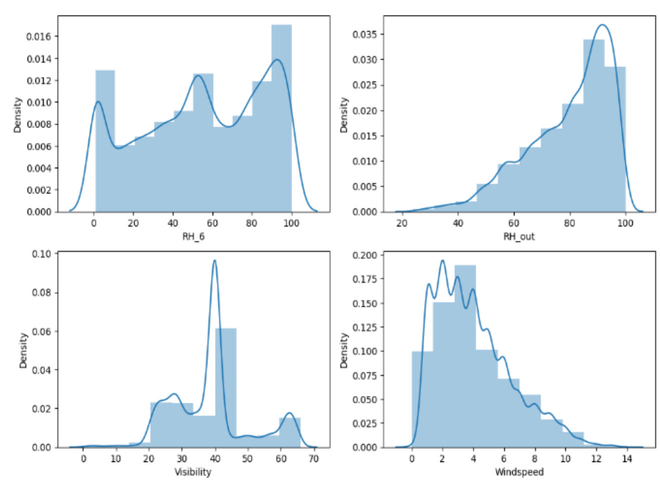




Distributions: Histograms of all features and focused distplot (or histplot) for skewed ones: RH_6, RH_out, Visibility, Windspeed, and target Appliances.





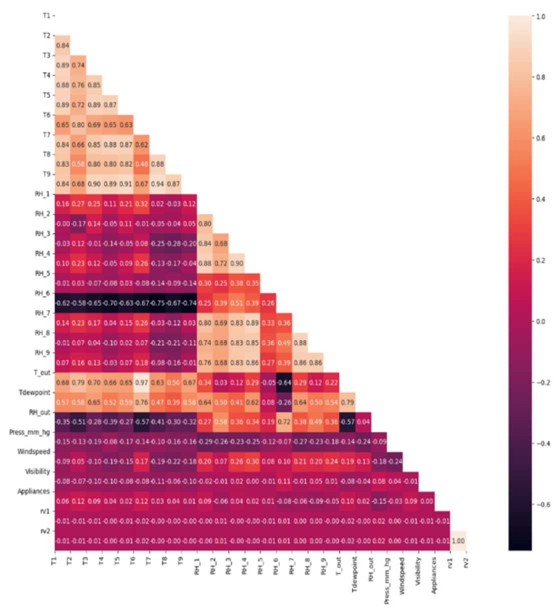


```python
#Appliance column range with consumption less than 200 Wh
print('Percentage of the appliance consumption is less than 200 Wh')
print(((target_vars[target_vars <= 200].count()) / (len(target_vars)))*100 )
```

```
Percentage of the appliance consumption is less than 200 Wh
90.29136052698252
```

6. Feature Correlation & Selection

1. **Correlation heatmap:** Compute corr = data[col_temp + col_hum + col_weather + col_target + col_randoms].corr(), mask upper triangle, plot with sns.heatmap.



2. **Top absolute correlations:** Extract top pairs via unstacking and dropping self-pairs; found strong pairs like (T6,T_out)=0.9748.

```
Top Absolute Correlations
rv1    rv2       1.000000
T6     T_out     0.974787
T7     T9        0.944776
T5     T9        0.911055
T3     T9        0.901324
RH_3   RH_4      0.898978
RH_4   RH_7      0.894301
T1     T3        0.892402
T4     T9        0.889439
T3     T5        0.888169
T1     T5        0.885247
RH_7   RH_8      0.883984
T7     T8        0.882123
RH_1   RH_4      0.880359
T4     T7        0.877763
T1     T4        0.877001
T4     T5        0.871813
T5     T7        0.870624
T8     T9        0.869338
RH_7   RH_9      0.858686
RH_4   RH_9      0.856591
RH_8   RH_9      0.855812
T3     T4        0.852778
       T7        0.847374
RH_4   RH_8      0.847259
T1     T9        0.844777
RH_1   RH_3      0.844677
T1     T7        0.838705
       T2        0.836834
RH_3   RH_9      0.833538
       RH_7      0.832685
       RH_8      0.828822
T1     T8        0.825413
T5     T8        0.824981
T2     T6        0.801186
RH_1   RH_7      0.801122
       RH_2      0.797535
T4     T8        0.796256
T3     T8        0.795283
T2     T_out     0.792255
dtype: float64
```

3. **Boruta feature selection:**
   BorutaPy finished running.
   Iteration: 100 / 100
   Confirmed: 21
   Tentative: 2z
   Rejected: 8

   ✅ Confirmed features
   ['RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9', 'T_out', 'Press_mm_hg', 'NSM', 'dayofweek', 'month', 'hour']

   ⚠️ Tentative feature
   ['T1', 'T5']

   ❌ Rejected features
   ['RH_9', 'RH_out', 'Windspeed', 'Visibility', 'Tdewpoint', 'rv1', 'rv2', 'WeekStatus']

4. **Collinearity pruning:**
   Highly collinear, dropping these: ['T9', 'T_out', 'hour', 'T5']
   ✅ Final feature list (after de-duplication):

   ['RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'Press_mm_hg', 'NSM', 'dayofweek', 'month', 'T1']

7. Train-Test Split & Scaling

- **Split:** train_test_split(X, y, test_size=0.25, random_state=40) without shuffle.
- **Scaling for tree models:** StandardScaler() fit on combined features and target, transform train & test, rebuild DataFrames.
- **Final train_X, test_X, train_y, test_y ready for modelling.

