UNIVERSIDADE PORTUCALENSE
DEPARTAMENTO
CIÊNCIA E TECNOLOGIA

# Time-series Forecasting of Electricity Prices for Industries in Portugal: A Real-world Internship-based Study

Master in Data Science | October 2025

Supervision | Prof. Doutora Isabel Seruca

Naiya Khalid | 50768

# Time-series Forecasting of Electricity Prices for Industries in Portugal: A Real-world Internship-based Study

**Naiya Khalid**

Master in Data Science

Master's Internship Project Report submitted to the Universidade Portucalense Infante D. Henrique to obtain the degree of Master in Data Science, carried out under the supervision of Doctor Isabel Seruca, Associate Professor of the Department of Science and Technology of Universidade Portucalense.

Department of Science and Technology, Universidade Portucalense
October 2025

# DEDICATION

To my beloved late father, whose memory continues to inspire me.

To my mother, whose unwavering dedication, boundless love, and gentle guidance have profoundly shaped every step of my journey.

To my husband, for his extraordinary support and unwavering belief that uplifts me daily.

To my sister, who supported me and encouraged me to choose the data science field.

And to my son, my greatest joy, who gives me the strength to persevere and pursue my passions.

# ACKNOWLEDGEMENTS

I would like to begin by expressing my deepest gratitude to Professor Isabel Seruca, whose guidance and unwavering support throughout this internship and thesis writing process have been invaluable. Her insightful feedback and encouragement were instrumental in the successful completion of this work.

I am profoundly thankful to my parents and my husband for their unconditional love, encouragement, and the values they instilled in me, which have been the foundation of my academic journey.

I also wish to sincerely thank my internship supervisors, Mr. Dinu Granci and Mr. Mário Granci, for their mentorship, support, and patience throughout the course of the project.

I also acknowledge the support of my team members, Raza Mehar and Athul Raj, for their collaboration and encouragement during the internship.

I extend my appreciation to the Portucalense University and its faculty for their dedication to my academic and personal development, and for providing the knowledge and tools that helped shape this thesis.

Finally, I would like to express my sincere thanks to everyone who, directly or indirectly, contributed to the realization of this work.

# ABSTRACT

This work presents the development of a short-term electricity price forecasting solution using machine learning, conducted during the author's internship at Vanaci Prime, an IT consulting firm specializing in data-driven solutions. The project was developed for the Portuguese market, in collaboration with a client from the energy sector, and addresses the critical need for accurate, real-time, and scalable electricity price forecasts to support strategic planning, cost optimization, and operational decision-making.

The forecasting system is built upon a unified dataset combining information from three authoritative sources: OMIE (day-ahead marginal electricity prices for the Iberian market), REN Datahub (electricity load and renewable generation), and the Copernicus Climate Data Store (meteorological variables including temperature, solar radiation, and precipitation). The combined data underwent extensive time-series preprocessing and feature engineering, including lag features, rolling statistics, calendar-based encodings, interaction terms, and normalized indicators reflecting load trends and solar activity.

Multiple machine learning models were explored—namely LightGBM, XGBoost, and Extra Trees—with LightGBM ultimately selected due to its superior performance and generalization capability.

A complete machine learning pipeline in Python was developed to automate the transformation of raw input data into model-ready features. To enhance the solution's practical utility, a consumer-focused component was added to estimate potential monthly savings based on predicted electricity prices.

The trained model and preprocessing pipeline are serialized and deployed using the FastAPI web framework to serve real-time predictions via web endpoints.

Overall, the work demonstrates how artificial intelligence techniques can be used to automate and optimize electricity price forecasting by integrating environmental, market, and load data, while offering a replicable methodology for similar forecasting challenges in the energy sector.

**Keywords:** Electricity price forecasting, Machine learning, Time series

# RESUMO

Este trabalho apresenta o desenvolvimento de uma solução de previsão de preços de eletricidade a curto prazo utilizando aprendizagem automática, realizado durante o estágio da autora na Vanaci Prime, uma empresa de consultoria em TI especializada em soluções orientadas por dados. O projeto foi desenvolvido para o mercado Português, em colaboração com um cliente do setor energético, e responde à necessidade crítica de previsões de preços de eletricidade precisas, em tempo real e escaláveis, para apoiar o planeamento estratégico, a otimização de custos e a tomada de decisões operacionais.

O sistema de previsão baseia-se num conjunto de dados unificado, que combina informações de três fontes oficiais: OMIE (preços marginais de eletricidade para o dia seguinte relativos ao mercado Ibérico), REN Datahub (consumo de electricidade e produção de energias renováveis) e o Copernicus Climate Data Store (variáveis meteorológicas como temperatura, radiação solar e precipitação). Os dados combinados foram sujeitos a um extenso pré-processamento de séries temporais e engenharia de atributos, incluindo variáveis defasadas, estatísticas móveis, codificações baseadas no calendário, termos de interação e indicadores normalizados que refletem tendências de carga e atividade solar.

Foram explorados vários modelos de aprendizagem automática—nomeadamente LightGBM, XGBoost e Extra Trees—tendo o modelo LightGBM sido selecionado pelo seu melhor desempenho e capacidade de generalização.

Foi desenvolvida uma pipeline completa de aprendizagem automática em Python para automatizar a transformação de dados brutos em atributos prontos para o modelo. Para aumentar a utilidade prática da solução, foi adicionada uma componente centrada no consumidor, capaz de estimar potenciais poupanças mensais com base nos preços de eletricidade previstos.

O modelo treinado e a pipeline de pré-processamento foram serializados e implementados através do framework web FastAPI para fornecer previsões em tempo real via endpoints web.

De uma forma geral, este trabalho demonstra como técnicas de inteligência artificial podem ser utilizadas para automatizar e otimizar a previsão de preços de eletricidade através da integração de dados ambientais, de mercado e de carga, disponibilizando uma metodologia replicável para desafios semelhantes no setor energético.

**Palavras-chave**: Previsão de preços de eletricidade, Aprendizagem automática, Séries temporais

# Table of Contents

# LIST OF FIGURES

# LIST OF TABLES

# GLOSSARY

| Abbreviation | Full Form / Meaning |
|---|---|
| AdaBoost | Adaptive Boosting |
| AEMPF / SEMPF | Advanced / Smart Electricity Market Price Forecasting |
| ANN / ANNs | Artificial Neural Network(s) |
| API | Application Programming Interface |
| AR | Autoregressive |
| ARIMA | Autoregressive Integrated Moving Average |
| ARMA | Autoregressive Moving Average |
| BPNN | Backpropagation Neural Network |
| CDS | Copernicus Climate Data Store |
| CNN / CNNs | Convolutional Neural Network(s) |
| CNN-GRU | Hybrid Convolutional and Gated Recurrent Unit Network |
| CNN-LSTM | Hybrid Convolutional and Long Short-Term Memory Network |
| DL | Deep Learning |
| DNN / DNNs | Deep Neural Network(s) |
| EEMD | Ensemble Empirical Mode Decomposition |
| EDA | Exploratory Data Analysis |
| Elastic Net | Regularized linear model combining L1 and L2 penalties |
| ELM | Extreme Learning Machine |
| EMPF | Electricity Market Price Forecasting |
| EPF | Electricity Price Forecasting |
| EU | European Union |
| Extra Trees | Extremely Randomized Trees |
| FFNN / FFNNs | Feedforward Neural Network(s) |
| GA | Genetic Algorithm (used for optimization with LSSVM) |
| GARCH | Generalized Autoregressive Conditional Heteroskedasticity |
| GARCH-t / SV-t | GARCH or SV models with t-distribution errors |
| GBRT | Gradient Boosted Regression Trees |
| GDP | Gross Domestic Product |
| GRU | Gated Recurrent Unit |
| GWh | Gigawatt-Hour |
| GW | Gigawatt |
| IQR | Interquartile Range |
| LASSO | Least Absolute Shrinkage and Selection Operator |
| LEAR | Linear Exponential Autoregressive Model |
| LightGBM | Light Gradient-Boosting Machine |
| LR | Linear Regression Model |

| Abbreviation | Full Form / Meaning |
|---|---|
| LSSVM | Least Squares Support Vector Machine |
| LSTM | Long Short-Term Memory |
| LSTMs | Long Short-Term Memory Networks |
| MAE | Mean Absolute Error |
| MAPE | Mean Absolute Percentage Error |
| MJ | Megajoule |
| ML | Machine Learning |
| MLP | Multilayer Perceptron |
| MW / $MW_p$ | Megawatt / Peak Megawatt |
| NVI | Net Volume Index |
| OMIE | Operador del Mercado Ibérico de Energía |
| Optuna | Automated hyperparameter optimization framework |
| PPP | Purchasing Power Parity |
| Q-Q | Quantile-Quantile (Plot/Test) |
| RBF | Radial Basis Function (kernel) |
| REMPE | Renewable Electricity Market Price Estimation |
| REN | Rede Eléctrica Nacional (Portuguese Datahub) |
| RF | Random Forest |
| RMSE | Root Mean Square Error |
| RVM | Relevance Vector Machine |
| SARIMA | Seasonal Autoregressive Integrated Moving Average |
| SDAC | Single Day-Ahead Coupling |
| SDAP | System Day-Ahead Price |
| SMP | System Marginal Price |
| Soft-Voting Ensemble | Ensemble technique averaging probabilities from multiple models |
| Stacking Ensemble | Ensemble technique combining predictions via meta-learner |
| SV | Stochastic Volatility |
| SVM | Support Vector Machine |
| Transformer / MHA | Multi-Head Attention Transformer Model |
| XGBoost | Extreme Gradient Boosting |

# 1. Chapter 1: Introduction

## 1.1 Overview and relevance of Electricity price forecasting

Electricity price forecasting (EPF) is an interdisciplinary field spanning electrical engineering, statistics, computer science, and finance. It focuses on predicting prices in wholesale electricity markets across multiple time horizons. These market timeframes include real-time, day-ahead, and long-term contracts. Day-ahead markets, especially prominent in Europe, are the primary focus of most EPF studies, accounting for nearly 90% of published research (Lago et al, 2021; Weron, 2014).

EPF plays a crucial role in modern energy markets, driven by the increasing penetration of renewables, market liberalization, and demand fluctuations. Since the liberalization of electricity markets, both industry and academia have shown strong interest in EPF. While methods from energy economics and finance are widely applied, their use requires caution due to the unique characteristics of electricity as a traded commodity. Unlike other markets, electricity demand is highly inelastic, shaped by time-specific patterns, and must be balanced precisely in real time to prevent blackouts or grid overload (Lago et al, 2021).

EPF holds vital practical relevance due to the dynamic and sensitive nature of power markets. Despite advancements in forecasting techniques, prices remain difficult to predict accurately because they are influenced by both long-term trends and sudden, short-term fluctuations triggered by market events. These challenges underscore the need for robust forecasting models that can handle volatility and extreme price behavior (Cu et al, 2025).

Various stakeholders in the electricity market, including power generators and electricity consumers, place considerable importance on price forecasting. Generators can use such forecasts to devise bidding strategies, while consumers rely on them to optimize production activities efficiently and reduce electricity costs (Feng et al, 2025).

Hence, accurate EPF is acknowledged as a key issue for the energy sector, supporting both strategic and operational decision-making. It benefits market participants, policymakers, consumers, and sustainability advocates by enabling better financial planning, cost optimization, risk management, and the advancement of sustainability initiatives.

In recent years, short-term electricity price prediction has become increasingly important due to evolving market structures and the rise of intraday trading. Accurate and timely forecasts help market participants manage assets more efficiently, mitigate price volatility, and seize new opportunities in a rapidly changing energy landscape.

## 1.2 Models for Electricity Price Forecasting

Historically, linear regression and early neural networks dominated the EPF field, but with the growth of data and computing power, machine learning methods have since become more prevalent. These changes have significantly influenced how modern EPF models are developed and applied (Jędrzejewski et al, 2022).

Despite the availability of various machine learning and statistical techniques, there is no consensus on which models consistently outperform others. Comparisons are often biased, test periods are too short, and key methodological details are missing in many studies. This lack of reproducibility and fair evaluation makes it difficult to assess true model performance and drives the need for more robust, transparent forecasting frameworks (Lago et al, 2021).

Recent literature on EPF highlights the need for more flexible models and the inclusion of new variables in research design (Gürtler & Paulsen, 2018). As a time series forecasting problem, EPF has benefited from advances in computational power and methods ranging from simple statistical techniques to sophisticated machine learning approaches (Petropoulos et al, 2022). However, much of the existing work remains centered on past and present energy systems, underscoring the need for models that explicitly account for the structural changes driven by the energy transition (Nitsch et al, 2024).

## 1.3 The Portuguese electricity consumption and supply market

In **2024,** most of Portugal's energy was used by the transport sector (about 36%) and industry (around 27%), followed by households (about 17%) and services (around 15%). This shows that transport and industry are still the main energy users in the country. For households, electricity remained the main source of energy, while biofuels, oil products, and natural gas were also used (ADENE, 2024; Direção-Geral de Energia e Geologia, 2024; Odyssee-MURE Project, 2024).

From a macroeconomic perspective, Portugal's total energy supply per unit of GDP (PPP) in 2023 was 1,925.9 MJ per 2015 USD PPP, reflecting a 34% decrease since 2000. In the European context, Portugal ranked 33rd in 2022 with 2,051.5 MJ per 2015 USD PPP, positioned close to Italy and Cyprus but above the EU average of 1,585.6 MJ. These figures emphasize both the progress in energy efficiency and the continuing relevance of electricity in shaping the country's economic and environmental strategies.

The ongoing energy transition is driven by the goal to make the EU climate-neutral by 2050, which is only possible by setting several targets for each country. For example, by 2030, Portugal must have a 45% reduction in greenhouse gas emissions (from 1990 levels),

a share of 47% renewable electricity, and an improvement of 35% in energy efficiency (European Commission, 2025).

As Portugal expands its renewable energy integration, EPF becomes vital for market stability and investment planning. The rising share of intermittent sources like solar and wind increases price volatility, particularly during periods of overgeneration or low demand. According to the Aurora Energy Research (2024), the planned addition of 5 GW of storage by 2030 will help reduce price spikes and negative pricing events. Nevertheless, these structural shifts add complexity to market dynamics, reinforcing the need for robust and adaptable forecasting models (Strategic Energy Europe, 2025).

## 1.4 Motivation and aim of the research

In order to address the shortcomings of existing research, which is closely associated with past and current energy system dynamics, this work proposes a novel assessment of various forecasting techniques. It seeks to address the inherent challenges and the increasing significance of EPF in the context of our contemporary and future energy landscape, namely, considering energy transition issues.

Building on this context, this thesis develops a machine learning–based forecasting solution tailored to the Portuguese electricity market. Using historical data from OMIE, REN Datahub**,** and the Copernicus Climate Data Store, the study applies and optimizes algorithms such as LightGBM, XGBoost, and Random Forest to deliver a practical tool for short-term electricity price prediction. Beyond improving accuracy, the solution is designed for real-world deployment through a FastAPI backend, addressing gaps in the literature on data integration, real-time applicability, and operational implementation.

## 1.5 Objectives of the work

This thesis presents the development of a short-term electricity price forecasting solution using machine learning, carried out during the author's internship at VanaciPrime, an IT consulting firm focused on data-driven solutions (Vanaci Prime, 2025), where the author integrated the development team.

The team was composed of a machine learning engineer, a team leader, and two interns, the author being one of them. The project was developed in collaboration with a client in the energy sector and aimed to develop an accurate machine learning–based forecasting model for Portugal's electricity marginal price, **capable of predicting prices seven days ahead** (`168 hours`). These forecasts are intended to support strategic planning, cost optimization, and operational decision-making.

The dataset used was compiled from three authoritative operational data sources that provide real-time updates:

- Iberian Electricity Market Operator (OMIE-Operador del Mercado Ibérico de Energía) – providing the target variable: hourly day-ahead marginal electricity prices;
- REN Datahub (Rede Eléctrica Nacional) – contributing explanatory variables related to electricity load, generation types (wind, solar, hydro, gas), and system-level operations.
- Copernicus Climate Data Store (CDS) – supplying weather-based features including solar radiation, temperature, and precipitation.

These datasets were collected retrospectively, reflecting real-time operational measurements. The time period of the dataset varied across different stages of the project:

- Exploratory Data Analysis (EDA): A broad dataset covering `20 March 2015` to `20 March 2025` (`10 years`) was used to identify patterns and assess long-term data completeness.
- Initial Modeling and Quality Checks: A more recent period, from `8 May 2020` to `20 March 2025`, was used to focus on higher-quality and more consistent records.
- Final Model, Pipeline, and FastAPI Deployment: The dataset was restricted to `1 January 2024` to `20 March 2025`, aligning with the most relevant and up-to-date period for practical forecasting.

Using this unified dataset, a series of predictive models will be trained and evaluated. The best-performing model will be retrained on the full dataset and serialized for deployment.

The project involved advanced feature engineering, model optimization with Optuna, and deployment through an automated pipeline and FastAPI service. To link the model's predictions to practical value, a savings estimation tool was also developed. Hence, the specific objectives of the thesis were summarized as follows:

- Understand the operational scope and strategic goals of VanaciPrime.
- Analyze the structure and pricing mechanisms of the Portuguese electricity market.
- Design and construct a unified dataset by integrating features from OMIE, REN, and CDS.
- Apply advanced feature engineering techniques (lags, rolling windows, weather interactions, calendar encodings) to capture temporal and seasonal dynamics.
- Train, compare, and tune multiple machine learning models using Optuna with time series–aware cross-validation, selecting the best performer based on MAE.
- Create a reusable machine learning pipeline that automates data selection, feature engineering, and scaling.

- Deploy the model using the FastAPI web framework to expose prediction endpoints, ensuring modular and scalable deployment.
- Develop a savings estimation tool to demonstrate real-world financial impact based on predicted prices.

Through this integrated approach, the thesis combines real-world data engineering, machine learning, and deployment practices, contributing to the broader field of EPF and its application in decision support systems.

## 1.6 Adopted Methodology

This internship project involved several activities that contributed to achieving the objectives defined for the work and presented in the previous section. These activities are presented in Table 1, in chronological order.

*Table 1: Project Roadmap*

| ***Date*** | ***Stage*** | ***Description*** |
|---|---|---|
| *Jan 2025* | 1. Company Integration and Business Understanding | • Introduction to team, equipment setup, company overview.<br>• Analysis of organizational documentation and project scope.<br>• Understand business needs: accurate, real-time electricity price forecasts for planning and cost optimization. |
| *Feb 2025* | 2. Project Initiation and Planning | • Define objectives and scope with **VanaciPrime** and the energy sector client.<br>• Outline requirements for a deployable machine learning-based forecasting solution.<br>• Identify forecasting challenges and set performance benchmarks (e.g., MAE target). |
| *Mar 2025* | 3. Data Collection and Processing | • Automate data extraction from 3 sources: OMIE, REN Datahub, and Copernicus CDS.<br>• Merge datasets and handle missing values and outliers.<br>• Perform exploratory analysis, remove irrelevant features, and assess seasonality/correlation trends. |
| *Apr 2025* | 4. Feature Engineering | • Create lag features (e.g., 1h, 24h, 168h) and rolling statistics (24h, 72h, 168h).<br>• Add calendar features (month, day, sin/cos encodings).<br>• Engineer interaction terms, weather-normalized features, and price/load differentials. |
| *May 2025* | 5. Model Development and Evaluation | • Train and compare LightGBM, XGBoost, Extra Trees, and Random Forest.<br>• Use 5-fold TimeSeriesSplit cross-validation.<br>• Optimize models using Optuna hyperparameter tuning.<br>• Evaluate MAE on train, validation, and test splits. |
| *Jun 2025* | 6. Model Selection and Pipeline Creation | • Choose the final model based on test performance and generalization.<br>• Build full ML pipeline with:<br>1. Column selector<br>2. Feature engineering block |

| | | |
|---|---|---|
| | | 3. RobustScaler for normalization.<br>• Serialize pipeline and model (.pkl). |
| | 7. Savings Calculation Integration | • Add a financial utility function to estimate user savings based on predicted prices.<br>• Calculate €/kWh and €/month savings using model outputs to improve real-world interpretability and business value. |
| *Jul-August(2025)* | 8. Deployment and Finalization | • Conduct final testing of the trained model and pipeline.<br>• Analyze feature importance to interpret model behavior.<br>• Develop backend API using FastAPI to expose model predictions.<br>• Document the whole process. |

The team leader of the development team supervised stages 3 to 6 of the project. Stages 7 and 8 of the project were performed in cooperation with another intern of the development team.

## 1.7 Structure of the report

This document was divided into seven chapters, structured to present the study and work carried out during the internship project.

Thus, Chapter 1 provides the background of electricity markets and highlights the importance of short-term electricity price forecasting. It explains the project's objectives, scope, and adopted methodology.

Chapter 2 contains a survey of the evolution of electricity price forecasting methods and the role of machine learning. It also highlights common forecasting challenges.

Chapter 3 describes the data sources (OMIE, REN, Copernicus CDS) that will form the dataset, the process of merging and cleaning them, and the performed exploratory data analysis (EDA). This includes feature distribution summaries, correlation analysis, and treatment of wind direction and missing data.

Chapter 4 details the final dataset and feature engineering strategy, presents the unified modeling setup used across all algorithms, and compares model performances using validation and test metrics. It concludes with a summary of important feature groups.

In Chapter 5, six modeling attempts are outlined, showing how model performance improved through several applied steps such as advanced lag features, momentum signals, Optuna hyperparameter tuning, and dataset splitting. The final experiment is selected based on its MAE results and generalization for improvements.

Chapter 6 presents the final LightGBM model trained on both the split and full datasets. It explains the machine learning pipeline structure, including feature engineering, scaling, and predictions. It also covers the savings estimation function, zero-value analysis of solar features, FastAPI, and endpoint readiness.

Finally, Chapter 7 draws the main conclusions. It provides a summary of the thesis contributions, including the creation of a scalable and interpretable prediction system. The limitations of the project are also discussed, and suggestions for further work are provided.

# 2. Chapter 2: Literature Review

## 2.1 Electricity Price Forecasting

The origins of EPF date back to the 1990s, when simple linear regression models were used with limited input features, such as past prices and load forecasts. Early models captured seasonality using dummy variables or sinusoidal functions. As nonlinear relationships between load and price were recognized, shallow neural networks (NNs) were introduced, paving the way for more complex machine learning approaches used in modern EPF (Jędrzejewski et al, 2022).

Beyond academic studies, industry reports provide valuable insights into evolving market dynamics. For instance, Giorgi M (2025) projects that Portugal is on track to become a net exporter of renewable energy by the mid-2030s, driven by wind, solar, hydropower, and battery storage deployment. Compared to Spain, Portugal's market conditions are more favorable for renewables, particularly wind, which achieved higher average prices of €55/MWh in 2024. These trends underscore the importance of incorporating storage behavior and renewable profitability into predictive models.

Similarly, according to the IEA (2023), in Portugal, transport and industry together account for over 60% of total final energy consumption, while electricity represents the largest share of residential demand (about 46%). These consumption patterns highlight the central role of electricity within the national energy system, underlining the importance of accurate EPF for both economic and policy decision-making.

To contribute to the ongoing research in this field, this study presents a focused review of machine learning techniques applied to EPF, with particular attention to models that are directly relevant to this project. While the broader literature includes a variety of statistical, econometric, and deep learning approaches, this review emphasizes advanced machine learning algorithms such as Gradient Boosting frameworks (LightGBM, XGBoost, CatBoost) and related ensemble methods, as these form the backbone of the forecasting models developed in this research. By examining their strengths, weaknesses, and performance across different datasets, the review highlights how such models compare to alternative approaches like support vector regression and neural networks.

The primary goal of this literature review is two-fold: firstly, to provide a structured and up-to-date overview of the state of the art in EPF, with a particular focus on tree-based ensemble learning and feature engineering strategies; and secondly, to identify the challenges that remain in improving predictive accuracy, reducing overfitting, and ensuring generalizability across time horizons and market conditions. By synthesizing and critically analyzing existing work, this study positions its own contribution—developing a robust

forecasting framework based on LightGBM, XGBoost, and CatBoost with optimized hyperparameters and engineered features—within the wider research landscape.

## 2.2 Early ML / Neural Network Approaches

Early applications of machine learning to EPF largely relied on artificial neural networks (ANNs) and feedforward neural networks (FFNNs) for short-term predictions, as discussed throughout this subsection and summarized in Table 2. These models demonstrated the ability to capture nonlinear relationships between demand and price, often outperforming classical statistical approaches such as ARIMA.

Short-term electricity price forecasting using neural networks (NNs) has been explored by several early studies. One of the earliest contributions was by Catalão et al. (2007), who applied a three-layer feedforward neural network, trained with the Levenberg-Marquardt algorithm, to forecast next-week electricity prices in the Spanish and Californian markets. The neural network outperformed traditional ARIMA models, achieving lower mean absolute percentage error (MAPE) values (3%–14% across different weeks), demonstrating the advantage of neural networks in capturing nonlinear dynamics in short-term electricity price fluctuations. Catalão et al. (2007) work highlights the early success of ANNs in short-term EPF and provides a foundation for later advanced ML and deep learning models.

For instance, Jufri et al. (2019) improved ANN performance through careful input selection and cross-validation, reaching a MAPE value of 5.36%. Similarly, Panapakidis & Moschakis (2019) demonstrated that FFNNs could effectively capture price dynamics, with MAPE values between 7.42% and 9.22%, highlighting the relevance of network design and parameter tuning.

Hybrid approaches also began to emerge. Khan et al. (2021)introduced a three-stage framework for short-term electricity price forecasting, combining ensemble empirical mode decomposition (EEMD) with an extreme learning machine (ELM). Earlier, ÖZGÜNER et al (2017) incorporated historical and forecasted load, System Marginal Price (SMP), and Net Volume Index (NVI), along with probabilistic techniques, achieving MAPE values between 9.77% and 21.81%.

In the Iberian market, Monteiro et al. (2015) analysed the relative importance of explanatory variables for day-ahead price forecasting. Using artificial neural networks, they compared a full explanatory model (EMPF), alternative reduced models (AEMPF/SEMPF), and a reference model (REMPE) that incorporates actual demand and generation values of the target day. Results showed that previous-day prices, aggregated power demand, generation types, and weather forecasts were the most influential predictors. The REMPE model achieved the lowest forecasting errors (MAPE), but it relied on information

unavailable at forecast time, serving instead as a theoretical benchmark. These findings highlighted the critical role of explanatory features in forecasting accuracy and revealed challenges in capturing price spikes, where market participant behaviour likely plays a role.

In summary, early work on electricity price forecasting applied artificial neural networks (ANNs) to deregulated day-ahead markets. Using a three-layer backpropagation network trained on historical load and price data, these studies showed that ANNs could capture the strong dependence between demand and price, achieving forecasting errors below 16% on weekdays and 20% on weekends. However, the models degraded in accuracy during periods of price spikes, highlighting the limitations of early ANN approaches and motivating the later integration of hybrid or more advanced architectures (Singhal & Swarup, 2011).

This shortcoming motivated the search for more robust approaches, including hybrid frameworks and advanced ensemble models. The progression from shallow neural networks to decomposition-enhanced and explanatory-variable-driven models set the stage for modern ensemble techniques, discussed in Section 2.3, which aim to balance predictive accuracy, generalizability, and robustness in volatile electricity markets.

*Table 2: ANN, FFNN, early ML models using historical load and price*

| ***Authors*** | ***Year*** | ***Predictor Variables*** | ***Methods*** | ***Evaluation Metric / Performance*** |
|---|---|---|---|---|
| (Catalão et al, 2007) | 2007 | Historical electricity prices (Spain & California markets) | 3-layer FFNN, LM algorithm | **MAPE: 3–14% (depending on week/market)** |
| (Singhal & Swarup, 2011) | 2011 | Historical load demand, historical electricity prices, other estimated market factors | Artificial Neural Network (3-layer backpropagation) | **Forecast error** <16% (weekday), <20% (weekend); effective for normal trends but weak during price spikes |
| (Monteiro et al, 2015) | 2015 | Previous-day prices, regional aggregated demand and generation, weather forecasts (wind, temp, irradiation), chronological variables | Artificial Neural Networks (EMPF, REMPE, AEMPF, SEMPF) | **EMPF** successfully forecast day-ahead prices; **REMPE** achieved lowest **MAPE** as a benchmark; price spikes less accurately predicted due to non-physical market factors |
| (ÖZGÜNER et al., 2017) | 2017 | Historical SMP, load, SDAP, NVI | ANN | **MAPE:** 9.77%–21.81% |
| (Jufri et al, 2019) | 2019 | Historical SMP (long & short-term) | ANN | **MAPE:** 5.36% |
| (Panapakidis & Moschakis, 2019) | 2019 | Historical SMP | FFNN | **MAPE:** 7.42%–9.22% |
| (Khan et al, 2021) | 2021 | Decomposed price series (EEMD), other historical inputs | EEMD + ELM (three-stage framework) | Reported improved accuracy vs. benchmarks (**MAPE** not explicitly given) |

## 2.3 Advanced ML and DL Approaches

Machine learning (ML) methods have gained prominence in EPF because of their ability to model nonlinear patterns. Algorithms such as XGBoost, LightGBM, and Extra Trees

often outperform traditional statistical models. They achieve this by capturing complex interactions without requiring strong assumptions about the data structure. Deep learning (DL) offers further advances, particularly in learning temporal dependencies. Popular architectures include Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. In addition, hybrid approaches that combine DL with decomposition or explainability techniques aim to improve both predictive accuracy and interpretability. Several studies highlight the effectiveness of these approaches, which are summarized in Table 3.

LSTM and Support Vector Machine (SVM) models are effective for capturing temporal dependencies in System Marginal Price (SMP) data. For instance, Bedir et al. (2021) applied time-lagged SMP inputs to LSTM models and periodic lookback to SVM models. Their results yielded MAPE values between 2.97% and 19.02%, showing that the choice of lag selection has a major impact on forecast accuracy. Similarly, Abdul Razak et al. (2023) combined the Least Squares Support Vector Machine (LSSVM) algorithm with Genetic Algorithms to optimize both model parameters and input selection, integrating historical SMP, demand forecasts, and generation mix, achieving MAPE values between 8.55% and 9.18%. These approaches highlight the benefit of advanced ML techniques for handling nonlinearity and sequential dependencies in electricity markets.

Other comparative studies reinforce these findings. Expanding on comparative analysis, Imani et al. (2020) compared multiple regression methods—SVM, Gaussian processes regression, regression trees, and multilayer perceptron (MLP)—for price forecasting. Results indicated that the SVM method provided the best performance across MAE, RMSE, R, and the total number of percentage error anomalies. Furthermore, on the implementation of deep learning (DL) techniques in electricity forecasting tasks, further work by Chatterjee (2023) explored deep learning models such as LSTMs, Convolutional Neural Networks (CNNs), CNN-LSTM hybrids, and Deep Neural Networks (DNNs) for EPF. Model performance was benchmarked using MAE and MAPE, highlighting the effectiveness of hybrid deep learning architectures.

On a market-specific level, Kapoor and Wichitaksorn (2023) analyzed the New Zealand electricity market by comparing statistical models (including GARCH and stochastic volatility with t-distribution variants) against advanced ML methods such as LSTM, GRU, XGBoost, and deep neural networks. Their findings showed that when combined with feature selection methods (LASSO, mutual information, and recursive feature elimination), the statistical models often outperformed ML alternatives, achieving lower forecast errors than both neural networks and ensemble methods. This highlights the importance of feature selection in EPF, sometimes even outweighing the choice of complex learning models.

In the Portuguese MIBEL market, recent research conducted by Pereira (2023) compared XGBoost and LSTM models using standard predictors (generation mix, grid usage, weather) and found that LSTM performed best when trained on more recent historical data. The study by César D (2023) further extended forecasting by incorporating sentiment analysis of financial news, where TF-IDF with Logistic Regression produced the most effective sentiment signals. While the enhanced model only outperformed the traditional one on certain days, it highlighted the role of sentiment features during periods of price volatility and political or economic change.

Overall, advanced ML and DL approaches provide substantial improvements in EPF, particularly in capturing nonlinear and temporal dependencies. However, evidence also suggests that model choice alone is insufficient—careful feature selection and hybridization with domain-specific information remain critical. This motivates the focus of the present research on combining strong learning algorithms with systematic feature engineering.

*Table 3: Sequence-based models for short-term forecasting.*

| ***Authors*** | ***Year*** | ***Predictor Variables*** | ***Methods*** | ***Evaluation Metric / Performance*** |
|---|---|---|---|---|
| (Imani et al., 2020) | 2020 | --- | SVM, Gaussian processes regression, regression trees, and multilayer perceptron | Best: **SVM (MAE, RMSE, error anomaly)** |
| (Bedir et al., 2021) | 2021 | Historical SMP (with time lags) | LSTM, SVM | **MAPE:** 2.97%–19.02% |
| (Abdul Razak et al., 2023) | 2023 | Historical SMP, demand & generation forecasts | LSSVM + GA | **MAPE:** 8.55%–9.18% |
| (Chatterjee, 2023) | 2023 | | LSTMs, CNNs, CNN-LSTM hybrids, and DNNs | Reported performance was benchmarks using **MAE and MAPE** |
| (Kapoor & Wichitaksorn, 2023) | 2023 | Demand, generation fuel, forward prices, reserve prices, HVDC transfer rate, weather variables | Statistical: GARCH, SV, GARCH-t, SV-t; ML: LSTM, GRU, XGBoost, LEAR, DNN; Feature selection: LASSO, MI, RFE | Best: GARCH-t and SV-t (with feature selection), improved MAPE & MASE by 2–3% over LEAR; ~40% gain vs. models without feature selection; Outperformed LSTM, GRU, XGBoost, DNN |
| (Pereira, 2023) | 2023 | Generation type, grid usage, weather (historical features from 2018/2021); sentiment features from *Negócios* | XGBoost, LSTM; sentiment analysis via Logistic Regression | LSTM (with 2021+ history) outperformed XGBoost; enhanced model with sentiment showed superior |

| *Authors* | *Year* | *Predictor Variables* | *Methods* | *Evaluation Metric / Performance* |
|---|---|---|---|---|
| | | newspaper (TF-IDF text representation) | | performance on certain days, especially during volatile or socio-economic events, though overall performance was similar to the traditional model |

## 2.4 Modern Ensemble Tree-Based Models

Ensemble learning has become one of the most effective strategies for improving accuracy and robustness in EPF. By combining multiple weak or strong learners, ensemble methods reduce variance, enhance stability, and often outperform standalone models. In particular, tree-based boosting and bagging methods (e.g: XGBoost, LightGBM, CatBoost, and Random Forests (RF) have been widely adopted for EPF due to their efficiency, scalability, and ability to capture nonlinear feature interactions.

Table 4 shows that modern ensemble methods have improved EPF accuracy and robustness. Likewise, Pan (2017) combined AdaBoost with SVM and BPNN, integrating weather data to improve forecast performance (MAPE values of 3.84%–8.43%). Moreover, Shim et al. (2023) used a two-step approach, first forecasting load via XGBoost, then predicting SMP using RF, LightGBM, and Linear Regression. Careful feature selection and hyperparameter tuning allowed ensemble models to achieve MAPE values as low as 2.70%, demonstrating the effectiveness of tree-based ML approaches for short-term price forecasting in complex electricity markets.

More recent work has explored integrating ensembles with deep learning and attention mechanisms. Using advanced deep learning and ensemble models, Laitsos et al. (2024) investigated short-term EPF in the Germany–Luxembourg market. Their study introduced several architectures, including a Convolutional Neural Network (CNN) with attention, a CNN–GRU hybrid with attention, an optimized Multi-Head Attention transformer, and two ensemble approaches (stacking and soft voting). Results showed strong predictive accuracy, with the soft voting ensemble (MAPE 6.125%) and CNN–GRU with attention (MAPE 6.333%) outperforming other models. Interestingly, the Multi-Head Attention model, despite mixed results in prior literature, also achieved competitive performance (MAPE 6.889%). These findings underscore the potential of attention-based architectures and ensemble strategies in improving EPF, which aligns closely with this study's focus on evaluating ensemble learning and boosting-based models.

