HOME ENERGY MANAGEMENT SYSTEM

Tirado Barragán Pablo

13/12/2025 ITS8080 ID:256063IV

Contenido

Introduction and project scope ... 2
Project planning ... 3
  Project Flowchart ... 3
  Effort Estimation ... 3
Data acquisition ... 4
Exploration and Visualization ... 4
  Dataset description ... 4
  Visualization (task 3) ... 5
Data cleaning ... 7
  Mandatory PV Analysis ... 7
  Forensic Data Audit ... 8
  Advanced Forensic Cleaning ... 9
    Physics-based safety mechanisms ... 9
    Temporal Reindexing ... 9
    Smoothing Temperature sensor noise ... 10
  Conclusion ... 10
Feature Engineering ... 10
  Statistical Analysis ... 10
  Advanced Feature Construction ... 12
    Cyclical time encoding ... 12
    Inertia and Autocorrelation ... 12
    Physics-based interactions ... 12
    Feature Ranking and Selection ... 12
  Dimensionality reduction ... 13
  Time series analysis ... 13
    Classical Additive decomposition ... 13
    Typical demand profiles ... 14
Predictive Modelling ... 15
  Statistical Modelling (ARIMA Family) ... 15
    Stationarity and Differencing ... 15
    Autocorrelation Structure (ACF and PACF) ... 15
  Model selection and training ... 16
    Non-Seasonal ARIMA (Aggressive) – ARIMA(4, 1, 4) ... 16
    Seasonal SARIMA (Expert) – SARIMA(1, 0, 1)x(0, 1, 1, 24) ... 16
  Performance Evaluation (walk-forward validation) ... 16
  Machine Learning Approach ... 17
    Model selection: eXtreme Gradient Boosting (XGBoost) ... 17
    Hyperparameter Optimization and Rationale ... 18
    Comparative Evaluation: XGBoost vs SARIMA ... 18
  Forecasting Pipeline ... 19
    Pipeline Architecture and Methodology ... 19
    Forecast Generation and comparison ... 19
    Quantitative Evaluation ... 20
  Conclusion ... 20
  Models with input: Impact Analysis ... 21
    The value of information ... 21

Quantitative evaluation 21
Visual Analysis of forecast dynamic 22
Conclusion 22
Optimization........ 22
Methodology........ 22
Demand Forecasting........ 23
Optimization Engine 23
Scenario Analysis and Results........ 23
Scenario A: PV Low (Cloudy/winter) 24
Scenario B: PV High (Sunny) 24
Conclusion........ 24

Introduction and project scope

Modern households face a key problem: how to balance energy use with renewable sources as solar panels and electric cars become more common, along with changing electricity prices. If not managed well, extra solar energy could be lost, and electricity bills may stay higher than needed.

In this work, a Home Energy Management System (HEMS) acts as a key controller. This study creates a complete data science process for a HEMS, with the goal of improving how a home battery storage system works.

The main objectives of the study are:

- To create data-driven models that accurately predict hourly electricity demand (kW) for the upcoming day
- To design an algorithm that reduce the annual electricity bill optimizing the schedule battery charging and discharging based in different variables.

Digitalization transforms the residential energy sector through several key mechanisms.

- Transforms the collected data in real-time to enable dynamic and precise billing.
- Optimize algorithms that allows devices to respond automatically to price signals without human intervention.
- Decentralize the manage of the systems from the active work of a unified system that controls everything to different active independent systems that manage their own generation and the sale of surplus energy to the grid

We are going to work with solar generation data, which requires us to focus on two main problems:

- Solar energy is intermittent and non-dispatchable. Without historical data and accurate predictions, it would be impossible to ensure grid stability or even a plan battery charging and discharging cycles. Representing the major financial and technical risk in a renewable system.

The use in private sectors and business sector. In a private sector, it can maximize the self-consumption and correct the sizing of batteries to cover nighttime or bad-weather demand. In the other side, by a business sector it could be utilized for peak shaving, that focus on the reduction of peaks in the contracted power, and for complying with sustainability

Project planning

To address energy use, a workflow using the Data Science Lifecycle was created. This method makes sure raw data changes into useful choices about how to operate.

Project Flowchart

To tackle the energy problem, a typical Data Science Lifecycle workflow was created. This ensure raw data changes into useful choices for operations. The plan is set up like a Directed Acyclic Graph (DAG), joining getting data to the final plan. The flow for this info is:

- Data Acquisition: Taking in train_test.csv, forecast.csv, and optimisation.csv.
- Project Planning: The actual phase of the project where we define the structure and the following actions.
- Audit and Cleaning (Data Cleaning): Finding odd outliers (like sun power at night) and filling in blanks (MICE).
- Feature Engineering: Turning timestamps into repeating types (Sine/Cosine) and making delay types to grab heat and use delay.
- Predictive Modeling: Making and matching math models (SARIMA) with Machine Learning (XGBoost).
- Optimization (Decision Making): Using guesses as input for a Linear Programming way that runs the battery.
- Evaluation: Figuring out net money saved.

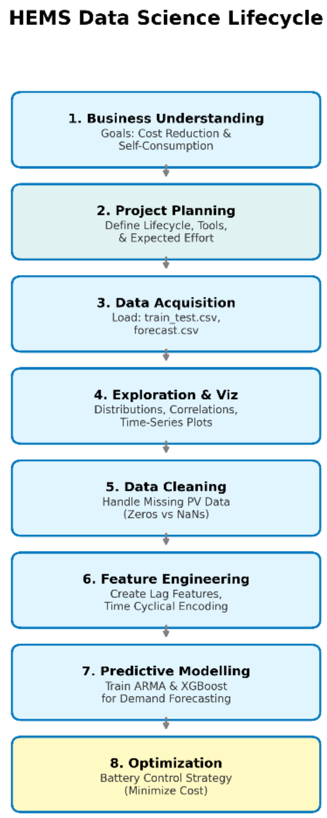


Figure 1: Project flowchart

Effort Estimation

The initial data exploration suggests that preprocessing, which includes cleaning and feature engineering, will likely be the most time-intensive part of this project, taking up around 60-70% of the work.

This was estimated because the actual sensor data often has issues like incorrect values and missing time points that we need to find and correct. Fixing these issues carefully is important; if not, the math tool needed for Task 11 won't work right or will give impossible answers. Also, creating features takes a lot of work because Machine

Figure 2: Project planning diagram

Learning models don't automatically get time-related ideas. So, we have to manually make variables that show cycles (like hours, days) and weather conditions so the models can learn harder trends well.

Data acquisition

After checking what we have in the current files (timestamp, demand, price, PV, weather), we don't think extra data is needed to meet the project's targets. Weather Data: The info we already have covers temperature, solar radiation, and wind speed, including past data and future predictions (forecast.csv).

Possible Improvement (Optional): The only external data that might help a bit is a local holiday calendar. This could aid the models to tell the difference between a regular Sunday (low demand) and a Holiday Monday (also low demand), preventing overestimations on unusual weekdays. Though, for what we're doing in this project, the data we have is good enough.

Exploration and Visualization

Dataset description

The analysis was followed using a dataset that represent the hourly electricity consumption for a household with some extra information as the weather and pricing data. The data was provided in three main files:

- train_xxxxxx.csv: Previous data for model training (demand, weather, price…). The xxxxxx represents the number associated to the student to choose the file that corresponds to each. In my case the file was train_256063.csv
- forecast.csv: future weather and price data for the prediction horizon.
- Optimisation.csv: Inputs for the battery control optimization task.

