Data Science Applied to Electrical Energy Systems

This detailed report provides an in-depth analysis of machine learning model's performance in predicting electricity prices, examining key metrics, trends, and areas for improvement to enhance forecasting accuracy and optimize energy market operations.

Prepared by:

Miloš Sarić

Advisors:

Dr. Aleksandar A. Sarić

Table of Contents

1. Giving Computers The Ability to Learn From Data

Machine learning— the application and science of algorithms that make sense of data—is the most thrilling area within computer science today. In an era where data is generated at unprecedented volumes, machine learning empowers us to transform this vast information into knowledge using self-learning algorithms. With the rise of powerful open-source libraries in recent years, there has never been a more opportune time to dive into machine learning, harness its algorithms to uncover patterns in data, and make accurate predictions about future outcomes.

Machine learning often comes up alongside terms like *big data* and *artificial intelligence* (A.I.), but it's not the same as either of them. To understand what machine learning is and why it's so useful, it helps to first understand big data and how machine learning works with it. Big data simply refers to the huge amounts of information being collected and stored today, whether from cameras, sensors, or social media. For example, Google processes over 20 petabytes of data every day—and that number is still growing. According to IBM, we create around 2.5 quintillion bytes of data *every single day*, and shockingly, 90% of all the data in the world was created in just the last two years. Machine learning makes sense of this overwhelming amount of data by finding patterns, learning from them, and helping us make smarter predictions and decisions.

Humans alone simply can't process or analyze the massive amounts of data being generated today. That's where machine learning comes in. It's a powerful tool designed to handle and make sense of huge, complex datasets with countless variables and features. One of the biggest advantages of machine learning—especially deep learning—is that it works even better with large amounts of data. The more data these techniques have to work with, the better they get at identifying patterns and trends hidden within it, which improves their ability to analyze information and make accurate predictions.

In today's world of modern technology, one thing we have plenty of is data—both structured and unstructured. During the second half of the 20th century, machine learning emerged as a branch of artificial intelligence (AI), focusing on self-learning algorithms that extract knowledge from data and use it to make predictions.

Machine learning plays a key role in artificial intelligence (A.I.) systems, acting as "the brain" that enables them to make sense of their surroundings and respond appropriately. While artificial intelligence can be described as a system that interacts with its environment—using sensors to gather information and tools to respond—machine learning powers the decision-making process. For example, Siri on an iPhone listens to a command through its microphone and responds through the speaker or display, but it needs machine learning to "understand" what was said and provide the right answer. Similarly, driverless cars use cameras, GPS, sonars, and lidar sensors to gather information about their surroundings. Machine learning processes all this data to determine the correct action, such as whether to accelerate, brake, or turn. In essence, machine learning is the engine that processes the data and drives decision-making in A.I. systems.

Machine learning is not only a growing focus in computer science research but also an increasingly integral part of our daily lives. It powers tools and services we rely on, such as email spam filters, text and voice recognition, web search engines, movie recommendations, mobile check deposits, and even

estimated meal delivery times. In the near future, safe and efficient self-driving cars may also join this list. Significant progress has been made in medical applications as well.

Beyond all this, machine learning is also being applied to one of today's most pressing challenges in electricity sector.

1.1. Why Use Machine Learning

How could we write a program to predict electricity prices for tomorrow using traditional programing techniques?

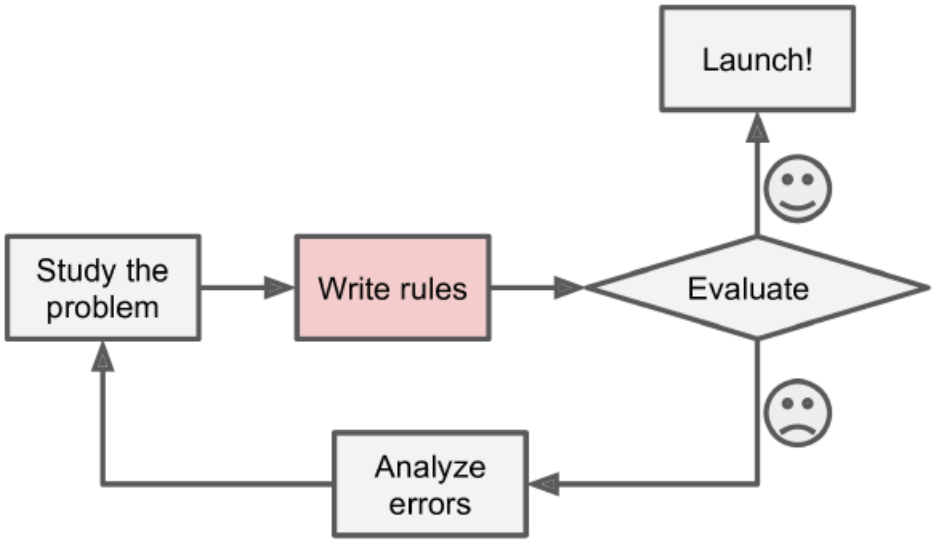

Figure 1.1. The traditional approach

1. First, we would "Study the problem" and understand the factors influencing electricity prices, such as demand, supply, weather conditions, fuel costs, government policies, and grid stability. We would Analyze historical data, identify patterns, and understand external influences like peak hours or unusual events (outages or extreme weather).
2. We would develop "*if-then rules*" based on insights from the first step.
   - If demand exceeds supply by X%, increase price by Y%
   - If the temperature is above 35 °C, and it's summer, increase prices by Z% during peak hours
   - If renewable energy output is low and fuel prices are high, adjust prices proportionally.

   These rules should be deterministic, meaning they rely strictly on predefined thresholds and conditions.
3. Test these rules against historical data to see how accurately they predict electricity prices. We would test our program, and repeat steps 1 and 2 until it is good enough.
4. Where the rules fail (predictions deviate significantly from actual prices) we need to identify the cause. Were critical factors missed? Were thresholds or parameters inaccurate? A sudden spike in prices due to an unanticipated power plant failure may indicate a missing rule or factor.
5. Update or add rules based on error analysis.

6. Once the rules consistently produce accurate predictions, deploy the system to predict electricity prices in real time.

Since the problem is not trivial, our program will likely become a long list of complex rules—pretty hard to maintain. In contrast, a system for predicting tomorrow's electricity price based on Machine Learning techniques automatically learns which factors and patterns (such as demand, weather, and fuel costs) are good predictors of price fluctuations by detecting complex relationships and trends in historical data. The program is much shorter, easier to maintain, and most likely more accurate.

