Master of Science in Geoenergy Engineering

Master Thesis

Automatic Selection of Predictive Models for Building Energy Demand

Prof. Alfonso Capozzoli

Prof. Hussain Syed Kazmi

PhD Candidate. Attila Bálint

Candidate:

Yeganeh Salar Babakhani

March 2025

ABSTRACT

Buildings account for more than one-third of total energy demand in many countries worldwide, positioning them as one of the major contributors to greenhouse gas emissions. Effective prediction of building energy demand is essential for optimizing electricity consumption and mitigating power shortages or waste. The importance of this task is increasing due to two significant transitions: the electrification of heating and transportation and the growing dependence on renewable energy sources. The former refers to the widespread adoption of electric vehicles and electric heating systems, such as heat pumps. The latter involves the expansion of renewable energy generation, which introduces challenges in balancing supply and demand, as electricity from sources like wind and solar is inherently intermittent and not always available when consumption peaks.

The principal objective of this project, building further on previous work at KU Leuven University, Belgium, was to develop machine learning models to create building energy demand forecasts for the future (day-ahead time scale). It therefore focuses on establishing a complete procedure for energy demand prediction including preprocessing, forecasting, data analysis, time series feature extraction and model recommendation. A crucial task in time-series forecasting is the identification of the most suitable forecasting model. The challenges associated with energy demand forecasting arise from a wide range of predictive algorithms and models can be adapted through minor or significant modifications which is time intensive. Additionally, the diversity of buildings, each with distinct energy consumption patterns, further complicates the forecasting process. Consequently, the project first constructs a comprehensive dataset comprising buildings with diverse energy demand patterns to evaluate the performance of various forecasting models across these buildings. Then, based on this analysis, it designs a recommender system to identify and suggest the most suitable forecasting models for a newly encountered building. To implement this recommender system effectively, time series feature extraction is a crucial step. Feature extraction facilitates the summarization of long time-series data into a lower-dimensional set of key values or statistics, thereby reducing data complexity while preserving essential patterns and information.

Finally, based on the developed dataset comprising five distinct building energy demand profiles along with their extracted features and the performance evaluation of over thirty forecasting models on these buildings, the recommender system is designed to identify the most suitable forecasting models. When the features of a previously unseen building are input into the system, it recommends the forecasting models that best align with the specific characteristics of that building. The evaluation results indicate that the top-performing models outperformed the baseline methods by approximately 12%, demonstrating the advantage of advanced forecasting techniques and the recommender system achieved an accuracy of approximately 0.05 in terms of MAE, (the average discrepancy between the predicted and actual values), highlighting its capability to provide reliable model recommendations based on the extracted building features.

Keywords: Building Energy Demand, Forecasting Model, Recommender System, Time-series Data, Machine learning.

ACKNOWLEDGMENTS

I want to convey my deep appreciation to Professor Alfonso Capozzoli, my supervisor, for his exceptional guidance and leadership throughout my master's thesis. His expertise and unwavering support were crucial in shaping the course of my research and nurturing my academic development.

Sincere gratitude is extended to Professor Hussain Syed Kazmi, my co-supervisor at KU Leuven University in Belgium, for his outstanding support, encouragement, and mentorship. Beyond academics, his assistance created a welcoming environment, enriching my experience. Prof. Kazmi's unwavering support significantly enhanced my overall well-being during my stay.

I also express sincere thanks to PhD Candidate Attila Balint, my co-supervisor, for invaluable guidance and assistance during my academic journey. His insights played a crucial role in shaping my research approach, and his generous support had a substantial impact on my work's development. Together with Professor Kazmi, their combined support not only contributed significantly to my academic growth but also played a vital role in shaping my personal experience in Belgium.

Finally, a big thank you to my family and friends in Iran and Italy. I couldn't have done any of this without their constant support. Their encouragement means the world to me, and I'm really grateful for everything they've done.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The operations of buildings account for 30% of global final energy consumption and 26% of global energy-related emissions (8% being direct emissions in buildings and 18% indirect emissions from the production of electricity and heat used in buildings). (*Tracking Clean Energy Progress 2023*, n.d.)

With increasing urbanization, the concentration of energy demand in buildings will further rise in the coming years. Thus, drastic steps will be required to ensure societal decarbonization. However, despite its importance, inefficient energy use in buildings remains a common challenge and frequently leads to unnecessarily high costs and greenhouse gas emissions. These challenges can be addressed, at least in part, through accurate energy demand prediction.

Several studies have been conducted on forecasting. One paper proposes a methodology for building-level energy forecasting by clustering weather data into distinct operational modes and comparing a baseline energy model with a cluster-based model, with the latter demonstrating superior performance (Desiree Arias-Requejo et al., n.d.). Another study utilized the ARIMA model to forecast electricity demand, leveraging historical data while accounting for seasonality and trends (Dr.S.Meenakshi et al., n.d.). Additionally, a study examines ten fundamental research questions that form the foundation of energy demand forecasting at both building and urban scales. The initial question highlights key applications of energy demand forecasting. Subsequent questions, investigate critical factors influencing the essential components of the forecasting process (Hussain Kazmi et al., n.d.-a). Furthermore, some studies have explored the selection of forecasting models. One particular study presents a general framework for forecast model selection utilizing meta-learning. This framework is assessed using time series data from the M1 and M3 competitions, covering annual, quarterly, and monthly models. A random forest is employed to determine the suitable forecasting method based on only two time series characteristics: the strength of seasonality and the strength of trend (Thiyanga S Talagala et al., n.d.-a). Moreover, many studies have investigated the time series features extraction. This study evaluates the effectiveness of various domain-informed and domain-agnostic features using an extensive meta-analysis dataset of building energy demand (Ada Canaydin et al., n.d.-a).

Forecasting is a key activity in any field for efficient operation. The rapid advances in computing technologies have enabled businesses to keep track of large numbers of time series. Hence, it is becoming increasingly common to have to regularly forecast many millions of time series. For example, large scale businesses may be interested in forecasting sales, cost, and demand for their thousands of products across various locations, warehouses, etc. Technology companies such as Google collect many millions of daily time series such as web-click logs, web search counts, queries,

revenues, number of users for different services, etc. Forecasting the energy demand of buildings benefits in grid, transition, renewable energy resources, greenhouse gas emission, etc.

There exist a variety of machine learning models capable of predicting the energy demand of buildings. These models are categorized into five distinct groups, as outlined below (*Five Machine Learning Types to Know*, n.d.):

1. Supervised machine learning: Supervised machine learning trains models on labelled datasets to predict target variables and is widely applied in risk assessment, image recognition, predictive analytics, and fraud detection. It comprises several algorithm types: regression algorithms for continuous value prediction (e.g., linear regression, random forest), classification algorithms for categorical labelling (e.g., logistic regression, SVMs), Naïve Bayes classifiers for large-scale classification, neural networks for complex pattern recognition, and random forest algorithms for enhanced predictive accuracy through decision tree aggregation.
2. Unsupervised machine learning: Unsupervised machine learning derives insights from unlabelled data, supporting exploratory analysis, pattern recognition, and predictive modelling. Key methods include clustering algorithms (e.g., K-means for market segmentation, hierarchical clustering for iterative grouping, and probabilistic clustering for density estimation) and association algorithms for identifying relationships within large datasets. Techniques such as principal component analysis (PCA) facilitate dimensionality reduction, while unsupervised models underpin recommendation systems and anomaly detection.
3. Self-supervised machine learning: Self-supervised learning (SSL) allows models to autonomously learn from unlabelled data, eliminating the need for extensive annotated datasets. Also referred to as predictive or pretext learning, SSL algorithms infer one portion of the input from another, effectively generating labels and converting unsupervised tasks into supervised ones. This approach is particularly advantageous in domains such as computer vision and natural language processing, where the demand for large-scale labelled data can be substantial and often impractical.
4. Reinforcement learning: Reinforcement learning (RL), also known as reinforcement learning from human feedback (RLHF), is a dynamic programming approach that optimizes algorithms through a reward-based system. In this framework, an agent interacts with a defined environment, taking actions to achieve a specific objective. Based on a predefined metric, the agent receives rewards for desirable actions and penalties for suboptimal ones, reinforcing

effective strategies through iterative learning. RL is widely applied in video game development and is commonly used to train robots in replicating human tasks.

5. Semi-supervised learning: The fifth category of machine learning represents a hybrid approach, integrating elements of both supervised and unsupervised learning. Semi-supervised learning algorithms are trained using a small labelled dataset alongside a significantly larger unlabelled dataset, where the labelled data serves as a guide for learning patterns within the unlabelled data. These models often employ unsupervised learning to identify data clusters, followed by supervised learning to assign labels to those clusters. An example of semi-supervised learning is Generative Adversarial Networks (GANs), a deep learning framework that generates unlabelled data by training two competing neural networks.

This project employs several supervised machine learning models to forecast energy demand. The models utilized include NaiveSeasonal, LinearRegression, LightGBM, RandomForest, and Prophet. These models are typically trained on historical data, learning seasonal patterns (such as daily and weekly trends) and leveraging these patterns to predict future demand. Despite their simplicity, these models are considered supervised since they rely on labelled training data, which consists of historical energy demand data with corresponding timestamps.

To identify the most suitable forecasting models for an unknown building with a distinct energy demand pattern, a recommender system is designed. Generally, recommender systems are employed in various decision-making contexts, such as selecting products to purchase, choosing music to listen to, or determining which online news to read. These systems are widely used across different domains, with well-known applications including playlist generators for video and music streaming services, product recommendations for online retail platforms, and content suggestions for social media and web-based content providers (*Recommender System*, n.d.). In this context, the recommender system is utilized to suggest the most appropriate machine learning models for forecasting the energy demand of an unknown building, based on the features extracted from that building's data.

1.1 Problem Statement

Several studies have been conducted on forecasting energy demand within the building sector using machine learning techniques. These studies generally focus on a particular type of building or concentrate on a single machine learning prediction model.

It is noteworthy that the application of machine learning models is often time-consuming. Consequently, applying multiple models in an effort to identify the best-performing one further

increases the time required. As the volume of data grows, the processing time expands as well. Thus, forecasting the energy demand of a building presents significant time challenges.

Therefore, advancements are needed in this field to eliminate the need for repeatedly applying various forecasting models to new buildings in order to identify the most suitable model, a process that is time-consuming.

1.2 THESIS GOAL

This thesis aims to reduce the time required for forecasting models. Additionally, it seeks to develop a recommender system that automatically suggests the most suitable prediction model based on the features of a new, unknown building. This work is implemented within a Python environment using machine learning algorithms. The models are trained on time series data, and the results are evaluated using appropriate metrics.

Therefore, this study will apply various prediction models to different building electricity demand patterns, storing their performance metrics to create a comprehensive dataset to avoid unnecessary expenditures of time and resources in future analyses.

The availability of an infinite number of forecasting models highlights the need for a recommender system to avoid redundancy. This system analyses the features of a new, unknown building and then suggests the most suitable forecasting models for that specific building.

2 METHODOLOGY

The buildings time series data are obtained from the Fluvius Open Data website which is an online platform provided by Fluvius, the Flemish distribution network operator, offering a wide range of energy-related datasets to the public (*Consumption Profiles of Digital Electricity Meters*, n.d.). The weather data are gained from Open-Meteo which is an open-source weather API offering free access to global weather forecasts (*Historical Weather API*, n.d.).

The raw data are prepared and cleaned to be ready for models. The data is split such that 80% is used for training, while the remaining 20% is reserved for testing. A persistence model is established as a reference, and the performance of various forecasting models applied to the data is compared against this reference and evaluated accordingly. The performance of the models is then assessed using the evaluation metric, relative Mean Absolute Error (rMAE), as detailed in Section 2.1, which outlines its calculation. In the next step, the features of time series data are extracted to be used as indices in recommender system. This system offers the top forecasting models for an unknown building based on building data features. The accuracy of the recommender system is evaluated using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), with the latter assigning greater weight to larger errors.

In this chapter, the methodology and the tools employed are detailed. The chapter is organized into three main subsections. The first subsection presents the software and methods utilized in this research. The second subsection (2.2) provides an explanation of the evaluation metrics, the chosen metric for assessing forecasting performance, and how the metric is calculated. The third subsection (2.3) outlines the workflow and sequence of steps undertaken to achieve the objectives of this study.

2.1 PYTHON AND MACHINE LEARNING

Python is a powerful programming language for data science, scientific computing, and machine learning (*Python*, n.d.). It is highly flexible, easy to learn, and has a vast number of libraries. The key libraries used in this project include NumPy (*NumPy*, n.d.), Pandas (*Pandas*, n.d.), Matplotlib (*Matplotlib*, n.d.), and Darts (*Darts*, n.d.). Python is also a general-purpose programming language commonly used for numerical and engineering computations. Additionally, it has gained increasing popularity in scientific computing and machine learning applications. In recent years, numerous software packages designed for forecasting tasks have been made open-source. These include specialized forecasting libraries developed in widely used programming languages. Many of these

libraries offer high-level interfaces to various underlying algorithms, encompassing linear models, tree-based methods, and neural networks.

Machine Learning (ML) is one of the fastest growing fields and have been booming in the recent years. It is a branch of artificial intelligence that focuses on how to program machines to learn from data. It is an approach to computer science that is primarily used for solving a wide range of problems in science, engineering, business and everyday life. Machine learning algorithms have become ubiquitous in data-driven fields, such as forecasting and natural language processing. These algorithms employ various techniques to identify patterns and make predictions, typically leveraging statistical and mathematical methods. Machine learning has long been a popular area of research, and scientific computing remains a crucial component of a data scientist's toolkit, particularly for managing complex and large datasets. This project will focus on forecasting through machine learning in Python, Jupyter notebook.

2.2 Evaluation Metrics

Before generating forecasts, it is essential to establish a performance metric to assess the accuracy of the predictions. In practice, there are numerous error metrics to choose from. For regression tasks, such as forecasting energy demand or production, the following error metrics are commonly utilized:

- Mean Error (ME)
- Mean Absolute Error (MAE)
- Relative MAE (rMAE)
- Mean Absolute Percentage Error (MAPE)
- Rˆ2
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

Most of these metrics evaluate the accuracy of forecasts by comparing the predicted values with the actual observed data. The distinction between squared and absolute errors is particularly relevant when higher errors need to be given greater or lesser weight. Unlike traditional error metrics, rMAE (relative Mean Absolute Error) is unique in that it compares the MAE of the forecasting model against a benchmark, in this case “NaiveSeasonal” (K=24).

Scikit-learn (*Scikit-Learn*, n.d.), a widely used Python library for machine learning in both industry and academia, provides various pre-implemented error metrics. These metrics can be readily applied in forecasting evaluations, with a particular focus on regression error metrics.

Beyond these standard metrics, distribution-based error assessments are also employed in some cases. These include:

- Tests for normality of residuals, using Q-Q plots (normal probability plots) for visual inspection and the Shapiro-Wilk test for statistical analysis.
- Tests for autocorrelation in residuals, often conducted using the Durbin-Watson statistic.

It is important to note that these error metrics are applicable only to point forecasts (also referred to as deterministic forecasts). When dealing with interval forecasts (or stochastic forecasts), alternative error metrics become more relevant, as they evaluate the full probability distribution of the forecasted variable. These include measures such as reliability, resolution, and uncertainty, which are sometimes aggregated into a single score (Kazmi Hussain & Balint Attila, n.d.).

MAE is a measure of the average magnitude of errors in a set of predictions, without considering their direction (positive or negative). It is the average over all the test samples of the absolute differences between the predicted values and the actual values. MAE provides an idea of how close the predictions are to the actual outcomes.

Mean Absolute Error Formula

$$MAE = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Where:

- n= number of data points in the test set
- y_i= actual value for the i-th data point
- $\hat{y}_i$= predicted value for the i-th data point
- $|y_i - \hat{y}_i|$= absolute error for the i-th data point

A lower MAE value indicates better model performance, as it means the predicted values are closer to the actual values.

In practice, scenario forecasts are often preferred over purely probabilistic forecasts, as they provide concrete trajectories of possible future outcomes, which can be directly integrated into optimization frameworks. However, the focus here remains on rMAE (relative Mean Absolute Error).

Relative Mean Absolute Error Formula

$$\text{rMAE} = \frac{MAE}{Baseline\ MAE}$$

Where:

- Baseline MAE = MAE calculated using a baseline model (often using the mean of the target variable as the prediction for all data points).

rMAE tells you how much the MAE of your model is relative to the MAE of the baseline model. A value of rMAE=1 means the model's performance is equivalent to the baseline model. Thus, value less than 1 means the model performs better than the baseline, and a value greater than 1 means the model performs worse than the baseline.

On the other hand, the evaluation metrics used to assess the accuracy of the recommender system's performance are primarily MAE, followed by RMSE, which assigns greater weight to larger errors. The calculation of MAE has been explained earlier. The computation for RMSE is as follows:

Root Mean Squared Error Formula

$$RMSE = \sqrt{\frac{1}{\text{n}}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where:

- n= number of data points (observations)
- y_i= actual value of the target variable
- $\hat{y}_i$= predicted value from the model
- $(y_i - \hat{y}_i)^2$= squared error (the difference between the actual value and predicted value, squared)

The sum of squared errors is averaged across all data points, and the square root of this average is then computed. Larger errors are given more weight due to the squaring component in the formula. RMSE is always non-negative, with lower values indicating better model performance. Additionally, RMSE maintains unit consistency, as it is expressed in the same units as the target variable (since it is the square root of squared differences).

2.3 IMPLEMENTATION

This project gathered the data and employed four fundamental steps—data processing, forecasting, feature extraction, and the construction of a recommender system—to accomplish its objectives. The following flowchart scheme overall methodological process and the subsections are explaining the manner in detail.

