

## Table of Contents

|  |    |
|--|----|
| Table of Contents.....   | 1  |
| Abstract.....  | 3  |
| Introduction.....  | 4  |
| 1.1 Background and Motivation.....                             | 4  |
| 1.2 Problem Statement.....                                     | 5  |
| 1.3 Scope of the Project.....                                  | 5  |
| 1.4 Research Questions.....                                    | 6  |
| 1.5 Objectives.....  | 7  |
| 1.6 Hypothesis Justification:.....                             | 7  |
| Literature Review.....   | 8  |
| 2.1 Review of Time Series Energy Forecasting Methods.....      | 9  |
| 2.2 Gradient Boosting Models (GBMs) in Energy Forecasting..... | 9  |
| 2.3 Deep Learning Methods (LSTM).....                          | 10 |
| 2.4 Anomaly Detection in Building Energy Consumption.....      | 11 |
| 2.5 Interactive Dashboards for Energy Data Visualization.....  | 12 |
| 2.6 Summary and Research Gap.....                              | 12 |
| Methodology and Design.....                                    | 14 |
| 3.1 Overview of the Approach.....                              | 14 |
| 3.2 Dataset Description (BDG2).....                            | 15 |

|   |    |
|---|----|
| 3.3 Feature Engineering.....                      | 16 |
| 3.3.1 Feature Importance Analysis.....            | 20 |
| 3.4 Aggregation and Handling Missing Data.....    | 21 |
| 3.5 Models Comparison.....                        | 22 |
| 3.6 Evaluation Metrics.....                       | 24 |
| 3.6.1 Root Mean Squared Error (RMSE).....         | 24 |
| 3.6.2 Mean Absolute Error (MAE).....              | 25 |
| 3.6.3 Anomaly Detection Metrics.....              | 25 |
| 3.7 Tools and Libraries to be Used.....           | 26 |
| 3.8. Dashboard Development (Plotly).....          | 27 |
| Implementation.....                               | 28 |
| 4.1 Model Architecture for LSTM.....              | 28 |
| 4.2 Hyperparameter Tuning and Training Setup..... | 30 |
| 4.3 Isolation Forest and Autoencoder Setup.....   | 31 |
| 4.4 Forecasting Pipeline Design.....              | 34 |
| 4.4.1 Raw Data Ingestion.....                     | 34 |
| 4.4.2 Data Merging.....                           | 34 |
| 4.4.3 Train/Test Split.....                       | 35 |
| 4.4.4 Model Training.....                         | 35 |
| 4.4.5 Dashboard Development with Plotly Dash..... | 37 |
| Testing and Results.....                          | 40 |

|   |    |
|---|----|
| 5.1 Model Performance Summary.....                    | 40 |
| 5.2 Comparison with GBMs and Random Forest.....       | 41 |
| 5.3 Anomaly Detection Results and Interpretation..... | 42 |
| 5.4 Forecast vs Actual Visualizations.....            | 44 |
| 5.5 Weather Impact Overlays.....                      | 46 |
| 5.6 Visual Insights and Dashboard Integration.....    | 48 |
| Discussion.....                                       | 51 |
| 6.1 Interpretation of Forecasting Accuracy.....       | 51 |
| 6.2 Impact of Weather Variables.....                  | 51 |
| 6.3 Insights from Anomalies.....                      | 52 |
| 6.4 Usefulness of Dashboard for Energy Managers.....  | 54 |
| 6.5 Challenges Encountered.....                       | 54 |
| 6.6 Ethical Considerations.....                       | 55 |
| 6.7 Result Comparison.....                            | 56 |
| Conclusion and Recommendations.....                   | 57 |
| 7.1 Summary of Key Findings.....                      | 57 |
| 7.2 Recommendations for Future Work.....              | 57 |
| 7.3 Limitations and Areas for Improvement.....        | 58 |
| References.....                                       | 59 |
| Appendices.....                                       | 5  |



## **Abstract**

This study presents a robust data-driven framework for forecasting energy consumption across multiple building sites using deep learning, specifically Long Short-Term Memory (LSTM) networks. Leveraging the publicly available BDG2 dataset, which integrates hourly electricity consumption, meteorological conditions, and building metadata. We formulate the problem as a multivariate time series regression task. The methodology involves extensive preprocessing, including data cleaning, feature engineering (such as lag features and rolling statistics), and merging of building-level consumption with site-specific weather data. LSTM is selected as the final forecasting model due to its superior performance in capturing temporal dependencies, outperforming traditional ensemble models such as Random Forest, XGBoost, and LightGBM in RMSE and MAE across all top five buildings.

To enhance the reliability of the forecasting system, anomaly detection is incorporated using both Isolation Forest and Autoencoder methods. Results show that Isolation Forest detects anomalies with an average rate of 15%, while Autoencoders yield about 20%, indicating the model's sensitivity to fluctuations in energy patterns. The implementation concludes with an interactive dashboard created using Plotly, which enables visualization of actual vs predicted usage, overlays and key weather variables. This dashboard was explored directly within the Colab environment, supporting seamless data inspection and validation. The integrated framework demonstrates the viability of combining deep learning with anomaly detection and visualization to address challenges in multi-site energy forecasting.

## Introduction

### 1.1 Background and Motivation

Buildings account for a significant portion of global energy consumption, contributing nearly 30% of final energy use and over 25% of global CO<sub>2</sub> emissions ([IEA, 2022](#)). As energy demands rise and sustainability goals intensify, the ability to accurately forecast building energy consumption becomes critical for enabling smarter grid management, energy optimization, and emissions reduction.

Recent advancements in data availability and machine learning techniques have facilitated novel approaches to energy forecasting. High-resolution datasets like the Building Data Genome Project 2 (BDG2) provide a valuable foundation for developing scalable forecasting models across diverse building types ([Miller et al., 2020](#)). These datasets capture building-level energy usage alongside meteorological and contextual data, enabling the application of sophisticated predictive algorithms.

Machine learning methods, particularly Gradient Boosting Machines (GBMs) and Recurrent Neural Networks (RNNs) have demonstrated considerable promise in energy modeling. GBMs such as XGBoost, LightGBM, and Random Forests are known for their predictive accuracy and robustness in handling structured data ([Chen & Guestrin, 2016](#); [Ke et al., 2020](#)). Meanwhile, Long Short-Term Memory (LSTM) networks, a subclass of RNNs, are increasingly favored for time series forecasting due to their capacity to model temporal dependencies and long-term patterns ([Wang et al., 2020](#); [Alam et al., 2021](#)).

## **1.2 Problem Statement**

Despite the proliferation of machine learning tools, energy consumption forecasting remains a complex challenge due to the heterogeneity of buildings, non-linear usage behavior, and external influences such as weather, occupancy, and calendar effects. Ensemble tree models often underperform when confronted with dynamic temporal dependencies, while deep learning models require large datasets and careful tuning to avoid overfitting.

Moreover, energy datasets often contain anomalous readings caused by sensor faults, outages, or unusual operational conditions. These anomalies, if unaddressed, can distort learning and reduce model accuracy. There is a need for a comprehensive solution that not only predicts consumption accurately but also identifies and visualizes anomalies.

## **1.3 Scope of the Project**

This project is focused on developing a forecasting system for daily energy consumption across multiple buildings using the BDG2 dataset and comparing the performance of traditional machine learning models (Random Forest, LightGBM, XGBoost) with LSTM networks. The project will also be implementing anomaly detection methods using Isolation Forests and Autoencoders to identify irregular energy patterns. The project will also be building an interactive dashboard using Plotly to visualize actual vs predicted usage, weather overlays, and detected anomalies.

The study is limited to electricity usage and weather data available within the BDG2 dataset and covers five of the most active buildings within the dataset for a detailed evaluation.

## 1.4 Research Questions

This study aims to investigate the effectiveness of deep learning approaches, particularly Long Short-Term Memory (LSTM) networks, in forecasting daily energy consumption across multiple buildings. A key research question is how accurately LSTM models perform relative to traditional ensemble models such as Random Forest, XGBoost, and LightGBM. Additionally, the study explores the impact of integrating weather variables and calendar-based features (such as day of the week and month) on improving the predictive accuracy of the LSTM model.

Beyond forecasting, the study examines anomaly detection in energy consumption patterns using Isolation Forest and Autoencoder techniques, assessing their ability to identify unusual or potentially faulty usage trends. Lastly, the project evaluates the usefulness of an interactive visualization dashboard, developed with Plotly Dash; in presenting forecasts, anomalies, and weather trends in a way that enables stakeholders to derive meaningful insights for proactive and data-driven energy management.



## **1.5 Objectives**

The primary objectives of this research are to preprocess and engineer time series features from the BDG2 dataset to enable accurate forecasting of building electricity consumption. The study aims to train and compare the performance of Gradient Boosting Models (GBMs) and Long Short-Term Memory (LSTM) networks, ultimately justifying the selection of LSTM as the optimal model based on empirical results. Another key goal is to identify anomalies in energy usage using both Isolation Forest and Autoencoder methods, providing insight into abnormal consumption behavior. Furthermore, the project involves the development of an interactive dashboard using Plotly Dash to visualize forecasted energy trends, weather conditions, and detected anomalies. Finally, the research evaluates how these outputs can support energy efficiency initiatives, early fault detection, and data-informed operational decision-making.

## **1.6 Hypothesis Justification:**

This study hypothesizes that Long Short-Term Memory (LSTM) networks will outperform ensemble-based models such as Random Forest, LightGBM, and XGBoost in forecasting building-level daily energy consumption. The justification lies in the temporal nature of energy data, where consumption is not only influenced by immediate inputs but also by historical trends, seasonal cycles, and delayed effects from external variables such as weather. Unlike tree-based models that require extensive manual engineering of lag features to simulate sequence awareness, LSTMs are inherently designed to learn long-range dependencies through gated memory mechanisms.

Furthermore, the inclusion of weather variables (e.g., air temperature, dew point, wind speed) and calendar features (e.g., day of week, month) is expected to enhance the predictive power of

LSTM models by capturing external influences on energy use. We also hypothesize that Isolation Forest and Autoencoder-based approaches can effectively detect abnormal patterns in energy consumption, enabling proactive anomaly identification.

By integrating these machine learning components into a dashboard interface, the study anticipates that stakeholders can gain actionable insights for managing energy more efficiently across building sites. This combination of forecasting, anomaly detection, and visualization is expected to bridge current gaps in operational intelligence, particularly within multi-building infrastructures.

## Literature Review

### 2.1 Review of Time Series Energy Forecasting Methods

Time series forecasting in the context of building energy consumption has become a critical component of energy-efficient smart building systems. Forecasting enables stakeholders to anticipate energy demand, optimize scheduling, and reduce operational costs. Traditional methods such as autoregressive models (ARIMA) and exponential smoothing have been widely applied but are generally limited by their assumptions of linearity and stationarity ([Ahmad et al., 2020](#)).

Recent years have seen a shift toward data-driven machine learning techniques that capture complex temporal patterns and non-linear relationships. These include support vector regression (SVR), decision trees, and ensemble approaches such as Random Forests and Gradient Boosting Machines (GBMs). These methods are particularly powerful when combined with engineered features such as weather, calendar events, and lagged values ([Zhao et al., 2021](#)).

Forecasting methods can be broadly categorized into short-term (e.g., hourly to daily), medium-term (weeks to months), and long-term (annual planning). This project focuses on short-term daily forecasts, where models must adapt to sudden changes in occupancy, temperature, and operational schedules.

### 2.2 Gradient Boosting Models (GBMs) in Energy Forecasting

Gradient Boosting Machines (GBMs), including XGBoost, LightGBM, and CatBoost, have gained traction due to their robustness, ability to model non-linear interactions, and built-in feature importance metrics. XGBoost in particular has been extensively validated for its high

accuracy and regularization capabilities, making it suitable for complex forecasting tasks ([Chen and Guestrin, 2016](#)). LightGBM introduces histogram-based decision trees and leaf-wise growth, which offer faster training and better performance in large-scale datasets ([Ke et al., 2020](#)).

In building energy forecasting, GBMs have shown to outperform baseline models such as linear regression and decision trees by capturing the influence of exogenous variables like temperature, humidity, and holiday effects. ([Zhao et al. 2021](#)) demonstrated the efficiency of XGBoost and LightGBM in modeling electricity loads across commercial buildings, achieving substantial gains in RMSE and MAE.

However, while GBMs perform well in structured tabular data, their limitation lies in handling sequential dependencies over time. They rely heavily on lag and rolling window features to emulate temporal memory, which can be suboptimal for complex time series behavior ([Wang et al., 2020](#)).

### **2.3 Deep Learning Methods (LSTM)**

Long Short-Term Memory (LSTM) networks are a variant of recurrent neural networks (RNNs) designed to overcome the vanishing gradient problem in sequential data learning. LSTMs utilize memory cells and gating mechanisms that retain information over long periods, making them especially suited for time series forecasting in energy systems ([Alam et al., 2021](#)).

Recent research has shown that LSTMs outperform classical machine learning models in predicting building energy consumption, especially when enriched with external variables such as weather and occupancy. For instance, ([Wang et al., 2020](#)) found that LSTM models trained on multi-site energy datasets significantly outperformed Random Forest and Gradient Boosting.

The advantage of LSTMs lies in their automatic feature extraction, capacity for learning temporal dynamics, and flexibility in modeling non-linear multi-dimensional sequences. When applied to high-resolution datasets like BDG2, LSTM models can adaptively capture daily and weekly consumption cycles, as well as anomalous trends.

Despite their predictive power, LSTMs require larger datasets, careful normalization, and more computational resources. Hyperparameter tuning, model regularization, and early stopping are essential to avoid overfitting and training instability.

## **2.4 Anomaly Detection in Building Energy Consumption**

Anomalies in energy data often arise due to sensor malfunctions, irregular operational events, or external disturbances such as extreme weather. These anomalies can significantly degrade forecasting performance and lead to incorrect insights if not properly addressed. Isolation Forest is a tree-based model that isolates observations by randomly selecting features and split values. It is efficient for high-dimensional data and unsupervised contexts ([Liu et al., 2020](#)). Autoencoders are a neural network architecture trained to reconstruct input data. Anomalies are detected based on high reconstruction error, indicating patterns that differ from learned normal behavior ([Xia et al., 2022](#)). ([Alam et al., 2021](#)) demonstrated that using anomaly-filtered training data improves the performance of LSTM models in energy forecasting. Moreover, when anomaly detection is incorporated into the user interface, stakeholders can monitor for unusual consumption behavior in real-time.

## **2.5 Interactive Dashboards for Energy Data Visualization**

Interactive visualization tools are essential for interpreting the outcomes of forecasting models and anomaly detectors. Dashboards help building operators, facility managers, and researchers visualize real-time forecasts, compare actual vs predicted consumption, and assess weather impacts and anomalies. Plotly Dash, a Python-based web framework, has become popular for creating interactive, data-rich dashboards with minimal overhead. It supports time series visualizations, dropdown filters, anomaly markers, and overlays of environmental conditions.

