Predicting Energy Prosumer Behaviour

Juliet Kern, Kevin Meng
20832291, 20850348

Abstract

This report investigates the prediction of energy behaviour in prosumers, individuals that produce and consume energy, with the objective of reducing energy imbalance costs. Various models including Linear Regression, eXtreme Gradient Boosting (XGBoost) Regression, Multi-Layer Perceptron (MLP), and CNNs are explored and assessed based on their performance. For the various models, and depending on the model characteristics, parameters were tuned to achieve the optimal performance. Our analysis reveals XGBoost Regression as the most effective model, based on its ability to handle complex and large datasets. This report is used for educational purposes, to learn about various machine learning models, and to better understand how complex sets of data can be used in machine learning to make the most effective predictions.

1 Introduction

The goal of this report is to create an energy prediction model of prosumers to reduce energy imbalance costs. In the context of energy, prosumers are individuals that both produce and consume energy. Energy prosumers have been important in facilitating the energy transition towards more sustainable types of production, such as integrating renewable energy sources into communities for individual or shared use [1]. The process of energy prosumption includes citizen participation in the energy market and leads to a remodeling of future energy features [2]. The growing number of prosumers presents a significant challenge in terms of rising energy imbalance and associated costs. Successfully integrating prosumers into the energy market necessitates complex processes, including testing the reliability of the energy supply network and evaluating the impact of energy production on costs [2]. Neglecting these issues could lead to increased operational costs, potential grid instability, and inefficient use of energy resources. Therefore, analyzing the energy imbalance of prosumers is a critical issue. By addressing this issue, the associated costs can be significantly reduced, facilitating a more sustainable integration of prosumers into the energy system. Furthermore, it can provide better insights in how to manage prosumers. Thus, encouraging more consumers to adopt the prosumer model and overall promoting renewable energy use and production [3]. Such advancements are key to evolving energy markets and ensuring long-term stability and efficiency.

This report discusses and compares different models in order to predict energy prosumer behaviour. In this analysis, the performance of Linear Regression, eXtreme Gradient Boosting (XGBoost) Regression, Multi-layer Perceptrons (MLP), and Convolutional Neural Networks (CNNs) are explored. In addition, varying model hyperparameters in order to optimize model performance. The inspiration and data used for this paper was retrieved from Kaggle Competition Enefit - Predict Energy Behavior of Prosumers [3].

2 Task and Data

The purpose of this task was to predict the amount of electricity produced and consumed by Estonian energy customers who have solar panels installed. Included in the dataset is weather data, the relevant energy prices, and records of the installed photovoltaic capacity [3].

2.1 Data Preprocessing

To process the data, the information was consolidated into one dataset. This was done in order for the models to determine relationships between all the separate datasets and information provided. All non-numerical values were encoded into numerical values in order to be processed by the models. For example, the date-time data had to be converted into separate numerical representations by day, month, year and hour in order to be able to use it within the models. Additional time-based features were created to help capture temporal dynamics in the data, especially for models which do not process the data as sequences, such as the linear regression and the XGBoost model. Lagged features of the target for each unique 'prediction_unit_id' value were created such that previous consumption and production values would have an influence on the current target prediction. This was done by creating additional features which represented the energy consumption or production of the client at the same time of day 2-15 days ago.

For models that require a time-step input, like CNNs, additional pre-processing needed to be done to effective capture the sequential relationships. By default, the data provided was sorted by 'datetime'. Creating sequences from the data in this format would be problematic due to subsequent rows representing the data for different clients, at the same 'datetime' value. This was accomplished by sorting the data first by 'prediction_unit_id', and then by 'datetime', ensuring that each subsequent row of data corresponded to the next hour's data for the same client. Sequences were 48 rows long (representing 24 hours) and taken with a stride of 3 rows in between consecutive sequences. The function for creating sequences also ensured that the data captured within each sequence only corresponded to a single 'prediction_unit_id' such that the sequence would not capture the end of one client's data, and the beginning of the next client's data.

Finally, due to the engineered lagging features, as well as inconsistencies in the weather data, there were several NaN values present in the data. Depending on the model used and its inability to handle missing data (Linear Regression, XGBoost Regression, CNN), the NaN values were dropped before processing.