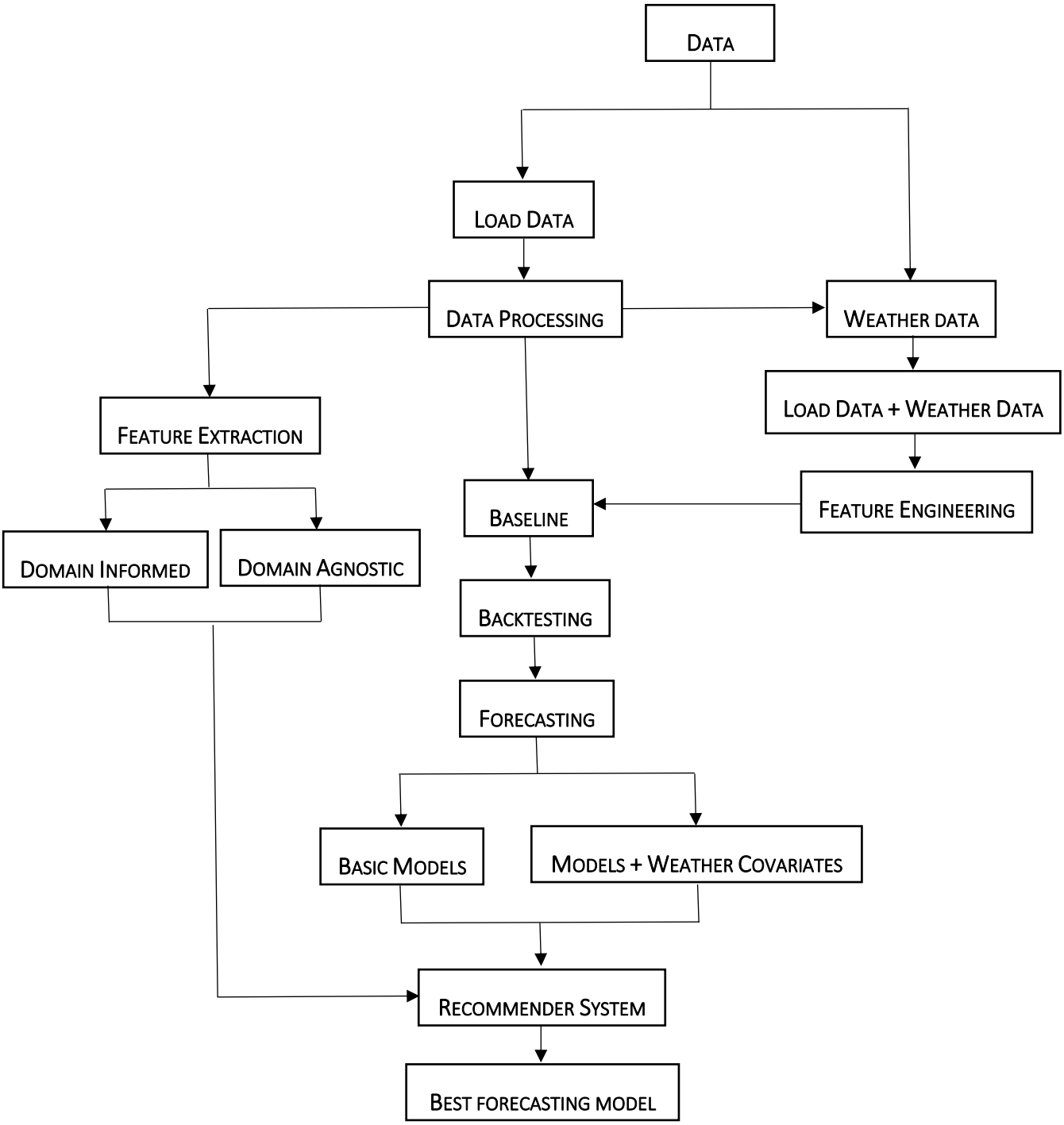


Figure 1 The flowchart of the entire methodological framework, systematically illustrating each step involved in the implementation of the proposed approach

2.3.1 Data Processing

In this step the electricity demand and weather API are cleaned and ordered to align best with the purpose of the project. This dataset contains the individual consumption data of 500 anonymised digital electricity meters. The digital electricity meter for which the grid user has asked Fluvius to read the quarterly values and for which we have all quarterly values available for the entire year (no unused meters). The scope with solar panels is determined if an injection volume was registered in all 12 months. The scope without solar panels are meters for which no injection volume was registered during the entire year. In this dataset, only EANs were retained that belonged to the residential contract category during the entire year. From this scope we extract 500 electricity meters as a sample, divided into the following categories:

- With solar panels (100)
- Without solar panels (100)
- With heat pump and with solar panels (100) (Heat pumps without solar panels are not shown here because they are rare)
- With solar panels and an electric vehicle charging at home (100)
- Without solar panels and with an electric vehicle charging at home (100)

Building Classification Based on Consumption Patterns. More details about the electricity time series data are provided in the table below.

Attribute	Details
Themes	Grid users and their consumption
Timezone	Europe/Brussels
Territory	Flemish region
License	Open data license - FLUVIUS
Publisher	Dataroom Fluvius

Table 1 Electricity time series Data Information

A unique case number is assigned to each category of buildings. This classification groups every hundred buildings (EAN_IDs) with similar consumption patterns into separate cases, ranging from 1 to 5 as follows:

- Case 1: Including photo voltaic panels (PV)
- Case 2: No specific consumption (None)
- Case 3: Including photo voltaic panels and heat pumps (PV+HP)
- Case 4: Including photo voltaic panels and electric vehicles (PV+EV)
- Case 5: including electric vehicles (EV)

The required Libraries (e.g. NumPy, pandas, matplotlib, Darts) and their packages (e.g. darts.metrics, matplotlib.pyplot, json) are imported. The data are loaded and translated into english, all five cases combined into one DataFrame. The datetime column converted to a proper datetime format and the "Net_Volume_Purchase" is calculated as the target of forecasting.

Then the acquisition points processed effectively downsamples to an hourly resolution while maintaining separate time series for each EAN_ID. Since resampling groups data into hourly intervals, we need to specify how to aggregate multiple values within each interval. So the ".mean()" function applied to compute the average of all observations that fall within the same hour.

A random building ID is selected from the dataset. A time-series plot of electricity consumption (Volume_Purchase_kWh), electricity injection (Volume_Injection_kWh), and net purchase (Net_Volume_Purchase_kWh) from July 1 - July 14, 2022 is shown for clarification. X-axis corresponds to the datetime range, and the unit is day. The units on the Y-axis are in kilowatt-hours (kWh), which is the unit of electrical energy measurement. The net volume purchase is predominantly negative, indicating the presence of photovoltaic panels (Building_ID 1054- Case 3).

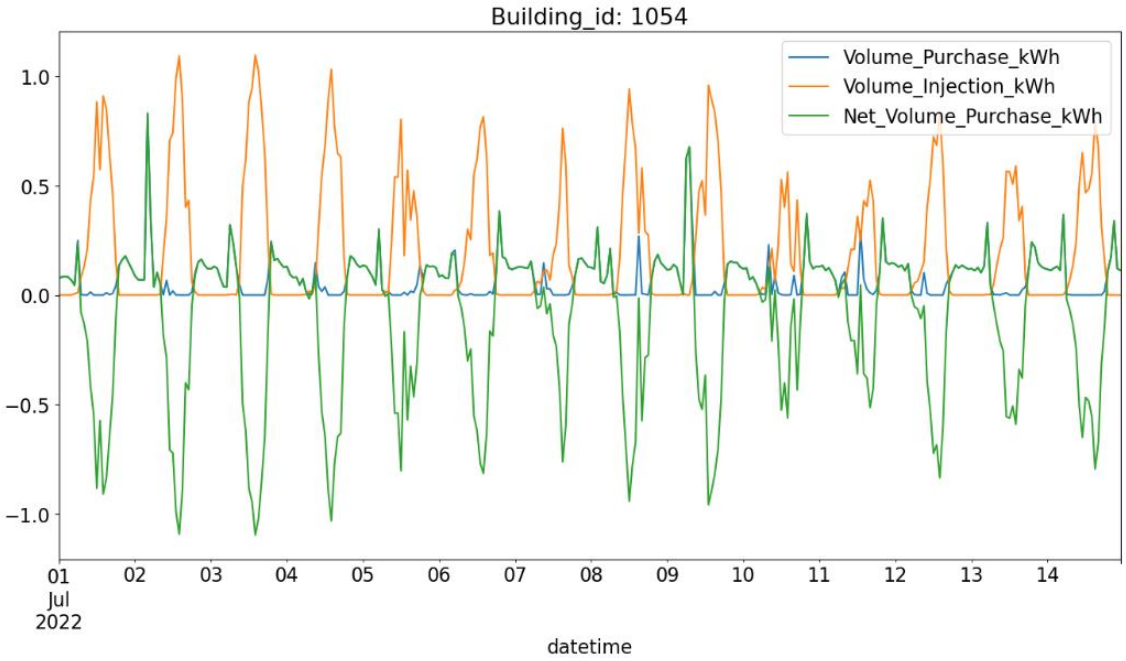


Figure 2 Volume Purchase, Volume Injection and Net Volume Purchase (kWh) over datetime (14 days)

In the next step, the weather data is loaded and merged with electricity load data using 'datetime' as the key. Finally, a new column is created named “day_ahead_forecast”. It contains the Net Volume Purchase (kWh) values from the previous day (24 hours ago). This is essentially a "naive day-ahead forecast", where the predicted value for a given hour is simply the value from the same hour on the previous day. This creates a simple baseline forecast where today's value is assumed to be the same as yesterday's which can be used to compare against more advanced forecasting models. It particularly is useful for lag-based features in machine learning models.

A time-series plot of actual energy consumption (Net_Volume_Purchase_kWh) and forecasted energy consumption (day_ahead_forecast) for the specific building ID (Building_id:1054) over a one week's worth of data (July 1 - July 7, 2022) is shown to visualize how well the day-ahead forecast compares to the actual energy usage. X-axis corresponds to the datetime range, and the unit is day. The units on the Y-axis are in kilowatt-hours (kWh), which is the unit of electrical energy measurement.

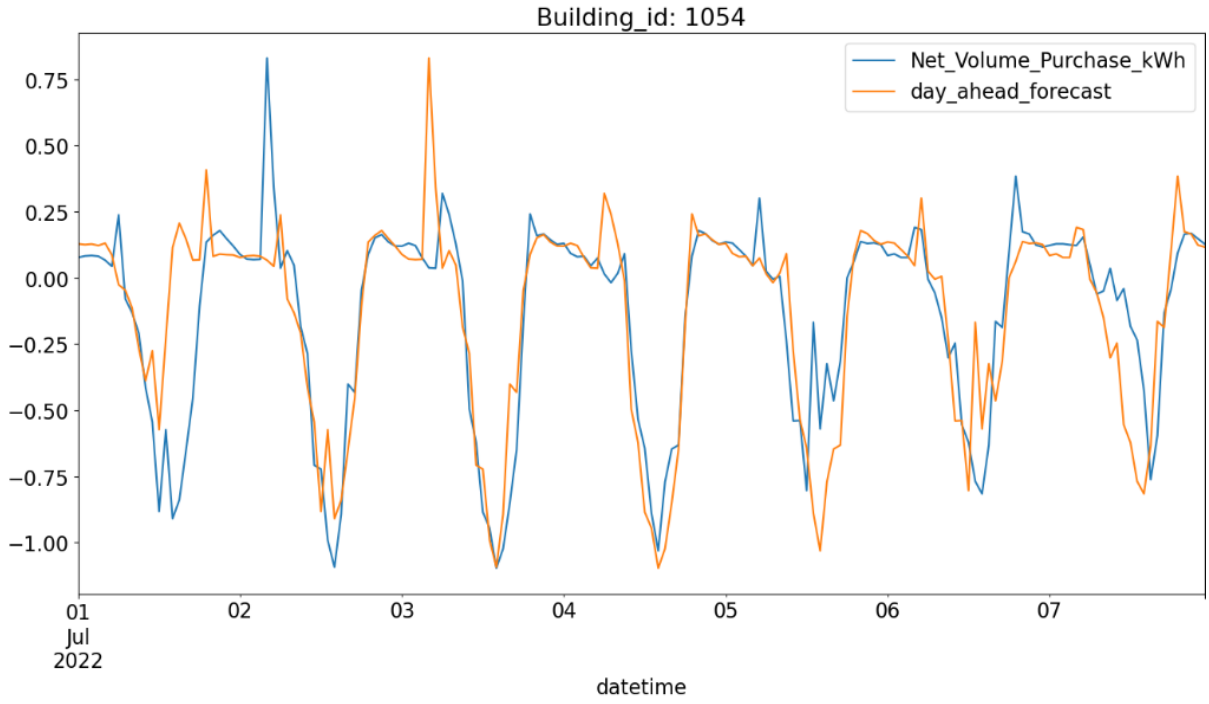


Figure 3 Net Volume Purchase and Day-Ahead Forecast (kWh) over datetime (7 days)

The visualization provides insights into the forecast accuracy, highlighting potential discrepancies between predicted and actual consumption. Almost close alignment suggests that the day-ahead forecast effectively captures the building's energy usage patterns.

2.3.2 Forecasting

Energy use improvement in buildings has been an active area in research. A large variety of methods and approaches has been proposed to improve the energy use in buildings. The proposed system for the Electricity Demand Prediction project leverages advanced machine learning algorithms and historical consumption data to forecast future electricity demands accurately. By analysing patterns and external factors such as weather, holidays, and economic indicators, it aims to provide utilities with valuable insights to optimize power generation and distribution, ensuring a more efficient and reliable energy supply. This predictive system will help reduce energy wastage, lower costs, and contribute to a more sustainable energy infrastructure.

To forecast the energy demand of buildings, both statistical and machine learning-based time series models, including NaïveSeasonal, LinearRegression, and Prophet (statistical models), as well as LightGBM, and RandomForest (machine learning models) are applied. The approach of the project is to collect and analyse the data related to electricity consumption, such as historical usage patterns, weather conditions, and other relevant factors, to make predictions about future demand.

As buildings energy demands are different based on their facilities and consumption, this study uses supervised learning models on five different building cases for short-term energy demand forecasting, leveraging historical data to train statistical and machine learning models. The goal is to select the model that offers the best predictive performance.

The forecasting approach in this project accomplish the steps outlined in the following.

2.3.2.1 Feature Engineering

Feature engineering is a very important step in machine learning. Feature engineering refers to the process of designing artificial features into an algorithm. These artificial features are then used by that algorithm in order to improve its performance, or in other words, reap better results (*Feature Engineering*, n.d.). This included renaming variables for clarity, creating new features such as heating and cooling degree days, normalizing continuous variables, and generating binary indicators. These transformations enhanced the model's ability to learn patterns in energy demand.

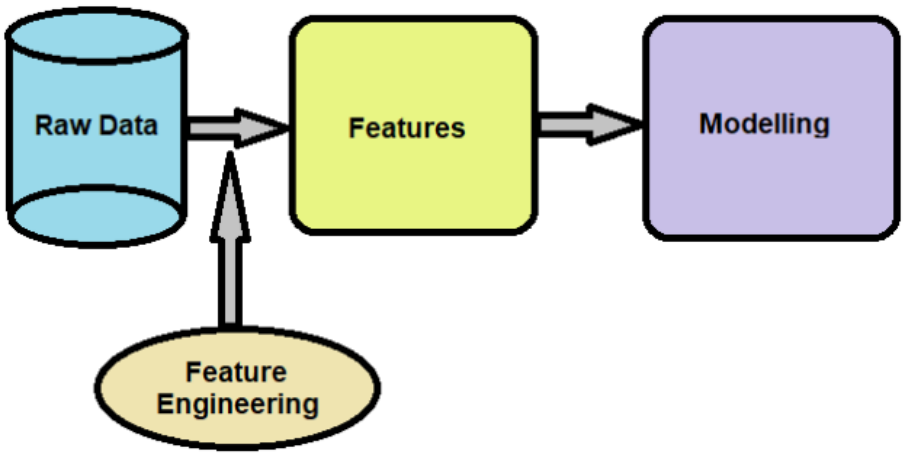


Figure 4 Contribution of Feature Engineering to the forecasting models

When feature engineering activities are done correctly, the resulting data set is optimal and contains all of the important factors that affect the problem. As a result of these data sets, the most accurate predictive models and the most useful insights are produced.

Several feature engineering techniques are applied:

- Renaming Columns: Makes variable names shorter and easier to use
- Dropping Unnecessary Columns: Removes target variables and categorical indicators to keep only relevant numerical features.
- Creating New Features: such as creating Binary indicator, converting percentage values to decimal, specifying threshold for Degree Days for Heating and Cooling, allowing the model to capture nonlinear effects by Squared versions
- Feature Scaling (Min-Max Normalization): Scales selected numerical features to the range [0,1] using “MinMaxScaler”, making training more stable.

Generally, the feature engineering leads to model interpretability improvement, model performance enhancement and Enables Nonlinear Relationships in forecasting.

The forecasting models in this study incorporate the following features as future covariates:

No.	Feature_Category	Covariate_Name
1	Weather Variables	Temperature (temperature_2m (°C))
2		Relative Humidity (relative_humidity_2m (%))
3		Rain (rain (mm))
4		Wind Speed (wind_speed_10m (km/h))
5		Soil Temperature (soil_temperature_0_to_7cm (°C))
6		Direct Radiation (direct_radiation (W/m²))
7	Time-based Features	is_day (Indicator for daytime)
8		is_night (Created as the inverse of is_day)
9		sunshine_duration (Converted from seconds to hours)
10	Degree Day Features (Heating & Cooling Demand Indicators):	heating_degree_days = max (0, temperature - 18)
11		cooling_degree_days = max (0, 18 - temperature)
12		Squared versions of these (heating_degree_days_2, cooling_degree_days_2) to capture non-linear effects

Table 2 Features categories and future covariates used in forecasting models

2.3.2.2 Baseline

The most basic forecasting algorithm, which serves as a benchmark for any more advanced model, is the persistence model. This approach assumes that past values will repeat in the future with a specific periodicity. For instance, the demand at time step “n” is predicted to be the same as at time step “n+k”, where “k” represents the data's periodicity (e.g., daily, weekly, or yearly). Surprisingly, this simple method often performs well for periodic time series, such as electricity demand and solar PV production. Establishing a baseline is essential on any time series forecasting problem. A baseline in forecast performance provides a point of comparison. It is a point of reference for all other modeling techniques. The goal is to get a baseline performance on time series forecast problem as quickly as possible so that can get to work better understanding the dataset and developing more advanced models (*How to Make Baseline Predictions for Time Series Forecasting with Python*, n.d.).

The “NaiveSeasonal” model is a simple forecasting model that predicts based on the past seasonal patterns in the data. The parameter K=24 indicates that the model uses a daily seasonality (since a day has 24 hours, it assumes a repeating pattern every 24 hours). The model will predict future values by

assuming that the demand at a specific time will be similar to the demand at the same time in the previous day. Using “NaiveSeasonal” as a starting point, provide the opportunity to compare how well the other more complex models (e.g., LightGBM, RandomForest) perform relative to this simple baseline. For example, if a more complex model like LightGBM has a much lower MAE than NaiveSeasonal, it indicates that the model is doing better than the simple seasonal baseline.

2.3.2.3 Backtesting

Creating an approach to model creation and selection which is more structured. In machine learning, models are not evaluated just once; instead, cross-validation is implemented to assess their performance. Similarly, in time series forecasting, rather than making a single prediction for the entire test set, it's better to update predictions regularly based on the specific use case—this could be daily, hourly, or at another interval. To achieve this, a function is implemented that evaluates different models using backtesting. This method, commonly used in time series analysis, serves as an equivalent of cross-validation for time series data. Backtesting provides a more reliable estimate of a model's forecasting accuracy compared to fitting a single model to the full dataset. However, the trade-off is that it requires more computational resources.