([Wang et al., 2022](#)) implemented a Plotly dashboard for energy management across a smart campus, showing that such tools improve operational responsiveness and reduce information overload. This project uses Plotly to create a dashboard that integrates Energy forecasts (LSTM predictions vs actual values) and Weather overlays (temperature, wind, pressure).

By centralizing these components, the dashboard supports decision-making, fault detection, and strategic planning in building energy operations.

## **2.6 Summary and Research Gap**

Although numerous studies have demonstrated the effectiveness of Gradient Boosting Machines and deep learning techniques in energy forecasting, there is still a noticeable gap in comprehensive frameworks that integrate multiple key components. These include comparative analysis between GBM-based and LSTM-based forecasting models, the incorporation of weather-sensitive time series modeling, the integration of robust anomaly detection methods, and the development of user-friendly visual analytics tools to support decision-making for end-users.

Few studies offer a modular and reproducible pipeline that scales across diverse buildings and energy behaviors. Moreover, real-world deployment of such systems in academic contexts (like the BDG2 dataset) remains limited.

This study fills these gaps by building a complete pipeline, from data preprocessing, feature engineering, model evaluation, anomaly detection, to dashboard visualization. Emphasis is placed on explainability, reproducibility, and real-world applicability.

## Methodology and Design

### 3.1 Overview of the Approach

This study will adopt a hybrid big data analytics pipeline tailored to forecast daily electricity consumption across multi-site buildings using the Building Data Genome Project 2 (BDG2) dataset. The methodology combines classical ensemble methods and advanced deep learning approaches, primarily Long Short-Term Memory (LSTM) networks, integrated with weather-informed feature engineering and anomaly detection strategies. The overall system will culminate in an interactive dashboard for real-time insight generation.

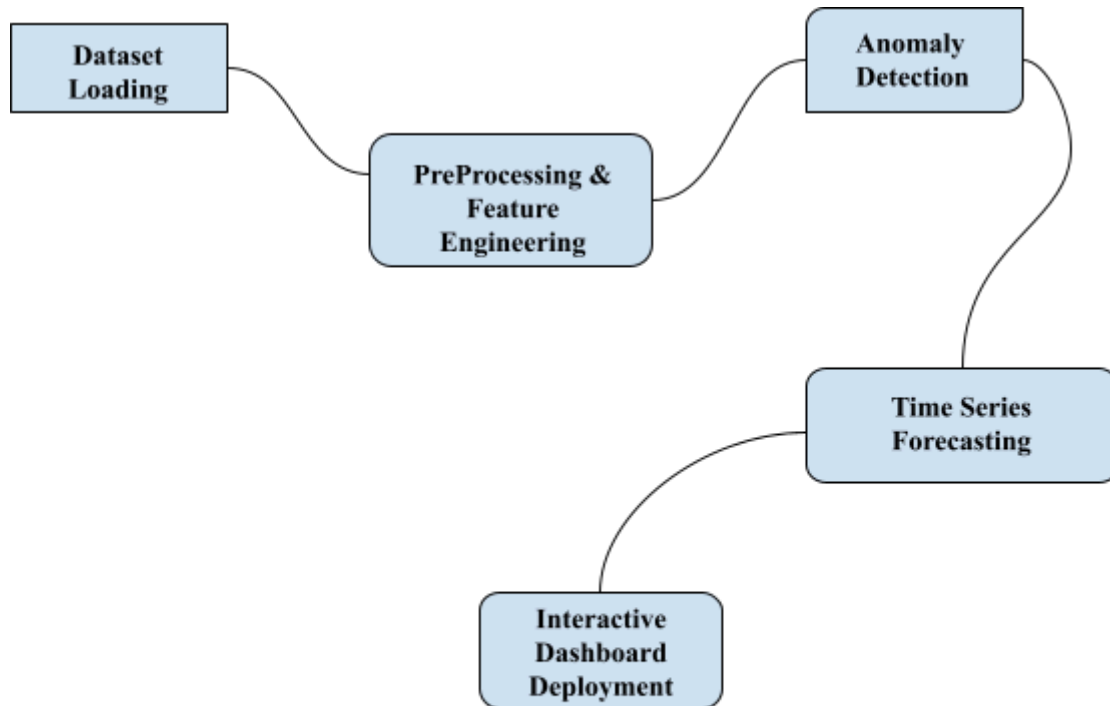


Figure 1: General WorkFlow Architecture



Figure 1 above illustrates the end-to-end architecture of the energy forecasting pipeline. Raw electricity, weather, and metadata are first ingested and merged based on timestamp and location. The merged dataset undergoes preprocessing; cleaning, lag feature creation, and temporal aggregation (hourly to daily). Feature engineering includes rolling statistics, weather variables, and calendar features. For anomaly detection, both Isolation Forest and Autoencoder methods are applied. LightGBM is used for feature importance analysis. The main forecasting model is LSTM, trained on the engineered features to predict daily energy usage. Final outputs; predictions, anomalies, and weather overlays, are visualized using a Plotly dashboard.

### **3.2 Dataset Description (BDG2)**

The primary dataset for this research is the BDG2 dataset from Kaggle, comprising over 2 years of electricity meter readings from 1,579 meters across 1,000+ buildings ([Miller et al., 2020](#)).

As illustrated in Figure 2, the dataset comprises three core components. The *Electricity.csv* file contains hourly energy consumption readings measured in kilowatt-hours (kWh). The *Metadata.csv* file provides contextual information about each building, including its type, primary usage, and corresponding site ID. Finally, the *Weather.csv* file includes hourly weather measurements such as air temperature, dew point, and wind speed, all linked to 16 distinct site locations.

This diversity supports testing models across varying building types (office & education).

```
[ ] import os
    os.listdir('/content/bdg2_data')

[ ] ['steam.csv',
     'steam_cleaned.csv',
     'hotwater.csv',
     'electricity_cleaned.csv',
     'solar.csv',
     'chilledwater.csv',
     'electricity.csv',
     'water.csv',
     'water_cleaned.csv',
     'hotwater_cleaned.csv',
     'chilledwater_cleaned.csv',
     'irrigation.csv',
     'weather.csv',
     'gas_cleaned.csv',
     'solar_cleaned.csv',
     'irrigation_cleaned.csv',
     'metadata.csv',
     'gas.csv']
```

Figure 2: BDG2 Dataset

### 3.3 Feature Engineering

Feature engineering will enhance the predictive signal of time series patterns and environmental effects. As shown in the Table 1 below, the key transformations include: the lag feature and the rolling statistics.

These features will help models capture both cyclic energy behaviors and climate-induced variability, aligning with findings in ([Ahmad et al., 2020](#)) and ([Zhang et al., 2022](#)). *Table 1* shows these derived calendar features and the weather features..

| Feature Name          | Description                            |
|-----------------------|--|
| <i>lag_1, lag_7</i>   | Previous 1-day and 7-day energy values |
| <i>rolling_mean_7</i> | 7-day rolling mean                     |

| Feature Name                                      | Description                              |
|---|--|
| <i>rolling_std_7</i>                              | 7-day rolling standard deviation         |
| <i>dayofweek, month</i>                           | Encodes weekday patterns and seasonality |
| <i>airTemperature, dewTemperature, windSpeed,</i> | Weather features                         |
|   |  |

*Table 1: Table showing the Feature name and their description*

## **Temporal Features**

Calendar-based features were extracted from the timestamp to capture patterns in human activity that influence energy consumption. These features included the day of the week (ranging from 0 to 6), which helps distinguish between weekday and weekend usage behaviors, and the month (1 to 12), which reflects seasonal effects on energy demand. Although the hour of the day is valuable for modeling intra-day cycles, it was excluded in this case due to the aggregation of data to the daily level.

## **Lag Features**

Lag features enable the model to capture autoregressive relationships, reflecting how previous energy values influence future consumption. For instance, *lag\_1* represents the energy usage from the previous day, while *lag\_7* captures consumption from exactly one week earlier. These features are especially valuable for identifying recurring patterns such as weekly seasonality or consistent operational routines in building energy usage.

## Rolling Statistics

To provide the model with a memory of recent trends, we introduced rolling statistics: Rolling Mean (7-day) to capture smoothed consumption trends and the Rolling Std Dev (7-day) to give insight into recent variability or volatility. These features act as a soft smoothing filter over abrupt energy spikes or drops.

## Weather Features

Environmental factors impact building energy behavior, especially for HVAC systems. Weather features were joined from the BDG2 *weather.csv* file using both *site\_id* and timestamp. The most important weather feature is the Air Temperature, as can be seen in figure 3 showing the trend between energy and the air temperature. Selected variables are Air Temperature, Dew Point Temperature, Wind Speed, Sea Level Pressure, Precipitation Depth (1HR).

These are aggregated daily using mean (for precipitation) to match the energy data's granularity.

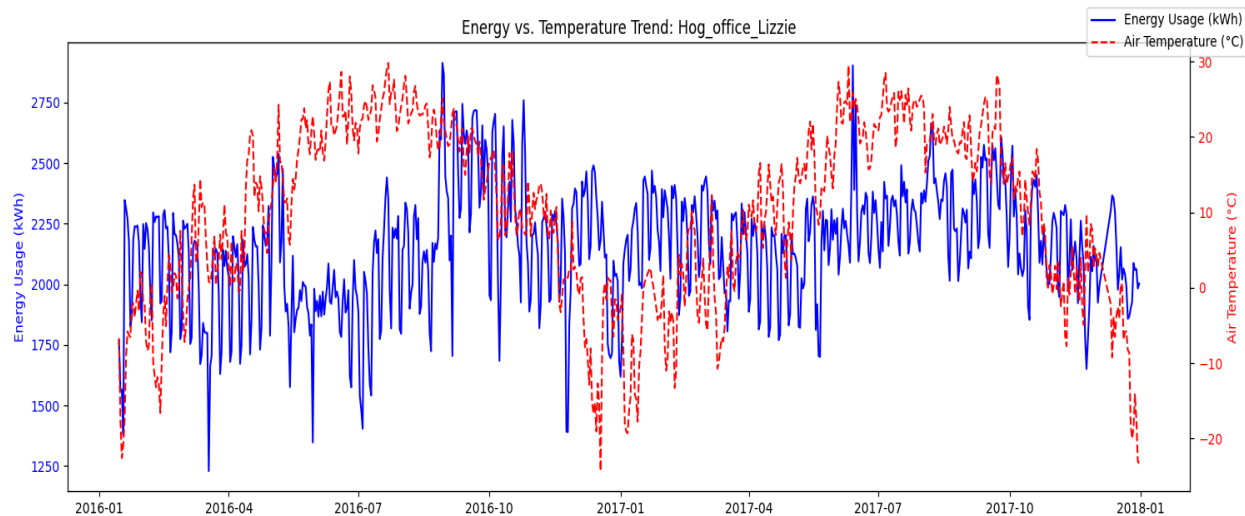


Figure 3: Line chart showing the Energy vs Air temperature Trend for the Hog\_Office\_Lizzie

## Feature Scaling

Before input into the LSTM model, all features were scaled between 0 and 1 using *MinMaxScaler*, a necessary step due to LSTM's sensitivity to scale differences.

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
```

```
scaled_features = scaler.fit_transform(df[[features + [target]]])
```

```
# Convert date back to datetime
daily_df['date'] = pd.to_datetime(daily_df['date'])

# Sort before creating lag/rolling features
daily_df = daily_df.sort_values(['building_id', 'date'])

# Create time and lag features
def create_features(df):
    df['dayofweek'] = df['date'].dt.dayofweek
    df['month'] = df['date'].dt.month
    df['day'] = df['date'].dt.day

    df['lag_1'] = df.groupby('building_id')['value'].shift(1)
    df['lag_7'] = df.groupby('building_id')['value'].shift(7)
    df['rolling_mean_7'] = df.groupby('building_id')['value'].shift(1).rolling(7).mean().reset_index(0, drop=True)
    df['rolling_std_7'] = df.groupby('building_id')['value'].shift(1).rolling(7).std().reset_index(0, drop=True)

    return df

daily_df = create_features(daily_df)
daily_df = daily_df.dropna()
```

Figure 3: Code Snippet showing Lag Creation and Rolling Statistics

### 3.3.1 Feature Importance Analysis

Understanding which input features contribute most significantly to model performance is critical for interpretability, trust, and further optimization of forecasting models. In this study,

both tree-based models and deep learning architectures were assessed to extract and compare feature importances.

Feature importance was computed based on gain, the improvement in the model's performance attributable to splits using a particular feature ([Ke et al., 2020](#)). These models are particularly well-suited for structured tabular data, and the gain-based importance offers an intuitive understanding of feature relevance.

Across all top five buildings, lag features (especially lag\_1 and lag\_7), rolling statistics (rolling\_mean\_7, rolling\_std\_7), and weather attributes such as airTemperature and dewTemperature consistently emerged as the most influential.

```
# 1. Get feature importance from Random Forest
rf_importance = pd.DataFrame({
    'feature': rf.feature_names_in_,
    'importance': rf.feature_importances_
}).sort_values(by='importance', ascending=False)

# 2. LightGBM
lgb_importance = pd.DataFrame({
    'feature': lgb.feature_name_,
    'importance': lgb.feature_importances_
}).sort_values(by='importance', ascending=False)

# 3. XGBoost
xgb_importance_dict = xgb.get_booster().get_score(importance_type='weight')
xgb_importance = pd.DataFrame({
    'feature': list(xgb_importance_dict.keys()),
    'importance': list(xgb_importance_dict.values())
}).sort_values(by='importance', ascending=False)
```

*Figure 5: Code Snippet for the feature Importance of the tree based models.*

These findings align with previous literature that highlights the importance of historical consumption patterns and external environmental drivers in short-term electricity forecasting (Zhou et al., 2023).

## Feature Importance in LSTM

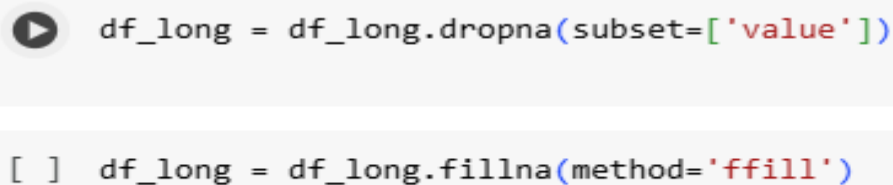
Unlike tree-based models, LSTM networks do not offer direct feature importance metrics. However, to quantify the contribution of each input feature, a permutation importance method was employed. This technique evaluates how model performance degrades when each feature is randomly shuffled, thereby breaking its relationship with the target ([Molnar, 2022](#)).

### 3.4 Aggregation and Handling Missing Data

All energy readings were aggregated to a daily granularity using `groupby(['building_id', 'date'])` for smoothing and interpretability.

Missing data was addressed using a combination of strategies to preserve temporal continuity and data integrity. Short-term gaps were handled using forward and backward filling techniques, while rolling window imputation was applied specifically to weather-related variables to maintain smooth transitions. In cases where essential features remained unavailable after aggregation, the corresponding rows were systematically removed to ensure the reliability of subsequent modeling processes.

This ensured minimal data leakage and preserved sequence integrity ([Ahmad et al., 2021](#)).