The main variables to identify in the model were:

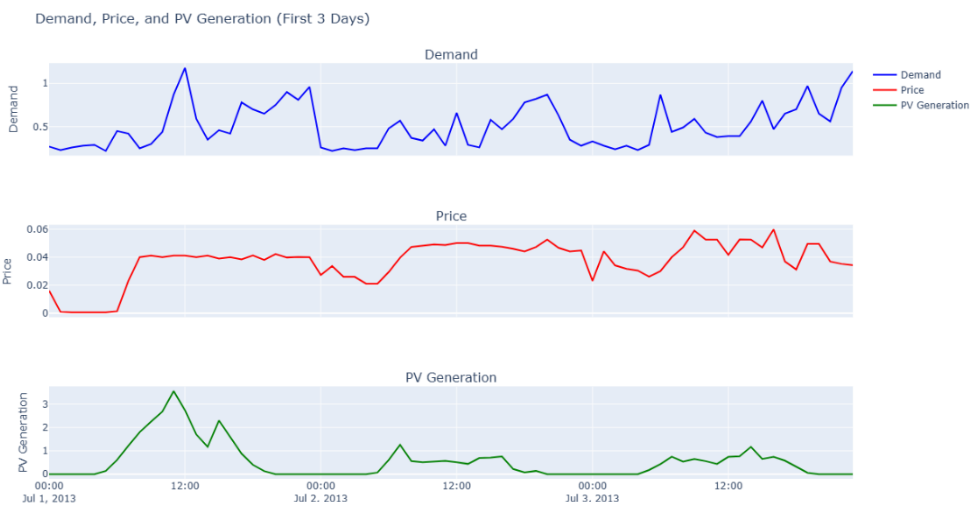


Figure 3: Visualization of demand, price and PV generation

- Target: demand (Electricity load in kW)
- Weather Features: temperature, pressure, wind_speed, cloud_cover and PV_generation.
- External: Price (tariff in €/kW)

The solar generation profile follows a distinct diurnal pattern (gaussian bell curve), peaking around midday when solar irradiance is highest. A key point is that power production stops at night. Daily peak output changes because of clouds and weather.

The household consumption is highly stochastic based on human behaviou. Two peaks time can be seen in the morning when people wake up and a bigger one in the evening when they use lights, cook and relax. It is necessary to say that there is still some demand at night, mainly because of appliances that run constantly.

The Electricity Price are dynamic.This feature is interesting to train predictive models and optimize the energy expenditure.

Visualization (task 3)

To prepare the data for cleaning, statistical summaries were generated, and key variables like demand, price, and PV generation were visualized. Histograms were used to look at distributions, and boxplots helped find outliers.

To get information of the features selected, there were made two different types of graphs:

- Histograms: The histograms can show the distribution patrons of the data to understand if there is a relationship between the amount and the quantity.
- Boxplots: The boxplots identify outliers easily by the way they are created, showing the 50% of the data in a "box" with a defined centre that identify the mean, then in a straight line the rest of the points followed by what can be considered outliers.

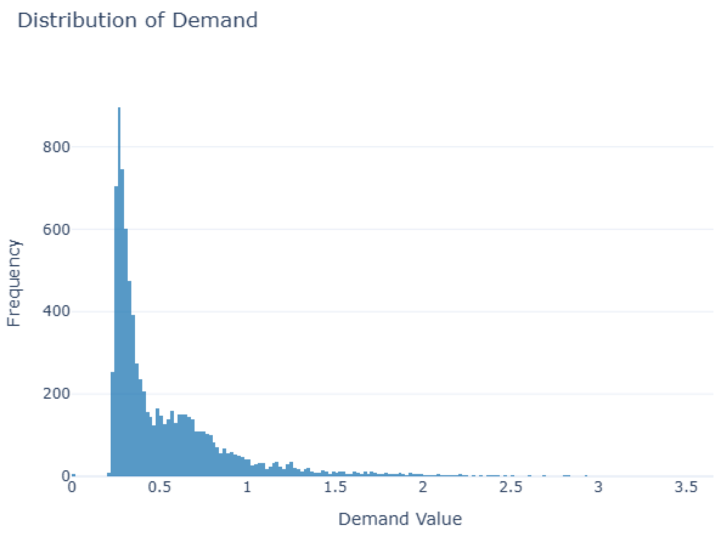


Figure 4: Demand Histogram

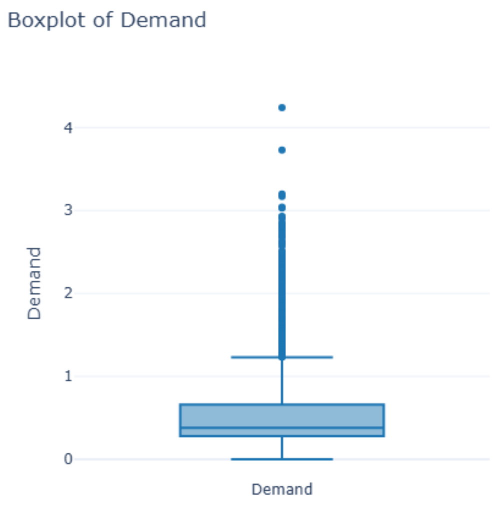


Figure 5: Demand Boxplot

Demand:

- Histogram: The demand seems to follows a gaussian-like distribution with a slight right skew. Most consumption values cluster around the mean, but the tail indicates occasional periods of high demand, likely driven by specific appliances of activities
- Boxplot: Outliers only appear in the upper range, suggesting moments of high consumption that statically can be considered abnormal, but because of the topic treated, it could only be some events or appliances that run simultaneously that is indeed needed.

Figure 6: Price Histogram

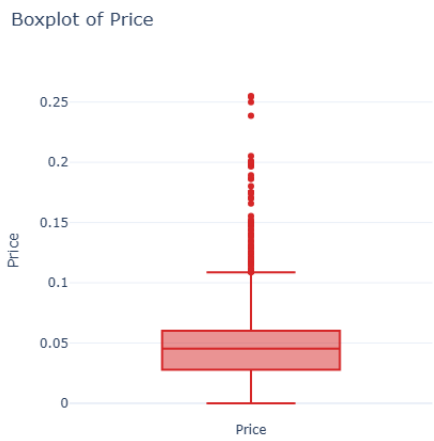


Figure 7: Price Boxplot

Price:

- Histogram:The price distribution is not a simple notrmal distribution, byt a multi-modal structure of several gaussians distribution. This behavior is common in electricity markets where the prices tend to cluster around specific levels depending on the time of day. This will have implications in the creation of linear models, that might struggle to predict its behaviour, prefixing some non-linear models or including time-based features.
- Boxplot: The Boxplot identifies numerous upper outliers, that represent market price spikes. These occur during periods of grid stress of high demand or low renewable supplies, which leads in the increase of the prices. These anomalies area one of the most important data points because with that we can optimize the strategy to minimize costs.

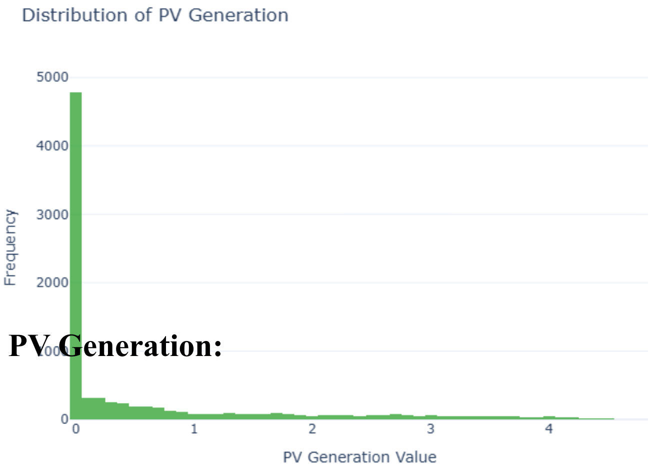


Figure 9: PV Generation Histogram

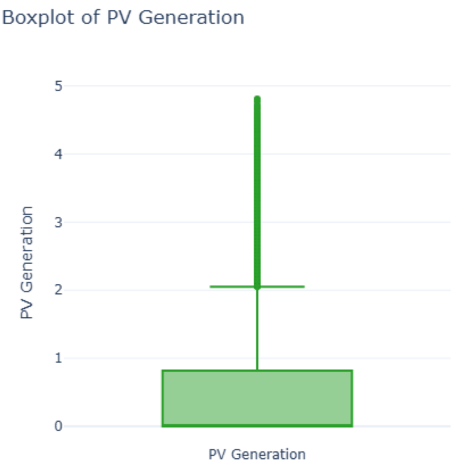


Figure 8: PV Generation Boxplot

PV Generation:

- Histogram: The most of the observations (almost 5000) are exactly 0kW. This is consistent, because the day-night cycle, at night there would be approximately 50% of the time night hours. The remaining data points form a long tail of positive values, which represents varying levels of solar irradiance during the day. It is important to keep the zero values so even if they make difficult to see the bell curve that indeed there is, these periods of zero generation represents the critical moments when the home relies entirely on the grid or the battery, another important problem we need to solve.
- Boxplot: The data is highly compressed due to the zero-inflated nature, but it still have significant number of positive outliers. A closer inspection of the outliers, reveals that many are small values of night hours of low-light conditions (Estonian data?). This is important because is illogical that a solar panel produce energy at 3AM, so it can be sensor noise or calibration errors. That's why, in the cleaning data it was forced the generation to 0kW during night hours to prevent this data to trick the optimization algorithm into believing the battery could be charged at night.