2.3.2.4 Forecasting Models

forecasting is the process of making predictions of the future based on past and present data and most commonly by analysis of trends and usually needed to determine when an event will occur or a need arise, so that appropriate actions can be taken.

In the proposed approach, various forecasting models are employed to predict the future energy demand of buildings. The models utilized in this study include NaïveSeasonal, LinearRegression, LightGBM, Prophet and RandomForest.

These models were implemented with basic form and with some modifications as outlined below:

1. **Lag Size (lags)**

- lags=24: Uses the past 24 hours (1-day history).
- lags=7 * 24 (168): Uses the past 7 days (1-week history).

Larger lags give the model more context but increase complexity.

2. **Use of Future Covariates (lags_future_covariates)**

- Models without future covariates rely only on past values.
- Models with lags_future_covariates= [0] use future covariates only for the current time step (e.g., temperature, relative_humidity, direct_radiation, etc.).

The models leverage weather conditions, temporal indicators (day/night), sunshine duration, and temperature-derived heating/cooling demand features to improve forecasting performance. Additionally, feature scaling ensures stability and comparability across different variables.

3. **Encoders (add_encoders)**

- Models without encoders use only raw time series data.
- Models with encoders transform time-related features (e.g., cyclic encoding for hour, categorical encoding for dayofweek).

4. **Forecasting Method (output_chunk_length)**

- output_chunk_length=1: Recursive forecasting (one-step-ahead, then using predictions as new inputs).
- output_chunk_length=30: Direct forecasting (predicts the next 30 steps all at once).

Recursive forecasting is step-by-step, while direct forecasting is for long-horizon predictions.

5. **Multi-Step Approach (multi_models)**

- Default (multi_models=True): A single model is trained to predict multiple steps.
- multi_models=False: A separate model is trained for each future step.

Some of these modifications were applied incrementally, while others were implemented simultaneously across models. These adjustments were made to determine which models outperform others.

The performance of the forecasting models is evaluated using the relative mean absolute error (rMAE) metric. This involves first calculating the mean absolute error (MAE) for each building's prediction. Subsequently, the relative mean absolute error (rMAE) is computed by comparing the results to the baseline model, which is the "NaiveSeasonal" model with a seasonal period of (k=24). This seasonal period of 24 corresponds to a 24-hour cycle, given that the data is hourly. Therefore, (k=24) indicates that the model assumes a seasonal pattern that repeats every 24 time steps, i.e., every 24 hours.

2.3.3 Feature Extraction

Feature extraction is a technique used to reduce the complexity of long time series data by summarizing it into a lower-dimensional set of key values or statistics. This process helps to preserve the most significant information and patterns while simplifying the data. For instance, instead of retaining all individual timestamps in a power consumption series, feature extraction condenses it into a few key features that reflect the overall behavior of the time series.

Once extracted, these features can be used for various tasks, such as:

- Unsupervised learning: Grouping buildings with similar power consumption patterns through clustering.
- Supervised learning: Using the extracted features in conjunction with known labels (e.g., building metadata) to train predictive models.

Various algorithms have been developed to convert time series data into a smaller set of essential features, where the number of features is much smaller than the original time series data. These features capture important statistics like averages, variability, and peak values. In this study, two methods for extracting features from the aggregated power consumption data of 500 buildings in the Fluvius dataset are explored:

1. Domain-Informed Feature Extraction: This method transforms each time series into a daily load profile and extracts relevant features such as statistical moments, peak observations, and their timings, which are critical factors for energy flexibility. The “IFEEL” package by Maomao Hu is used for this process.
2. Domain-Agnostic Feature Extraction: This approach takes a more general perspective, analyzing the entire time series to capture long-term trends, seasonal patterns, autocorrelations, and spectral variations. It is designed to capture generic features like trends and seasonality, without assuming domain-specific knowledge. The “tsfeatures” package by Nixtla is used for this method.

The features from both methods are combined into an aggregated feature vector, significantly reducing the data size and transforming the high-resolution time series into a lower-dimensional space. After feature extraction, clustering techniques will be applied to group buildings based on their power usage patterns, revealing clusters of buildings with similar behaviors (Hussain Kazmi & Ada Canaydin, n.d.).

To begin, the necessary packages are installed, and the required libraries are loaded:

- tsfeatures: Used for domain-agnostic feature extraction, with custom modifications available in the src_nixtla folder.
- ifeel_extraction and ifeel_transformation: Domain-informed features are derived using the IFEEL package, specifically designed for electricity load analysis. Custom modifications are made to the original IFEEL code.

In this project, both domain-informed and domain-agnostic feature extraction techniques are applied using the respective IFEEL and Nixtla packages.

Data Quality Check and Pre-processing

Before proceeding with feature extraction, it is crucial to evaluate the quality of the time series data and address any inconsistencies. This is achieved through a set of data quality check and pre-processing functions, which can be applied repeatedly by simply modifying the input time series.

Overview of Functions:

1. calculate_data_quality_metrics (df: pd.DataFrame)
   This function calculates three key metrics for each building and returns:

- Missing Percentage: The percentage of missing values in the energy consumption data.
- Null Percentage: The percentage of null values (0.00 kWh) in the energy consumption data.
- Duration: The total number of days of data collection for each building.

2. plot_quality_metrics (missing_perc: pd.Series, null_perc: pd.Series, duration: pd.Series, dataset_name: str)
   This function generates visualizations to assess the data quality:

- Heatmap of Missing Data: A visual representation of missing data in the time series.
- Top 25 Buildings with Highest Null Percentage: A bar chart showing buildings with the highest percentage of null values.
- Measurement Duration Distribution: A histogram displaying the distribution of data collection durations.

3. filter_buildings (df: pd.DataFrame, missing_perc: pd.Series, null_perc: pd.Series, duration: pd.Series, missing_threshold: float = 50, null_threshold: float = 50, duration_threshold: int =30)
   This function filters buildings based on specified thresholds for missing values, null values, and duration, returning a cleaned DataFrame. The thresholds are:

- Missing Threshold: Maximum allowable percentage of missing values.
- Null Threshold: Maximum allowable percentage of null values.
- Duration Threshold: Minimum required duration for data collection.

4. check_constant_days (df: pd.DataFrame)
   Identifies days with constant energy values for each building, returning a summary DataFrame and printing details about unique IDs with constant values.
5. remove_constant_days_from_buildings (df: pd.DataFrame)
   This function removes days with constant values and eliminates entire buildings with excessive constant days.
6. impute_missing_values_with_weekly_median (df: pd.DataFrame)
   Fills in missing values using the weekly median of daily usage profiles, returning the DataFrame with imputed values.

Next, the overall data quality of the dataset is visualized to make informed decisions about the necessary pre-processing steps.

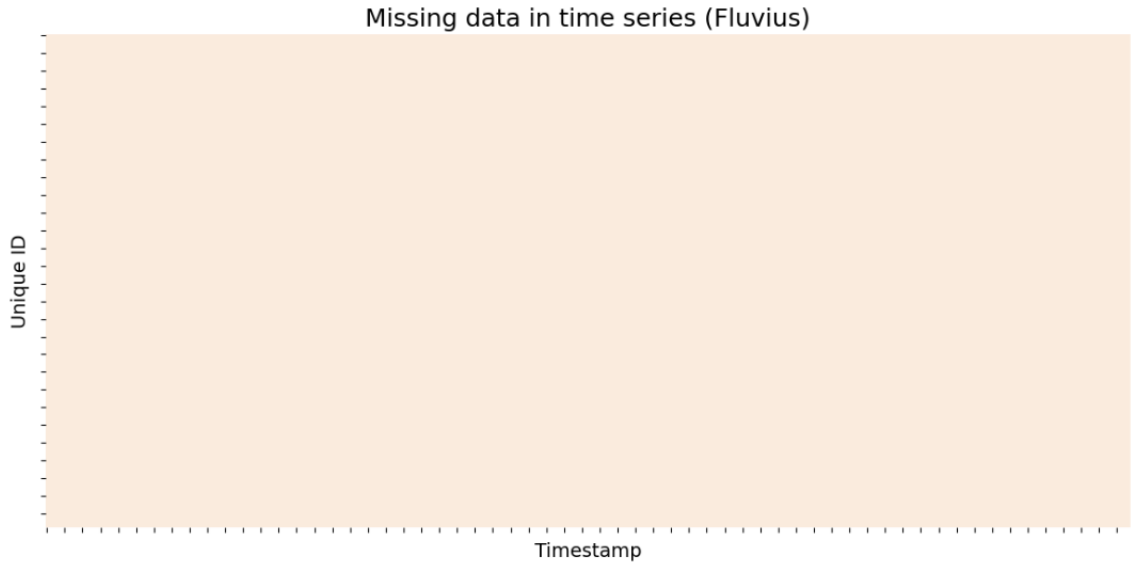


Figure 5 Unique IDs over Time- Missing data in the time series

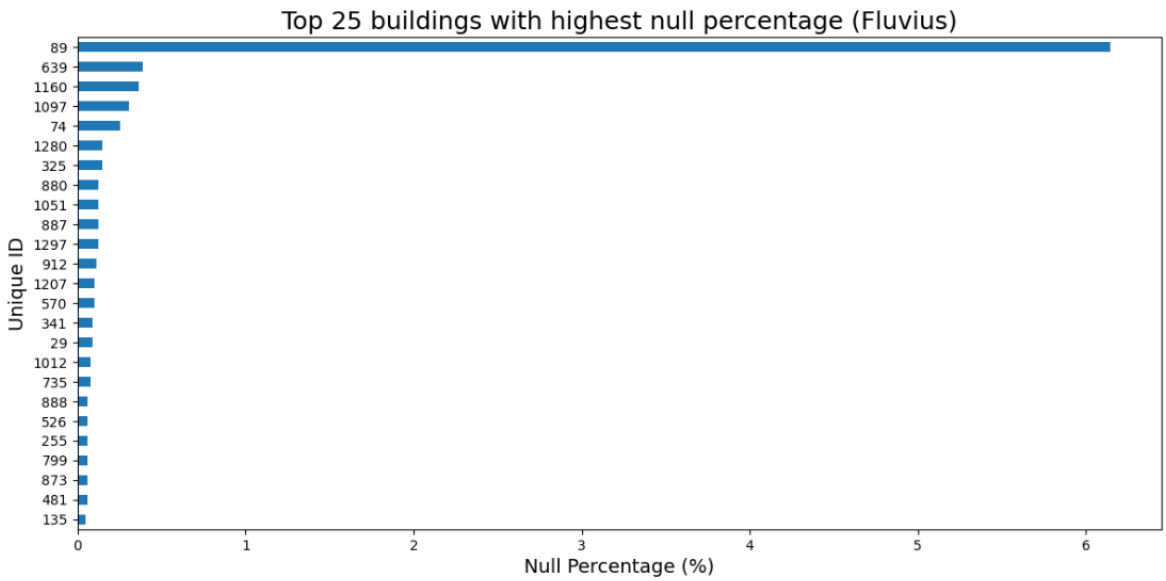


Figure 6 Unique IDs over Null percentage- Top 25 buildings with the highest percentage of null values

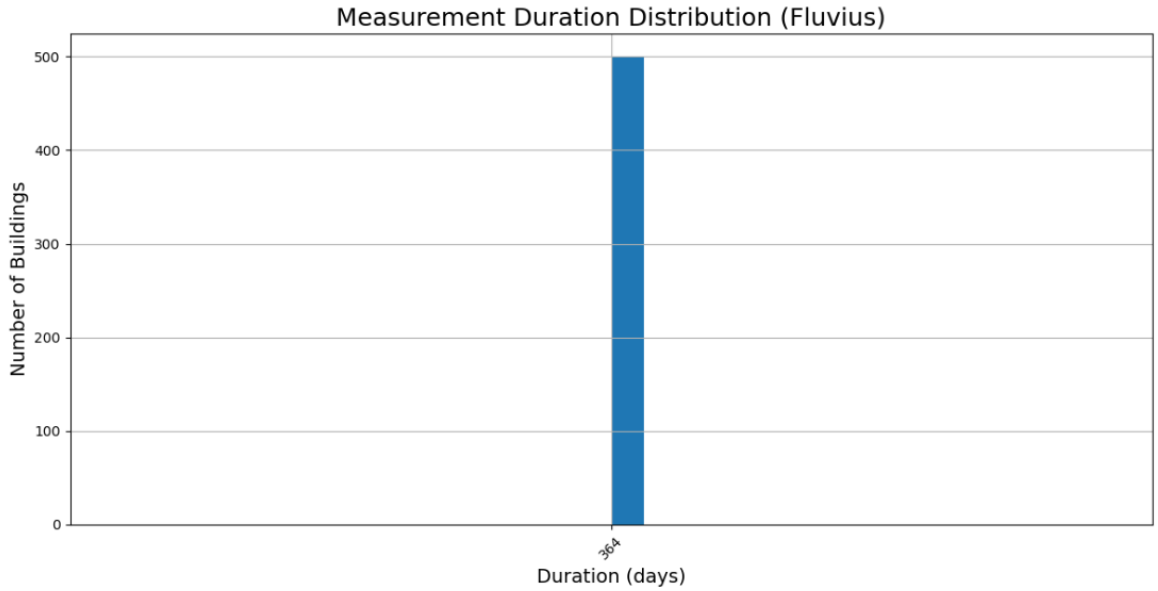


Figure 7 Number of buildings over Duration- The measurement duration distribution across all buildings

According to the graphs, the dataset contains no missing values, but one building (with unique ID 89) has a significant number of null values compared to other buildings, roughly 6%. Several other buildings also have null values, but these are below 1%. On a positive note, the data collection duration is consistent across all buildings, with each having one full year of observations. Now the pre-processing stage can be proceeded to make ready the dataset for further analysis.

Data Quality Filtering: To ensure the dataset meets quality standards, buildings that don't meet certain criteria should be filtered out. By default, buildings with more than 50% null values, over 50% missing values, or fewer than 30 days of data are removed. The thresholds code script is as follows:

```
custom_missing_threshold = 60
custom_null_threshold = 60
custom_duration_threshold = 20

df_processed = filter_buildings(
    df,
    missing_perc,
    null_perc,
    duration,
    missing_threshold=custom_missing_threshold,
    null_threshold=custom_null_threshold,
    duration_threshold=custom_duration_threshold
)
```

Figure 8 Predefined quality thresholds for missing values, null values and duration

According to the predefined quality thresholds, there is no building with more than 50% null values, more than 50% missing values, or less than 30 days of readings to be excluded from the dataset.

By completing these steps, the dataset is properly cleaned and prepared for feature extraction.

Domain-agnostic feature extraction (Nixtla package)

Initially, the domain-agnostic feature extraction method is applied using the Nixtla package, which provides a set of features that can be utilized across various domains without requiring domain-specific expertise. In the Nixtla package, the “freq” parameter within the tsfeatures function defines the frequency of the time series data, representing the number of data points per unit of time. Adjusting this parameter to match the dataset's specific frequency is crucial for accurate feature extraction. In the case of the Fluvius dataset, where the data is recorded at 1-hour intervals, a total of 24 data points are captured per day. Consequently, the freq parameter is set to 24 when applying the tsfeatures function to ensure precise extraction of relevant features.

Missing values, as well as buildings (unique IDs) with missing feature values, are imputed using a K-Nearest Neighbors (KNN) imputer to ensure data completeness.

Additionally, features related to the time series length, including 'frequency,' 'series_length,' 'seasonal_period,' and 'nperiods,' are removed to prevent potential data leakage in subsequent analysis tasks.

Domain-informed feature extraction (IFEEL package)

The IFEEL package facilitates the extraction of specialized features for building energy demand analysis by leveraging domain-specific knowledge. Unlike general domain-agnostic feature extraction methods that analyze entire time series, IFEEL focuses on daily load profiles for each building. To capture key characteristics at the building level, summary statistics—including the minimum, maximum, median, mean, and standard deviation—are calculated for each extracted feature.

This approach results in the generation of 21 distinct features:

- 13 Global Features (GFs): Derived directly from raw time-series data.
- 8 Peak-Period Features (PFs): Identified during peak consumption periods using the Symbolic Aggregate Approximation (SAX) technique, which converts daily load data into symbolic representations to detect recurring consumption patterns in smart meter data.

Given this methodology, each building ultimately has 105 features (21 features × 5 summary statistics). The feature extraction process is carried out through three key functions:

- process_site: Processes each building individually by cleaning, resampling, and formatting data to match IFEEL's requirements. It then extracts global and peak-period features from daily load profiles.
- process_one_by_one: Iterates over all buildings in the dataset, applying process_site to each one.
- merge_features: Combines individual feature files from each building into a single dataset, enabling a comprehensive overview of extracted features across all buildings.

Adjustable Parameters

- In the process_site function, parameters such as time_business_start, time_business_end, and alphabet_size can be adjusted to refine the analysis. The SAX alphabet size is set to 7 to balance interpretability and granularity.
- In process_one_by_one, the freq parameter is set to "60T" to match the dataset's 1-hour sampling intervals.

Following feature extraction, the features from all buildings are merged into a single dataset, with unique_id set as the index. To simplify data handling, bracketed lists in feature names—such as 'Peak_all: time' and 'Peak_all: duration'—are reformatted by removing the brackets. A thorough check

for missing values is performed on the extracted features, and any missing values are imputed using a K-Nearest Neighbors (KNN) imputer to ensure data completeness. Unlike the domain-agnostic Nixtla features, which analyze entire time series, IFEEL features are aggregated at the building level. To achieve this, summary statistics—including the minimum, maximum, median, mean, and standard deviation—are computed for each extracted IFEEL feature.

Clustering

In this section, the feature matrix is utilized to cluster buildings with similar Distributed Energy Resource (DER) profiles. The approach involves reducing the dimensionality of the feature matrix to a two-dimensional space and applying an unsupervised clustering algorithm for effective grouping.

Before clustering, a DataFrame is constructed, incorporating both the input features and the target variables. Initially, the two extracted feature sets, ifeel_features_df and nixtla_features_df, are merged to form the final feature matrix, which serves as the foundation for the analysis.