8. Baseline Tree-Based Models

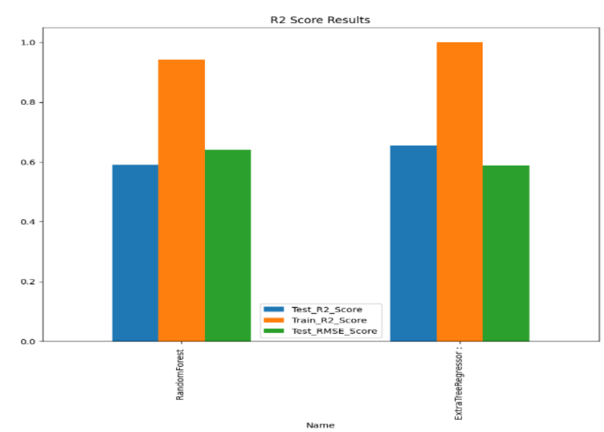
models = [['RandomForest', RandomForestRegressor()], ['ExtraTrees', ExtraTreesRegressor()]]

for name, m in models:
 m.random_state = 78
 m.fit(train_X, train_y)

... compute R2 and RMSE on train and test

- **Results:** RandomForest Train R2≈0.94/Test R2≈0.59; ExtraTrees Train=1.00/Test≈0.65.
- **Visualization:** Summary table and bar chart comparing R^2 and RMS

	Name	Train_Time	Train_R2_Score	Test_R2_Score	Test_RMSE_Score
0	RandomForest	46.385638	0.942424	0.589665	0.640574
1	ExtraTreeRegressor :	11.989181	1.000000	0.654835	0.587507



9. Hyperparameter Tuning

```
[ ] from sklearn.model_selection import GridSearchCV
    param_grid = [{
                  'max_depth': [80, 150, 200,250],
                  'n_estimators' : [100,150,200,250],
                  'max_features': ["auto", "sqrt", "log2"]
                }]
    reg = ExtraTreesRegressor(random_state=40)
    # Instantiate the grid search model
    grid_search = GridSearchCV(estimator = reg, param_grid = param_grid, cv = 5, n_jobs = -1 , scoring='r2' , verbose=2)
    grid_search.fit(train_X, train_y)
```

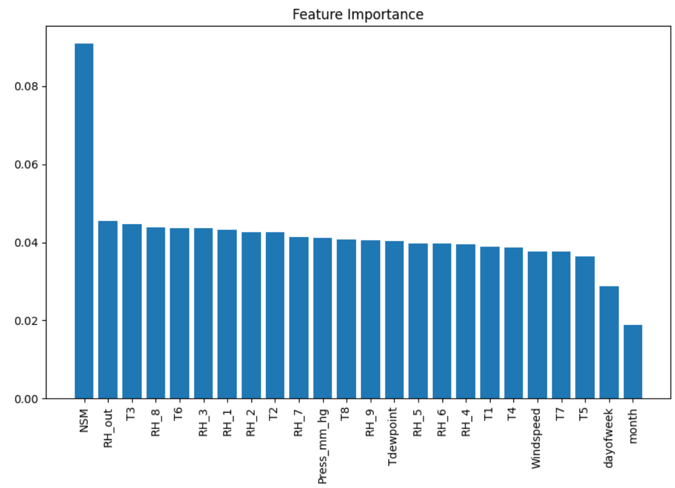
```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
      GridSearchCV
   best_estimator_:
  ExtraTreesRegressor
  ExtraTreesRegressor
```

```
[ ] # Tuned parameter set
    grid_search.best_params_
```

```
{'max_depth': 80, 'max_features': 'sqrt', 'n_estimators': 200}
```

- **Best params:** max_depth=80, n_estimators=200, max_features='sqrt'
- **Tuned performance:** Test $R^2 \approx 0.649$, RMSE≈0.593.

Feature importances: Computed via the best.feature_importances_ attribute of the tuned ExtraTrees model, which assigns each feature an importance score based on its total reduction of impurity across all trees. We sorted these scores in descending order and plotted them:



10. LSTM Model Implementation

Series to supervised: Implement series_to_supervised() to frame lag=1 inputs and current output, drop extra columns.