Data cleaning

The goal of this phase is to transform the raw dataset into a high-quality, physically consintent time series. While the assignment specifically requests the restoration of pv_mod1, our analysis required a broader cleaning strategy to prepare the entire dataset (demand, price and total pv) for the optimization task.

That is why this section is divided into two parts:

- Mandatory pv analysis: addressing the specific requirements for pv_mod1 (Deletion vs Univariate vs MICE)
- Advanced forensic cleaning: Additional steps we took to correct phusical errors (nighttime solar noise) and temporal discontinuities that the basic assignment didn't cover but can improve our HEMS success.

First, we can visualize the missing data in each feature of the pv_mod1, pv_mod2 and pv_mod3:

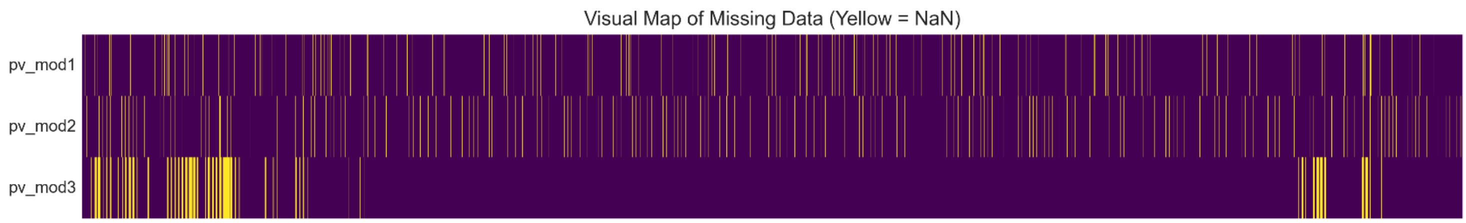


Figure 10: Missing values for pv_mod1, pv_mod2 and pv_mod3

Mandatory PV Analysis

In first place we must identify the issues that pv_mod1, pv_mod2 and pv_mod3 significant blocks of missing data.

The missingness is likely Missing at Random (MAR) or Missing Completely at Random (MCAR) due to sensor communication failures, as the gaps span full hours/day independent of the generation value itself.

The outliers, as explained in the previous task, were detected physically impossible values as the negative generation and sensor noise during night hours.

That's why we ended up using primarily imputation methods implementations on the three pv_mod before applying the best one to the full system

Forensic Data Audit

We implemented different strategies on pv_mod1 before applying the optimal method to the full system:

- Deletion: Removing rows with missing values. Rejected as it breaks time-series continuity
- Univariate imputation: linear interpolation. Efficent for short gaps but created unrealistic flat lines during longer outages
- Multivariate imputation: We applied MICE using temperature and hour as context predicators, which successful reconstructed the solar curves, leveraging correlations between heat, time and sunlight.

We chose the last method for its consistency.

After MICE method, it is preserved the tdistribution best without reducing the row count

	Original	Univariate	MICE (Expert)
Count	0.6622	8759.0000	8759.0000
Mean	0.6622	0.6600	0.6596
Std	1.1080	1.1028	1.1065
Min	0.0000	0.0000	0.0000
25%	0.0000	0.0000	0.0000
50%	0.0000	0.0000	0.0000
75%	0.8300	0.8300	0.8200
max	4.8100	4.8100	4.8100

Also, after the change, we compared the results of the previous data vs the MICE method data, which shows that the data is consistent.

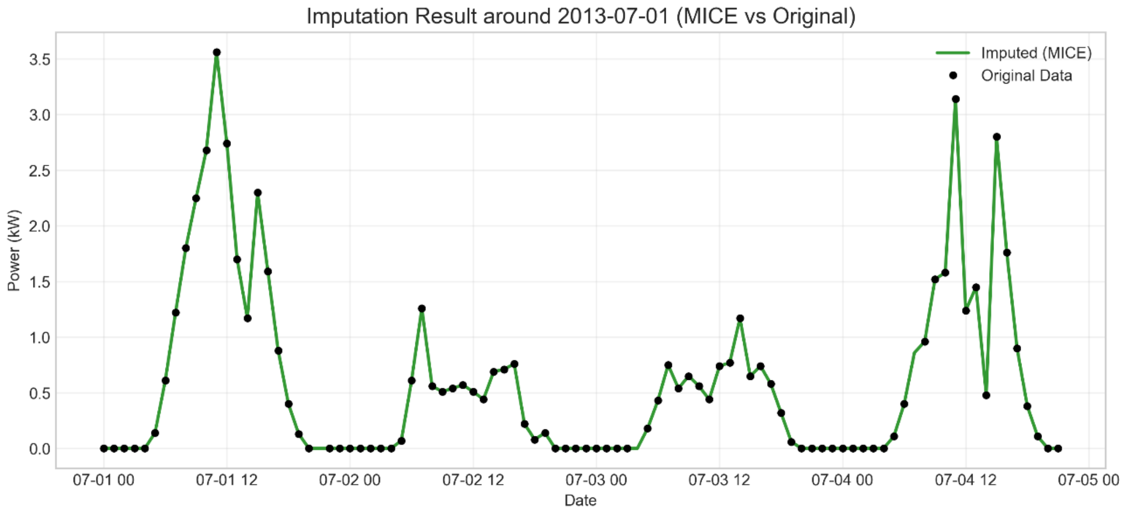


Figure 11: Comparision previous data vs MICE

With that, the plot visualizes the reconstruction of a missing data ingerval using MICE. The green line in the absence of the black dots, reconstructs the natural diurnal solar cycle. This demonstrates that MICE algorithm learned the correlation of time, temperature and pv generation, synthesizing realistic values during outages.

MICE is the superior technique It restored the full row count while preserving the statical variance of the original data, which grant the realistic climatic data been capted.

Advanced Forensic Cleaning

We went beyond the assignment's basic data inputting and used a Scientific Cleaning Pipeline on the full dataset, which included pv_total, demand, and price. The first check suggested the data was good. Still, we took these steps to keep the HEMS system strong if sensor issues or unusual situations arise later.

Physics-based safety mechanisms

To assure that the optimization algorithm never receives invalid data, we hard-coded tow physical constraints:

- The **night mask protocol** that is the absolute inability of produce solar generation at night to prevent any sensor confusing the algorithm of the battery, enforcing a rule where all pv generation is strictly set to 0.0kW between 21:00 and 5:00 (it could even covered more range of time, but this is a small range to not surpass the limits)
- Non-negative constraint, that ensures that no future sensor error could introduce negative generation.

Temporal Reindexing

The study reveals gaps in the timestamp index, and some models require a perfect time frequency, so we reindexed the entire dataset to a strict hourly frequency. The newly created gaps from this reindexing were filled using the MICE method described previously, ensuring a realistic 24-hour cycle without losing synchronization with the calendar.

To be more specific, there only left one more hour and is the last hour of the year 2013.

Smoothing Temperature sensor noise

There was applied a moving average filter to the raw temperature to remove high-frequency sensor noice by ensuring the temperature feature to reflect stable, long-term climate trends and provide more robust and reliable predicator for the forecasting model.