Subsequently, a target variable, Class_Name, is generated from the metadata. This variable categorizes households based on the type of DERs they possess, such as solar panels (PV), electric vehicles (EV), and heat pumps (HP). The feature matrix is then combined with the target variable, ensuring a comprehensive dataset for clustering. To maintain data consistency, any rows containing NaN values are removed, as differences between the metadata and extracted features may arise due to preprocessing steps. To prepare for clustering, the numerical features are standardized to ensure equal contribution to the analysis. This prevents any feature from dominating the results due to scale differences. StandardScaler from the sklearn library is applied to transform the numerical data, ensuring a mean of 0 and a standard deviation of 1. Dimensionality reduction is then performed to facilitate visualization and identify patterns or clusters. t-distributed Stochastic Neighbor Embedding (t-SNE) is employed for this purpose, with a fixed random_state to ensure reproducibility. A DataFrame is subsequently created to store the t-SNE results alongside the corresponding class labels.

The clustering process is carried out using K-Means, with the number of clusters set to five, as there are known distinct groups. The final step involves visualizing the clustering results by plotting the ground truth against the t-SNE components, along with the cluster boundaries. To delineate each cluster visually, convex hulls are used.

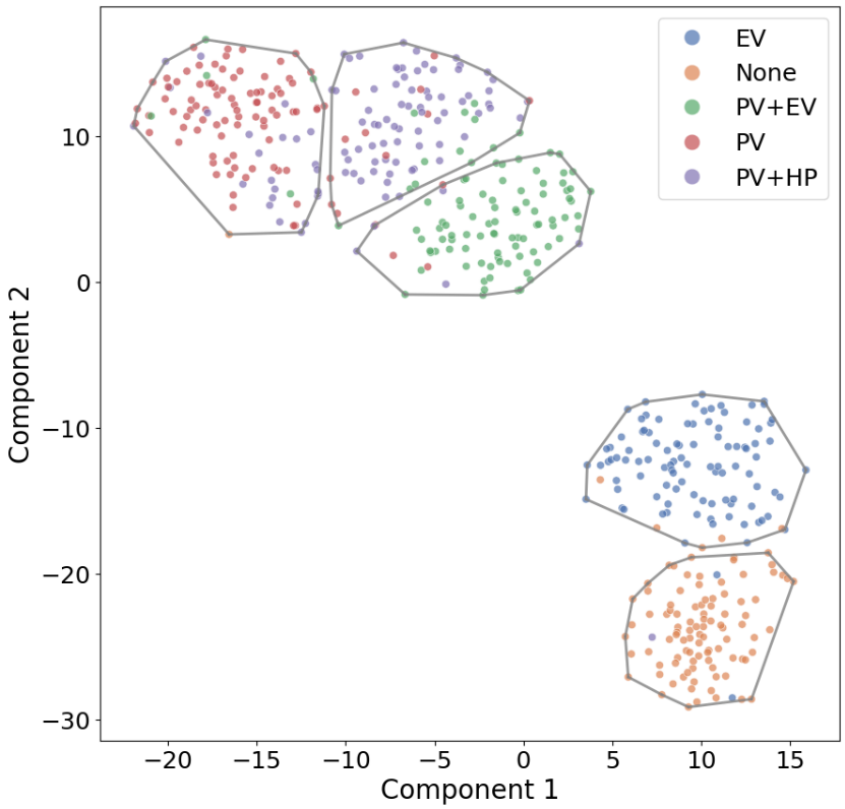


Figure 9 The scatter plot of projected high-dimensional data into a 2D space. Each colour corresponds to a different class, illustrating the distribution and separation of clusters.

As it is depicted in the graph, each dot represents a unique building, while the grey lines illustrate how the buildings are grouped using the K-Means algorithm.

2.3.4 Recommender System

A recommender system, is a type of machine learning that leverages data to predict, filter, and identify the most relevant options for individuals from an ever-expanding pool of choices.

A recommendation system is an algorithm, often tied to machine learning, that utilizes Big Data to suggest additional products or services to consumers. These suggestions are based on various factors and relevant data. Recommender systems are valuable tools as they enable users to discover products or services that they might not have discovered on their own or would have taken too much time to find. These systems are trained to analyze past behaviors and features to predict consumer interests and desires. They effectively guide consumers toward products or services that align with their preferences, ranging from books and videos to health classes and clothing (*Recommendation System*, n.d.).

In this project the recommender system is extended to suggest the best forecasting models for an unknown building. Thus, it is not necessary to run multiple time-consuming forecasting models to identify the best performance for a new building. Since several predictive models have already been applied to five hundred buildings with varying consumption patterns, a wealth of comprehensive data and results has already been collected. Additionally, the features of these buildings have already been extracted. Therefore, by inputting the time series features of a new unknown building, the recommender system can suggest the top forecasting models that are most suitable for that building. The following flowchart illustrates the overall framework of the recommender system.

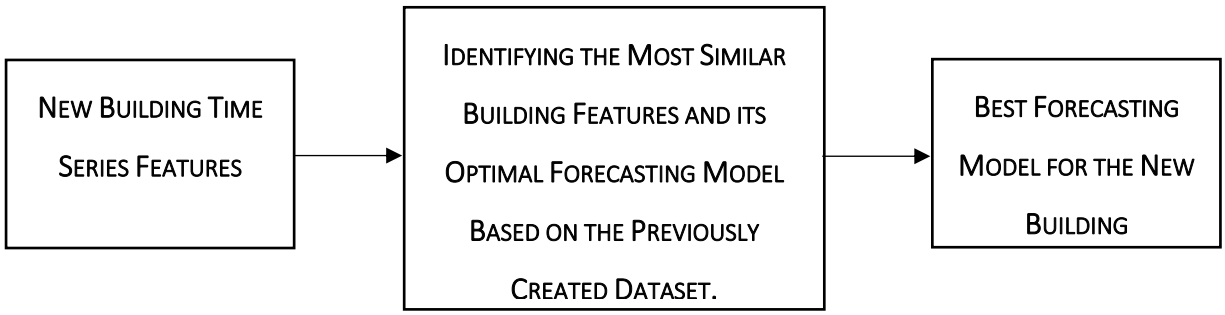


Figure 10 The flowchart of the general structure of the recommender system

Operational Framework

To construct this system within a Python environment, the necessary libraries are first imported. Subsequently, both domain-informed and domain-agnostic features, extracted in the previous step, are loaded alongside the forecasting performance metrics. At this stage, a comprehensive dataset is assembled, encompassing building IDs, extracted time series features, and prediction performance results. The extracted features serve as input data, while the forecasting models represent the target variable. The models are trained on 80% of the data. To enhance accuracy, training is conducted separately for each case, enabling the model to capture diverse consumption patterns. Finally, the trained models are tested on the remaining 20% of each building case, thereby constructing the necessary matrices for evaluation.

To approach this target, the K-Nearest Neighbors (KNN) model is used. KNN is a supervised learning algorithm that can be applied to both classification and regression tasks. The algorithm predicts the class or value of a given test instance by measuring its distance from all training data points. It then selects the K nearest neighbors—i.e., the K training points closest to the test instance. In classification, KNN assigns the test instance to the class with the highest probability, determined by the majority

class among the selected K neighbors. In regression, the predicted value is computed as the mean of the target values of these K nearest neighbors (*K-Nearest Neighbor*, n.d.).

The functioning of the K-Nearest Neighbors (K-NN) algorithm can be outlined as follows:

- Determine the Number of Neighbors (K): Select an appropriate value for K, representing the number of nearest neighbors to consider.
- Compute Distance Metrics: Calculate the Euclidean distance between the test data point and all training data points.
- Identify the Nearest Neighbors: Select the K data points with the shortest Euclidean distances.
- Classify the Test Instance: Count the occurrences of each category among the K nearest neighbors.
- Assign a Category: Assign the test instance to the category with the highest frequency among the K neighbors.
- Model Readiness: The classification or regression model is now prepared for making predictions.

Accuracy Assessment

To assess the accuracy of the system's recommendations, the evaluation process is carried out as follows:

First, the closest training sample to each test sample is identified. Next, the optimal forecasting model and its corresponding rMAE value for the nearest training sample are determined. Once the optimal forecasting model for the training sample is identified, its corresponding performance value for the test sample is selected. Additionally, the best forecasting model, characterized by the lowest rMAE value, is identified for each test sample. The difference between these two evaluation values is then computed for each test instance. Finally, the Mean Absolute Error (MAE) is averaged across all test samples, providing a measure of the overall accuracy of the designed recommender system.

Therefore, this system identifies the most appropriate forecasting models for building energy demand, tailored to the unique and unknown electricity consumption patterns of a building.

3 DATA ANALYSIS

In this chapter, the dataset is initially examined to identify underlying patterns, noise levels, and correlations with weather-related covariates. To facilitate this analysis, the necessary libraries and packages are imported. Furthermore, both the dataset and performance results are loaded to support subsequent evaluations.

3.1 ANALYSING TIME SERIES DATA

Performance of Net Volume Purchase (NVP) over Datetime

Generating a separate line plot for each EAN_ID in the dataset, showing how the net volume purchase (Net_Volume_Purchase_kWh) varies over time (Datetime). The following graphs illustrate a randomly selected building as a representative example for each building case.

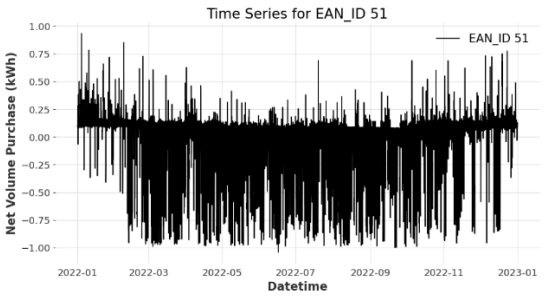


Figure 11 Plot of NVP over Time (1 year)- Case1 includina PV

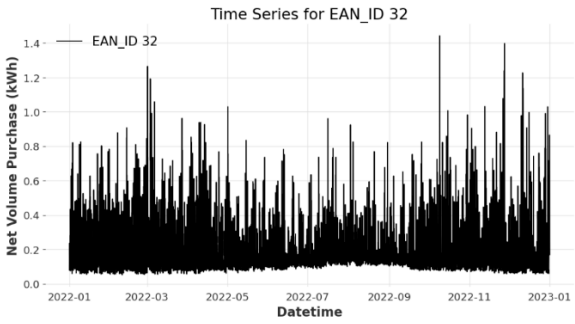


Figure 12 Plot of NVP over Time (1 year)- Case2 including No specific installation

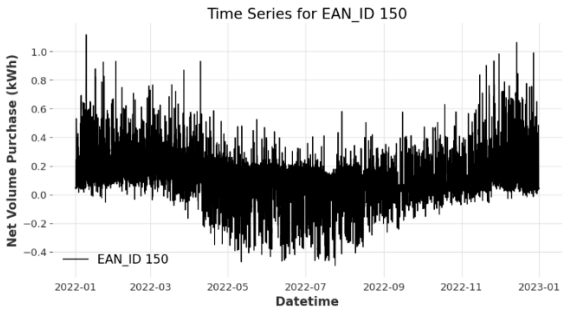


Figure 13 Plot of NVP over Time (1 year)- Case3 including PV+HP

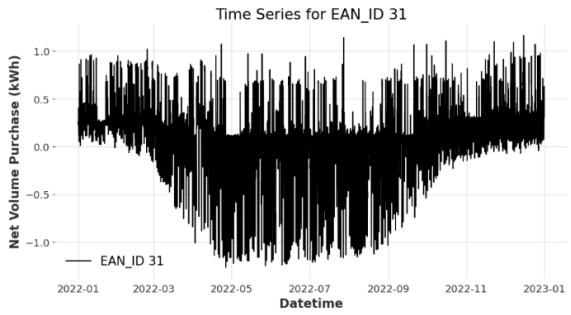


Figure 14 Plot of NVP over Time (1 year)- Case4 including PV+EV

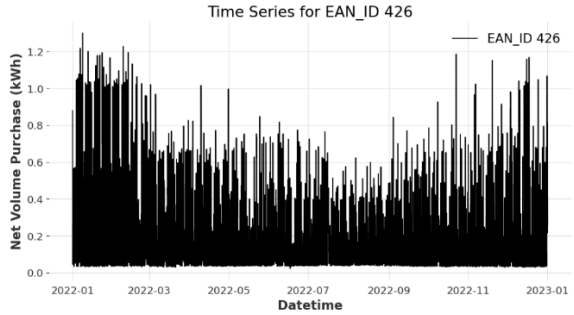


Figure 15 Plot of NVP over Time (1 year)- Case5 including EV

The graphs show the variation of Net Volume Purchase (kWh) over the year 2022 for the specific building of each case.

The three graphs corresponding to buildings with EAN_IDs 51, 150, and 31 represent cases 1, 3, and 4, respectively. These cases include photovoltaic (PV) panels, which are evident from the graphical representations. The presence of negative net volume purchase in these graphs indicates the amount of electricity generated by the PV panels. Furthermore, the graphs for cases 3 and 4 illustrate a reduction in electricity production, as these cases involve heat pumps (HP) and electric vehicle (EV) consumers, respectively. In contrast, the buildings with EAN_IDs 32 and 426 correspond to cases 2 and 5, which do not incorporate PV panels. Consequently, no electricity production is observed in their respective graphs.

Autocorrelation Function (ACF) and **Partial Autocorrelation Function (PACF)**

The Autocorrelation Function (ACF) plot is utilized to analyze the extent to which past values influence future values in a time series, while the Partial Autocorrelation Function (PACF) plot is employed to assess the direct relationship between a given time step and its preceding lags. Both ACF and PACF measure how a time series is related to its past values (lags), but the key difference is ACF measures the total correlation between a time series and its lagged values, including both direct and indirect relationships. PACF measures only direct correlation, removing the effect of intermediate lags.

These plots are particularly useful for identifying seasonality in time series data. The presence of significant spikes at specific lags in the ACF plot indicates the existence of seasonal or persistent patterns. The following graphs serve as illustrative examples for each case.

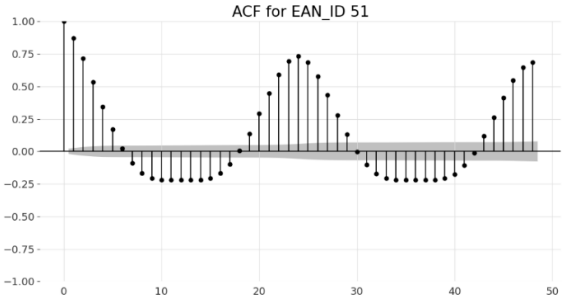


Figure 16 Autocorrelation Function (ACF) for 50 timestamp- Case1 including PV

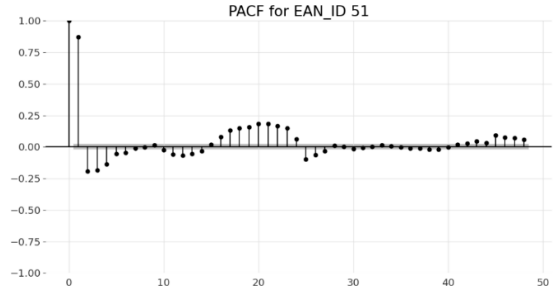


Figure 17 Partial Autocorrelation Function (PACF) for 50 timestamp- Case1 including PV

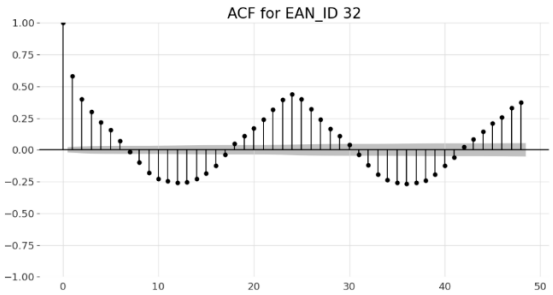


Figure 18 Autocorrelation Function (ACF) for 50 timestamp- Case2 including No specific installation

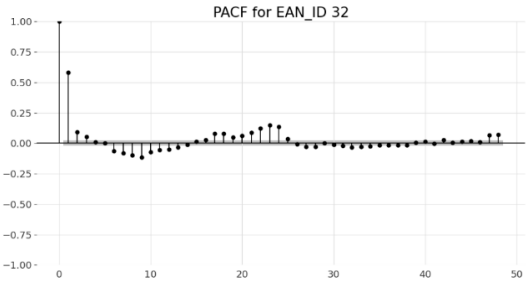


Figure 19 Partial Autocorrelation Function (PACF) for 50 timestamp- Case2 including No specific installation

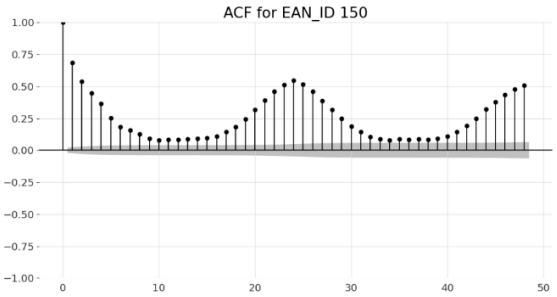


Figure 20 Autocorrelation Function (ACF) for 50 timestamp- Case3 including PV+HP

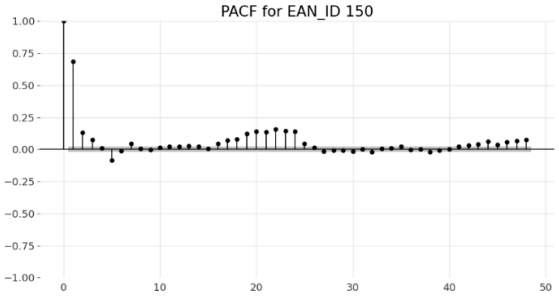


Figure 21 Partial Autocorrelation Function (PACF) for 50 timestamp- Case3 including PV+HP

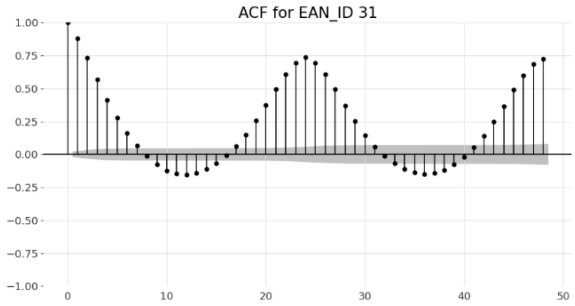


Figure 22 Autocorrelation Function (ACF) for 50 timestamp- Case4 including PV+EV

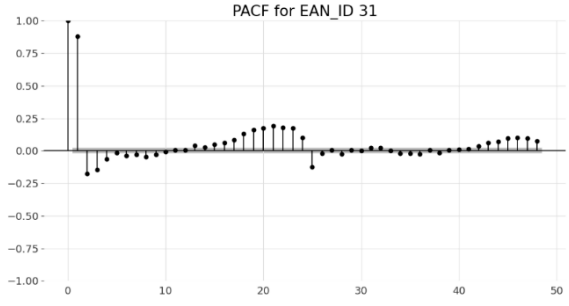


Figure 23 Partial Autocorrelation Function (PACF)- Case4 including PV+EV

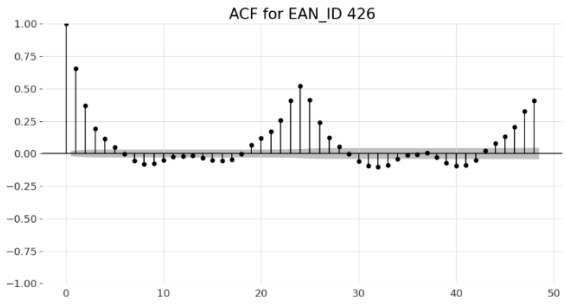


Figure 24 Autocorrelation Function (ACF) for 50 timestamp- Case5 including EV

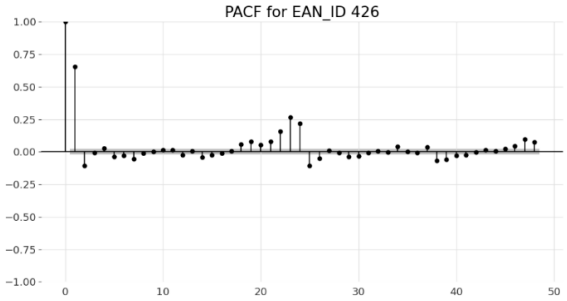


Figure 25 Partial Autocorrelation Function (PACF) for 50 timestamp- Case5 including EV

In these graphs, the Y-axis in both the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots represent correlation values ranging from -1 to 1, indicating the strength of the relationship between the time series and its lagged values. The X-axis represents the lag values, where, given the hourly nature of the time series data, the numbers correspond to hours. The ACF plots exhibit high correlation values at lags 24 and 48, suggesting the presence of daily seasonality in the data.