In the New England electricity market, another study by Agrawal et al. (2019) proposed a novel ensemble model combining Relevance Vector Machines (RVMs) and Extreme

Gradient Boosting (XGBoost) for next-hour electricity price forecasting. Two RVM models, based on Gaussian radial basis and polynomial kernels, were first trained to capture nonlinear patterns. Their outputs, along with an XGBoost model, were stacked using Elastic Net regression, and final forecasts were generated via bagging. Compared with baseline models—including multilayer perceptron, random forest, support vector machine, recurrent neural network, and LASSO regression—the proposed ensemble achieved the highest predictive accuracy with a Mean Absolute Error (MAE) of 2.6 and remained computationally efficient (training time of 88 seconds). The study emphasized the practical benefits of highly accurate next-hour forecasts in enabling price-directed energy consumption strategies and noted the growing importance of incorporating renewable-related variables for future model development.

In the Spanish electricity market, Díaz et al. (2019) applied Gradient Boosted Regression Trees (GBRT) within a quantile regression framework to forecast day-ahead prices. Their model outperformed multiple linear regression, achieving RMSE of 2.78 €/MWh and MAE of 1.94 €/MWh, while also providing reliable prediction intervals with 90% of errors below 6.8 €/MWh. To enhance interpretability, partial dependence analysis was employed, revealing non-linear and asymmetric effects of system variables on price formation.

Collectively, these studies confirm that ensemble tree-based models deliver state-of-the-art performance in EPF while also offering flexibility for integration with hybrid designs, interpretability methods, and domain-specific features. This review, therefore, positions gradient boosting methods—LightGBM, XGBoost, and CatBoost—as the methodological core of the present study, where they are optimized and benchmarked to achieve robust, generalizable forecasts.

*Table 4: XGBoost, Random Forest, LightGBM; hyperparameter optimization and feature selection*

| ***Authors*** | ***Year*** | ***Predictor Variables*** | ***Methods*** | ***Evaluation Metric / Performance*** |
|---|---|---|---|---|
| (Pan, 2017) | 2017 | Historical SMP, weather | AdaBoost + SVM + BPNN | **MAPE:** 3.84%–8.43% |
| (Agrawal et al., 2019) | 2019 | Historical electricity prices (hourly data, New England market, 2012–2013); implied market volatility factors | Ensemble of Relevance Vector Machines (RBF & polynomial kernels), Extreme Gradient Boosting, Elastic Net stacking, bagging | **MAE =** 2.6 on **test set; training time** = 88s; outperformed ANN, SVM, RNN, Random Forest, and LASSO |
| (Díaz et al., 2019) | 2019 | Publicly available real-time TSO data (system state variables) | Gradient Boosted Regression Trees (Quantile Regression) + Partial Dependence Analysis | RMSE = 2.78 €/MWh, MAE = 1.94 €/MWh, MAPE = 0.059; 90% of errors < 6.8 €/MWh |

| *Authors* | *Year* | *Predictor Variables* | *Methods* | *Evaluation Metric / Performance* |
|---|---|---|---|---|
| (Shim et al., 2023) | 2023 | Forecasted load, date, fuel price/capacity, Historical SMP | XGBoost, RF, LightGBM, LR | **MAPE:** 2.70%–6.93% |
| (Laitsos et al., 2024) | 2024 | Historical price; load; generation by fuel (brown coal/lignite, hard coal, gas, nuclear, hydro—run-of-river, reservoir, pumped storage—solar, wind onshore); weather (temperature, humidity); calendar/cyclical features (hour, day-of-week, month, quarter, weekend; sin/cos encodings) | CNN with attention; Hybrid CNN–GRU with attention; Multi-Head Attention (Transformer); Soft-voting ensemble; Stacking ensemble; Optuna hyperparameter tuning | **MAPE: 6.125%**–10.699% (best = 6.125) |

## 2.5 Summary and Challenges in Electricity Price Forecasting

The literature on EPF has evolved through several methodological phases. Early statistical models, such as autoregressive and GARCH-type approaches, offered interpretable structures but were limited in capturing the complex nonlinear dynamics of electricity markets. The subsequent wave of artificial neural networks introduced greater flexibility and nonlinearity handling, though at the cost of interpretability and with heightened risk of overfitting. More recently, hybrid methods combining signal decomposition, optimization techniques, or domain-specific variables have shown improved predictive stability but often at the expense of increased complexity and computational demands.

The state of the art is now dominated by modern ensemble tree-based algorithms—particularly gradient boosting models such as LightGBM, XGBoost, and CatBoost—and, in parallel, by hybrid deep learning architectures. These approaches consistently outperform earlier methods by capturing nonlinear interactions, integrating diverse explanatory features, and leveraging sophisticated training strategies.

However, several challenges remain. First, many models are prone to overfitting, particularly when feature sets are large or insufficiently regularized. Second, interpretability continues to be a concern, with advanced models often functioning as "black boxes" that provide limited transparency for market operators and policymakers. Third, the richness of features employed in prior work varies considerably, and relatively few studies systematically integrate weather, load, and renewable generation variables in a unified framework. Finally, while substantial research has been conducted for markets such as Spain or broader European regions, focused studies on the Portuguese electricity market remain scarce.

This work addresses these gaps by developing a forecasting framework based on ensemble boosting algorithms (LightGBM, XGBoost, Extra Trees, and CatBoost), enhanced

through hyperparameter optimization and feature engineering. By tailoring explanatory features to the Portuguese context and systematically evaluating model performance, the research contributes both methodologically and contextually, providing a more accurate and interpretable approach to short-term electricity price forecasting in Portugal.

Building on these insights, the next chapter outlines the methodological framework of this research. It details the dataset construction and exploratory data analysis.

# 3. Chapter 3: Dataset and Exploratory Data Analysis

This chapter outlines the data sources, preprocessing steps, and exploratory analysis conducted to prepare the dataset for predictive modeling. The methodology begins with a detailed examination of the original dataset, which combines electricity market prices, load forecasts, renewable generation data, and meteorological information. These initial steps are essential to understand data quality, feature distributions, relationships, and patterns, which form the foundation for model development in subsequent chapters.

## 3.1 Data Sources and Description

The forecasting model developed in this project relies on the following three key data sources:

- The Iberian Electricity Market Operator **(OMIE- Operador del Mercado Ibérico de Energia)** for marginal electricity prices and traded energy.
- **REN Datahub (Rede Eléctrica Nacional)** for national electricity load and renewable energy indicators.
- **Copernicus Climate Data Store** for meteorological variables.

Table 5 shows an overview of these data sources, along with a summary description of associated variables, data collection frequency, time periods, and roles in model building.

Each source is then described in more detail in the following subsections, covering its background, the data retrieved, and how it supports the research objectives.

*Table 5: Overview of Data Sources*

| *Source* | *Variables* | *Frequency* | *Period* | *Role in Model* |
|---|---|---|---|---|
| *OMIE* | Hourly day-ahead electricity prices and traded energy | Hourly | 2024-01-01 to 2025-05-31 | Target variable and market activity |
| *REN Data Hub* | Load forecasts, generation types, import/export flows | Hourly | Same period | Supply/demand explanatory variables |
| *Copernicus CDS (ERA5)* | Weather variables: temperature, solar radiation, wind, precipitation | Hourly | Same period | Weather-related explanatory variables |

### 3.1.1 OMIE – Iberian Electricity Market Prices

#### Background

OMIE is the designated NEMO (Nominated Electricity Market Operator) for the Iberian Peninsula OMIE (2025), managing the day-ahead and intraday electricity markets for Spain and Portugal. It has operated under the Single Day-Ahead Coupling (SDAC) since 2014 and adheres to European market algorithms. OMIE coordinates energy bids across Europe.

### OMIE Data Rationale & Variables considered

As the electricity price is the target variable to predict (primary target variable for forecasting), the "`Preços_marginais_sistema_portugues`" column is used, which shows how much electricity costs every hour in Portugal.

The variables presented in Table 6 represent market electricity prices and traded energy volumes across Portugal and Spain and were, therefore, considered in the study.

*Table 6: OMIE – Iberian Market Variables*

| ***Feature Name*** | ***Description*** |
|---|---|
| *Preços marginais sistema portugues* | Electricity price in Portuguese market (€/MWh) → **Target variable** |
| *Preços marginais sistema espanhol* | Electricity price in Spanish market (€/MWh) |
| *Energia negociada Mercado Diário* | Energy traded in the daily market (MWh) |
| *Energia total de aquisição casada sistema espanhol* | Energy bought in Spain (MWh) |
| *Energia total de venda casada sistema espanhol* | Energy sold in Spain (MWh) |
| *Energia total de aquisição casada sistema portugues* | Energy bought in Portugal (MWh) |
| *Energia total de venda casada sistema portugues* | Energy sold in Portugal (MWh) |
| *Importação a partir de Portugal para Espanha* | Energy imported from Portugal to Spain (MWh) |
| *Exportação de Espanha para Portugal* | Energy exported from Spain to Portugal (MWh) |
| *Energia Mercado Ibérico incluindo bilaterais* | Total Iberian Market energy including bilateral deals (MWh) |

This project focuses on forecasting the day-ahead market prices (`"Preços marginais sistema portugues"`), which represent the official hourly prices published by OMIE in €/MWh. These prices are determined daily at 12:00 CET based on bids submitted by electricity producers and consumers and reflect the equilibrium between supply and demand for the following 24 hours. In this study, the day-ahead price (`"Preços marginais sistema portugues"`) serves as the primary target variable in the forecasting model. However, the proposed model extends the forecast horizon to seven days ahead (`168 hours`), predicting future hourly prices beyond the standard day-ahead period by leveraging historical, calendar, and weather-related features.


*Figure 1: Evolution of day-ahead minimum, average and maximum price of Portugal. Source OMIE*

Figure 1 tracks the **day-ahead electricity prices** in Portugal from 2007 to 2024, showing the minimum, average, and maximum prices per year. The yellow line shows the maximum price, the light-yellow line shows the minimum price, and the green line shows the arithmetic average price. The graph shows that electricity prices in Portugal were stable for many years but experienced an extreme spike in 2022. After this peak, prices have since started to fall but remain higher than they were before 2021.

### 3.1.2 REN Data Hub – Load and Generation Data

**Background**

**REN (Redes Energéticas Nacionais)** is responsible for managing Portugal's electricity transmission infrastructure and energy system; in simpler words, REN manages the electricity and natural gas networks in Portugal. In June 2021, REN launched the Data Hub—an open data platform offering hourly updates and historical trends of national electricity and natural gas systems (REN – Redes Energéticas Nacionais, 2025). The **REN Data Hub** provides operational data for Portugal's electricity system.

**REN Data Rationale & Variables considered**

The electricity price depends a lot on how much energy is being produced or used. Also, renewable sources (like solar or wind) are cheaper and affect the price. Hence, the following issues are relevant:

- **Hourly total electricity consumption** (GWh and peak MW), including hourly energy demand.
- **Hourly renewable generation**, broken down by technology (such as wind, solar, and hydro power production).
- **System import/export balances:** How much electricity is imported/exported.

Following the above line of reasoning, the following variables from the REN Data Hub were used, as depicted in Table 7:

*Table 7: REN Data Hub Variables*

| ***Feature Name*** | ***Description*** |
|---|---|
| *Generation - Solar [MW] Day Ahead* | Predicted solar energy generation for tomorrow (MW) |
| *Generation - Wind Onshore [MW] Day Ahead* | Predicted wind energy generation for tomorrow (MW) |
| *Biomass - Actual Aggregated [MW]* | Real-time biomass energy generation (MW) |
| *Fossil Gas - Actual Aggregated [MW]* | Fossil gas electricity generation (MW) |
| *Fossil Hard coal - Actual Aggregated [MW]* | Electricity from hard coal (MW) |
| *Hydro Pumped Storage - Actual Aggregated [MW]* | Generation from pumped hydro (MW) |
| *Hydro Pumped Storage - Actual Consumption [MW]* | Consumption for pumping water (MW) |
| *Hydro Run-of-river and poundage - Actual Aggregated [MW]* | Run-of-river hydro generation (MW) |
| *Hydro Water Reservoir - Actual Aggregated [MW]* | Reservoir-stored hydro power (MW) |
| *Other - Actual Aggregated [MW]* | Other non-specified sources (MW) |
| *Solar - Actual Aggregated [MW]* | Real-time solar generation (MW) |
| *Wind Onshore - Actual Aggregated [MW]* | Real-time onshore wind generation (MW) |
| *Day-ahead Total Load Forecast [MW]* | Forecasted load (electricity demand) (MW) |
| *Actual Total Load [MW]* | Real electricity demand (MW) |

This data is crucial or capturing supply-demand dynamics and assessing how renewables influence electricity prices. The data was collected in hourly resolution and aligned with the price series. These variables offer crucial explanatory power by linking price movements to underlying demand and supply (renewables) dynamics.

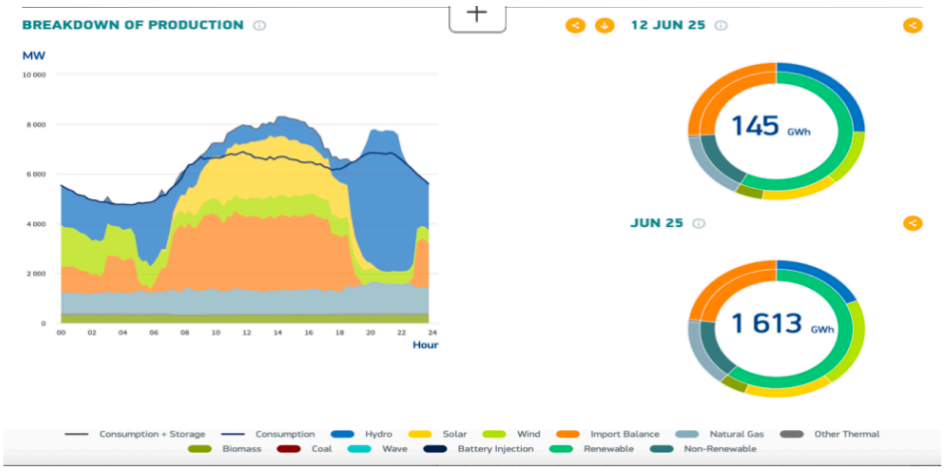



*Figure 2: Daily electricity production breakdown – June 12, 2025. Source: REN Data Hub, Daily Balance Dashboard.*

Figure 2 shows the daily electricity balance, regarding the total electricity consumption, peak load, percentage of use of renewable sources. On June 12, 2025, Portugal's total electricity consumption reached **145 GWh**. The load curve in Figure 2 shows strong solar

generation between 08:00 and 18:00 (yellow area), significant hydroelectric output throughout the day (blue), and moderate wind generation (light green). Natural gas (grey) and imports (orange) covered evening peaks when solar dropped. This distribution highlights the intraday balancing between renewables and dispatchable sources, crucial for understanding price volatility.

### 3.1.3 Copernicus Climate Data Store – Weather Variables

**Background**

ERA5 is the latest generation global climate reanalysis by ECMWF/Copernicus, providing hourly, gridded data since 1940 at 0.25°×0.25° resolution. It combines observations and physics-based reconstructions into a consistent dataset. It's a European climate data platform that provides free weather and climate information (Copernicus Climate Change Service, 2025).

**Copernicus Data Rationale & Variables considered**

This data source was used to download weather conditions for Portugal, as weather affects both energy production and consumption. Weather impacts renewable energy production (e.g., more sun = more solar energy) and electricity demand (e.g., hotter days = more air conditioning = more demand = higher prices).

Environmental data was, thus, obtained from the Copernicus Climate Data Store, which provides high-resolution, hourly weather variables for Europe.

The following variables were used as shown in Table 8:

*Table 8: Copernicus Climate Data Store (ERA5) Variables*

| ***Feature Name*** | ***Description*** |
|---|---|
| *Speed_km_h* | Wind speed in km/h |
| *Angle* | Wind direction angle (degrees) |
| *Direction* | Wind direction as cardinal points (N, NE, etc.) |
| *Radiação Solar (W/m²)* | Solar radiation intensity (W/m²) |
| *Precipitação (mm) - Peso da Régua* | Rainfall amount in Peso da Régua (mm) |
| *Lisboa temperature °C* | Air temperature in Lisbon °C |
| *Porto temperature °C* | Air temperature in Porto °C |
| *Évora temperature °C* | Air temperature in Évora °C |
| *Media Temperatura °C* | Mean temperature of Lisbon, Porto, and Évora °C |

These weather variables help to explain demand patterns (e.g., heating and cooling), feed-in from solar generation, and extreme events potentially affecting supply/demand balance.

### 3.1.4 Additional Derived Features (Engineered from Date & Time)

The features shown in Table 9 were not taken directly from the data sources but were generated from timestamps during preprocessing:

*Table 9: Additional features in the Dataset*

| ***Feature Name*** | ***Description*** |
|---|---|
| *Data e Dia da Semana* | The date and name of the weekday |
| *Hora* | Hour of the day |
| *Day of the week* | 0 to 6 → Monday to Sunday |
| *week or weekend* | Weekday/weekend classification |
| *Holiday* | Flag for Portuguese national holidays |

### 3.1.5 Dataset Overview

The initial dataset (.xlsx format) contains 38 features and was compiled by merging data from the three data sources previously described and presented in Table 5. The dataset comprises high-frequency hourly data from January 1, 2024, to May 31, 2025, forming the foundation for feature engineering and model training. It contains **87674 records** and **38 attributes**.

As the dataset was compiled from three different sources, some variables originating from **OMIE** and **Copernicus** are named in **Portuguese.** Throughout this chapter, these variables were used in their original form (as originally named) for Exploratory Data Analysis (EDA). However, the names were later translated during the model-building stage.

Table 10 provides a brief description of each of the 38 features considered in the initial dataset:

*Table 10: Dataset (Used for First EDA)*

| ***Feature Name*** | ***Description*** |
|---|---|
| *Data e Dia da Semana* | The calendar date and day name (e.g., Monday) |
| *Hora* | The time of day in hours (e.g., 13:00) |
| *Day of the week* | Number from 0 to 6 representing the weekday (0 = Monday) |
| *week or weekend* | Whether the day is a weekday or weekend |
| *Holiday* | Whether the day is a public holiday in Portugal |
| *Generation - Solar [MW] Day Ahead* | Predicted solar energy generation for tomorrow (MW) |
| *Generation - Wind Onshore [MW] Day Ahead* | Predicted onshore wind energy for tomorrow (MW) |
| *Biomass - Actual Aggregated [MW]* | Real biomass electricity generation (MW) |
| *Fossil Gas - Actual Aggregated [MW]* | Real electricity from fossil gas (MW) |
| *Fossil Hard coal - Actual Aggregated [MW]* | Real electricity from coal (MW) |
| *Hydro Pumped Storage - Actual Aggregated [MW]* | Electricity generated by hydro storage (MW) |
| *Hydro Pumped Storage - Actual Consumption [MW]* | Electricity used to pump water for hydro storage (MW) |

| *Feature Name* | *Description* |
|---|---|
| *Hydro Run-of-river and poundage - Actual Aggregated [MW]* | Natural flowing hydroelectric generation (MW) |
| *Hydro Water Reservoir - Actual Aggregated [MW]* | Water stored in reservoirs for hydro power (MW) |
| *Other - Actual Aggregated [MW]* | Electricity from other sources not listed (MW) |
| *Solar - Actual Aggregated [MW]* | Real solar energy generation (MW) |
| *Wind Onshore - Actual Aggregated [MW]* | Real onshore wind generation (MW) |
| *Day-ahead Total Load Forecast [MW]* | Predicted electricity demand (MW) for the next day |
| *Actual Total Load [MW]* | Real electricity demand (MW) |
| *Speed km h* | Wind speed (in kilometers per hour) |
| *Angle* | Wind direction angle (degrees) |
| *Direction* | Wind direction (e.g., N, NE, SW) |
| *Radiação Solar (W/m²)* | Solar radiation strength (watts per square meter) |
| *Precipitação (mm) - Peso da Régua* | Rainfall in the region of Peso da Régua (mm) |
| *Lisboa temperature °C* | Temperature in Lisbon °C |
| *Porto temperature °C* | Temperature in Porto °C |
| *Évora temperature °C* | Temperature in Évora °C |
| *Media Temperatura °C* | Average of Lisbon, Porto, Évora temperatures |
| *Preços marginais sistema espanhol* | Electricity price in Spanish market (€/MWh) |
| *Preços marginais sistema portugues* | **Electricity price in Portuguese market** (€/MWh) → **Target** |
| *Energia negociada Mercado Diário* | Energy traded in the daily market (MWh) |
| *Energia total de aquisição casada sistema espanhol* | Energy bought in Spain (MWh) |
| *Energia total de venda casada sistema espanhol* | Energy sold in Spain (MWh) |
| *Energia total de aquisição casada sistema portuguas* | Energy bought in Portugal (MWh) |
| *Energia total de venda casada sistema portugues* | Energy sold in Portugal (MWh) |
| *Importação a partir de Portugal para Espanha* | Energy imported from Portugal to Spain (MWh) |
| *Exportação de Espanha para Portugal* | Energy exported from Spain to Portugal (MWh) |
| *Energia Mercado Ibérico incluindo bilaterais* | Total Iberian Market energy including side deals (MWh) |

## 3.2 Data Cleaning

Initial cleaning steps were applied to ensure data quality and integrity before analysis and modeling. These included:

- **Missing value analysis**: Null values were counted for each numerical column.
- **Duplicate detection**: Duplicate records were identified and counted.
- **Data type conversions**: Date and time fields were converted to appropriate datetime formats to allow for feature extraction.

Categorical variables (such as wind direction) were prepared for encoding, and temporal variables (e.g., hour, day, month) were extracted to assist in capturing periodic effects in later modeling.

## 3.3 Exploratory Data Analysis (EDA)

To better understand the structure and quality of the dataset before modeling, an extensive EDA was conducted (as shown in Appendix A) using Python libraries such as Pandas, NumPy, Seaborn, Matplotlib, and SciPy (Matplotlib (2025); NumPy (2025); SciPy (2025); Pandas (2025); Seaborn (2025)). The EDA process undertaken is overall described below and further detailed in the following subsections.

The analysis began by loading the dataset described in Section 3.1.5 and depicted in Table 10, a filtered dataset containing 38 features, which combined electricity market data, meteorological variables, renewable energy generation, and calendar-based attributes.

The features were grouped into logical categories: time-based variables (such as hour, date, day of the week, and holiday indicators), categorical variables (such as wind direction), and numerical variables (such as energy generation and weather measurements). The EDA focused primarily on analyzing the numerical features, which excluded redundant or collinear variables as well as categorical and time-related columns. Data quality checks revealed the presence of some missing values and a small number of duplicate rows, which were considered during preprocessing.

The distributions of each numerical variable were visualized using histograms and Quantile-Quantile (Q-Q) plots to assess normality. Additionally, boxplots were used to detect potential outliers in each feature. These visual inspections provided insights into skewed distributions and variable ranges, which are crucial for modeling decisions such as normalization or transformation.

A correlation heatmap was constructed to identify strong linear relationships between features. In particular, the analysis focused on identifying variables with a Pearson correlation coefficient greater than ±0.4 with respect to the target variable — the "*Preços marginais sistema portugues*". Several features, such as actual load, solar radiation, and market energy traded, showed significant correlations with the target and were therefore flagged as potentially important for predictive modeling.

Temporal features such as hour, day, and month were extracted from the original date field to facilitate temporal pattern analysis. These were also correlated with the electricity price to detect any seasonality or cyclical effects. Furthermore, categorical wind direction data were one-hot encoded to allow for quantitative analysis, and their correlation with the price was also explored.

Through this EDA process, key variables were identified for feature engineering, and several patterns and trends were uncovered that guided later stages of model development.

This initial analysis played a critical role in selecting relevant features, removing redundant or weak predictors, and shaping the data pipeline.

### 3.3.1 Null Values

A null value analysis was conducted to assess data completeness. Most features had no missing values, ensuring modeling reliability. Only two features had missing data:

- `Generation_Solar[MW]_Day_Ahead/BZN|PT`: 3457 missing values
- `Generation_Wind_Onshore[MW]_Day Ahead/BZN|PT`: 3457 missing values

These two columns had a significant amount of missing values, which are likely due to measurement or collection gaps. Since these values are not concentrated in any particular period, we handled them using forward fill imputation to preserve temporal structure.

### 3.3.2 Summary of Feature Distributions (Histograms and QQ Plots)

To better understand the behavior of each variable in the dataset, histograms and QQ plots were analyzed. Histograms show how often different values occur, while QQ plots compare the variable's distribution to a normal distribution. These visualizations, presented in Figure 3, revealed several important patterns.

**Skewed and Non-Normal Features (Most Features)**

Many features in the dataset, especially those related to renewable energy (e.g., solar, wind, hydro), weather conditions (e.g., solar radiation, precipitation), and cross-border trade (imports/exports), are **strongly right-skewed**.

- These variables often have many zero or low values (e.g., no sun at night, no rainfall), followed by occasional high peaks.
- Their QQ plots deviate significantly from the diagonal line, confirming **non-normality** and **heavy tails**.
- The **target variable**, `"Preços_marginais_sistema_portugues"`, follows a similar pattern—mostly low prices with occasional **extreme spikes**.

This behavior is common in electricity markets and indicates that traditional linear models may not perform well. Instead, **tree-based models like LightGBM** are more suitable as they do not assume a normal distribution and can naturally handle skewed, non-linear relationships.

**Closer to Normal Features**

Some variables showed more stable and symmetric distributions:

- `"Actual_Total_Load [MW]"`, `"Media_Temperatura °C"`, and `"Energia_Mercado_Ibérico_incluindo_bilaterais"` are **roughly bell-shaped**, with only slight skewness.

- Their QQ plots follow the normal line more closely, suggesting these features are more predictable and **strong candidates** for modeling.

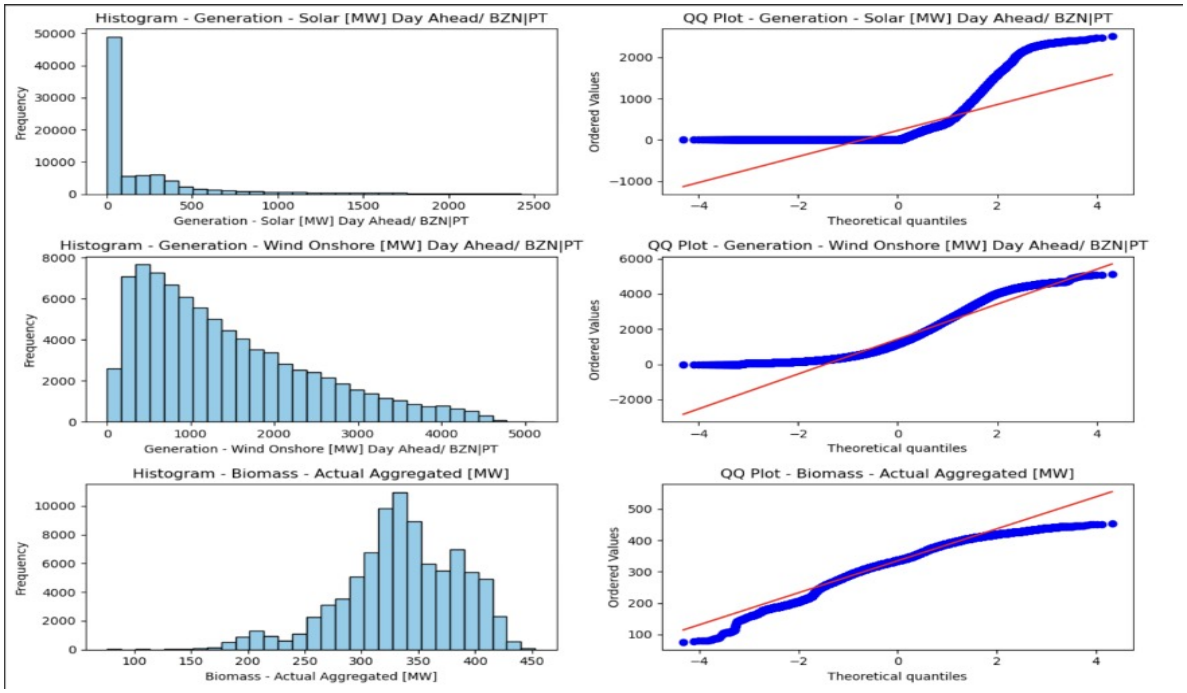
**Special Cases**

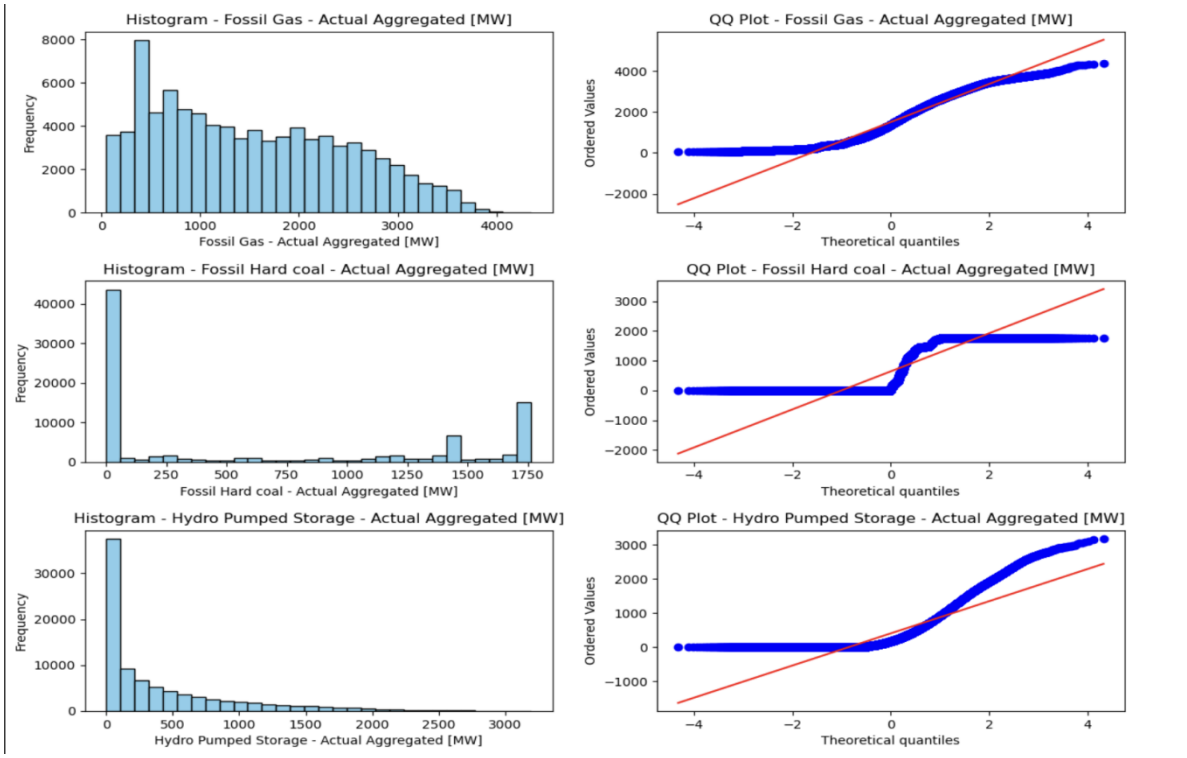
- "Fossil_Hard_coal_Actual_Aggregated[MW]", "Hydro_Pumped _Storage_Actual_Aggregated[MW]" and "Hydro_Pumped_ Storage_Actual_Consumption[MW]" often stay at zero, indicating **intermittent or reserve usage**.
- "Biomass_Actual_Aggregated[MW]" tends to operate within a narrow, consistent range, suggesting **base-load behavior**.
- "Wind_Onshore_Actual_Aggregated [MW] displays multiple peaks (multi-modal), hinting at **prevailing wind directions**, which could be explored further as categorical patterns.

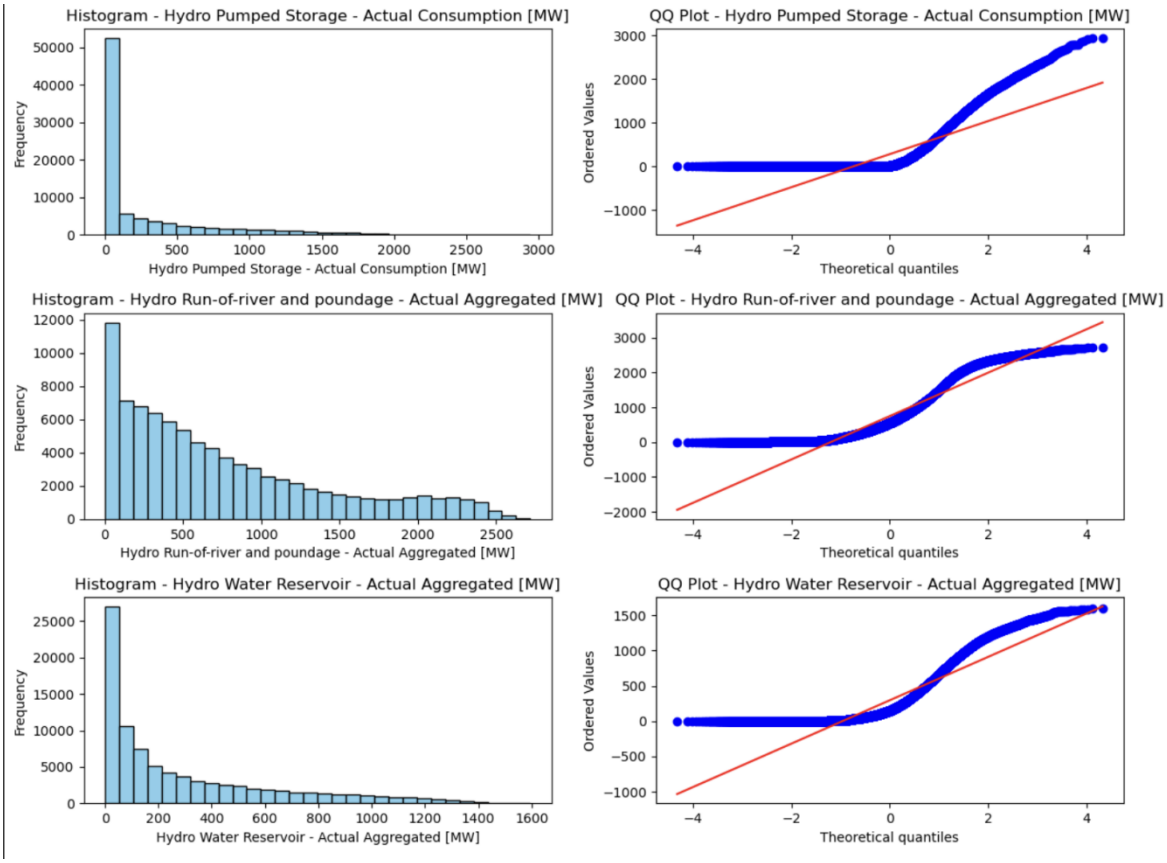
**Overall Implications for Modeling:**

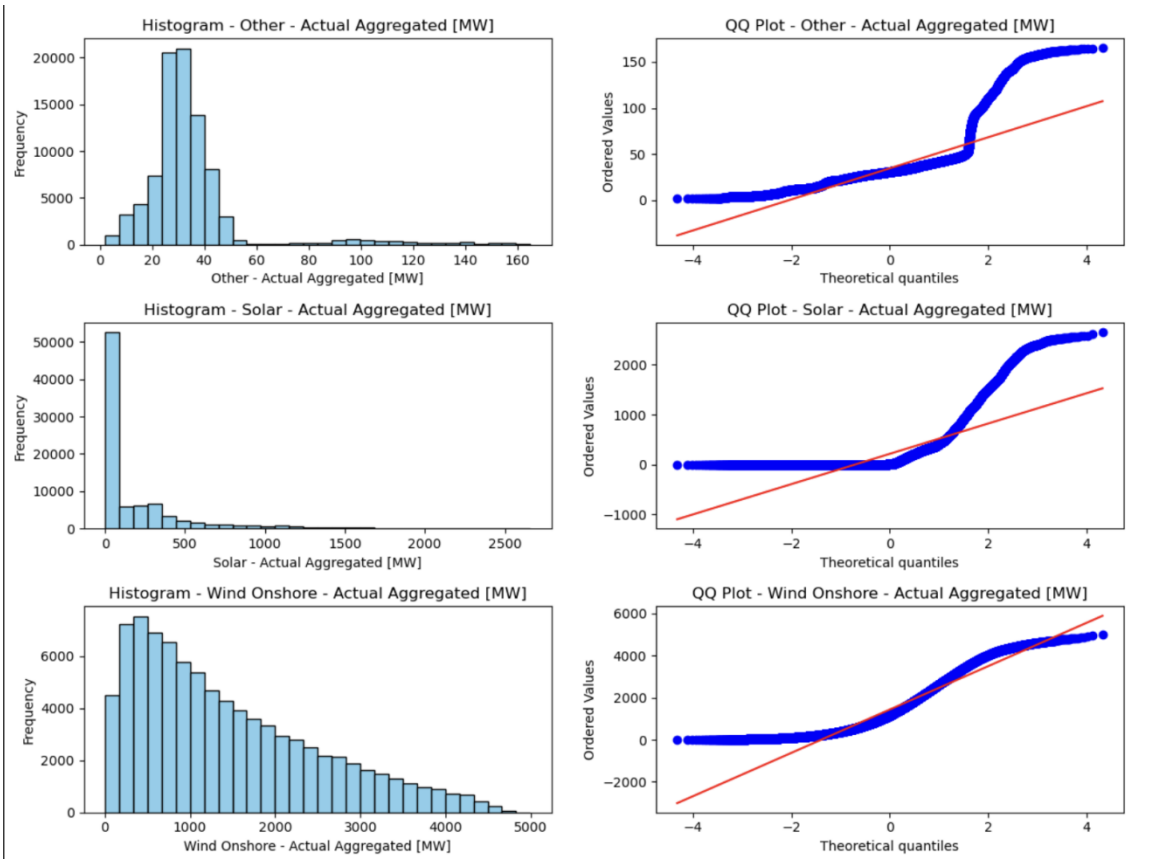
Many features, especially those related to renewable energy generation (solar, wind, hydro pumped storage), precipitation, and solar radiation, are highly **right-skewed and non-normal**, with many zero values. The target variable ("Preços marginais sistema portugues") is also **right-skewed with price spikes**. This non-normal nature suggests that models robust to such distributions (like tree-based models such as LightGBM) are suitable. Scaling these features is also crucial for some algorithms, and advanced transformations (e.g., log, Box-Cox) might be considered for highly skewed features if using models sensitive to normality, though tree-based models are less impacted. The near-normal distribution of total load, temperature, and some market energies is beneficial.