3 Loss Function

The metric chosen to measure the performance of algorithms was Mean Absolute Error (MAE). Due to its lower sensitivity to outliers compared to Mean Squared Error (MSE), MAE is the preferred metric to use for the task of energy consumption/production prediction, where unexpected circumstances leading to significant outliers are common and unpredictable outcomes are inevitable (such as sudden, unexpected changes in weather).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i|$$

4 Models

4.1 Linear Regression

Linear regression is a simple model that predicts through learned linear relationships between the features and the target. It is easy to implement and takes little computational resources to train, serving as a baseline model for our project. However, linear regression works most effectively when there is a linear relationship between these variables. In the context of energy prediction, the relationship between variables such as time of day and temperature could likely be linearly related to

energy consumption, and weather conditions such as the amount of sunlight are likely to be linearly related to energy production through solar panels.

However, the complexity and likely non-linear nature of energy behaviour patterns results in predictions using linear regression models that are not optimal. The presence of outliers, multi-collinearity among features, or complex interactions that cannot be captured by a linear model may lead to less accurate predictions. This can be observed in Figure 1, where the data points do not all fall on the diagonal line, which would indicate perfect predictions. Instead, they are dispersed, which shows variance from the ideal linear relationship. This model yielded an validation and test MAE of approximately 91.14 and 130.0 respectively.

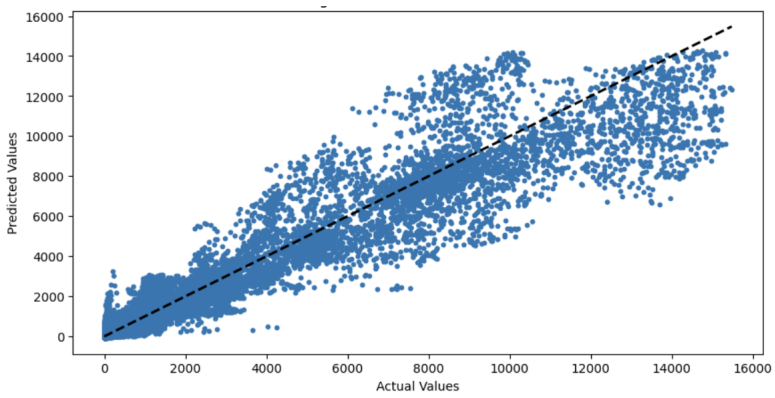

Figure 1: Linear Regression Predicted Values vs Actual Values.

4.2 XGBoost Regression

The XGBoost regression model combines predictions from various decision trees to predict a continuous target variable. This model was chosen due to the complexity and characteristics of the dataset used. XGBoost is designed to be able to handle large datasets, missing values, and potential collinearity among features, which were all issues presented within the dataset.

Table 1: XGBoost Regression Parameter Exploration

Parameters Changed	Value	Validation MAE	Computation Speed (seconds)
Learning rate (learning_rate)	1	72.86	14.94
	0.1	52.14	42.97
	0.01	61.36	43.43
	0.001	210.74	51.53
Number of Trees (n_estimators)	10	211.13	4.69
	50	66.15	6.46
	100	60.96	8.41
	500	53.51	23.56
	1000	52.14	42.74
	1500	51.94	52.40
Regularization (reg_lambda)	0.1	54.25	43.01
	1	52.14	42.74
	2	52.9	42.77
Early Stopping (early_stopping_rounds)	100	50.49	52.40

The initial XGBoost regression model had a learning rate of 0.1, 1000 trees and no regularization techniques implemented. After training for 1000 iterations, this model reported a final train MAE of

45.94 and validation MAE of 52.14. The difference between the test and training suggest overfitting (training MAE - validation MAE = 6.20), so the hyperparameters were adjusted to appropriately regularize the model. The hyperparameter exploration aimed to maintain a low MAE near or below the initial value of 52.14, and to decrease the difference between the train and validation MAE to demonstrate reduced overfitting. The hyperparameter exploration values can be seen in Table 1.