Correlation between Net Volume Purchase (NVP) and weather covariates

Correlation is a key statistical concept that researchers employ to analyze connections within the data. It helps to Understand the relationship between variables. The connection between two or more variables is known as their correlation. Correlation refers to the degree to which the variables change together or co-vary. It looks at the simultaneous fluctuations in both or all variables measured. A high correlation indicates the variables tend to move in tandem. A low correlation means the variables are not closely associated with their fluctuations. Knowing the correlation helps discover important

relationships between different factors. It provides insight into how changes in one variable may correlate with or predict changes in another. Values closer to 1 or -1 represent stronger positive or negative correlations, while those closer to 0 indicate little connection between the variables (*Correlation in Machine Learning*, n.d.).

The following heatmap indicates correlation between weather covariates and target variable (Net Volume Purchase).

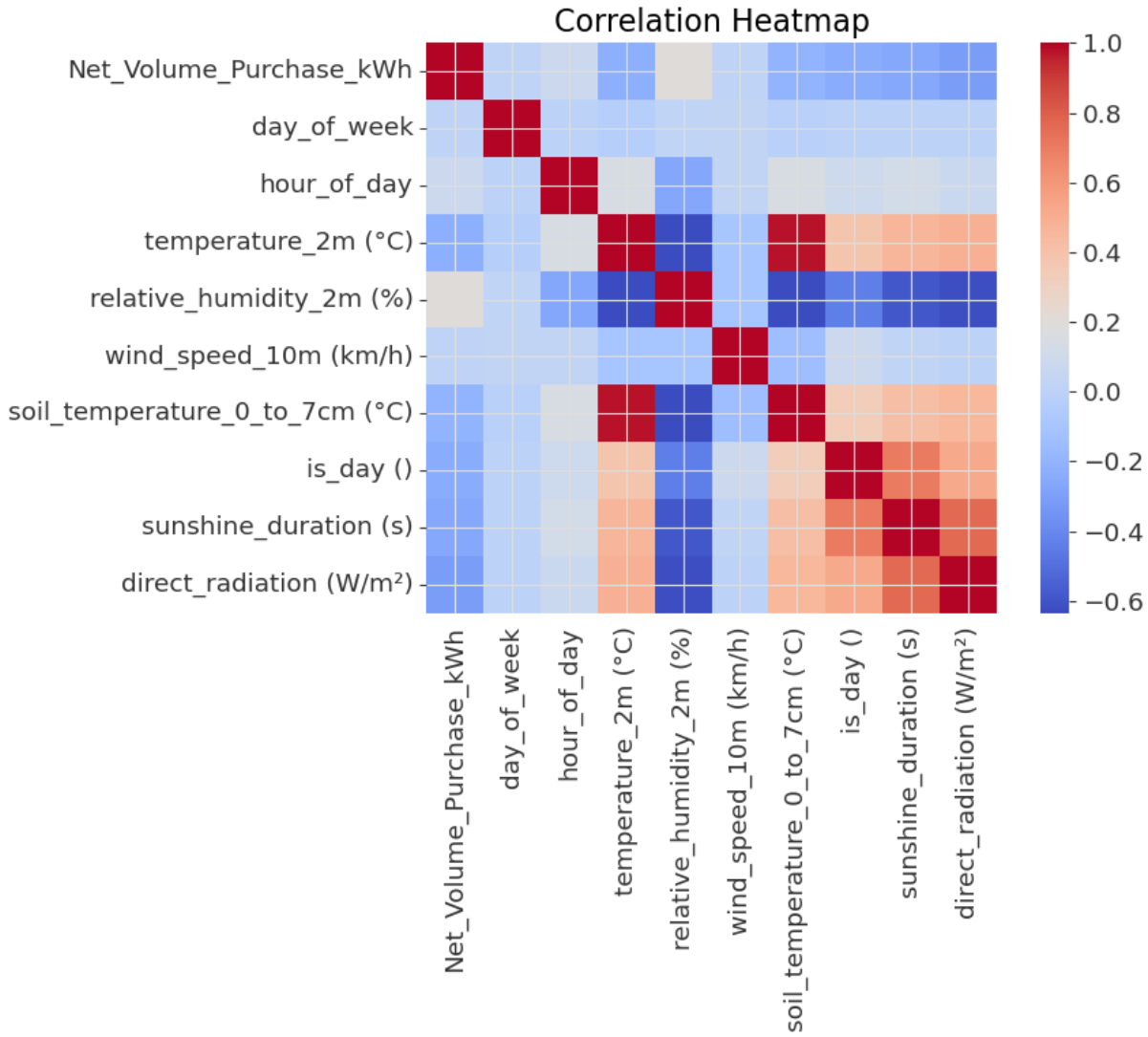


Figure 26 The correlation heatmap of net volume purchase (kWh) and Covariates

As observed, certain exogenous weather variables, such as temperature, direct radiation, and sunshine duration, exhibit a moderate negative correlation with the target variable, Net_Volume_Purchase. This indicates that an increase in temperature, direct radiation, and sunshine duration is associated with a decrease in Net_Volume_Purchase.

3.2 ANALYZING FORECASTING

Here, the results are presented through visualizations to facilitate a more comprehensive analysis. The necessary libraries are imported, and both the dataset and performance metrics are loaded accordingly.

To enhance clarity in visualization, the building cases are categorized based on their shape in certain plots. Additionally, the stored performance of various forecasting models is assessed using the rMAE metric, retrieved during this phase.

Define Shape Labels

The shape_labels dictionary maps specific marker shapes ("o", "^", "s", "P", "D") to corresponding building types:

- "o" → PV (Photovoltaic)
- "^" → None (No specific installation)
- "s" → PV + HP (Photovoltaic + Heat Pump)
- "P" → PV + EV (Photovoltaic + Electric Vehicle)
- "D" → EV (Electric Vehicle)

The data is recorded at an hourly frequency. A forecast for the following day is generated based on the “Net_Volume_Purchase_kWh” column. This code adds a new column that predicts the net volume purchase for the next day, using the current data as a reference.

The following graph compares the forecast errors “dayaheadforecast” against energy consumption “Net_Volume_Purchase_kWh” while distinguishing different building types based on their energy configurations.

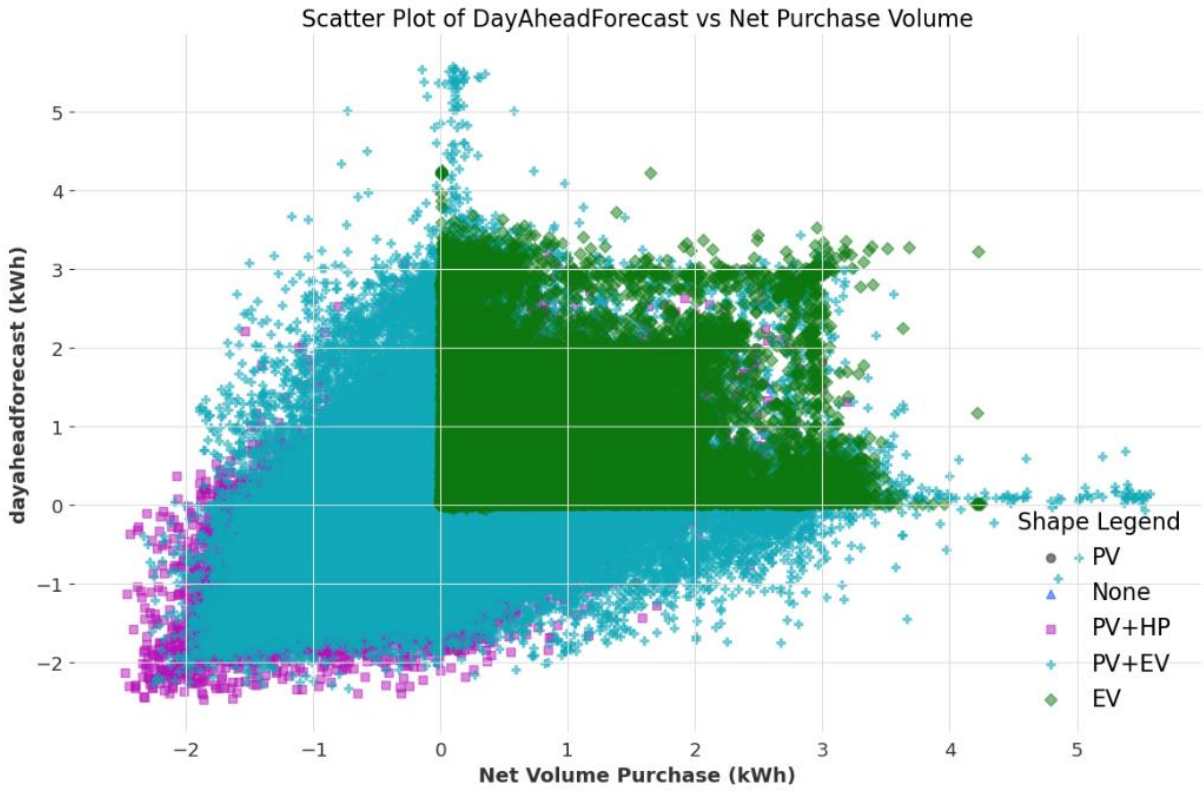


Figure 27 The scatter plot of day ahead forecast over net volume purchase (kWh) for building cases including PV, No specific installation, PV+HP, PV+EV and EV

The performance of certain cases is not clearly distinguishable in this graph due to overlapping data points. Therefore, the day-ahead forecast for *Net_Volume_Purchase* is plotted separately for each case to enhance clarity.

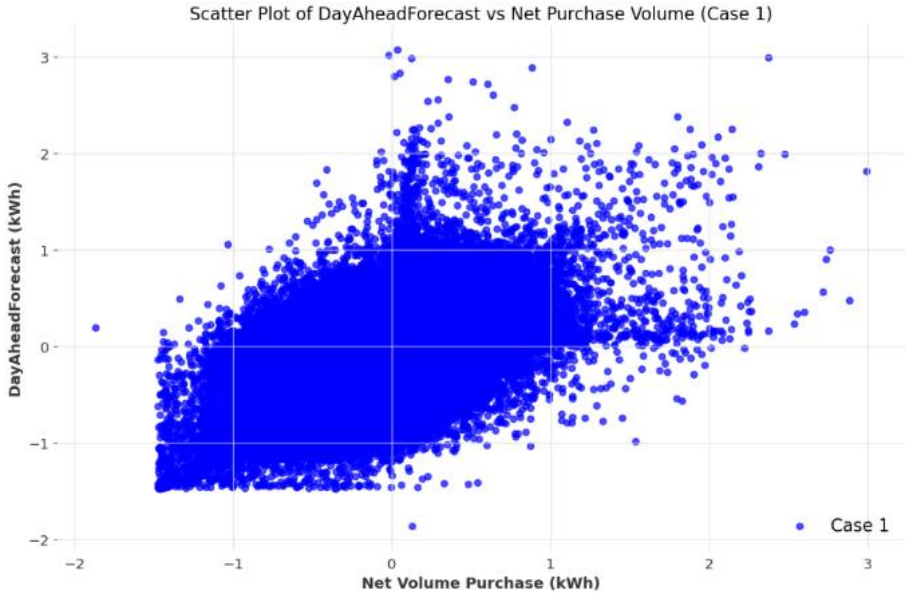


Figure 28 The scatter plot of day ahead forecast over net volume purchase (kWh)- Case1 including PV

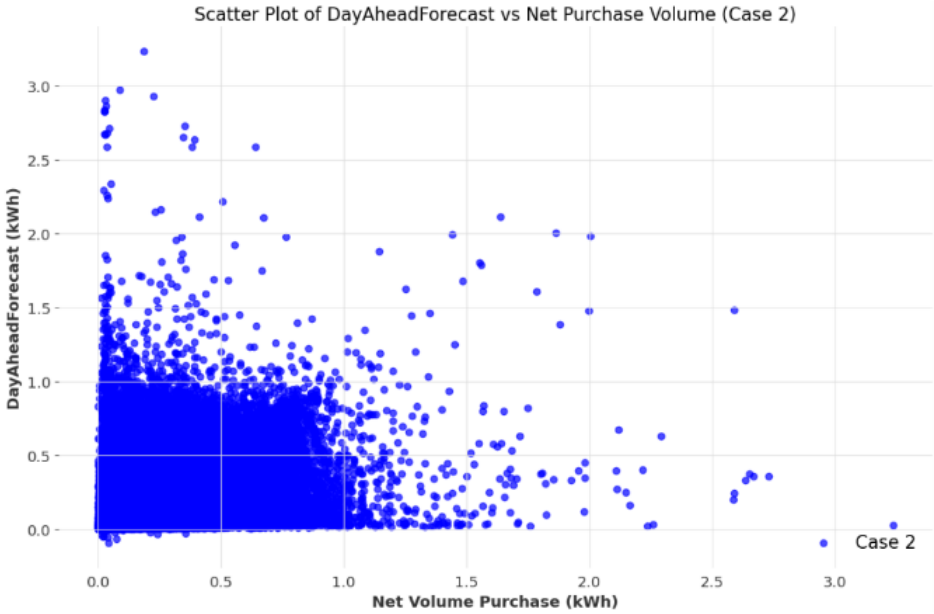


Figure 29 The scatter plot of day ahead forecast over net volume purchase (kWh)- Case2 including No specific installation

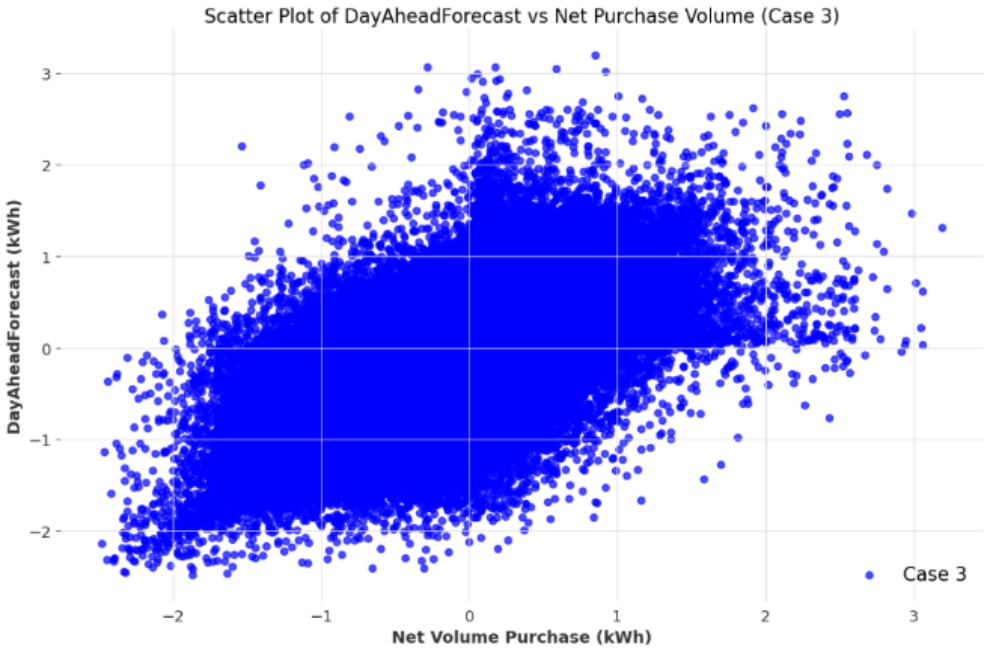


Figure 30 The scatter plot of day ahead forecast over net volume purchase (kWh)- Case3 including PV+HP

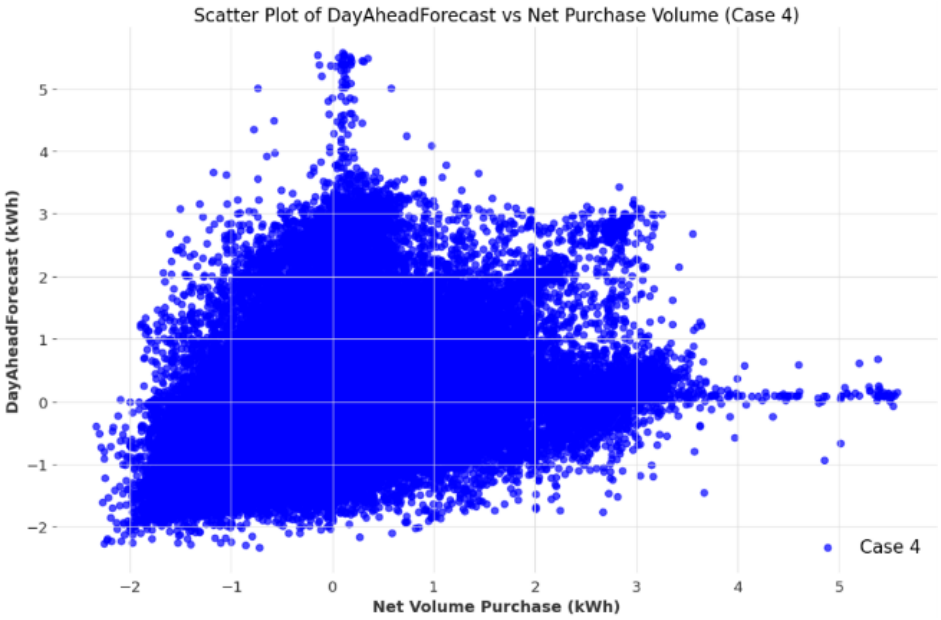


Figure 31 The scatter plot of day ahead forecast over net volume purchase (kWh)- Case4 including PV+EV