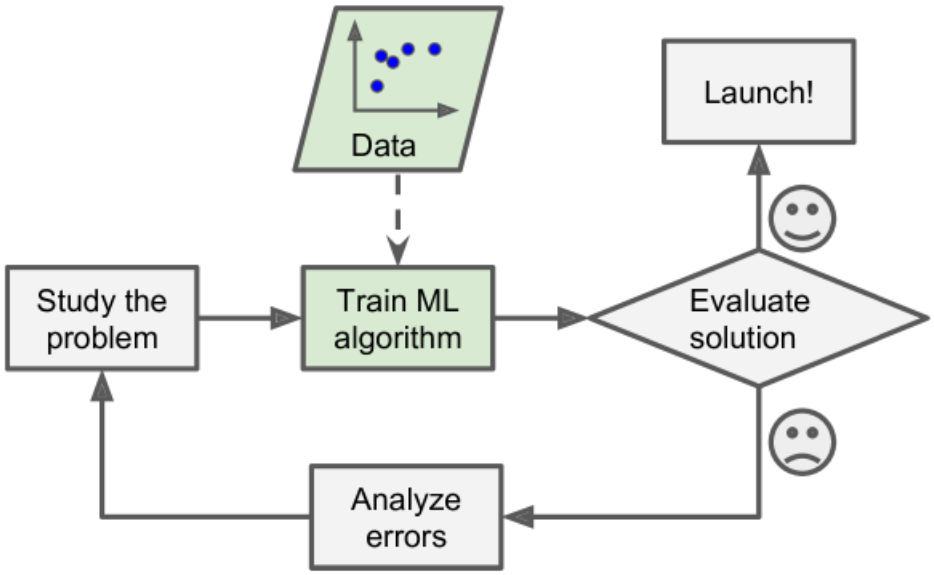

Figure 1.2. Machine Learning approach

If electricity market operators notice that prices always spike during extreme heatwaves and adjust their rule-based system to account for this, they might miss new scenarios, like sudden price increases caused by unexpected power plant failures. A rule-based system for predicting electricity prices would require constant updates to include new rules for each unforeseen situation. For example, if renewable energy supply drops unexpectedly due to unusual weather conditions, the rules would need to be rewritten to capture this pattern.

In contrast, a system based on Machine Learning techniques automatically learns that unexpected weather patterns or supply shortages are becoming significant predictors of price spikes. It adjusts its predictions by analyzing historical data and identifying these trends, all without requiring manual updates (as shown in the diagram).

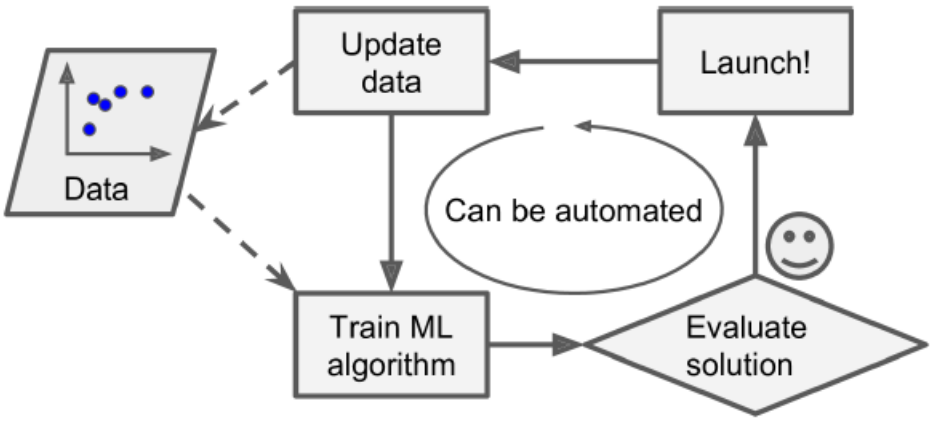

Figure 1.3. Automatically adapting to change

Finally, Machine Learning can help humans better understand electricity price dynamics. ML algorithms can be inspected to see what they have learned about predicting prices (though for some algorithms, like neural networks, this can be challenging). For example, once an ML model has been trained on sufficient electricity market data, it can reveal which factors—such as peak demand times, weather patterns, or fuel price fluctuations—it considers the best predictors of price changes. This analysis can uncover unexpected correlations, such as a strong link between wind speed and price dips due to increased wind energy supply, leading to a deeper understanding of the market and more informed decision-making.

Applying Machine Learning techniques to analyze large volumes of electricity market data can help uncover patterns that were not immediately obvious. This process, known as data mining, might reveal insights like the influence of regional weather anomalies on price fluctuations, the hidden impact of industrial demand surges, or seasonal trends in renewable energy contributions that significantly affect pricing.

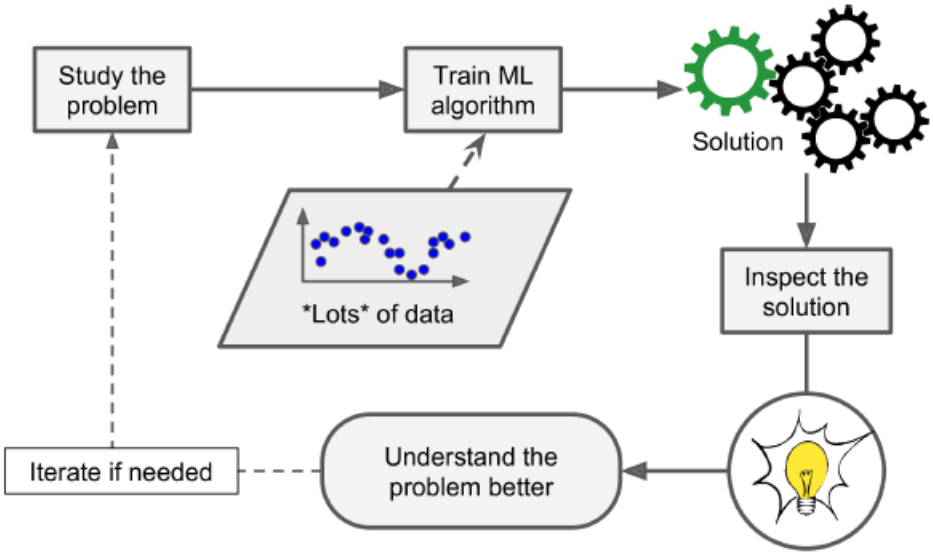

Figure 1.4. Machine Learning can help humans learn

To summarize, Machine Learning is great for predicting electricity prices because:

1. Problems requiring extensive manual adjustments or complex rule sets: Instead of writing and maintaining long lists of rules for different scenarios, a single Machine Learning model can simplify predictions and often achieve better accuracy.
2. Complex problems with no clear traditional solution: ML can model intricate relationships, like the combined effects of weather, demand, and renewable energy supply, which are challenging to solve with traditional methods.
3. Fluctuating environments: ML models can adapt to changing patterns, such as shifts in energy consumption trends or the introduction of new power sources.
4. Gaining insights from complex data: ML can analyze large electricity market datasets to uncover hidden patterns and trends, providing valuable insights for decision-making.