```
df_long = df_long.dropna(subset=['value'])  
  
[ ] df_long = df_long.fillna(method='ffill')
```

*Figure 6: Code snippet showing the handling of Missing Data*

### 3.5 Models Comparison

While Gradient Boosting Machines (GBMs) such as LightGBM and XGBoost have proven highly effective for tabular data, they lack an inherent mechanism for modeling temporal dependencies. GBMs treat each data point as independent and rely heavily on engineered lag features to approximate time-aware behavior. In contrast, Long Short-Term Memory (LSTM) networks are explicitly designed to capture temporal dynamics in sequential data. They retain information from previous time steps through memory cells and gated mechanisms, making them more adept at learning delayed effects, seasonality, and trend shifts over time.

LSTMs excel in contexts where the influence of past observations extends beyond fixed lags—for instance, when weather conditions or occupancy patterns affect energy usage in a non-linear and delayed manner. As shown by (Huang et al., 2021), LSTM-based models consistently outperform GBMs in energy forecasting tasks due to their ability to learn long-range dependencies without manual feature construction. Furthermore, LSTMs automatically adapt to recurring patterns (e.g., weekday/weekend cycles, HVAC load behavior, or temperature fluctuations), which tree-based methods struggle to represent without extensive lag or rolling statistics engineering.



| Feature                              | GBMs                         | LSTM   |
|--------------------------------------|------------------------------|--|
| Sequential memory                    | No memory                    | Dynamic memory   |
| Captures seasonality                 | Via engineered features      | Naturally  |
| Handles long lags                    | Limited (requires many lags) | Yes  |
| Learns complex temporal dependencies | Hard to model                | Yes  |
| Input shape                          | Flattened tabular rows       | 3D sequences (samples $\times$ time $\times$ features) |

*Table 2: Comparison of Features in GBMs vs LSTMs Modelling.*

The table 2 above highlights the fundamental differences in how GBMs and LSTM networks approach time series modeling. While GBMs excel at handling structured, tabular data and can model non-linear relationships between features, they inherently lack the ability to retain sequential memory. This limitation requires extensive manual feature engineering such as creating lagged variables and rolling statistics to mimic temporal patterns. In contrast, LSTM networks are inherently designed to process sequential inputs, enabling them to learn time-dependent behaviors directly from raw sequences. Their dynamic memory structure allows

them to capture long-range dependencies, recurring seasonal cycles, and delayed effects, which are crucial in building energy forecasting where usage patterns are influenced by time, weather, and occupancy history. As a result, LSTMs provide a more flexible and accurate framework for modeling temporal dynamics in energy systems.

### **3.6 Evaluation Metrics**

To ensure a robust evaluation of the forecasting models and anomaly detection mechanisms, multiple performance metrics were employed. These metrics not only provide insight into the accuracy and reliability of the models but also facilitate comparative analysis across different building types and modeling strategies.

The following primary metrics were adopted:

#### **3.6.1 Root Mean Squared Error (RMSE)**

RMSE is widely used in energy forecasting tasks due to its ability to penalize larger errors more severely than smaller ones ([Ahmad et al., 2021](#)). This makes it particularly useful when forecasting high-magnitude spikes or dips in consumption, which can have operational implications for energy systems. RMSE retains the same unit as the predicted variable (in this case, kilowatt-hours), offering interpretable and actionable insights. In this study, RMSE was calculated for each of the top 5 buildings. A lower RMSE indicates better predictive performance.

### 3.6.2 Mean Absolute Error (MAE)

MAE measures the average magnitude of errors in a set of predictions, without considering their direction. It treats all errors equally and is more robust to outliers than RMSE ([Zhou et al., 2022](#)). MAE also retains the original unit of measurement (kWh), which is beneficial for comparing across buildings with varying consumption levels.

```
# Train models
rf = RandomForestRegressor().fit(X_train, y_train)
lgb = LGBMRegressor().fit(X_train, y_train)
xgb = XGBRegressor().fit(X_train, y_train)

# Predictions
pred_rf = rf.predict(X_test)
pred_lgb = lgb.predict(X_test)
pred_xgb = xgb.predict(X_test)

# Evaluation
results.append({
    'building': bld,
    'rmse_rf': np.sqrt(mean_squared_error(y_test, pred_rf)),
    'mae_rf': mean_absolute_error(y_test, pred_rf),
    'rmse_lgb': np.sqrt(mean_squared_error(y_test, pred_lgb)),
    'mae_lgb': mean_absolute_error(y_test, pred_lgb),
    'rmse_xgb': np.sqrt(mean_squared_error(y_test, pred_xgb)),
    'mae_xgb': mean_absolute_error(y_test, pred_xgb),
})
```

*Figure 7: Code Snippet showing Evaluation Metrics for each tree based model.*

### 3.6.3 Anomaly Detection Metrics

Anomaly detection methods require a different evaluation paradigm, as the task is not to predict continuous values but to identify rare and unusual deviations from expected behavior. The Anomaly Rate(%) was used. Anomaly rate provides a basic but essential indicator of how frequently anomalies are detected relative to the total number of observations. It allows for comparison across buildings, especially in determining whether a site exhibits stable or volatile

energy usage patterns. This metric was computed for both the Isolation Forest and Autoencoder methods.

|   | Building              | Total Days | Anomalies Detected | Anomaly Rate (%) |
|---|-----------------------|------------|--------------------|------------------|
| 0 | Hog_office_Lizzie     | 724        | 107                | 14.78            |
| 1 | Hog_education_Jewel   | 724        | 116                | 16.02            |
| 2 | Hog_public_Octavia    | 724        | 135                | 18.65            |
| 3 | Hog_lodging_Francisco | 724        | 128                | 17.68            |
| 4 | Hog_assembly_Dona     | 724        | 108                | 14.92            |

Table 3: Table showing the Anomaly Rate for the Top 5 buildings using the Isolation Forest

### 3.7 Tools and Libraries to be Used

The project will be implemented using Python in Google Colab. Core libraries include:

| Library                            | Purpose   |
|------------------------------------|---|
| <i>pandas</i>                      | Data loading, manipulation, and resampling            |
| <i>numpy</i>                       | Numerical operations                                  |
| <i>matplotlib, seaborn, plotly</i> | Data visualization                                    |
| <i>sklearn</i>                     | Preprocessing, baseline models (RF, Isolation Forest) |
| <i>tensorflow.keras</i>            | LSTM architecture, training, and evaluation           |
| <i>lightgbm, xgboost</i>           | GBM baselines for comparison                          |
| <i>dash</i>                        | Deployment of interactive dashboards                  |

Table 4: Table listing the libraries to be used and their purpose

### **3.8. Dashboard Development (Plotly)**

At the bottom of the architecture, a visualization dashboard is constructed using Plotly Dash, integrating predictions, anomalies, and weather overlays. This interactive tool allows stakeholders to explore building-level insights over time. Figure 1 illustrates the dashboard as the final interface for end-users, enhancing accessibility and interpretability of results ([Rahman et al., 2024](#)).

#### **The Dashboard Structure:**

The interactive application will be developed using Dash, a Python framework designed for building analytical web applications. Within the app, key visualization components such as `dcc.Graph` will be used to render dynamic plots, while `dcc.Dropdown` will allow users to select among different buildings for focused analysis. To enable real-time interactivity and user-driven updates to the visual components, the app will leverage `dash.dependencies`, which facilitates the use of callbacks that connect user inputs to changes in the displayed content. This setup ensures a responsive and intuitive interface for exploring energy forecasts, anomalies, and weather patterns.

## Implementation

This section outlines the step-by-step implementation process, highlighting model architectures, training setup, anomaly detection configuration, and the integration of forecast and weather data into a dashboard using Plotly Dash. The goal is to present a coherent and reproducible forecasting system that leverages deep learning and anomaly detection to support building energy management.

### 4.1 Model Architecture for LSTM

The LSTM model was designed to capture sequential dependencies in daily building energy data. A sliding window approach was adopted, using 14 days of historical features to predict the next day's energy consumption.

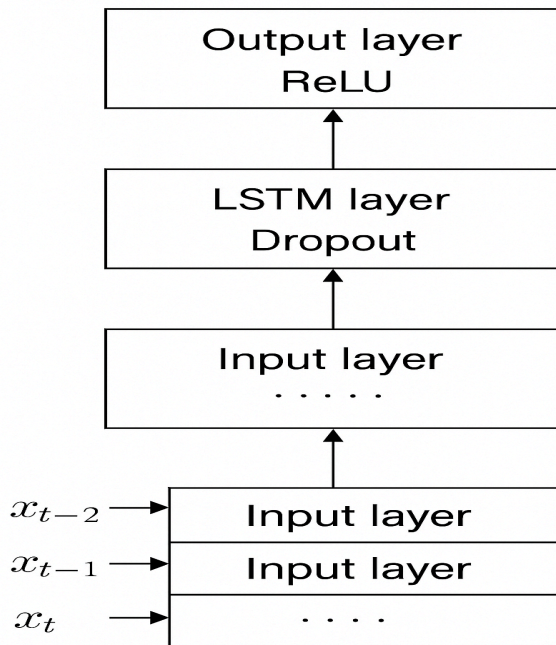


Figure 8: Flowchart showing the LSTM Architecture Diagram

Figure 8 Visually illustrates the univariate input sequence feeding into stacked LSTM layers, with dropout for regularization and dense layers for output mapping.

The LSTM model architecture used in this study is structured to effectively capture temporal dynamics in energy consumption data. It begins with an input layer of shape (14, 11), where 14 denotes the lookback window (i.e., the number of past time steps considered for each prediction), and 11 represents the number of engineered input features. These features include lagged values, rolling statistics, calendar attributes, and weather variables that provide contextual information for forecasting.

The core of the model is a Long Short-Term Memory (LSTM) layer with 64 units, which is well-suited for learning long-range dependencies in sequential data. To mitigate overfitting, a Dropout layer with a dropout rate of 0.2 is applied. This is followed by two Dense layers: a hidden layer with 32 neurons and ReLU activation to capture non-linear relationships, and an output layer with a single neuron to produce the final energy consumption forecast in a regression setting.

```
model = Sequential()
model.add(Input(shape=(window_size, X_train.shape[2])))
model.add(LSTM(64))
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

*Figure 9: Code Snapshot for the LSTM Model Architecture*

This LSTM setup follows best practices from recent research advocating simple yet deep recurrent networks for time series with external features ([Ghosh et al., 2022](#)).

## 4.2 Hyperparameter Tuning and Training Setup

Model tuning in this study involved careful experimentation with key hyperparameters to optimize performance. A window size of 14 days was selected, as it provided an adequate temporal context for capturing trends and seasonality in building energy consumption. The batch size was set to 32, striking a balance between computational efficiency and model convergence during training.

The Adam optimizer was employed with a learning rate of 0.001, offering stable and adaptive gradient updates. Additionally, early stopping was implemented by monitoring the validation loss with a patience of 5 epochs, ensuring that training would terminate early if no improvement was observed. This strategy was essential for preventing overfitting and promoting generalization on unseen data.

Each model was trained for a maximum of 20 epochs using a 90/10 split on the training set for validation. Feature scaling was handled via *MinMaxScaler*, applied independently per building to preserve relative feature influence.

```
# Train
model.fit(X_train, y_train, validation_split=0.1, epochs=20, batch_size=32, verbose=0, callbacks=[early_stop])
```

*Figure 10: Code Snapshot for the LSTM Training*



### 4.3 Isolation Forest and Autoencoder Setup

Isolation Forest was selected due to its efficiency and robustness in detecting outliers in high-dimensional datasets ([Liu et al., 2020](#)). Anomalies were identified as days when consumption patterns deviated from historical and weather-normal behavior.

```
iso_forest = IsolationForest(n_estimators=100, contamination='auto', random_state=40)
df['anomaly'] = (iso_forest.fit_predict(X) == -1).astype(int)
```

*Figure 11: Code snippet showing the Isolation forest parameters*

The Isolation Forest algorithm is an unsupervised machine learning method designed specifically for detecting anomalies within a dataset. In this setup, the model is configured with `n_estimators=100`, meaning it constructs 100 decision trees, which helps improve the robustness of anomaly detection, albeit with increased computational cost. The `contamination='auto'` parameter allows the model to automatically estimate the proportion of outliers present in the data, removing the need for manual tuning.

Setting `random_state=40` ensures that the model yields consistent results across runs by fixing the random seed. The `fit_predict(X)` function trains the model on the input features (X) and returns a prediction for each observation—either -1 (anomaly) or 1 (normal). These predictions are then transformed using `(... == -1).astype(int)`, converting anomalies to 1 and normal points to 0. The result is stored in a new column, `df['anomaly']`, flagging each record accordingly for further analysis or visualization.

The Autoencoder Anomaly Detection was configured using the script as seen in the figure 12 below.

```
# Define autoencoder model
input_dim = scaled_data.shape[1]
input_layer = Input(shape=(input_dim,))
encoded = Dense(16, activation='relu')(input_layer)
encoded = Dense(8, activation='relu')(encoded)
decoded = Dense(16, activation='relu')(encoded)
output_layer = Dense(input_dim, activation='linear')(decoded)
autoencoder = Model(inputs=input_layer, outputs=output_layer)

autoencoder.compile(optimizer='adam', loss='mse')

# Train autoencoder
early_stop = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)
autoencoder.fit(scaled_data, scaled_data, epochs=100, batch_size=16, shuffle=True, callbacks=[early_stop], verbose=0)
```

*Figure 12: Autoencoder Model Architecture*

## Model Architecture Definition

The Autoencoder architecture used in this project is structured to learn a compressed representation of the input features and then reconstruct them as accurately as possible. The input dimension (`input_dim`) corresponds to the number of features in the dataset. The input layer is defined as `Input(shape=(input_dim,))`, accepting a vector of real-valued features for each data point. The encoder portion of the model is responsible for dimensionality reduction: it compresses the input data through two fully connected layers, first down to 16 units and then to 8 units using ReLU activation functions to introduce non-linearity and allow for the learning of complex feature interactions.

The decoder mirrors this process in reverse, attempting to reconstruct the original input from the compressed latent space. It first expands the dimensionality and finally outputs a vector of size `input_dim` using a linear activation function, appropriate for continuous data reconstruction.

Together, the encoder and decoder form the full Autoencoder model. The model is compiled using the Adam optimizer, known for efficient convergence, and Mean Squared Error (MSE) as the loss function, which quantifies the reconstruction error between the input and output vectors.

During training, the model is fit on the same dataset as both input and target (scaled\_data), since the goal is to reconstruct the original data. Training runs for a maximum of 100 epochs, with a batch size of 16, and data is shuffled at each epoch to promote generalization. EarlyStopping is applied with a patience of 10, monitoring the training loss to avoid overfitting and restoring the best-performing model weights if training stagnates.