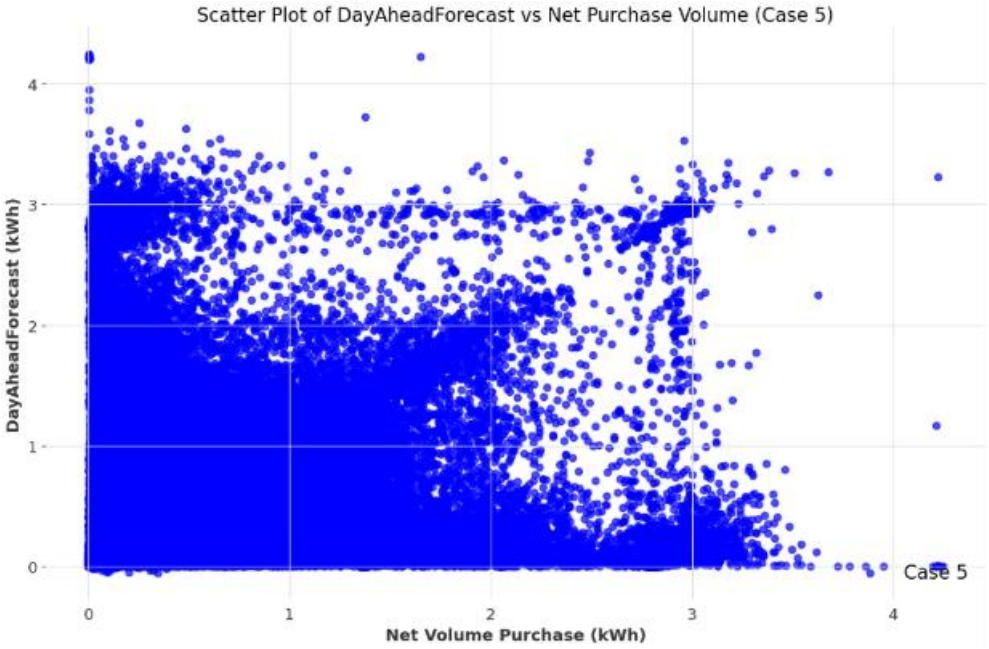


Figure 32 The scatter plot of day ahead forecast over net volume purchase (kWh)- Case5 including EV

As shown in the plot, most building cases exhibit a similar range of net volume purchase. However, buildings using Electric Vehicles (Case 5) and those with no specific installations (Case 2) show no injection volume, as there is no Photovoltaic (PV) system in these cases. On the other hand, the cases involving PV, PV+HP (Heat Pump), and PV+EV (Electric Vehicles) configurations display negative net volume purchase values, which is attributed to the energy produced by the PV systems.

4 RESULTS

The objectives of this thesis are twofold: first, to evaluate the performance of various forecasting models across different electricity consumption patterns in residential buildings to construct a comprehensive dataset; and second, to develop a recommender system that identifies the most suitable forecasting model for each building based on the extracted features of the time series data. To achieve this goal, the four key steps—data processing, forecasting, feature extraction, and the implementation of a recommender system—were carried out as described in Chapter 2. The results are presented and discussed in the following subsections.

4.1 FORECASTING PERFORMANCE

In the context of forecasting, the performance of each model is assessed using the relative mean absolute error (rMAE). Thus, for each forecasting model, an evaluation metric is generated for each building. The values of Net_Volume_Purchase_kWh were aggregated to calculate the annual net electricity purchase volume for each building. Then, the rMAE is plotted against the yearly net purchase volume (kWh) for each forecasting model. Each building case is depicted using unique shapes and colors to highlight the performance of the models for each building, or more specifically, for each electricity consumption pattern.

The following two plots illustrate the performance of forecasting models, both of which rank among the top ten. The remaining eight models are presented in the Appendix.

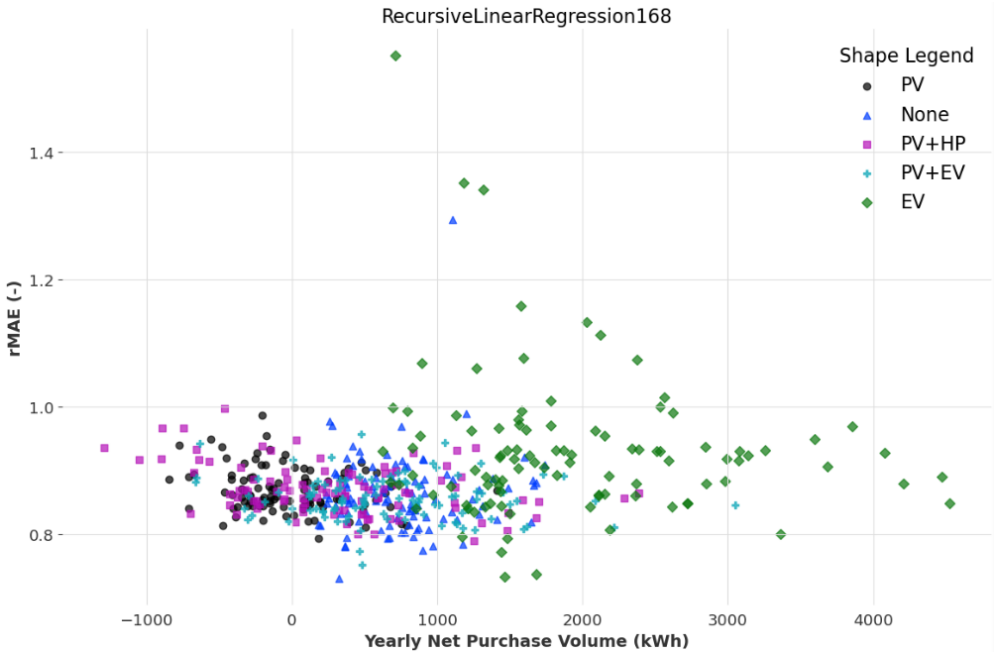


Figure 33 Forecasting evaluation metric (rMAE) over annual net purchase volume (kWh)- RecursiveLinearRegression168

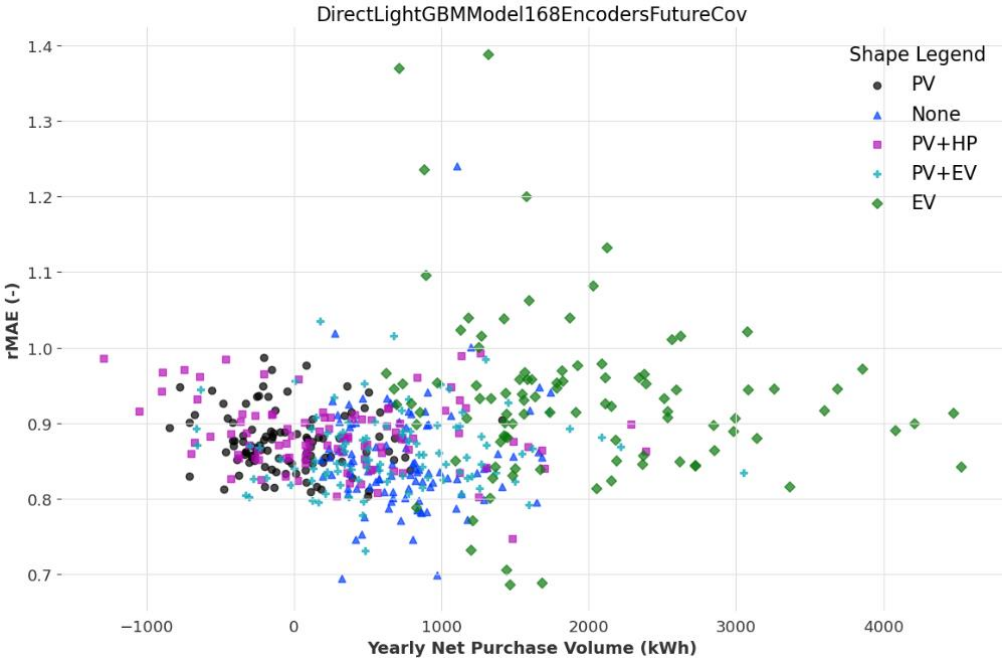


Figure 34 Forecasting evaluation metric (rMAE) over annual net purchase volume (kWh)- DirectLightGBMModel168EncodersFutureCov

The scatter plots illustrate the performance of the prediction models. The y-axis represents rMAE, a forecast evaluation metric, while the x-axis corresponds to the annual net purchase volume. Each point in the graph represents a building. As observed, the majority of buildings exhibit an rMAE of less than 1, indicating that the forecasting models presented—LightGBMModel and LinearRegression—outperform the baseline model and provide pretty accurate predictions. However, some outlier points are noticeable, mainly associated with buildings that include electric vehicle consumption. This observation supports the notion that electric vehicles contribute to significant variations in electricity consumption.

As depicted in the bar chart below, the average rMAE values for each model across all buildings (EAN_IDs) have been computed. A horizontal bar chart is presented, where each bar represents the

average rMAE for a specific model. The y-axis displays the forecasting models' names, while the x-axis represents the corresponding average rMAE values.

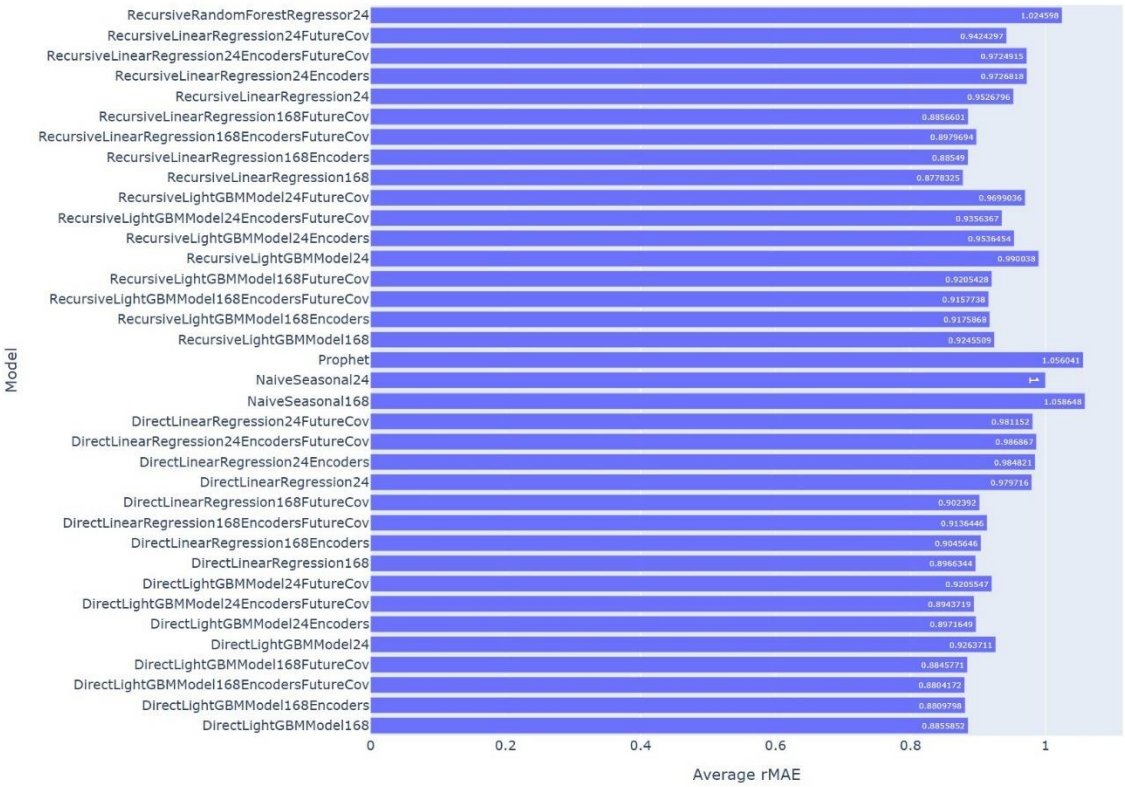


Figure 35 Average forecasting evaluation metric (rMAE) for each forecasting model across 500 buildings

As illustrated in the bar chart, almost all forecasting models outperform the baseline. The baseline model is "naiveseasonal24" and all other models are compared and evaluated against this persistence model. The results, shown in the bar chart, indicate that the baseline model has an average rMAE of 1, while all other models, except for the naiveseasonal168, Prophet, and RecursiveRandomForestRegressor24, achieve a lower average rMAE, demonstrating better performance.

To enhance visualization, a heatmap is generated and presented below. The y-axis represents the forecasting models, while the x-axis corresponds to five distinct building cases, each case comprising 100 buildings (EAN_IDs). A color scale is incorporated to illustrate rMAE values, ranging from 0 to 4.

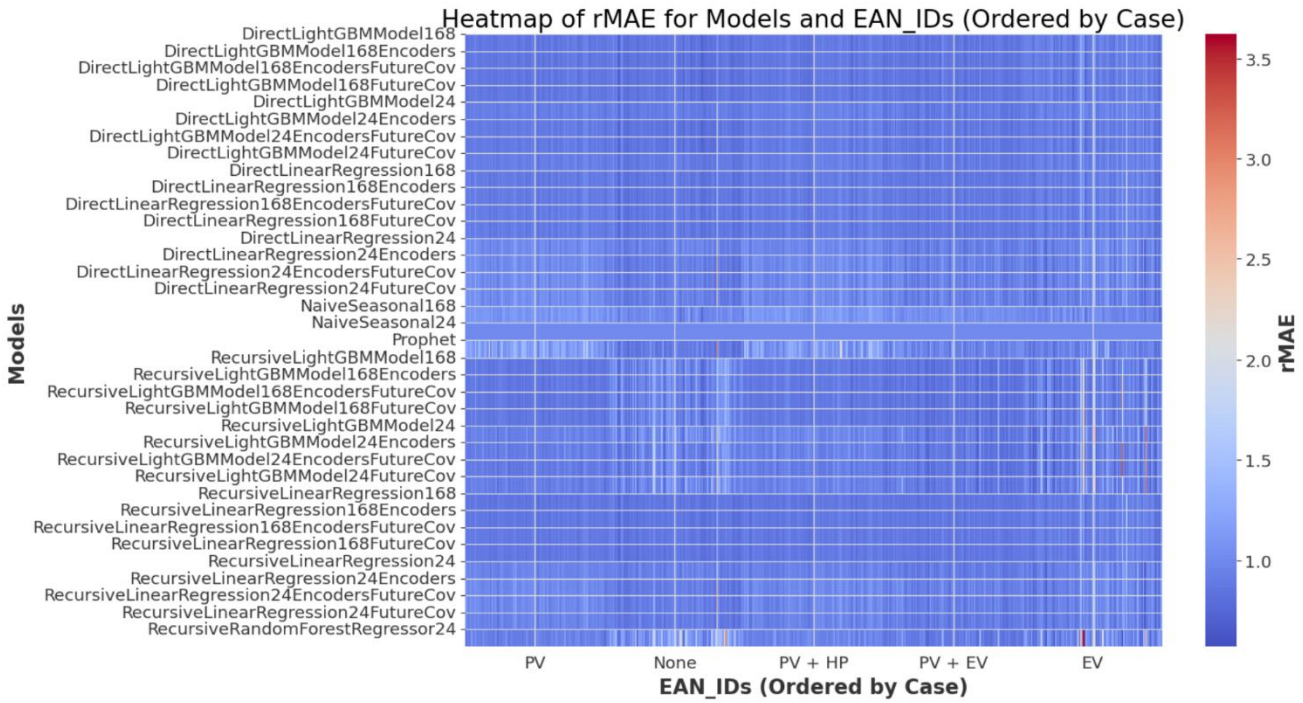


Figure 36 The heatmap of average forecasting evaluation metric (rMAE) for each forecasting model across building cases including PV, None (No specific installation), PV+HP, PV+EV and EV

The darkest blue corresponds to the lowest rMAE, while the darkest red represents the highest rMAE. Therefore, cells that appear darker blue indicate that the corresponding model outperforms others for the respective building cases.

In the subsequent step, the results are presented using a boxplot. A boxplot is a graphical representation that summarizes the distribution of a dataset by displaying its minimum, first quartile (Q1), median, third quartile (Q3), and maximum values. It also highlights any potential outliers in the data. The central box in the plot represents the interquartile range (IQR), which contains the middle 50% of the data. The line inside the box indicates the median value, while the whiskers extending from the box show the range of the data within 1.5 times the IQR. Data points outside this range are considered outliers and are displayed individually. Boxplots provide a concise view of the data's spread, skewness, and presence of outliers, making them a useful tool for visualizing the variation in rMAE values across different forecasting models.

Figure 37 The boxplot of forecasting evaluation metric (rMAE) for each forecasting model- Each colour represents a model group including basic and modified models

In the boxplot, the x-axis represents rMAE values, while the y-axis corresponds to the different prediction models. Each boxplot represents the distribution of rMAE values for a specific forecasting model, highlighting the overall spread of errors. Additionally, certain individual data points fall outside the expected range and are classified as outliers. Outliers in a boxplot are conventionally identified as values that exceed 1.5 times the interquartile range (IQR). As observed, certain forecasting models exhibit a higher number of outliers compared to others, despite being applied to the same dataset. The presence of outliers in the boxplot can be attributed to several factors:

- Model Sensitivity to Data Variability: Some models demonstrate a higher sensitivity to fluctuations in the dataset, resulting in greater deviations in their predictions.
- Model Complexity and Overfitting: More complex models, particularly those incorporating encoders and future covariates, may overfit specific patterns within the data, leading to a broader distribution of errors.

- Prediction Stability: Certain models produce more stable forecasts, yielding a narrower error distribution and consequently fewer outliers.

In the plot, different colors differentiate groups of models, including their basic and modified versions. Furthermore, the five best-performing models, identified based on the lowest rMAE values, are highlighted with a red rectangle. Notably, these top-performing models exhibit shorter boxplots with minimal outliers, which are not widely dispersed, indicating greater prediction stability.

4.2 Recommender System Accuracy

The recommender system is assessed using two evaluation metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). These metrics are employed to analyze the distribution of errors in the data by examining the relationship between the two metrics.