Conclusion

As this last cleaning process didn't really change anything but some noise features or 1 only value in the data points, I am going to write here the changes by hand and to support the changes, the changes are still visible in the files made by the code:

Metric	Before Cleaning	After Scientific Refinement	Action
Time Gaps	1 missing hour (2013-12-31 23:00:00)	0 gaps (Full range recovered)	Temporal Reindexing
Solar Noise	164 sensor noise points	0 noise points in 21:00-05:00 range	Night Mask Protocol
Temperature	Noisy (Jitter)	Smoothed	Feature Optimization
PV Total	Inconsistent with modules	Reconstructed (2,743 rows updated)	Data Consistency Check

Feature Engineering

Following the data cleaning, the next step in the Data Science Lifecycle is Feature Engineering. Here, we aim to turn raw timestamps and weather sensor info into useful predictors. These predictors should catch the non-linear aspects, seasonal changes, and inertia found in home energy use.

This process involves three different phases:

- Statical diagnosis of the target variable
- Construction of external features that helps the future model performance
- Dimensionality reduction based on correlation and Variance inflation Factor analysis.

To keep the changes of the data cleaning phase, we created a new csv that is going to be used (train_256063_cleaned2.csv) as it has no more missing data.

Statistical Analysis

In this part, we analyse the statistical properties of the target variables.

For this mandatory task we have to do a Distribution Diagnosis Visual inspection of the demand probability density function and a Q-Q plot that show deviation from a normal distribution.

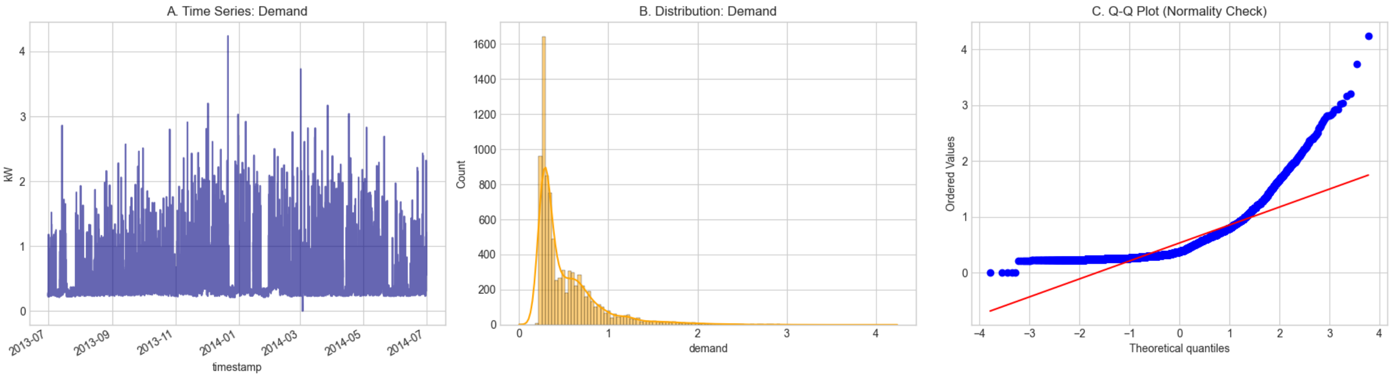


Figure 12: Statistical Analysis of Electricity Demand
A) Hourly time series
B) Histogram showing right-skewness
C) QQ Plot confirming non-normality

The statistical metrics confirm the visual assessment:

- The skewness value (2.4840) shows the distribution leans heavily to the right. This means consumption is usually low, but there are occasional spikes when demand is high, like when people are cooking or turning on their HVAC systems.
- The kurtosis value (8.8004) is high, meaning the distribution has fat tails. This suggests extreme demand events happen more often than one would expect if the data followed a normal distribution.

The objective of this HEMS project is financial optimization, so transforming the target would require inverse transformation during the optimization phase, which could introduce errors in cost of calculations. Its because that that we retain the raw units of kW for the target variable.

To understand the correlation of the features with the demand, we created a ranking of correlation:

Feature	Pearson Correlation
Hour	0.2837
Price	0.2279
Pv_mod2	0.1183
Pv_mod3	0.1183
Pv_mod1	0.1183
Pv	0.1143
Direct_normal_irradiance	0.0965
Diffuse_radiation	0.0943
Direct_radiation	0.0626
Heating_degree	0.0539

Advanced Feature Construction

To capture the complexity of the temporal dynamics for energy usage, we worked with three categories of features aside the raw data.

Cyclical time encoding

Raw integer for hours or moth presents a mathematical discontinuity that comprehend the last vale (23) far from the first (0). To resolve this we mapped time variables onto unit circle using trigonometric transformations.

This allows the model to understand the continuity between 23:00 and 00:00.

Inertia and Autocorrelation

The energy demand has high inertia, the consumption at time t is heavily dependent on the previous consumption and the consumption generated 24 hours before. With this information we generated the next features:

- Lag-1h: Captures continuity
- Lag-24h: Captures seasonality
- Rolling Mean (24h): Captures the day before tred

Physics-based interactions

As the relationship of some immediate features, we made some correlations of them to make features that has more predictable potential.

- Heating/Cooling degrees; Derived from temperature to isolate HVAC requirements
- Price-hour interaction: A feature created to identify the price sensitivity during peak hours, to make easy the detection of demand response behaviors.

Feature Ranking and Selection

Evaluating the predictive power of the engineered features using Pearson correlation coefficients against the variable demand, we have the next results:

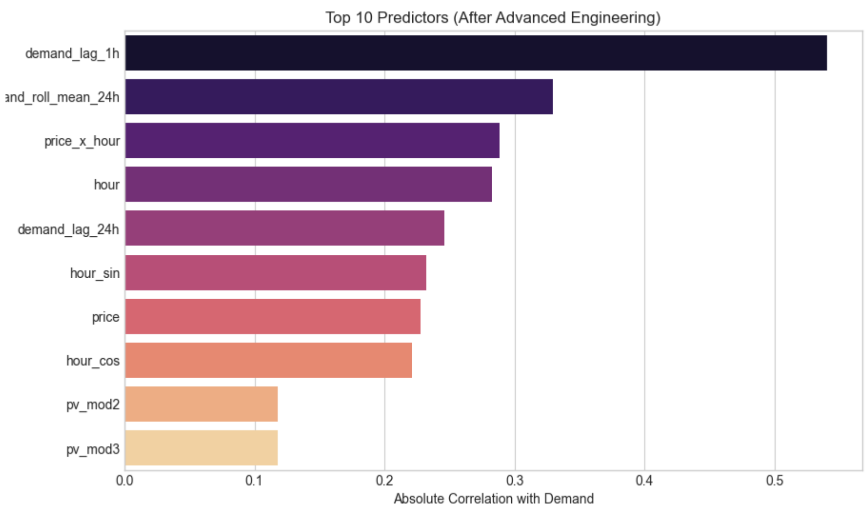


Figure 13: Top 12 features ranked by absolute pearson correlation with Demand

This again shows how the demand is strongly correlated with the previous demand in the system. Then we also confirm that the cyclical demand is stablished strongly to show that the certain period in the day implies more demand. Finalizing with the third best as the price_x_hour, that is a variable created to show the correlation between the hour and price, that is consistent because the benefits of the market by increasing the prices when it is supposed to be more demand, fitting it with also the time of the data.

Solar radiation and pv have poor rankings since they mainly drive energy supply, not demand. Their association with electricity use is indirect and non-linear, which leads to low correlation scores.

Dimensionality reduction

To finish this part, the csv is going to be reduced to prevent the course of dimensionality of some ML models and to reduce redundant and noise features.

The feature pruning affected the next features:

- Pv_mod1, pv_mod2 and pv_mod3: All those features were collinear with the total pv generation variable created before

Linear dependencies: Features like heating_degree were removed in favor of temp_sqared to keep the dataset "lean" while reaaining non-linear signals.

This phase is finished with the file train_256063_features_lean.csv with 29 optimized features.

Time series analysis

To understand what shapes electricity use in a home, we broke down the demand data. This let us separate regular patterns, like daily routines and seasonal weather changes, from random variations.

Classical Additive decomposition

The time series was modelled using a classical additive decomposition defined as:

$$yt = Tt + St + Rt$$

Where:

- Tt is the trend-cycle, getting the long term progression of the series
- St is the Seasonal component, representing the repeating daily cycle
- Rt denotates the Resudials, the random noise remaining after extracting the pattern

To understand the hidden structure in household electricity use, we looked at each part of the demand variable yt. This let us separate expected patterns (like daily routines and climate changes each season) from random variation. We chose the additive model instead of the multiplicative model. When we viewed the data, the size of daily changes seemed steady all year and didn't grow with the trend.

