UNIVERSITY OF ENGINEERING AND TECHNOLOGY PESHAWAR, JALOZAI CAMPU

LAB REPORT OF MACHINE LEARNING

Open Ended Lab title: Design a model to Forecast in the CityLearn challenge.

Submitted By: Afnan Ahmad

Reg # 21JZELE0437

Submitted To:

Engr. Irshad Ullah

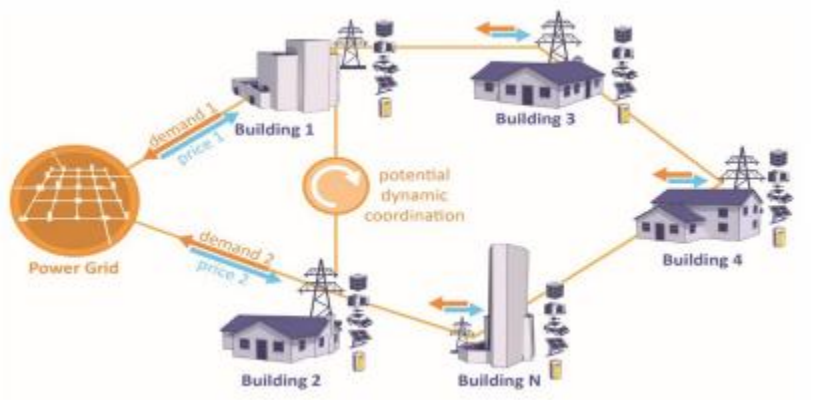
GitHub Repository:

https://github.com/AfnanAhmad82/Machine-learning-labs/tree/main/Open%20ended-Lab

DATE OF SUBMISSION: JUNE, 2025

I N T R O D U C T I O N

The global transition toward smarter, more sustainable energy systems has amplified the importance of accurate load forecasting in electrical grids. As electricity demand becomes increasingly dynamicâdriven by the integration of renewable energy sources, electric vehicles, and decentralized energy productionâutilities are under greater pressure to ensure efficient, stable, and responsive energy distribution. A critical component in this complex landscape is the ability to predict nonshiftable electrical loadsâthose essential and inflexible demands that cannot be deferred or rescheduled without compromising user needs or safety.



Non-shiftable loadsâsuch as lighting, heating, refrigeration, and life-support systemsârepresent a foundational portion of daily energy consumption. These loads are consistent and time-bound, making their accurate prediction vital for real-time supply-demand balancing, peak load mitigation, and optimal allocation of energy resources. Traditional forecasting methods, often based on statistical techniques or historical averaging, typically fall short in capturing the complex, nonlinear relationships inherent in modern consumption patternsâespecially when influenced by external factors such as weather, occupant behavior, and pricing signals. With the advancement of machine learning (ML) and deep learning, there is now a transformative opportunity to improve the accuracy and flexibility of load forecasting. ML models, particularly those suited for time-series forecasting, can learn hidden patterns in sequential data and incorporate diverse contextual features to make data-driven, future-aware predictions. Architectures like Long Short-Term Memory (LSTM) networks have shown promising results due to their ability to capture long-range temporal dependenciesâmaking them especially useful in environments where energy demand changes dynamically over time. This report explores a machine learning-based approach to forecasting non-shiftable load within the framework of the CityLearn environmentâan open-source reinforcement learning platform designed for simulating energy coordination and demand response in smart urban districts. While CityLearn primarily focuses on controlling shiftable loads using reinforcement learning agents, a crucial parallel task is the accurate forecasting of non-shiftable loads, which directly impacts how effectively these agents can perform. By designing a dedicated ML model to forecast non-shiftable consumption in the CityLearn Challenge, this work addresses a core need in smart energy systems: ensuring essential demand is met reliably and efficiently. The challenge not only requires precise model selection, but also emphasizes preprocessing, feature engineering, and guarding against data

leakage. In doing so, it reflects the broader industry shift toward data-driven energy management, contributing to the global goal of building resilient, low-carbon, and intelligent power infrastructures.

O B J E C T I V E S:

The main goal of this project is to build a machine learning model that can accurately predict the non-shiftable electrical load in smart grid buildings. We use the CityLearn Challenge data, which includes time-based features, weather, electricity prices, and carbon intensity. The model helps us understand and forecast the electricity demand that cannot be delayed or shifted. The specific goals of this project are:

To design and develop a forecasting model for the CityLearn Challenge to predict non-shiftable load in buildings.