1.2. Different Machine Learning Approaches

The term "*machine learning*" is used in a broad sense, referring to methods that help find patterns in large amounts of data or make predictions based on what we've learned from analyzing known data. This is a general definition that covers many different techniques. Machine learning is usually divided into two main types: Supervised Learning and Unsupervised Learning. There's also a third type, called Reinforcement Learning, which is often included as well.

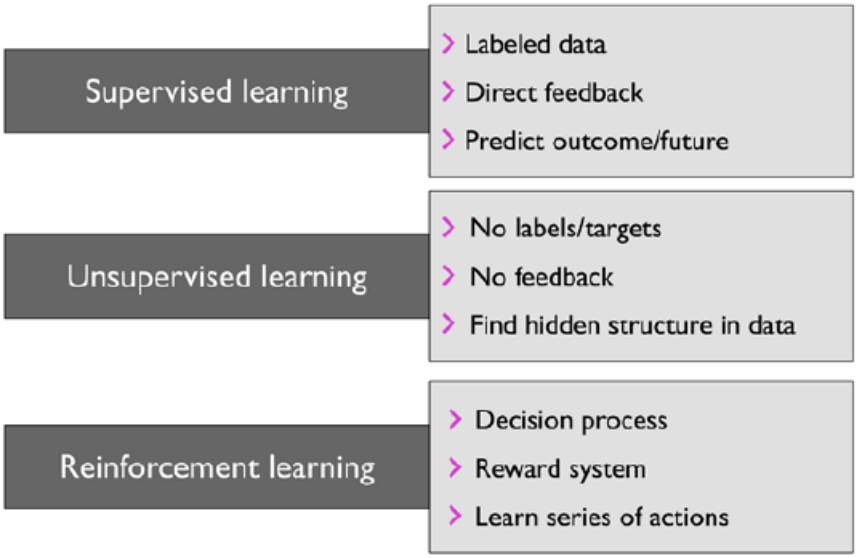

Figure 2.1. The three different types of machine learning

2. Making predictions about the future with supervised learning

In supervised learning, the primary aim is to develop a model using labeled training data to make accurate predictions on new or unseen data. The term "supervised" highlights the use of training examples where the input data comes paired with known output labels. Essentially, supervised learning involves capturing and modeling the relationship between these inputs and their corresponding labels. For this reason, it's often thought of as a process of "learning from labels."

Figure 2.1. illustrates the typical workflow of supervised learning. In this process, labeled training data is provided to a machine learning algorithm, which then builds a predictive model. This model is designed to generate predictions for new, unlabeled input data.

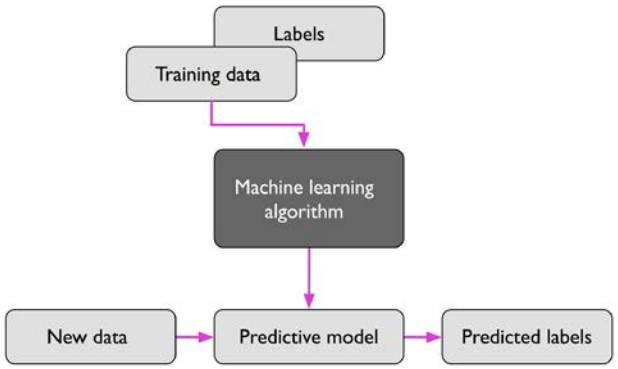

Figure 2.1. Supervised Learning Process

2.1. Regression – Predicting Electricity Price for Tomorrow

To predict a numeric target value, such as tomorrow's electricity price, based on a set of features (demand, renewable generation, past electricity prices, etc.), we use a type of task called regression. To train the system, we provide it with numerous examples of historical data, including both the predictors (the input features) and the labels (the actual electricity prices).

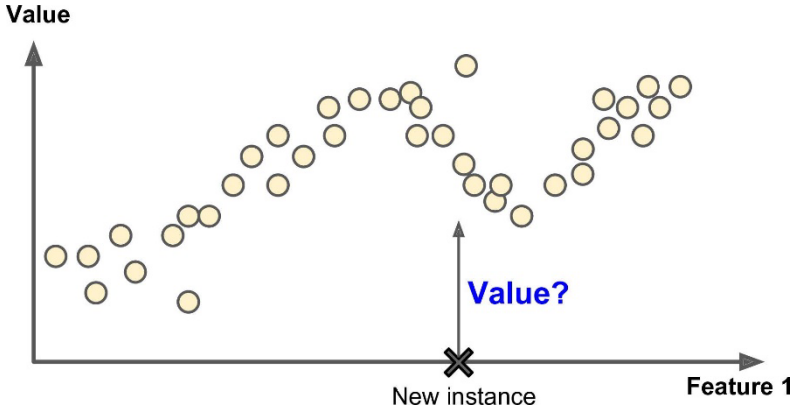

Figure 2.1.1. Regression

To illustrate how data used to predict electricity price change throughout a day, I've chosen to plot data from a random day (27th April 2024). This approach provides a clear snapshot of the trends and variations over time.

