Efficient energy prediction through the use of machine learning techniques

1st Stanislav Teghipco
Dept. of Computer Science
University of Camerino
Camerino, Italy
stanislav.teghipco@studenti.unicam.it

2nd Luca Bianchi
Dept. of Computer Science
University of Camerino
Camerino, Italy
luca.bianchi@studenti.unicam.it

3rd Héctor Ochoa-Ortiz
Dept. of Computer Science
University of Camerino
Camerino, Italy
hector.ochoaortiz@unicam.it

4th Federico Maria Cruciani
Dept. of Computer Science
University of Camerino
Camerino, Italy
federicomar.cruciani@studenti.unicam.it

***Abstract*—Accurate prediction of household appliance energy consumption is essential for optimizing energy use and enhancing sustainability. This paper presents a comprehensive study on the development of predictive models for estimating the energy consumption of household appliances. Utilizing a rich dataset that includes energy consumption readings, indoor humidity and temperature measurements from room sensors and weather data from stations, we investigate various machine learning techniques, including linear regression, decision trees, and neural networks. Feature engineering processes, such as time-series decomposition and the integration of environmental variables, are employed to improve model performance. This work aims to establish a robust framework for energy consumption prediction, which can be utilized for smarter energy management systems and more efficient energy usage practices. Future research directions include the incorporation of real-time data and extending the model to cover a wider range of household appliances.**

***Index Terms*—energy, machine learning, sustainability, prediction models**

I. Introduction

The number of electrical devices in our homes is rapidly increasing due to various factors, such as the rise of electric vehicles, the growing use of electric heating systems, and the expansion of digital and smart home products. This surge places considerable strain on the electric grid infrastructure, eventually leading to reliability issues and failures [1]. Accurate prediction of energy consumption can play a crucial role in mitigating these challenges. Moreover, efficient energy prediction can reduce costs for users and benefit the environment by enhancing sustainability [2], [3].

In this paper, we compare different machine learning models to identify the best one for predicting the energy consumption of a small household. Our goal is to improve the stability and quality of the grid by enabling energy companies to better plan for periods of high consumption, thereby reducing power outages and their associated costs.

All the code for our approaches, techniques, graphs, and models can be found in the GitHub repository[1], which we used to facilitate collaboration and coordination throughout the project.

II. Literature Review

The importance of energy prediction has led to numerous contributions in recent years, as highlighted by Sun et al. [4] and Lu et al. [5]. Despite these extensive contributions, there is still a gap in applying these technologies to help energy companies plan their operations effectively.

Mohapatra et al. [6] take a different approach by exploring the optimization of models to minimize the required computing power, enabling them to run on infrastructures with limited resources.

Additionally, a study by Olu-Ajayi et al. [7] delves into predicting the potential energy consumption of buildings during their early design stages, providing valuable insights for future construction projects.

III. The Dataset

For this research, we used the "Appliances Energy Prediction Data" dataset, which is available on GitHub [8] and Kaggle [9]. This dataset originates from a study by Candanedo et al. [10] and contains data collected from a house over 137 days (approximately 4.5 months), with measurements taken every 10 minutes. The dataset includes:

- Readings from temperature (Celsius) and humidity (%) sensors located in various rooms (kitchen, living room, laundry room, bathroom, outside the building, ironing room, teenager room, parents room).
- Energy consumption of appliances (Wh).
- Energy consumption of light fixtures (Wh).

Additionally, the dataset includes weather data obtained from the nearest weather station at Chievres Airport (Belgium), specifically containing:

[1]GitHub repository: https://github.com/Staffilon/EnergyConsumptionForecast

- Outside temperature (Celsius).
- Outside humidity (%).
- Pressure (mmHg).
- Wind speed (m/s).
- Visibility (km).
- Dew point (Celsius).

There are also two random variables (rv1, rv2) included for testing regression models.

By examining the dataset, we can derive meaningful insights. Figure 1 illustrates the energy consumption over time, showing the period during which measurements were taken and the actual values of appliance energy consumption. Many peaks and valleys are evident.

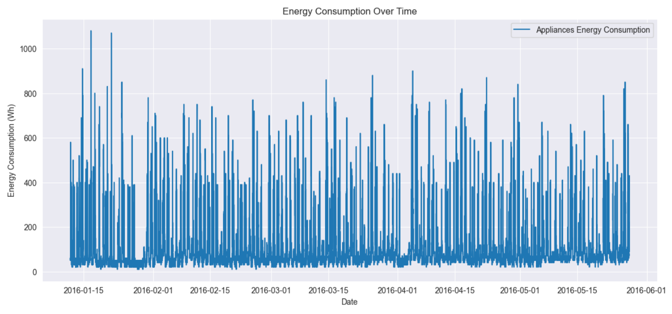

Fig. 1. Energy Consumption Over Time

To investigate the sources of outliers, we analyzed the data on an hourly, daily, and monthly basis. The hourly analysis revealed interesting patterns, as seen in Figure 2. Notably, energy consumption spikes around 5 PM, which aligns with the time people typically return home from work or other activities.