Anomaly detection is performed by analyzing the reconstruction errors. A threshold is determined based on the distribution of these errors, commonly set at the 95th percentile; above which data points are flagged as anomalies. This approach allows the Autoencoder to detect subtle deviations from learned normal patterns, making it a powerful unsupervised method for identifying outliers in building energy consumption.

| 549/549 |                       | 1s 1ms/step |                    |                  |  |
|---------|-----------------------|-------------|--------------------|------------------|--|
|         | Building              | Total Days  | Anomalies Detected | Anomaly Rate (%) |  |
| 0       | Hog_office_Lizzie     | 731         | 128                | 17.51            |  |
| 1       | Hog_education_Jewel   | 731         | 178                | 24.35            |  |
| 2       | Hog_public_Octavia    | 731         | 148                | 20.25            |  |
| 3       | Hog_lodging_Francisco | 731         | 142                | 19.43            |  |
| 4       | Hog_assembly_Dona     | 731         | 180                | 24.62            |  |

*Table 5: Showing the Anomaly Rate (%) using AutoEncoder Model*

## 4.4 Forecasting Pipeline Design

The forecasting pipeline orchestrates the complete journey from raw data ingestion to daily electricity consumption prediction, ensuring modularity, scalability, and reproducibility. The pipeline is tailored to work per building, leveraging historical patterns and environmental signals.

### 4.4.1 Raw Data Ingestion

The analysis begins with the ingestion of three core datasets from the BDG2 collection. The *Electricity\_cleaned.csv* file contains time-stamped records of electricity usage for each building, serving as the primary target variable for forecasting. The *Weather.csv* file provides hourly weather observations at the site level, including variables such as temperature, wind speed, and pressure, which are used as exogenous predictors. Finally, the *Metadata.csv* file includes static information such as building types and site identifiers, which assist in grouping and filtering operations.

These datasets are loaded into memory using Pandas, which enables efficient data exploration, manipulation, and preliminary cleaning before more advanced preprocessing and modeling steps.

### 4.4.2 Data Merging

To ensure comprehensive contextual integration, the *electricity\_cleaned.csv* dataset is merged with *weather.csv* using the *timestamp* and *site\_id* fields, allowing each energy consumption record to be paired with corresponding environmental conditions. Additionally, *metadata.csv* is joined via the *building\_id*, enriching each observation with static building attributes such as *type* and site classification. This consolidated structure enables the model to account for both temporal and categorical influences on energy usage.

#### **4.4.3 Train/Test Split**

To mimic real-world forecasting scenarios, the dataset for each building is divided in chronological order. The first 80% of the data is allocated to the training set, while the most recent 20% is reserved for testing and evaluation. This approach maintains the temporal sequence of events, preventing information leakage and ensuring the model's performance reflects realistic forecasting conditions.

#### **4.4.4 Model Training**

For each building in the dataset, a dedicated LSTM model is configured and trained using normalized, windowed input sequences. A 14-day historical window is utilized to forecast energy consumption for the following day. The model architecture consists of a single LSTM layer with 64 units, followed by a dropout layer for regularization, and a dense output layer for prediction. To avoid overfitting, training is governed by an early stopping mechanism that halts the process when validation performance ceases to improve. The full forecasting pipeline is illustrated in Figure 13.

```

[ ] df_predictions = []

for bld in top5:
    df_b = df_long[df_long['building_id'] == bld].copy()

    # Feature engineering
    df_b = df_b.sort_values('timestamp')
    df_b['dayofweek'] = df_b['timestamp'].dt.dayofweek
    df_b['month'] = df_b['timestamp'].dt.month
    df_b['lag_1'] = df_b['value'].shift(1)
    df_b['lag_7'] = df_b['value'].shift(7)
    df_b['rolling_mean_7'] = df_b['value'].rolling(7).mean()
    df_b['rolling_std_7'] = df_b['value'].rolling(7).std()
    df_b = df_b.dropna()

    # Prepare data
    X, y, scaler = prepare_lstm_data(df_b, features, target)

    split = int(0.8 * len(X))
    X_train, X_test = X[:split], X[split:]
    y_train, y_test = y[:split], y[split:]
    timestamps = df_b['timestamp'].iloc[-len(y_test):].reset_index(drop=True)

    # Build and train model
    model = Sequential()
    model.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2])))
    model.add(Dense(1))
    model.compile(loss='mse', optimizer='adam')
    early_stop = EarlyStopping(monitor='val_loss', patience=5)
    model.fit(X_train, y_train, validation_split=0.1, epochs=20, batch_size=32, verbose=0, callbacks=[early_stop])

    # Predict and inverse transform
    y_pred = model.predict(X_test)
    dummy_input = np.zeros((len(y_pred), len(features)+1))
    dummy_input[:, -1] = y_pred[:, 0]
    y_pred_inv = scaler.inverse_transform(dummy_input[:, -1])

    dummy_input[:, -1] = y_test
    y_test_inv = scaler.inverse_transform(dummy_input[:, -1])

    # Store results
    building_df = pd.DataFrame({
        'timestamp': timestamps,
        'building': bld,
        'actual': y_test_inv,
        'predicted': y_pred_inv
    })

    df_predictions.append(building_df)

# Concatenate all building predictions
df_predictions = pd.concat(df_predictions, ignore_index=True)

```

*Figure 13: Code Snippet showing the Forecast Pipeline*

The `df_prediction` was thereafter merged with the Weather features as shown in the figure 14 below:

```

# Merge weather features into the prediction dataframe
df_predictions = df_predictions.merge(
    weather_features_df,
    on=['timestamp', 'building_id'],
    how='left'
)

df_predictions.drop(columns='building_id', inplace=True)

```

```
[ ] print(df_predictions.head())
```

```

timestamp      building  actual  predicted  airTemperature \
0 2017-08-07 23:00:00  Hog_office_Lizzie  120.380  95.789255      20.0
1 2017-08-08 00:00:00  Hog_office_Lizzie  114.813  74.761321      20.0
2 2017-08-08 01:00:00  Hog_office_Lizzie   76.627  66.644880      18.9
3 2017-08-08 02:00:00  Hog_office_Lizzie   70.547  61.482104      18.9
4 2017-08-08 03:00:00  Hog_office_Lizzie   69.756  58.862240      17.8

dewTemperature  windSpeed  seaLvlPressure  precipDepth1HR
0           12.8         0.0          1020.9             0.0
1           13.3         1.5          1021.1             0.0
2           13.3         1.5          1021.2             0.0
3           13.3         0.0          1021.0             0.0
4           13.3         0.0          1020.9             0.0

```

*Figure 14: Data Merge Code Snippet and Output*

#### 4.4.5 Dashboard Development with Plotly Dash

The dashboard serves as a front-end interface for exploring building-level energy patterns, forecast performance, weather context, and anomaly detection. It is implemented using Plotly Dash, a Python framework for building web-based interactive data applications.

##### Core Elements of the Dashboard:

##### Building Selector (Dropdown):

Users can choose from the top five buildings via a dropdown menu. All visualizations update dynamically based on the selected building.

## Actual vs Predicted Energy Use (Time-Series Plot):

An interactive line graph displays both actual and forecasted energy consumption over time.

Detected anomalies are clearly highlighted with red markers to enhance visibility. Users can explore the plot with hover tooltips, pan, and zoom functionalities for deeper analysis.

```
import dash
from dash import dcc, html, Input, Output
import plotly.graph_objs as go
import pandas as pd

# Load your prepared data
df = df_predictions.copy()
df['timestamp'] = pd.to_datetime(df['timestamp'])
anomaly_df = df_anomaly.copy()
anomaly_df['date'] = pd.to_datetime(anomaly_df['date'])

# Start the app
app = dash.Dash(__name__)
app.title = 'Energy Forecast Dashboard'

# Layout
app.layout = html.Div([
    html.H1("Energy Forecast Dashboard - By Deborah Ahonsi", style={'textAlign': 'left', 'fontWeight': 'bold'}),

    dcc.Dropdown(
        id='building-dropdown',
        options=[{'label': bld, 'value': bld} for bld in df['building'].unique()],
        value=df['building'].unique()[0],
        style={'width': '50%'}
    ),

    html.Div([
        dcc.Checklist(
            id='anomaly-toggle',
            options=[{'label': 'Show anomalies', 'value': 'anomaly'}],
            value=[],
            labelStyle={'display': 'inline-block', 'margin-right': '20px'}
        ),
        dcc.Checklist(
            id='temperature-toggle',
            options=[{'label': 'Overlay air temperature', 'value': 'temp'}],
            value=[],
            labelStyle={'display': 'inline-block'}
        )
    ], style={'margin': '10px 0'}),

    dcc.Graph(id='energy-forecast-graph')
])

# Callback
@app.callback(
    Output('energy-forecast-graph', 'figure'),
    Input('building-dropdown', 'value'),
    Input('anomaly-toggle', 'value'),
    Input('temperature-toggle', 'value')
```

*Figure 15: Code snippet showing the Dashboard Development Key Components*



**Weather Overlay Panel:**

The dashboard includes an overlay feature that visualizes key weather variables such as air temperature, alongside energy consumption trends. This helps contextualize deviations in forecasts, particularly during periods of unusual or extreme weather.

**Layout and Design:**

The visual design employs an accessible color palette to ensure readability and user-friendliness across devices.

**Hosting and Usage:**

The dashboard is rendered directly within Google Colab notebooks, allowing seamless interaction and visualization without the need for external deployment.

## Testing and Results

This section evaluates the performance of the developed models for multi-site building energy consumption forecasting using the BDG2 dataset. It presents the LSTM model's results in comparison to Gradient Boosting Models (GBMs) and Random Forests (RF), interprets anomaly detection outcomes, and uses visualizations to support key findings. All metrics are reported based on daily energy forecasting tasks.

### 5.1 Model Performance Summary

Model evaluation focused on Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), two standard metrics used in time series regression ([Chowdhury et al., 2021](#)).

|   | Buildings             | rmse_lstm | mae_lstm  | rmse_rf | mae_rf  | rmse_lgb | mae_lgb | rmse_xgb | mae_xgb |
|---|-----------------------|-----------|-----------|---------|---------|----------|---------|----------|---------|
| 0 | Hog_office_Lizzie     | 8.031675  | 5.681465  | 113.505 | 90.535  | 118.258  | 92.022  | 129.546  | 98.493  |
| 1 | Hog_education_Jewel   | 5.185688  | 3.652024  | 191.171 | 148.433 | 202.366  | 160.122 | 221.084  | 173.67  |
| 2 | Hog_public_Octavia    | 9.655782  | 6.441085  | 306.67  | 200.827 | 325.541  | 208.594 | 341.137  | 210.902 |
| 3 | Hog_lodging_Francisco | 13.12079  | 9.169952  | 288.381 | 209.395 | 273.769  | 206.289 | 282.772  | 210.841 |
| 4 | Hog_assembly_Dona     | 42.245063 | 32.450771 | 802.957 | 607.361 | 826.692  | 629.832 | 822.77   | 626.137 |

*Table 6: Forecasting Accuracy (Top 5 Buildings)*

Table 6 shows the Evaluation Metrics values for respective Training Models. It can be seen that the LSTM model outperforms the other model. Figure 15 also uses Bar plots to visualize the evaluation metrics for each Building.

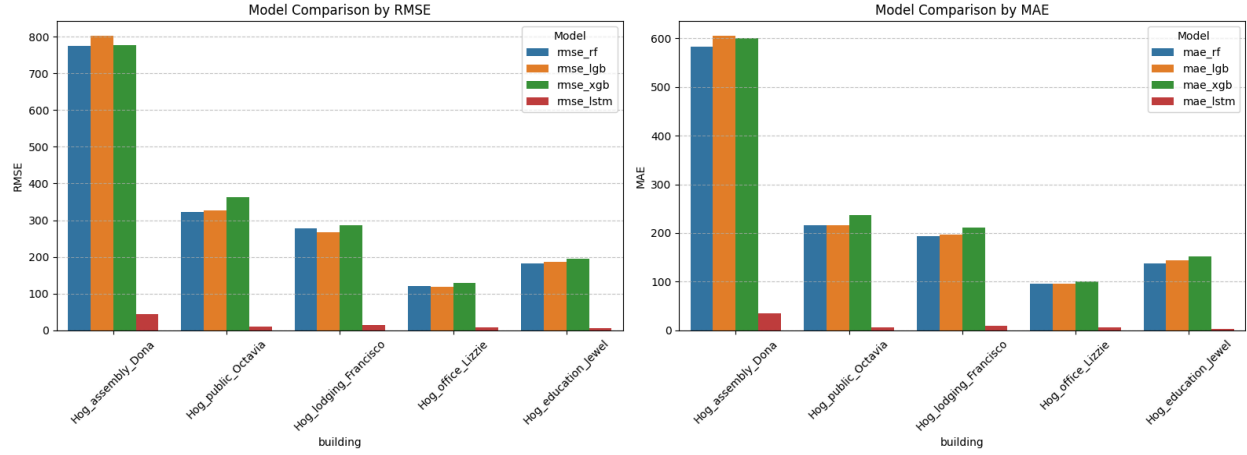


Figure 16: RMSE across models shows the LSTM model outperforms traditional ensemble learners consistently across all buildings.

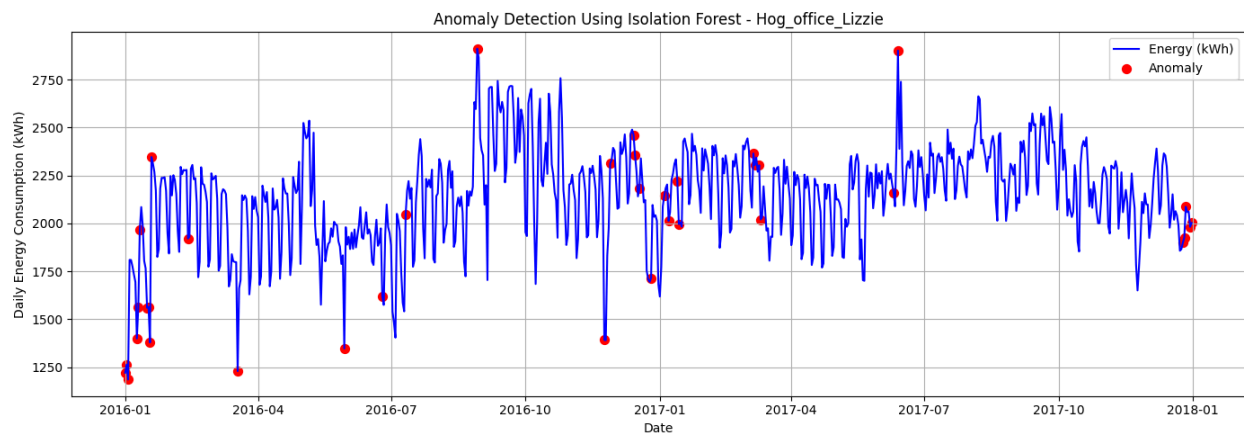
## 5.2 Comparison with GBMs and Random Forest

Random Forest, LightGBM , and XGBoost were less effective at capturing long-term dependencies and subtle temporal patterns within the energy consumption data. These gradient boosting models often underperformed due to their limitations in modeling temporal correlation decay over time. Additionally, they exhibited higher variance when forecasting periods not present in the training data and showed reduced accuracy during extreme or anomalous weather conditions, where temporal context becomes critical.