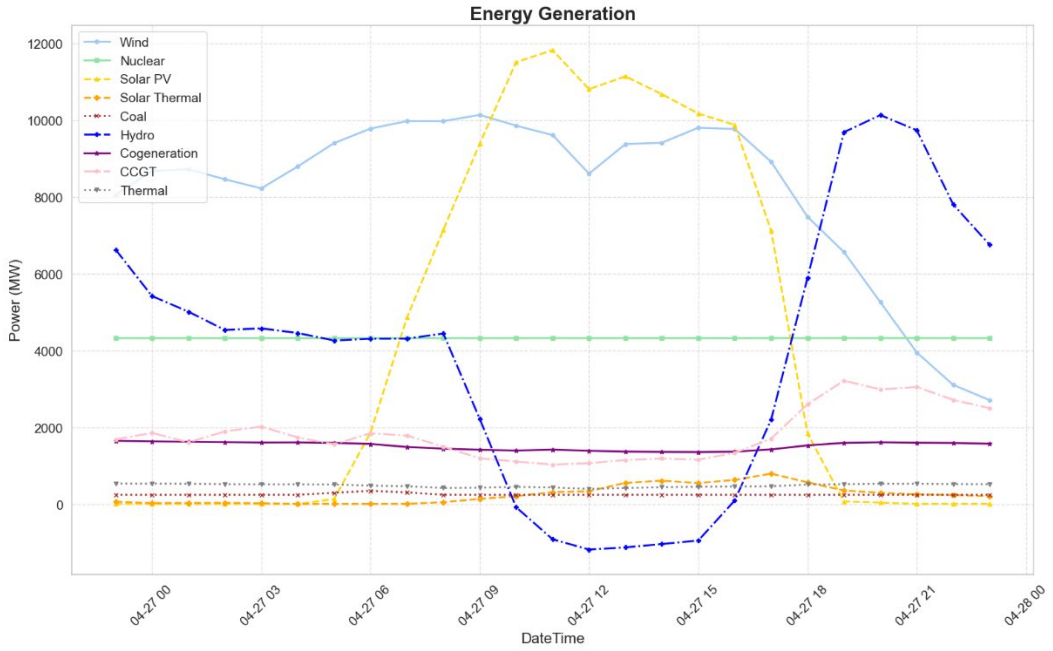

Figure 2.1.2. Energy Generation from Different Sources

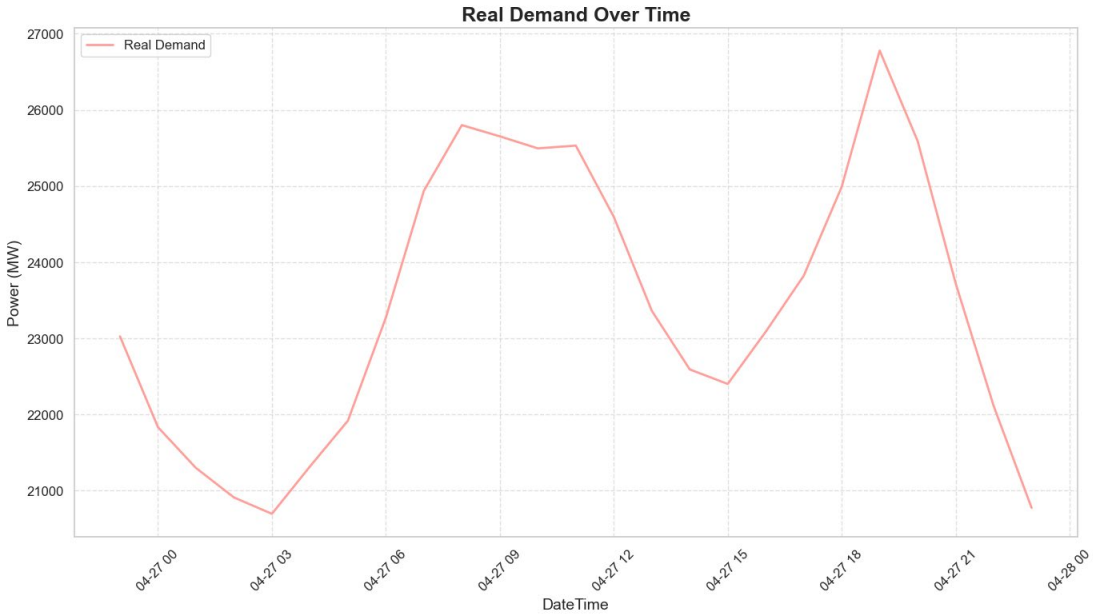

Figure 2.1.3. Real Demand

In the context of electricity grids, generation and consumption must always remain balanced to ensure stability and reliable operation. This balance is crucial because electricity cannot be easily stored

at scale, and any significant mismatch between supply and demand can lead to grid instability, blackouts, or overloading of infrastructure.

ELECTRICITY PRICES FOR THREE RANDOM DAYS:

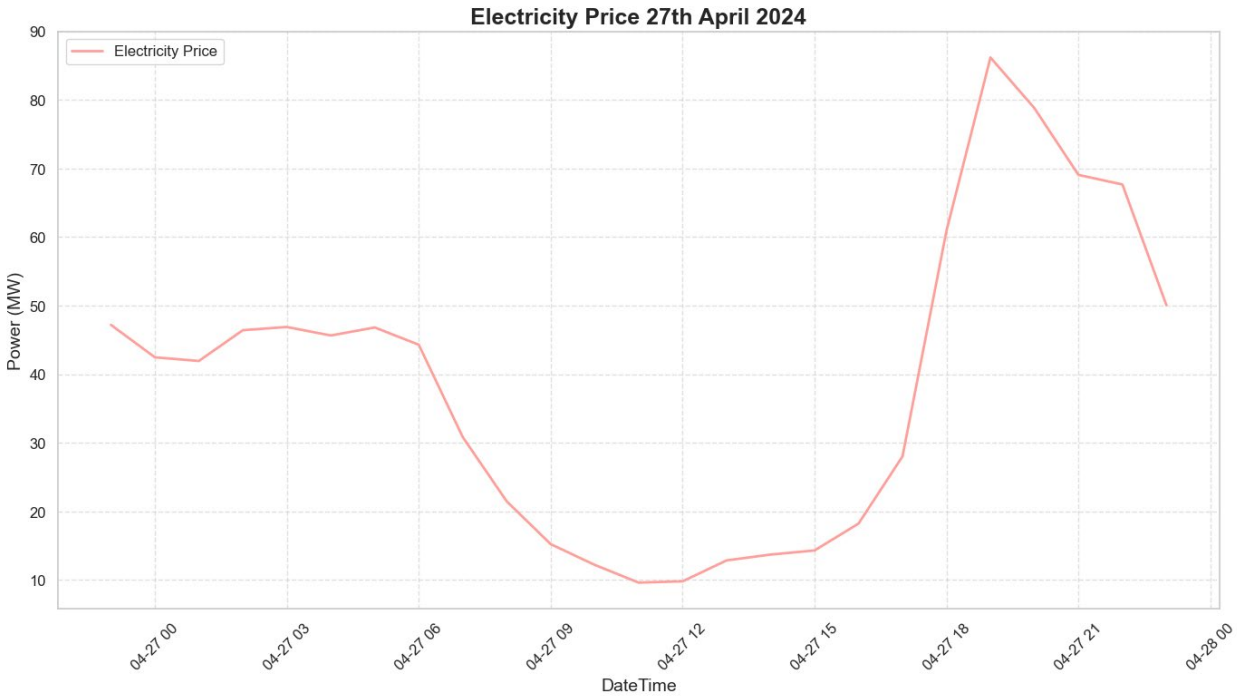

Figure 2.1.4. 27th April

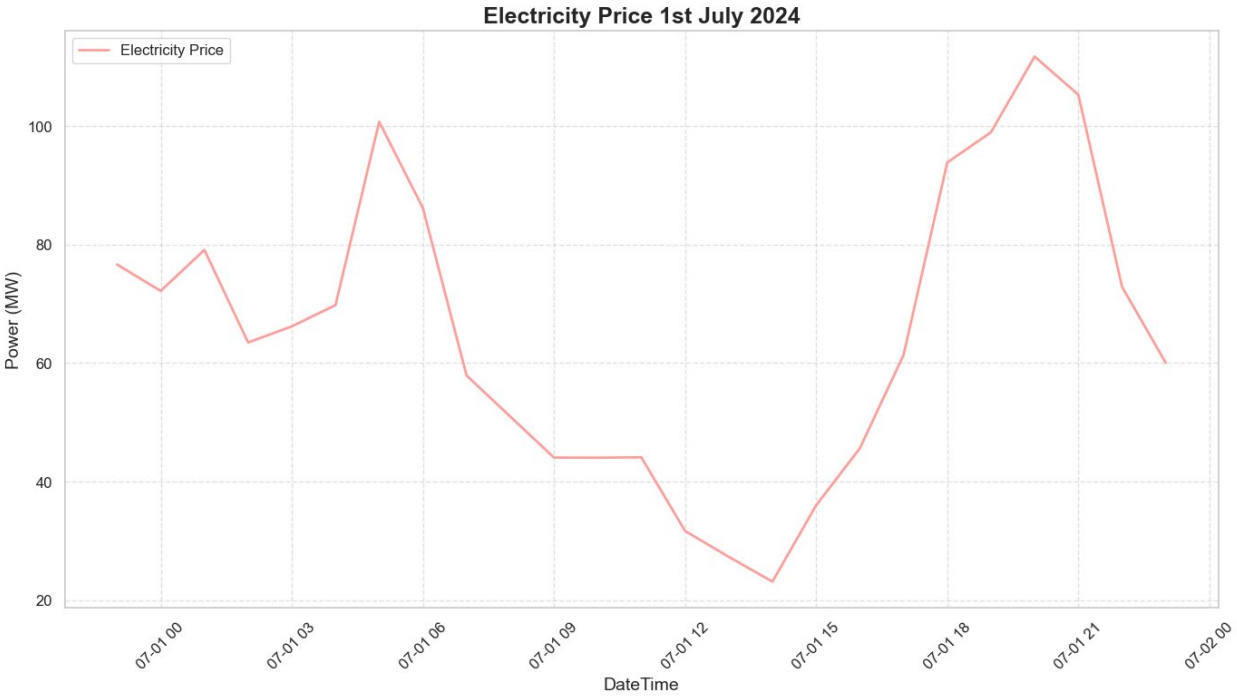

Figure 2.1.5. 1st July

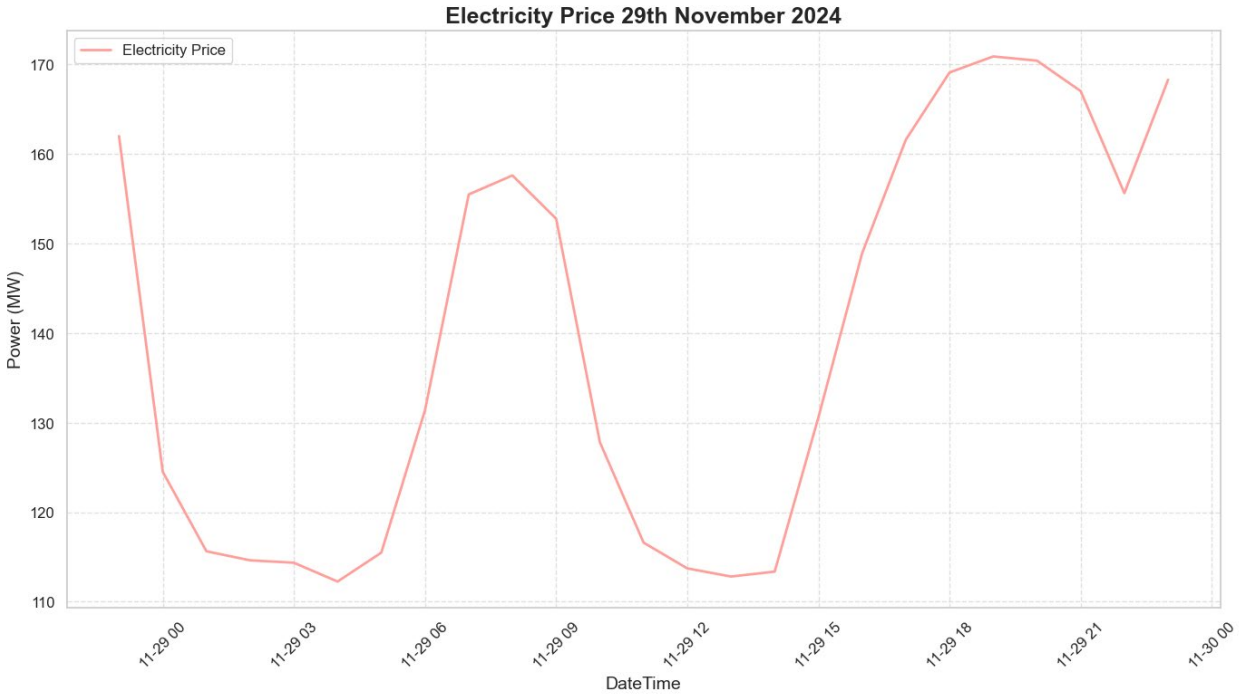

Figure 2.1.6. 29th November

Figure 2.1.7. Electricity Price for 1 Year

From these plots we can notice that electricity prices change rapidly throughout the day. Predicting electricity price is important because it helps energy providers and consumers plan and optimize their costs. Electricity prices change constantly due to factors like demand, weather conditions, fuel costs, and the availability of renewable energy sources like wind and solar. By forecasting these changes, businesses can make informed decisions about when to buy or sell energy, while consumers can adjust their usage to save money. Accurate predictions also ensure a more stable energy market and support efficient grid management, balancing supply and demand effectively.