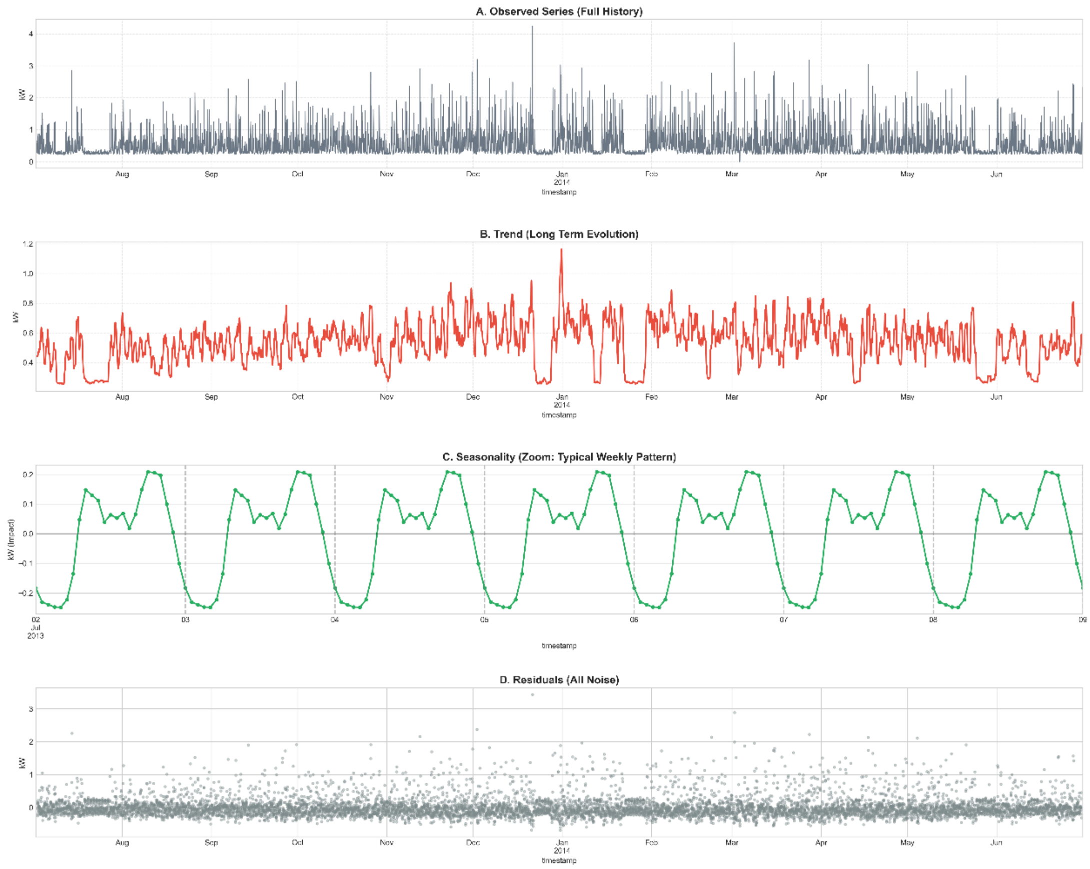


Figure 14: The four panel decomposition plot
A) Observed data
B) Trend (smooth long-term movement)
C) Seasonality (zoomed 1-week view showing the 24h cycle)
D Resuduals (Stochastic noise)

The March peak probably lines up with a changeover time when the temperature swings a lot. At these times, homes might need a lot of heat when it gets cold at night, but not much cooling when it warms up during the day. This creates a big difference in energy use between day and night, which makes the seasonal signal stronger.

Typical demand profiles

The standard behavior of the household was obtained by the derived typical demand profiles by aggregating the time series.

The profiles were created by the hourly arithmetic mean method

- The dataset was separated into different subsets based on day type (weekday and weekend) and season (summer and winter)
- For each subset, is calculated the mean demand of every hour in a day. This creates the average of a specific day and eliminate the anomalies to reveal the underlying routine.

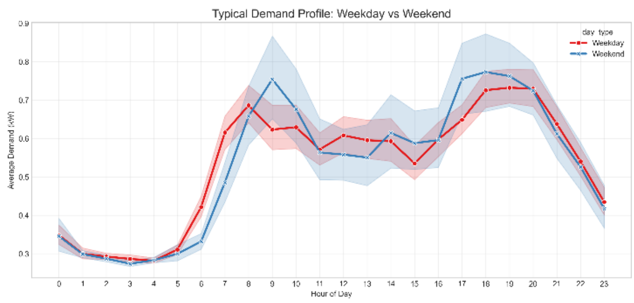


Figure 15: Comparison Weekday and Weekend demand

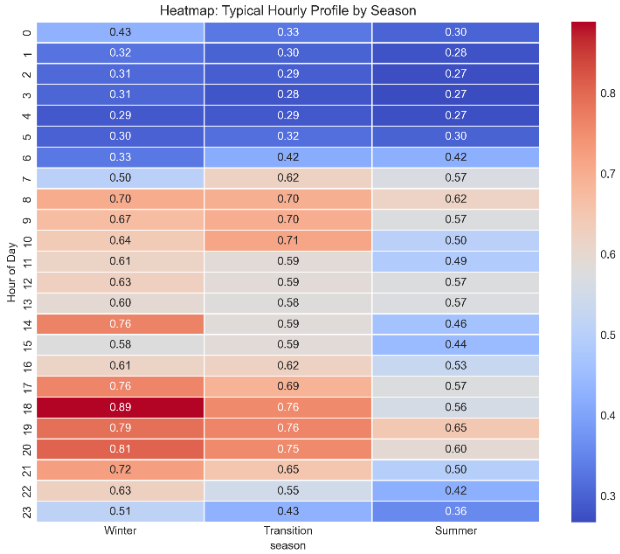


Figure 16: Comparison seasonal demand

Predictive Modelling

Statistical Modelling (ARIMA Family)

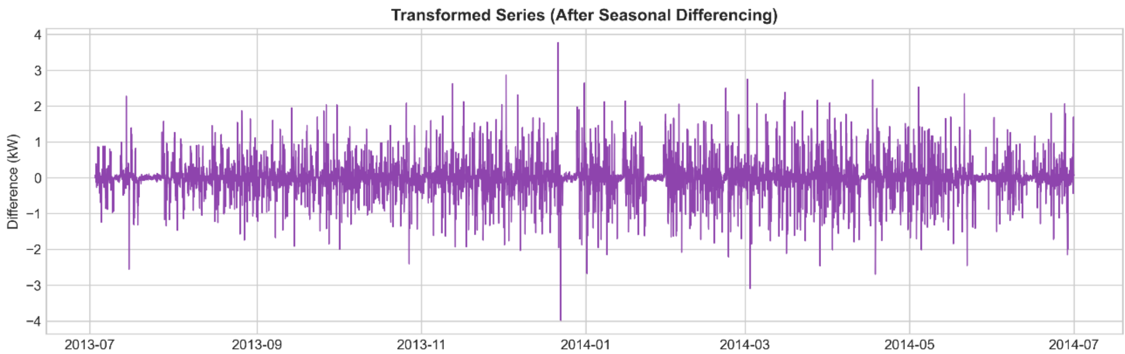
To create a basic prediction, we used Auto-Regressive Integrated Moving Average (ARIMA) models. These models forecast future demand by looking at the series' past patterns and correcting errors.

Stationarity and Differencing

Statistical algorithms need data with a constant average and variance.