Since MAE and RMSE are aggregated summary statistics that condense the overall error into single values, analyzing raw and squared errors provides a more comprehensive understanding of the full distribution of errors rather than relying solely on a single metric. Raw errors reveal absolute deviations before computing MAE, while squared errors highlight the extent to which large errors influence RMSE. This detailed examination offers deeper insights into the nature of the errors, allowing for a more informed interpretation before merely reporting MAE and RMSE.

The histograms below depict the distribution of raw errors and squared errors, offering a visual representation of their dispersion and variability. Additionally, each plot includes a line over the bar chart, representing the Kernel Density Estimate (KDE) curve. The KDE curve serves as an estimate of the probability density function (PDF) of the data, smoothing the histogram to provide a continuous representation of the underlying distribution. The peaks in the KDE curve highlight regions where values are most concentrated, while the overall shape of the curve provides insights into the skewness, modality, and spread of the errors.

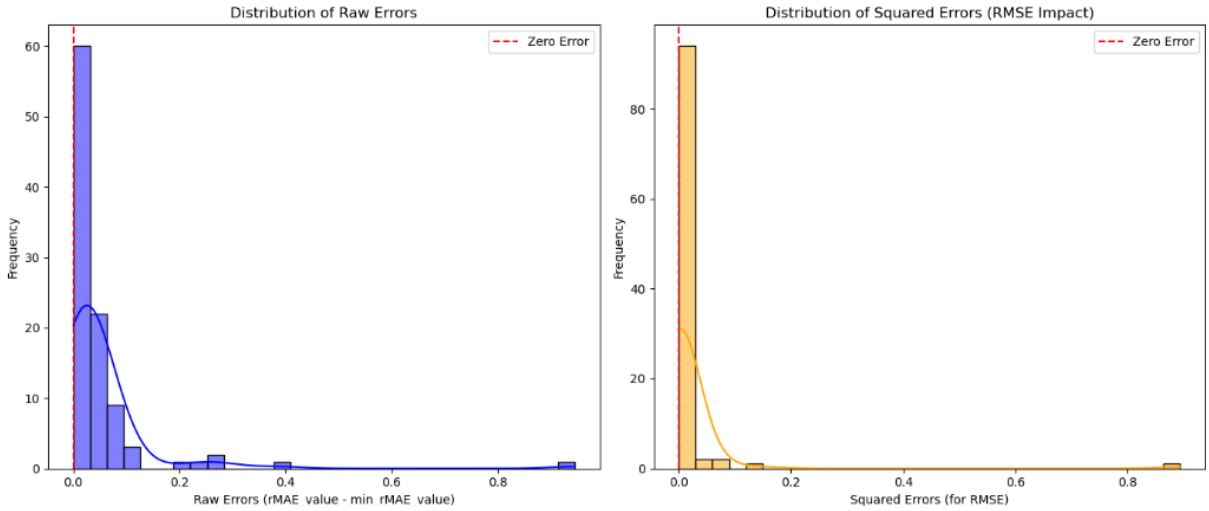


Figure 38 The bar plot of frequency over (Left graph) raw errors and (Right graph) squared errors- Representing the distribution of raw errors and squared errors respectively

In the graphs, the y-axis represents frequency, how many times a particular error value (or a range of values) occurs in the dataset. The taller the bar, the more data points fall in that error range. The x-axis displays raw error and squared error values separately. The red dashed line at zero indicates the ideal case where there is no error, and the line over the bar chart represents the Kernel Density Estimate (KDE) curve.

In the left plot, the raw error histogram naturally contains both small and large errors, so its tail is more visible in its original form. The majority of errors are concentrated near zero, with a long tail extending to the right. This suggests that most predictions made by the recommender system are close to the optimal values, with only a few significant deviations caused by outliers. In the right plot, the frequency of squared errors near zero is higher, making the bar taller compared to raw errors. Since most errors are small (between 0 and 1), squaring them makes them even smaller and pushes them closer to zero. The right histogram bars indicate that squared errors for extreme values occur less frequently; however, when they do appear, they exhibit significantly higher magnitudes.

In this revised version, both raw errors and squared errors are presented within the same figure, enabling a direct comparison between their distributions.

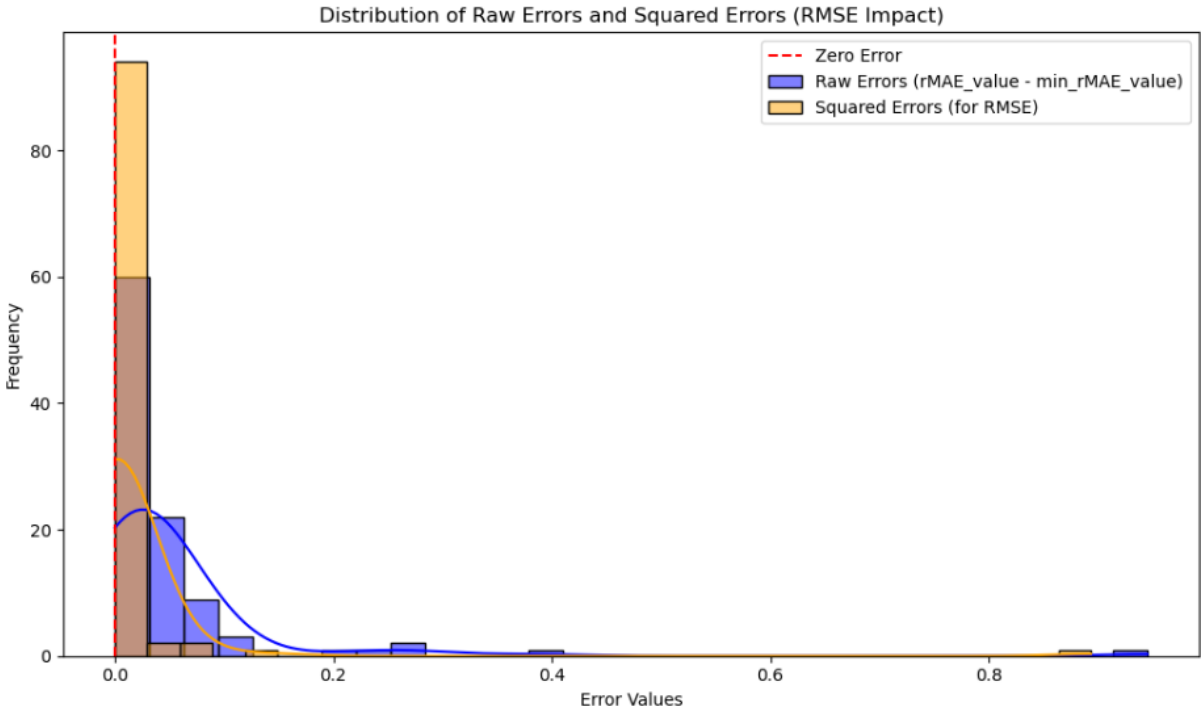


Figure 39 The bar plot of frequency over error values- Blue colour represents the distribution of raw errors and Yellow colour represents the distribution of squared errors

Raw errors are generally small (mostly less than 1), and when squared, the resulting squared errors tend to be even smaller. As a result, most squared errors are smaller in magnitude than their corresponding raw errors because squaring small numbers further reduces their values. However, large raw errors are infrequent in the dataset, so the squared errors for large raw errors are also rare. While the tail of the squared error distribution is influenced by these occasional larger squared values, it does not extend as far as the tail of the raw error distribution due to the infrequency of large raw errors. For small raw errors (less than 1), squaring them results in even smaller values, and since most raw errors are small, this further reduces their contribution to the overall error distribution. Consequently, the tail of the squared error KDE is shorter.

Furthermore, a boxplot is provided to visualize the distribution of MAE and RMSE. The box represents the interquartile range (IQR), capturing the middle 50% of the data.

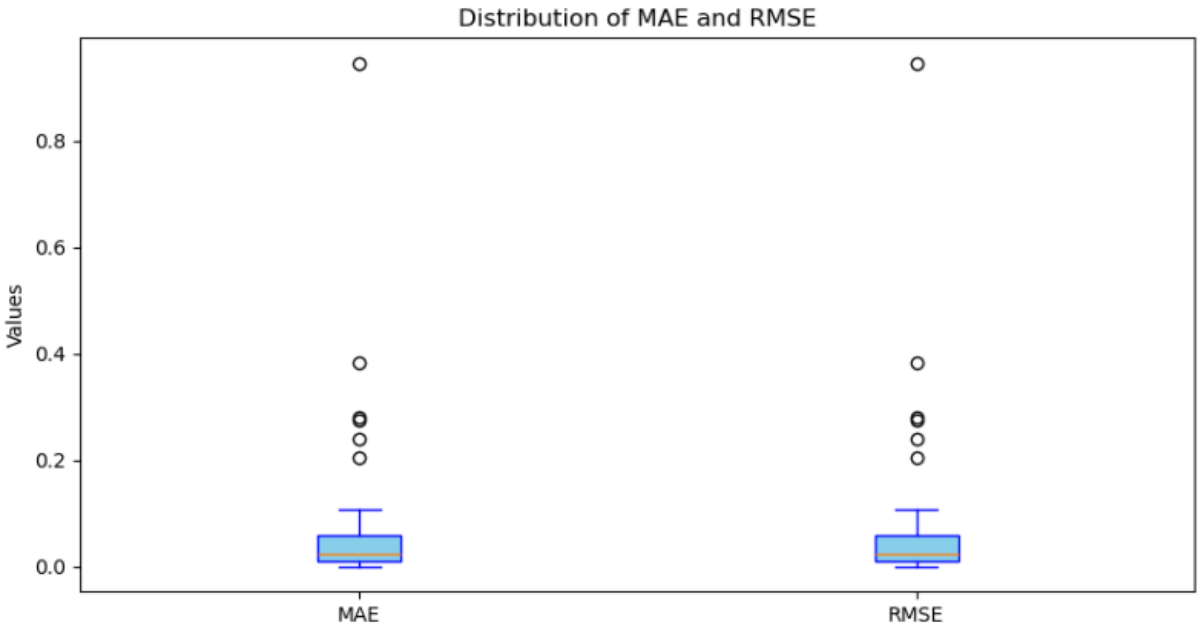


Figure 40 The boxplot of recommender system evaluation metrics- (Left boxplot) represents the distribution of mean absolute error (MAE) and (Right boxplot) represents the distribution of root mean squared error (RMSE) distribution

As observed, both MAE and RMSE exhibit similar distributions. The median of RMSE is same as of MAE, indicating that prediction errors are small and relatively consistent, the squared errors in RMSE are very close to the absolute errors in MAE. This is common when there are no large outliers, or the dataset has low variance. The interquartile range (IQR) of both metrics indicates that the error distributions are relatively similar. As illustrated in Fig. 37, the errors are tightly clustered around zero with minimal dispersion, the squared values used for RMSE remain close to the absolute values used for MAE, resulting in a negligible difference between MAE and RMSE.

Both MAE and RMSE exhibit a few outliers, represented by dots above the whiskers, indicating instances where the errors are considerably larger than usual. Notably, the most extreme outliers appear similar for both metrics. Since the errors are generally below 1, squaring them does not substantially amplify their magnitude, resulting in RMSE values that closely resemble those of MAE. If RMSE was strongly affected by large errors, its boxplot would appear noticeably higher than that of MAE.

To present the exact values of MAE and RMSE on average, the following plot is provided.

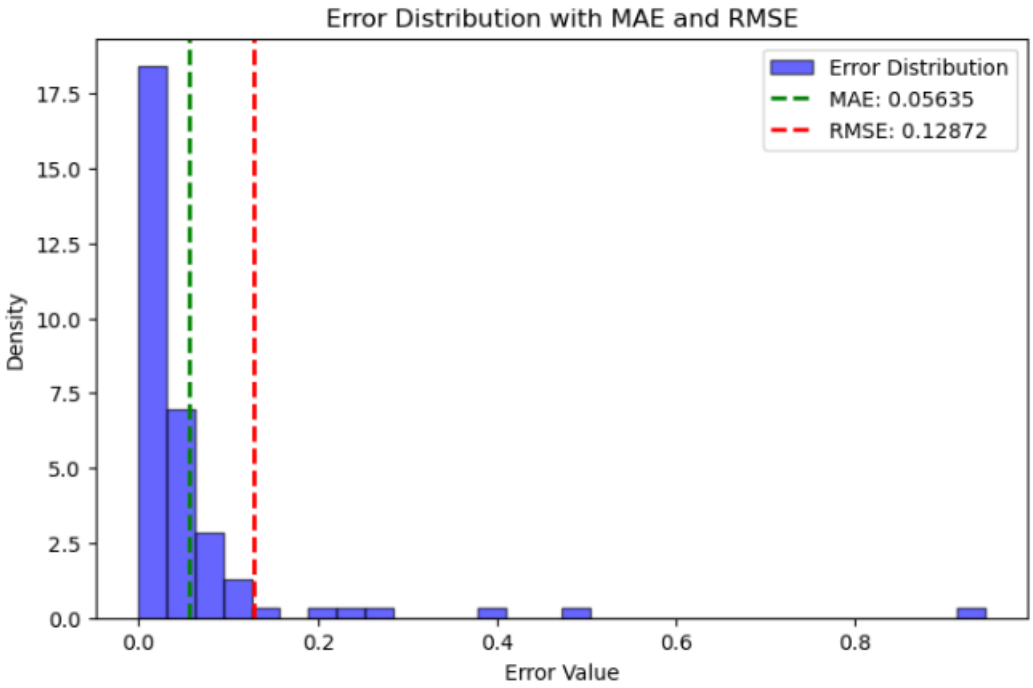


Figure 41 The histogram of density over error values- Representing error distribution along with mean absolute error (MAE) and root mean squared error (RMSE) as reference lines.

The histogram reveals a highly right-skewed distribution, indicating that while most errors are small, a few large errors are present. The green line (MAE = 0.05635) is positioned closer to zero, representing the average absolute error. In contrast, the red line (RMSE = 0.12872) is further to the right, reflecting the impact of squaring errors, which amplifies larger deviations and results in a higher average value for RMSE compared to MAE. (Both MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) measure the average discrepancy between predicted and actual values).

4.3 Discussion of Results

Analyzing the data and results provides a clear understanding of the relationships within the dataset and how they contribute to the observed outcomes. By examining these connections, it becomes evident how specific patterns and characteristics in the data influence the resulting trends and distributions. This analysis not only clarifies the underlying factors driving the results but also enhances the interpretation of key findings, offering deeper insights into their implications.

- The bar chart (Figure 33) depicts forecasting models over average rMAE, nearly all forecasting models outperform the baseline models, achieving an improvement of approximately 12% in the best-case scenario. Among the applied models, the two top-performing ones are RecursiveLinearRegression168 and DirectLightGBMModel168EncodersFutureCov.
- The scatter plots (Figure31, Figure 32) further validate the performance of these models. The majority of data points, each representing a building, are positioned below an rMAE of 1, indicating that the forecasting model outperforms the baseline (rMAE = 1). Additionally, some outliers are observed in these plots, most of which correspond to the building case-5, which includes buildings with electric vehicle consumption.
- The heatmap (Figure 34) supports this conclusion. As indicated by the heatbar, where the best-performing models are represented in dark blue, the heatmap reveals that most cells are darker compared to those corresponding to the baseline forecasting models. Notably, the two forecasting models, RecursiveLinearRegression168 and DirectLightGBMModel168EncodersFutureCov, exhibit the darkest cells across all building cases, reinforcing their superior performance.
- The boxplot (Figure 35) of forecasting models illustrates that, overall, the most rMAE values range approximately from 0.7 to 0.95. The presence of outliers indicates that some rMAE values are higher than most of the model's results. However, for the two best-performing models, DirectLightGBMModel168EncodersFutureCov and RecursiveLinearRegression168, the outliers are not significantly higher compared to the majority of results. Moreover, the boxplots for these models are notably shorter than those of other forecasting models, suggesting more consistent and stable performance.
- The recommender system was evaluated using two metrics, both of which demonstrated its strong performance. The average MAE of the suggested models compared to the actual best forecasting model for a building is 0.05635, while the average RMSE, which emphasizes the impact of larger errors, is 0.12872. These results highlight the accuracy of the recommender system in identifying the most suitable forecasting models for an unknown building.

5 CONCLUSION

The main objective of this research was to develop machine learning models for forecasting building energy demand on a day-ahead timescale while addressing a fundamental challenge in time-series forecasting—identifying the most suitable predictive model for a given dataset.

To achieve this, the study pursued two key phases: first, to construct an extensive dataset that captures the performance of various forecasting models across diverse building electricity consumption patterns; and second, to design a recommender system capable of identifying the most appropriate forecasting model for an unknown building based on extracted time-series features. Therefore, a comprehensive methodology was established, encompassing data preprocessing, feature extraction, forecasting, performance evaluation, and model recommendation.

The dataset used in this research consists of electricity consumption data from 500 buildings in Brussels, obtained from the Fluvius platform, while weather-related covariates were retrieved from a weather API. To conduct the procedure, various machine learning algorithms were implemented within a Python-based analytical framework. The process began with data preprocessing, followed by the implementation of over thirty forecasting models, both in their standard forms and with specific modifications. This approach was further refined through extensive data analysis and the extraction of relevant time-series features, which were subsequently leveraged in developing the recommender system.

Subsequently, the performance of the forecasting models was compared against persistence models across five building categories, including electricity demand patterns associated with photovoltaic panels, photovoltaic panels combined with heat pumps, photovoltaic panels with electric vehicles, standalone electric vehicles, and buildings without specific high-consumption appliances. The evaluation was carried out using the relative mean absolute error (rMAE). Nearly all models outperformed the baseline, with the best-case scenario achieving an improvement of approximately 12%. The two top-performing models across all building cases were RecursiveLinearRegression168 and DirectLightGBMModel168EncodersFutureCov. Furthermore, the recommender system successfully identified the most suitable forecasting model for a new building based on its time series features, achieving a mean absolute error (MAE) of 0.05. This low error value indicates a minimal discrepancy between the predicted and actual values, demonstrating the system's effectiveness in accurately selecting the optimal prediction model for an unseen building.

In this study, a straightforward approach was adopted by employing relatively simple forecasting models and designing a basic recommender system. This forecasting framework resulted in the

creation of a comprehensive dataset, eliminating the need to perform many forecasting models for each new building individually. The recommender system subsequently suggests the most suitable forecasting model for buildings with unknown energy demand patterns, significantly reducing computational time while ensuring accurate model selection even in the absence of metadata.