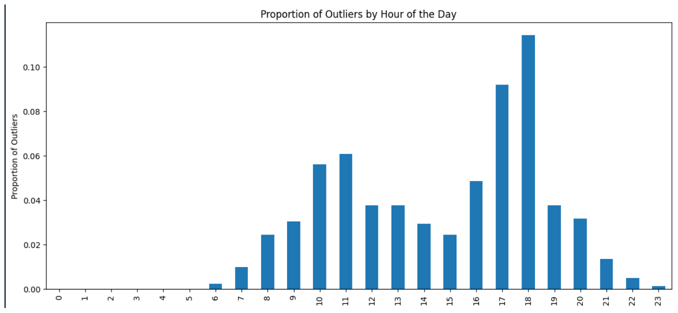

Fig. 2. Outliers Measured by Hour of the Day

In Figure 3, we analyze the distribution of energy consumption and observe that the highest frequency of usage falls within the 50-70 Wh range. Although the distribution is right-skewed, indicating occasional higher consumption values, most usage appears to be within the 0-100 Wh range. This suggests a consistent usage of certain appliances, such as refrigerators, which run continuously throughout the day. The higher consumption values likely correspond to the usage of additional appliances when the household members are home.

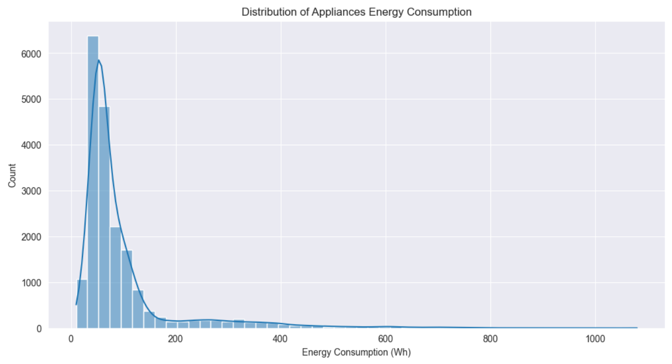

Fig. 3. Distribution of Appliances Energy Consumption

IV. Feature Engineering

After analyzing the dataset, we applied several preprocessing steps to extract more meaningful features from the existing ones before proceeding with model training.

Firstly, we performed the following operations on the "date" field of the dataset:

- Extracted the exact date from the "date" field.
- Derived hours and seconds from the timestamp for finer temporal granularity.
- Determined the day of the week for each record.

The idea behind these steps was to leverage the *temporal* information during model training. During the analysis, we discovered interesting patterns based on the hour of the day, which turned out to be one of the most correlated features with energy consumption.

By plotting the correlation matrix in Figure 4, we can see that the "hours" attribute has the highest positive correlation with energy consumption. This observation justifies our focus on temporal features. To further support this, Figure 5 shows that there is significant variation in energy consumption based on the hour of the day, which is valuable information for the model.

Several additional transformations were applied to the dataset to create new features and prepare the data for modeling:

- A logarithmic transformation was applied to the "Appliances" variable. As shown in Figure 6, using the logarithm results in a distribution resembling a normal distribution.
- An interaction term was created between "hours" and "lights".
- The hourly average of "appliances" was calculated.
- The data was resampled to calculate the mean of numeric columns for 30-minute and 1-hour intervals.
- Qualitative predictors were created for low and high energy consumption based on the hourly average. Consumption is classified as *low* when appliance consumption is significantly lower than the hourly average, and *high* in the opposite case.