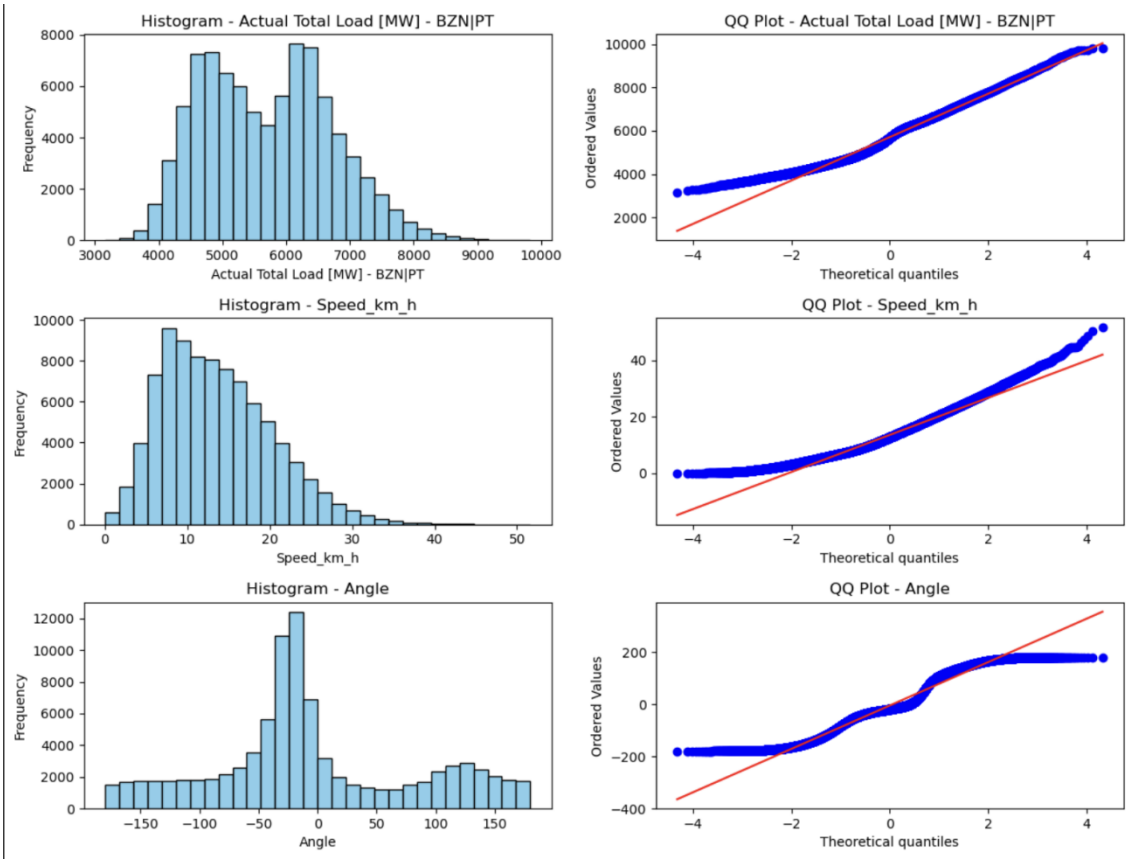
*Figure 3: Histograms and QQ plots*

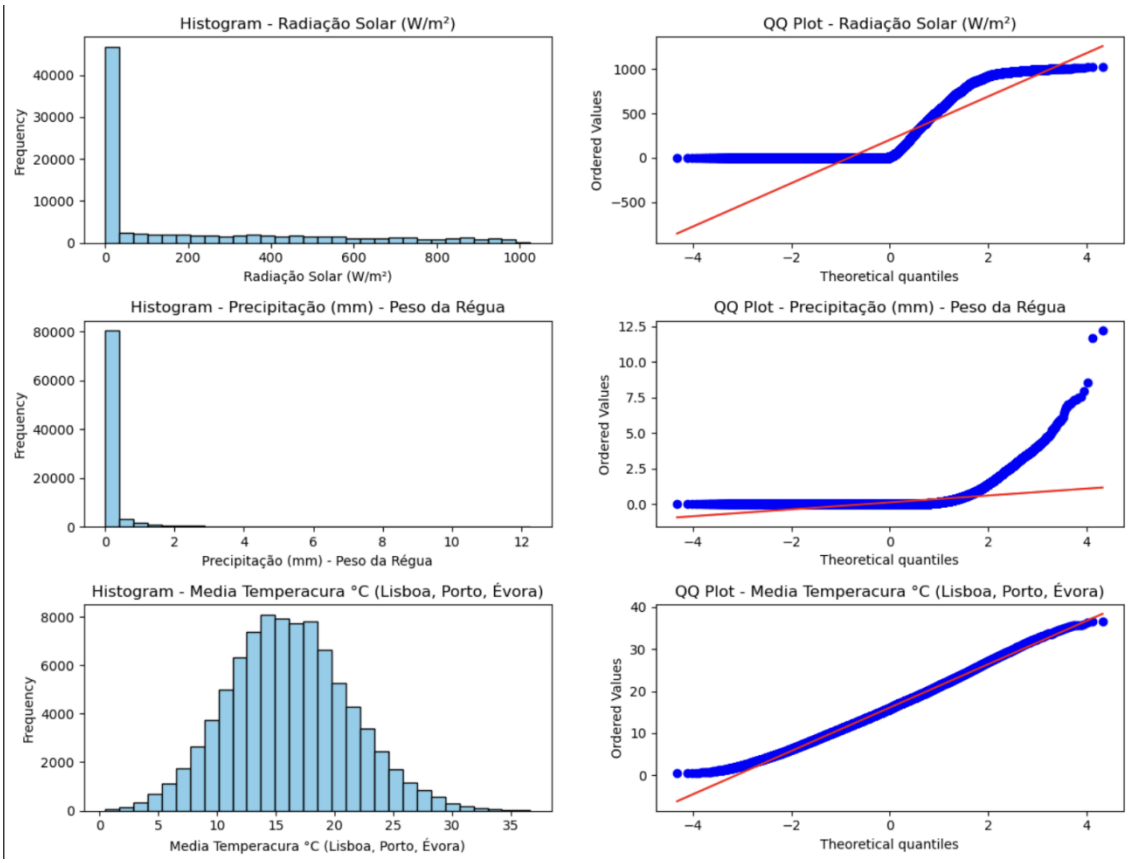


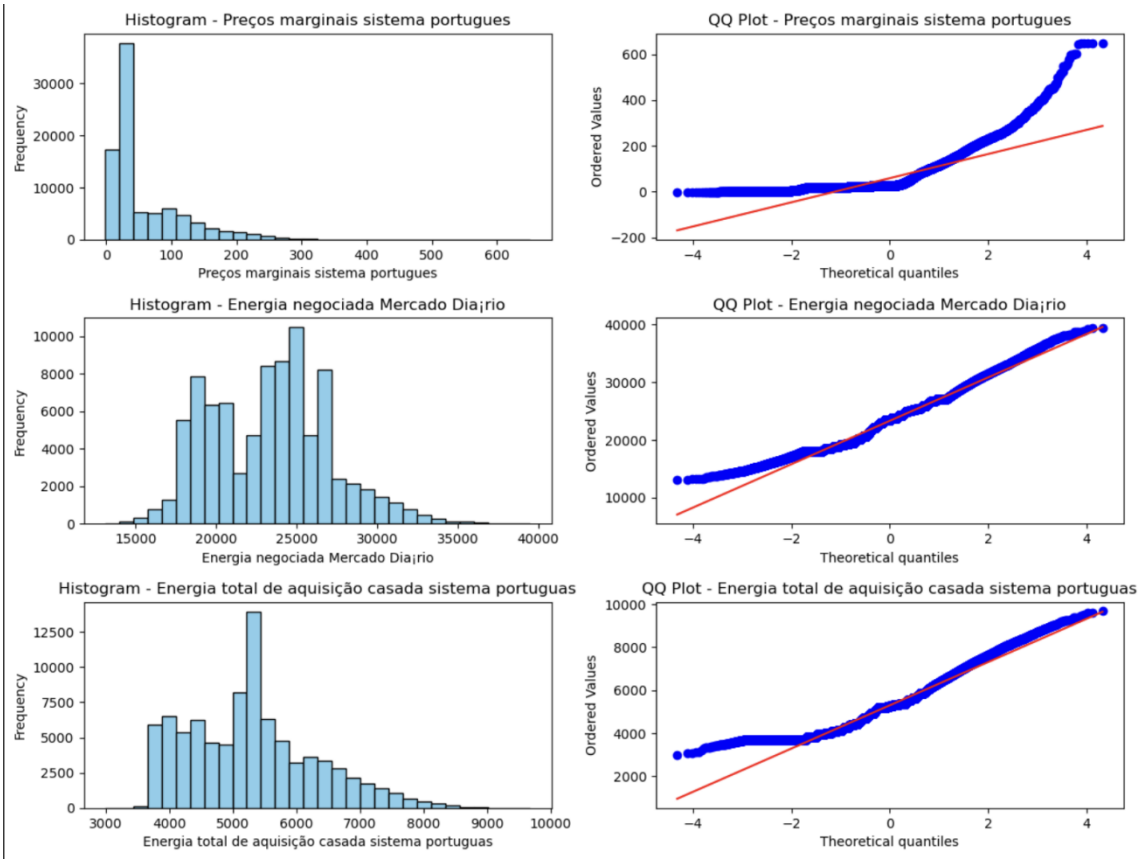



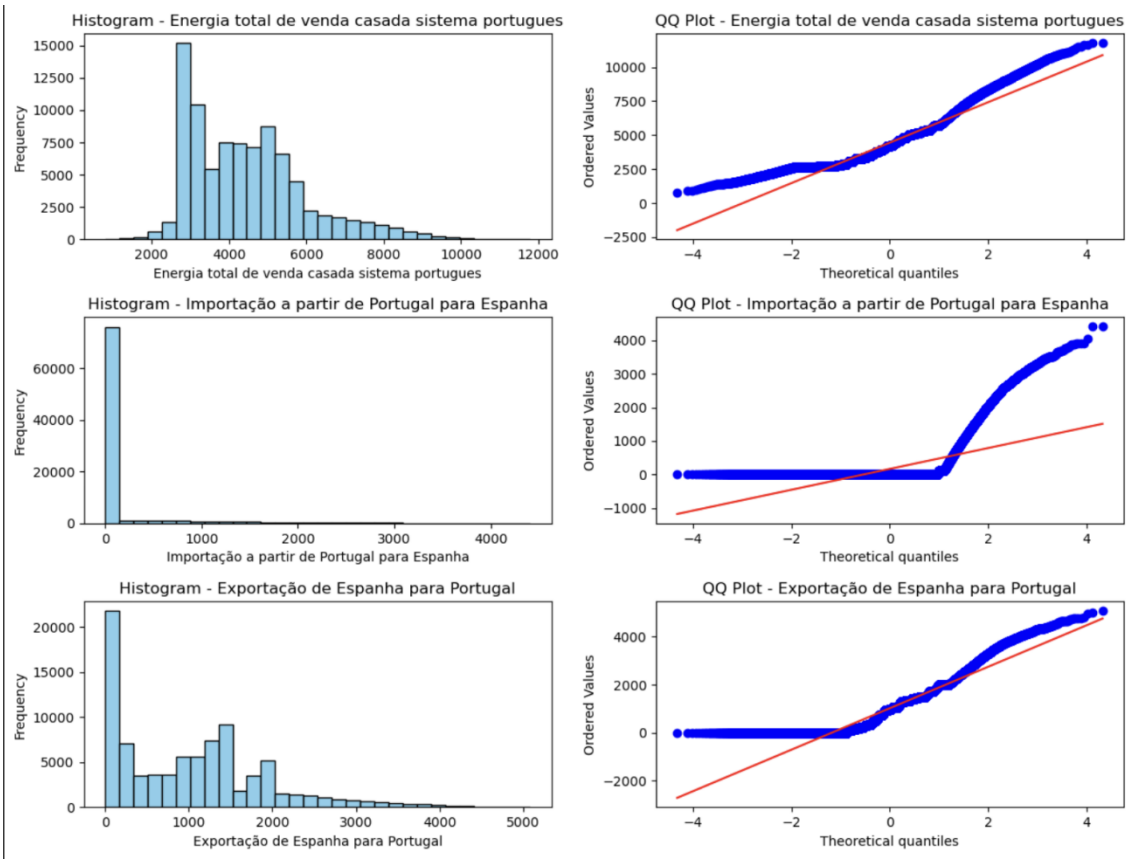

Histogram - Hydro Pumped Storage - Actual Consumption [MW]
Frequency
Hydro Pumped Storage - Actual Consumption [MW]
QQ Plot - Hydro Pumped Storage - Actual Consumption [MW]
Ordered Values
Theoretical quantiles
Histogram - Hydro Run-of-river and poundage - Actual Aggregated [MW]
Frequency
Hydro Run-of-river and poundage - Actual Aggregated [MW]
QQ Plot - Hydro Run-of-river and poundage - Actual Aggregated [MW]
Ordered Values
Theoretical quantiles
Histogram - Hydro Water Reservoir - Actual Aggregated [MW]
Frequency
Hydro Water Reservoir - Actual Aggregated [MW]
QQ Plot - Hydro Water Reservoir - Actual Aggregated [MW]
Ordered Values
Theoretical quantiles

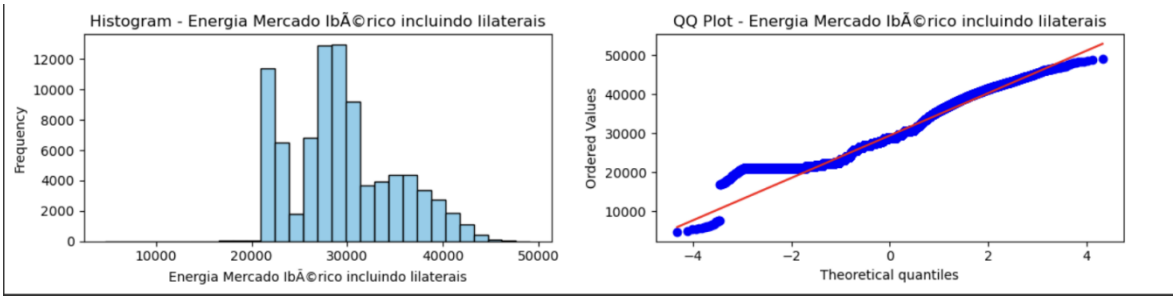

Histogram - Other - Actual Aggregated [MW]
Frequency
Other - Actual Aggregated [MW]
QQ Plot - Other - Actual Aggregated [MW]
Ordered Values
Theoretical quantiles
Histogram - Solar - Actual Aggregated [MW]
Frequency
Solar - Actual Aggregated [MW]
QQ Plot - Solar - Actual Aggregated [MW]
Ordered Values
Theoretical quantiles
Histogram - Wind Onshore - Actual Aggregated [MW]
Frequency
Wind Onshore - Actual Aggregated [MW]
QQ Plot - Wind Onshore - Actual Aggregated [MW]
Ordered Values
Theoretical quantiles


Histogram - Actual Total Load [MW] - BZN|PT
Frequency
Actual Total Load [MW] - BZN|PT
QQ Plot - Actual Total Load [MW] - BZN|PT
Ordered Values
Theoretical quantiles
Histogram - Speed_km_h
Frequency
Speed_km_h
QQ Plot - Speed_km_h
Ordered Values
Theoretical quantiles
Histogram - Angle
Frequency
Angle
QQ Plot - Angle
Ordered Values
Theoretical quantiles


Histogram - Radiação Solar (W/m²)
Frequency
Radiação Solar (W/m²)
QQ Plot - Radiação Solar (W/m²)
Ordered Values
Theoretical quantiles
Histogram - Precipitação (mm) - Peso da Régua
Frequency
Precipitação (mm) - Peso da Régua
QQ Plot - Precipitação (mm) - Peso da Régua
Ordered Values
Theoretical quantiles
Histogram - Media Temperacura °C (Lisboa, Porto, Évora)
Frequency
Media Temperacura °C (Lisboa, Porto, Évora)
QQ Plot - Media Temperacura °C (Lisboa, Porto, Évora)
Ordered Values
Theoretical quantiles


Histogram - Preços marginais sistema portugues
Frequency
Preços marginais sistema portugues
QQ Plot - Preços marginais sistema portugues
Ordered Values
Theoretical quantiles
Histogram - Energia negociada Mercado Dia¡rio
Frequency
Energia negociada Mercado Dia¡rio
QQ Plot - Energia negociada Mercado Dia¡rio
Ordered Values
Theoretical quantiles
Histogram - Energia total de aquisição casada sistema portuguas
Frequency
Energia total de aquisição casada sistema portuguas
QQ Plot - Energia total de aquisição casada sistema portuguas
Ordered Values
Theoretical quantiles


Histogram - Energia total de venda casada sistema portugues
Frequency
Energia total de venda casada sistema portugues
QQ Plot - Energia total de venda casada sistema portugues
Ordered Values
Theoretical quantiles
Histogram - Importação a partir de Portugal para Espanha
Frequency
Importação a partir de Portugal para Espanha
QQ Plot - Importação a partir de Portugal para Espanha
Ordered Values
Theoretical quantiles
Histogram - Exportação de Espanha para Portugal
Frequency
Exportação de Espanha para Portugal
QQ Plot - Exportação de Espanha para Portugal
Ordered Values
Theoretical quantiles




### 3.3.3 Analysis of Feature Distributions: Box Plot Insights

This subsection presents an analysis of key feature distributions using box plots, shown in Figure 4, highlighting central tendencies, spread, and the presence of outliers.

**Price and Market Variables:**

- "Preços marginais sistema portugues" (**Target):** The box plot for the target variable shows a compact inter-quartile range (IQR) at lower values, indicating that electricity prices are mostly low. However, a long upper whisker and numerous distinct outliers extending to very high values are evident. This confirms the presence of **frequent positive price spikes**, a critical characteristic for energy price forecasting.
- "Energia negociada Mercado Diário" & "Energia Mercado Ibérico incluindo bilaterais": These variables exhibit relatively symmetric box plots with narrow IQRs and whiskers, centered around their median values. This suggests a more stable and predictable range of daily energy trading volumes. Some outliers exist at both lower and upper extremes, representing unusual trading activity.
- "Energia total de aquisição/venda casada sistema portugues": These show moderate right-skewness, with the bulk of data at lower volumes and a tail of higher outliers, indicating that most acquisitions/sales occur at modest levels, with occasional larger transactions.
- "Importação/Exportação a partir de Portugal para Espanha": These plots are heavily skewed to the right, with a substantial portion of data at zero and many outliers representing periods of significant trade. This indicates that cross-border electricity flows are often minimal or non-existent, but can have large, infrequent bursts.

**Generation Mix Variables:**

- The below-mentioned variables consistently show extremely compressed box plots near zero, with very long right whiskers and extensive outliers. This indicates that generation from these sources (or consumption for pumped storage) is **frequently zero or very low** but can occasionally reach high levels. This reflects their

intermittent nature (solar) or dispatch based on demand/market conditions (coal, pumped storage):

1. "Generation-Solar [MW]_Day Ahead/ BZN|PT"
2. "Solar-Actual_Aggregated [MW]"
3. "Hydro_Pumped_Storage-Actual_Aggregated [MW]"
4. "Hydro_Pumped_Storage-Actual_Consumption [MW]"
5. "Fossil_Hard_coal-Actual_Aggregated [MW]"

- The below-mentioned variables exhibit right-skewed distributions with a larger IQR than the previous group, but still many outliers extending to higher generation values. This suggests more consistent, but still variable, operation at lower to moderate capacities, with peaks reflecting high demand or favourable conditions:

1. "Generation-Wind_Onshore [MW]_Day_Ahead"
2. "Wind_Onshore-Actual_Aggregated [MW]"
3. "Fossil_Gas-Actual_Aggregated [MW]"
4. "Hydro_Run-of-river_and_poundage-Actual_Aggregated [MW]"
6. "Hydro_Water_Reservoir-Actual_Aggregated [MW]"

- "Biomass-Actual_Aggregated[MW]": The box plot suggests a more concentrated distribution around its median, with fewer extreme outliers compared to other renewables. This implies a more stable, potentially base-load-like, generation pattern.
- "Other-Actual_Aggregated[MW]": Displays a relatively symmetric distribution with a concentrated IQR, centered around 30-40 MW, and a typical range of outliers. This suggests a more consistent and predictable generation contribution from "other" sources.

**Load and Weather Variables:**

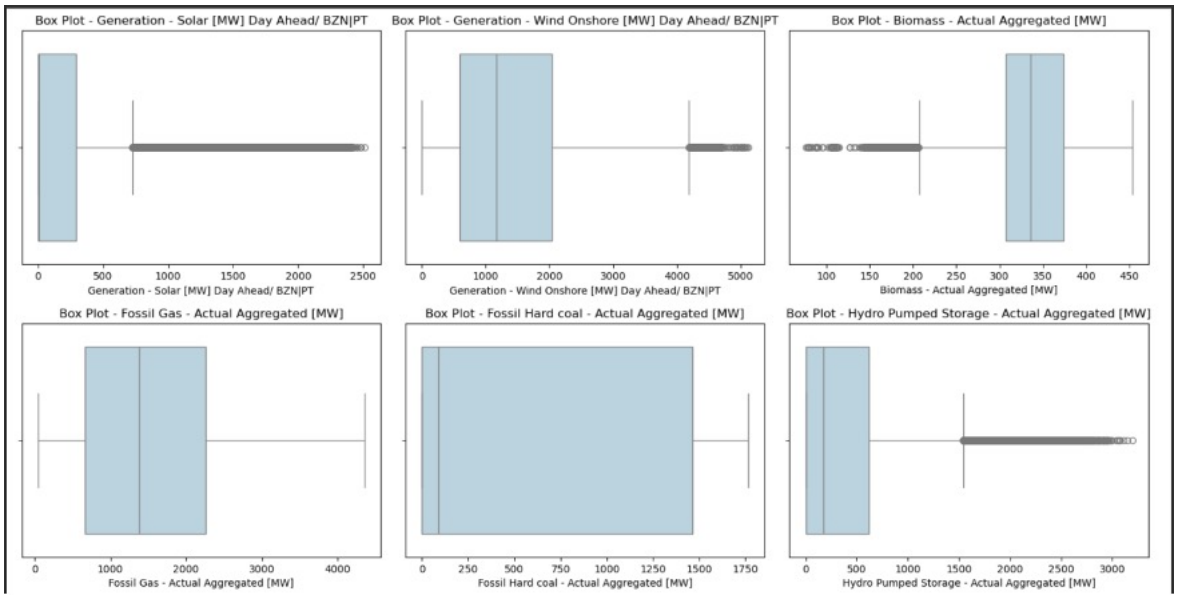
- "Actual_Total_Load[MW]": The box plot is fairly symmetric, with a well-defined IQR and whiskers, and limited outliers. This indicates that total electricity demand generally falls within a predictable range, with only rare, extreme high or low load events.
- "Speed_km_h": Shows a right-skewed distribution, with most wind speeds at lower values and a tail of higher speed outliers.
- "Angle": The box plot shows a wide range of values and no distinct central point, supporting the histogram's indication that the data are multi-modal. This means the wind tends to blow from several predominant directions rather than a single dominant one. Extreme values at both ends represent outliers.

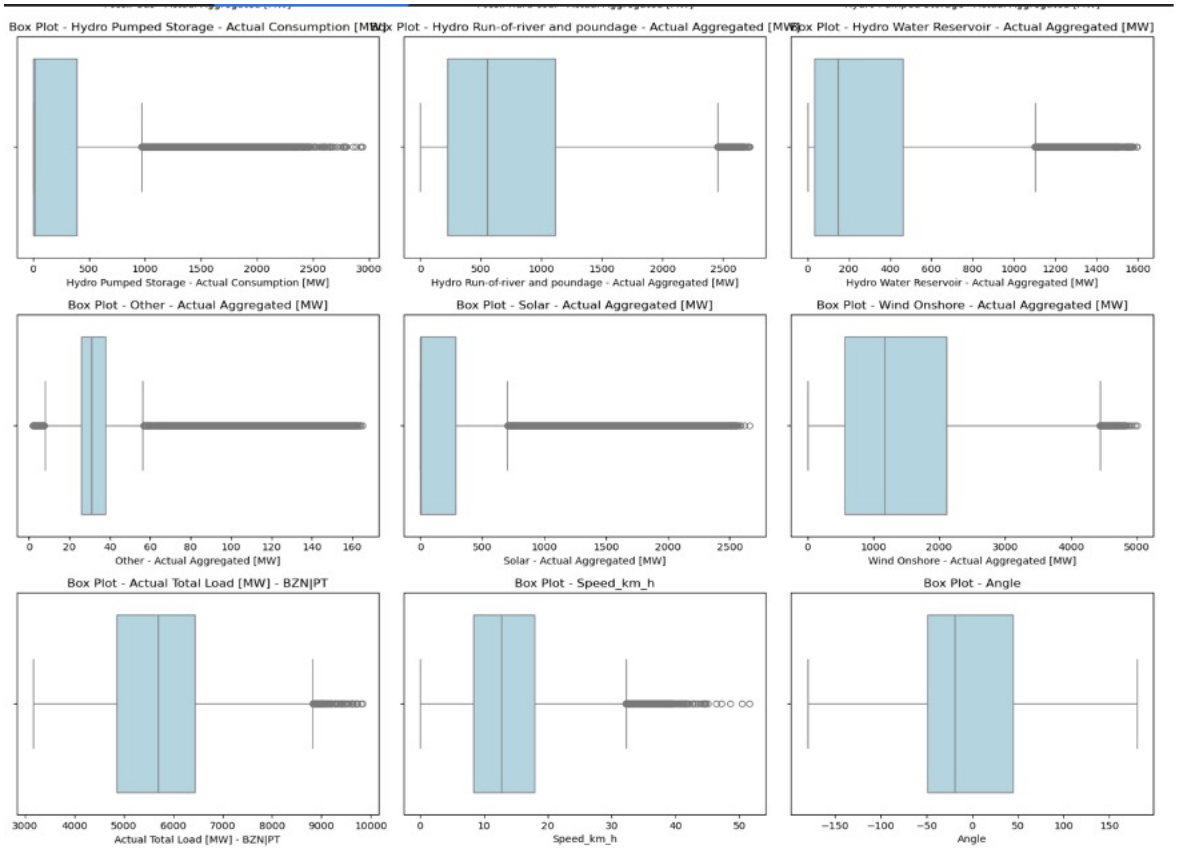
- "Radiação_Solar(W/m²)": A highly compressed box at zero with a very long whisker and many outliers to the right, confirming that solar radiation is often zero (at night) but can reach high peak values.
- "Precipitação(mm)-Peso_da_Régua": The box plot is extremely condensed at zero, with numerous outliers indicating that rainfall is mostly absent but can occur in significant amounts.
- "Media_Temperatura_°C(Lisboa,Porto,Évora)": This variable presents a relatively symmetric and well-distributed box plot with few outliers, indicating a stable and predictable temperature range, which is generally well-behaved.

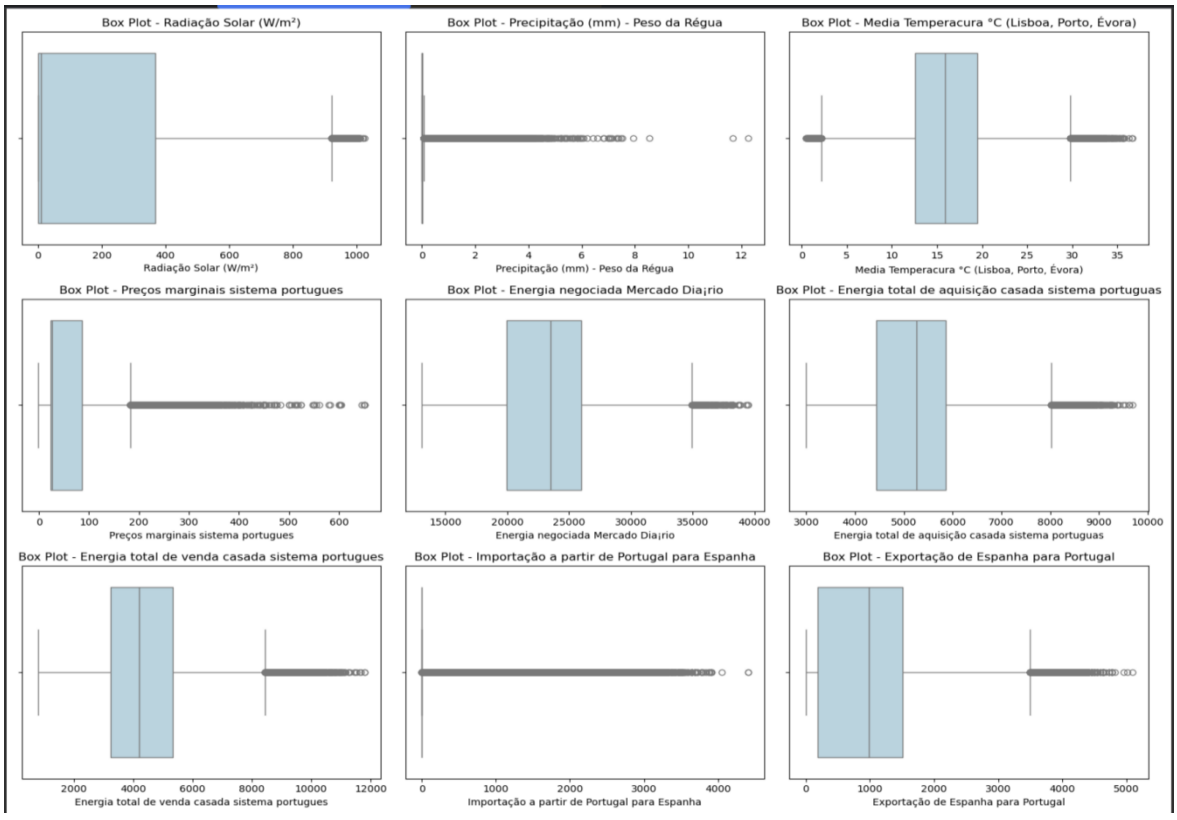
**Overall Conclusion:**

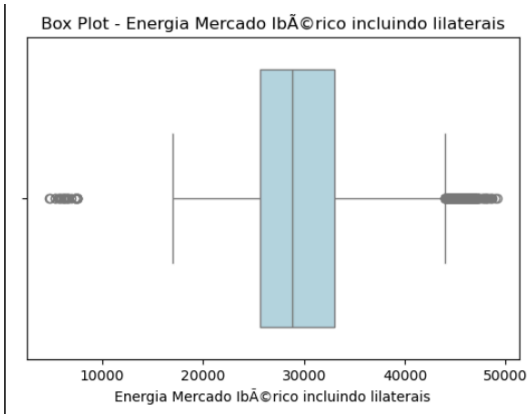
The box plot analysis strongly reinforces the highly variable and often non-normal nature of many energy generation and weather-related features, as well as the target electricity price. The prevalent right-skewness and significant presence of outliers (especially for prices and intermittent renewables) underscore the importance of using robust modeling techniques, such as tree-based ensemble models like LightGBM, which are inherently adept at handling skewed data and capturing the impact of extreme values without extensive manual outlier treatment or complex data transformations. Features like `Total Load` and `Temperature`, which show more regular distributions, are likely to be consistently strong predictors.

*Figure 4: Analysis of Feature Distributions: Box plots*




Box Plot - Hydro Pumped Storage - Actual Consumption [MW]
Hydro Pumped Storage - Actual Consumption [MW]
Box Plot - Hydro Run-of-river and poundage - Actual Aggregated [MW]
Hydro Run-of-river and poundage - Actual Aggregated [MW]
Box Plot - Hydro Water Reservoir - Actual Aggregated [MW]
Hydro Water Reservoir - Actual Aggregated [MW]
Box Plot - Other - Actual Aggregated [MW]
Other - Actual Aggregated [MW]
Box Plot - Solar - Actual Aggregated [MW]
Solar - Actual Aggregated [MW]
Box Plot - Wind Onshore - Actual Aggregated [MW]
Wind Onshore - Actual Aggregated [MW]
Box Plot - Actual Total Load [MW] - BZN|PT
Actual Total Load [MW] - BZN|PT
Box Plot - Speed_km_h
Speed_km_h
Box Plot - Angle
Angle


Box Plot - Radiação Solar (W/m²)
Radiação Solar (W/m²)
Box Plot - Precipitação (mm) - Peso da Régua
Precipitação (mm) - Peso da Régua
Box Plot - Media Temperacura °C (Lisboa, Porto, Évora)
Media Temperacura °C (Lisboa, Porto, Évora)
Box Plot - Preços marginais sistema portugues
Preços marginais sistema portugues
Box Plot - Energia negociada Mercado Dia¡rio
Energia negociada Mercado Dia¡rio
Box Plot - Energia total de aquisição casada sistema portuguas
Energia total de aquisição casada sistema portuguas
Box Plot - Energia total de venda casada sistema portugues
Energia total de venda casada sistema portugues
Box Plot - Importação a partir de Portugal para Espanha
Importação a partir de Portugal para Espanha
Box Plot - Exportação de Espanha para Portugal
Exportação de Espanha para Portugal




### 3.3.4 Correlation Analysis of Numerical Features

The correlation heatmap presented in Figure 5 reveals several important relationships between the variables that can guide feature selection and model building.