In the parameter exploration, the learning rate was modified to values above and below the initial value of 0.1 to inspect how the performance of the model was impacted. If the learning rate is appropriately set the model can quickly converge towards a lower loss, making significant progress after a reasonable number of iterations. At a learning rate too high, the model may overshoot the optimal values leading to divergence or erratic behavior. At a learning rate too low, the training may progress slowly, requiring more iterations to reach convergence. At a learning rate of 0.01 or smaller, the model was seen to have a more steady but slow progress of decreasing validation MAE. After 1000 iterations the validation MAE for a learning rate of 0.01 and 0.001, was 61.36 and 210.74 respectively, with smaller increments of loss as iterations increased. Eventually, for both learning rates, the models will likely achieve lower losses but at the cost of computational time, requiring many more iterations to reach the desired accuracy. A learning rate of 1 was too high, as the model yielded an increase in validation MAE to 72.86. The optimal learning rate chosen was 0.1 after 1000 iterations yielding a train and validation MAE of 45.94 and 52.14 respectively.

In the interest of conserving computation time, the number of trees were modified below the initial amount of 1000, from 10 to 500 trees. At a minimal amount of 10 trees there was a significant rise in MAE to 211.13, however achieved a much quicker result only taking 4.69 seconds to compile. At 50, 100 and 500 trees, there was a steady decline in validation MAE from 66.15, 60.96 and 53.51 respectively with a steeper incline in computation speed of 6.46, 8.41 and 23.56 seconds respectively. In the interest of achieving minimum MAE, the number of trees were modified above the initial amount of 1000 to 1500, however with 500 more trees there was only a 0.20 decrease in MAE and approximately 9.60 increase in computation time. Furthermore, with an increased number of trees, this model was increasingly complex and demonstrated signs of overfitting yielding a train and validation MAE of 42.59 and 51.94 respectively. The optimal number of trees of 1500 was selected, even at the higher computational cost due to the goal of having the MAE reduced.

To improve signs of overfitting, early stopping (early_stopping_rounds = 100) was used and was found to be most effective, by selecting the lowest validation MAE achieved while fitting the data to the model. Introducing early stopping resulted in a validation MAE of 50.60 with the same optimal parameters found previously (1500 trees, 0.1 learning rate). This allowed the model to select the validation MAE value that was most optimal and stopped it from increasing. In an attempt to further reduce overfitting, the lambda value for L2 regularization was modified. Values above and below the default value of 1.0 demonstrated no improvement in the MAE or computation speed, so therefore were not changed. This could be as a result of redundant effects of early stopping and additional regularization techniques. After exploration of regularization techniques, the methods that yielded the lowest train and validation MAE of 44.90 and 50.49 respectively, and had minimal signs of overfitting were chosen: early stopping.

The final model had a learning rate of 0.1, 1500 trees and early stopping. This model yielded a validation and test MAE of 50.49 and 50.60 respectively. This can be observed in Figure 2, where the data points fall close to the diagonal line, representing perfect predictions.

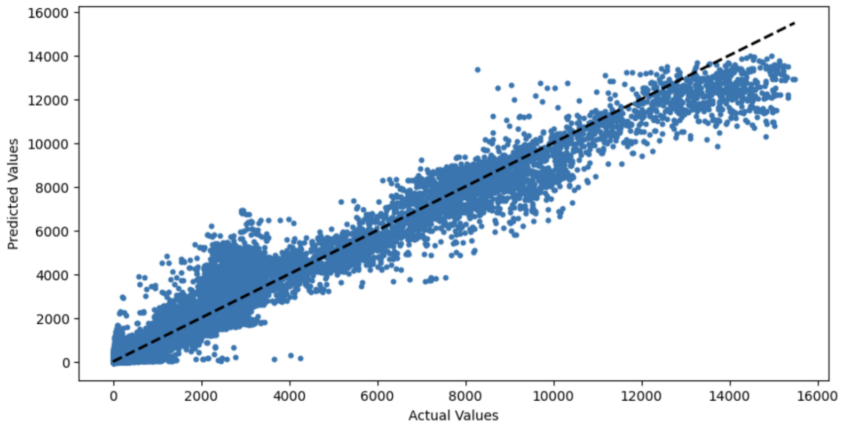

Figure 2: XGBoost Regression Predicted Values vs Actual Values.

4.3 Multi-Layer Perceptron

The MLP was determined to be a suitable model to try for this energy prediction task as well. It has a simple neural network architecture and is a basic form of deep learning that is well-balanced for both performance and complexity. The MLP is also a highly adaptable model - it can be applied to a wide range of tasks with varying complexities due to its flexible architecture. A variety of architectures were tested to find the optimal balance between the model's performance and complexity, and because the exact complexity of the energy consumption/production task is unknown, a wide range of parameters were tested. The MLP architecture which was initially implemented on the dataset consisted of a single hidden layer with ReLU non-linearity, and Adam optimizer. The first variations tested were the effect of the number of hidden neurons in the single hidden layer.