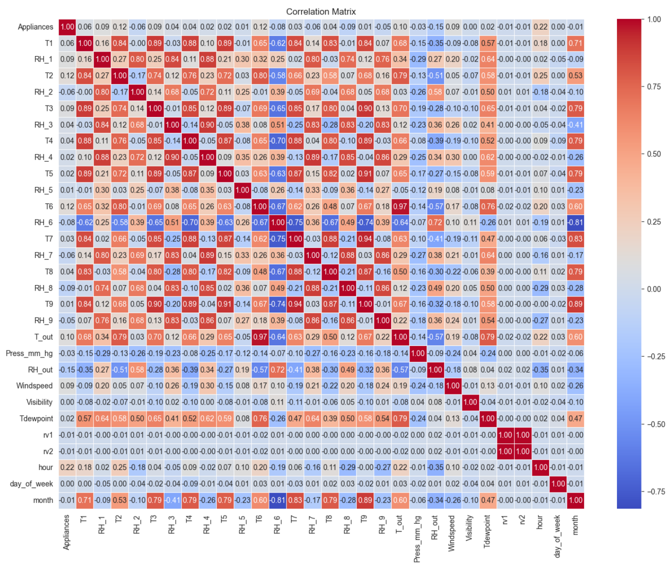
Fig. 4. Correlation Matrix

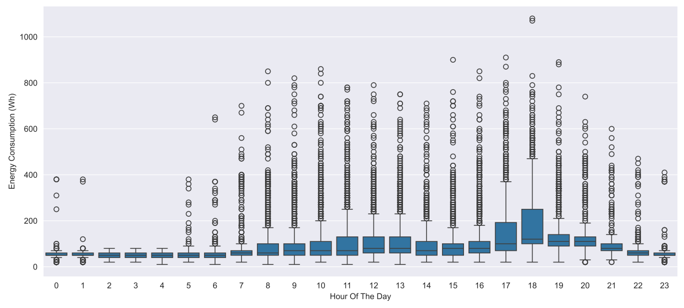

Fig. 5. Appliances Energy Consumption by Hour of the Day

- One-Hot Encoding was applied to the categorical features “weekday” and “hours” to facilitate their use in models.
- All features were scaled to follow a normal distribution.

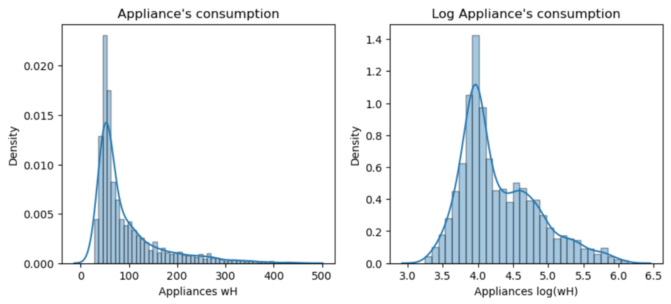

Fig. 6. Comparison of Distributions: Appliances and Log Appliances

V. The Research Approach

The primary objective of this study was to compare the performance of various machine learning models in forecasting the energy consumption of appliances in a low-energy building and to determine the best one. Our research comprised the following phases:

1) **Study Phase:** In this phase, we identified specific models to consider for our analysis.
2) **Training Phase:** Once the models were identified, we trained them to the best of their capabilities.
3) **Analysis Phase:** We analyzed the results each model provided for the forecast.
4) **Comparison Phase:** Finally, we compared the results to determine their performance using reliable comparison metrics.

The research questions we proposed were:

- Which models can predict the energy consumption of a small household with a low error rate?
- Which model gives the best results in terms of reducing prediction error?
- How can the models be leveraged to ensure energy reduction and sustainability in the household?

A. Comparison Metrics

To execute a meaningful comparison, we needed metrics to evaluate the models and determine the best-performing ones. The chosen metrics were:

- **Mean Absolute Error (MAE)**: The average of the absolute differences between the predicted values and the actual values, without considering their directions.
- **Mean Squared Error (MSE)**: The average of the squared differences between the predicted values and the actual values. This metric gives more weight to larger errors by squaring them.
- **Root Mean Squared Error (RMSE)**: The square root of the Mean Squared Error. It provides an error metric on the same scale as the data.
- **R^2 Score**: Represents the proportion of the variance for a dependent variable that’s explained by one or more independent variables in a regression model. Also known as the coefficient of determination.

The best-performing model should have lower error metrics because it indicates that the model is able to forecast future appliance consumption more accurately. Conversely, it should have a higher R^2 value, which indicates that the model is able to explain a larger proportion of the variance in the target variable.

One important consideration is that models tailored for time series forecasts are often univariate, meaning they typically rely on the historical values of the target variable itself for making predictions. This can sometimes result in a lower R^2 value compared to multivariate regression models that use multiple predictors. However, this does not necessarily mean that the univariate time series models are inferior. Metrics like MAE, MSE, and RMSE are often more appropriate for evaluating their performance, as they directly measure the accuracy of the forecasts without being influenced by the complexity of the model.

VI. Model Analysis

In this section, we will delve into the models we used, from the libraries we relied on to how we fit the data and made forecasts. To create a solid comparison baseline, we chose four models from each category: **Time Series** models and **Regression models**. This approach gives us enough data to make meaningful comparisons and draw valuable conclusions.