As expected, **solar-related variables** show strong internal consistency: "*Generation – Solar (Day Ahead)*" is highly correlated with "*Radiação Solar (W/m²)*" (+0.65), and "*Solar – Actual Aggregated [MW]*" also displays a strong correlation with solar radiation. This confirms that forecasted and actual solar generation are largely driven by measured sunlight levels. The solar forecast also shows a moderate positive correlation (+0.46) with "*Energia Mercado Ibérico incluindo bilaterais*", suggesting a possible seasonal or demand-related link to market activity.

In the **hydro generation category**, "*Hydro Run-of-river and poundage – Actual Aggregated [MW]*" is strongly correlated with "*Hydro Water Reservoir – Actual Aggregated [MW]*", while "*Hydro Pumped Storage – Actual Consumption [MW]*" shows a good correlation with "*Hydro Run-of-river and poundage*". These patterns suggest that some hydro variables may capture overlapping operational dynamics and could lead to redundancy if used together without adjustment.

For **wind-related variables**, "*Generation – Wind Onshore (Day Ahead)*" is positively correlated with "*Speed_km_h*", as expected from the physical relationship between wind speed and wind power output. This indicates that both measured wind speed and forecasted wind generation may provide complementary information.

**Market activity variables** also display strong internal relationships. "*Actual Total Load [MW]*" has a very strong correlation with "*Energia negociada Mercado Diário*" and good correlations with "*Energia total de aquisição casada sistema português*", "*Energia total de venda casada sistema português*", and "*Energia Mercado Ibérico incluindo bilaterais*". Similarly, "*Energia negociada Mercado Diário*" is strongly correlated with "*Energia total de aquisição casada sistema português*" and "*Energia Mercado Ibérico incluindo bilaterais*", and moderately correlated

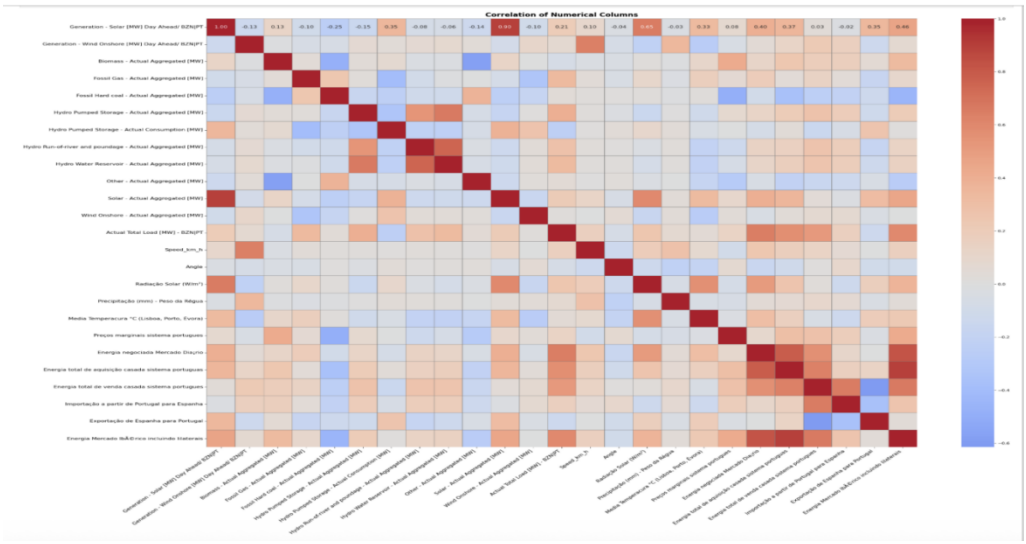
with "*Energia total de venda casada sistema português*". These high correlations suggest that market trading metrics tend to move together and may represent similar aspects of demand and trading volume.

In terms of **cross-border electricity flows**, "*Energia total de venda casada sistema português*" has a strong negative correlation with "*Exportação de Espanha para Portugal*", while "*Importação a partir de Portugal para Espanha*" is moderately negatively correlated with the same export variable. This reflects the inverse relationship between electricity imports and exports, depending on net flow direction.

Finally, **fossil fuel generation** shows an important price relationship. "Fossil Hard Coal - Actual Aggregated [MW]" has a good negative correlation with both "*Energia Mercado Ibérico incluindo bilaterais*" and "*Preços marginais sistema português*", indicating that higher coal generation tends to be associated with lower market activity and lower electricity prices.

From a modeling perspective, these findings highlight the need to handle highly correlated variables carefully to avoid multicollinearity. In particular, solar-related variables, hydro measures, and market activity indicators may require dimensionality reduction or selective inclusion. Conversely, variables such as coal generation, cross-border flows, and weather–generation interactions could be valuable predictors of price dynamics and should be retained for further testing.

*Figure 5: Correlation of Numerical features*




### 3.3.5 Correlation Analysis of Numerical Features with Target Variable

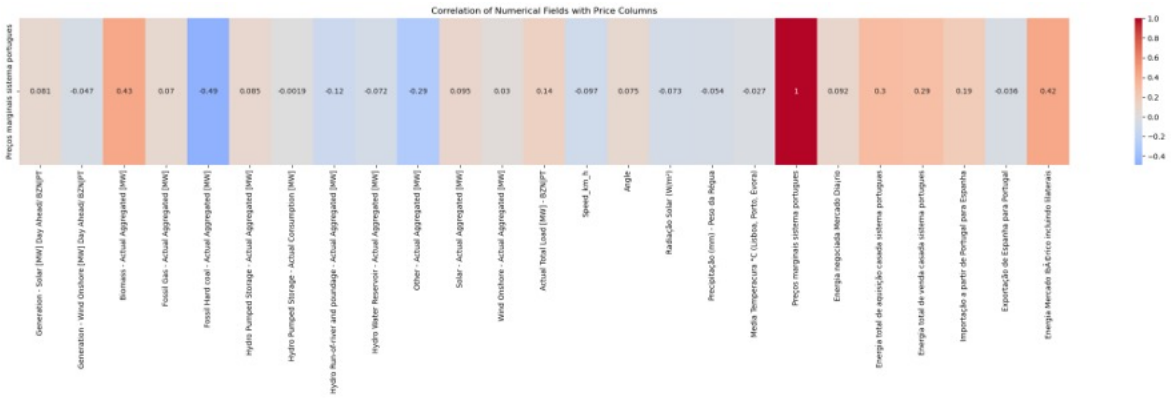
In this subsection, the correlation between the target variable "*Preços marginais sistema português*" and all the other numerical variables in the dataset is analyzed and illustrated in Figure 6 using Pearson's correlation coefficient.

The results indicate that the strongest positive correlation is observed for "Biomass - Actual Aggregated [MW]" (+0.43), suggesting that higher biomass generation is generally associated with higher electricity prices. A similarly strong positive relationship is found for "Energia Mercado Ibérico incluindo bilaterais" (+0.42), indicating that greater total energy traded in the Iberian market, including bilateral contracts, tends to coincide with price increases.

On the other hand, the strongest negative correlation is seen for "Fossil Hard Coal - Actual Aggregated [MW]" (r = –0.49), implying that higher coal-based electricity generation is typically associated with lower prices. Another notable negative correlation is observed for "Other - Actual Aggregated [MW]" (r = –0.29), indicating that increased generation from other, less common sources also tends to reduce prices.

Most other variables, including those related to "hydroelectric generation", "solar generation", "weather conditions" (e.g., wind speed, solar radiation, precipitation, temperature), and cross-border electricity flows, show very weak correlations with the target variable. This suggests that, within the studied period, these factors had limited direct influence on price movements compared to biomass generation, coal generation, and Iberian market trading volumes.

*Figure 6: Correlation of Numerical features with price only*




## 3.3.6 Correlation Analysis of Temporal Features

In this subsection, temporal components were extracted from the existing date and time fields to evaluate their linear relationship with the electricity price.

### Datetime Parsing and Feature Extraction

The "Data e Dia da Semana" column, which stores full timestamps, was converted to Python's datetime format. From it, the following calendar-based features were extracted:

- "Hour": Parsed from the "Hora" column.

- "Day", "Month", and "Year": Extracted from the main date field.

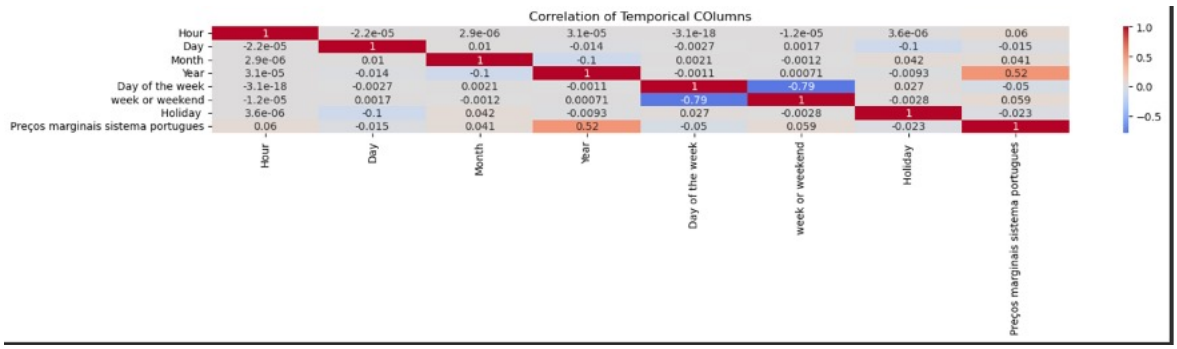
This step transforms raw timestamps into structured temporal variables for analysis.

The correlation heatmap in Figure 7 reveals several relationships. For example, "*Day of the Week*" and "*Week or Weekend*" have a strong negative correlation (r = –0.79) because one is directly derived from the other—weekdays and weekends are coded differently, so they essentially carry the same information. Similarly, "*Month*" and "*Year*" show a small negative correlation (r = –0.10), likely because the data does not cover every month in every year, creating an imbalance.

On the other hand, variables such as "*Hour*", "*Day*", and "*Holiday*" have little or no correlation with most other time-based features, meaning they provide more independent information. "*Holiday*" also has a slight negative correlation (r = –0.10) with "*Month*", suggesting certain months have more or fewer holidays in the data.

From a modeling perspective, strong correlations (like between "*Day of the Week*" and "*Week or Weekend*") can cause redundancy, so it may be better to keep only one to avoid multicollinearity, while low correlations suggest variables can contribute unique, non-overlapping insights for prediction.

*Figure 7: Correlation of Temporal Features*




### 3.3.7 Correlation Analysis Temporal Features with Target Variable

A correlation matrix in Figure 8 was computed to quantify the linear relationship between temporal features and the target variable "Preços marginais sistema portugues". The matrix included, "Hour of the day", "Day of the month", "Month", "Year, Day of the week", "Weekend indicator", "Holiday" indicator. The analysis provides insights into how temporal factors may influence electricity prices and, therefore, their potential relevance in predictive modeling.

The "Hour" variable showed a very weak positive correlation with "Preços marginais sistema portugues" (r = 0.060). This suggests that while prices may vary slightly

depending on the hour, the effect is minimal. Nevertheless, intraday patterns in demand and renewable generation could still make the "*Hour*" variable valuable in capturing daily price fluctuations in non-linear models.

The correlation between "*Day*" and "Preços marginais sistema portugues" is negligible (r = -0.015). This indicates that the position of a day within a month has no strong linear relationship with price levels. Its predictive usefulness is likely limited unless it interacts with other variables (e.g., seasonal demand patterns).

"*Month*" exhibits a small positive correlation with "Preços marginais sistema portugues" (r = 0.041). While weak, this could indicate a minor seasonal effect on prices, possibly due to changes in weather, demand patterns, or renewable generation across different months.

"*Year*" has a moderately strong positive correlation with "Preços marginais sistema portugues" (r = 0.52), making it one of the most relevant temporal predictors. This reflects a general upward or downward trend in prices over time, possibly driven by market changes, fuel costs, or regulatory factors.

The correlation between "*Day of the Week*" and "Preços marginais sistema portugues" is very small (r = -0.050). This implies that prices do not vary systematically by weekday in a strong linear fashion. However, operational and market factors such as industrial demand may still introduce patterns that a model could capture in a non-linear way.

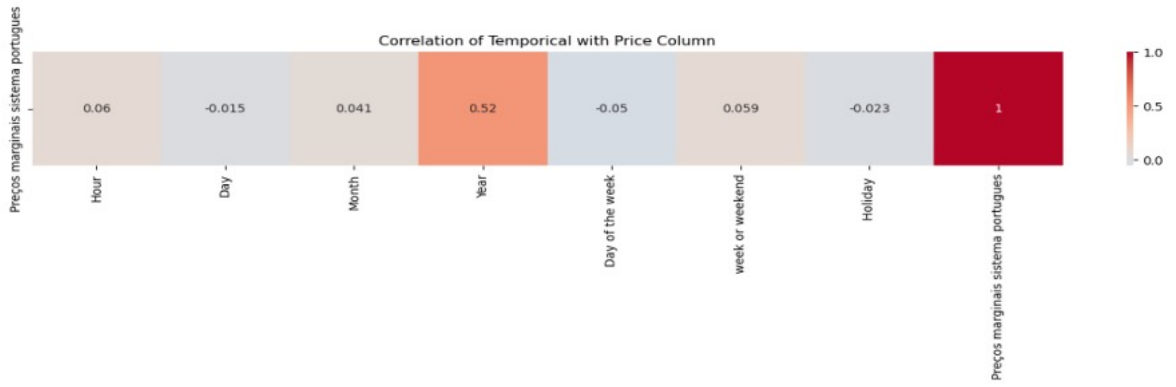
The "Weekday or Weekend" variable shows a very small positive correlation with "Preços marginais sistema portugues" (r = 0.059). While the effect is minimal, it might still help differentiate demand and generation patterns between weekdays and weekends.

The "Holiday" variable has a weak negative correlation with "Preços marginais sistema portugues" (r = -0.023). This suggests that, on average, holidays are slightly associated with lower prices, possibly due to reduced industrial demand. However, the relationship is not strong enough to be a major factor in isolation.

**Overall Implications for Modeling**

Among the temporal variables examined, "Year" stands out as the most significant predictor, likely reflecting long-term market trends. "*Hour*", "*Weekday/Weekend*", and "*Month*" show weak but potentially useful relationships that may become important in combination with other features. Variables such as "*Day*" and "*Holiday*" have minimal standalone correlation but may still contribute in interaction-based or non-linear modeling approaches.

*Figure 8: Correlation of Temporal Features with Price(Target variable)*




### 3.3.8 One-Hot Encoding of Directional Wind Features

In the dataset, the `Direction` (Wind Direction) variable was originally stored as a categorical label (e.g., "N", "NE", "E"), representing discrete wind directions. To incorporate this feature into the correlation analysis, the categorical values were transformed into a binary format using one-hot encoding. This process creates separate indicator variables for each possible direction, where a value of 1 denotes the presence of that specific direction at a given observation and 0 indicates otherwise. The resulting binary variables were then concatenated with the "*`Preços marginais sistema português`*" (Portuguese market marginal price) column, and the pairwise correlations were computed. This approach enables the identification of which specific wind directions exhibit stronger or weaker statistical associations with electricity prices, without imposing an artificial numerical ordering on the categorical values.

### 3.3.9 Correlation Analysis of Wind Direction Features

Figure 9 presents the correlation matrix for one-hot encoded wind direction variables and Portuguese marginal electricity prices. As expected, each variable is perfectly correlated with itself, resulting in a correlation coefficient of 1 along the diagonal. The off-diagonal values indicate mostly weak negative correlations between different wind directions (ranging approximately from -0.03 to -0.14). This outcome is consistent with the physical nature of wind data: when wind originates from a given direction, it is unlikely to simultaneously come from another direction, especially from opposing sectors.

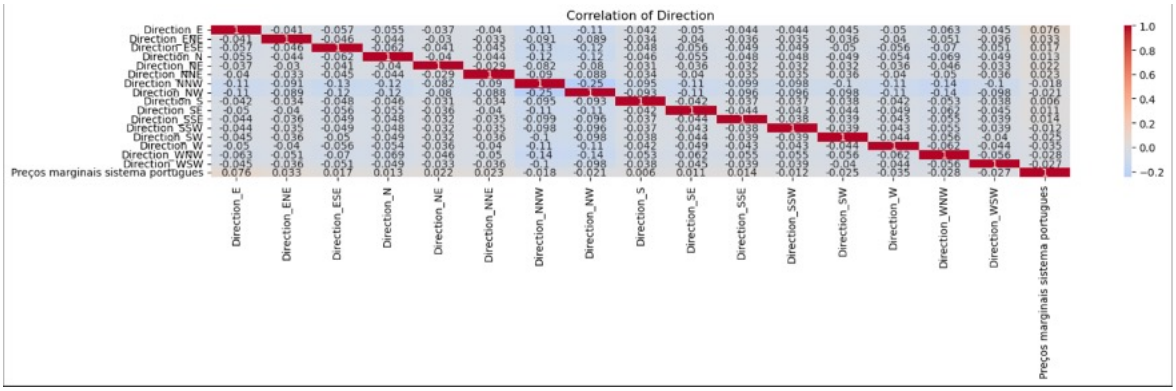
The strongest observed negative correlation occurs between "`Direction_NW`" and "`Direction_NNW`" ($r = -0.25$), reflecting the mutual exclusivity between these closely related but distinct directional categories.

The correlations between individual wind directions and the Portuguese marginal electricity price variable (`"Preços marginais sistema portugues"`) are minimal, mostly

between -0.03 and 0.08. This suggests that wind direction alone has a negligible direct influence on electricity price fluctuations within the dataset.

Overall, the analysis indicates no presence of multicollinearity concerns among the wind direction variables, and their contribution to price variability appears limited when considered in isolation.

*Figure 9: Correlation Analysis of Wind Direction Features*



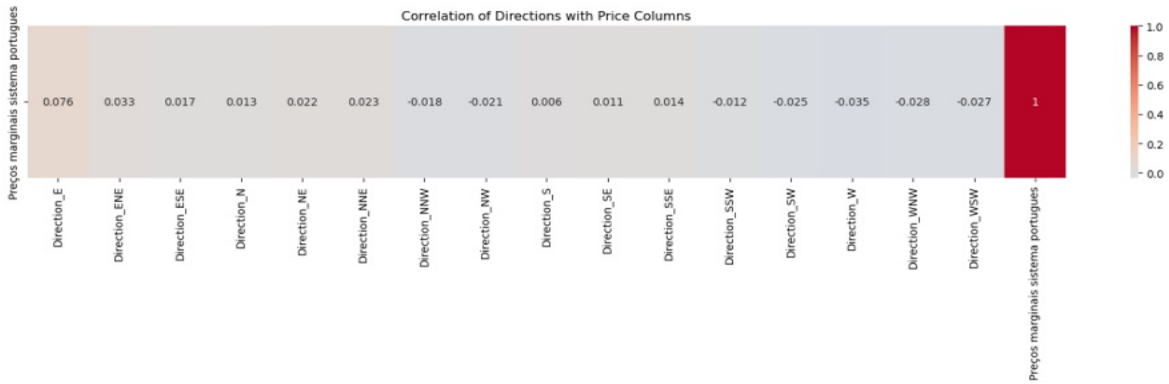

### 3.3.10 Correlation Analysis of Wind Direction Features with Target Variable

The correlation analysis shown in Figure 10 between wind direction categories and "`Preços marginais sistema português`" shows no meaningful linear relationships. The strongest observed correlation is for "`Direction_E`" (r = 0.076), followed by "`Direction_ENE`" (r = 0.033), both of which are extremely weak. Negative values, such as "`Direction_W`" (r = –0.035), are also very small. This means that, on their own, wind directions do not have a clear direct effect on price.

Because wind directions are stored as one-hot encoded variables, only one direction can be "active" (1) at a time for each observation, while the rest are "inactive" (0). This makes the columns linked to each other — if we know all but one direction, we can work out the missing one. In statistics, this is called perfect multicollinearity. In linear models (like linear regression), this can cause problems and usually means one of the direction columns needs to be removed. However, in tree-based models (like LightGBM or XGBoost), this is not an issue, so all directions can be kept in the dataset.

Even though the individual correlations are weak, wind direction could still influence prices in more complex, indirect ways. Models that can capture non-linear patterns and interactions between variables might still find useful signals from these features.

*Figure 10: Correlation Analysis of Wind Direction Variables with Target Variable*



# 4. Chapter 4: Final Data Preparation, Predictive Modeling, and Model Selection

This chapter presents the final form of the dataset and the modeling approaches used to predict electricity market prices in Portugal. It begins by detailing the preprocessing and data quality checks conducted to finalize the feature set, followed by a description of how the dataset evolved through iterative engineering stages. Next, a comparative evaluation of several machine learning models, suitable for predictive tasks, considering a continuous target variable, is provided, highlighting the performance of different algorithms on the same engineered feature space.

## 4.1 Dataset Evolution and Feature Changes

As the project progressed from initial exploratory analysis to model development, the dataset underwent important transformations in structure and content. The version used in the early EDA phase contained 38 features, including raw energy generation data, weather variables, electricity prices, and some market transaction indicators. The initial dataset also included information related to the Spanish electricity market, such as "*Preços marginais sistema espanhol*" and bilateral trade volumes, which were later deemed unnecessary for the scope of the work undertaken within this internship project, which focuses exclusively on the Portuguese market.

In contrast, the finalized dataset used for modeling comprises **47 features**, reflecting the integration of several engineered variables and the removal of extraneous or redundant columns.

Worth mentioning additions include calendar-related variables such as "`Year`", "`Month`", and "`Day`", which were extracted from the original timestamp to capture time-based patterns. The "`Angle`" feature—representing wind direction in degrees—was already present in the earlier dataset and retained in the final version. To enrich its modeling utility, this circular variable was further one-hot encoded into **16** binary directional features (e.g., "`direction_N`", "`direction_NE`"), enabling models to learn potential relationships between specific wind directions and renewable energy output.

Table 11 provides a comparative overview of key changes between the original EDA dataset and the final engineered version. The final names of columns used are also shown in the table.

*Table 11: Original and Final Dataset: comparison of feature sets before and after filtering and engineering*

| Feature Category | Original EDA Dataset (~38 features) | Final Dataset (47 features) | Change Description |
|---|---|---|---|
| Date & Calendar Features | `Data e Dia da Semana`, `Hora`, `Day of the week`, `week or weekend`, `Holiday` | Same, plus: `Year`, `Month`, `Day` | Added calendar breakdown |
| Load Forecast & Demand | `Day-ahead Total Load Forecast [MW]`, `Actual Total Load [MW]` | Only: `Actual Total Load [MW]` | Forecast load removed |
| Energy Generation (Portugal) | Renewable and fossil generation features including solar, wind, hydro, biomass, etc. | Retained with cleaned names | Retained core generation features |
| Hydro Reservoir | `Hydro Water Reservoir - Actual Aggregated [MW]` | Removed | Excluded due to low relevance or quality |
| Weather Variables | Wind speed, direction (`Direction`), angle, solar radiation, precipitation, temperature | Retained most; replaced `Direction` with one-hot encoding of `Angle` | Encoded wind direction; simplified temperature fields |
| Wind Direction | `Direction` (categorical) | 16 binary features (e.g., `direction_N`, `direction_SW`) | One-hot encoded wind angle |
| Electricity Prices | Portuguese and Spanish marginal prices | Only: "`Preços marginais sistema português`" | Removed Spanish price |
| Market Transactions | Portuguese and Spanish acquisition, sale, import/export columns | Retained only Portuguese-side transactions | Dropped Spanish-related trade features |
| Temperature Fields | Lisboa, Porto, Évora, and their average | Only: `Lisboa temperature °C` | Simplified temperature representation |
| Text Encoding Fixes | Included characters like …`vora`, `PreÁos`, `Peso da RÈgua` | Cleaned and standardized (e.g., `Évora`, `Preços`, `Peso da Régua`) | Applied proper decoding for readability |

This structured evolution reflects a targeted effort to refine the dataset for modeling accuracy and relevance, enhancing the quality and interpretability of machine learning inputs. The final dataset with **47 features** is presented in Table 12.

*Table 12: Final Dataset Feature Overview*

| Feature Name | Description |
|---|---|
| *`Data e Dia da Semana`* | Timestamp column (date + weekday) |
| *`Hora`* | Hour of the day (0–23) |
| *`Year`, `Month`, `Day`* | Calendar components extracted from timestamp |
| *`Day of the week`* | Day name or index (e.g., Monday, Tuesday...) |
| *`week or weekend`* | Binary indicator: Weekday (0), Weekend (1) |

| *Feature Name* | *Description* |
|---|---|
| *Holiday* | Binary indicator for national/regional holidays |
| *Generation – Solar [MW] Day Ahead/ BZN* | Predicted solar energy generation for tomorrow (in MW) |
| *`Generation – Wind Onshore [MW] Day Ahead/ BZN* | Predicted onshore wind energy for tomorrow (in MW) |
| *Biomass – Actual Aggregated [MW]* | Actual biomass generation |
| *Fossil Gas – Actual Aggregated [MW]* | Actual fossil gas-based generation |
| *Fossil Hard coal – Actual Aggregated [MW]* | Actual coal-based generation |
| *Hydro Pumped Storage – Actual Aggregated [MW]* | Actual hydro pumped storage generation |
| *Hydro Pumped Storage – Actual Consumption [MW]* | Consumption for hydro pumping |
| *Hydro Run-of-river and poundage – Actual Aggregated [MW]* | Run-of-river hydro generation |
| *Other – Actual Aggregated [MW]* | Other miscellaneous generation sources |
| *Solar – Actual Aggregated [MW]* | Actual solar generation |
| *Wind Onshore – Actual Aggregated [MW]* | Actual onshore wind generation |
| *Actual Total Load [MW] – BZN* | Real electricity demand (in MW) |
| *Speed_km_h* | Wind speed in km/h |
| *Angle* | Wind direction in degrees (0–360) |
| *Radiação Solar (W/m²)* | Solar radiation in W/m² |
| *Precipitação (mm) – Peso da Régua* | Precipitation in mm (Peso da Régua station) |
| *Lisboa temperature ° C* | Ambient temperature in Lisbon (°C) |
| *Preços marginais sistema portugues* | Target variable: Portuguese marginal electricity price |
| *Energia negociada Mercado Diário* | Daily traded energy volume in market |
| *Energia total de aquisição casada sistema portugues* | Total energy acquired in matched market transactions |
| *Energia total de venda casada sistema portugues* | Total energy sold in matched market transactions |
| *Exportação de Espanha para Portugal* | Electricity exports from Spain to Portugal |
| *Energia Mercado Ibérico incluindo bilaterais* | Total Iberian market energy (including bilateral) |
| *direction N, direction NE, …, direction WSW* (16 total) | One-hot encoded wind direction categories based on `Angle` |

For the sake of traceability Table 13, Table 14, Table 15 show respectively which features in the final dataset came from each data source - **OMIE**, **REN Data Hub**, and **Copernicus Climate Data Store**.

*Table 13: Features from OMIE (Iberian Electricity Market Operator)*

| Feature Name | Description |
|---|---|
| `Preços marginais sistema portugues` | Portuguese marginal electricity price (target variable) |
| `Energia negociada Mercado Diário` | Energy volume traded in the daily electricity market |
| `Energia total de aquisição casada sistema portugues` | Total matched energy acquired in the Portuguese system |
| `Energia total de venda casada sistema portugues` | Total matched energy sold in the Portuguese system |
| `Exportação de Espanha para Portugal` | Cross-border electricity export to Portugal from Spain |
| `Energia Mercado Ibérico incluindo lilaterais` | Total Iberian market energy including bilateral contracts |
| `Year, Month, Day, Day of the week, Hora` | Time breakdown extracted from timestamps related to OMIE price/timing |
| `Holiday, week or weekend` | Calendar tags added based on OMIE market calendar |

*Table 14: Features from REN (Rede Eléctrica Nacional - Portugal Grid Operator)*

| Feature Name | Description |
|---|---|
| `Generation - Solar [MW] Day Ahead/ BZN` | Predicted solar energy generation for tomorrow (in MW) |
| `Generation - Wind Onshore [MW] Day Ahead/ BZN` | Predicted onshore wind energy for tomorrow (in MW) |
| `Biomass - Actual Aggregated [MW]` | Actual biomass generation |
| `Fossil Gas - Actual Aggregated [MW]` | Actual fossil gas generation |
| `Fossil Hard coal - Actual Aggregated [MW]` | Actual coal-based generation |
| `Hydro Pumped Storage - Actual Aggregated [MW]` | Actual pumped storage generation |
| `Hydro Pumped Storage - Actual Consumption [MW]` | Pumped storage consumption |
| `Hydro Run-of-river and poundage - Actual Aggregated [MW]` | Run-of-river hydro generation |
| `Other - Actual Aggregated [MW]` | Other aggregated sources of generation |
| `Solar - Actual Aggregated [MW]` | Actual solar generation |
| `Wind Onshore - Actual Aggregated [MW]` | Actual wind generation |
| `Actual Total Load [MW] - BZN` | Real electricity demand (in MW) |

*Table 15: Features from Copernicus Climate Data Store (CDS)*

| Feature Name | Description |
|---|---|
| `Speed km h` | Wind speed in kilometers per hour |
| `Angle` | Wind direction in degrees (0–360) |
| `Radiação Solar (W/m²)` | Surface solar radiation in watts per square meter |
| `Precipitação (mm) - Peso da Régua` | Total daily precipitation at Peso da Régua |
| `Lisboa temperature ° C` | Daily average temperature in Lisbon |
| `direction N, direction NE, ..., direction_WSW` | One-hot encoded directional wind categories based on `Angle` |

## 4.2 Final Data Summary and Quality Checks

Before performing feature engineering and modeling, a summary analysis was conducted (as shown in Appendix B) to validate the final dataset structure in Table 12 and assess

potential quality issues. That helped to generate descriptive statistics, detect anomalies, and confirm data integrity.

The results serve as a final checkpoint following EDA, and they help to ensure that the dataset is ready for downstream processing. The analysis included four main steps:

**STEP 1: Annual Average Price Trend:**

The first part of the analysis computes the average electricity price per year using the column "`Preços marginais sistema Portuguese`", the target variable for forecasting. This step provides a high-level view of market evolution over time and helps to identify trends, structural shifts, or price regimes. The output is a table (Table 16) with the average price for each year in the dataset and a bar chart visualization (Figure 11) that highlights price fluctuations year-over-year.

This visual and statistical summary in Figure 11 reinforces the temporal nature of electricity prices and supports the later use of calendar-based features (e.g., year, month) in modeling. The output shows that the average electricity price in Portugal shows significant variation over the years. Prices were relatively low in 2020 (€36.22/MWh), surged dramatically in 2021 (€112.00/MWh) and 2022 (€167.91/MWh), then declined in 2023 (€88.26/MWh) and 2024 (€63.44/MWh). A moderate increase is observed in 2025 (€94.17/MWh). These fluctuations likely reflect global energy market volatility, regulatory changes, and the impact of renewable integration. This trend further justifies the inclusion of calendar-based features (e.g., year, month) to capture temporal dynamics in modeling.

*Figure 11: Average Electricity price per year (2020-2025)*

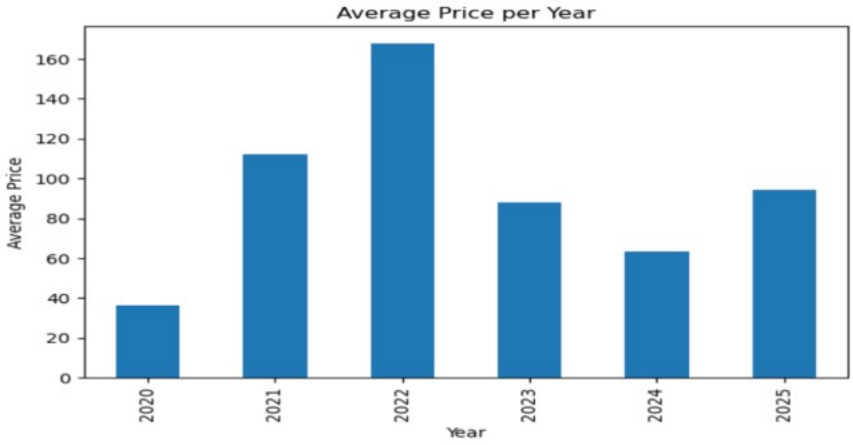



Figure 12 shows a more detailed view, depicting the average electricity price per month within the period 2023-2025.

*Figure 12: Average Monthly Electricity Price Per Year*


The results in Table 16 show the average electricity price per year from 2020 to 2025, likely in €/MWh (or €/kWh if already converted).

*Table 16: Average electricity price per year*

| *Year* | *Average Price (€/MWh or €/kWh)* | *Meaning* |
|---|---|---|
| *2020* | 36.22 | Low prices – possibly due to COVID-19 and reduced demand |
| *2021* | 112.00 | Big increase – start of energy crisis and market volatility |
| *2022* | 167.91 | Very high – peak of energy crisis (Ukraine war, gas shortages) |
| *2023* | 88.26 | Prices cooled down, but still elevated |
| *2024* | 63.44 | Stabilizing – market calming, better renewables/gas conditions |
| *2025* | 94.17 | Moderate increase – possibly due to seasonal or supply-demand shifts |

**Insights:**

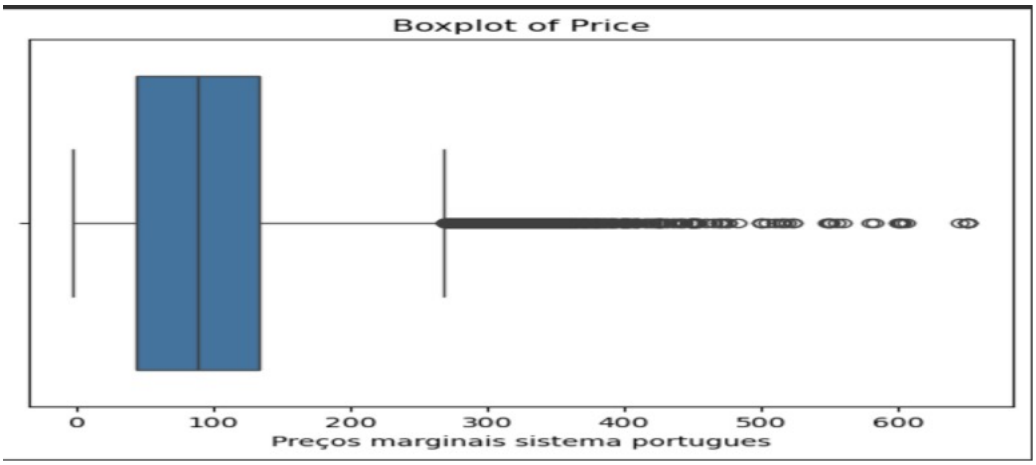
- **2020** had the lowest prices**,** likely due to the COVID-19 pandemic reducing electricity demand.
- **2021–2022** shows a huge spike**,** consistent with the real-world energy crisis.
- **2023–2025** suggest a return toward stability**,** though 2025 shows a slight rise again.

**STEP 2: Outlier Detection Using IQR**

To identify extreme price values that could distort modeling results, outliers in the target variable "`Preços marginais sistema portugues`" were detected using the Interquartile Range (IQR) method. The IQR is defined as the range between the first (Q1) and third (Q3) quartiles. Values lying below Q1 − 1.5×IQR or above Q3 + 1.5×IQR were considered outliers.

**Results:** As shown in Figure 13, a total of 769 outliers were identified in the electricity price data. These primarily represent unusually high price spikes, which are a known characteristic of electricity markets, especially during demand surges or supply shortages.