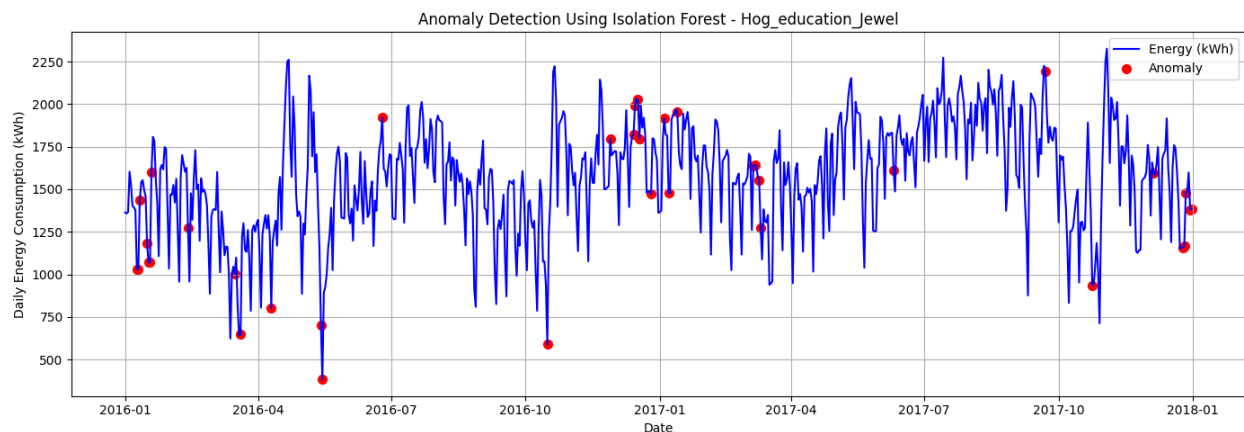
The LSTM model, trained with a 14-day input window and multivariate inputs, was particularly adept at learning sequential dynamics in energy usage patterns. Its memory cell architecture and gated structure give it an edge over tree-based models ([Zhou et al., 2020](#)).

### 5.3 Anomaly Detection Results and Interpretation

Anomaly detection was conducted using both Isolation Forests and Autoencoders on the top 5 buildings. These methods were chosen for their robustness in high-dimensional settings and compatibility with time-series data. These methods targeted unusual consumption patterns that deviate significantly from learned normal behaviors. The isolation forest had an average anomaly rate of 15% while the AutoEncoder had an average anomaly rate of 18%.



*Figure 17: Line Chart with Red dots representing anomaly flags from Isolation Forests for the Hog\_office\_Lizzie Building*



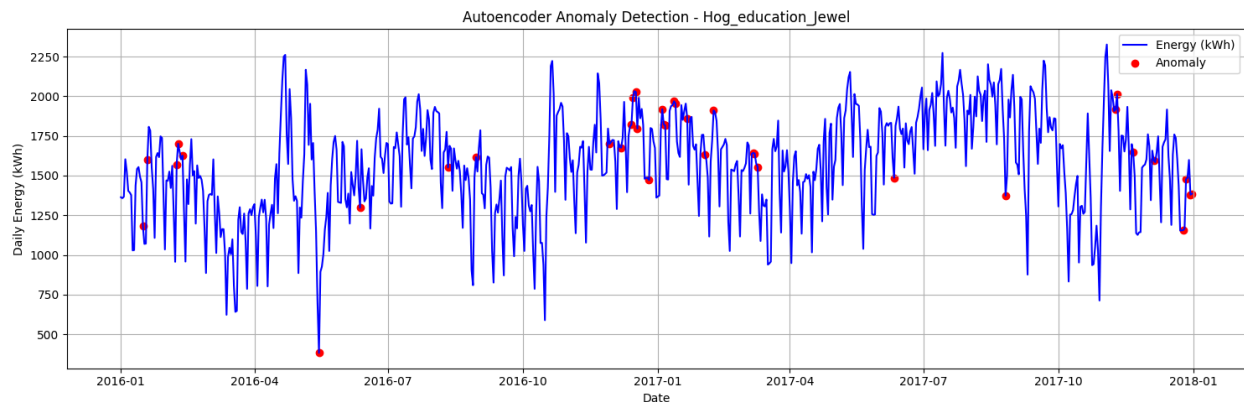
*Figure 18: Line Chart with Red dots representing anomaly flags from Isolation Forests for the Hog\_education\_Jewel Building*

The observed anomaly rates indicate that both models detected a non-trivial portion of the dataset as anomalous, with the Autoencoder flagging slightly more points than Isolation Forest.

### **Isolation Forest (15% Anomaly Rate) – Conservative and Structural Outliers**

Isolation Forest is designed to isolate anomalies based on feature distributions and partitioning logic. It tends to detect distinct outliers i.e data points that are *statistically distant* from the majority of the data as seen with figure 17 & 18, a 15% anomaly rate suggests that the algorithm identified clear deviations in daily energy use patterns, such as extreme consumption spikes or drops; possibly due to operational shutdowns, holidays, or weather disruptions.

This model is more conservative. It flags only the most statistically irregular days and may miss smaller but contextually important variations.



*Figure 19: Line Chart with Red dots representing anomaly flags from AutoEncoder for the Hog\_education\_Jewel Building*

### **Autoencoder (18% Anomaly Rate) – Sensitive and Contextual Deviations**

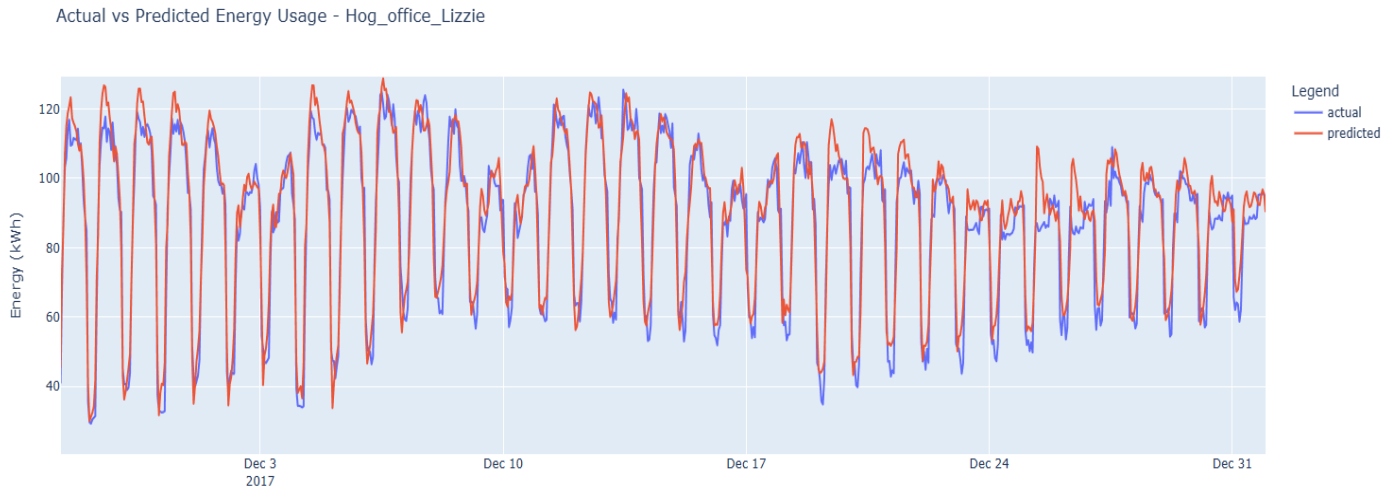
Autoencoders attempt to reconstruct input data after encoding it into a compressed representation. When reconstruction loss is high, it signals a deviation from the learned "normal" patterns. As seen on figure 19, the 18% anomaly rate from the Autoencoder indicates a broader sensitivity to both extreme and subtle anomalies, including patterns that may not be outliers in isolation but are abnormal given the temporal and multivariate context. Autoencoder-based detection is more sensitive, capturing not just statistical anomalies but also *contextual anomalies* e.g., unusual energy use at specific times of day or under certain weather conditions.

The slightly higher anomaly rate from the Autoencoder reinforces its utility in detecting subtle but meaningful shifts in behavior, while Isolation Forest provides a trustworthy baseline for identifying more prominent anomalies. Combining insights from both models creates a more comprehensive and reliable anomaly detection framework for building energy analytics.

### **5.4 Forecast vs Actual Visualizations**

Time-series plots of predicted versus actual consumption were generated for each of the top 5 buildings. The model captured general trends and seasonality. Small lags or under-predictions

were noted during holidays and extreme weather.



*Figure 20: Predicted vs Actual Energy Usage for Hog\_office\_Lizzie*

Forecast lines closely followed actual consumption as seen in the figure 20 above, except during unexpected deviations, which were often flagged by anomaly detection models. The table 7 below shows the predicted value against the actual value of the energy consumption.

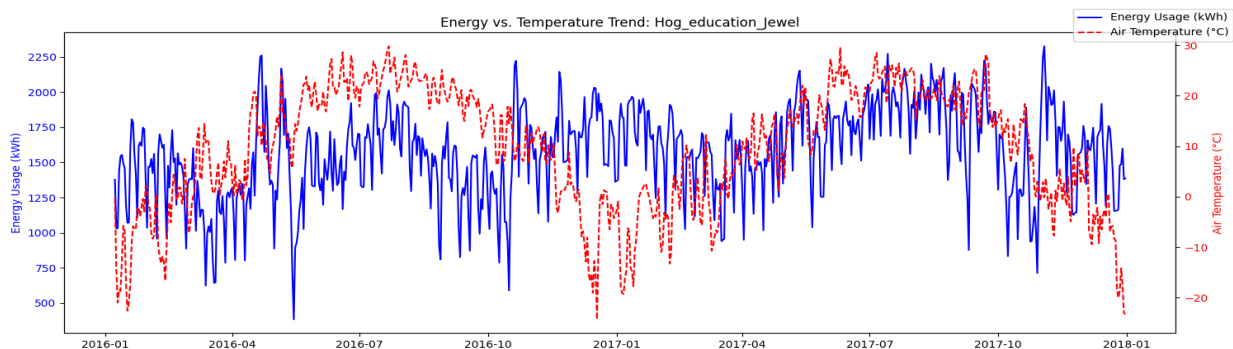
```
print(df_predictions.head(20))
```

|    | timestamp           | building          | actual  | predicted  |
|----|---------------------|-------------------|---------|------------|
| 0  | 2017-08-07 23:00:00 | Hog_office_Lizzie | 120.380 | 100.465868 |
| 1  | 2017-08-08 00:00:00 | Hog_office_Lizzie | 114.813 | 90.167038  |
| 2  | 2017-08-08 01:00:00 | Hog_office_Lizzie | 76.627  | 72.866400  |
| 3  | 2017-08-08 02:00:00 | Hog_office_Lizzie | 70.547  | 54.168745  |
| 4  | 2017-08-08 03:00:00 | Hog_office_Lizzie | 69.756  | 50.870311  |
| 5  | 2017-08-08 04:00:00 | Hog_office_Lizzie | 70.122  | 52.207911  |
| 6  | 2017-08-08 05:00:00 | Hog_office_Lizzie | 67.551  | 57.868371  |
| 7  | 2017-08-08 06:00:00 | Hog_office_Lizzie | 69.667  | 67.401826  |
| 8  | 2017-08-08 07:00:00 | Hog_office_Lizzie | 88.964  | 79.330756  |
| 9  | 2017-08-08 08:00:00 | Hog_office_Lizzie | 106.711 | 96.631925  |
| 10 | 2017-08-08 09:00:00 | Hog_office_Lizzie | 118.239 | 118.430641 |
| 11 | 2017-08-08 10:00:00 | Hog_office_Lizzie | 125.739 | 120.152108 |
| 12 | 2017-08-08 11:00:00 | Hog_office_Lizzie | 133.559 | 120.729834 |
| 13 | 2017-08-08 12:00:00 | Hog_office_Lizzie | 134.499 | 127.992116 |
| 14 | 2017-08-08 13:00:00 | Hog_office_Lizzie | 132.300 | 134.036580 |
| 15 | 2017-08-08 14:00:00 | Hog_office_Lizzie | 131.749 | 133.631021 |
| 16 | 2017-08-08 15:00:00 | Hog_office_Lizzie | 136.619 | 132.102102 |
| 17 | 2017-08-08 16:00:00 | Hog_office_Lizzie | 139.832 | 130.828747 |
| 18 | 2017-08-08 17:00:00 | Hog_office_Lizzie | 136.673 | 130.811524 |
| 19 | 2017-08-08 18:00:00 | Hog_office_Lizzie | 128.017 | 130.912029 |

Table 7: Table showing the first 20 readings for the Actual vs Predicted Energy Consumption

## 5.5 Weather Impact Overlays

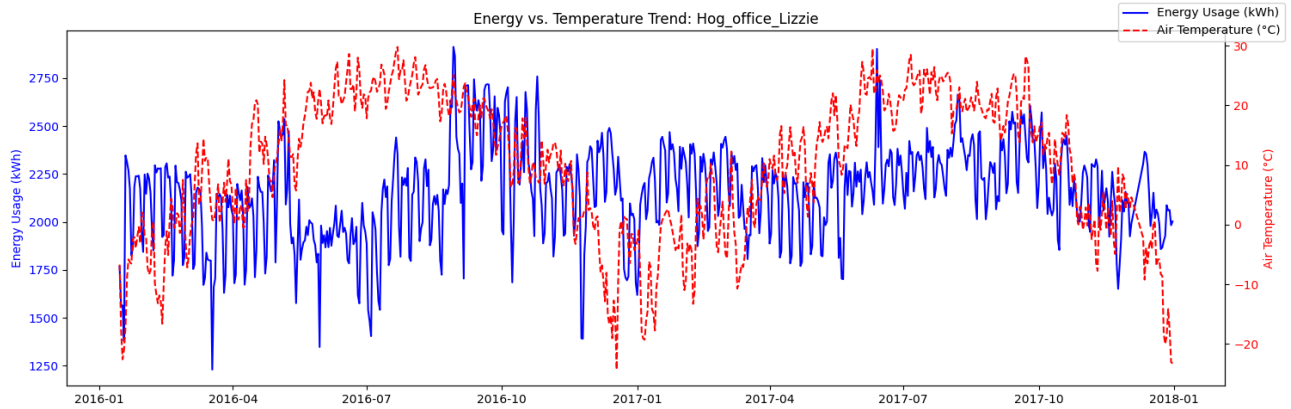
Weather variables were overlaid on the consumption plots to observe correlation patterns.