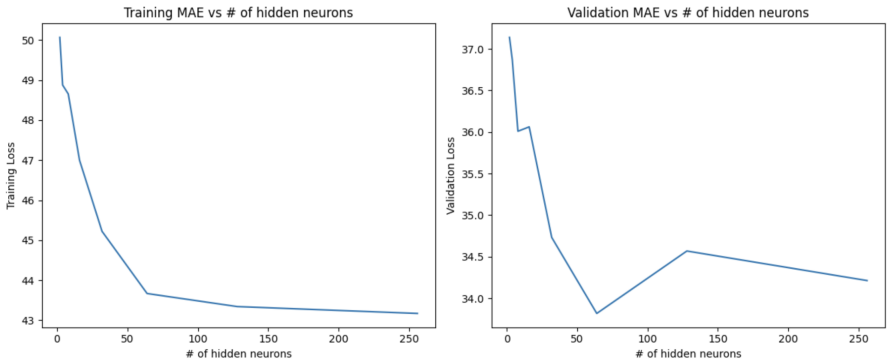

Figure 3: MLP Number of Hidden Neurons vs MAE for Training and Validation Data.

The optimal number of hidden neurons was determined to be 64, evident by the low train and validation MAE. As seen in Figure 3, the increase in validation loss despite the continuing decrease in training loss as the number of hidden neurons increases above 64 indicates overfitting. Subsequent trials aimed to vary the depth of the MLP network and the number of neurons in each layer: Model 1 had two layers with 64 and 32 neurons, Model 2 had two layers of 64 and 64 neurons, Model 3 had three layers of 64, 32, and 32 neurons, and Model 4 had three layers of 64, 64, and 32 neurons. Model 4 achieved the best performance, with train and validation losses of 38.50, and 32.64 respectively. This can be observed in Figure 4, demonstrating the proximity of the predicted values to the actual values.

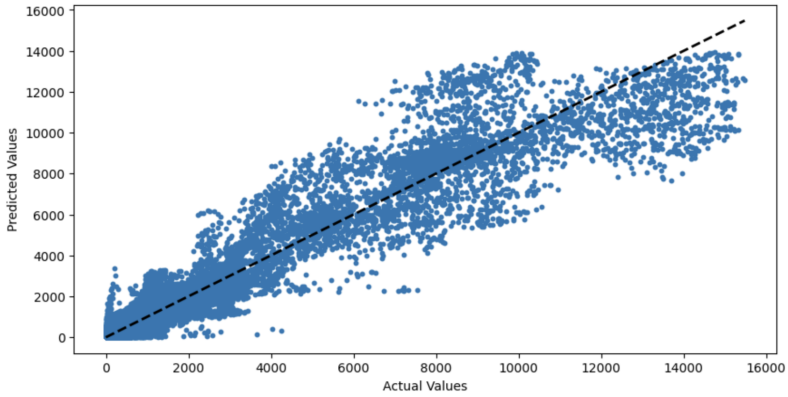

Figure 4: MLP Predicted Values vs Actual Values.

4.4 Convolutional Neural Network

Convolutional neural networks are also a model which is capable of predicting non-linear, time-dependent trends in data when specifically designed for such tasks. Commonly applied to image-related tasks, CNNs can be applied to numerical data as well to detect similar pattern hierarchies. The time-series sequences of data unique to each client were used as the input to the CNN model.

The baseline CNN model consisted of three 1D convolutional layers with 32, 64, and 64 hidden units, each with a stride of 1 with no padding. Input data was processed through the first convolutional layer, followed by a ReLU activation function, and then downsampled with a 1D max pooling layer with a size of 2 and a stride of 2. The resulting feature maps were fed into the second convolutional layer, followed by ReLU activation and identical max pooling. The third layer output was subsequently put through another ReLU, followed by a Dense layer with 64 units, another ReLU activation, and finally through a Dense layer with 1 unit. After training for 10 epochs, this model reported a training and validation loss of 101.16 and 140.80 respectively.