*Figure 13: Boxplot for Electricity price spikes*




The boxplot in Figure 13 clearly illustrates the behavior that most data points are concentrated in the lower price range. The upper whisker extends significantly, with numerous individual outliers beyond 200 €/MWh—some even exceeding 600 €/MWh. These outliers confirm the heavy-tailed, right-skewed distribution already observed in the EDA phase. Their presence justifies the use of robust, non-linear models (e.g., LightGBM) and scaling techniques (such as RobustScaler) that are less sensitive to extreme values.

**STEP 3: Missing Values Summary**

A thorough check for missing values was conducted across all features. The result showed that no null values were present in the dataset, confirming its completeness and readiness for modeling. This step verifies that previous preprocessing steps (such as forward-fill imputation and data integration) were successful and that no additional handling of missing data is required at this stage.

**STEP 4: Duplicate Rows Check & Exporting Results**

An analysis was performed to identify any duplicate records that might exist in the dataset. The result showed that no duplicate rows were found**,** indicating that the data is uniquely indexed and free from redundancy. This reinforces the integrity of the dataset and reduces the risk of data leakage or bias during model training.

The script detects and outputs any duplicated records. All outputs — including the average prices, outlier list, missing values, and duplicates —are saved to a single Excel file (**result.xlsx**) with separate sheets for documentation. This not only helps to keep records of the final data checks but also facilitates sharing results with collaborators or including them in appendices.

This summary analysis validated the quality and structure of the dataset after EDA and before applying advanced transformations. It ensures confidence in the data pipeline and motivates the upcoming feature engineering techniques — such as calendar-based variables,

lag features, rolling statistics, and the inclusion of external data sources (e.g., weather and demand indicators) — which are introduced in the next section.

## 4.3 Unified Modeling Approach for All Algorithms

Before selecting the final model, an extensive experimental phase was conducted using several machine learning algorithms, suitable for predictive tasks, considering a continuous target variable "`Preços marginais sistema portugues`". This phase aimed to evaluate the general performance of different models on the electricity price prediction task and to identify which techniques were most suitable for further development. Although a formal pipeline structure was implemented only after the final model was selected, the same methodological framework was applied consistently across all models during this stage. The set of 7 models included Decision Tree, Random Forest, XGBoost, LightGBM, Extra Trees, CatBoost and AdaBoost, as shown in Table 17.

The modeling process began with substantial feature engineering tailored for time series regression. Specifically, lag features were generated for key numerical variables such as electricity generation by source (e.g., solar, wind, hydro), weather conditions (e.g., temperature, solar radiation, precipitation), and energy market indicators. Lagged values were calculated on a daily basis, extending up to seven days (i.e., 168 hours) to capture temporal dependencies. Additionally, time variables such as hour of day and month of year were encoded using sine and cosine transformations to reflect their cyclical nature.

To ensure reproducibility and comparability across models, the dataset was divided into training, validation, and test sets using an 80/20 split, followed by a further 20% split of the training data to serve as a validation set. All models were trained and evaluated using the same feature set and the same data partitions. The performance metric used across all models was Mean Absolute Error (MAE), which was calculated for the training, validation, and test datasets.

Hyperparameter tuning was incorporated where applicable. For instance, "RandomizedSearchCV" was applied to the XGBoost model to identify optimal parameter combinations. Moreover, feature importance scores derived from the optimized XGBoost model were used to filter the most relevant predictors. A reduced version of the XGBoost model was then trained using only the top 95% of features ranked by cumulative importance, which slightly improved generalization and interpretability.

While no formal "sklearn.pipeline.Pipeline" was implemented at this stage, the consistent data preparation and modeling approach allowed for a fair and rigorous comparison across different machine learning techniques. The insights from this comparative modeling phase

directly informed the selection and development of the final predictive model, which is further elaborated in the subsequent sections.

## 4.4 Comparative Evaluation of Machine Learning Models

Following the application of a unified modeling methodology across all candidate algorithms, a comparative evaluation was conducted to measure and contrast the performance of each model. This evaluation aimed to identify which algorithm demonstrated the best balance between accuracy, generalization, and stability when predicting electricity market prices in Portugal.

As previously stated, the 7 models assessed were Decision Tree, Random Forest, Extra Trees, LightGBM, XGBoost, CatBoost, and AdaBoost. Each model was trained using the same feature set, which included lag variables (up to seven days), cyclic time encodings (sin and cos transformations for hour and month), and relevant market, weather, and generation data.

The same data splits were maintained across all models to ensure a fair comparison: 80% of the data was used for training and validation, with 20% of the training portion reserved for validation, and the remaining 20% of the original dataset used as a holdout test set.

Performance was evaluated using the Mean Absolute Error (MAE) metric across the training, validation, and test datasets. The results are summarized in Table 17.

*Table 17: MAE Comparison of Machine Learning Models*

| ***Model*** | ***Lag Features*** | ***Cyclic Time Encoding*** | ***MAE on Training*** | ***MAE on Validation*** | ***MAE on Test*** |
|---|---|---|---|---|---|
| Decision Tree | Yes | Yes | 5.50 | 14.52 | 14.39 |
| Random Forest | Yes | Yes | 1.77 | 9.92 | 9.66 |
| *Extra Trees* | Yes | Yes | 1.43 | **9.42** | **9.14** |
| *LightGBM* | Yes | Yes | 4.57 | 9.63 | 9.38 |
| *XGBoost* | Yes | Yes | 4.15 | 9.68 | 9.43 |
| CatBoost | Yes | Yes | 8.05 | 10.63 | 10.46 |
| AdaBoost | Yes | Yes | 19.53 | 19.81 | 19.34 |

As shown in Table 17 ensemble-based tree models significantly outperformed simpler models such as Decision Tree and AdaBoost, which struggled to generalize beyond training data. The Extra Trees Regressor delivered the lowest MAE on both the validation (9.42) and test (9.14) sets, indicating robust generalization. Random Forest also performed strongly, achieving a validation MAE of 9.92 and test MAE of 9.66. LightGBM and XGBoost exhibited similar validation performance but with slightly higher error margins compared to Extra Trees.

The final goals set for this project, as specified by the Project Lead, were that the Test MAE should be close to the MAE of validation and both MAEs should be less than 5. Furthermore, the acceptable prediction error was set at a Test MAE below 5 €/MWh to ensure forecasts are sufficiently accurate for operational use. These goals will be pursued in the experiments described in the next chapter.

While CatBoost and AdaBoost were included for completeness, their results were less competitive, with notably higher MAE scores. AdaBoost in particular showed signs of underfitting and poor fit to the non-linear characteristics of the data.

Based on these findings, the three models with the most promising performance—LightGBM**,** XGBoost, and Extra Trees—were selected for further refinement in the next phase of experimentation. These models were subjected to more advanced tuning techniques and deeper feature evaluations to explore their full potential in the context of electricity price forecasting.

Although Extra Trees achieved the best performance metrics in this comparison, the final model selection also considered additional criteria such as hyperparameter flexibility, interpretability, training speed, and integration with optimization frameworks.

# 5. Chapter 5: Modeling- Experiments and Results

## Iterative Model Enhancements and Intermediate Experiments

This chapter presents the complete process of developing and evaluating predictive models for EPF. The objective was to identify the most accurate and robust machine learning approach for the task by systematically testing multiple algorithms, tuning their hyperparameters, and engineering relevant features.

The early modeling attempts are described, including the use of lag features, rolling statistics, and various regression models such as Extra Trees, XGBoost, and LightGBM. Each approach is evaluated using consistent train, validation, and test splits to ensure fair comparison. Optuna, a powerful hyperparameter optimization framework, is used to fine-tune model performance.

Through a series of iterative experiments, it is assessed how different model configurations, data preprocessing steps, and feature engineering choices affect prediction accuracy. The results from each trial are reported using the MAE as the main evaluation metric.

The chapter concludes by selecting the best-performing model—LightGBM with advanced feature engineering—for final deployment, setting the stage for the implementation and integration work described in the next chapter.

## 5.1 First Modeling Attempt

Before arriving at the final predictive model, several intermediate strategies were explored to improve model accuracy and generalization. One such approach focused on enriching the feature set and altering the problem formulation. This included the introduction of short-term lag features (1–5 hours) for the electricity price, engineering rolling statistics (mean and standard deviation) using window sizes of 24, 48, and 72 hours, and normalizing all numerical predictors using standard scaling.

Additionally, the target variable was redefined as the electricity price shifted by 168 hours (7 days), aligning the prediction task with a weekly forecasting horizon. Directional wind variables were excluded from the dataset to reduce redundancy and noise.

The tree-based models assessed in the previous Chapter —Extra Trees, LightGBM, and XGBoost—were selected for further evaluation, each undergoing hyperparameter optimization with Optuna. Results were compared using the MAE on training, validation, and test splits.

As stated previously, the final goals set for this project, according to the specification made by the Project Lead, were that the Test MAE should be close to the MAE of validation

and both MAEs should be less than 5. Furthermore, the acceptable prediction error was set at a Test MAE below 5 €/MWh, to ensure forecasts are sufficiently accurate for operational use.

Despite the competitive performance in terms of numerical MAE values, this approach was not used for final model deployment. One limitation was the use of a 7-day shifted target, which, although useful for mid-term forecasting, did not align with the business requirement of 7-day price forecasting; furthermore, the MAE values obtained seemed to be too good to be true. Additionally, the normalization step introduced interpretability challenges and required consistent scaling at deployment, complicating the production pipeline.

Nevertheless, this set of experiments, shown in Table 18, was valuable in informing subsequent modeling efforts, particularly in highlighting the predictive value of short-term lags and rolling trends, which were later integrated more effectively in the final modeling framework.

*Table 18: MAE Results for the First Modelling Attempt*

| Model | Train MAE | Val MAE | Test MAE |
|---|---|---|---|
| *LightGBM* | 0.0053 | 0.0037 | 0.0037 |
| *ExtraTrees* | 0.1225 | 0.1351 | 0.1394 |
| *XGBoost* | 0.0004 | 0.0052 | 0.0051 |

## 5.2 Second Modeling Attempt

After the initial experimentation with core models, further refinement attempts were made to improve prediction accuracy before selecting the final model. These attempts involved systematic changes in feature engineering and model evaluation design. One such approach included the creation of extended lag features for electricity prices (up to 12 hours) and the generation of rolling mean and standard deviation features over a wider range of time windows, including 96, 120, 144, and 168 hours. Direction-related variables were again excluded to reduce noise, and all features were normalized using “StandardScaler” to improve convergence for models like XGBoost and LightGBM.

This phase also introduced a more rigorous and time-aware train/validation/test split based on full calendar years: 2020–2023 for training, 2024 for validation, and 2025 for testing. Hyperparameter tuning was performed for each model using Optuna to ensure that performance comparisons were made on optimized configurations.

Table 19 summarizes the results of this modeling iteration. Despite considerable efforts, this approach did not yield satisfactory performance, particularly on the test set, where all models showed relatively high error. As a result, this methodology was not selected for final

deployment, but it provided critical insight into model behavior over different time structures and helped guide the development of the final model.

*Table 19: MAE Results for the Second Modeling Attempt*

| *Model* | *Training MAE* | *Validation MAE* | *Test MAE* |
|---|---|---|---|
| *Extra Trees* | 7.5869 | 31.5220 | 37.2930 |
| *LightGBM* | 12.7103 | 27.5529 | 38.7253 |
| *XGBoost* | 15.5315 | 27.5528 | 38.1136 |

The large MAE differences between Table 18 and Table 19 are mainly due to changes in the problem setup and evaluation strategy. In the first attempt, the task was an easier 7-day-ahead forecast with looser time splits, which likely introduced data leakage and produced unrealistically low errors. In the second attempt, the task was a harder hourly next-day forecast with stricter year-based splits, preventing leakage but increasing difficulty. These changes, along with different lag and rolling feature ranges, explain the much higher MAE values in Table 19.

## 5.3 Third Modeling Attempt

As part of the model selection process, the three advanced machine learning algorithms—Extra Trees Regressor, XGBoost**,** and LightGBM—were implemented and optimized using Optuna hyperparameter tuning. The dataset used for these experiments included lagged prices, rolling statistics, and domain-specific engineered features. All models were evaluated using consistent data splits: 2024 for training, early 2025 for validation, and later months of 2025 for testing.

The Extra Trees Regressor, a robust ensemble method, was tuned over 30 trials. Although it achieved reasonable performance on the training set (MAE: 2.61), it exhibited significant generalization issues, with a high validation MAE of 10.40 and test MAE of 10.33**,** suggesting it was not well-suited for the time-dependent nature of electricity prices.

The XGBoost model, a powerful gradient boosting framework, showed improved generalization with a validation MAE of 9.04 and test MAE of 8.81**,** but still lacked the accuracy required for deployment. Even after fine-tuning over 50 Optuna trials and incorporating lagged and statistical features, the prediction error remained high relative to operational needs.

The third LightGBM model incorporated a comprehensive set of lags, calendar, interaction, and rolling window features, along with feature scaling via “RobustScaler”. It achieved impressive training performance (MAE: 0.0513) and low-test MAE (0.3313), yet the validation MAE remained elevated at 1.0779, indicating a potential overfit to the training data and a lack of robustness in unseen validation scenarios.

*Table 20: MAE Results for the third Modeling Attempt*

| *Model* | *Train MAE* | *Validation MAE* | *Test MAE* |
|---|---|---|---|
| *Extra Trees* | 2.6078 | 10.3992 | 10.3323 |
| *XGBoost* | 2.1777 | 9.0444 | 8.8066 |
| *LightGBM (v3)* | 0.0513 | 1.0779 | 0.3313 |

## 5.4 Fourth Modelling Attempt: Feature-Enriched LightGBM with Extended Lags and Momentum

In an effort to improve model generalization and prediction accuracy, a fourth modeling attempt was conducted using LightGBM, incorporating a more extensive set of engineered features and a refined validation strategy. The feature set included lag features at multiple temporal resolutions (`1`, `3`, `5`, `7`, `12`, and `24` hours), first-order differenced values, and rolling statistics (mean, standard deviation, min, max, momentum) for three window sizes (`24`, `48`, and `72` hours). A sine-cosine transformation of the month variable was used to encode cyclical calendar patterns, and extreme target values were clipped between the 5th and 95th percentiles to mitigate the influence of outliers. The target was set as the marginal electricity price 168 hours (one week) into the future.

To tune the LightGBM hyperparameters, Optuna was employed over 250 trials, optimizing for MAE using a stratified validation set. Following hyperparameter optimization, the final model was trained on a combined training and validation dataset and evaluated on a held-out test set. This model achieved Train MAE of 1.76**,** Validation MAE of 1.73, and a significantly higher Test MAE of 7.36. While the train and validation results were closely aligned—indicating reduced overfitting—the sharp increase in test error revealed that the model struggled to generalize to future unseen periods. As such, this version was not adopted for final deployment, but it provided a strong foundation for subsequent iterations that focused on better temporal validation and feature simplification.

*Table 21: MAE Results – Fourth Modeling Attempt (Advanced LightGBM)*

| *Model Version* | *Train MAE* | *Validation MAE* | *Test MAE* |
|---|---|---|---|
| *LightGBM (Lag & Momentum v4)* | 1.7614 | 1.7318 | 7.3615 |

## 5.5 Fifth Modelling Attempt: Increasing Optuna Trials and Final Algorithm Selection

Following multiple iterations and architectural refinements, the next logical step involved optimizing model performance through increased Optuna trial counts. It was assumed that a more extensive hyperparameter search would help identify better model settings, which could improve how well the candidate models—Extra Trees, XGBoost, and LightGBM—perform on unseen data.

For the Extra Trees Regressor, trials were scaled from 10 to 200. However, results plateaued early, with no observed improvement beyond Validation MAE of 6.17 and Test MAE of 5.38, indicating that additional trials did not provide meaningful performance gains. Similarly, XGBoost with 96 Optuna trials yielded a Validation MAE of 6.79 and Test MAE of 7.15, both significantly higher than acceptable thresholds, reaffirming that the model was overfitting and not well-tuned to the underlying distribution.

Conversely, LightGBM demonstrated clear improvements with increased trials. At 100 trials**,** it achieved its best performance with a Train MAE of 0.0874**,** Validation MAE of 1.0283, and Test MAE of 0.3347. These results indicated strong generalization capacity and consistent performance across all data splits. Given this consistent outperformance, LightGBM was selected for further development and final model construction. The next phase focused on refining the LightGBM model using domain-specific features, advanced lags, and robust validation strategies.

*Table 22: MAE Results - Fifth Modeling Attempt: Increasing Optuna Trials and Final Algorithm Selection*

| ***Model*** | ***Trials*** | ***Train MAE*** | ***Validation MAE*** | ***Test MAE*** | ***Notes*** |
|---|---|---|---|---|---|
| *Extra Trees + Optuna* | 10–200 | ~0 | 6.1723 | 5.3797 | Plateaued performance |
| *XGBoost + Optuna* | 96 | 0.2708 | 6.7933 | 7.1525 | Overfitting, high generalization gap |
| *LightGBM + Optuna* | 50 | 0.0727 | 1.0893 | 0.3420 | Good performance |
| // | 100 | 0.0874 | 1.0283 | 0.3347 | **Best overall performance** |
| // | 150 | 0.0800 | 1.0156 | 0.3704 | Slight degradation |

## 5.6 Sixth Modeling Attempt: Fast LightGBM on Split Data (Fixed Parameters)

In this modeling attempt, a faster and lightweight version of the LightGBM model was explored by applying a direct training strategy on pre-split data, without engaging in additional hyperparameter tuning. The primary motivation for this approach was to test the effectiveness of previously tuned hyperparameters and evaluate the model's generalization capability across distinct data segments. For this model, a final version of the dataset was used, containing **22 core features.** The final dataset will be further explained in Chapter 6 and presented in Table 24. The dataset was divided into three logical periods:

**Training Set**: Entire year 2024

**Validation Set**: January 1 to February 14, 2025

**Test Set**: February 15 to March 20, 2025

The model used robust calendar, lag, interaction, and rolling features engineered in earlier iterations. A `RobustScaler` was applied to each subset before training, ensuring resistance to outliers in numerical distributions.

The LightGBM regressor was trained using a fixed set of previously optimized parameters (e.g., `learning_rate = 0.0462`, `num_leaves = 221`) derived from earlier experiments. No Optuna optimization or early stopping was performed in this attempt, which made the training process significantly faster.

This model provided a useful benchmark to assess how well a manually tuned model performs on temporally split data. However, its performance metrics suggested some degree of overfitting, with particularly low MAE on the training set but a sharp increase on the validation set.

*Table 23: MAE Results for the Sixth Modeling Attempt: Fast LightGBM*

| ***Dataset Split*** | ***MAE*** |
|---|---|
| *Train (2024)* | 0.0515 |
| *Validation (Jan–Feb 14)* | 1.4949 |
| *Test (Feb 15–Mar 20)* | 0.4282 |

# 6. Chapter 6: Final Model Deployment and Application Integration

This chapter presents the final steps taken to prepare the electricity price prediction model for deployment. After selecting the most promising model based on performance across training, validation, and test periods (the LightGBM model), the same configuration was retrained on the full dataset to produce the final version.

The complete machine learning pipeline is also described, including data preprocessing, feature engineering, and scaling. An explanation of how backend endpoints were developed using FastAPI to support integration with future applications is also included.

## 6.1 Final Dataset Preparation

Before training the final model, the dataset underwent an important feature reduction and cleaning step. This was done to make the model more efficient and to ensure it only used the most relevant variables for predicting electricity prices. The goal was to remove unnecessary, redundant, or low-contribution features that could add noise to the model without improving performance.

First, some time-related variables like "`Year`", "`Month`", "`Day`", and "`Day of the week`" were removed. These had already been transformed into more useful cyclical features using sine and cosine encoding (to capture seasonality), so the original columns were no longer needed. Other columns, such as "`Data e Dia da Semana`" and "`Hora`", which came in text format, were also removed because they didn't add any useful information after the main "`date`" column had been properly processed.

A set of features related to wind direction, such as "`direction_N`", "`direction_S`", and many others, was also excluded. These were one-hot encoded categories indicating the wind's compass direction at a given hour, but they turned out to have little to no impact on model accuracy. Additionally, a feature called "`Angle`", which represented wind direction in degrees, was also dropped due to low relevance and overlap with the direction categories.

Some renewable energy generation-related features were removed because they had mostly zero values throughout the dataset. For example, "`Fossil Hard coal - Actual Aggregated [MW]`" showed up very rarely and didn't offer enough variability to be useful for prediction. Similarly, a few columns with outdated or overly sparse data were dropped as they didn't help to improve the model's learning process.

Finally, we simplified the naming of some features to make the dataset cleaner and more readable. For instance, "`Radiação Solar (W/m²)`" was renamed to "`solar_radiation`, "`Precipitação (mm) - Peso da Régua`" became "`precipitation_mm`", "`Lisboa`

`temperature °C"` was shortened to "`lisbon_temp_c"` and "*`Preços marginais sistema portugues"`* to "`portugal_marginal_price"`. These renaming's helped to streamline the data processing and made the code easier to manage.

The final dataset used for model deployment contained **22 core features,** as shown in Table 24**:**

*Table 24: Final Features Used in the Deployed Model*

| ***Feature Name*** | ***Description*** |
|---|---|
| *date* | Timestamp of the observation (hourly resolution) |
| *solar_gen_day_ahead* | Forecasted solar generation for the Portuguese zone [MW] (OMIE) |
| *wind_onshore_gen_day_ahead* | Forecasted onshore wind generation for the Portuguese zone [MW] (OMIE) |
| *biomass_actual* | Actual aggregated generation from biomass sources [MW] (REN Data Hub) |
| *fossil_gas_actual* | Actual aggregated generation from fossil gas [MW] (REN Data Hub) |
| *hydro_pumped_actual* | Actual generation from pumped-storage hydro plants [MW] (REN Data Hub) |
| *hydro_pumped_consumption* | Consumption of electricity for pumping in hydro storage [MW] (REN Data Hub) |
| *hydro_runofriver_actual* | Actual generation from run-of-river hydro plants [MW] (REN Data Hub) |
| *other_actual* | Actual aggregated generation from other sources [MW] (REN Data Hub) |
| *solar_actual* | Actual aggregated solar generation [MW] (REN Data Hub) |
| *wind_onshore_actual* | Actual aggregated onshore wind generation [MW] (REN Data Hub) |
| *total_load_actual* | Actual total electricity load in Portugal [MW] (REN Data Hub) |
| *wind_speed_kmh* | Wind speed in km/h from reanalysis data (Copernicus CDS) |
| *solar_radiation* | Solar radiation (W/m²) from reanalysis data (Copernicus CDS) |
| *precipitation_mm* | Precipitation in millimeters (Copernicus CDS) |
| *lisbon_temp_c* | Temperature in °C measured in Lisbon (Copernicus CDS) |
| *portugal_marginal_price* | Marginal price of electricity in Portugal (€/MWh) (OMIE Market Data) |
| *daily_market_energy* | Total energy traded in the daily electricity market (MWh) (OMIE) |
| *total_energy_acquisition* | Total matched energy acquired in the Portuguese system (MWh) (OMIE) |
| *total_energy_sale* | Total matched energy sold in the Portuguese system (MWh) (OMIE) |
| *export_spain_to_portugal* | Exported electricity from Spain to Portugal (MWh) (OMIE/ENTSO-E) |
| *iberian_market_energy* | Total energy in the Iberian market including bilateral contracts (MWh) (OMIE) |

In summary, this feature reduction process (as depicted in Table 25) helped to focus on a clean, meaningful, and balanced set of inputs for the final LightGBM model, improving both model performance and interpretability.

*Table 25:Features Removed in Final Dataset Pruning*

| ***Feature Removed*** | ***Reason (where known)*** |
|---|---|
| *Data e Dia da Semana, Hora, Day of the week* | Redundant after date parsing |
| *Year, Month, Day, Holiday, week or weekend* | Replaced by cyclic encodings in earlier models, later dropped |
| *Fossil Hard coal - Actual Aggregated [MW]* | Mostly zeros or no added value |
| *Angle* | Not significantly correlated with price |
| *direction_* (all wind direction dummies)* | Low feature importance, removed for model simplification |
| *Radiação Solar (W/m²)* | Renamed as `solar_radiation` |
| *Precipitação (mm) - Peso da Régua* | Renamed as `precipitation_mm` |
| *Lisboa temperature ° C* | Renamed as `lisbon_temp_c` |
| *Preços marginais sistema portugues* | Shifted to create `target` and lag features |
| *Energia Mercado IbÃ©rico incluindo lilaterais* | Renamed as `iberian_market_energy` |

Figure 14 presents the first ten rows of the final dataset used for electricity price forecasting. Each row corresponds to an hourly observation and includes the target variable (`Portugal's marginal price`) along with the engineered features derived from OMIE, REN DataHub, and Copernicus CDS sources. For the Final Model, Pipeline, and FastAPI Deployment, the dataset was restricted to `1 January 2024` to `20 March 2025`, aligning with the most relevant and up-to-date period for practical forecasting. The sample illustrates the structure of the dataset after preprocessing, feature engineering, and integration of market, load, renewable generation, and weather-related variables.

*Figure 14: First 10 rows of the final dataset*

| | date | solar_gen_day_ahead | wind_onshore_gen_day_ahead | biomass_actual | fossil_gas_actual | hydro_pumped_actual | hydro_pumped_consumption |
|---|---|---|---|---|---|---|---|
| 0 | 2023-12-31 23:59:59.984 | 0 | 349 | 324 | 218 | 0 | 1703 |
| 1 | 2024-01-01 00:59:59.984 | 0 | 356 | 325 | 215 | 0 | 2057 |
| 2 | 2024-01-01 01:59:59.984 | 0 | 377 | 323 | 215 | 0 | 2416 |
| 3 | 2024-01-01 02:59:59.984 | 0 | 384 | 325 | 215 | 0 | 2067 |
| 4 | 2024-01-01 03:59:59.984 | 0 | 377 | 304 | 211 | 0 | 1896 |
| 5 | 2024-01-01 04:59:59.984 | 0 | 374 | 322 | 214 | 1 | 1273 |
| 6 | 2024-01-01 05:59:59.984 | 0 | 363 | 318 | 308 | 453 | 16 |
| 7 | 2024-01-01 06:59:59.984 | 183 | 346 | 318 | 521 | 956 | 6 |
| 8 | 2024-01-01 07:59:59.984 | 655 | 342 | 322 | 535 | 900 | 2 |
| 9 | 2024-01-01 08:59:59.984 | 1104 | 346 | 320 | 524 | 865 | 2 |

10 rows × 22 columns

| hydro_runofriver_actual | other_actual | solar_actual | ... | wind_speed_kmh | solar_radiation | precipitation_mm | lisbon_temp_c | portugal_marginal_price |
|---|---|---|---|---|---|---|---|---|
| 1723 | 13 | 0 | ... | 4.883598 | 0.000000 | 0.000000 | 11.49 | 71.96 |
| 1187 | 13 | 0 | ... | 5.322486 | 0.000000 | 0.000000 | 11.69 | 63.33 |
| 829 | 13 | 0 | ... | 4.169583 | 0.000000 | 0.000000 | 11.25 | 50.09 |
| 1021 | 13 | 0 | ... | 4.401760 | 0.000000 | 0.000000 | 10.97 | 47.50 |
| 661 | 12 | 0 | ... | 2.795044 | 0.000000 | 0.000000 | 10.57 | 43.50 |
| 646 | 13 | 0 | ... | 4.136323 | 0.000000 | 0.000000 | 10.35 | 42.50 |
| 886 | 13 | 0 | ... | 4.104097 | 0.000000 | 0.000011 | 10.22 | 42.09 |
| 1290 | 13 | 2 | ... | 4.187094 | 0.000000 | 0.000021 | 10.08 | 42.50 |
| 1728 | 13 | 96 | ... | 5.182555 | 0.372252 | 0.000023 | 10.00 | 42.59 |
| 1752 | 13 | 337 | ... | 8.742517 | 71.686058 | 0.000021 | 9.90 | 43.37 |

| daily_market_energy | total_energy_acquisition | total_energy_sale | export_spain_to_portugal | iberian_market_energy |
|---|---|---|---|---|
| 16574.3 | 5628.0 | 3915.2 | 1712.8 | 27462.7 |
| 15865.9 | 5195.1 | 4069.0 | 1126.1 | 26156.5 |
| 15096.7 | 5066.0 | 3373.3 | 1692.7 | 25072.3 |
| 14183.9 | 4960.2 | 2851.8 | 2108.4 | 23682.0 |
| 13894.7 | 5059.6 | 2633.0 | 2426.6 | 23035.9 |
| 13812.4 | 4800.0 | 2427.5 | 2372.5 | 22707.4 |
| 13850.4 | 4655.5 | 2401.5 | 2254.0 | 22748.9 |
| 13839.1 | 4617.3 | 2578.0 | 2039.3 | 22862.1 |
| 14195.3 | 4679.5 | 2755.4 | 1924.1 | 23434.7 |
| 14608.5 | 4742.9 | 2763.9 | 1979.0 | 23963.9 |

## 6.2 Final Model Selection: Optuna-Tuned LightGBM on Split Data

To develop a robust and generalizable electricity price forecasting model, a final modeling approach was designed using the **LightGBM** algorithm (as shown in Appendix C), a highly efficient and accurate gradient boosting framework. Unlike earlier modeling attempts, this version integrated a more sophisticated preprocessing pipeline, an enriched feature set, and automated hyperparameter tuning via Optuna to achieve strong performance while minimizing overfitting.

The predictive target remained the "`portugal_marginal_price`", forecasted **168 hours (i.e., one week) into the future**. To support this forecast, a wide range of lagged and derived features were engineered from the original dataset. Lag-based features included historical values such as the price 167 and 166 hours before the target (`price_bef, price_bef1`), as well as `168-hour` lagged values "`solar_radiation`", "`total_load_actual`", and "`iberian_market_energy`". These temporal features provided context about past market conditions when making future predictions.

To further enhance the model's understanding of underlying temporal patterns, calendar features were extracted from the timestamp column. The month and day were transformed into sine and cosine encodings, enabling the model to learn smooth cyclical seasonality. Additional interaction features were created to represent the relationship between these temporal cycles and energy features, such as the interaction between "month_sin" and "solar_radiation_168". The feature set also included rolling window statistics—mean, minimum, maximum, and standard deviation—calculated over 24, 72, and 168-hour windows for variables like the "portugal_marginal_price", "solar_radiation", and "total_load_actual". These features helped the model to identify trends, fluctuations, and volatility in energy consumption and generation.

Several other features were computed based on ratios, differences, and momentum indicators. For example, the ratio of "solar_radiation" to "total_load_actual", the difference between price lags (price_bef_diff), normalized solar intensity (solar_rad_168_norm), and composite indicators such as price_bef × total_load_actual_168 were included. Altogether, these features captured both short-term changes and long-term dynamics within the electricity market.

Once the full feature set was prepared, the data was split chronologically to preserve the temporal structure of the forecasting task. The training set comprised data from the entire year 2024. The validation set included the period from January 1 to February 14, 2025, while the test set included data from February 15 to March 20, 2025. A RobustScaler was applied to the feature sets, with the scaler being fitted only on the training data and then applied to the validation and test sets to ensure that data leakage did not occur during scaling.

To optimize the model's performance, the Optuna framework was used for automated hyperparameter tuning. The objective was to minimize the MAE on the validation set using early stopping to prevent overfitting. Key hyperparameters such as learning rate, number of leaves, feature and bagging fractions, and minimum child samples were tuned across several trials. Each trial trained a LightGBM model using the training set and monitored performance on the validation set. Early stopping was employed with a patience of 50 rounds to halt training when further improvements were no longer observed.

Several optimization runs were conducted with increasing numbers of Optuna trials: 50, 100, 150, and 200. While the models from 50 and 100 trials showed reasonable performance, further increasing the search space to 150 trials led to a significant reduction in training error without sacrificing generalization. Interestingly, the 200-trial model produced nearly identical results to the 150-trial model, suggesting that convergence had been achieved. Therefore, as depicted in Table 26, the model with 150 trials was selected as the final

configuration, offering the best balance between computational efficiency and predictive accuracy.

The final LightGBM model was trained using the best hyperparameters discovered by Optuna, and its performance was evaluated across all three data segments. The results were as follows: the training MAE was 0.0533, the validation MAE was 1.0497, and the test MAE was 0.3277 (Table 26). These results indicated that the model was able to fit the training data well while also generalizing effectively to unseen future data, especially given the challenging one-week-ahead forecasting horizon.

Although not deployed directly, this version of the model played a critical role in final model selection. Its strong generalization performance provided evidence that the selected hyperparameters and feature engineering approach were robust. Feature importance analysis was also performed. The top contributors included lagged price values ("`price_bef_169`", "`price_bef_170`"), "`solar_radiation`", "`total_load_actual`", and composite interaction terms, confirming their critical role in the forecasting process. The trained model and its corresponding scaler were preserved using `.pkl` serialization to support reproducibility, comparison, and potential future deployment. This ensured that the entire preprocessing and modeling pipeline could be reapplied to new data without retraining from scratch.

In summary, this Optuna-tuned LightGBM model, trained and validated on chronologically split data, served as the core benchmark for the project. It not only delivered strong forecasting accuracy across all time segments but also informed the development of the final full-dataset model used for deployment.

*Table 26: Final Model MAE: Optuna-Tuned LightGBM on Split Data- Model Performance Comparison Across Trial Counts*

| *Trial Count* | *Train MAE* | *Validation MAE* | *Test MAE* |
|---|---|---|---|
| *50 trials* | 0.1404 | 1.0584 | 0.3392 |
| *100 trials* | 0.0851 | 1.0549 | 0.3272 |
| ***150 trials*** | **0.0533** | **1.0497** | **0.3277** |
| *200 trials* | 0.0533 | 1.0497 | 0.3277 |

In addition to reporting MAE scores, prediction vs actual plots were generated for both the validation and test sets. These visualizations confirmed that the model captured the general direction and level of electricity prices, although some short-term volatility remained challenging to predict precisely. The visual inspection supported the quantitative findings and demonstrated the model's consistency across unseen data.

*Figure 15: Actual Vs Prediction Prices (Validation Set)*

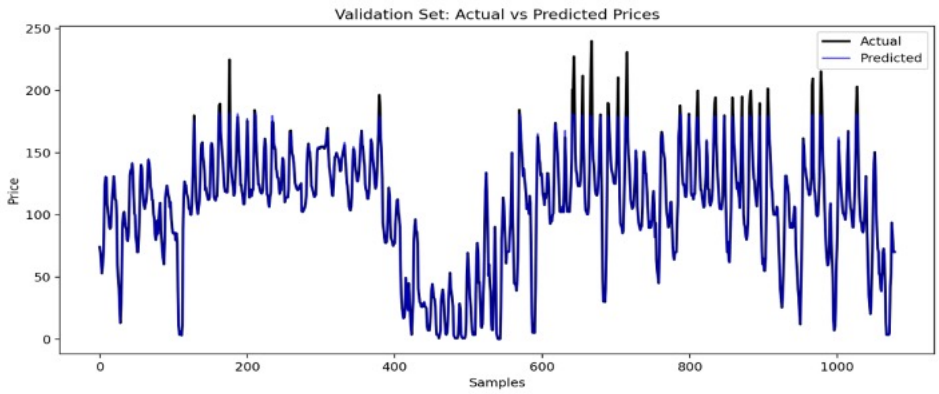



Figure 15 illustrates the model's performance on the "validation set," which is used during model development to tune hyperparameters and prevent overfitting. Similar to the test set, the blue line (predicted) generally tracks the black line (actual)**.** This consistency between validation and test set performance is a good sign, implying that the model is not overfitting to the training data and maintains its predictive ability across different subsets of the data. Again, some discrepancies between actual and predicted values are visible, indicating that the model isn't perfect but captures the general trends.