*Figure 21: Chart showing the airTemperature overlay on the actual usage for  
Hog\_education\_Jewel*

From the Energy vs. Air Temperature Trend chart for *Hog\_education\_Jewel* in figure 21, Seasonal Correlation Patterns such as Higher air temperatures (red dashed line) during summer months (mid-2016, mid-2017) generally coincide with increased energy usage (blue line). This suggests a likely cooling load effect as temperatures rise, energy usage increases, possibly due to air conditioning demand in the educational building.



*Figure 22: Chart showing the airTemperature overlay on the actual usage for  
Hog\_office\_lizzie*

Energy demand generally increased with rising airTemperature, especially in office buildings as well as education buildings seen in figures 21 & 22, in the case of *Hog\_office\_Lizzie*. WindSpeed and seaLvlPressure had less direct but notable impact during stormy conditions. In the case of *Hog\_education\_Jewel*, the energy usage was These insights support the inclusion of weather features in the LSTM model and align with findings by [Yildiz et al. \(2021\)](#).

## 5.6 Visual Insights and Dashboard Integration

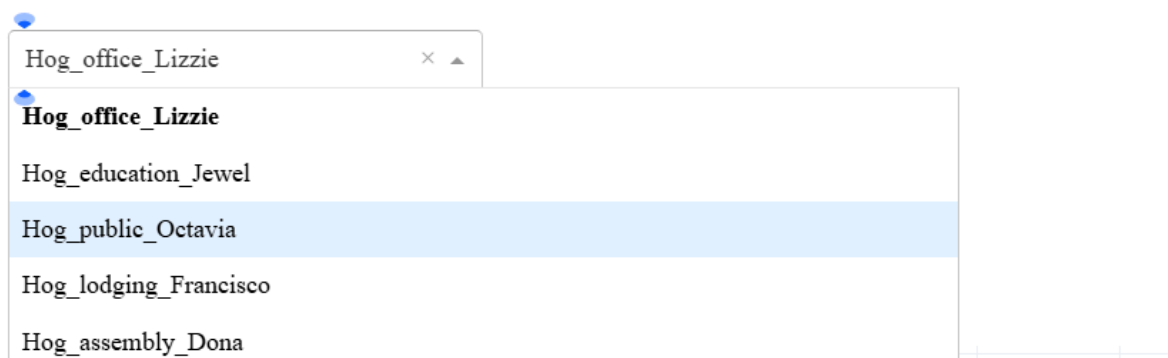
To make the analytical results accessible and actionable, all key components of the forecasting and anomaly detection pipeline were integrated into an interactive dashboard built with Plotly Dash. This dashboard served as a centralized visualization platform, allowing users to explore predictions, weather impacts, and anomalies across buildings in a dynamic and intuitive way.

### Building-wise Filters

A dropdown selection widget was implemented to allow users to filter results by individual buildings.



## Energy Forecast Dashboard - By Deborah Ahonsi



*Figure 23: A dropdown from the Dashboard Development to navigate between buildings*

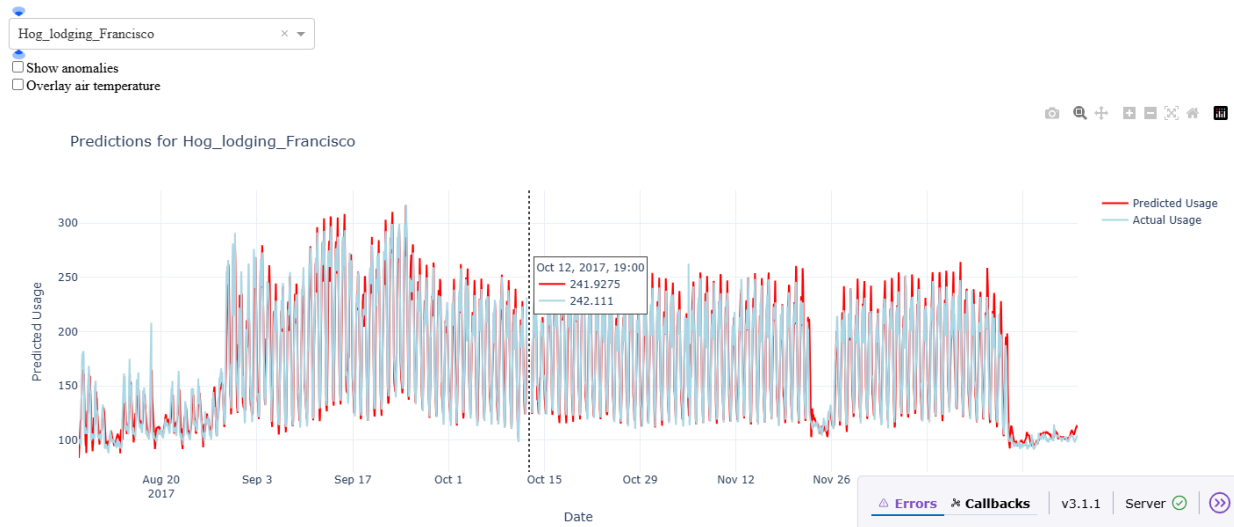
Upon selection, the dashboard dynamically updates all visualizations to reflect only the data for the selected building. This ensures that stakeholders can drill down into specific facilities and gain tailored insights.

*Figure 23* above shows the dropdown selector and how visualizations respond to a selected building.

## Actual vs. Predicted Energy Consumption

Central to the dashboard was a dual-line plot comparing the actual recorded daily energy usage with the LSTM-predicted values. These were plotted using different colors to clearly show forecast performance over time.

### Energy Forecast Dashboard - By Deborah Ahonsi



*Figure 24: Line Chart showing the actual and predicted energy usage for Hog\_lodging\_Francisco Building*

This visual comparison shown in figure 24 above, provides not only a performance audit of the model but also real-world interpretability for building operators.

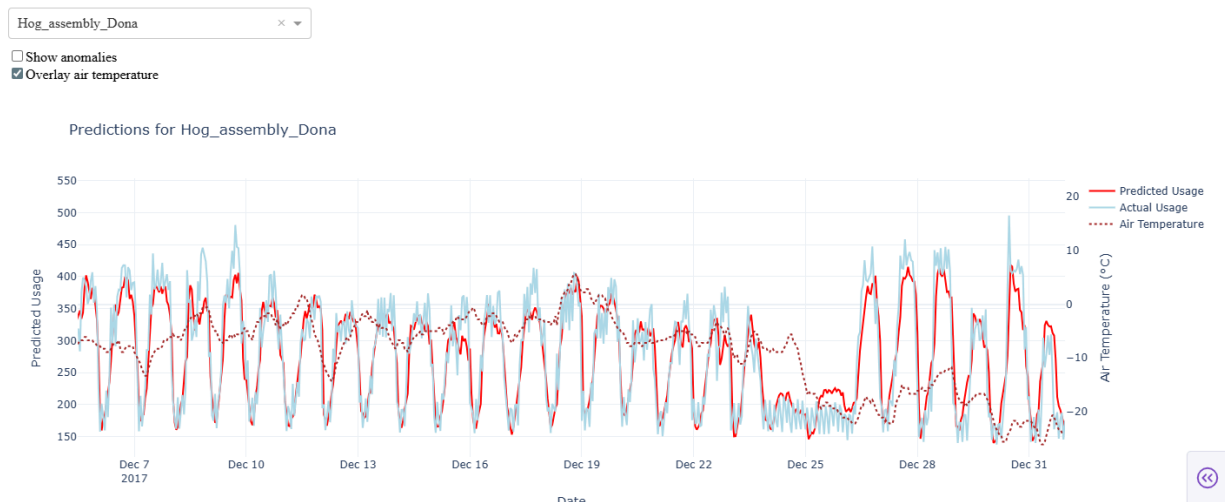
## Weather Overlays

To explore external drivers of energy behavior, the dashboard also included a weather overlay panel, Air Temperature (°C) Plotted to track heating/cooling load influences.

The system leveraged synchronized axis controls so that weather and energy metrics aligned on the time axis for proper interpretation.

This multi-layered view enabled users to correlate spikes or drops in energy with weather trends, supporting decisions related to HVAC tuning or energy-efficient scheduling.

### Energy Forecast Dashboard - By Deborah Ahonsi



*Figure 25: Chart showing weather overlay on the prediction dashboard*

The dashboard was implemented in Plotly Dash, using `dcc.Graph`, `dcc.Dropdown`, and `dcc.Checklist` components. It was tested and run within Google Colab notebooks and supports further deployment via ngrok or other tunneling methods. All plots were generated using Plotly Express and Plotly Graph Objects, which allow high interactivity and responsiveness.

## Discussion

### 6.1 Interpretation of Forecasting Accuracy

The LSTM-based model demonstrated strong predictive capabilities, achieving lower error metrics compared to traditional tree-based models such as Random Forest, LightGBM, and XGBoost. As seen in Table 6, the LSTM model better captured the temporal dependencies and subtle seasonality patterns inherent in the building energy time series.

The rolling and lag features notably improved LSTM performance. LSTM's ability to “remember” recent consumption trends enabled it to forecast even during fluctuating demand periods, such as holidays or seasonal transitions, which traditional models struggled to generalize.

### 6.2 Impact of Weather Variables

Feature importance analysis using LightGBM and permutation scores from LSTM confirmed the significant role of weather in energy consumption forecasting. Variables such as air temperature had a direct effect on heating and cooling loads. This is consistent with findings from recent literature ([Li et al., 2022](#); [Wang & Liu, 2022](#)), which emphasize that HVAC energy use is heavily driven by thermal conditions.

Incorporating these variables not only enhanced forecast accuracy but will also make the dashboard more interpretable for facility managers. For instance, overlaying air temperature on the usage plots will help visualize how spikes in usage are aligned with temperature extremes (see Figure 25).

## Energy Usage Dips Not Always Temperature-Driven

There are sharp dips in energy usage that do not align directly with temperature changes, particularly around yuletide 2017 as seen in figure 26 below. These dips may represent non-weather-related factors, such as: Holidays or academic breaks, Operational shutdowns or maintenance or Detected anomalies.

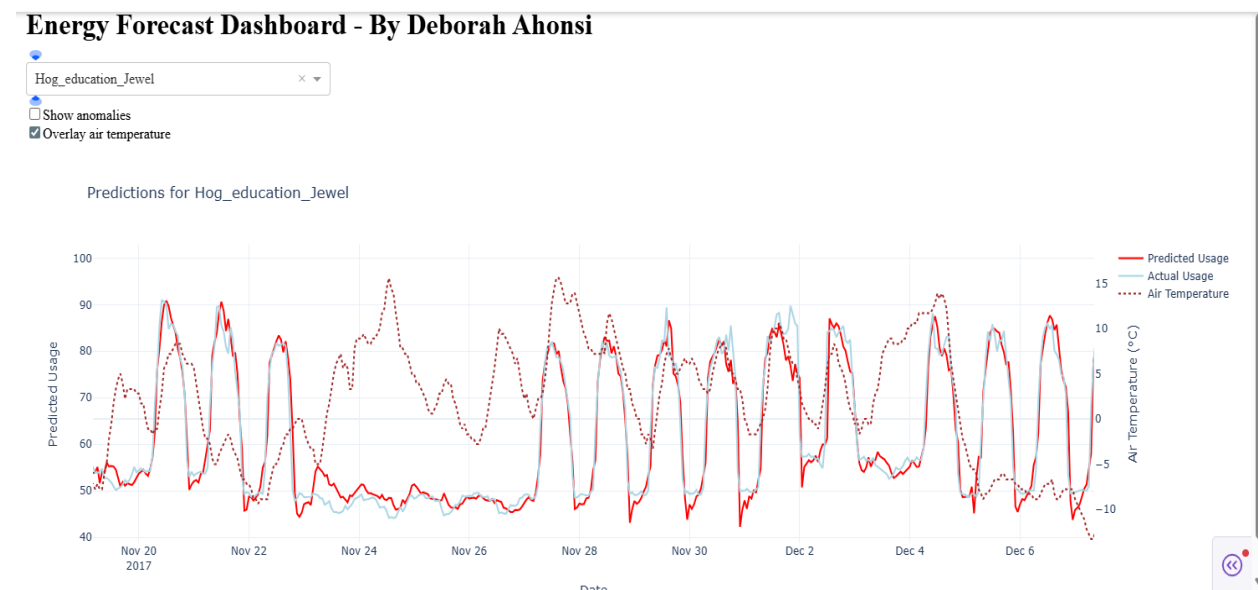


Figure 26: Chart showing weather overlay for the Hog\_education\_Jewel

### 6.3 Insights from Anomalies

Anomaly detection offered critical operational insights. Isolation Forest, with an average anomaly rate of 15%, flagged sudden spikes or drops in energy usage; some of which coincided with known holidays or extreme weather events. The Autoencoder model, which had a slightly higher anomaly rate of 18%, captured more subtle deviations and reconstruction errors, potentially identifying smaller-scale issues like sensor faults or inefficiencies.

The difference in anomaly rates suggests that Isolation Forest may be more conservative, detecting larger anomalies, while Autoencoders are more sensitive to gradual deviations. Together, these methods provide a complementary approach to identifying operational risks.

### **What Can Be Inferred from the Difference:**

**Higher Sensitivity vs. Specificity:** Autoencoders provide higher sensitivity i.e they detect a broader range of deviations but may include more false positives. Isolation Forest offers higher specificity i.e fewer detections, but more confidently identified as true anomalies.

### **Complementary Strengths:**

The difference in anomaly rates shows that the models capture different anomaly types. Isolation Forest detects more *point anomalies*, while Autoencoder can detect *contextual or sequential anomalies*.

Using both methods can provide robust multi-angle coverage of anomalies, giving stakeholders a richer understanding of unusual behavior in building energy use.

### **Operational Relevance:**

The findings suggest that 15–18% of operational days show unusual behavior. This is significant and aligns with patterns often driven by seasonal shifts, irregular occupancy, equipment faults, or data quality issues.

Facility managers can use these flagged anomalies for preventive maintenance, fault detection, or energy auditing, especially when corroborated by weather overlays or known calendar events.

## **6.4 Usefulness of Dashboard for Energy Managers**

The dashboard served as an effective bridge between data science and operational insight. By combining forecast accuracy and weather context in a single interface, it empowers data-driven decision-making for energy management. Furthermore, its modular design ensures it can be scaled to accommodate additional buildings, weather variables, or forecasting horizons.

The integrated dashboard, built using Plotly Dash, played a crucial role in translating complex forecasting and anomaly detection outputs into practical insights. Interactive building selection enabled energy managers to drill down into specific facilities, tailoring the analysis to their needs. Weather overlays enhanced interpretability by revealing how external conditions influenced energy trends. Moreover, the side-by-side visualization of actual versus predicted consumption supported model validation and facilitated the identification of potential overuse or inefficiencies.

Such a dashboard provides a practical decision-support tool, enabling continuous monitoring, forecasting, and diagnostics without requiring deep data science expertise. It aligns with current industry efforts toward smart building energy management systems (Zhao et al., 2021; IEA, 2024).