Factors Influencing Electricity Price Changes

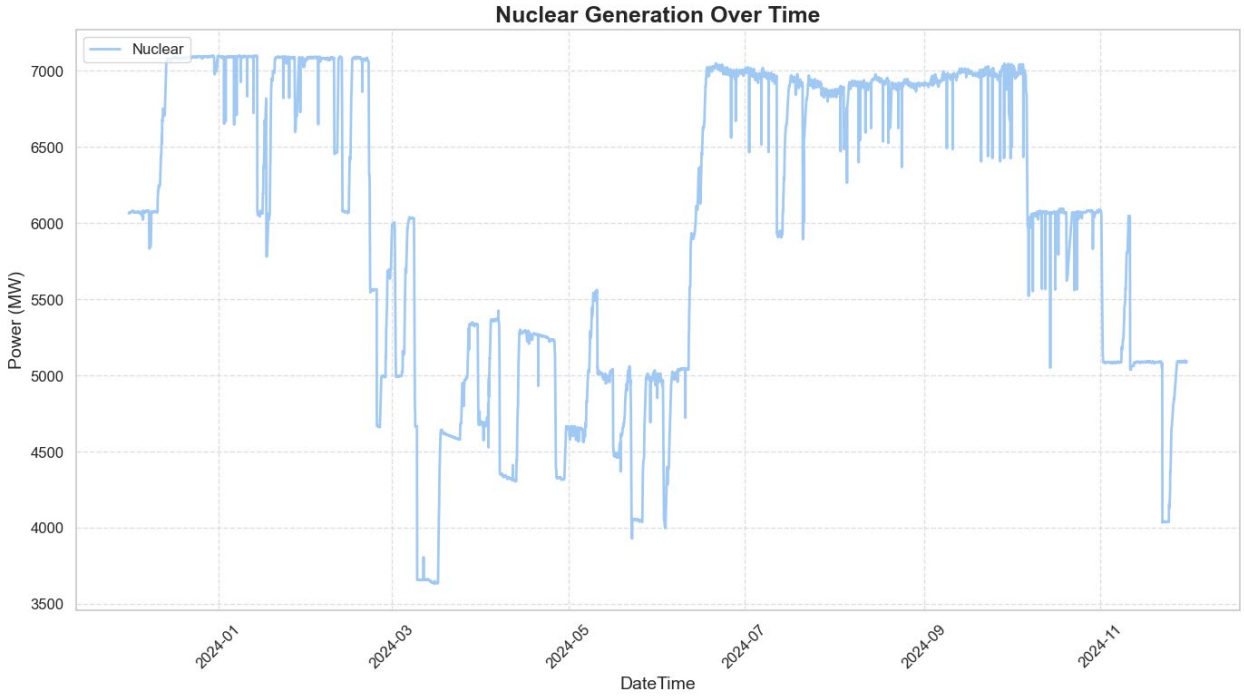

Figure 2.1.8. Nuclear Generation

Nuclear power generation in Spain tends to dip during the summer for a few reasons. One of the main factors is related to the cooling needs of nuclear power plants. These plants depend on water from nearby rivers or the sea to cool their reactors, but during the summer months, the water temperature rises due to the warmer weather. This makes it harder for the plants to use water efficiently for cooling. In some cases, regulations also restrict the temperature of water being released back into the environment to protect ecosystems, which can force plants to lower their output or even temporarily shut down.

Moreover, summer in Spain brings a surge in electricity demand, mainly due to air conditioning use. This often leads to a greater reliance on other energy sources, such as solar, wind, or natural gas, which can be more easily adjusted to meet the higher demand. As a result, nuclear plants may operate at reduced capacity during the summer months to accommodate these factors.