*Figure 16: Actual vs. Predicted Prices (Test Set)*

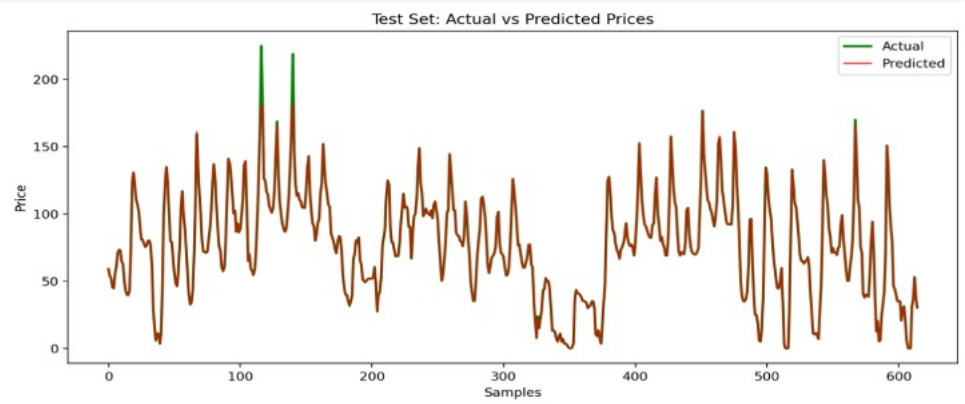



Figure 16 shows how well the model predicts prices on new, unseen data (the "test set"). The green line (predicted) largely follows the red line (actual), indicating that the model has learned the underlying patterns and can generalize reasonably well to new data. However, there are points where the predicted prices miss the actual peaks and valleys, suggesting room for improvement in prediction accuracy for extreme values.

### 6.2.1 Comparison of Final Model Selection with Sixth Modelling Attempt: Fast LightGBM on Split Data (Fixed Parameters) and Improvements:

In Chapter 5, the Six Modeling Attempt was introduced, a fast LightGBM model trained on split data using fixed hyperparameters. While it offered quicker training, its performance revealed overfitting—achieving very low error on the training set but significantly higher errors on the validation and test sets. In contrast, the Final Model Selection: Optuna-Tuned LightGBM on Split Data showed improved generalization and predictive accuracy, particularly on unseen data. This model leveraged Optuna for automated hyperparameter tuning and incorporated a richer feature set, including lagged variables, rolling statistics, interaction terms, and trend-based ratios. As shown in Table 27, despite slightly higher training error, the Final Model Selection: Optuna-Tuned LightGBM on Split Data outperformed the simpler approach on validation and test data, making it the more reliable choice for deployment.

*Table 27: Comparison of Final model with Fifth Modelling Attempt*

| ***Model*** | ***Train MAE*** | ***Val MAE*** | ***Test MAE*** |
|---|---|---|---|
| ***Final Model (Optuna with 150 Trails)*** | **0.0533** | **1.0497** | **0.3277** |
| *Sixth Modeling Attempt (Fixed Params)* | 0.0515 | 1.4949 | 0.4282 |

## 6.3 Final Model Trained on Full Dataset (No Split)

### 6.3.1 Model overview

Following the successful development and tuning of the Optuna-optimized LightGBM model on the split dataset, a final version of the model (as shown in Appendix D) was trained on the entire available dataset without separating it into training, validation, and test partitions. The purpose of this step was to fully utilize all available historical data and maximize the model's exposure to patterns that may improve its predictive performance before deployment. This approach is appropriate because the model's generalization capabilities had already been validated during the split-based experiments. By training on the full dataset, the model can potentially learn subtler temporal trends and achieve even lower error rates.

### 6.3.2 Data Loading and Preparation

The script begins by importing essential Python libraries such as Pandas, NumPy, LightGBM, and Scikit-learn, which are widely used for data processing, machine learning, and evaluation. The dataset is loaded from an Excel file containing hourly electricity market variables for 2024 and 2025. This dataset includes information such as `Portugal's marginal price`, `solar radiation`, `total load`, and `Iberian market energy`. A

copy of the original dataset is created to preserve the raw data, and the script prints column names and sample rows to verify successful loading.

### 6.3.3 Feature Engineering: Creating the Target and Future Variables

The feature engineering pipeline applied in this phase was identical to the one used in the split-based version. To predict electricity prices **seven days ahead**, the target variable (`target`) is generated by shifting the current price (`portugal_marginal_price`) **168 hours forward** (24 hours × 7 days = 168). This means that each record's target corresponds to the price exactly one week after that timestamp. Similarly, lagged price features (`price_bef`, `price_bef1`) and future-shifted weather and market variables (e.g., `solar_radiation_168`, `iberian_market_energy_168`, `total_load_actual_168`) are created. These features help the model learn temporal patterns and relationships between current and future values. The original price column is then dropped to avoid data leakage since the shifted versions now represent the prediction target and relevant lagged inputs.

### 6.3.4 Calendar and Seasonal Features

Calendar-based variables are extracted from the `date` column, including `year, month`, and `day`, to capture long-term seasonality. To represent cyclical time patterns more effectively, sine and cosine transformations are applied to month and day values (e.g., `month_sin`, `month_cos`). This approach ensures that December and January are treated as close points in the yearly cycle — something linear features cannot represent well.

### 6.3.5 Interaction Features

Several interaction features are created to model the combined effects of time and energy-related variables. For example, multiplying `month_sin` by `solar_radiation_168` allows the model to understand how the impact of solar radiation varies seasonally. Other interactions combine load, radiation, and market energy to improve the model's ability to detect complex nonlinear relationships between factors influencing electricity prices.

### 6.3.6 Lag and Rolling Window Features

Lag features such as `price_bef_169` and `price_bef_170` represent slightly older price values, giving the model additional historical context. Rolling window features are then calculated using a 24-hour window, which captures daily averages, minimums, and maximums of both price and solar radiation. These features summarize short-term fluctuations, helping the model understand local trends or volatility.

Later, rolling statistics with longer windows (24, 72, and 168 hours) are added for load and price. These multi-scale averages and standard deviations help capture different temporal dynamics — daily, three-day, and weekly patterns.

### 6.3.7 Ratio, Difference, and Trend Features

To detect proportional changes and price evolution, new features such as `solar_rad_to_load_168` (ratio between solar radiation and load), `price_trend` (difference between two recent prices), and `price_bef_diff_pct` (percentage change in prices) are computed. These features allow the model to sense not only raw values but also rate of change and relative relationships**,** which are crucial for dynamic systems like energy markets.

### 6.3.8 Data Cleaning

Since rolling and shift operations naturally create missing values at the start or end of the dataset, all `NaN` rows are dropped to ensure consistent input data. The script also counts how many zero values exist in solar-related columns, identifying whether solar data may be sparse or unreliable in certain periods.

### 6.3.9 Feature and Target Separation

After preparing all engineered variables, the feature matrix `X` is defined by removing non-predictive columns such as `date`, `year`, `month`, and `day`, while the target variable `y` is set as the shifted price (`target`). Before modeling, the input data is scaled using `RobustScaler`, which minimizes the effect of outliers compared to standard scaling. This helps LightGBM process all features on a similar scale and improves model stability.

### 6.3.10 Model Training Using Optimized Parameters

The model uses LightGBM— a fast and efficient gradient boosting algorithm ideal for large datasets and structured tabular data. The model was trained using the same best hyperparameters identified earlier through Optuna tuning over `150–200 trials`. These parameters had already been shown to deliver optimal results during split-based testing, so reusing them ensured consistency while saving computational time. The LightGBM model was trained on the full scaled dataset with the following configuration and carefully tuned subsampling and regularization settings:

```
best_params = {
    'n_estimators': 1000,
    'learning_rate': 0.0462,
    'num_leaves': 221,
    'feature_fraction': 0.9794,
    'bagging_fraction': 0.8991,
    'bagging_freq': 4,
    'min_child_samples': 54,
    'random_state': 42
}
```

### 6.3.11 Model Saving and Evaluation

The model's performance is evaluated on the full dataset using Mean Absolute Error (MAE), which measures the average difference between actual and predicted prices in euros. A lower MAE indicates higher predictive accuracy. The final model achieved a MAE of `0.0822`, which was the lowest MAE obtained across all experiments in the project. This confirms that, after thorough validation, retraining on the full dataset provided additional gains in performance, making it the most suitable candidate for deployment.

In addition to training and evaluation, the model and scaler were saved in `.pkl` format using Joblib**,** to facilitate deployment and reproducibility. The LightGBM model was stored as `"final_lightgbm_full_dataset.pkl"`, and the corresponding scaler was saved as `"final_scaler.pkl"`. This allows future predictions to be generated consistently and ensures that the entire preprocessing and inference pipeline remains aligned with the training procedure.

### 6.3.12 Generating and Saving Predictions

To align each prediction with its true future timestamp, a new column `predicted_date` is created by adding 168 hours (7 days) to each original input date. A separate DataFrame is generated containing the `input date`, `predicted date`, `actual price`, and `predicted price`. This file (`actual_vs_predicted_7days_ahead.csv`) is saved for visualization, analysis, and validation. The script also prints useful information about how many rows were used, along with the range of input and predicted dates, giving a clear overview of the model's training and temporal coverage.

After saving the predictions, the script displays a summary of the model's performance and data coverage. The following output confirms that the model was successfully trained and that predictions were generated for all available records:

```
Model trained on full dataset
Total rows used for training and prediction: 9785
Number of predicted values: 9785
First input date: 2024-01-07 22:59:59.984000
Last input date: 2025-03-12 22:59:59.982000
First predicted date: 2024-01-14 22:59:59.984000
Last predicted date: 2025-03-19 22:59:59.982000
```

### 6.3.13 Feature Importance Analysis

To support model interpretability, feature importance scores were also extracted and analyzed. Figure 17 illustrate the most important features influencing the model's electricity price predictions. Historical price data, such as `price_bef_169` and `price_bef_170`, were by far the most influential, followed by features capturing price trends, load behavior, and weather conditions (e.g., `wind_speed_kmh`, `total_energy_sale`, `total_load_actual`).

Other factors such as "load_change" or specific energy sources like "hydro_pumped_actual", had a much smaller impact on the model's predictions. The bar chart provides a clear visual overview of the top 20 features, supporting interpretability and highlighting the central factors driving the model's decisions.

*Figure 17: Top 20 Important features in electricity price prediction*

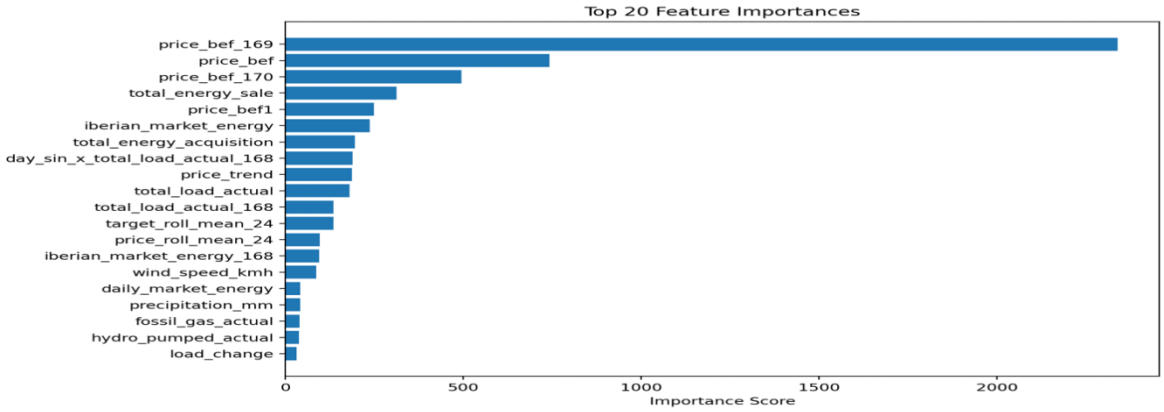


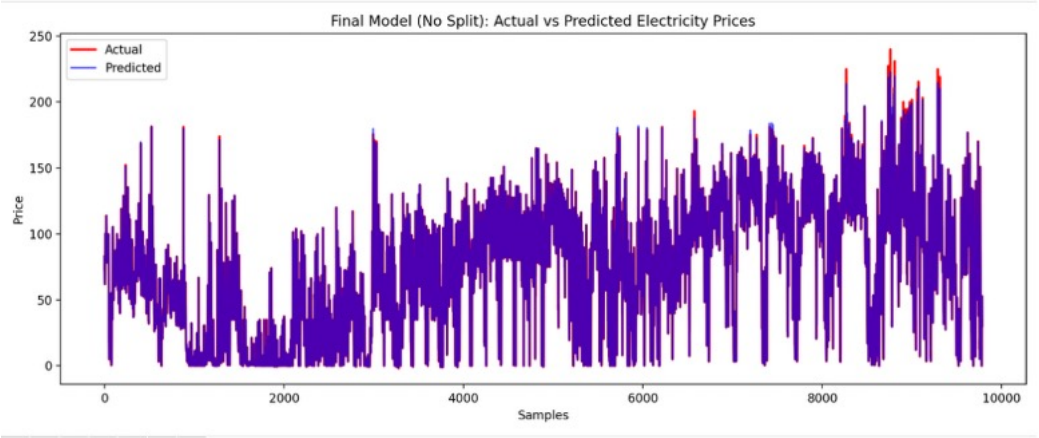

*Table 28: Top 10 Most Important Features in the Final Model*

| *Rank* | *Feature* | *Importance* |
|---|---|---|
| *1* | price_bef_169 | 13,922 |
| *2* | price_bef_170 | 4,796 |
| *3* | price_trend | 4,443 |
| *4* | wind_speed_kmh | 4,233 |
| *5* | total_energy_sale | 4,187 |
| *6* | price_bef_x_total_load_actual_168 | 4,107 |
| *7* | total_energy_acquisition | 3,779 |
| *8* | iberian_market_energy | 3,702 |
| *9* | total_load_actual_168 | 3,541 |
| *10* | price_bef | 3,455 |

The feature importance analysis provided in Table 28 revealed that lagged electricity prices (e.g., "price_bef_169", "price_bef_170") were the most influential predictors, confirming the strong temporal dependency of electricity prices on past values. In addition, derived features such as price_trend and interaction terms (e.g., "price_bef_x_total_load_actual_168") captured meaningful relationships between price dynamics and demand. Market-related indicators ("total_energy_sale, total_energy_acquisition", "iberian_market_energy") and exogenous variables like "wind_speed_kmh" and "total_load_actual_168 further contributed significantly. Together, these results highlight that both historical price lags and market-demand interactions play a central role in shaping electricity price forecasts.

### 6.3.14 Visualization of Model Predictions

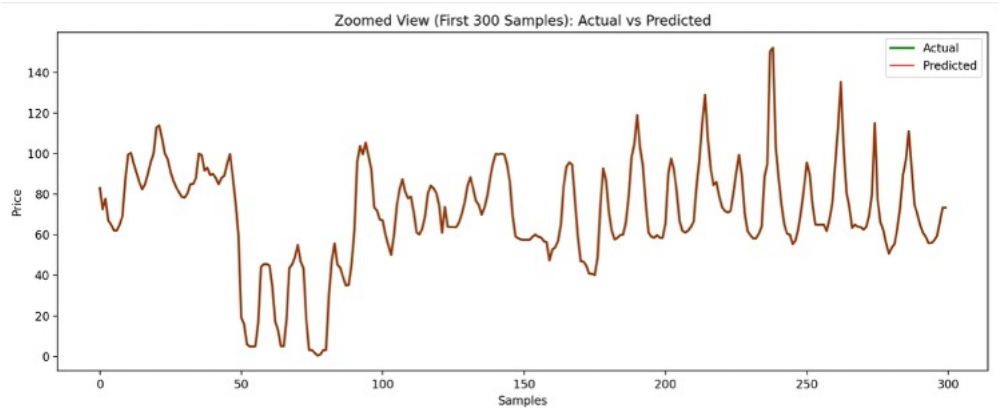
*Figure 18: Final model (No split): Actual Vs Predicted Prices*




To assess how well the model predicts electricity prices, two time-series plots were generated: a full view (Figure 18) and a zoomed-in section (Figure 19*)*.

Figure 18 displays the model's predicted electricity prices `(blue line)` against the actual historical prices `(black line)` over the entire dataset. It shows that the predicted prices generally follow the same overall trends, capturing the major increases and decreases observed in the actual prices.

*Figure 19: Actual Vs Predicted Prices Zoomed View (First 300 Samples)*




In Figure 19 a closer examination of the first 300 data points provides more detail. This view confirms that the predicted prices closely track the actual prices. The model successfully captures the short-term fluctuations, including peaks and troughs, demonstrating its ability to mirror the real-world price movements quite accurately. Both visualizations confirm that the model's predictions are highly consistent with the actual electricity prices, indicating strong predictive performance.

Altogether, this final model combines a rich set of engineered features with robust tuning and full-data training, resulting in the most accurate and deployment-ready forecasting solution developed in the project. Overall, the script builds a complete and explainable **7-**

day-ahead electricity price prediction model using engineered temporal, weather, and energy features.
Every transformation — from feature creation to scaling, model training, and evaluation — contributes to capturing the underlying temporal dynamics of the electricity market.
The trained model can later be used for real-time forecasts, provided that new incoming data follows the same feature structure and preprocessing steps.

*Table 29: Comparison: Final Model (Full Dataset) vs. Final Model with Split*

| *Criteria* | *Split-Based Final Model* | *Full Dataset Final Model* |
|---|---|---|
| ***Training Data*** | Train/Val/Test split (2024–Mar 2025) | Entire dataset (no split) |
| ***Optuna Tuning*** | Yes (100 trials) | Yes (same best parameters) |
| ***Feature Engineering*** | Advanced (lags, rolling, interactions, ratios) | Same |
| ***Model Type*** | LightGBM | LightGBM |
| ***MAE – Train (2024)*** | 0.0851 | – |
| ***MAE – Validation (Jan–Feb 14)*** | 1.0549 | – |
| ***MAE – Test (Feb 15–Mar 20)*** | 0.3272 | – |
| ***MAE – Full Dataset*** | – | **0.0822** |
| ***Use Case*** | Model evaluation, selection | Final training before deployment |

### 6.3.15 Results and Model Capabilities

The final LightGBM model achieved an **`MAE of 0.0824`**, representing an extremely low error across all hourly predictions. This demonstrates that the model can accurately reproduce the true price trends over a 7-day forecasting horizon when trained on the entire dataset.

The final LightGBM model was trained on the full dataset, using the previously validated Optuna-tuned hyperparameters. The dataset spans hourly records from `January 1, 2024,` to `May 31, 2025`, and the target variable was constructed by shifting the electricity price forward by `168 hours` (one week). After processing and removing rows with incomplete lag or rolling features, the input period used for training runs from `January 7, 2024,` to `March 12, 2025`, while the corresponding predicted (target) dates span `January 14, 2024,` to `March 19, 2025`. For example, an input row corresponding to `2024-01-07 22:59:59` is used to predict the price at `2024-01-14 22:59:59`.

In practical terms, for every hourly input, the model takes historical and lagged features such as prices, load, and weather conditions, and outputs the model's estimate of the electricity price exactly seven days later at the same hour. The generated `CSV` file, `actual_vs_predicted_7days_ahead.csv`, captures this clearly: it lists the input timestamp, the predicted timestamp (input + 168 hours), the actual price at that predicted timestamp, and the model's predicted price. Including the predicted date in the `CSV` removes

any ambiguity about which future hour is being forecasted. A sample of this `CSV` is shown in Figure 20, demonstrating how each input corresponds to a specific predicted hour and the associated actual and predicted prices.

*Figure 20: Model Actual vs Predicted 7days ahead CSV sample*

| A | B | C | D |
|---|---|---|---|
| input_date | predicted_date | actual_price_7d_ahead | predicted_price_7d_ahead |
| 1/7/2024 11:00:00 PM | 1/14/2024 11:00:00 PM | 82.97 | 82.97710067422206 |
| 1/8/2024 12:00:00 AM | 1/15/2024 12:00:00 AM | 72.52 | 72.52256234137366 |
| 1/8/2024 1:00:00 AM | 1/15/2024 1:00:00 AM | 77.7 | 77.67691734398673 |
| 1/8/2024 2:00:00 AM | 1/15/2024 2:00:00 AM | 67 | 67.05101852111407 |
| 1/8/2024 3:00:00 AM | 1/15/2024 3:00:00 AM | 64.99 | 64.96317942932244 |
| 1/8/2024 4:00:00 AM | 1/15/2024 4:00:00 AM | 62.04 | 62.04317767162798 |
| 1/8/2024 5:00:00 AM | 1/15/2024 5:00:00 AM | 62.1 | 62.080151522148064 |
| 1/8/2024 6:00:00 AM | 1/15/2024 6:00:00 AM | 65.1 | 65.09756242698647 |
| 1/8/2024 7:00:00 AM | 1/15/2024 7:00:00 AM | 69.13 | 69.16063011994393 |
| 1/8/2024 8:00:00 AM | 1/15/2024 8:00:00 AM | 86.74 | 86.7066453420787 |
| 1/8/2024 9:00:00 AM | 1/15/2024 9:00:00 AM | 99.52 | 99.54615329061217 |
| 1/8/2024 10:00:00 AM | 1/15/2024 10:00:00 AM | 100.33 | 100.31179264350614 |
| 1/8/2024 11:00:00 AM | 1/15/2024 11:00:00 AM | 94.92 | 94.8647356792488 |
| 1/8/2024 12:00:00 PM | 1/15/2024 12:00:00 PM | 90.3 | 90.22116652181671 |
| 1/8/2024 1:00:00 PM | 1/15/2024 1:00:00 PM | 85.7 | 85.62081745309244 |
| 1/8/2024 2:00:00 PM | 1/15/2024 2:00:00 PM | 82.41 | 82.43026654459987 |
| 1/8/2024 3:00:00 PM | 1/15/2024 3:00:00 PM | 84.99 | 84.92059560374504 |
| 1/8/2024 4:00:00 PM | 1/15/2024 4:00:00 PM | 90.1 | 90.1662337975833 |
| 1/8/2024 5:00:00 PM | 1/15/2024 5:00:00 PM | 95.82 | 95.76003583324862 |
| 1/8/2024 6:00:00 PM | 1/15/2024 6:00:00 PM | 100 | 100.13486318592616 |
| 1/8/2024 7:00:00 PM | 1/15/2024 7:00:00 PM | 112.85 | 112.97760758062537 |
| 1/8/2024 8:00:00 PM | 1/15/2024 8:00:00 PM | 113.9 | 113.73703922159298 |
| 1/8/2024 9:00:00 PM | 1/15/2024 9:00:00 PM | 107.58 | 107.35124022073525 |
| 1/8/2024 10:00:00 PM | 1/15/2024 10:00:00 PM | 100 | 99.8582930797843 |
| 1/8/2024 11:00:00 PM | 1/15/2024 11:00:00 PM | 97.21 | 97.23920331220823 |
| 1/9/2024 12:00:00 AM | 1/16/2024 12:00:00 AM | 91 | 90.97565195596846 |

**Note:** The model was trained and validated using historical data (with the target generated through a `168-hour` temporal shift). For future or live forecasts, the latest available input data must be pre-processed using the same feature engineering and scaling steps. The model will then generate 7-day-ahead predictions relative to the most recent timestamp in the input data.

The model was trained and tested using historical data, mainly for evaluation. In theory, the data is up to May 31, 2025. It could use the last week of that data to forecast June 1–7. However, in this study, the predictions were made within the available dataset to validate model accuracy, not to extend forecasts beyond it.

For future deployment, the same model can predict June prices or later — we just need to provide the most recent input data, and the model will forecast the next 7 days automatically.

### Interpretation and Final Considerations

As shown in Table 29 both final models use the same feature engineering pipeline and Optuna-tuned hyperparameters. The difference lies in training data: the first model used a train/validation/test split to evaluate generalization, while the second was retrained on the full dataset to leverage all historical data. The split-based evaluation ensured model robustness, and the full-dataset model was saved as the final version ready for deployment.

This process — validating a model using a proper split and subsequently retraining it on the full dataset — is widely recognized as a best practice in machine learning workflows, particularly when preparing a model for real-world application (Data Science Stack Exchange, 2025; Stack Overflow, 2025).

## 6.4 Deployment Pipeline and Price Prediction Interface

After finalizing the LightGBM model using the full dataset without any data split, the subsequent phase involved operationalizing the model through the development of an automated preprocessing pipeline (as shown in Appendix E). This pipeline was designed to replicate all feature engineering and transformation steps consistently, ensuring that any future data input undergoes identical processing to that of the original training data. The integration of this pipeline enables the model to be deployed effectively for practical forecasting applications, including real-time price prediction and user-facing applications such as price comparison and potential cost savings.

The pipeline was constructed using the `Pipeline` module from the `scikit-learn` library. It is composed of three primary components: `DataFrameSelector`, `FeatureEngineer`, and `RobustScaler`. Each component fulfils a specific role in ensuring the reproducibility, reliability, and scalability of the electricity price forecasting system.

### 6.4.1 DataFrameSelector

The `DataFrameSelector` class is a custom transformer responsible for extracting the relevant input variables required for feature engineering and modeling. Since the raw datasets may originate from various sources and contain either Portuguese or English column names, this component first harmonizes column names by mapping Portuguese field names to their English equivalents.

**Functions and Responsibilities:**

- Ensures consistent column naming conventions across datasets.
- Selects only the required input variables used in the modeling process.
- Validates data integrity by checking for missing or misnamed columns and raising errors if discrepancies are detected.

This approach provides flexibility in integrating heterogeneous data sources, guaranteeing that subsequent pipeline stages always receive properly formatted and validated inputs.

### 6.4.2 FeatureEngineer

The FeatureEngineer component encapsulates the logic for generating all predictive features used in the final LightGBM model. Its primary objective is to reconstruct the exact

transformations applied during model training, thereby maintaining consistency between the training and inference phases. The following key features were created:

- **Target and Lag Features**: The model predicts the `portugal_marginal_price` 168 hours (`1 week`) ahead. Lag Features such as "`price_bef(167 hours ahead)`" and "`price_bef1 (166 hours ahead)`" help the model to capture short-term temporal dependencies and recent price movements.
- **Calendar and Seasonal Features**: Extracts the `year`, `month`, and `day`, and converts them into `sine/cosine` form to reflect cyclical seasonality (e.g., monthly seasonality). This cyclical encoding technique preserves the periodic nature of time-related variables, allowing the model to learn seasonal patterns in electricity prices.
- **Interaction Features**: Captures interactions between time (e.g., `month_sin`) and energy variables like "`solar_radiation`" or "`daily_market_energy`". These features help the model identify complex nonlinear relationships between solar activity, market conditions, and seasonal trends.
- **Rolling Statistics**: Calculates rolling averages, minimums, and maximums for selected variables like "`portugal_marginal_price`" and "`solar_radiation`" over time windows (24, 72, 168 hours) to help the model understand medium- and long-term fluctuations in the energy market.
- **Ratio and Trend Features**: Features such as the ratio of solar radiation to total load (`solar_rad_to_load_168`) and percentage differences between consecutive lag prices (`price_bef_diff_pct`) provide additional insights into the dynamics of renewable generation and demand changes.
- **Lag Features**: Includes additional lagged prices like "`price_bef_169`" and "`price_bef_170`" to improve model memory to capture persistence patterns in price evolution.

After all transformations are applied, rows containing missing values (arising from lag or rolling operations) are removed to ensure the integrity of the training and prediction processes. Additionally, non-predictive columns such as `date`, `year`, `month`, `day`, and `portugal_marginal_price` are dropped to prevent information leakage.

### 6.4.3 RobustScaler

Following feature engineering, the `RobustScaler` is applied to normalize all numeric features. Unlike traditional standardization methods, the `RobustScaler` utilizes the interquartile range (IQR), which reduces the influence of extreme outliers commonly found in electricity market data. This scaling step enhances model stability and ensures consistent feature magnitudes during prediction.

### 6.4.4 Zero Value Analysis in Solar Features:

During transformation, the pipeline also prints diagnostics showing how often zero values occur in solar-related features. This is useful for identifying if certain variables might be unreliable or sparse.

For example:

- "solar_radiation_168" had 48.17% zero values.
- "solar_radiation_168_roll_min_24" had 100% zero values.
- Other derived features like "price_bef1_x_solar_radiation_168" also had ~48% zeros.

This confirmed earlier insights during modeling, where solar features were often zero—particularly during nighttime or cloudy periods—and were therefore considered for potential removal depending on their importance.

### 6.4.5 Model Integration and Prediction

After integrating the trained LightGBM model with the finalized preprocessing pipeline, the system was executed to generate hourly electricity price forecasts seven days (168 hours) ahead of each input timestamp.
This marks the final stage of the modeling process, where the trained model translates past market behaviour into forward-looking price estimates.

**Prediction Results and Output Structure**

The model produced a structured output that aligns each input date (the date and time of known data) with its corresponding predicted date (exactly seven days later) and the associated predicted price. To ensure a comprehensive evaluation, the actual observed prices for the predicted period were also included for comparison in a single file. A portion of the resulting dataset is presented below in Figure 21:

*Figure 21: Pipeline_classes Actual vs Predicted 7days ahead CSV samples*

| | A | B | C | D |
|---|---|---|---|---|
| | input_date | predicted_date | actual_price_7d_ahead | predicted_price_7d_ahead |
| | 1/7/2024 11:00:00 PM | 1/14/2024 11:00:00 PM | 82.97 | 82.97710067422206 |
| | 1/8/2024 12:00:00 AM | 1/15/2024 12:00:00 AM | 72.52 | 72.52256234137366 |
| | 1/8/2024 1:00:00 AM | 1/15/2024 1:00:00 AM | 77.7 | 77.67691734398673 |
| | 1/8/2024 2:00:00 AM | 1/15/2024 2:00:00 AM | 67 | 67.05101852111407 |
| | 1/8/2024 3:00:00 AM | 1/15/2024 3:00:00 AM | 64.99 | 64.96317942932244 |
| | 1/8/2024 4:00:00 AM | 1/15/2024 4:00:00 AM | 62.04 | 62.04317767162798 |
| | 1/8/2024 5:00:00 AM | 1/15/2024 5:00:00 AM | 62.1 | 62.080151522148064 |
| | 1/8/2024 6:00:00 AM | 1/15/2024 6:00:00 AM | 65.1 | 65.09756242698647 |
| | 1/8/2024 7:00:00 AM | 1/15/2024 7:00:00 AM | 69.13 | 69.16063011994393 |
| | 1/8/2024 8:00:00 AM | 1/15/2024 8:00:00 AM | 86.74 | 86.7066453420787 |
| | 1/8/2024 9:00:00 AM | 1/15/2024 9:00:00 AM | 99.52 | 99.54615329061217 |
| | 1/8/2024 10:00:00 AM | 1/15/2024 10:00:00 AM | 100.33 | 100.31179264350614 |
| | 1/8/2024 11:00:00 AM | 1/15/2024 11:00:00 AM | 94.92 | 94.8647356792488 |
| | 1/8/2024 12:00:00 PM | 1/15/2024 12:00:00 PM | 90.3 | 90.22116652181671 |
| | 1/8/2024 1:00:00 PM | 1/15/2024 1:00:00 PM | 85.7 | 85.62081745309244 |
| | 1/8/2024 2:00:00 PM | 1/15/2024 2:00:00 PM | 82.41 | 82.43026654459987 |
| | 1/8/2024 3:00:00 PM | 1/15/2024 3:00:00 PM | 84.99 | 84.92059560374504 |
| | 1/8/2024 4:00:00 PM | 1/15/2024 4:00:00 PM | 90.1 | 90.1662337975833 |
| | 1/8/2024 5:00:00 PM | 1/15/2024 5:00:00 PM | 95.82 | 95.76003583324862 |
| | 1/8/2024 6:00:00 PM | 1/15/2024 6:00:00 PM | 100 | 100.13486318592616 |
| | 1/8/2024 7:00:00 PM | 1/15/2024 7:00:00 PM | 112.85 | 112.97760758062537 |
| | 1/8/2024 8:00:00 PM | 1/15/2024 8:00:00 PM | 113.9 | 113.73703922159298 |
| | 1/8/2024 9:00:00 PM | 1/15/2024 9:00:00 PM | 107.58 | 107.35124022073525 |
| | 1/8/2024 10:00:00 PM | 1/15/2024 10:00:00 PM | 100 | 99.8582930797843 |
| | 1/8/2024 11:00:00 PM | 1/15/2024 11:00:00 PM | 97.21 | 97.23920331220823 |
| | 1/9/2024 12:00:00 AM | 1/16/2024 12:00:00 AM | 91 | 90.97565195596846 |
| | 1/9/2024 1:00:00 AM | 1/16/2024 1:00:00 AM | 86.59 | 86.57880114782117 |

**Interpretation of Results**

Figure 21 clearly illustrates how the model aligns each hourly input observation with a forecasted value seven days later. Key observations include:

- **Temporal consistency:** Each forecast is precisely mapped to the same hour one week ahead, maintaining perfect chronological alignment (e.g., `1/7/2024 23:00` → `1/14/2024 23:00`).
- **High prediction accuracy:** The predicted prices are almost identical to the actual observed market values, with minimal deviation. For instance:
  - At `1/7/2024 23:00`, the actual price was `€82.97/MWh`, and the model predicted `€82.98/MWh` (difference: `€0.01/MWh`).
  - At `1/8/2024 10:00`, the actual price was `€100.33/MWh`, and the model predicted `€100.31/MWh` (difference: `€0.02/MWh`).
    These very small differences indicate that the model successfully captured the short-term dynamics of electricity price variation.
- **Stable trend reproduction:** Across consecutive hours, the model's predictions follow the same directional movement as the real data — rising and falling consistently with market trends. This suggests effective learning of daily demand patterns and supply fluctuations.
- **Quantitative evaluation:** When assessed over the full test period, the Mean Absolute Error (MAE) remained within a very narrow range, confirming both stability and predictive reliability.

The resulting predictions were exported into a `CSV` file for visualization and evaluation. These forecasts can then be compared with real market data to assess model accuracy, typically using metrics such as Mean Absolute Error (MAE).

Additionally, predicted and actual prices can be visualized over time to observe how closely the model captures market fluctuations, enabling deeper interpretability and validation of forecasting performance.

Overall, the model serves as a predictive tool for estimating hourly electricity prices seven days ahead. It processes historical and external factors—such as load, weather, and renewable generation—to forecast future market behavior. Its purpose is to enable price forecasting, trend analysis, and strategic planning in energy management systems, helping both operators and researchers anticipate short-term price fluctuations.

### 6.4.6 Calculating Monthly Savings for End Users

To make the model results more actionable and user-friendly, an additional step was added to estimate monthly cost savings for consumers. This component is beneficial if the model is integrated into an energy consumption dashboard or mobile app.

**Explanation of the Code Logic:**

```
def calcular_poupanca(consumo_mensal_kwh, preco_atual_pago):
    preco_previsto_mwh = predictions.mean()
    preco_previsto_kwh = preco_previsto_mwh / 1000
    poupanca_media_kwh = preco_atual_pago - preco_previsto_kwh
    poupanca_total_mensal = poupanca_media_kwh * consumo_mensal_kwh
    return preco_previsto_kwh, poupanca_media_kwh, poupanca_total_mensal
```

- **Inputs**:
  - `"consumo_mensal_kwh"`: Monthly electricity usage of a user (e.g., `1000 kWh`).
  - `"preco_atual_pago"`: The user's current electricity rate (e.g., `€0.12/kWh`).
- **Description**:
  - Computes the average predicted price using the model output.
  - Converts `€/MWh` to `€/kWh`.
  - Calculates the difference between the user's current price and the predicted market price.
  - Multiply this difference by monthly consumption to estimate total potential savings.

**Results obtained and interpretation:**

Results show that for a user consuming `1000 kWh`/month and paying `€0.12/kWh`, the model estimated:

- Predicted price: `€0.0721/kWh`
- Savings per kWh: `€0.0479`
- Total monthly savings: `€47.87`

This indicates the user could save almost `€48` per month if switching to a plan aligned with the predicted market price. While it does not predict future consumption, it uses the model's forecast of average electricity price to simulate potential savings based on user input.

This makes it a forward-looking tool, even though it's based on historical prediction modeling.