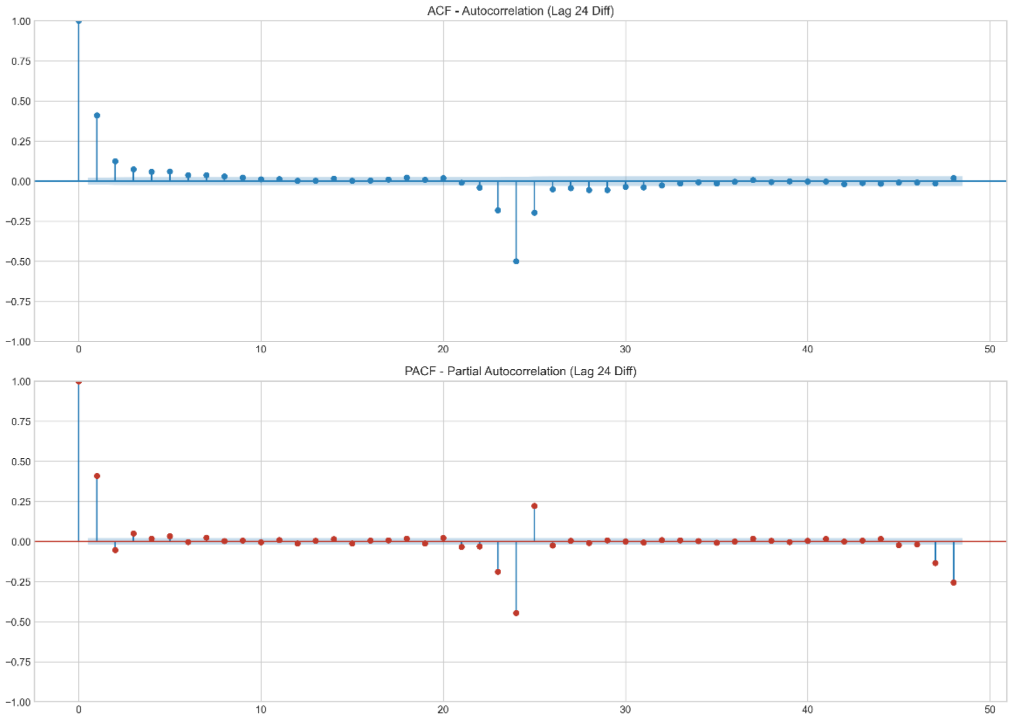
The Augmented Dickey-Fuller (ADF) Test was used on the raw demand series (yt). The ADF Statistic was -11.76 with a p-value = 0.0000 (< 0.05).

The raw series is stationary enough for modelling without first-order differencing (d=0). A seasonally differenced series was tested because of the strong 24-hour cycle seen in the ACF plots, and this confirmed a stronger stationary state.



Autocorrelation Structure (ACF and PACF)

To determine the model parameters (p,q), we analyzed the orrelograms of the differenced series:



- ACF: significant peak the next 24 hours and still some persistence in the 48 hours. This strongly suggest the need for seasonal moving average. Also obvious persistence the next hour
- PACF: Sharp cut-off after lag 2, indicating that the immediate short term demand dependes mostly on the previous 1-2 hours.

Model selection and training

Based on the plots generated, we chose two models to train them using a smart selection process to reduce the Akaike Information Criterion (AIC).

Non-Seasonal ARIMA (Aggressive) – ARIMA(4, 1, 4)

A complex model tries to capture the daily wave pattern through autoregression, without knowing about the 24-hour cycle.

Seasonal SARIMA (Expert) – SARIMA(1, 0, 1)x(0, 1, 1, 24)

The model uses both a Seasonal MA term (Q=1) and Seasonal Differencing (D=1) at a 24-period lag. This setup lets the model use data from the same time yesterday to help make its predictions.

Performance Evaluation (walk-forward validation)

The simulation was performed as a walk-forward validation on the final weeks of the dataset (168 horas). The model predicted 24-hours ahead, received the real data, updated their internal state and predicted the next day.

Model	Configuration	NRMSE (Out-of-Sample)	Performance Notes
Model A	ARIMA(4, 1, 4)	0.1671 (16.7%)	Struggles to anticipate peak timing; reactive rather than predictive.
Model B	SARIMA(1,0,1)(0,1,1)24	0.1495 (14.9%)	Winner. Superior peak alignment due to seasonal awareness.

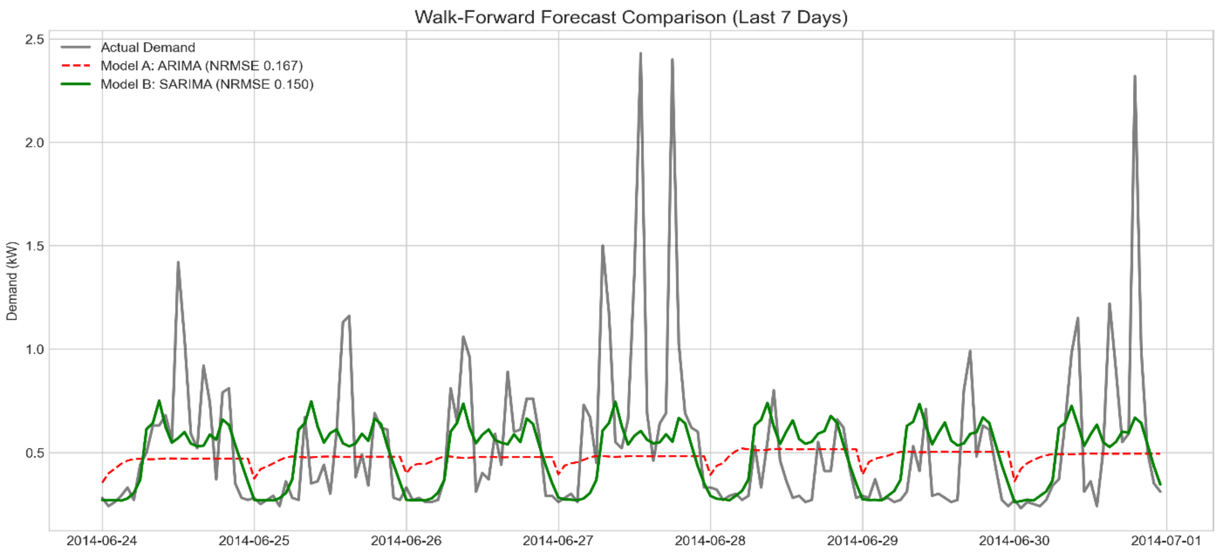


Figure 17: Walk-forward forecast comparison
Green – Sarima
Red – Arima
Gray – Actual demand

A comparison shows that the standard ARIMA model reacts to changes and often misses the exact timing of daily demand peaks. The SARIMA model, on the other hand, uses seasonal differencing (D=1, S=24) to recall the 24-hour cycle of the household, which helps it predict morning and evening increases more accurately. Because of this, SARIMA had a lower NRMSE of 14.95%, which is better than the non-seasonal ARIMA's 16.71%. So, SARIMA was chosen as the statistical baseline for comparisons with the Machine Learning models.

Machine Learning Approach

Model selection: eXtreme Gradient Boosting (XGBoost)

Statistical models like SARIMA are good at spotting seasonal changes, but they have problems with complicated links and adding outside factors. So, we picked XGBoost (eXtreme Gradient Boosting) as our go-to machine learning method.

XGBoost handles non-linearity by using decision trees, letting it model thresholds, like HVAC activation above 25°C. It uses external features such as temperature, price, and cyclical time to improve predictions, unlike SARIMA, which depends on past target values. Regularization parameters prevent overfitting on small datasets (one year).

Hyperparameter Optimization and Rationale

For objective model setup, we skipped manual adjustments and used RandomizedSearchCV with Time Series Cross-Validation. This approach checks different hyperparameter settings while keeping the data's time order intact.

Hyperparameter	Selected Value	Rationale / Impact
n_estimators	300	The number of boosting rounds. A moderate value (300) provides sufficient learning capacity without the diminishing returns and overfitting risk of higher values (e.g., 1000).
learning_rate	0.01	A low learning rate (shrinkage) ensures the model learns slowly and robustly, preventing it from getting stuck in local minima during gradient descent.
max_depth	5	Constrains the depth of each tree. A depth of 5 captures interactions (e.g., Hour/times Price) without memorizing noise in the training data.
subsample	0.7	Trains each tree on only 70% of the data, introducing randomness that reduces variance and improves generalization.

Comparative Evaluation: XGBoost vs SARIMA

Both models underwent the same Walk-Forward Validation on the last week of the data set (168 hours). Each model was retrained and updated with real data every day.