To understand what non-shiftable load means and why it is important in smart energy systems.

To explore and clean the CityLearn dataset and choose useful features related to non-shiftable energy usage.

To create new features (like time of day, weather, prices) that help improve prediction accuracy.

To build and train a deep learning model to predict non-shiftable load over time.

To check the model's accuracy using standard error metrics such as MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and MAPE (Mean Absolute Percentage Error).

To study how input data like predicted weather or prices affect the modelâs performance.

To find possible issues like data leakage or bad input data and suggest simple ways to reduce their effect. These steps support the bigger aim of using artificial intelligence in smart grids to make energy use more efficient and prepare cities for future energy needs

D ATA O V E RV I E W:

The data used in this project is sourced from the CityLearn Challenge 2022 repository (Phase-All dataset) and consists of four distinct datasets: Building 2, Carbon Intensity, Pricing, and Weather. These datasets collectively provide a rich set of temporal, environmental, economic, and operational features essential for accurate forecasting of non-shiftable electrical load in smart buildings.

building 2 dataset

carbon intensity dataset

pricing dataset

weather dataset

All datasets are time-synchronized to ensure consistency across features and enable the development of robust forecasting models. Preprocessing steps such as missing data imputation, normalization, and temporal feature engineering were applied to enhance model performance. The integration of these datasets allows the model to capture complex interactions between building operations, environmental conditions, economic factors, and carbon emissions, facilitating more accurate and reliable forecasting of non-shiftable electrical load in urban smart grids

D ATA P R E P R O C E S S I N G:

This section outlines the preprocessing steps applied to prepare the raw datasets for model development. These steps ensured data consistency, minimized noise, and enhanced the predictive potential of input features. The process began by loading and merging the four key datasets â Building 2, Weather, Carbon Intensity, and Pricing â using a common time index. This integration created a unified dataset encompassing temporal, environmental, and economic variables. To reduce redundancy and focus on relevant predictors, the following non-essential columns were removed: indoor_dry_bulb_temperature

average_unmet_cooling_setpoint_difference

indoor_relative_humidity

Subsequently, outliers were detected and handled across key numerical features to ensure robust model performance. The columns were reordered to logically group features, enhancing readability and consistency during modeling. Missing values were identified and addressed using appropriate imputation strategies to preserve data quality. To capture the cyclical nature of temporal features such as hour and month, cyclical encoding was applied using sine and cosine transformations. This enabled the model to better understand periodic patterns in the data. A correlation matrix was computed to analyze inter-feature relationships and support feature selection by identifying strongly correlated or irrelevant variables. The final dataset was partitioned into three sets using a 70:20:10 split ratio:

Training Set: (876, 26)

Validation Set: (1752, 26)

Test Set: (876, 26)

Datetime components were extracted where applicable to enhance temporal context. For feature scaling, two normalization techniques were applied:

StandardScaler was used for weather-related continuous variables.

MinMaxScaler was applied to economic and target variables.

The scaled data was then saved as three separate sets â training, validation, and testing â for use during model training and evaluation. After model predictions, the target variable non_shiftable_load was inverse transformed to return results to their original scale, ensuring meaningful interpretation

M O D E L A R C H I T E C T U R E:

This section outlines the architecture and implementation of the deep learning model developed for forecasting non-shiftable load using the CityLearn dataset. The modeling process begins by loading the preprocessed training, validation, and test datasets, each containing normalized values. Using a sliding window approach, the time series data is reshaped into sequences of 24 time steps to predict the next step (1-step ahead). This captures temporal patterns crucial for accurate load forecasting.

data preparation and sequencing:

The datasets are loaded from saved CSV files containing normalized data. A sliding window method is used to generate sequences of length 24 time steps, which are used as model inputs. The target for prediction is the immediate next step following each sequence, enabling the model to learn temporal dependencies.

```
import os
path_dataset =r'G:\Afnan_0437_OpenEnded\Dataset'
path_tr = os.path.join(path_dataset, 'CityLearn_train.csv')
df_tr = pd.read_csv(path_tr)
train_set = df_tr.iloc[:].values
path_v = os.path.join(path_dataset, 'CityLearn_validation.csv')
df_v = pd.read_csv(path_v)
validation_set = df_v.iloc[:].values
path_te = os.path.join(path_dataset, 'Citylearn_test.csv')
df_te = pd.read_csv(path_te)
test_set = df_te.iloc[:].values

path_scaler = os.path.join(path_dataset, 'Scaler.pkl')
scaler         = pickle.load(open(path_scaler, 'rb'))

train_set.shape, validation_set.shape, test_set.shape
```

The model architecture is based on a RNN network designed to handle sequential time-series data efficiently.