However, it is important to recognize that expanding the dataset by incorporating additional forecasting models and increasing the number of buildings would create a more extensive and comprehensive dataset, ultimately leading to cover more prediction models and more electricity demand patterns. This may involve implementing more advanced machine learning algorithms and models capable of capturing complex data patterns with higher accuracy, thereby achieving a more substantial performance improvement over the baseline models. Various sophisticated models account for nonlinear behaviour, and some include hyperparameters that can be fine-tuned to optimize performance. Additionally, incorporating a wider range of building energy demand patterns would enhance the recommender system's ability to provide accurate model recommendations across diverse energy consumption profiles. However, it is important to acknowledge that such extensions go beyond the scope and objectives of the current study.

Furthermore, the recommender system was designed to identify the most suitable forecasting model based on the time series characteristics of a building without prior specific information. To enhance the accuracy and reliability of this system, a more comprehensive dataset with higher precision is required, which can be achieved through the integration of additional methodologies.

REFERENCES

Ada Canaydin, Chun Fu, Attila Balint, Mohamad Khalil, Clayton Miller, & Hussain Kazmi. (n.d.). *Interpretable domain-informed and domain-agnostic features for supervised and unsupervised learning on building energy demand data*.

Consumption profiles of digital electricity meters. (n.d.). [Dataset].

Correlation in machine learning. (n.d.). https://medium.com/@abdallahashraf90x/all-you-need-to-know-about-correlation-for-machine-learning-e249fec292e9

Darts. (n.d.). [Computer software].

Desiree Arias-Requejo, Carlos J. Alonso-Gonzalez, & Belarmino Pulido. (n.d.). *Energy demand prediction in smart buildings using advanced machine learning techniques*.

Dr.S.Meenakshi, Dr.T.Papitha Christobel, Dr.Harshini Manoharan, & Monisha N. (n.d.). *Enhanced Electricity Demand Prediction Using Machine Learning Techniques*.

Feature Engineering. (n.d.). https://builtin.com/articles/feature-engineering#:~:text=Feature%20engineering%20is%20a%20machine,while%20also%20enhancing%20model%20accuracy.

Five machine learning types to know. (n.d.).

Historical Weather API. (n.d.). [Dataset].

How to Make Baseline Predictions for Time Series Forecasting with Python. (n.d.). https://machinelearningmastery.com/persistence-time-series-forecasting-with-python/

Hussain Kazmi & Ada Canaydin. (n.d.). *Time series feature extraction and clustering for building energy demand*.

Hussain Kazmi, Chun Fu, & Clayton Miller. (n.d.). *Ten questions concerning data-driven modelling and forecasting of operational energy demand at building and urban scale*.

Kazmi Hussain & Balint Attila. (n.d.). *Forecasting energy data*.

K-Nearest Neighbor. (n.d.). https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

Matplotlib. (n.d.). [Computer software]. https://matplotlib.org/3.5.3/api/matplotlib_configuration_api.html

NumPy. (n.d.). [Computer software].

Pandas. (n.d.). [Computer software].

Python. (n.d.). [Computer software].

Recommendation System. (n.d.). https://www.nvidia.com/en-us/glossary/recommendation-system/

Recommender system. (n.d.).

Scikit-learn. (n.d.). [Computer software].

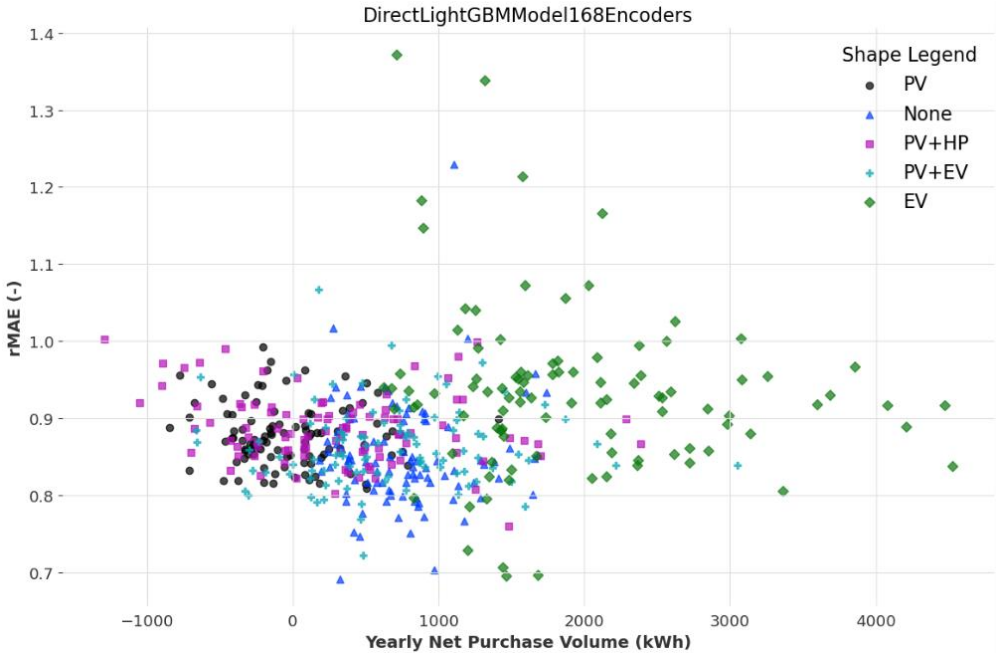
Thiyanga S Talagala, Rob J Hyndman, & George Athanasopoulos. (n.d.). *Meta-learning how to forecast time series*.

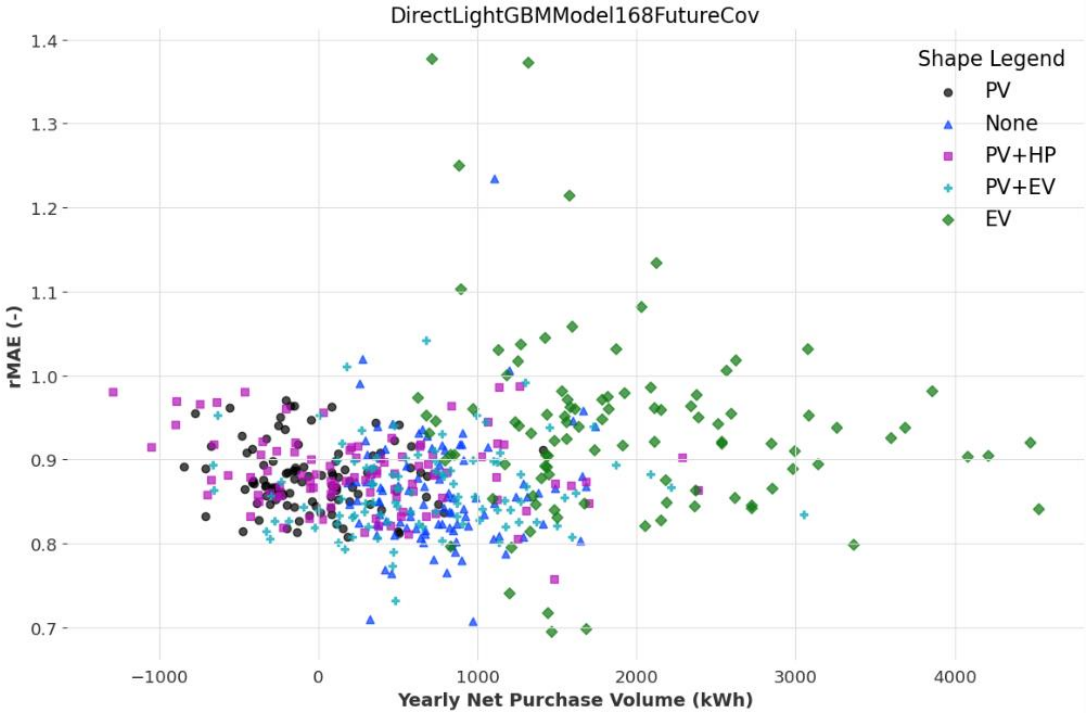
Tracking Clean Energy Progress 2023. (n.d.). https://www.iea.org/reports/tracking-clean-energy-progress-2023

APPENDICES

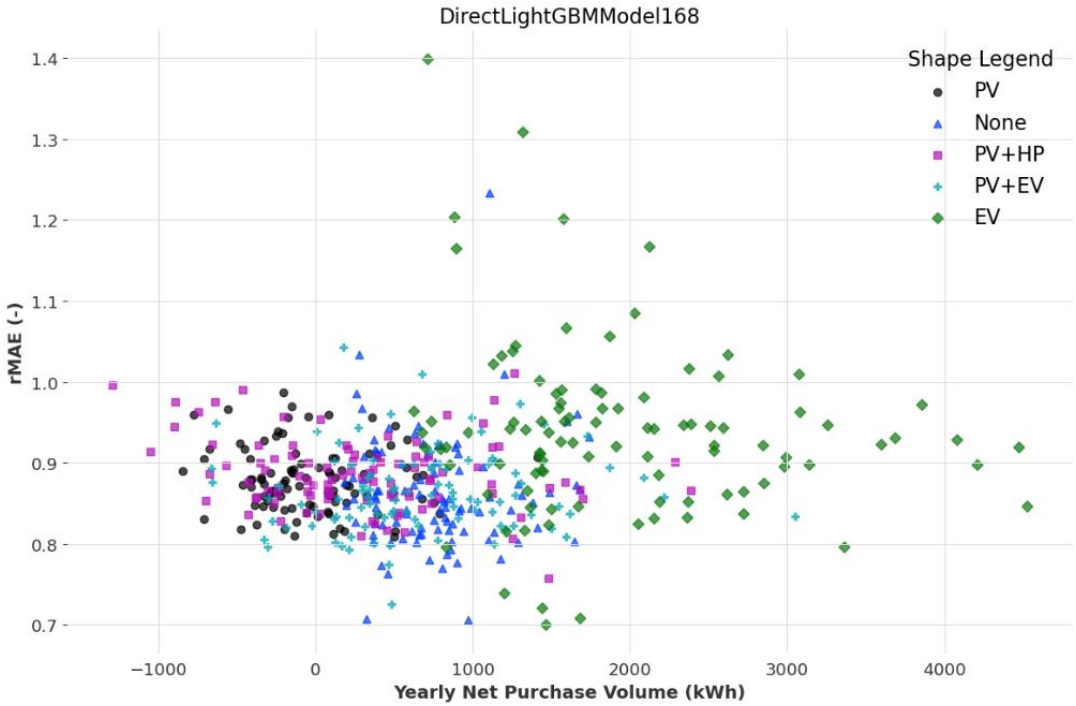
Appendix

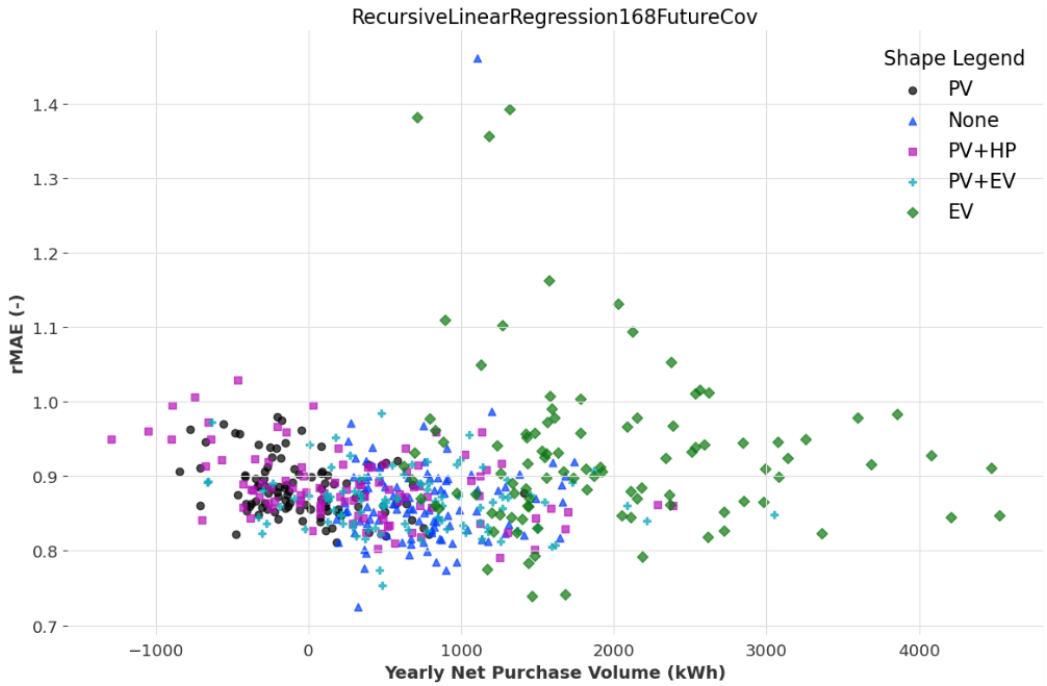
This appendix presents the remaining eight plots depicting the forecasting evaluation metric (rMAE) over the annual net purchase volume (kWh) for the top ten forecasting models, further supporting the findings discussed in Chapter 4.



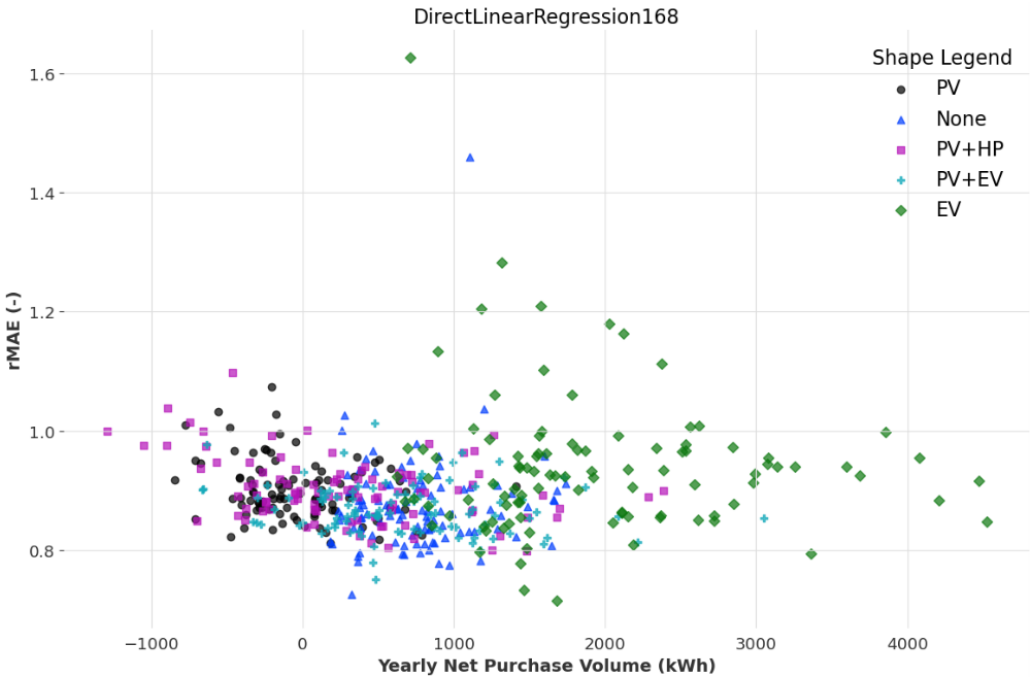


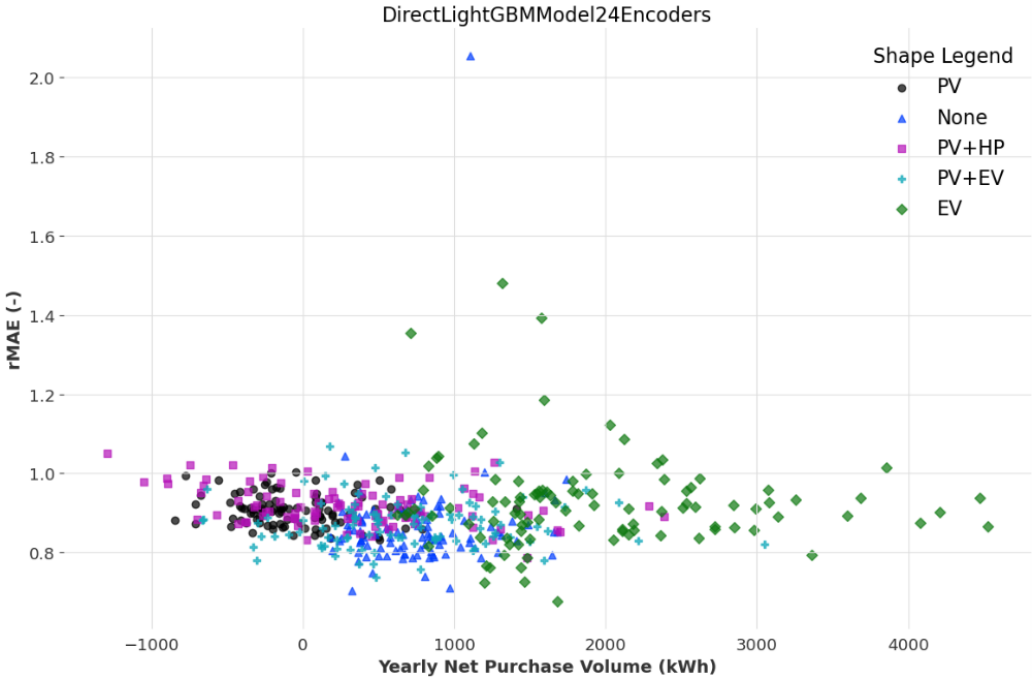

RecursiveLinearRegression168Encoders
Shape Legend
PV
None
PV+HP
PV+EV
EV
rMAE (-)
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
−1000
0
1000
2000
3000
4000
Yearly Net Purchase Volume (kWh)


DirectLightGBMModel168
Shape Legend
PV
None
PV+HP
PV+EV
EV
rMAE (-)
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
−1000
0
1000
2000
3000
4000
Yearly Net Purchase Volume (kWh)


RecursiveLinearRegression168FutureCov
Shape Legend
PV
None
PV+HP
PV+EV
EV
rMAE (-)
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
−1000
0
1000
2000
3000
4000
Yearly Net Purchase Volume (kWh)


DirectLightGBMModel24EncodersFutureCov
Shape Legend
PV
None
PV+HP
PV+EV
EV
rMAE (-)
1.8
1.6
1.4
1.2
1.0
0.8
−1000
0
1000
2000
3000
4000
Yearly Net Purchase Volume (kWh)


DirectLinearRegression168
Shape Legend
PV
None
PV+HP
PV+EV
EV
rMAE (-)
1.6
1.4
1.2
1.0
0.8
−1000
0
1000
2000
3000
4000
Yearly Net Purchase Volume (kWh)


DirectLightGBMModel24Encoders
Shape Legend
PV
None
PV+HP
PV+EV
EV
rMAE (-)
2.0
1.8
1.6
1.4
1.2
1.0
0.8
−1000
0
1000
2000
3000
4000
Yearly Net Purchase Volume (kWh)