A. Employed Libraries and Hardware

When it comes to the libraries we used, we stuck with industry standards. For data manipulation and analysis, we used numpy and pandas. For preprocessing and evaluation metrics, we turned to scikit-learn, while statsmodels handled our statistical modeling. For visualization, we relied on matplotlib and seaborn. We also used Keras for building and training our neural network models.

Here's a quick rundown of the main libraries:

- **pandas:** This was our go-to for data manipulation and analysis. It provides powerful data structures like DataFrame, which are essential for handling time series data.
- **numpy:** We used numpy for numerical operations on large, multi-dimensional arrays and matrices, which are crucial for data processing.
- **scikit-learn:** This library provided tools for data preprocessing (like StandardScaler) and model evaluation metrics (such as mean absolute error and mean squared error).
- **statsmodels:** We used statsmodels for statistical modeling and hypothesis testing, including models like ARIMA and SARIMAX.
- **matplotlib:** This plotting library was used to create static, interactive, and animated visualizations.
- **seaborn:** Built on top of matplotlib, seaborn provided a high-level interface for drawing attractive statistical graphics.
- **Keras:** A high-level API for neural networks that runs on top of TensorFlow. We used it for building and training our deep learning models.

These libraries, alongside others tailored for specific models, provided us with a comprehensive toolkit for data manipulation, model building, and evaluation, enabling us to effectively analyze and compare the performance of different time series and regression models.

As for the employed hardware, we each used our personal laptops to execute the files. Additionally, we made extensive use of **Google Colab**, a hosted **Jupyter Notebook** service that provides free access to computing resources, including GPUs and TPUs. This allowed us to speed up the training process and handle larger datasets more efficiently.

B. Dataset Split

To train and test the models, we split the original dataset into 80% for training and 20% for validating the results. Since a sequential split is necessary for data forecasting, shuffling the data was not required.

C. Training Approach

In our comparison, we had to use different training methods for the two types of models.

We were able to train our regression models using the entire training dataset at once. This approach yielded good metrics and worked well for these models.

However, the same approach didn't work well for our forecasting models. This is why we turned to the **Expanding Window Approach**. This technique systematically updates the training data with new data points from the test set as they become available. By always including the most recent data, the model can continuously learn and improve its predictions. This involves expanding the training window to incorporate all previous observations up to the current time, allowing the model to adapt to new information. The process works as follows: at each step, a new batch of test data is added to the training set, the model is retrained, and predictions are made for the next batch. This method helps avoid issues with non-stationarity and changing patterns in the data, making it particularly effective for time series forecasting.

A crucial aspect of this approach is the **batch size**, which refers to the number of new data points added to the training set at each step. The batch size significantly affects the model's training time, so it's important to find a balance that provides good results without excessive training time.

D. Time Series Models

Time series models [11] are vital tools for analyzing and forecasting data points collected at specific time intervals. These models treat time as an independent variable and aim to understand underlying patterns to predict future values accurately. When working with this type of data, it's important to consider three key characteristics: autocorrelation, seasonality, and stationarity. Understanding these aspects helps in choosing the right time series model for analysis.

Autocorrelation: Autocorrelation refers to the similarity between observations based on the time lag between them. In simpler terms, data points that are closer in time tend to be more similar than those further apart. Recognizing autocorrelation helps identify repeating patterns or cycles in the data, which can be crucial for accurate forecasting.

Seasonality: Seasonality involves regular fluctuations that occur at consistent intervals, such as daily, monthly, or yearly cycles. For instance, electricity consumption might peak during the day and drop at night, or retail sales might spike during the holiday season. Identifying seasonality in the data allows us to incorporate these patterns into our models, improving their predictive power.

Stationarity: A time series is stationary if its statistical properties, like mean and variance, remain constant over time. Stationarity is important because many time series models assume the data is stationary. Non-stationary data can show trends, seasonality, or other structures that can affect model performance. If the time series is not stationary, we can apply transformations like differencing or detrending to make it stationary.

In this section, we'll explore different types of time series models, such as Moving Average, Exponential Smoothing, and Seasonal AutoRegressive Integrated Moving Average (SARIMA). Each model has its own methodology and application areas, allowing us to choose the best fit for specific forecasting needs.

1) ETS: The Exponential Smoothing State Space Model (ETS) [12] is a powerful and flexible class of models that are used in the context of data forecasting. The ETS acronym stands for Error, Trend, and Seasonality, capturing three key components of time series data:

- **Error (E):** This component captures the random fluctuations in the data.
- **Trend (T):** This component captures the underlying direction in which the data is moving over time, which can be either additive or multiplicative.
- **Seasonality (S):** This component captures the repeating short-term cycles in the data, which can also be either additive or multiplicative.