Model Code:

The following Python code defines the RNN-based neural network architecture used for forecasting non-shiftable load.

```python
class RNNModel(nn.Module):
    def __init__(self, num_features=29):
        super(RNNModel, self).__init__()

        self.rnn1 = nn.RNN(input_size=num_features, hidden_size=8, batch_first=True)
        self.rnn2 = nn.RNN(input_size=8, hidden_size=20, batch_first=True)

        self.flatten = nn.Flatten()
        self.output_layer = nn.Linear(20, 1)

    def forward(self, x):
        # x shape: (batch_size, time_steps, num_features)
        x, _ = self.rnn1(x)
        x, _ = self.rnn2(x)

        # Take the output at the last time step (like Keras default behavior)
        x = x[:, -1, :]  # shape: (batch_size, 20)

        x = self.flatten(x)
        output = self.output_layer(x)

        return output
```

training setup:

Loss Function and Optimization:

The model is trained using Mean Squared Error (MSE) loss function optimized with the Adam optimizer. A learning rate scheduler reduces the learning rate if the validation loss plateaus to fine-tune training.

Training Configuration:

Component	Setting / Details
Loss Function	Mean Squared Error (MSE)
Optimizer	Adam
Learning Rate	0.001
Scheduler	ReduceLROnPlateau (factor=0.5, patience=10)
Forecasting Type	Univariate, 1-step ahead
Input Shape	(batch size, 24, 29)
Target Variable	Non-shiftable load

This architecture effectively captures temporal dependencies in energy consumption data, while techniques such as dropout and layer normalization improve model robustness and generalization.

Training Procedure

The training process was initialized with a starting epoch of 1 and set to run for 50 epochs. The model was trained using mini-batches with a batch size of 32 to efficiently manage memory and improve training speed. Data loaders were created for the training, validation, and test datasets to facilitate batch-wise data feeding into the model. Mean Squared Error (MSE) loss was used as the primary criterion for optimization, while Mean Absolute Error (MAE) was also prepared for evaluation. A utility was implemented to save the best model based on validation performance during training. Additionally, training and validation loss curves were plotted to monitor the modelâs learning progress. The total training time was recorded to understand the computational cost

R E S U LT S:

To evaluate the model performance, we used the Mean Squared Error (MSE) as the primary metric. MSE calculates the average of the squares of the errors between the predicted and actual values. It gives more weight to larger errors, making it a reliable measure for assessing forecasting accuracy in time-series models.

RNN (Recurrent Neural Network): A simpler sequential model without gating mechanisms.

The models were trained for multiple epochs, and the validation MSE was tracked at each step

MSE(RNN)	0.027

F U T U R E W O R K A N D L I M I TAT I O N S:

Future work:

There are several avenues to extend and improve the current model. First, integrating attention mechanisms such as self-attention or Transformer based encoders may further enhance temporal dependency modeling, especially for long sequences. Additionally, exploring ensemble models or hybrid architectures that combine CNNs or GRUs with LSTM can be promising for improving forecasting accuracy. Future work may also involve deploying the trained model in a real-time environment using IoT-based systems. This would enable dynamic load forecasting and timely energy distribution in smart grids. Moreover, using more granular datasets (hourly or 15-min intervals) and additional external factors like temperature, weather, and socio-economic variables could significantly improve prediction performance.

Limitations:

Despite the promising results, the proposed model has certain limitations. Firstly, the model performance heavily depends on the quality and quantity of input features; insufficient or noisy data may lead to poor generalization. Secondly, although the LSTM model captures sequential dependencies well, it still struggles with long-range dependencies when compared to attention-based models. Another limitation is computational cost. The training time on standard hardware (CPU/GPU) is relatively high, and hyperparameter tuning further increases the training duration.

Lastly, the model was trained and evaluated on historical data only, without incorporating real-time changes in grid behavior or unexpected disruptions, which might limit its robustness in production environments.

C O N C L U S I O N:

This project focused on forecasting non-shiftable load using machine learning techniques applied to the CityLearn dataset. A deep learningbased approach was developed with a multi-layer LSTM architecture, enhanced by Layer Normalization and Xavier initialization, to effectively capture the temporal dependencies inherent in non-shiftable energy consumption patterns. The dataset was carefully preprocessed, and the model was trained over 50 epochs using MSE