First, the number of hidden units in each layer was modified to evaluate its impact on model performance. More complicated features are detected deeper into the network, and therefore more filters are required for the model to express more complex combinations of the features in the earlier layers. At the same time, too many hidden units in the first two layers can cause the model to learn overly complex features too early, leading to overfitting. The number of hidden units was doubled in each layer to 64, 128, and 128, however, these changes yielded worse train and validation losses of 115.08 and 205.22 respectively. An additional convolutional layer was then added to the previous architecture, yielding a model with 4 layers with 64, 128, 128, and 128 hidden units respectively. Each subsequent layer in a convolutional neural network results in more complex patterns detected. As evident from the increased train loss and validation losses of 117.63 and 186.70 (when compared to the baseline CNN model), the more complex features were not needed for the prediction of energy consumption/production.

The next parameter varied was the kernel size - the kernel sizes in the baseline model were increased to 5 in the first and second layers. Increasing the kernel size increases the receptive region of each convolution, helping to capture extended patterns and longer-term trends in the time-series data. However, in doing so, the filters can also negatively impact the effectiveness of finer features. The train and validation loss was negatively affected by the increase in kernel sizes, and the model overfit to the training data significantly, evident from the MAE losses of 109.23 and 315.66 respectively, and therefore the original kernel sizes of 3 were kept. Since the CNN model complexity has been shown to have an inverse relationship with performance in this specific task, the number of convolutional layers was decreased from 3 to 2. A simpler model could yield better performances due to better generalization, less overfitting, and better parameter efficiency. The simpler model architecture outperformed the baseline model with train and validation losses of 90.13 and 101.97, showing that

energy consumption and production patterns are not very complex. The prediction of this model compared the the actual targets of the validation set is shown in Figure 5.

To reduce overfitting, dropout was introduced into the previous model, placed after the second convolutional layer. Dropout values of 0.5, 0.3, and 0.1 were tested, with the best loss from these trials being achieved with a dropout of 0.1, at 115.79 and 214.05 for train and validation sets. Despite dropout's typical regularizing effect, applying dropout on a very simple model such as this one has the potential to reduce the model's capacity such that more epochs are required to learn the necessary patterns for a more accurate prediction. Finally, the Adam optimizer's beta 1 and beta 2 parameters were varied. The beta 1 parameter was decreased to 0.85 from the default of 0.9, reducing the momentum effect and having the optimization be more responsive to recent gradients. The beta 2 parameter was decreased to 0.995 from 0.999 for a more aggressive scaling of the learning rate, which could speed up convergence. These, however, yielded train and validation losses of 102.95 and 168.28 which were worse than the baseline performance.

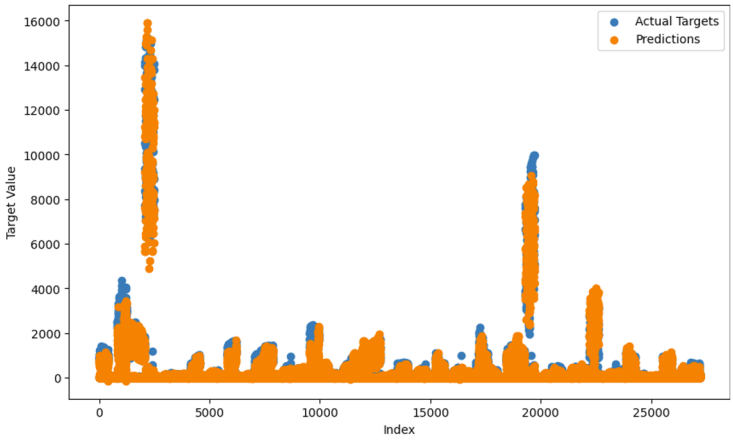

Figure 5: CNN Predicted Values vs Actual Values.

5 Discussion

5.1 Model Comparison

Table 2: MAE Comparison Between Models

Model	Validation MAE	Test MAE
Linear Regression	91.14	130.0
XGBoost Regression	50.49	50.60
MLP	32.64	72.60
CNN	101.97	92.44