The model can combine these components in various ways, resulting in different types of exponential smoothing models.

In our configuration of the model, we used the **Akaike Information Criterion** (AIC) to select the best combination of parameters. A lower AIC indicates a better model. The parameters of choice were: `Trend: add, Seasonal: None`. For the batch size used in the forecasting segment, the value was set to **100**.

2) ARIMA: AutoRegressive Integrated Moving Average (ARIMA) [13] is a popular and widely used class of models for time series forecasting. The ARIMA model combines three components:

- **AutoRegressive (AR):** This component involves regressing the variable on its own lagged (past) values. The order of the AR term is denoted by p.
- **Integrated (I):** This component involves differencing the data to make it stationary, which means removing trends and seasonality. The order of the I term is denoted by d.
- **Moving Average (MA):** This component involves modeling the error term as a linear combination of past error terms. The order of the MA term is denoted by q.

In our configuration of the model, we used the **Augmented Dickey-Fuller (ADF)** test to determine the stationarity of the series. The ADF test returned a p-value of 0, indicating that the series is stationary and does not require differencing (i.e., $d = 0$). Following this, we employed the AIC to select the best combination of AR and MA terms. The combination that resulted in the lowest AIC value was $(p, d, q) = (2, 0, 2)$, which we used in our final ARIMA configuration.

3) SARIMA: The Seasonal AutoRegressive Integrated Moving Average (SARIMA) [14] is an extension of the ARIMA model that explicitly supports univariate time series data with a seasonal component. SARIMA incorporates seasonal elements into the ARIMA framework, making it suitable for data with regular seasonal patterns. The SARIMA model includes:

- **Seasonal AutoRegressive (SAR):** This term represents the relationship between an observation and a lagged observation from the previous season.
- **Seasonal Differencing (SI):** This term is used to remove the seasonal component of the time series.
- **Seasonal Moving Average (SMA):** This term represents the relationship between an observation and a lagged error from the previous season.

Similarly to the ARIMA model, we used the ADF test to analyze the stationarity of the series, obtaining the same results. Then we employed the AIC to select the best combination of AR, MA, and seasonal terms. The combination that resulted in the lowest AIC value was $(p, d, q) = (2, 0, 2)$ for the non-seasonal components and $(P, D, Q, m) = (1, 0, 1, 24)$ for the seasonal components, which we used in our final SARIMA configuration.

4) LSTM: Long Short-Term Memory (LSTM) [15] networks are a specialized form of recurrent neural network (RNN) designed to address their traditional limitations, specifically the issue of long-term dependencies and the vanishing gradient problem. The LSTM network architecture includes:

- **Cell State:** This represents the memory of the network, capable of carrying information across many time steps, thus preserving long-term dependencies.
- **Forget Gate:** This gate decides whether the information from the previous timestamp should be retained or forgotten.
- **Input Gate:** This gate determines the importance of the new information being added to the cell state from the current input.
- **Output Gate:** This gate controls the output based on the cell state and the current input, determining the next hidden state.

In our configuration of the LSTM model, we used a bidirectional LSTM network with two layers of Bidirectional LSTM units (100 and 50 units respectively), dropout layers (each with a dropout rate of 0.2), and a Dense layer with a single neuron. The model was compiled using the mean squared error loss function and the Adam optimizer. We trained the model for 70 epochs with a batch size of 10, using a validation set to monitor the training process. This configuration allowed the LSTM network to effectively learn and capture the underlying patterns in the time series data for accurate forecasting.

E. Linear Regression Models

Linear regression models aim to predict a target variable by establishing a linear relationship with one or more predictor variables. The main difference between such models and the previously illustrated ones is that this category of models can't natively consider the temporal information of a time series. However, as we saw in the feature engineering section, by extracting new features from the timestamp of the time series, we managed to train the models to include these as well.

After trying many possible linear regression models, we propose the four prominent regression models that performed better and are known for their effectiveness in various prediction tasks: Random Forest, Extra Trees, XGBoost, and LightGBM. Each of these models employs different techniques to extend the basic linear regression and improve prediction accuracy.

1) Random Forest Regressor: The Random Forest [16] is an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time. In this case, the Random Forest Regressor combines the results of the trees by considering the average prediction of individual trees. The idea behind the creation of multiple trees is to reduce overfitting and improve predictive accuracy by leveraging the wisdom of the crowd.