### 6.4.7 Summary

The deployment pipeline successfully automates data preprocessing, ensures consistent feature engineering, and integrates seamlessly with the trained LightGBM model. It not only provides reliable electricity price forecasts but also supports consumer-facing applications such as savings estimation. The design is robust, modular, and scalable, making it well-suited for real-world integration in energy management tools or digital platforms.

## 6.5 Prediction Pipeline and Supporting Utility Scripts

To ensure clear organization, reusability, and easier maintenance of the prediction system, the implementation was divided into several Python scripts based on functionality.
Following the final model and `Pipeline_classes.py`, two additional scripts support the automated prediction and deployment stages: `pipeline.py` and `utils.py`, both of which support the execution of the API endpoints and the final prediction results.

### 6.5.1 `utils.py`: Data and Model Utilities

The `utils.py` script (as shown in Appendix F) contains helper functions for loading the required data and model. Separating these responsibilities into a utility module improves code clarity and makes it easier to debug or to modify file paths and loading procedures in one centralized location.

The script uses the following libraries:

- `pandas`: to read Excel data files.
- `joblib`: to load the pre-trained machine learning model.
- `os`: (partially included) for potential dynamic path management, though hardcoded paths are currently used.

**Key Variables**

- `MODEL_PATH` – points to the saved machine-learning model (`.pkl`) trained on the complete dataset and stored locally.
- `DATA_PATH` – points to the Excel file containing the latest input data used for predictions. The data is expected to be already prepared with the same structure used during training.

**Functions**

**(a)** **`get_data(path=DATA_PATH):`** This function reads the Excel file containing the new data that will be used for making predictions. This approach ensures that any data-loading issues are reported clearly without crashing the app.

- It prints the file path being accessed for logging purposes.
- If the file exists and is successfully read using `pd.read_excel()`, it returns the resulting DataFrame.
- If the file is not found, it catches the `FileNotFoundError` and returns an empty DataFrame after printing an error message.

**(b)** **`get_model(path=MODEL_PATH):`** This function loads the trained LightGBM model from the specified file path using `joblib`.

- It prints the path being accessed for easier debugging.
- If the model file exists, it is loaded and returned.
- If not, it catches the `FileNotFoundError` and returns `None`, allowing other parts of the application to handle the error gracefully.

### 6.5.2 `pipeline.py`: Main Prediction Flow

The `pipeline.py` script (as shown in Appendix G) acts as the bridge between feature engineering, model loading, and prediction generation. Rather than redefining preprocessing steps, it reuses the modular components implemented earlier in `Pipeline_classes.py` and the data/model utilities from `utils.py`.

**Main Imports**: The script starts by importing all the necessary modules, including:

- `pandas`: for handling data structures.
- `Pipeline` and `RobustScaler` from `sklearn`: to create and normalize the pipeline.
- Custom classes (`DataFrameSelector`, `FeatureEngineer`) from `Pipeline_classes.py`.
- Utility functions (`get_data`, `get_model`) from `utils.py`.

**Function `get_prediction_pipeline():`** This function constructs a complete `Pipeline` object using three main steps:

- `DataFrameSelector`: Selects the relevant input columns.
- `FeatureEngineer`: Applies custom feature engineering to enhance predictive power.
- `RobustScaler`: Normalizes the processed features to reduce the effect of outliers.

This function returns the pipeline, which is later used to transform new input data into a format suitable for model prediction.

**Function `run_prediction():`** This function orchestrates the end-to-end prediction workflow:

1. Loads the prepared dataset using `get_data()` from `utils.py`.
2. Constructs the preprocessing pipeline via `get_prediction_pipeline()`.
3. Transforms the input data to match training conditions.
4. Loads the trained LightGBM model using `get_model()` from `utils.py`.
5. Generates price predictions using the model's `predict()` method.
6. Returns the prediction results as a list.

Errors during data or model loading are caught early, ensuring the function fails gracefully without interrupting the API or deployment workflow. This function is critical as it links data, preprocessing, model loading, and prediction generation into one streamlined operation.

**Function `calculate savings ():`** This function extends the system's functionality by estimating potential user savings based on predicted market prices, following the same logic described earlier in **Section 6.4.6**. It uses the user's monthly energy consumption, current electricity price, and the model's predicted prices to calculate and return the potential savings and predicted average price, ensuring clear and user-friendly output formatting.

**Results and Interpretation**

Using the trained LightGBM model, the pipeline successfully produced `seven-day-ahead forecasts` for hourly electricity prices. The Table 30 demonstrates how closely predicted prices align with actual values:

*Table 30: Pipeline.py Actual vs Predicted 7days ahead CSV sample*

| INPUT DATE | PREDICTED DATE | ACTUAL PRICE (€) | PREDICTED PRICE (€) |
|---|---|---|---|
| 2024-01-07 23:00 | 2024-01-14 23:00 | 82.97 | 82.98 |
| 2024-01-08 00:00 | 2024-01-15 00:00 | 72.52 | 72.52 |
| 2024-01-08 01:00 | 2024-01-15 01:00 | 77.70 | 77.68 |
| 2024-01-08 02:00 | 2024-01-15 02:00 | 67.00 | 67.05 |
| 2024-01-08 03:00 | 2024-01-15 03:00 | 64.99 | 64.96 |

Differences were typically below `€1`, confirming strong generalization to unseen data. On average, the predicted market price was `0.072 €/kWh (72 €/MWh)`. When compared with a standard retail price of `0.12 €/kWh`, this represents an estimated `saving of ≈ 0.048 €/kWh`, or about `€47.9 per month` for a typical household.

These results confirm that the implemented pipeline not only reproduces the training performance but also supports actionable, user-level insights such as cost comparison and potential savings estimation.

### 6.5.3 Importance of Modular Script Structure

Dividing the application into well-defined scripts— `Pipeline_classes.py`, `utils.py`, and `pipeline.py` —was a deliberate design choice to enhance maintainability and reliability:

- Data or model loading issues can be quickly resolved within `utils.py`
- Pipeline or prediction-flow issues can be isolated in `pipeline.py`.
- Preprocessing or feature-engineering adjustments can be handled within `Pipeline_classes.py`.

This modular architecture simplifies debugging, allows independent testing of each component, and ensures a smooth deployment within the FastAPI environment for real-time electricity price forecasting.

## 6.6 Final API Endpoint Implementation

The final step in deploying the electricity price prediction system was to create a functional API that users can interact with. This was accomplished using the `FastAPI` framework (Sebastián Ramírez, 2018), a modern and efficient tool for building APIs with Python. The API script (as shown in Appendix H) is responsible for exposing endpoints that allow users to retrieve predicted electricity prices and calculate potential savings based on their current electricity usage.

**API Setup:** The API Setup script begins by importing the necessary libraries:

- `FastAPI` and `HTTPException` from the `fastapi` module to define the web server and handle errors.
- `BaseModel` from `pydantic` to define the structure of input data.
- The functions `run_prediction()` and `calculate_savings()` from the `Pipeline` script, which handle data processing, model inference, and savings calculation.

An instance of the `FastAPI` application is created with metadata such as the API's title, description, and version.

```
app = FastAPI(
    title="Electricity Price Prediction API",
    description="An API to predict electricity prices and calculate potential savings.",
    version="1.0.0"
)
```

**Input Data Schema:** In Input Data Schema the `SavingsInput` class is defined using `pydantic.BaseModel`. This class ensures that any data sent to the savings endpoint contains the following two fields:

- `monthly_consumption_kwh`: Monthly electricity usage in `kWh`.
- `current_price_kwh`: The price the user currently pays in `€/kWh`.

This enforces strict input validation to prevent errors from invalid user inputs.

**Caching Predictions:** A simple caching mechanism is implemented using a Python dictionary named `predictions_cache`. Its purpose is to store the model predictions when the server starts so that the model does not need to be reloaded and rerun on every request. This improves the performance and response time of the API.

```
predictions_cache = {
    "predictions": None
}
```

**Startup Event:** The `@app.on_event("startup")` function is executed automatically when the server starts. It runs the prediction pipeline once and stores the predictions in the cache. If predictions are successfully generated, a confirmation message is printed. If the model or data fails to load, a warning is printed instead. This approach ensures that the system is ready to respond to user requests as soon as it is up.

**API Endpoints:** The script defines three API routes:

1. **`GET /`:** This is a simple welcome route to confirm that the server is running. It returns a basic welcome message.
2. **`GET /predict`:** This endpoint returns the cached electricity price predictions. If the predictions are not available (e.g., due to a model loading error), it raises a 500 internal server error.
3. **`POST /calculate_savings`:** This endpoint accepts user input in the format defined by `SavingsInput` and returns the estimated savings using the cached model predictions. Internally, it calls the `calculate_savings()` function (which should be used here instead of `calcular_poupanca`, which appears to be a leftover from an earlier version). If there is an issue with predictions or calculations, it raises an appropriate error to inform the client.

## 6.7 Summary

In this chapter, the backend of the electricity price prediction system was structured into modular and maintainable components. In a first stage, pipeline classes (`FeatureEngineer`, `DataFrameSelector`) were defined to handle data transformation. Next, A utilities script was also built to manage data and model loading. Afterwards, the pipeline script was implemented, which brings together preprocessing, model inference, and user savings calculation.

Finally, the API endpoint script was developed using FastAPI, allowing users to interact with the model via a web interface. The API provides two main services: returning predicted electricity prices and estimating monthly savings based on user consumption.

Breaking the system into logical and reusable scripts ensures that the project is easy to debug, scale, and maintain. This completes the technical deployment of the model, enabling real-time electricity price forecasting and user cost analysis through a functional API interface.

# 7. Chapter 7: Conclusions

## 7.1 Summary of the thesis

The main objective of this thesis was to develop an accurate machine learning–based forecasting model for Portugal's electricity marginal price, capable of predicting prices seven days ahead (`168 hours`). To achieve this, multiple datasets were collected and integrated from authoritative sources (**OMIE, REN**, and **Copernicus CDS**), covering market, load, renewable generation, and weather variables. A series of feature engineering strategies was applied, including lagged variables, rolling statistics, calendar features, and interaction terms, to capture temporal dynamics and domain-specific relationships.

Several modeling attempts were conducted to refine both the feature set and the prediction horizon. In the initial stages, alternative formulations such as 7-day shifted targets and extensive lag structures were tested with tree-based models, including LightGBM, XGBoost, and Extra Trees. Although some experiments showed unrealistically low errors, further validation revealed that such approaches were not well aligned with the business requirement of 7-day hourly forecasts. Insights from these iterations, however, were crucial in identifying the most predictive features, particularly short-term lagged prices and load-based variables.

The final model (LightGBM) was developed using optimized hyperparameters, systematic train-validation-test splits, and a carefully engineered feature space. Results showed that the selected model achieved strong performance with MAE values within the acceptable range (<5 €/MWh), meeting the project's accuracy requirements. Beyond predictive accuracy, the project contributed with an operational end-to-end pipeline, where the final trained model was deployed through FastAPI endpoints, enabling real-time integration with applications and decision-support systems.

This work demonstrates that robust feature engineering combined with advanced tree-based models can deliver accurate and practical electricity price forecasts for the Portuguese market. By bridging data preprocessing, modeling, and deployment, the project moves beyond theoretical experimentation and provides a usable framework for real-world applications.

## 7.2 Assessment of the thesis contributions

From this study, three main contributions can be highlighted.

First, the development of an end-to-end EPF pipeline**.** The core objective was to build a forecasting model for Portugal's electricity marginal price, capable of predicting prices seven days ahead (`168 hours`), using a unified dataset compiled from OMIE, REN,

and Copernicus CDS data sources. The final model integrates advanced feature engineering (calendar, weather, renewable generation, and lagged load/price effects) with Optuna-based hyperparameter tuning to achieve robust predictive performance. The modular pipeline automates preprocessing, feature engineering, and prediction.

Second, a systematic evaluation of machine learning models was performed**.** Several algorithms (LightGBM, XGBoost, CatBoost, and Extra Trees) were compared across multiple iterations, including baseline models, enhanced feature sets, and regularized final models. This comparative analysis provided insight into which methods generalize best to unseen data in the electricity market context.

Third, a practical deployment through a FastAPI backend was provided**.** Beyond model training, the project contributes as an operational prototype for real-world usability. The trained model was embedded in a scalable pipeline and exposed via FastAPI endpoints, allowing predictions to be integrated into web applications or energy management tools.

Hence, beyond model accuracy, a significant contribution of this work lies in the integration of a pipeline and API endpoints, designed to ensure that the forecasting framework is not only theoretically sound but also practically deployable. The modular pipeline automates preprocessing, feature engineering, and prediction, while the FastAPI endpoints provide a direct interface for real-time or scheduled forecasting. This bridges the gap between academic modeling and applied energy analytics, offering scalability for future applications.

## 7.3 Considerations on model performance and selection

Across all the experiments performed in the context of this work on EPF, the LightGBM algorithm consistently outperformed alternative models, confirming its suitability for complex, high-dimensional time series prediction problems. When evaluated on the full dataset, the final Optuna-tuned LightGBM model achieved a MAE of 0.0822**.** This was the lowest error recorded across the entire project, reflecting substantial improvements compared to earlier model iterations. Importantly, the close alignment between validation and test MAE suggests that the model generalized well, avoiding the overfitting issues that often affect time series forecasting.

The feature importance analysis highlighted the dominance of lagged price variables (particularly `price_bef_169 and price_bef_170`), followed by variables capturing trends, wind speed, energy transactions, and load dynamics. This aligns with both market intuition and findings in the literature, where autoregressive behavior and demand-supply conditions remain central drivers of electricity prices.

The performance of the LightGBM model in this study is in line with broader research trends that favor ensemble tree-based methods for electricity price forecasting. Similar to findings by Lago (2021) and Marcjasz (2023), boosting-based approaches demonstrated both high accuracy and stability across multiple test scenarios. Compared to earlier neural network-based approaches, e.g., Catalão (2007), the results suggest that modern gradient boosting methods can achieve superior accuracy while requiring less computational complexity.

At the same time, the study reaffirms the importance of carefully engineered features, especially lagged prices and system load indicators, which have consistently been identified as critical inputs in prior forecasting frameworks (Weron, 2014). By incorporating both market-based and meteorological variables, this work contributes to the growing literature that highlights the multi-factorial drivers of electricity prices.

Overall, the findings confirm that advanced ensemble methods such as LightGBM, when paired with rigorous hyperparameter tuning and domain-specific feature engineering, can provide accurate, interpretable, and deployable electricity price forecasts.

## 7.4 Limitations

Despite the promising results achieved with the proposed electricity price prediction framework, several limitations remain. First, the model's performance is strongly dependent on the quality and completeness of the input data. Some features, particularly solar-related variables, contained long sequences of missing or zero values, which reduced their predictive contribution and, in certain cases, led to their removal. Similarly, the dataset covered only up to March 2025, limiting the ability to capture long-term structural changes in electricity markets.

Second, although extensive feature engineering was performed—including lagged variables, rolling statistics, calendar encodings, and interaction features—the approach relied primarily on manually designed features. More advanced techniques, such as automatic feature extraction through deep learning or embedding methods, were not explored due to time and resource constraints.

Third, while multiple models (LightGBM, XGBoost, CatBoost**,** and Extra Trees) were evaluated with hyperparameter optimization via Optuna, the experiments remained restricted to tree-based learners. Alternative approaches such as recurrent neural networks (RNNs), temporal convolutional networks (TCNs), or hybrid ensemble methods could potentially capture additional nonlinearities and long-range dependencies in electricity price dynamics.

Fourth, the project faced challenges with generalization. In several iterations, validation MAE and test MAE diverged, indicating overfitting to certain time periods or market conditions. Although regularization and feature selection mitigated this issue to some extent, ensuring stable performance across different future horizons remains difficult given the inherent volatility of electricity markets.

Finally, deployment considerations introduced further limitations. Normalization and feature engineering steps required consistent preprocessing at inference time, increasing pipeline complexity. Moreover, the current implementation focused on hourly seven-day-ahead (168-hour) forecasts rather than next-day predictions and did not extend to probabilistic forecasting or scenario analysis, which are often critical in real-world energy market applications.

## 7.5 Future Work

Building on the current findings, several directions can be pursued to improve and extend this work. First, incorporating richer and more diverse data sources could strengthen predictive performance. For example, integrating cross-border interconnection flows, fuel prices, or demand forecasts from neighboring markets may provide additional explanatory power. Similarly, higher-quality renewable generation datasets, with fewer missing values, would enhance the model's ability to capture the impact of weather variability.

Second, advanced modeling approaches could be explored. While this project focused on tree-based learners, deep learning architectures such as recurrent neural networks (**RNNs**), long short-term memory networks (**LSTMs**), or temporal convolutional networks (**TCNs**) may better capture nonlinear and long-range temporal dependencies. Hybrid ensemble strategies that combine machine learning with statistical time-series methods could also offer more robust performance.

Third, expanding the prediction task beyond deterministic point forecasts towards probabilistic forecasting would provide more actionable insights for energy stakeholders. Quantifying uncertainty in predictions is particularly valuable in volatile markets, as it allows operators and traders to assess risks and optimize strategies under different scenarios.

Fourth, further work is needed to improve deployment readiness. Simplifying the preprocessing pipeline, ensuring automated handling of missing or anomalous data, and optimizing for computational efficiency will support real-time or near-real-time forecasting in production systems. Integration with user interfaces or decision-support dashboards could also make the predictions more accessible to end-users.

Finally, evaluating the models under different market regimes and stress conditions—such as sudden renewable surges, unexpected demand peaks, or policy changes—would

provide a stronger test of robustness. Such analysis could guide the development of adaptive models that update dynamically as market conditions evolve.

# 8. Chapter 8: References


Abdul Razak, I. A. W., Abdullah, W. S. W., & Sulaima, M. F. (2023). Enhanced Short-term System Marginal Price (SMP) Forecast Modelling Using a Hybrid Model Combining Least Squares Support Vector Machines and the Genetic Algorithm in Peninsula Malaysia. *International Journal of Intelligent Systems and Applications in Engineering*, *11*(4), 289–298. https://ijisae.org/index.php/IJISAE/article/view/3525

ADENE – Agência para a Energia. (2024). *Energia em Números 2024*. ADENE. https://www.adene.pt/

Agrawal, R. K., Muchahary, F., & Tripathi, M. M. (2019). Ensemble of relevance vector machines and boosted trees for electricity price forecasting. *Applied Energy*, *250*, 540–548. https://doi.org/https://doi.org/10.1016/j.apenergy.2019.05.062

Bedir, B., Kanar, E., Akay, M., Abut, F., & Erdoğan, M. (2021). *Forecasting the System Marginal Price Using Long Short-Term Memory and Support Vector Machine*.

Catalão, J. P. S., Mariano, S. J. P. S., Mendes, V. M. F., & Ferreira, L. A. F. M. (2007). Short-term electricity prices forecasting in a competitive market: A neural network approach. *Electric Power Systems Research*, *77*(10), 1297–1304. https://doi.org/https://doi.org/10.1016/j.epsr.2006.09.022

Chatterjee, P. Y. M. F.-N. F. P.-R. J. (2023). *Machine Learning Algorithms and Applications in Engineering* (1st ed.). CRC Press.

Copernicus Climate Change Service (C3S). (2025). *Climate Data Store (CDS)*. https://cds.climate.copernicus.eu/

Cu, Y., Wang, K., Zhang, L., Liu, Z., Liu, Y., & Mo, L. (2025). A Time Series Decomposition-Based Interpretable Electricity Price Forecasting Method. *Energies*, *18*(3). https://doi.org/10.3390/en18030664

Data Science Stack Exchange. (2025). *Is it always better to use the whole dataset to train the final model?* Stack Exchange.

Díaz, G., Coto, J., & Gómez-Aleixandre, J. (2019). Prediction and explanation of the formation of the Spanish day-ahead electricity price through machine learning regression. *Applied Energy*, *239*, 610–625. https://doi.org/https://doi.org/10.1016/j.apenergy.2019.01.213

Direção-Geral de Energia e Geologia. (2024). *Balanço Energético Nacional 2024 (Provisório)*. Ministério Do Ambiente e Da Ação Climática. https://www.dgeg.gov.pt/

European Commission, D.-G. for C. A. (2025). *European Climate Law*. Climate Action — European Commission. https://climate.ec.europa.eu/eu-action/european-climate-law_en

Feng, C., Shao, L., Wang, J., Zhang, Y., & Wen, F. (2025). Short-term Load Forecasting of Distribution Transformer Supply Zones Based on Federated Model-Agnostic Meta Learning. *IEEE Transactions on Power Systems*, *40*(1), 31–45. https://doi.org/10.1109/TPWRS.2024.3393017

Giorgi M. (2025, April 9). Portugal to become net exporter of renewable energy before 2035. Strategic Energy Europe. *Aurora Energy Research. (2024, June)*. https://strategicenergy.eu/portugal-exporter-of-renewable-energy/

Gürtler, M., & Paulsen, T. (2018). Forecasting performance of time series models on electricity spot markets: a quasi-meta-analysis. *International Journal of Energy Sector Management*, *12*(1), 103–129. https://doi.org/10.1108/IJESM-06-2017-0004

IEA. (2023). *Efficiency & Demand – Portugal*. International Energy Agency (IEA). https://www.iea.org/countries/portugal/efficiency-demand

Imani, M. H., Bompard, E., Colella, P., & Huang, T. (2020). Predictive methods of electricity price: an application to the Italian electricity market. *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, 1–6. https://doi.org/10.1109/EEEIC/ICPSEurope49358.2020.9160561

Jędrzejewski, A., Lago, J., Marcjasz, G., & Weron, R. (2022). Electricity Price Forecasting: The Dawn of Machine Learning. *IEEE Power and Energy Magazine*, *20*(3), 24–31. https://doi.org/10.1109/MPE.2022.3150809

César D. M. P. (2023, June 7). *Portugal Electricity Price Forecast using Machine Learning and Sentiment Analysis*. https://fenix.tecnico.ulisboa.pt/cursos/memec21/dissertacao/283828618790899

Jufri, F. H., Oh, S., & Jung, J. (2019). Day-Ahead System Marginal Price Forecasting Using Artificial Neural Network and Similar-Days Information. *Journal of Electrical Engineering & Technology*, *14*(2), 561–568. https://doi.org/10.1007/s42835-018-00058-w

Kapoor, G., & Wichitaksorn, N. (2023). Electricity price forecasting in New Zealand: A comparative analysis of statistical and machine learning models with feature selection. *Applied Energy*, *347*, 121446. https://doi.org/https://doi.org/10.1016/j.apenergy.2023.121446

Khan, S., Aslam, S., Mustafa, I., & Aslam, S. (2021). Short-Term Electricity Price Forecasting by Employing Ensemble Empirical Mode Decomposition and

Extreme Learning Machine. *Forecasting*, *3*(3), 460–477. https://doi.org/10.3390/forecast3030028

Lago, J., Marcjasz, G., De Schutter, B., & Weron, R. (2021). Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy*, *293*, 116983. https://doi.org/10.1016/J.APENERGY.2021.116983

Laitsos, V., Vontzos, G., Bargiotas, D., Daskalopulu, A., & Tsoukalas, L. H. (2024). Data-Driven Techniques for Short-Term Electricity Price Forecasting through Novel Deep Learning Approaches with Attention Mechanisms. *Energies*, *17*(7). https://doi.org/10.3390/en17071625

Marcjasz, G., Narajewski, M., Weron, R., & Ziel, F. (2023). Distributional neural networks for electricity price forecasting. *Energy Economics*, *125*, 106843. https://doi.org/10.1016/J.ENECO.2023.106843

Matplotlib Development Team. (2025). *Matplotlib*. https://matplotlib.org/

Monteiro, C., Fernandez-Jimenez, L. A., & Ramirez-Rosado, I. J. (2015). Explanatory Information Analysis for Day-Ahead Price Forecasting in the Iberian Electricity Market. *Energies*, *8*(9), 10464–10486. https://doi.org/10.3390/en80910464

Nitsch, F., Schimeczek, C., & Bertsch, V. (2024). Applying machine learning to electricity price forecasting in simulated energy market scenarios. *Energy Reports*, *12*, 5268–5279. https://doi.org/10.1016/J.EGYR.2024.11.013

NumPy D. (2025). *NumPy*. https://numpy.org/

Odyssee-MURE Project. (2024). *Portugal Country Profile – Energy Efficiency and Indicators*. Odyssee-MURE. https://www.odyssee-mure.eu/

OMIE. (2025). *Operador del Mercado Ibérico de Energía (OMIE)*. https://www.omie.es/pt/market-results/daily/daily-market/day-ahead-price

ÖZGÜNER, E., Tör, O., & Guven, A. (2017). Probabilistic day-ahead system marginal price forecasting with ANN for the Turkish electricity market. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, *25*, 4923–4935. https://doi.org/10.3906/elk-1612-206

Pan, K. ; S. W. ; W. X. ; L. J. (2017). A Short-Term Marginal Price Forecasting Model Based on Ensemble Learning. *Proceedings of the 2017 International Conference on Progress in Informatics and Computing (PIC)*, 81–85.

Panapakidis, I. P., & Moschakis, M. N. (2019). Comparison of Machine Learning Models for the Prediction of System Marginal Price of Greek Energy Market . *International Journal of Information, Control and Computer Sciences*, *12.0*(3). https://doi.org/10.5281/zenodo.2643644

Pereira, C. D. M. (2023). *Portugal electricity price forecast using machine learning and sentiment analysis* [Master's dissertation]. Instituto Superior Técnico, Universidade de Lisboa.

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Ben Taieb, S., Bergmeir, C., Bessa, R. J., Bijak, J., Boylan, J. E., Browell, J., Carnevale, C., Castle, J. L., Cirillo, P., Clements, M. P., Cordeiro, C., Cyrino Oliveira, F. L., De Baets, S., Dokumentov, A., … Ziel, F. (2022). Forecasting: theory and practice. *International Journal of Forecasting*, *38*(3), 705–871. https://doi.org/10.1016/J.IJFORECAST.2021.11.001

REN – Redes Energéticas Nacionais. (2025). *REN Datahub*. https://datahub.ren.pt/pt/

SciPy Developers. (2025). *SciPy* . https://scipy.org/

Sebastián R. (2018). *FastAPI: Modern, fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints*. FastAPI Documentation. https://fastapi.tiangolo.com

Shim, S. W., Lee, D. H., Roh, J. H., & Park, J.-B. (2023). A Machine Learning-Based Algorithm for Short-Term SMP Forecasting Using 2-Step Method. *Journal of Electrical Engineering & Technology*, *18*(3), 1493–1501. https://doi.org/10.1007/s42835-023-01473-4

Singhal, D., & Swarup, K. S. (2011). Electricity price forecasting using artificial neural networks. *International Journal of Electrical Power & Energy Systems*, *33*(3), 550–555. https://doi.org/https://doi.org/10.1016/j.ijepes.2010.12.009

Stack Overflow. (2025). *Need for both CV and train-test split to assess a model's performance*. Stack Overflow.

Strategic Energy Europe. (2025, April 9). *Portugal to become net exporter of renewable energy before 2035, says Aurora Energy*. Strategic Energy Europe. https://strategicenergy.eu/portugal-exporter-of-renewable-energy/

The Pandas Development Team. (2025). *Pandas*. https://pandas.pydata.org/

Vanaci Prime. (2025). *Vanaci Prime*. https://vanaciprime.com/

Waskom, M. (2025). *Seaborn*. https://seaborn.pydata.org/

Weron, R. (2014). Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting*, *30*(4), 1030–1081. https://doi.org/https://doi.org/10.1016/j.ijforecast.2014.08.008

# 9. Appendices

## 9.1 Appendix A: Python Script for Exploratory Data Analysis (EDA)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns

df = pd.read_excel("/Users/naiyakhalid/Desktop/intership work /Prediction
project /Analysis/old-filtered-data/final-filtered-data.xlsx")

correlation_target = ['Preços marginais sistema portugues']

multi_corr_columns = [
                        'Day-ahead Total Load Forecast [MW] - BZN|PT',
                        'Lisboa temperature ° C',
                        'Porto temperature ° C',
                        'Évora temperature ° C',
                        'Preçs marginais sistema espanhol',
                        'Energia total de aquisição casada sistema
espanhol',
                        'Energia total de venda casada sistema espanhol',
                      ]

price_columns = ['Preços marginais sistema portugues']

tempo_columns = [
    'Day of the week',
    'week or weekend',
    'Holiday ',
    'Data e Dia da Semana',
    'Hora',
]

categorical_columns = [
    'Direction'
]

df_numerical = df.drop(columns=tempo_columns + multi_corr_columns +
categorical_columns)
df_tempo = df[tempo_columns + price_columns]
df_direction = df[categorical_columns+price_columns]
print(df_tempo.columns)

price_column = 'Preços marginais sistema portugues'
# Create a dictionary to store data for output
output = {}
print(df.head())

# Handle missing values using forward fill to maintain temporal order
df = df.ffill()

# Verify missing values after filling
df_null_nb = df_numerical.isnull().sum()
print(df_null_nb)

df_duplicates_nb = df_numerical.duplicated().sum()
```

```
print(df_duplicates_nb)

rows = len(df_numerical.columns)          # One row per feature
cols = 2                                  # Histogram and QQ plot

fig, axes = plt.subplots(rows, cols, figsize=(cols * 6, rows * 3))

# If there's only one row, keep axes 2D
if rows == 1:
    axes = np.expand_dims(axes, axis=0)

for i, col in enumerate(df_numerical.columns):
    # Histogram
    df_numerical[col].plot(kind='hist', bins=30, ax=axes[i][0],
color='skyblue', edgecolor='black')
    axes[i][0].set_title(f'Histogram - {col}')
    axes[i][0].set_xlabel(col)
    axes[i][0].set_ylabel('Frequency')
    # QQ Plot
    stats.probplot(df_numerical[col].dropna(), dist="norm",
plot=axes[i][1])
    axes[i][1].set_title(f'QQ Plot - {col}')

plt.tight_layout()
plt.show()

fig, axes = plt.subplots(rows, 3, figsize=(cols * 8, rows * 4))
axes = axes.flatten()

# Loop through each numeric column and plot the boxplot
for i, col in enumerate(df_numerical.columns):
    sns.boxplot(x=df_numerical[col], ax=axes[i], orient='h',
color='lightblue')
    axes[i].set_title(f'Box Plot - {col}')

# Remove any extra/unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()

correlation_value = df_numerical.corr()
print(correlation_value)

plt.figure(figsize=(len(df_numerical.columns), len(df_numerical.columns)
* 0.3))
sns.heatmap(correlation_value.T, annot=True, cmap='coolwarm', center=0)
plt.title("Correlation of Numerical Columns")
plt.tight_layout()
plt.show()

plt.figure(figsize=(len(df_numerical.columns), len(df_numerical.columns)
* 0.3))
sns.heatmap(correlation_value[correlation_target].T, annot=True,
cmap='coolwarm', center=0)
plt.title("Correlation of Numerical Fields with Price Columns")
plt.tight_layout()
plt.show()
```

```
over_40_corr =
correlation_value[correlation_target][correlation_value.abs() >
0.4].dropna().drop(correlation_target)
print(over_40_corr)

# ✅ Step 1: Convert datetime and hour columns safely
df_tempo = df[tempo_columns + price_columns].copy()

#Step 1: Convert datetime and hour columns
df_tempo.loc[:, 'Data e Dia da Semana'] = pd.to_datetime(df_tempo['Data e
Dia da Semana'])
df_tempo.loc[:, 'Hour'] = pd.to_datetime(df_tempo['Hora'],
format='%H:%M:%S').dt.hour
# df_tempo['Data e Dia da Semana'] = pd.to_datetime(df_tempo['Data e Dia
da Semana'])
# df_tempo['Hour'] = pd.to_datetime(df_tempo['Hora'],
format='%H:%M:%S').dt.hour

# Step 2: Extract additional date-related features
df_tempo['Day'] = df_tempo['Data e Dia da Semana'].dt.day
df_tempo['Month'] = df_tempo['Data e Dia da Semana'].dt.month
df_tempo['Year'] = df_tempo['Data e Dia da Semana'].dt.year
# df_tempo = df[tempo_columns + price_columns]

# Step 3: Compute correlation matrix with price
correlation_columns = [
    'Hour', 'Day', 'Month', 'Year',
    'Day of the week', 'week or weekend', 'Holiday ',
    'Preços marginais sistema portugues'
]

correlation_value_tempo = df_tempo[correlation_columns].corr()
print(correlation_value_tempo)
print(df_direction.head())  # ✅ Just view it, not call it

direction_encoded = pd.get_dummies(df[['Direction']], prefix='Direction')
df_direction = pd.concat([direction_encoded, df[['Preços marginais
sistema portugues']]], axis=1)

plt.figure(figsize=(len(df_tempo.columns) *3, len(df_tempo.columns) *
0.3))
sns.heatmap(correlation_value_tempo.T, annot=True, cmap='coolwarm',
center=0)
plt.title("Correlation of Temporical COlumns")
plt.tight_layout()
plt.show()

plt.figure(figsize=(len(df_tempo.columns) * 3 , len(df_tempo.columns) *
0.3))
sns.heatmap(correlation_value_tempo[correlation_target].T, annot=True,
cmap='coolwarm', center=0)
plt.title("Correlation of Temporical with Price Column")
plt.tight_layout()
plt.show()
print(df_direction)

direction_encoded = pd.get_dummies(df[['Direction']], prefix='Direction')
df_direction = pd.concat([direction_encoded, df[['Preços marginais
sistema portugues']]], axis=1)
correlation_value_direction = df_direction.corr()
```

```
plt.figure(figsize=(len(correlation_value_direction.columns) ,
len(correlation_value_direction.columns) * 0.3))
sns.heatmap(correlation_value_direction.T, annot=True, cmap='coolwarm',
center=0)
plt.title("Correlation of Direction")
plt.tight_layout()
plt.show()

plt.figure(figsize=(len(correlation_value_direction.columns) ,
len(correlation_value_direction.columns) * 0.3))
sns.heatmap(correlation_value_direction[correlation_target].T,
annot=True, cmap='coolwarm', center=0)
plt.title("Correlation of Directions with Price Columns")
plt.tight_layout()
plt.show()
```

## 9.2 Appendix B: Python Script for Final Data Summary and Quality Checks

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the Excel file
df = pd.read_excel('/Users/naiyakhalid/Desktop/intership work /Prediction
project /Analysis/new-data/updated_data.xlsx')

price_column = 'Preços marginais sistema portugues'
# Create a dictionary to store data for output
output = {}
df.head()

# 1. Average price per year
if 'Year' in df.columns and price_column in df.columns:
    avg_price_per_year =
df.groupby('Year')[price_column].mean().reset_index()
    avg_price_per_year.columns = ['Year', 'Average Price']
    print("\nAverage Price Per Year:\n", avg_price_per_year)
    output['Average Price Per Year'] = avg_price_per_year

    # Optional: plot it
    avg_price_per_year.plot(kind='bar', x='Year', y='Average Price',
title='Average Price per Year', legend=False)
    plt.ylabel('Average Price')
    plt.tight_layout()
    plt.show()
else:
    print("\nMissing 'Year' or specified price column.")

# 2. Outliers in 'price' column using IQR
if price_column in df.columns:
    Q1 = df[price_column].quantile(0.25)
    Q3 = df[price_column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[price_column] < lower_bound) | (df[price_column] >
```

```
upper_bound)]
    print(f"\nNumber of outliers in 'price': {len(outliers)}")
    output['Outliers in Price'] = outliers
    # Optional: visualize
    sns.boxplot(x=df[price_column])
    plt.title('Boxplot of Price')
    plt.show()
else:
    print("\n'price' column not found for outlier detection.")

# 3. Null values count per column
null_counts = df.isnull().sum().reset_index()
null_counts.columns = ['Column', 'Null Count']
output['Null Values'] = null_counts

# 4. Duplicate rows
duplicates = df[df.duplicated()]
output['Duplicate Rows'] = duplicates

# Now, to print them:
print("--- Null Values Count ---")
print(output['Null Values'])

print("\n--- Duplicate Rows ---")
print(output['Duplicate Rows'])

with pd.ExcelWriter('result.xlsx') as writer:
    for sheet_name, data in output.items():
        data.to_excel(writer, sheet_name=sheet_name, index=False)
print("Analysis complete. Results saved to 'result.xlsx'.")
```