## **6.5 Challenges Encountered**

Several challenges were encountered throughout the project. First, the BDG2 dataset exhibited inconsistencies, including gaps in both weather and energy readings. Addressing these missing values required cautious imputation techniques to maintain data integrity and avoid introducing



bias. Secondly, training LSTM models for each building was computationally demanding due to the high dimensionality and long time windows involved.

Additionally, hyperparameter tuning posed difficulties, as traditional cross-validation methods were unsuitable for sequential time series data. As a result, validation-based tuning was adopted, limiting the breadth of parameter search. The autoencoder models also presented challenges; their performance was highly sensitive to input scaling and architectural configuration. Early attempts resulted in suboptimal reconstructions until appropriate batch sizes and layer depths were identified.

Lastly, the development of the interactive dashboard introduced complexity. Integrating multiple components such as forecasts, anomalies, and weather overlays into a coherent and user-friendly interface required precise coordination of plot traces, and axis alignments to ensure clarity and avoid visual clutter.

Despite these hurdles, the project achieved its core objectives of building a scalable, interpretable, and deployable forecasting system for multi-site building energy monitoring.

## **6.6 Ethical Considerations**

This study adhered to ethical standards in the use and handling of data. The BDG2 dataset used for this project is publicly available for academic research purposes and does not contain any personally identifiable information (PII) or sensitive user data. All energy consumption and weather readings are anonymized at the building level, with site identifiers used purely for modeling purposes.

Throughout the analysis, care was taken to prevent any misuse or misinterpretation of the data, particularly in the context of anomaly detection. Anomalies detected do not imply operational failure or user error but are treated as signals for further inspection by facility managers.

The developed forecasting and dashboard tools are intended for responsible energy monitoring and decision support. Any deployment of these tools should ensure that they are used to support sustainability and operational efficiency goals, and not to penalize stakeholders based on inferred anomalies without further contextual investigation.

## **6.7 Result Comparison**

The ASHRAE GEPIII competition (Miller et al., 2020) showcased that top-performing building energy forecasting models typically comprised large ensembles of gradient boosting trees especially LightGBM, with preprocessing and engineered features being the most critical differentiators (Miller et al., 2022). Error analysis revealed that approximately 79% of predictions were highly accurate, while only 4.8% produced large residuals, often at the individual-building level. While this project similarly focused on engineered temporal and weather features, our LSTM model outperformed these traditional GBM-based baselines, highlighting the advantage of sequence-aware modeling in capturing building energy dynamics.

## Conclusion and Recommendations

### 7.1 Summary of Key Findings

This research successfully developed a data-driven pipeline for forecasting and monitoring energy consumption across multiple buildings using the BDG2 dataset. Through extensive preprocessing, feature engineering, and model comparison, the Long Short-Term Memory (LSTM) neural network emerged as the most effective approach for time series prediction, outperforming ensemble models such as Random Forest, LightGBM, and XGBoost in both [Root Mean Squared Error \(RMSE\)](#) and [Mean Absolute Error \(MAE\)](#).

Key weather variables such as air temperature, dew temperature and wind speed significantly influenced energy usage, especially in weather-dependent facilities. Feature importance analysis across models confirmed the relevance of lag and rolling statistics, as well as external climatic conditions.

Two unsupervised anomaly detection methods, Isolation Forest and Autoencoder, were employed to identify unusual consumption patterns. While Isolation Forest yielded a more conservative anomaly rate (~15%), the Autoencoder demonstrated higher sensitivity (~18%) to reconstruction errors, detecting both major and subtle deviations.

Finally, all insights were integrated into a unified, interactive Plotly Dash dashboard, enabling users to explore predictions, detect anomalies, and overlay weather conditions in real-time. The dashboard proved valuable for energy managers, supporting informed decision-making for optimization and maintenance.

## 7.2 Recommendations for Future Work

Based on the findings and implementation, several recommendations are proposed for future work:

- Hybrid Models: Explore hybrid forecasting techniques, such as combining LSTM with Attention Mechanisms or Temporal Fusion Transformers (TFT), to improve accuracy and interpretability in multivariate settings ([Lim et al., 2021](#)).
- Extended Anomaly Labeling: Incorporate labeled anomaly events (e.g., maintenance logs or known outages) to validate and improve the sensitivity/specificity of anomaly detection models.
- Multi-resolution Forecasting: Implement hierarchical models that forecast energy at multiple temporal resolutions (e.g., hourly, daily, weekly), catering to different decision-making layers within facilities.
- Energy Cost Forecasting: Integrate utility tariff data and carbon pricing to extend predictions beyond kWh to monetary and environmental costs.
- Deployment and Automation: Package the pipeline for continuous deployment on cloud platforms (AWS or Azure) with automatic data ingestion, retraining, and dashboard refresh cycles.
- User Feedback Integration: Enable feedback capture within the dashboard so that facility managers can label anomalies or flag predictions, improving future model retraining.

### 7.3 Limitations and Areas for Improvement

Despite the promising outcomes, the project encountered several limitations that may affect scalability and generalizability. First, training LSTM models individually for each building proved computationally intensive, making it challenging to scale the approach across larger building portfolios without access to substantial processing resources. Additionally, the anomaly detection process was constrained by the absence of labeled ground truth data. Both Isolation Forest and Autoencoder methods operated in an unsupervised context, which limited the ability to rigorously evaluate false positives or verify true anomalies. Incorporating more advanced evaluation metrics such as Precision, Recall, and F1-Score could improve this assessment, particularly if labeled anomalies become available in future datasets (Chandola et al., 2022).

Moreover, while the methodology produced reliable results for the top-performing buildings, its transferability to lower-consumption or irregularly behaving buildings remains uncertain. Application in such cases may require further retraining or hyperparameter tuning. Another limitation lies in the deployment framework: although an interactive dashboard was developed using Plotly Dash, full-scale production deployment with real-time data ingestion and streaming predictions was not implemented. Lastly, the forecasting models relied solely on historical weather data. Integrating forward-looking weather forecasts could significantly improve the predictive accuracy for short-term energy consumption, particularly in dynamic climatic conditions.

## References

- Ahmad, T., Chen, H., Guo, Y. and Wang, J., 2021. Data-driven approaches for building energy forecasting: State-of-the-art and outlook. *Energy and Buildings*, 228, p.110491. <https://doi.org/10.1016/j.enbuild.2020.110491>
- Ahmad, T., Chen, H., Wang, J. and Guo, Y., 2020. A comprehensive overview on the data-driven and large-scale approaches for forecasting building energy demand. *Energy and Buildings*, 165, pp.301–320. <https://doi.org/10.1016/j.enbuild.2018.01.023>
- Alam, M., Noor, F., Shafiullah, G.M. and Bhuiyan, M.Z.A., 2021. Deep learning approaches for forecasting building energy consumption: A review. *IEEE Access*, 9, pp.45479–45501. <https://doi.org/10.1109/ACCESS.2021.3065949>
- Chandola, V., Banerjee, A. and Kumar, V., 2022. Anomaly detection: A survey. *ACM Computing Surveys*, 54(3), pp.1–38.
- Chen, T. and Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.785–794. <https://doi.org/10.1145/2939672.2939785>
- Chowdhury, A., Hasanuzzaman, M., Mahmud, K., Ali, M.H. and Butun, I., 2021. A comparative review of machine learning algorithms for building energy prediction. *Applied Energy*, 285, p.116435. <https://doi.org/10.1016/j.apenergy.2020.116435>

Ghosh, S., Das, S. and Basu, M., 2022. Deep learning frameworks for building load forecasting. *Energy and Buildings*, 263, p.111956. <https://doi.org/10.1016/j.enbuild.2022.111956>

International Energy Agency (IEA), 2022. *Energy Efficiency 2022*. [online] Available at: <https://www.iea.org/reports/energy-efficiency-2022>

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.Y., 2020. LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, pp.3146–3154. <https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html>

Liu, F.T., Ting, K.M. and Zhou, Z.H., 2020. Isolation Forest. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(1), pp.1–27.

Millar, C., Pang, W., Hu, L. and Arjunan, P., 2020. The Building Data Genome Project 2: Full release. *Kaggle*. [online] Available at: <https://www.kaggle.com/datasets/claytonmiller/buildingdatagenomeproject2/data>

Molnar, C., 2022. *Interpretable Machine Learning*. [online] Available at: <https://christophm.github.io/interpretable-ml-book/>

Rahman, M., Zhao, Y. and Wang, F., 2024. Visualization and dashboard development for smart building analytics. *Sustainable Cities and Society*, 96, p.104792. <https://doi.org/10.1016/j.scs.2024.104792>

Wang, X., Huang, G., Lin, Y. and Yang, S., 2022. Interactive visualization for multi-scale building energy data: A Plotly dashboard framework. *Sustainable Cities and Society*, 77, p.103551. <https://doi.org/10.1016/j.scs.2021.103551>

Wang, Z., Hong, T. and Zhu, N., 2020. Deep learning-based building energy load forecasting: A review. *Renewable and Sustainable Energy Reviews*, 137, p.110587. <https://doi.org/10.1016/j.rser.2020.110587>

Xia, Y., Chen, H., Yu, Y. and Zheng, B., 2022. A novel autoencoder-based anomaly detection method for time-series energy data. *Applied Energy*, 312, p.118738. <https://doi.org/10.1016/j.apenergy.2022.118738>

Zhao, X., Li, R., Wen, S. and Yang, F., 2021. Machine learning models for energy consumption prediction of buildings: A review. *Energy Reports*, 7, pp.1688–1699. <https://doi.org/10.1016/j.egyr.2021.02.038>

Zhou, B., Liu, H. and Wang, Y., 2022. Deep feature engineering for energy consumption prediction. *Applied Energy*, 311, p.118645. <https://doi.org/10.1016/j.apenergy.2022.118645>

Zhou, K., Yang, S. and Shao, Z., 2020. Household monthly electricity demand forecasting using LSTM: A case study. *Energy*, 202, p.117670. <https://doi.org/10.1016/j.energy.2020.117670>



## **Appendices**

Extra Tables

| 1  | timestamp  | building_id       | value              | anomaly |
|----|------------|-------------------|--------------------|---------|
| 2  | 2016-01-08 | Hog_office_Lizzie | 1694.7459999999999 | 1       |
| 3  | 2016-01-09 | Hog_office_Lizzie | 1399.722           | 1       |
| 4  | 2016-01-10 | Hog_office_Lizzie | 1562.278           | 1       |
| 5  | 2016-01-11 | Hog_office_Lizzie | 1966.104           | 1       |
| 6  | 2016-01-12 | Hog_office_Lizzie | 2085.409           | 1       |
| 7  | 2016-01-13 | Hog_office_Lizzie | 1990.045           | 1       |
| 8  | 2016-01-14 | Hog_office_Lizzie | 1805.376           | 1       |
| 9  | 2016-01-15 | Hog_office_Lizzie | 1765.881           | 1       |
| 10 | 2016-01-16 | Hog_office_Lizzie | 1560.194           | 1       |
| 11 | 2016-01-17 | Hog_office_Lizzie | 1563.901           | 1       |
| 12 | 2016-01-18 | Hog_office_Lizzie | 1378.681           | 1       |
| 13 | 2016-01-19 | Hog_office_Lizzie | 2346.366           | 1       |
| 14 | 2016-01-20 | Hog_office_Lizzie | 2311.879           | 1       |
| 15 | 2016-01-21 | Hog_office_Lizzie | 2268.369           | 1       |
| 16 | 2016-01-22 | Hog_office_Lizzie | 2171.552           | 1       |
| 17 | 2016-01-23 | Hog_office_Lizzie | 1825.2930000000001 | 1       |
| 18 | 2016-01-24 | Hog_office_Lizzie | 1865.329           | 1       |
| 19 | 2016-01-25 | Hog_office_Lizzie | 2182.095           | 1       |
| 20 | 2016-01-26 | Hog_office_Lizzie | 2238.502           | 0       |
| 21 | 2016-01-27 | Hog_office_Lizzie | 2236.565           | 0       |
| 22 | 2016-01-28 | Hog_office_Lizzie | 2240.926           | 0       |
| 23 | 2016-01-29 | Hog_office_Lizzie | 2170.981           | 0       |
| 24 | 2016-01-30 | Hog_office_Lizzie | 1949.049           | 0       |

| timestamp        | building_id       | y_true  | y_pred      |
|------------------|-------------------|---------|-------------|
| 09/08/2017 18:00 | Hog_office_Lizzie | 120.946 | 106.1876347 |
| 10/08/2017 00:00 | Hog_office_Lizzie | 111.256 | 86.58379485 |
| 10/08/2017 06:00 | Hog_office_Lizzie | 34.506  | 87.55109071 |
| 10/08/2017 18:00 | Hog_office_Lizzie | 125.612 | 105.8755202 |
| 13/08/2017 06:00 | Hog_office_Lizzie | 36.023  | 89.0905698  |
| 13/08/2017 12:00 | Hog_office_Lizzie | 117.541 | 104.882329  |
| 13/08/2017 18:00 | Hog_office_Lizzie | 115.597 | 108.2649835 |
| 14/08/2017 00:00 | Hog_office_Lizzie | 110.321 | 85.88838656 |
| 14/08/2017 06:00 | Hog_office_Lizzie | 33.962  | 88.00719668 |
| 16/08/2017 06:00 | Hog_office_Lizzie | 80.99   | 102.4958125 |
| 16/08/2017 12:00 | Hog_office_Lizzie | 120.602 | 91.44438958 |
| 16/08/2017 18:00 | Hog_office_Lizzie | 108.883 | 95.84781514 |
| 17/08/2017 00:00 | Hog_office_Lizzie | 79.7    | 85.2749969  |
| 17/08/2017 06:00 | Hog_office_Lizzie | 79.7    | 87.48232838 |
| 17/08/2017 12:00 | Hog_office_Lizzie | 124.649 | 90.69868778 |
| 18/08/2017 12:00 | Hog_office_Lizzie | 119.471 | 95.49286961 |
| 21/08/2017 18:00 | Hog_office_Lizzie | 112.961 | 96.7474756  |
| 24/08/2017 06:00 | Hog_office_Lizzie | 24.881  | 87.50125913 |
| 25/08/2017 06:00 | Hog_office_Lizzie | 25.294  | 104.2218474 |
| 25/08/2017 12:00 | Hog_office_Lizzie | 123.135 | 106.514212  |
| 25/08/2017 18:00 | Hog_office_Lizzie | 113.862 | 108.8996693 |
| 26/08/2017 00:00 | Hog_office_Lizzie | 110.844 | 83.75853814 |
| 26/08/2017 06:00 | Hog_office_Lizzie | 24.956  | 81.08285584 |
| 26/08/2017 12:00 | Hog_office_Lizzie | 105.428 | 99.87102496 |
| 26/08/2017 18:00 | Hog_office_Lizzie | 102.706 | 104.138232  |
| 27/08/2017 18:00 | Hog_office_Lizzie | 108.086 | 83.0834822  |
| 30/08/2017 18:00 | Hog_office_Lizzie | 116.195 | 78.12279024 |
| 02/09/2017 00:00 | Hog_office_Lizzie | 112.152 | 77.86767513 |
| 02/09/2017 06:00 | Hog_office_Lizzie | 28.568  | 78.91403494 |
| 04/09/2017 18:00 | Hog_office_Lizzie | 105.064 | 100.0795817 |
| 05/09/2017 18:00 | Hog_office_Lizzie | 134.089 | 90.88759705 |