More in detail, the Random Forest, among the possible ensemble methods, uses the bagging technique, thanks to which the trees in the Random Forest run in parallel, meaning that there is no actual interaction between these trees during the training phase.

The model can be easily described using the following formulation:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} T_i(x)$$

where T_i represents the i-th decision tree in the forest.

The most relevant advantages of this model are the following:

- It runs efficiently on large datasets.
- It can handle thousands of input variables without variable deletion.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.

In our case, the parameters of the model that performed the best are:

- *max_depth*: 800 (The maximum depth of the tree)
- *min_samples_leaf*: 10 (The minimum number of samples required to be at a leaf node)
- *min_samples_split*: 5 (The minimum number of samples required to split an internal node)
- *n_estimators*: 60 (The number of trees in the forest)

2) Extra Trees Regressor: The Extra Trees Regressor, or Extremely Randomized Trees Regressor [17], is similar to the Random Forest Regressor but differs in the way trees are constructed. Instead of looking for the most discriminative threshold, thresholds are chosen at random. Usually, the training phase of a tree involves determining the best threshold to perform the splitting and obtain the maximum information gain, but this doesn't apply in this algorithm. Instead, it builds many different trees by randomizing the chosen features and the respective split threshold in each node.

As in the previous case, to combine the results from each tree we computed the mean of all predictions, meaning that the model will be:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} T_i(x)$$

where T_i represents the i-th decision tree, but each of them is constructed considering random splits.

By creating trees in such a way, the resulting trees will be much more diversified and uncorrelated, meaning that the overall model will be able to further reduce the variance compared to Random Forests. Furthermore, the training phase will be much more efficient since it doesn't have to do many computations by only randomizing the features and thresholds.

Regarding the used parameters for this model, they are the following:

- *max_depth*: 80 (The maximum depth of the tree)
- *n_estimators*: 200 (The number of trees in the forest)

3) XGBoost Regressor: XGBoost, or Extreme Gradient Boosting [18], is a powerful ensemble learning method based on decision trees. It is designed to be efficient, flexible, and highly scalable, often achieving state-of-the-art performance. Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems.

In the case of the regressor, the results of the built trees will be averaged and returned as a single result.

The key features and concepts behind this algorithm are:

- The gradient boosting technique it uses, thanks to which it iteratively adds decision trees to the model, each one correcting the errors made by the previous ones, minimizing the loss function.
- It includes regularization terms in its objective function to prevent overfitting.
- Parallel processing, thanks to which training process is greatly sped up.

In this case, the formula of the model is the typical gradient descent formulation:

$$\hat{y}^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i)$$

where η is the learning rate and f_t is the t-th tree added to the model.

Related to the dataset, the most relevant parameters of the model that we used are:

- *eta*: 0.1 (Step size shrinkage used in update to prevent overfitting)
- *max_depth*: 4 (Maximum depth of a tree)
- *booster*: *gbtree* (Which booster to use to train the model)

4) LightGBM Regressor: Light Gradient-Boosting Machine (LightGBM) [19] is another open-source implementation of the gradient-boosting algorithm, designed to be distributed, efficient, and optimized for large datasets. It is able to filter out less useful data instances by using the techniques of Gradient-based One Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

The main features of the LightGBM algorithm are the following:

- Unlike level-wise tree growth used in many other gradient boosting implementations, LightGBM grows trees leaf-wise, choosing the leaf with the maximum loss to split. This might lead to deeper and more accurate trees.

- LightGBM uses histogram-based algorithms to bucket continuous feature values into discrete bins, reducing the training process.
- It uses, as already mentioned, GOSS to focus on data points with larger gradients (paying more attention to *hard-to-predict* samples) and EFB to combine mutually exclusive features to reduce dimensionality.

Lastly, regarding the used parameters for this model, they are:

- *max_depth*: 800 (Maximum tree depth for base learners)
- *n_estimators*: 40 (Number of boosted trees to fit)

VII. Obtained results

In Table I, we can see the obtained results from each of the models, based on the metrics that we established before.

TABLE I
Performance Metrics for Different Models

Model	MAE	MSE	R²	RMSE
ETS	70.21	19015.31	-1.29	137.89
ARIMA	52.02	8200.08	0.01	90.55
SARIMA	51.24	8077.14	0.02	89.87
LSTM	42.299	4509.528	0.188	67.153
Random Forest Regressor	20.552	1367.768	0.55	36.983
Extra Trees Regressor	20.295	1374.939	0.55	37.080
XGBoost Regressor	20.861	1448.366	0.52	38.057
LightGBM Regressor	20.445	1468.749	0.52	38.324

The comparison of different machine learning approaches, including linear regression, decision trees, and neural networks, provided valuable insights into their strengths and limitations. Among these, the Random Forest Regressor and the Extra Trees Regressor showed the most promise in terms of reducing prediction error, suggesting that they are well-suited for handling the complexity and variability inherent in energy consumption data.