Quantitative results:

Model	NRMSE (Walk-Forward)	Relative Performance
Statistical (SARIMA)	0.1497 (14.97%)	Baseline
ML (XGBoost)	0.1374 (13.74%)	Winner (-8.2% Error)

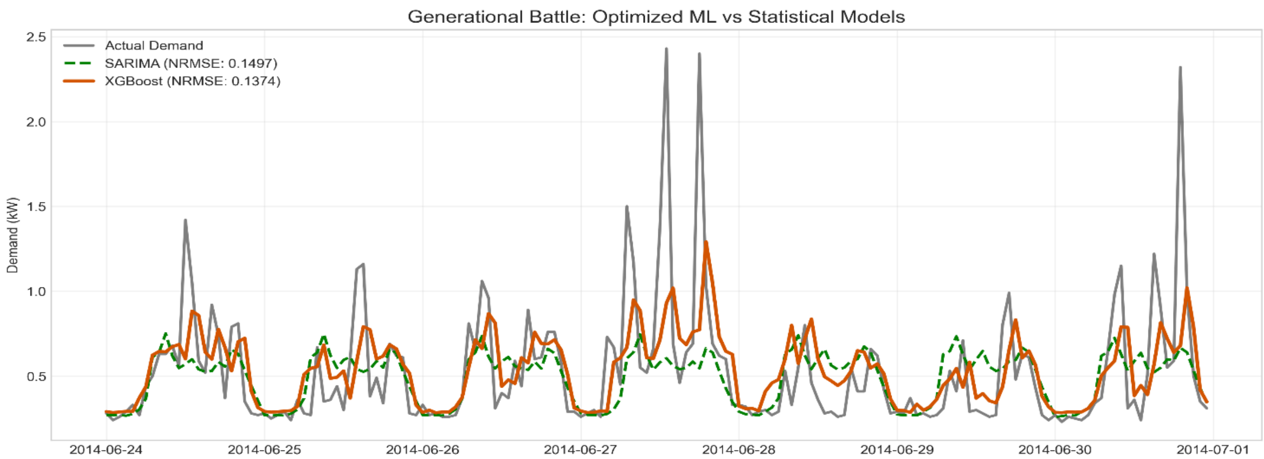


Figure 18: ML vs Statistical Modeling
Orange: ML
Green: Sarima
Gray: Actual demand

The XGBoost model achieved an 8.2% less error relative to the statistical model. Feature analysis also helps to confirm that wile lag features provide the inertia baseline, the Cyclical Time and Temperature features allow XGBoost to correct the forecast during non-standard peaks. Consequently, XGBoost performs generally better.

Forecasting Pipeline

Pipeline Architecture and Methodology

To shift from experimental models to a production system for the Home Energy Management System (HEMS), we built a reproducible Rolling Forecast Pipeline. The pipeline handles the data flow automatically, from raw input to hourly demand predictions ($t_{+1} \dots t_{+24}$), making sure that the optimization algorithm (Task 11) gets the newest, most precise info.

Instead of predicting the full seven days horizon at ones, we adopted a 24-hours rolling window approach. This simulates a realistic HEMS operation.

- The system predicts the demand for the day
- The real data for the day becomes available
- The model history is updated, internal states are refreshed, and system predicts the next day

The validation of our model is based on two naïve heuristics:

- Seasonal Naïve: Assumes that the next day will be as the same day one week ago.
- Daily Average: Use the mean consumption of the previous 24 hours to create a flat line in which the predictions are done.

Forecast Generation and comparison

For the week of July 1st to 7th, we made predictions using the best Statistical (SARIMA) and Machine Learning (XGBoost) models from our earlier work

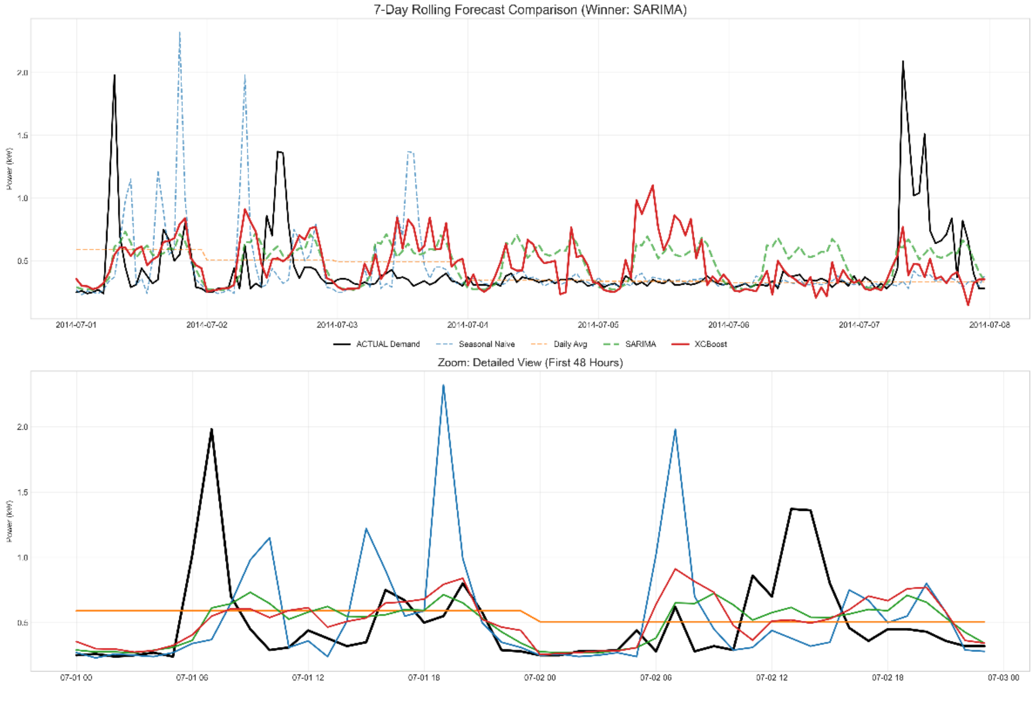


Figure 19: 7-day forecast comparison

Quantitative Evaluation

To judge how well the models predicted values for the week of July 1st to 7th, I found the Normalized Root Mean Squared Error (NRMSE) and Mean Absolute Error (MAE) for each one.

Model	Type	NRMSE (Lower is Better)	MAE (Lower is Better)
SARIMA	Statistical (Seasonal)	0.1548 (Winner)	0.1957
Daily Average	Baseline (Naive)	0.1588	0.1584 (Winner)
XGBoost	Machine Learning	0.1768	0.2140
Seasonal Naive	Baseline (Naive)	0.2073	0.1845

Conclusion

We are going to forget about the quantity evaluation due to a main problem, the week of July 1^{st}-7^{th} took place one of the most important events in Estonia, where I assume that the HEMS takes place.

This festival took place between 5 and 6 of July, explaining the absence of demand on the intermediate days. Because these 4 days the demand is so scarce, the daily average fits well, so there are no peaks in almost the whole day.

Because of that, we have to forget about the number and look for the independent days. The daily average its enough if there isn't any peak, but it's a poor method to keep track of the demand.

The Seasonal Naïve model yielded the worst performance since the test week coincided with the Song Festival (an anomaly), the model blindly copied the "normal" high activity that adapts after several days of error.

The Robustness of SARIMA, despite the anomaly caused by the Song Festival, SARIMA demonstrated superior robustness (NRMSE 0.1548). Unlike XGBoost, which over-reacted to weather conditions by predicting non-existent peaks, SARIMA adapted to the low-occupancy "base load" while preserving the 24-hour cycle. Its conservative, smoothed forecasting style avoids dangerous overestimation, making it the most reliable engine for the battery optimization in Task 11.

The "Hallucinations" of XGBoost suffered from contextual overfitting, falsely correlating warm weather with high demand despite the home being unoccupied during the festival. The model lacked a signal to override this weather sensitivity; incorporating a custom holiday calendar feature or special days calendar feature dependant on the house would likely rectify this by allowing the algorithm to explicitly recognize such non-standard low-occupancy periods.

Models with input: Impact Analysis

The value of information

To assess the actual importance of external input variables, we did an ablation study. This study compared a baseline model without specific data to a model that accounts for context. Both models used the same gradient boosting algorithm to isolate the impact of the data.