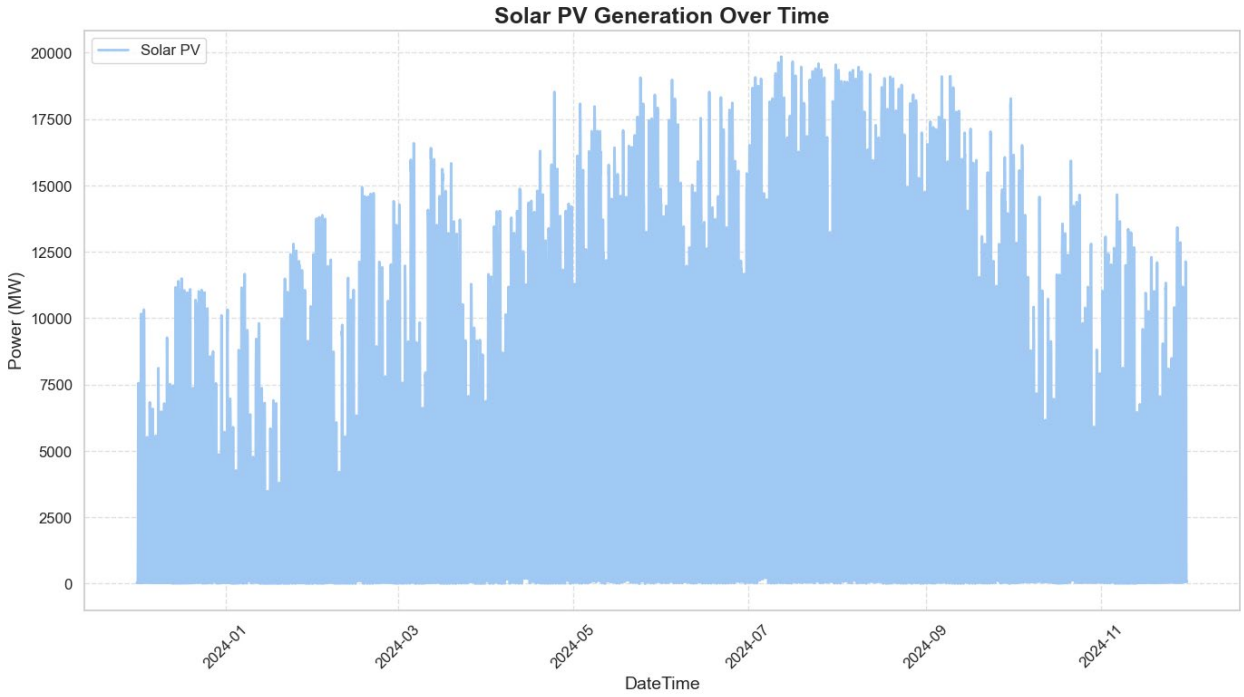

Figure 2.1.9. Solar PV Generation

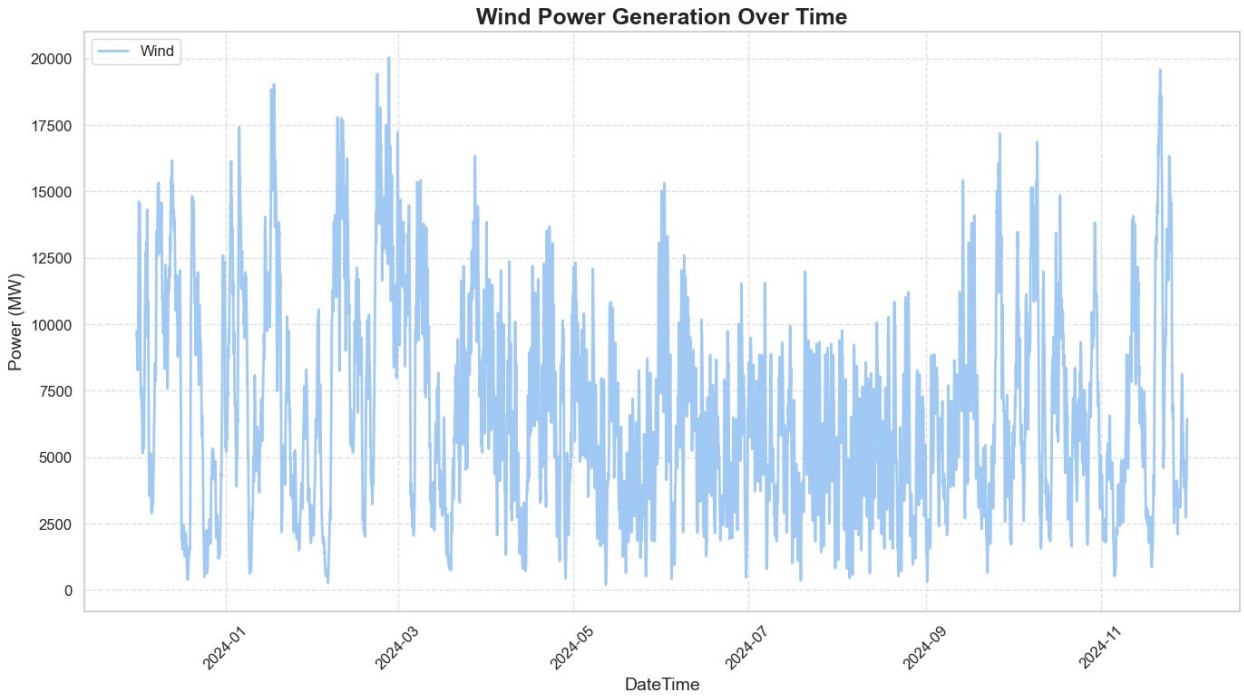

Figure 2.1.10. Wind Power Generation

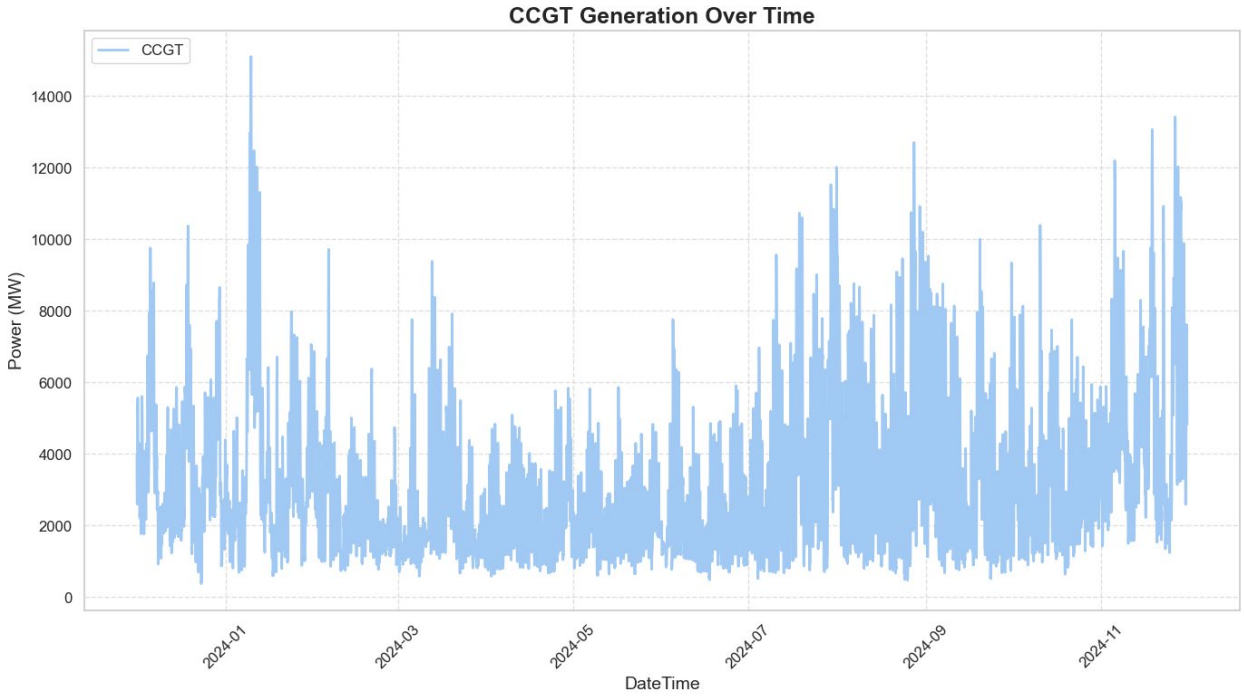

Figure 2.1.11. CCGT Generation

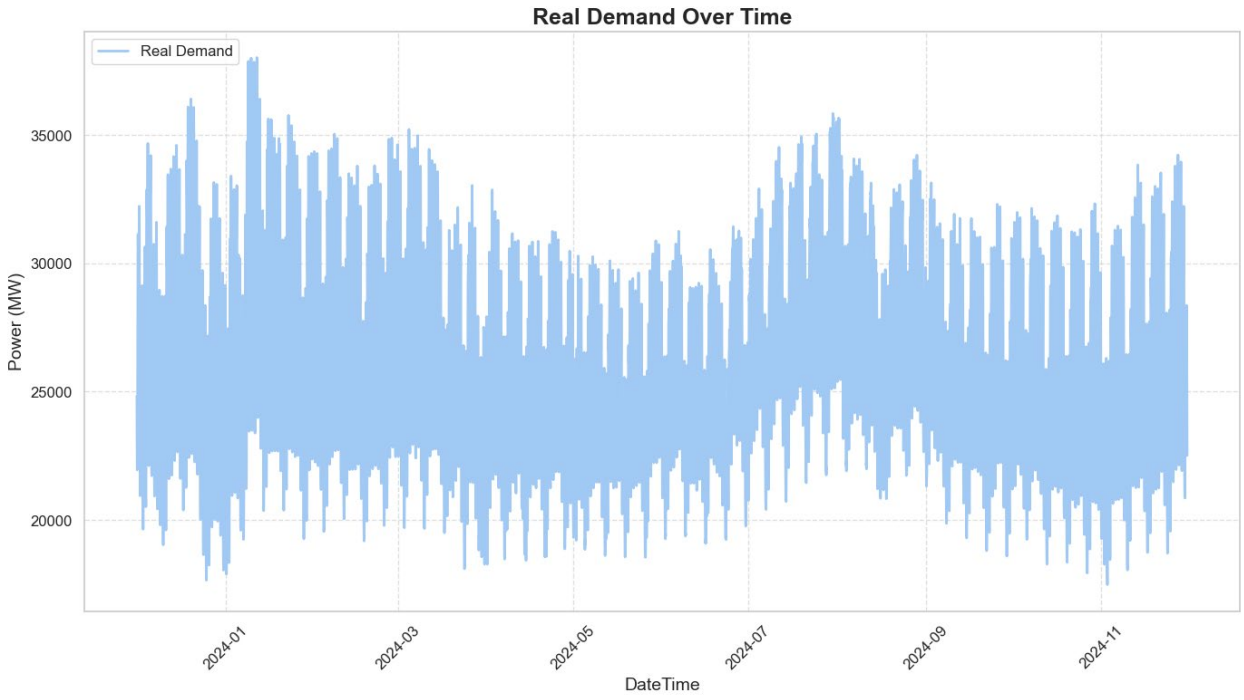

Figure 2.1.12. Real Demand/Consumption

2.2. Understand the Data

2.2.1. Histograms

Another quick way to get a feel of the type of data we are dealing with is to plot a histogram for each numerical attribute. A histogram shows the number of instances (on the vertical axis) that have a given value range (on the horizontal axis). A histogram is a graphical representation of the distribution of numerical data. It is used to summarize the frequency or count of data points within certain ranges or intervals, known as bins.