Overall, the other linear regression models, the XGBoost Regressor and the LightGBM Regressor, performed slightly worse than the Random Forest and Extra Trees Regressors but still achieved very good results based on the metrics.

Based on the results, the time series models (ETS, ARIMA, SARIMA, and LSTM) performed the worst. Aside from the R² metric, which may not be the most suitable for evaluating these models, we can see that they struggled to make accurate predictions. This could be attributed to the relatively short duration of the data (4.5 months), which might not provide enough information for the models to capture meaningful patterns. Additionally, the lack of strong seasonal patterns further limited their effectiveness.

This can be further analyzed in Figure 7, where we report the performance of the models in a scatter plot based on RMSE and R². Below, we can see a cluster of points, representing the regression models, indicating high R² and low RMSE. On the other hand, the time series models are positioned quite far away. In particular, the ETS model shows notably the worst performance out of the eight models.

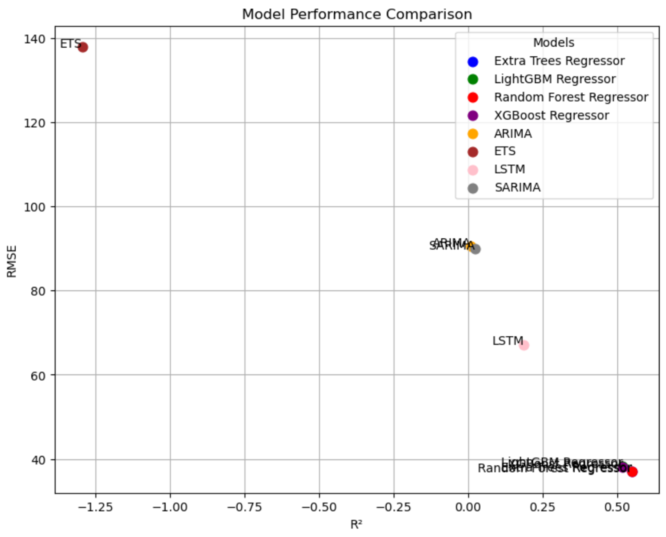

Fig. 7. Model Performance Comparison

Lastly, Figure 8 shows the predictions made by the Random Forest Regressor. Even though this was one of the best models, we can see that it still didn't match the actual expected values perfectly. However, it gets fairly close, except for the outliers that it didn't manage to identify.

Despite this, the results are not enough to conclude that regression models will always perform better than time series models in every scenario. The obtained results are significant in this specific case, but in others, it might be the opposite. While time series models can take full advantage of the temporal information inherent in a time series, regression models cannot do the same. In this particular dataset, we can safely assume that regression models performed better for the following reasons: even though 4.5 months worth of data seems substantial, for time series models that try to grasp temporal patterns, it might still not be enough. Through the analysis, we observed that the "hours" feature was an important one, but other than this detail, there weren't any other recurrent temporal patterns that time series models could utilize, limiting their performance. The dataset comprises many features, which usually benefit regression models. While time series models are designed to work even in the absence of such features, regression models are the only ones that can truly utilize this additional information. As already mentioned, the R² measures how well independent variables explain the dependent variables, but in the case of time series models such as ARIMA, SARIMA, and ETS, which are trained without considering independent variables, the resulting value of this metric will naturally be low.

VIII. Conclusions

In conclusion, this study has demonstrated the efficacy of various machine learning models for predicting household energy consumption. These models can be employed by energy companies to better plan their maintenance activities and

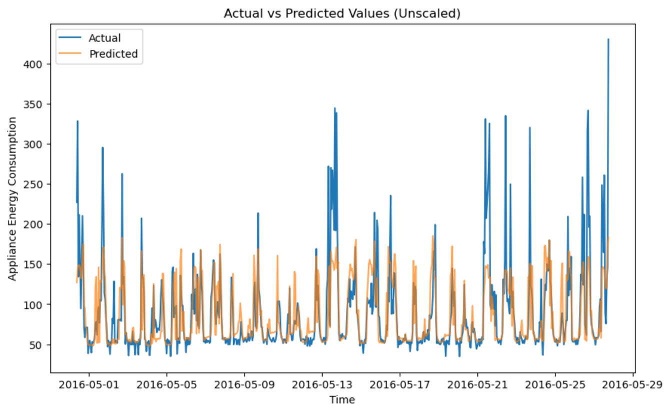

Fig. 8. Random Forest Regressor Predictions

prepare for high-intensity periods. As a result, the power grid will become more stable, leading to a higher quality of service for users and reducing resource waste through more efficient planning.