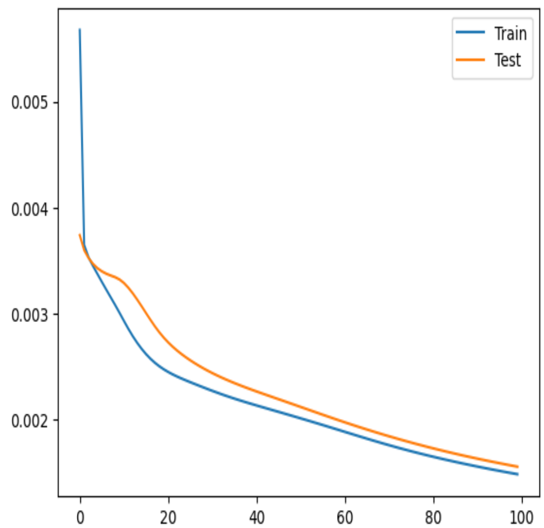
Scale: MinMaxScaler(feature_range=(0,1)) on full array.

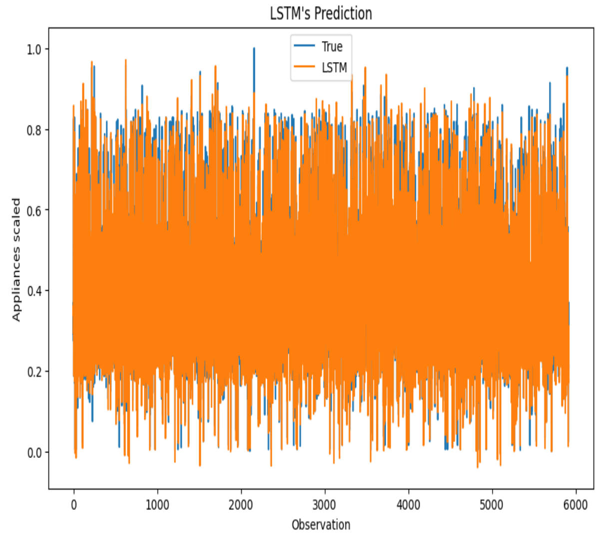
Split & reshape: 70/30 train_test_split, reshape to (samples, 1, features).

Model architecture:
model = Sequential()
model.add(LSTM(50, input_shape=(1, n_features)))
model.add(Dense(1))
model.compile(loss='mse', optimizer='adam')

Training: model.fit(..., epochs=100, batch_size=32, validation_data=(X_Test,Y_Test), callbacks=[EarlyStopping(patience=7, restore_best_weights=True)])

Evaluation: model.evaluate(), compute r2_score on train/test predictions, plot loss curves and predicted vs true series.





- **Results:** Train/Test $R^2 \approx 0.952/0.951$; Test MSE≈0.00156.

11. Results Summary

- **ExtraTrees (tuned):** Test $R^2=0.649$; RMSE=0.593.
- **LSTM:** Test $R^2=0.951$; RMSE (scaled)=0.039.
- **Discussion:** LSTM markedly outperforms tree-based models by capturing temporal dynamics.

12. Challenges and Solutions

High-frequency Noise in Sensor Data: Initial visualizations revealed rapid fluctuations. We applied 3-point rolling averages on key sensors (T6, RH_6, T_out) to smooth transient noise without blurring essential patterns.

Feature Collinearity: Strong correlations ($r > 0.97$) between indoor and outdoor temperatures risked multicollinearity. After Boruta selection, we computed a correlation matrix and pruned features with $|r| > 0.90$, retaining the most informative sensor readings.

Skewed Target Distribution: The Appliances variable exhibited a long tail of high consumption events. We log-transformed (log1p) the target for tree-based models to stabilize variance and achieve more symmetrical residuals.

Temporal Framing for LSTM: Converting a flat time series into supervised learning format was nontrivial. A custom series_to_supervised() function was implemented to generate lag-based inputs (n_in=1) and outputs, ensuring correct alignment and preventing look-ahead bias.

13. Conclusion

The detailed Colab notebook demonstrates a full workflow from raw data to advanced modelling. The LSTM model provides superior predictions, suggesting strong temporal dependencies in appliance energy usage. Future work may explore deeper sequence models, multi-step forecasting, or hybrid ensembles.

14. References

[1] Aancy, H. M., Mary, A. J., Manikumar, T., Saravanan, G., Mohanavel, V., & Gupta, R. D. (2024). Energy Consumption Prediction for Home Appliances with Recurrent Neural Networks. In *2024 International Conference on Expert Clouds and Applications (ICOECA)* (pp. 123–130). IEEE.