- Baseline Model: This model relies entirely on system inertia (autoregression), assuming "tomorrow will be like yesterday".
- Context-aware model: This model is bringing the engineered features from task 5, including temperature, cyclical time and price.

The first training stage experienced a silent error. An audit showed that temperature data was disconnected because column names didn't match. To fix this, a correction pipeline was put in place to standardize feature names, which allowed the model to process the weather data.

Quantitative evaluation

The models were evaluated using a walk-forward validation on the final test week, mirroring the production environment.

Model Variant	MAE (kW)	NRMSE	Improvement (MAE)
1. Baseline (AR Only)	0.2145	0.1246	0.0% (Reference)
2. Weekly Extended	0.1940	0.1205	+9.56%
3. Full Context (Weather + Price)	0.1910	0.1200	+10.97% (Winner)

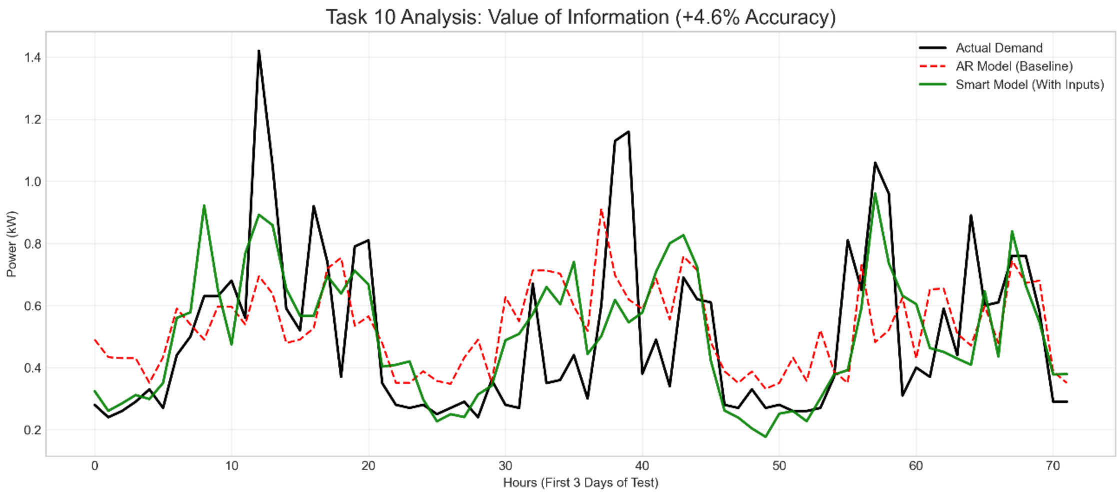


Figure 20: Forecast Analysis
Red: Baseline
Green: Smart Model
Black: Actual Deman

Visual Analysis of forecast dynamic

The impact of the exogenous features is visually evident in the time-series comparison

The baseline model exhibits a characteristic phase lag that could predict the peak after it has already started, reacting only to the previous day's inertia.

The smart model aligns significantly better with the onset of demand ramps. By utilizing the hour and temperature features, the model anticipates human activity and heating needs before they appear in the lag history.

The inclusion of the weather data allowed the smart model to better estimate the magnitude of the peaks in the evenings, where the baseline tended to under-predict extreme values during colder intervals.

Conclusion

The inclusion of exogenous variable yielded a 10%97 reduction of MAE. The analysis shows that what people want in housing depends not just on past trends but changes with the current environment. So, we chose the multivariate Full Context model for the last stage.

Optimization

After the demand forecast is made, the system uses a Linear Programming (LP) solver to find the best energy flow schedule. The model aims to lower the total electricity cost over a 24-hour period (T=24).

The objective function minimizes the cost of energy imported form the grid while maximizing revenue form exports.

The optimization is subject to the following physical restrictions:

$$Power\ balance: P_{grid} + P_{pv} + P_{bat\_dis} - P_{bat\_ch} = P_{demand}$$

$$SOC(t) = SOC(t+1) + \left(p_{bat_{ch}} * n - \frac{P_{bat_{dis}}}{n}\right) * \ \Delta t$$

Capacity limits:

$$0\ \leq P_{grid}, P_{bat} \leq 5kW$$

$$0 \leq SOC(t) \leq 10kW$$

For the implementation was used the scipy.optimize.linprog library via the HiGHS method

In the initial state, the charge is set at the 50% in the code

Methodology

The solution consists of two integrated modules, a demand forecaster and a linear optimization engine.

Demand Forecasting

Before optimization, the system must predict the household consumption.

- Model: HistGradientBoostingRegressor
- Features: Based on the "Full Context" model established in previous tasks, utilizing autoregression of 24 hours form forecast.csv, cyclical time using the sine and cosine representation and the weather.
- The result is that the model successfully fenerated a 24-hour load profile to serve as the baseline constraint for the optimizer.

Optimization Engine

To figure out the best control plan, a Linear Programming (LP) setup was done using the scipy.optimize.linprog tool. Decision Variables: The solver handles a state vector x of size 120 (5 variables * 24 time steps).

- P_{grid_in} (power imported)
- P_{grid_out} (power exported)
- P_{bat_ch} (battery charging)
- P_{bat_dis} (battery discharging)
- E_{bat} (battery energy state / SOC)

Restrictions:

- Power Balance: At any time, supply must be equal demand

$$P_{grid\_in} + P_{bat\_dis} + P_{PV} = P_{demand} + P_{grid\_out} + P_{bat\_ch}$$

- Battery Dynamics: Energy stored at time relates to time

$$E_t = E_{t-1} + \left(P_{bat\_ch} * \Delta t\right) - \left(P_{bar\_dis} * \Delta t\right)$$

- Physical Bounds

$$0 \leq P \leq 5 \; (for\ all\ power\ flows)$$
$$0 \leq E \leq 10 \; (for\ battery\ capacity)$$

Scenario Analysis and Results

The EMS was tested under two distinct weather conditions to evaluate robustness and strategy adaptation.

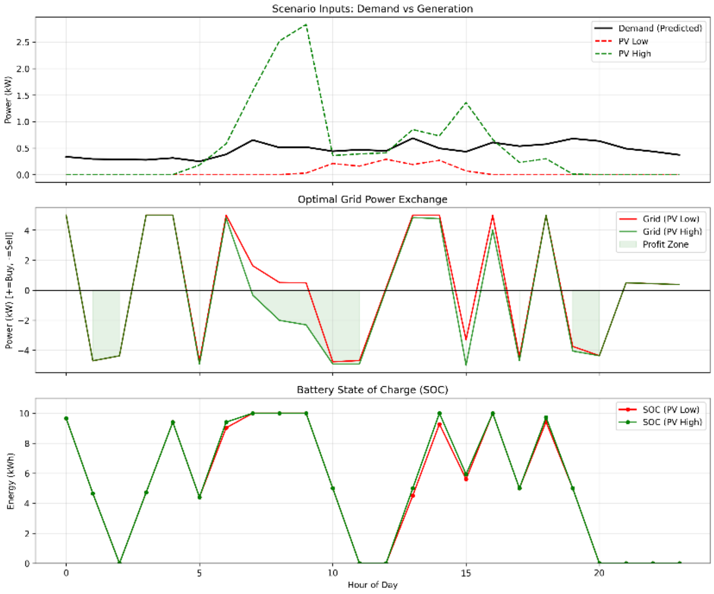


Figure 21: Scenarios Analysis

Scenario A: PV Low (Cloudy/winter)

The solar generation was minimal and insufficient to cover household load. The optimizer utilized price arbitrage. The battery charged from the grid during cheaper night hours and discharged during peak pricing to reduce costs.

The financial result: -0.20€.

Scenario B: PV High (Sunny)

The solar irradiance exceeded demand during midday.

The optimizer prioritized self-consumption ad export. The battery absorbed excess PV to prevent the waste of the extra energy, and the system maximized exports to the grid when prices were favorable

The financial result: 1.21€.

Conclusion

The system focused on using energy and exporting it. The battery stored extra solar energy to stop waste. Task 11 shows that a smart energy system using data can cut energy costs. By mixing good machine learning predictions with simple math, the system follows hardware rules while using renewable energy and market prices. The difference in results (-0.20 € vs -1.21 €) shows how important solar energy is for making money, and battery control keeps things running well even when it's cloudy.