The test MAE of the four machine learning models: Linear Regression, XGBoost regression, MLP and CNN in predicting the energy behaviour of prosumers were compared. The results, as seen in Table 2, show that the test MAE for all models were relatively low, with XGBoost Regression performing the best. The XGBoost Regression model achieved a test MAE of 50.60 due to its ability to capture intricate patterns in large or complex datasets. XGBoost's ability to manage missing values and its robustness to overfitting also contribute to its optimal performance. The MLP demonstrated good performance but was not as effective as XGBoost and yielded a test MAE of 72.60. This could be due to its architecture being more suited to capturing non-linear relationships, however has some limitations when detailing with large datasets. The CNN model yielded the third best test MAE result of 92.44. CNNs are traditionally used for spatial data like images and may not be ideally

suited for time series or tabular data. While CNNs can be adapted for time series data by treating time steps as spatial dimensions, they might not be as effective as models specifically designed for time-series analysis or complex regression tasks. Linear Regression performed the worst in this comparison, yielding a final test MAE of 130.0. This is expected as linear regression models are limited in handling complex, non-linear relationships within data. The dataset used in this analysis contains nonlinear interactions, and can be influenced by multiple variables, therefore may not be adequately modeled by a linear approach. This limitation leads to a higher MAE, indicating poorer performance in capturing the underlying patterns of the data.

5.2 Train, Validation and Test Split

The dataset was split into a train, validation, and test with a 70-15-15 split. With 638 days of data collected, this split allowed the validation and test sets, consisting of 96 days of data each, to be large enough to be representative of the energy production trends. Due to the time-series nature of the data, it was split without being shuffled to prevent future leakage which could lead to better, unrealistic results when future data is used to predict the past. As a result, the train set consisted of the first 70% of the time over which the data was collected, followed by the validation set consisting of the next 15%, and the test set for the final 15%.

However, splitting the dataset in this manner can lead to discrepancies between the training data compared to the test and validation data. The dataset contains data from August 2021 to May 2023. Therefore the majority of the training data would be in the first two years, and the validation and test falling in the latter year. Over time, the underlying distribution of data can shift. This can cause models trained on historical data to become less accurate as they are applied to more recent data. If the training data does not capture these shifts, the model's predictions may become less reliable over time. Furthermore, having the validation and test sets consisting of 96 days of data results in the limited representation of weather conditions and seasons - two factors that are crucial in determining energy consumption and production patterns. This is a limitation of the study.

6 Conclusion

We attempted to predict energy consumption and production targets with the goal of reducing energy imbalance costs. The performance of Linear regression, XGBoost regression, Multi-Layer Perceptrons, and CNN algorithms were compared based on their performance on the energy prediction task, their complexity, and their computational requirements for training and inference. The dataset consisted of a variety of features corresponding to gas prices, electricity prices, forecast weather, historical weather, and other information regarding the location, time, and photovoltaic capacity of the prosumer. Additional features were engineered, such as lagging time-related features to help capture additional information, and additional data processing was done to create time-series sequences as the input to the CNN. The dataset was split in a 70-15-15 split chronologically to prevent data leakage. The loss function was chosen to be Mean Absolute Error, suitable for energy prediction due to the frequent and unpredictable outliers in energy prediction data, where alternative such as Mean Squared Error loss could penalize too much. The best-performing model based on test MAE was the XGBoost (50.60), followed by MLP (72.60), CNN (92.44), and linear regression (130.0). Through parameter variations, it was shown that complexity did not always correlate to better performance and that the best-performing models and parameters are those that are best suited to the data. Despite a complex architecture, the CNN was outperformed by simpler models such as the XGBoost and MLP, whose architectures are better suited to capturing the patterns for energy prediction. Future work could explore alternative models that are capable of processing sequences, such as transformers, and Recurrent Neural Networks with Long Short-Term Memory and Gated Recurrent Units to capture long-term dependencies and explore the trade-off between the increased complexity of these models and performance. Further exploration could be done into decision trees due to the proven performance of the XGBoost model for this data, such as Light Gradient Boosting Machine (LightGBM), and random forests.

7 References

[1] Parra-Domínguez, E. Sánchez, and Á. Ordóñez, "The Prosumer: A systematic review of the new paradigm in Energy and Sustainable Development," Sustainability, vol. 15, no. 13, p. 10552, 2023. doi:10.3390/su151310552

[2] J.-L. Hu and M.-Y. Chuang, "The importance of energy prosumers for affordable and Clean Energy Development: A review of the literature from the viewpoints of management and policy," Energies, vol. 16, no. 17, p. 6270, 2023. doi:10.3390/en16176270

[3] "Enefit - predict energy behavior of prosumers," Kaggle, https://www.kaggle.com/competitions/predict-energy-behavior-of-prosumers/code (accessed Dec. 21, 2023).