## 9.3 Appendix C: Python Script for Final Model Selection: Optuna-Tuned LightGBM on Split Data

```
ort pandas as pd
import numpy as np
from lightgbm import LGBMRegressor, early_stopping, log_evaluation #
Import early_stopping and log_evaluation
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import RobustScaler
import optuna
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

# Load data
data = pd.read_excel("/Users/naiyakhalid/Desktop/intership work
/Prediction project /2024_2025_data.xlsx")
df = data.copy()
df.head(10)

# Target and lag features
df['target'] = df['portugal_marginal_price'].shift(-168)
df['price_bef'] = df['portugal_marginal_price'].shift(-167)
df['price_bef1'] = df['portugal_marginal_price'].shift(-166)
df['solar_radiation_168'] = df['solar_radiation'].shift(-168)
df['iberian_market_energy_168'] = df['iberian_market_energy'].shift(-168)
df['total_load_actual_168'] = df['total_load_actual'].shift(-168)

# Drop original target column
```

```
df.drop(columns=['portugal_marginal_price'], inplace=True)

# Calendar features
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df['day'] = df['date'].dt.day
df['month_sin'] = np.sin(2 * np.pi * df['month'] / 12)
df['month_cos'] = np.cos(2 * np.pi * df['month'] / 12)
df['day_sin'] = np.sin(2 * np.pi * df['day'] / 31)
df['day_cos'] = np.cos(2 * np.pi * df['day'] / 31)

# Interaction features
df['month_sin_x_solar_radiation_168'] = df['month_sin'] *
df['solar_radiation_168']
df['month_sin_x_iberian_market_energy_168'] = df['month_sin'] *
df['iberian_market_energy_168']
df['solar_radiation_168_x_iberian_market_energy_168'] =
df['solar_radiation_168'] * df['iberian_market_energy_168']
df['day_sin_x_total_load_actual_168'] = df['day_sin'] *
df['total_load_actual_168']
df['month_cos_x_solar_radiation_168'] = df['month_cos'] *
df['solar_radiation_168']

# More lags
df['price_bef_169'] = df['price_bef'].shift(-1)
df['price_bef_170'] = df['price_bef'].shift(-2)

# Rolling features
window_size = 24
df['target_roll_mean_24'] =
df['target'].rolling(window=window_size).mean()
df['solar_radiation_168_roll_mean_24'] =
df['solar_radiation_168'].rolling(window=window_size).mean()
df['target_roll_min_24'] = df['target'].rolling(window=window_size).min()
df['solar_radiation_168_roll_min_24'] =
df['solar_radiation_168'].rolling(window=window_size).min()
df['target_roll_max_24'] = df['target'].rolling(window=window_size).max()
df['solar_radiation_168_roll_max_24'] =
df['solar_radiation_168'].rolling(window=window_size).max()

# Ratio and diff features
df['solar_rad_to_load_168'] = df['solar_radiation_168'] /
df['total_load_actual_168'].replace(0, np.nan)
df['solar_rad_168_norm'] = df['solar_radiation_168'] /
df['solar_radiation_168_roll_max_24'].replace(0, np.nan)
df['price_trend'] = df['price_bef'] - df['price_bef1']
df['load_change'] = df['total_load_actual_168'] - df['total_load_actual']
df['price_bef_diff_pct'] = (df['price_bef'] - df['price_bef1']) /
df['price_bef1'].replace(0, np.nan)
df['price_bef_x_total_load_actual_168'] = df['price_bef'] *
df['total_load_actual_168']
df['price_bef1_x_solar_radiation_168'] = df['price_bef1'] *
df['solar_radiation_168']
df['price_bef_squared'] = df['price_bef'] ** 2
df['price_bef_diff'] = df['price_bef'] - df['price_bef1']

# More rolling features
windows = [24, 72, 168]
for w in windows:
    df[f'price_roll_mean_{w}'] = df['price_bef'].rolling(window=w).mean()
```

```
    df[f'load_roll_std_{w}'] =
df['total_load_actual'].rolling(window=w).std()

# Drop NaNs
df = df.dropna().reset_index(drop=True)

# Prepare features and target
X = df.drop(columns=['date', 'target'])
y = df['target']

# Train/Val/Test split masks
train_mask = df['year'] == 2024
val_mask = (df['year'] == 2025) & ((df['month'] == 1) | ((df['month'] ==
2) & (df['day'] < 15)))
test_mask = (df['year'] == 2025) & (((df['month'] == 2) & (df['day'] >=
15)) | (df['month'] == 3))

# Split data
X_train, y_train = X[train_mask], y[train_mask]
X_val, y_val = X[val_mask], y[val_mask]
X_test, y_test = X[test_mask], y[test_mask]

# Drop calendar cols after split
cols_to_drop = ['year', 'month', 'day']
X_train = X_train.drop(columns=cols_to_drop)
X_val = X_val.drop(columns=cols_to_drop)
X_test = X_test.drop(columns=cols_to_drop)

# Scale features
scaler = RobustScaler()
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train),
columns=X_train.columns, index=X_train.index)
X_val_scaled = pd.DataFrame(scaler.transform(X_val),
columns=X_val.columns, index=X_val.index)
X_test_scaled = pd.DataFrame(scaler.transform(X_test),
columns=X_test.columns, index=X_test.index)

# Optuna tuning
def objective(trial):
    params = {
        'n_estimators': 1000,
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.2),
        'num_leaves': trial.suggest_int('num_leaves', 20, 300),
        'feature_fraction': trial.suggest_float('feature_fraction', 0.4,
1.0),
        'bagging_fraction': trial.suggest_float('bagging_fraction', 0.4,
1.0),
        'bagging_freq': trial.suggest_int('bagging_freq', 1, 10),
        'min_child_samples': trial.suggest_int('min_child_samples', 5,
100),
        'random_state': 42
    }

    model = LGBMRegressor(**params)
    model.fit(
        X_train_scaled, y_train,
        eval_set=[(X_val_scaled, y_val)],
        eval_metric='mae',
        callbacks=[early_stopping(stopping_rounds=50), log_evaluation(0)]
    )
```

```
    preds = model.predict(X_val_scaled)
    mae = mean_absolute_error(y_val, preds)
    return mae

# Run optimization
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=200)

# Train final model
best_params = study.best_params
best_params['n_estimators'] = 1000
best_params['random_state'] = 42

final_model = LGBMRegressor(**best_params)
final_model.fit(
    X_train_scaled, y_train,
    eval_set=[(X_val_scaled, y_val)],
    eval_metric='mae',
    callbacks=[early_stopping(stopping_rounds=50), log_evaluation(0)] #
Corrected line
)

# Predictions and evaluation
y_train_pred = final_model.predict(X_train_scaled)
y_val_pred = final_model.predict(X_val_scaled)
y_test_pred = final_model.predict(X_test_scaled)

train_mae = mean_absolute_error(y_train, y_train_pred)
val_mae = mean_absolute_error(y_val, y_val_pred)
test_mae = mean_absolute_error(y_test, y_test_pred)

print(f"Train MAE: {train_mae:.4f}")
print(f"Validation MAE: {val_mae:.4f}")
print(f"Test MAE: {test_mae:.4f}")

import matplotlib.pyplot as plt

# Plot for Validation Set
plt.figure(figsize=(10, 5))
plt.plot(y_val.values, label="Actual", color="black", linewidth=2)
plt.plot(y_val_pred, label="Predicted", color="blue", alpha=0.7)
plt.title("Validation Set: Actual vs Predicted Prices")
plt.xlabel("Samples")
plt.ylabel("Price")
plt.legend()
plt.tight_layout()
plt.show()

# Plot for Test Set
plt.figure(figsize=(10, 5))
plt.plot(y_test.values, label="Actual", color="green", linewidth=2)
plt.plot(y_test_pred, label="Predicted", color="red", alpha=0.7)
plt.title("Test Set: Actual vs Predicted Prices")
plt.xlabel("Samples")
plt.ylabel("Price")
plt.legend()
plt.tight_layout()
plt.show()
```

## 9.4 Appendix D: Python Script for Final Model Trained on Full Dataset (No Split)

```
import pandas as pd
import numpy as np
from lightgbm import LGBMRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import RobustScaler
import joblib
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

# Load data
data = pd.read_excel("/Users/naiyakhalid/Desktop/intership work
/Prediction project /2024_2025_data.xlsx")
df = data.copy()
print(data.head())
print("\nColumns detected:", list(data.columns))

# Feature engineering
df['target'] = df['portugal_marginal_price'].shift(-168)
df['price_bef'] = df['portugal_marginal_price'].shift(-167)
df['price_bef1'] = df['portugal_marginal_price'].shift(-166)
df['solar_radiation_168'] = df['solar_radiation'].shift(-168)
df['iberian_market_energy_168'] = df['iberian_market_energy'].shift(-168)
df['total_load_actual_168'] = df['total_load_actual'].shift(-168)

# Drop original target column
df.drop(columns=['portugal_marginal_price'], inplace=True)

# Calendar features
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df['day'] = df['date'].dt.day
df['month_sin'] = np.sin(2 * np.pi * df['month'] / 12)
df['month_cos'] = np.cos(2 * np.pi * df['month'] / 12)
df['day_sin'] = np.sin(2 * np.pi * df['day'] / 31)
df['day_cos'] = np.cos(2 * np.pi * df['day'] / 31)

# Interaction features
df['month_sin_x_solar_radiation_168'] = df['month_sin'] *
df['solar_radiation_168']
df['month_sin_x_iberian_market_energy_168'] = df['month_sin'] *
df['iberian_market_energy_168']
df['solar_radiation_168_x_iberian_market_energy_168'] =
df['solar_radiation_168'] * df['iberian_market_energy_168']
df['day_sin_x_total_load_actual_168'] = df['day_sin'] *
df['total_load_actual_168']
df['month_cos_x_solar_radiation_168'] = df['month_cos'] *
df['solar_radiation_168']

# Lag features
df['price_bef_169'] = df['price_bef'].shift(-1)
df['price_bef_170'] = df['price_bef'].shift(-2)

# Rolling features
```

```
window_size = 24
df['target_roll_mean_24'] =
df['target'].rolling(window=window_size).mean()
df['solar_radiation_168_roll_mean_24'] =
df['solar_radiation_168'].rolling(window=window_size).mean()
df['target_roll_min_24'] = df['target'].rolling(window=window_size).min()
df['solar_radiation_168_roll_min_24'] =
df['solar_radiation_168'].rolling(window=window_size).min()
df['target_roll_max_24'] = df['target'].rolling(window=window_size).max()
df['solar_radiation_168_roll_max_24'] =
df['solar_radiation_168'].rolling(window=window_size).max()

# Ratio and diff features
df['solar_rad_to_load_168'] = df['solar_radiation_168'] /
df['total_load_actual_168'].replace(0, np.nan)
df['solar_rad_168_norm'] = df['solar_radiation_168'] /
df['solar_radiation_168_roll_max_24'].replace(0, np.nan)
df['price_trend'] = df['price_bef'] - df['price_bef1']
df['load_change'] = df['total_load_actual_168'] - df['total_load_actual']
df['price_bef_diff_pct'] = (df['price_bef'] - df['price_bef1']) /
df['price_bef1'].replace(0, np.nan)
df['price_bef_x_total_load_actual_168'] = df['price_bef'] *
df['total_load_actual_168']
df['price_bef1_x_solar_radiation_168'] = df['price_bef1'] *
df['solar_radiation_168']
df['price_bef_squared'] = df['price_bef'] ** 2
df['price_bef_diff'] = df['price_bef'] - df['price_bef1']

# More rolling features
windows = [24, 72, 168]
for w in windows:
    df[f'price_roll_mean_{w}'] = df['price_bef'].rolling(window=w).mean()
    df[f'load_roll_std_{w}'] =
df['total_load_actual'].rolling(window=w).std()


# Drop NaNs from rolling calculations and shifts
df = df.dropna().reset_index(drop=True)

# --- COUNT ZERO VALUES IN SOLAR COLUMNS ---
solar_cols = [col for col in df.columns if 'solar' in col]

print("\n=== ZERO COUNT PER SOLAR COLUMN ===")
for col in solar_cols:
    zero_count = df[col].eq(0).sum()
    total_count = len(df)
    print(f"{col}: {zero_count} zeros out of {total_count} rows
({zero_count/total_count:.2%})")
print("=== END ZERO COUNT ===\n")

# Prepare features and target
X = df.drop(columns=['date', 'target', 'year', 'month', 'day'])  # Drop
date/calendar columns
y = df['target']

# Scale features
scaler = RobustScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns,
index=X.index)
```

```
# Best Optuna parameters
best_params = {
    'n_estimators': 1000,
    'learning_rate': 0.0462,
    'num_leaves': 221,
    'feature_fraction': 0.9794,
    'bagging_fraction': 0.8991,
    'bagging_freq': 4,
    'min_child_samples': 54,
    'random_state': 42
}

# Final model training
final_model = LGBMRegressor(**best_params)
final_model.fit(X_scaled, y)

# Save model and scaler
joblib.dump(final_model, "final_lightgbm_full_dataset.pkl")
joblib.dump(scaler, "final_scaler.pkl")

# Evaluate on full data
y_pred = final_model.predict(X_scaled)
mae_full = mean_absolute_error(y, y_pred)
print(f"\n✅ Model trained on full dataset")
print(f"📊 MAE on Full Data: {mae_full:.4f}")

# # --- Save Actual vs Predicted results ---
# Add true predicted date
df['predicted_date'] = df['date'] + pd.Timedelta(hours=168)

# Save results
results_df = pd.DataFrame({
    'input_date': df['date'],
    'predicted_date': df['predicted_date'],
    'actual_price_7d_ahead': y,
    'predicted_price_7d_ahead': y_pred
})
# # Save to CSV
results_df.to_csv("actual_vs_predicted_7days_ahead.csv", index=False)
print("📁 Saved actual vs predicted prices to
'actual_vs_predicted_7days_ahead.csv'")

# Total no of rows used for training and prediction:
print(f"Total rows used for training and prediction: {len(df)}")
print(f"Number of predicted values: {len(y_pred)}")

# First and last input dates that used for prediction
print("First input date:", df['date'].min())
print("Last input date:", df['date'].max())

# compute target date (input + 168h)
df['predicted_date'] = df['date'] + pd.Timedelta(hours=168)
print("First predicted date:", df['predicted_date'].min())
print("Last predicted date:", df['predicted_date'].max())

# Feature importance
importances = final_model.feature_importances_
features = X.columns
importance_df = pd.DataFrame({'Feature': features, 'Importance':
```

```
importances}).sort_values(by='Importance', ascending=False)

print("\n🔍 Top 10 Most Important Features:")
print(importance_df.head(10))

# Plot top 20 features
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'][:20][::-1],
importance_df['Importance'][:20][::-1])
plt.title("Top 20 Feature Importances")
plt.xlabel("Importance Score")
plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt

# Plot Actual vs Predicted values for the full dataset
plt.figure(figsize=(12, 5))
plt.plot(y.values, label="Actual", color="red", linewidth=2)
plt.plot(y_pred, label="Predicted", color="blue", alpha=0.7)
plt.title("Final Model (No Split): Actual vs Predicted Electricity
Prices")
plt.xlabel("Samples")
plt.ylabel("Price")
plt.legend()
plt.tight_layout()
plt.show()

# Zoom in to first 300 samples for clarity
plt.figure(figsize=(12, 5))
plt.plot(y.values[:300], label="Actual", color="green", linewidth=2)
plt.plot(y_pred[:300], label="Predicted", color="red", alpha=0.7)
plt.title("Zoomed View (First 300 Samples): Actual vs Predicted")
plt.xlabel("Samples")
plt.ylabel("Price")
plt.legend()
plt.tight_layout()
plt.show()
```

## 9.5 Appendix E: Python Script for Deployment Pipeline (Pipeline_classes) script and Price Prediction Interface

```
import pandas as pd
import numpy as np
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import RobustScaler
from lightgbm import LGBMRegressor
import joblib
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
class DataFrameSelector(BaseEstimator, TransformerMixin):
    """
    Custom transformer to select specific columns from a DataFrame
    Handles both English and Portuguese column names
    """

    def __init__(self, required_columns=None, column_mapping=None):
        # Updated required columns based on your actual data structure
```

```
        self.required_columns = required_columns or [
            'date', 'solar_gen_day_ahead', 'wind_onshore_gen_day_ahead',
            'biomass_actual', 'fossil_gas_actual', 'hydro_pumped_actual',
            'hydro_pumped_consumption', 'hydro_runofriver_actual',
            'other_actual', 'solar_actual', 'wind_onshore_actual',
            'total_load_actual', 'wind_speed_kmh', 'solar_radiation',
            'precipitation_mm', 'lisbon_temp_c',
'portugal_marginal_price',
            'daily_market_energy', 'total_energy_acquisition',
            'total_energy_sale', 'export_spain_to_portugal',
'iberian_market_energy'
        ]

        # Your actual Portuguese to English mapping
        self.column_mapping = column_mapping or {
            'Data e Dia da Semana': 'date',
            'Generation - Solar [MW] Day Ahead/ BZN|PT':
'solar_gen_day_ahead',
            'Generation - Wind Onshore [MW] Day Ahead/ BZN|PT':
'wind_onshore_gen_day_ahead',
            'Biomass - Actual Aggregated [MW]': 'biomass_actual',
            'Fossil Gas - Actual Aggregated [MW]': 'fossil_gas_actual',
            'Hydro Pumped Storage - Actual Aggregated [MW]':
'hydro_pumped_actual',
            'Hydro Pumped Storage - Actual Consumption [MW]':
'hydro_pumped_consumption',
            'Hydro Run-of-river and poundage - Actual Aggregated [MW]':
'hydro_runofriver_actual',
            'Other - Actual Aggregated [MW]': 'other_actual',
            'Solar - Actual Aggregated [MW]': 'solar_actual',
            'Wind Onshore - Actual Aggregated [MW]':
'wind_onshore_actual',
            'Actual Total Load [MW] - BZN|PT': 'total_load_actual',
            'Speed_km_h': 'wind_speed_kmh',
            'Radiação Solar (W/m²)': 'solar_radiation',
            'Precipitação (mm) - Peso da Régua': 'precipitation_mm',
            'Lisboa temperature ° C': 'lisbon_temp_c',
            'Preços marginais sistema portugues':
'portugal_marginal_price',
            'Energia negociada Mercado Dia¡rio': 'daily_market_energy',
            'Energia total de aquisição casada sistema portuguas':
'total_energy_acquisition',
            'Energia total de venda casada sistema portugues':
'total_energy_sale',
            'Exportação de Espanha para Portugal':
'export_spain_to_portugal',
            'Energia Mercado IbÃ©rico incluindo lilaterais':
'iberian_market_energy'
        }

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        df = X.copy()

        # Check if we need to map Portuguese columns to English
        portuguese_cols =
set(self.column_mapping.keys()).intersection(set(df.columns))
        if portuguese_cols:
```

```
            print(f"🔄 Mapping Portuguese columns to English:
{portuguese_cols}")
            # Create mapping for existing Portuguese columns
            mapping_to_apply = {pt_col: en_col for pt_col, en_col in
self.column_mapping.items()
                                if pt_col in df.columns}
            df = df.rename(columns=mapping_to_apply)

        # Ensure all required columns are present after mapping
        missing_cols = set(self.required_columns) - set(df.columns)
        if missing_cols:
            available_cols = list(df.columns)
            raise ValueError(f"Missing required columns: {missing_cols}.
Available columns: {available_cols}")

        # Select only the required columns
        return df[self.required_columns].copy()

class FeatureEngineer(BaseEstimator, TransformerMixin):
    """
    Custom transformer for feature engineering
    Updated to work with your actual data structure
    """

    def __init__(self, prediction_horizon=168):
        self.prediction_horizon = prediction_horizon

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        df = X.copy()

        # Create target and shifted features (using
portugal_marginal_price as target)
        df['target'] = df['portugal_marginal_price'].shift(-
self.prediction_horizon)
        df['price_bef'] = df['portugal_marginal_price'].shift(-
(self.prediction_horizon - 1))
        df['price_bef1'] = df['portugal_marginal_price'].shift(-
(self.prediction_horizon - 2))

        # Shift other important features by prediction horizon
        df['solar_radiation_168'] = df['solar_radiation'].shift(-
self.prediction_horizon)
        df['iberian_market_energy_168'] =
df['iberian_market_energy'].shift(-self.prediction_horizon)
        df['total_load_actual_168'] = df['total_load_actual'].shift(-
self.prediction_horizon)

        # Calendar features
        df['year'] = df['date'].dt.year
        df['month'] = df['date'].dt.month
        df['day'] = df['date'].dt.day

        # Cyclical encoding
        df['month_sin'] = np.sin(2 * np.pi * df['month'] / 12)
        df['month_cos'] = np.cos(2 * np.pi * df['month'] / 12)
        df['day_sin'] = np.sin(2 * np.pi * df['day'] / 31)
        df['day_cos'] = np.cos(2 * np.pi * df['day'] / 31)
```

```
        # Interaction features
        df['month_sin_x_solar_radiation_168'] = df['month_sin'] *
df['solar_radiation_168']
        df['month_sin_x_iberian_market_energy_168'] = df['month_sin'] *
df['iberian_market_energy_168']
        df['solar_radiation_168_x_iberian_market_energy_168'] =
df['solar_radiation_168'] * df[
            'iberian_market_energy_168']
        df['day_sin_x_total_load_actual_168'] = df['day_sin'] *
df['total_load_actual_168']
        df['month_cos_x_solar_radiation_168'] = df['month_cos'] *
df['solar_radiation_168']

        # Lag features
        df['price_bef_169'] = df['price_bef'].shift(-1)
        df['price_bef_170'] = df['price_bef'].shift(-2)

        # Rolling features
        window_size = 24
        df['target_roll_mean_24'] =
df['target'].rolling(window=window_size).mean()
        df['solar_radiation_168_roll_mean_24'] =
df['solar_radiation_168'].rolling(window=window_size).mean()
        df['target_roll_min_24'] =
df['target'].rolling(window=window_size).min()
        df['solar_radiation_168_roll_min_24'] =
df['solar_radiation_168'].rolling(window=window_size).min()
        df['target_roll_max_24'] =
df['target'].rolling(window=window_size).max()
        df['solar_radiation_168_roll_max_24'] =
df['solar_radiation_168'].rolling(window=window_size).max()

        # Ratio and diff features
        df['solar_rad_to_load_168'] = df['solar_radiation_168'] /
df['total_load_actual_168'].replace(0, np.nan)
        df['solar_rad_168_norm'] = df['solar_radiation_168'] /
df['solar_radiation_168_roll_max_24'].replace(0, np.nan)
        df['price_trend'] = df['price_bef'] - df['price_bef1']
        df['load_change'] = df['total_load_actual_168'] -
df['total_load_actual']
        df['price_bef_diff_pct'] = (df['price_bef'] - df['price_bef1']) /
df['price_bef1'].replace(0, np.nan)
        df['price_bef_x_total_load_actual_168'] = df['price_bef'] *
df['total_load_actual_168']
        df['price_bef1_x_solar_radiation_168'] = df['price_bef1'] *
df['solar_radiation_168']
        df['price_bef_squared'] = df['price_bef'] ** 2
        df['price_bef_diff'] = df['price_bef'] - df['price_bef1']

        # More rolling features
        windows = [24, 72, 168]
        for w in windows:
            df[f'price_roll_mean_{w}'] =
df['price_bef'].rolling(window=w).mean()
            df[f'load_roll_std_{w}'] =
df['total_load_actual'].rolling(window=w).std()

        # # Drop NaNs from rolling calculations and shifts
        # df = df.dropna().reset_index(drop=True)
```

```
        # df = df.drop(columns=['year', 'month', 'day', 'target',
'portugal_marginal_price', 'date'])

        # Drop rows with NaNs produced by shifts/rolling BUT KEEP
ORIGINAL INDEX (do NOT reset)
        df_valid = df.dropna()

        # Save indices of valid rows for perfect alignment later
        self.feature_index = df_valid.index

        # Drop columns not used for prediction (keep features only)
        cols_to_drop = ['date', 'portugal_marginal_price', 'target',
'year', 'month', 'day']
        df_features = df_valid.drop(columns=[c for c in cols_to_drop if c
in df_valid.columns], errors='ignore')

        return df_features

        # --- COUNT ZERO VALUES IN SOLAR COLUMNS ---
        solar_cols = [col for col in df.columns if 'solar' in col]

        print("\n=== ZERO COUNT PER SOLAR COLUMN ===")
        for col in solar_cols:
            zero_count = df[col].eq(0).sum()
            total_count = len(df)
            print(f"{col}: {zero_count} zeros out of {total_count} rows
({zero_count / total_count:.2%})")
        print("=== END ZERO COUNT ===\n")

        return df

#----------------------
# Build pipeline (selector optional if needed)
# ----------------------
pipeline = Pipeline([
    ('feature_engineer', FeatureEngineer(prediction_horizon=168)),
    ('scaler', RobustScaler())
])

# ----------------------
# Load raw data (same file used in final model)
# ----------------------
data = pd.read_excel("/Users/naiyakhalid/Desktop/intership work
/Prediction project /2024_2025_data.xlsx")
# ensure date parsed as datetime
data['date'] = pd.to_datetime(data['date'])

# ----------------------
# Fit-transform pipeline on full dataset (training scenario)
# ----------------------
X_transformed = pipeline.fit_transform(data)  # feature_engineer stores
feature_index internally

# Save pipeline for reuse (you can load it later in a separate script)
joblib.dump(pipeline, "preprocessing_pipeline.pkl")
print("✅ Pipeline saved as 'preprocessing_pipeline.pkl'")

# ----------------------
# Load / train / or load the existing model. Here we assume model already
trained and saved as in your final script.
```

```
# If you're training here, train on X_transformed and target taken from
data.loc[feature_index, 'target'] etc.
# We'll assume model file exists:
# ----------------------
model = joblib.load('final_lightgbm_full_dataset.pkl')

# ----------------------
# Make predictions on transformed features
# ----------------------
predictions = model.predict(X_transformed)

# ----------------------
# Now use the stored feature_index to align dates & actuals exactly
# ----------------------
fe = pipeline.named_steps['feature_engineer']
if not hasattr(fe, 'feature_index') or fe.feature_index is None:
    raise RuntimeError("FeatureEngineer did not store feature_index. Make
sure pipeline.fit_transform(data) was called.")

feature_idx = fe.feature_index  # this is a pandas Index of original rows
that survived

# input dates taken from original data at those indices
input_dates = data.loc[feature_idx, 'date'].reset_index(drop=True)
predicted_dates = input_dates + pd.Timedelta(hours=168)

# actual price 7d ahead: shift original price and select same indices
actual_shifted = data['portugal_marginal_price'].shift(-
168).loc[feature_idx].reset_index(drop=True)

# assemble results exactly aligned
results_df = pd.DataFrame({
    'input_date': input_dates,
    'predicted_date': predicted_dates,
    'actual_price_7d_ahead': actual_shifted,
    'predicted_price_7d_ahead':
pd.Series(predictions).reset_index(drop=True)
})

# Save CSV that will match your final-model CSV ordering
results_df.to_csv("pipeline_aligned_actual_vs_predicted.csv",
index=False)
print("✅ Saved pipeline_aligned_actual_vs_predicted.csv")
print(results_df.head(12))


# # 3️⃣ Build pipeline
# pipeline = Pipeline([
#     # ('selector', DataFrameSelector()),
#     ('feature_engineer', FeatureEngineer()),
#     ('scaler', RobustScaler()),
# ])
#
# # Load data
# data = pd.read_excel("/Users/naiyakhalid/Desktop/intership work
/Prediction project /2024_2025_data.xlsx")
#
# # Fit and transform
# new_data = pipeline.fit_transform(data)
# data.head(10)
```

```
# len(new_data)
#
# # Save pipeline
# import joblib
# joblib.dump(pipeline, "preprocessing_pipeline.pkl")
# print("✅ Pipeline saved as 'preprocessing_pipeline.pkl'")
#
# # Use pipeline for predictions # Load model and predict
# model = joblib.load('final_lightgbm_full_dataset.pkl')
# predictions = model.predict(new_data)
# print(predictions)
#
# # Number of rows dropped due to shifts/NaNs
# rows_dropped = len(data) - len(new_data)
#
# # Align dates properly
# predicted_dates = data['date'].iloc[rows_dropped:] +
pd.Timedelta(hours=168)
# results_df = pd.DataFrame({
#     'input_date': data['date'].iloc[rows_dropped:],
#     'predicted_date': predicted_dates,
#     'predicted_price_7d_ahead': predictions
# })
#
# results_df.to_csv("predicted_7days_ahead_with_dates.csv", index=False)
# print("✅ Predictions with dates saved")
# print(results_df.head(10))

# # Função para calcular economia baseada no modelo doctrinaire
def calcular_poupanca(consumo_mensal_kwh, preco_atual_pago):
    """
    consumo_mensal_kwh: float - Consumo mensal do cliente em kWh
    preco_atual_pago: float - Preço atual pago pelo cliente em €/kWh
    """
    # Prever o preço médio usando o modelo treinado
    preco_previsto_mwh = predictions.mean()
    preco_previsto_kwh = preco_previsto_mwh / 1000  # Convert from €/MWh
to €/kWh

    poupanca_media_kwh = preco_atual_pago - preco_previsto_kwh
    poupanca_total_mensal = poupanca_media_kwh * consumo_mensal_kwh

    return preco_previsto_kwh, poupanca_media_kwh, poupanca_total_mensal


# Exemplo de uso da função calcular poupança
consumo = 1000  # kWh
preco_atual = 0.12  # €/kWh
preco_previsto, poupanca_media, poupanca_total =
calcular_poupanca(consumo, preco_atual)
print(f"Preço médio previsto: {preco_previsto:.4f} €/kWh")
print(f"Poupança média por kWh: {poupanca_media:.4f} €/kWh")
print(f"Poupança total mensal: {poupanca_total:.2f} €")
```

## 9.6 Appendix F: Python Script for `utils.py`: Data and Model Utilities

```
import pandas as pd
import joblib
```

```
import os

# MODEL_DIR = os.path.join(os.path.dirname(__file__), '..', 'models')
# MODEL_PATH = os.path.join(MODEL_DIR, 'final_lightgbm_full_dataset.pkl')

MODEL_PATH =
"/home/athul/vanaci_legal/electricity_price_prediction/final_lightgbm_ful
l_dataset.pkl"  # Update this path as needed

DATA_PATH = "/Users/naiyakhalid/Desktop/intership work /Prediction
project /2024_2025_data.xlsx" # Update this path as needed

def get_data(path: str = DATA_PATH) -> pd.DataFrame:

    print(f"Loading data from: {path}")
    try:
        data = pd.read_excel(path) # , parse_dates=['Data e Dia da
Semana']
        return data
    except FileNotFoundError:
        print(f"Error: Data file not found at {path}")
        return pd.DataFrame()


def get_model(path: str = MODEL_PATH):
    print(f"Loading model from: {path}")
    try:
        model = joblib.load(path)
        return model
    except FileNotFoundError:
        print(f"Error: Model file not found at {path}")
        return None
```

## 9.7 Appendix G: Python Script for `pipeline.py` script

```
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import RobustScaler

from Pipeline_classes import DataFrameSelector, FeatureEngineer
from utils import get_data, get_model


def get_prediction_pipeline() -> Pipeline:
    return Pipeline([
        ('selector', DataFrameSelector()),
        ('feature_engineer', FeatureEngineer()),
        ('scaler', RobustScaler()),
    ])


def run_prediction() -> list:
    raw_data = get_data()
    if raw_data.empty:
        print("Stopping prediction due to data loading error.")
        return []

    processing_pipeline = get_prediction_pipeline()

    print("Transforming data...")
```

```
    processed_data = processing_pipeline.fit_transform(raw_data)
    print(f"Data transformed successfully. Shape:
{processed_data.shape}")

    model = get_model()
    if model is None:
        print("Stopping prediction due to model loading error.")
        return []

    print("Making predictions...")
    predictions = model.predict(processed_data)
    print("Predictions generated.")

    return predictions.tolist()


def calculate_savings(consumo_mensal_kwh: float, preco_atual_pago: float,
predictions: list) -> dict:
    """
    Calculates potential savings based on model predictions.

    Args:
        consumo_mensal_kwh (float): Monthly consumption in kWh.
        preco_atual_pago (float): Current price paid by the client in
€/kWh.
        predictions (list): A list of predicted prices in €/MWh.

    Returns:
        dict: A dictionary containing the predicted price and savings.
    """
    if not predictions:
        return {
            "predicted_price_kwh": None,
            "average_saving_kwh": None,
            "total_monthly_saving": None,
            "error": "Could not generate predictions."
        }

    preco_previsto_mwh = pd.Series(predictions).mean()
    preco_previsto_kwh = preco_previsto_mwh / 1000  # Convert from €/MWh
to €/kWh

    poupanca_media_kwh = preco_atual_pago - preco_previsto_kwh
    poupanca_total_mensal = poupanca_media_kwh * consumo_mensal_kwh

    return {
        "predicted_price_kwh": round(preco_previsto_kwh, 4),
        "average_saving_kwh": round(poupanca_media_kwh, 4),
        "total_monthly_saving": round(poupanca_total_mensal, 2)
    }

# def calcular_poupanca(predictions, consumo_mensal_kwh,
preco_atual_pago):
#     """
#     consumo_mensal_kwh: float - Consumo mensal do cliente em kWh
#     preco_atual_pago: float - Preço atual pago pelo cliente em €/kWh
#     """
#     # Prever o preço médio usando o modelo treinado
#     preco_previsto_mwh = predictions.mean()
#     preco_previsto_kwh = preco_previsto_mwh / 1000  # Convert from
```

```
€/MWh to €/kWh

#     poupanca_media_kwh = preco_atual_pago - preco_previsto_kwh
#     poupanca_total_mensal = poupanca_media_kwh * consumo_mensal_kwh

#     return preco_previsto_kwh, poupanca_media_kwh,
poupanca_total_mensal
```

## 9.6 Appendix H: Python Script for Final API Endpoint Implementation

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel

# from Pipeline_classes import calcular_poupanca
from Pipeline import run_prediction, calculate_savings

app = FastAPI(
    title="Electricity Price Prediction API",
    description="An API to predict electricity prices and calculate
potential savings.",
    version="1.0.0"
)
class SavingsInput(BaseModel):
    monthly_consumption_kwh: float
    current_price_kwh: float

# --- Caching Predictions ---
predictions_cache = {
    "predictions": None
}

@app.on_event("startup")
def startup_event():
    print("🚀 Server starting up! Running initial prediction
pipeline...")
    predictions_cache["predictions"] = run_prediction()
    if not predictions_cache["predictions"]:
        print("⚠️ Warning: Initial predictions could not be generated.")
    else:
        print("✅ Initial predictions are cached and ready.")

@app.get("/")
def read_root():
    return {"message": "Welcome to the Electricity Price Prediction
API!"}

@app.get("/predict", tags=["Prediction"])
def get_predictions():
    predictions = predictions_cache["predictions"]
    if not predictions:
        raise HTTPException(status_code=500, detail="Model predictions
are not available. Check server logs.")

    return {"price_predictions_eur_per_mwh": predictions}

@app.post("/calculate_savings", tags=["Savings Calculator"])
def post_calculate_savings(data: SavingsInput):
    predictions = predictions_cache["predictions"]
    if not predictions:
```

```
        raise HTTPException(status_code=500, detail="Model predictions
are not available. Check server logs.")

    savings_result = calcular_poupanca(
        consumo_mensal_kwh=data.monthly_consumption_kwh,
        preco_atual_pago=data.current_price_kwh,
        predictions=predictions
    )

    if "error" in savings_result:
        raise HTTPException(status_code=500,
detail=savings_result["error"])

    return savings_result
```


UNIVERSIDADE
PORTUCALENSE

upt.pt