This work was conducted with a limited dataset from a single house, which may affect the representativeness of the data. Additionally, details on the house's construction, thermal insulation, and other relevant factors are not provided in the dataset. Furthermore, the dataset has been extensively studied in the literature, which limits the potential for groundbreaking discoveries.

Looking ahead, this project should be kept up-to-date as new models emerge and the field of artificial intelligence continues to evolve, offering new techniques and solutions to support sustainable energy transitions. With more extensive data, additional machine learning techniques could be explored, and strategies to adapt household consumption patterns to utilize energy when it is cheaper or for grid balancing could be developed, thereby broadening the scope of research on household energy consumption.

References

[1] K. Brigham, "Why the electric vehicle boom could put a major strain on the U.S. power grid," Jul. 2023, section: Climate. [Online]. Available: https://www.cnbc.com/2023/07/01/why-the-ev-boom-could-put-a-major-strain-on-our-power-grid.html

[2] International Energy Agency, "Emissions savings – Multiple Benefits of Energy Efficiency – Analysis," Tech. Rep., Mar. 2019. [Online]. Available: https://www.iea.org/reports/multiple-benefits-of-energy-efficiency/emissions-savings

[3] Sense Team, "Why Energy Efficiency is a Win-Win for Consumers and Utilities," Jun. 2023. [Online]. Available: https://blog.sense.com/why-energy-efficiency-is-a-win-win-for-consumers-and-utilities/

[4] Y. Sun, F. Haghighat, and B. C. Fung, "A review of the-state-of-the-art in data-driven approaches for building energy prediction," *Energy and Buildings*, vol. 221, p. 110022, 2020.

[5] C. Lu, S. Li, and Z. Lu, "Building energy prediction using artificial neural networks: A literature survey," *Energy and Buildings*, vol. 262, p. 111718, 2022.

[6] S. K. Mohapatra, S. Mishra, H. K. Tripathy, and A. Alkhayyat, "A sustainable data-driven energy consumption assessment model for building infrastructures in resource constraint environment," *Sustainable Energy Technologies and Assessments*, vol. 53, p. 102697, 2022.

[7] R. Olu-Ajayi, H. Alaka, I. Sulaimon, F. Sunmola, and S. Ajayi, "Building energy consumption prediction for residential buildings using deep learning and other machine learning techniques," *Journal of Building Engineering*, vol. 45, p. 103406, 2022.

[8] L. Candanedo, "LuisM78/Appliances-energy-prediction-data," 2017. [Online]. Available: https://github.com/LuisM78/Appliances-energy-prediction-data

[9] Gokagglers, "Appliances Energy Prediction," 2017. [Online]. Available: https://www.kaggle.com/datasets/loveall/appliances-energy-prediction

[10] L. M. Candanedo, V. Feldheim, and D. Deramaix, "Data driven prediction models of energy use of appliances in a low-energy house," *Energy and Buildings*, vol. 140, pp. 81–97, Apr. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778816308970

[11] Marco Peixeiro, "The complete guide to time series models," 2023. [Online]. Available: https://builtin.com/data-science/time-series-model

[12] Ajay Taneja, "Time series forecasting: Ets models," 2022. [Online]. Available: https://www.linkedin.com/pulse/time-series-forecasting-ajay-taneja/

[13] Neha Bora, "Understanding arima models for machine learning," 2021. [Online]. Available: https://www.capitalone.com/tech/machine-learning/understanding-arima-models/

[14] Ritu Santra, "Introduction to sarima model," 2023. [Online]. Available: https://medium.com/@ritusantra/introduction-to-sarima-model-cbb885ceabe8

[15] Shipra Saxena, "What is lstm? introduction to long short-term memory," 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[16] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 10 2001.

[17] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, 04 2006.

[18] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system." in *KDD*, B. Krishnapuram, M. Shah, A. J. Smola, C. Aggarwal, D. Shen, and R. Rastogi, Eds. ACM, 2016, pp. 785–794. [Online]. Available: http://dblp.uni-trier.de/db/conf/kdd/kdd2016.html#ChenG16

[19] G. Ke, Q. Meng, T. Finely, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30 (NIP 2017)*, December 2017. [Online]. Available: https://www.microsoft.com/en-us/research/publication/lightgbm-a-highly-efficient-gradient-boosting-decision-tree/