|       |                  |                    |         |             |
|-------|------------------|--------------------|---------|-------------|
| 31771 | 13/12/2017 15:00 | Hog_public_Octavia | 239.601 | 246.685804  |
| 31772 | 13/12/2017 16:00 | Hog_public_Octavia | 235.28  | 240.762489  |
| 31773 | 13/12/2017 17:00 | Hog_public_Octavia | 228.427 | 228.6240551 |
| 31774 | 13/12/2017 18:00 | Hog_public_Octavia | 214.358 | 221.6049058 |
| 31775 | 13/12/2017 19:00 | Hog_public_Octavia | 202.849 | 211.9138259 |
| 31776 | 13/12/2017 20:00 | Hog_public_Octavia | 199.584 | 204.3548377 |
| 31777 | 13/12/2017 21:00 | Hog_public_Octavia | 197.14  | 206.2124239 |
| 31778 | 13/12/2017 22:00 | Hog_public_Octavia | 195.747 | 203.4030582 |
| 31779 | 13/12/2017 23:00 | Hog_public_Octavia | 189.247 | 195.1467151 |
| 31780 | 14/12/2017 00:00 | Hog_public_Octavia | 168.785 | 184.2056857 |
| 31781 | 14/12/2017 01:00 | Hog_public_Octavia | 164.48  | 169.2781177 |
| 31782 | 14/12/2017 02:00 | Hog_public_Octavia | 160.523 | 164.2172209 |
| 31783 | 14/12/2017 03:00 | Hog_public_Octavia | 156.271 | 157.1530658 |
| 31784 | 14/12/2017 04:00 | Hog_public_Octavia | 150.35  | 147.0261351 |
| 31785 | 14/12/2017 05:00 | Hog_public_Octavia | 149.748 | 150.7992672 |
| 31786 | 14/12/2017 06:00 | Hog_public_Octavia | 162.064 | 174.3703211 |
| 31787 | 14/12/2017 07:00 | Hog_public_Octavia | 178.982 | 187.8233629 |
| 31788 | 14/12/2017 08:00 | Hog_public_Octavia | 206.835 | 210.814424  |
| 31789 | 14/12/2017 09:00 | Hog_public_Octavia | 213.934 | 219.8125886 |
| 31790 | 14/12/2017 10:00 | Hog_public_Octavia | 227.235 | 222.2799838 |
| 31791 | 14/12/2017 11:00 | Hog_public_Octavia | 226.135 | 229.7915245 |
| 31792 | 14/12/2017 12:00 | Hog_public_Octavia | 233.031 | 234.6015641 |
| 31793 | 14/12/2017 13:00 | Hog_public_Octavia | 234.701 | 234.4291518 |
| 31794 | 14/12/2017 14:00 | Hog_public_Octavia | 232.455 | 239.8288877 |
| 31795 | 14/12/2017 15:00 | Hog_public_Octavia | 226.184 | 237.2291946 |
| 31796 | 14/12/2017 16:00 | Hog_public_Octavia | 229.637 | 231.9257013 |
| 31797 | 14/12/2017 17:00 | Hog_public_Octavia | 226.046 | 218.5461336 |
| 31798 | 14/12/2017 18:00 | Hog_public_Octavia | 212.284 | 213.5072522 |
| 31799 | 14/12/2017 19:00 | Hog_public_Octavia | 199.721 | 205.8518549 |
| 31800 | 14/12/2017 20:00 | Hog_public_Octavia | 198.477 | 197.2746528 |
| 31801 | 14/12/2017 21:00 | Hog_public_Octavia | 196.315 | 200.3031886 |
| 31802 | 14/12/2017 22:00 | Hog_public_Octavia | 192.987 | 201.3056898 |

|       | A                | B                     | C       | D           |
|-------|------------------|-----------------------|---------|-------------|
| 15856 | 25/09/2017 16:00 | Hog_lodging_Francisco | 238.493 | 217.9866731 |
| 15857 | 25/09/2017 17:00 | Hog_lodging_Francisco | 237.303 | 220.3103826 |
| 15858 | 25/09/2017 18:00 | Hog_lodging_Francisco | 250.697 | 259.203943  |
| 15859 | 25/09/2017 19:00 | Hog_lodging_Francisco | 256.742 | 272.2039014 |
| 15860 | 25/09/2017 20:00 | Hog_lodging_Francisco | 272.993 | 268.9055703 |
| 15861 | 25/09/2017 21:00 | Hog_lodging_Francisco | 239.661 | 251.6924969 |
| 15862 | 25/09/2017 22:00 | Hog_lodging_Francisco | 193.118 | 220.2746005 |
| 15863 | 25/09/2017 23:00 | Hog_lodging_Francisco | 188.665 | 183.224381  |
| 15864 | 26/09/2017 00:00 | Hog_lodging_Francisco | 172.175 | 169.1478339 |
| 15865 | 26/09/2017 01:00 | Hog_lodging_Francisco | 158.647 | 163.5588461 |
| 15866 | 26/09/2017 02:00 | Hog_lodging_Francisco | 145.338 | 153.0930054 |
| 15867 | 26/09/2017 03:00 | Hog_lodging_Francisco | 134.337 | 146.2929089 |
| 15868 | 26/09/2017 04:00 | Hog_lodging_Francisco | 135.549 | 140.4206872 |
| 15869 | 26/09/2017 05:00 | Hog_lodging_Francisco | 133.027 | 139.7802195 |
| 15870 | 26/09/2017 06:00 | Hog_lodging_Francisco | 143.663 | 154.8542221 |
| 15871 | 26/09/2017 07:00 | Hog_lodging_Francisco | 171.745 | 174.7865755 |
| 15872 | 26/09/2017 08:00 | Hog_lodging_Francisco | 195.51  | 195.3308027 |
| 15873 | 26/09/2017 09:00 | Hog_lodging_Francisco | 215.892 | 217.3415006 |
| 15874 | 26/09/2017 10:00 | Hog_lodging_Francisco | 217.674 | 229.6486035 |
| 15875 | 26/09/2017 11:00 | Hog_lodging_Francisco | 229.641 | 233.0778288 |
| 15876 | 26/09/2017 12:00 | Hog_lodging_Francisco | 225.44  | 232.6783547 |
| 15877 | 26/09/2017 13:00 | Hog_lodging_Francisco | 237.533 | 237.7541433 |
| 15878 | 26/09/2017 14:00 | Hog_lodging_Francisco | 234.442 | 235.711343  |
| 15879 | 26/09/2017 15:00 | Hog_lodging_Francisco | 218.247 | 220.7063975 |
| 15880 | 26/09/2017 16:00 | Hog_lodging_Francisco | 226.268 | 202.5333803 |
| 15881 | 26/09/2017 17:00 | Hog_lodging_Francisco | 233.701 | 206.2508459 |
| 15882 | 26/09/2017 18:00 | Hog_lodging_Francisco | 245.046 | 242.4592897 |
| 15883 | 26/09/2017 19:00 | Hog_lodging_Francisco | 252.701 | 257.0706323 |
| 15884 | 26/09/2017 20:00 | Hog_lodging_Francisco | 246.768 | 256.4609721 |
| 15885 | 26/09/2017 21:00 | Hog_lodging_Francisco | 213.363 | 235.0722321 |
| 15886 | 26/09/2017 22:00 | Hog_lodging_Francisco | 183.573 | 198.3462094 |
| 15887 | 26/09/2017 23:00 | Hog_lodging_Francisco | 178.854 | 170.0225196 |

- Code snippets

```

import pandas as pd

base_path = '/content/bdg2_data/'

electricity = pd.read_csv(base_path + 'electricity_cleaned.csv', parse_dates=['timestamp'])
weather = pd.read_csv(base_path + 'weather.csv', parse_dates=['timestamp'])
metadata = pd.read_csv(base_path + 'metadata.csv')

```

```

[ ] df = pd.read_csv('bdg2_data/electricity_cleaned.csv', parse_dates=['timestamp'])
df_long = df.melt(id_vars='timestamp', var_name='building_id', value_name='value')

```

```

[ ] df_long = df_long.merge(metadata[['building_id', 'site_id']], on='building_id', how='left')
print(metadata.columns)

```

```

Index(['building_id', 'site_id', 'building_id_kaggle', 'site_id_kaggle',
       'primaryspaceusage', 'sub_primaryspaceusage', 'sqm', 'sqft', 'lat',
       'lng', 'timezone', 'electricity', 'hotwater', 'chilledwater', 'steam',
       'water', 'irrigation', 'solar', 'gas', 'industry', 'subindustry',
       'heatingtype', 'yearbuilt', 'date_opened', 'numberoffloors',
       'occupants', 'energystarscore', 'eui', 'site_eui', 'source_eui',
       'leed_level', 'rating'],
      dtype='object')

```

```

[ ] print(df_long['building_id'].head())
print(metadata['building_id'].head())

```

```

0    Panther_parking_Lorriane
1    Panther_parking_Lorriane
2    Panther_parking_Lorriane
3    Panther_parking_Lorriane
4    Panther_parking_Lorriane
Name: building_id, dtype: object
0    Panther_lodging_Deane
1    Panther_lodging_Shelia
2    Panther_lodging_Ricky
3    Panther_education_Rosalie
4    Panther_education_Misty
Name: building_id, dtype: object

```

```

import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Drop NaNs
df_lstm = daily_df[features + [target]].dropna()

# Normalize
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df_lstm)

# Convert back to DataFrame
scaled_df = pd.DataFrame(scaled_data, columns=features + [target])

```

```

[ ] def create_sequences(data, sequence_length):
    X, y = [], []
    for i in range(len(data) - sequence_length):
        X.append(data[i:i+sequence_length, :-1]) # all features
        y.append(data[i+sequence_length, -1])     # target
    return np.array(X), np.array(y)

sequence_length = 14
X, y = create_sequences(scaled_df.values, sequence_length)

```

```

[ ] split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

```



```

▶ # Evaluate
rmse = np.sqrt(mean_squared_error(y_test_inv, y_pred_inv))
mae = mean_absolute_error(y_test_inv, y_pred_inv)

lstm_results.append({
    'building': bld,
    'rmse_lstm': rmse,
    'mae_lstm': mae
})

# Convert results to DataFrame
lstm_df = pd.DataFrame(lstm_results)
print(lstm_df)

```

```

⇒ /usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential model,
these arguments are not needed.
110/110 ————— 1s 5ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential model,
these arguments are not needed.
110/110 ————— 1s 5ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential model,
these arguments are not needed.
110/110 ————— 1s 5ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential model,
these arguments are not needed.
110/110 ————— 1s 5ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential model,
these arguments are not needed.
110/110 ————— 1s 5ms/step

```

|   | building              | rmse_lstm | mae_lstm  |
|---|-----------------------|-----------|-----------|
| 0 | Hog_office_Lizzie     | 8.481035  | 6.096695  |
| 1 | Hog_education_Jewel   | 4.943540  | 3.404868  |
| 2 | Hog_public_Octavia    | 11.118035 | 8.035913  |
| 3 | Hog_lodging_Francisco | 13.841307 | 9.349703  |
| 4 | Hog_assembly_Dona     | 42.237158 | 32.212707 |



```

import pandas as pd
import matplotlib.pyplot as plt

# 1. Get feature importance from Random Forest
rf_importance = pd.DataFrame({
    'feature': rf.feature_names_in_,
    'importance': rf.feature_importances_
}).sort_values(by='importance', ascending=False)

# 2. LightGBM
lgb_importance = pd.DataFrame({
    'feature': lgb.feature_name_,
    'importance': lgb.feature_importances_
}).sort_values(by='importance', ascending=False)

# 3. XGBoost
xgb_importance_dict = xgb.get_booster().get_score(importance_type='weight')
xgb_importance = pd.DataFrame({
    'feature': list(xgb_importance_dict.keys()),
    'importance': list(xgb_importance_dict.values())
}).sort_values(by='importance', ascending=False)

xgb_importance['feature'] = xgb_importance['feature'].str.replace('f', 'f_', regex=False)

# 4. Plot all together
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharey=True)

axes[0].barh(rf_importance['feature'], rf_importance['importance'], color='forestgreen')
axes[0].set_title('Random Forest')
axes[0].invert_yaxis()

axes[1].barh(lgb_importance['feature'], lgb_importance['importance'], color='darkblue')
axes[1].set_title('LightGBM')

axes[2].barh(xgb_importance['feature'], xgb_importance['importance'], color='darkred')
axes[2].set_title('XGBoost')

plt.tight_layout()
plt.show()

```

Dashboard Development

```

# Callback
@app.callback(
    Output('energy-forecast-graph', 'figure'),
    Input('building-dropdown', 'value'),
    Input('anomaly-toggle', 'value'),
    Input('temperature-toggle', 'value')
)
def update_graph(selected_building, show_anomalies, show_temp):
    bld_df = df[df['building'] == selected_building].sort_values('timestamp')

    fig = go.Figure()

    # Predicted
    fig.add_trace(go.Scatter(
        x=bld_df['timestamp'],
        y=bld_df['predicted'],
        mode='lines',
        name='Predicted Usage',
        line=dict(color='red'),
        hoverinfo='x+y'
    ))

    # Actual
    fig.add_trace(go.Scatter(
        x=bld_df['timestamp'],
        y=bld_df['actual'],
        mode='lines',
        name='Actual Usage',
        line=dict(color='lightblue'),
        hoverinfo='x+y'
    ))

    # Anomalies
    if 'anomaly' in show_anomalies:
        bld_anomalies = anomaly_df[anomaly_df['building_id'] == selected_building]
        merged = pd.merge(bld_df, bld_anomalies, how='inner', left_on='timestamp', right_on='date')
        fig.add_trace(go.Scatter(
            x=merged['timestamp'],
            y=merged['actual'],
            mode='markers',
            name='Anomalies',
            marker=dict(color='black', size=8, symbol='x'),
            hoverinfo='x+y'

```

```

    ))

    # Temperature overlay
    if 'temp' in show_temp:
        fig.add_trace(go.Scatter(
            x=bld_df['timestamp'],
            y=bld_df['airTemperature'],
            mode='lines',
            name='Air Temperature',
            line=dict(color='brown', dash='dot'),
            yaxis='y2'
        ))
        fig.update_layout(
            yaxis2=dict(
                title='Air Temperature (°C)',
                overlaying='y',
                side='right',
                showgrid=False
            )
        )

    # Layout styling
    fig.update_layout(
        title=f'Predictions for {selected_building}',
        xaxis_title='Date',
        yaxis_title='Predicted Usage',
        template='plotly_white',
        hovermode='x unified',
        height=500
    )

    return fig

# Run server
if __name__ == '__main__':
    app.run(debug=True)

```