The data is divided into intervals, or bins. The width of each bin represents a range of values, and the number of data points falling into each bin is counted. The height of each bar in the histogram represents the frequency (or count) of data points within the corresponding bin.

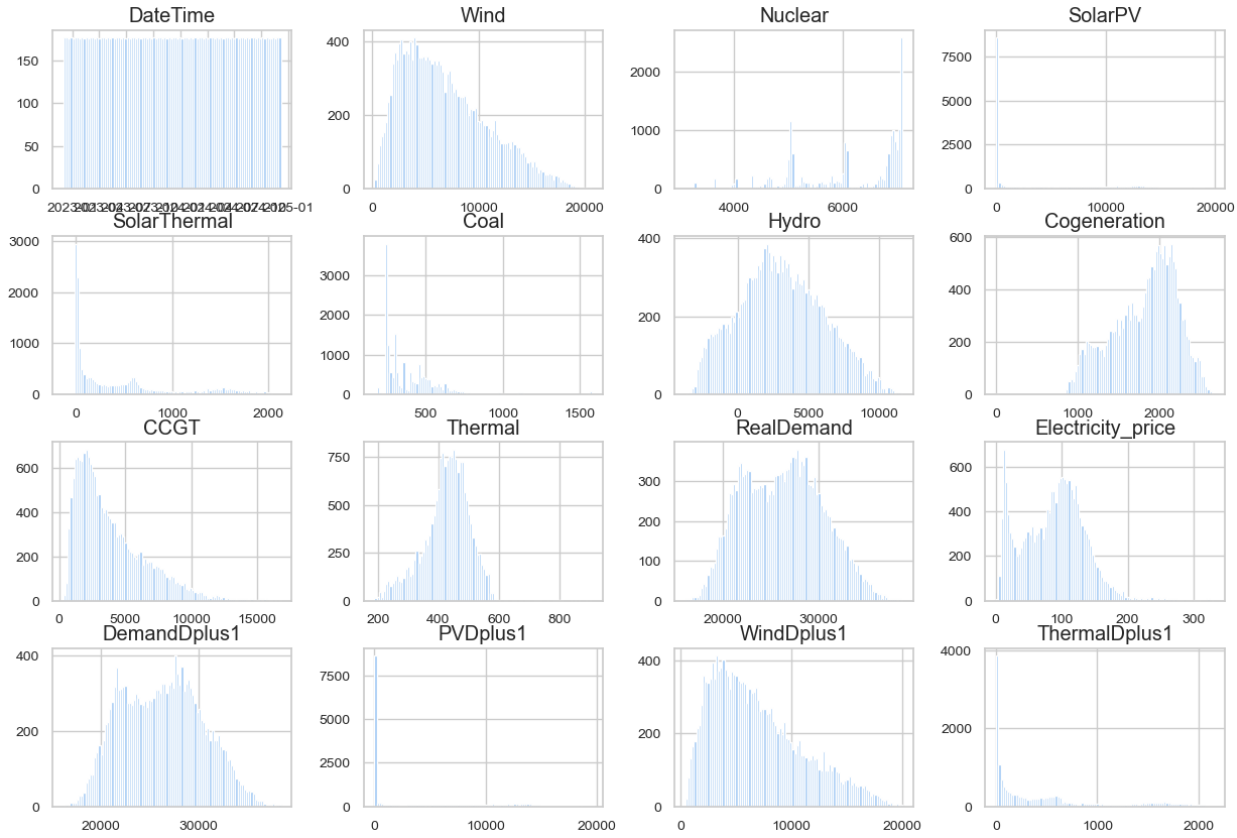

Figure 25.2.1.1. A Histogram for Each Numerical Attribute

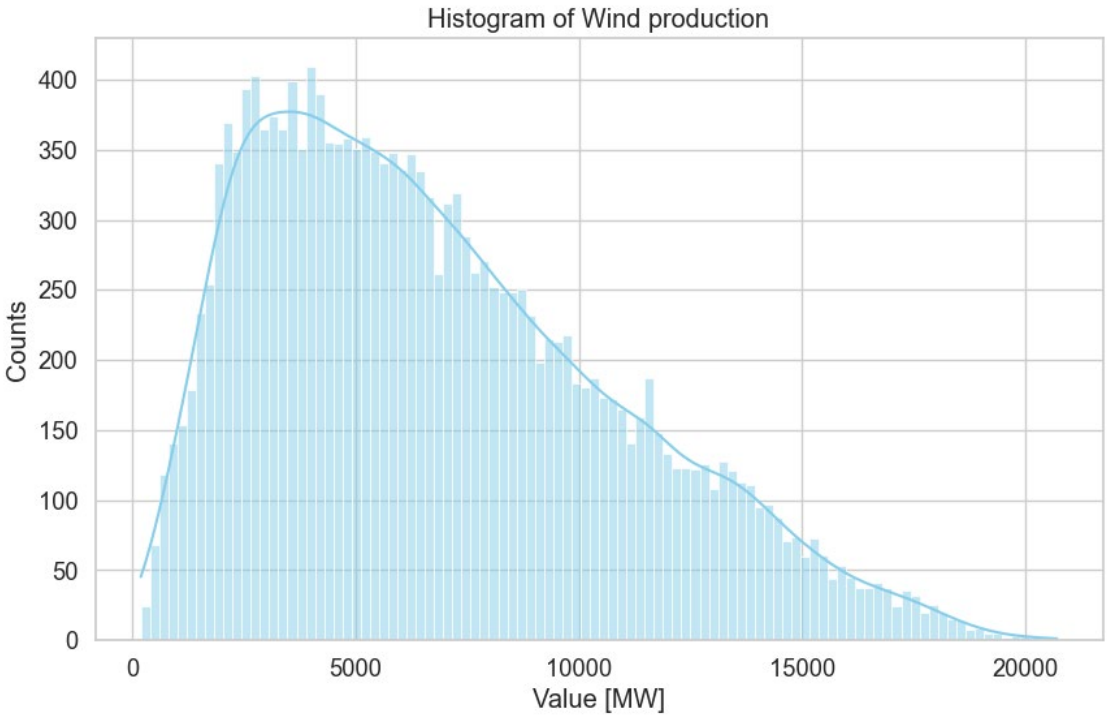

Figure 26.2.1.2. Histogram of Wind Production

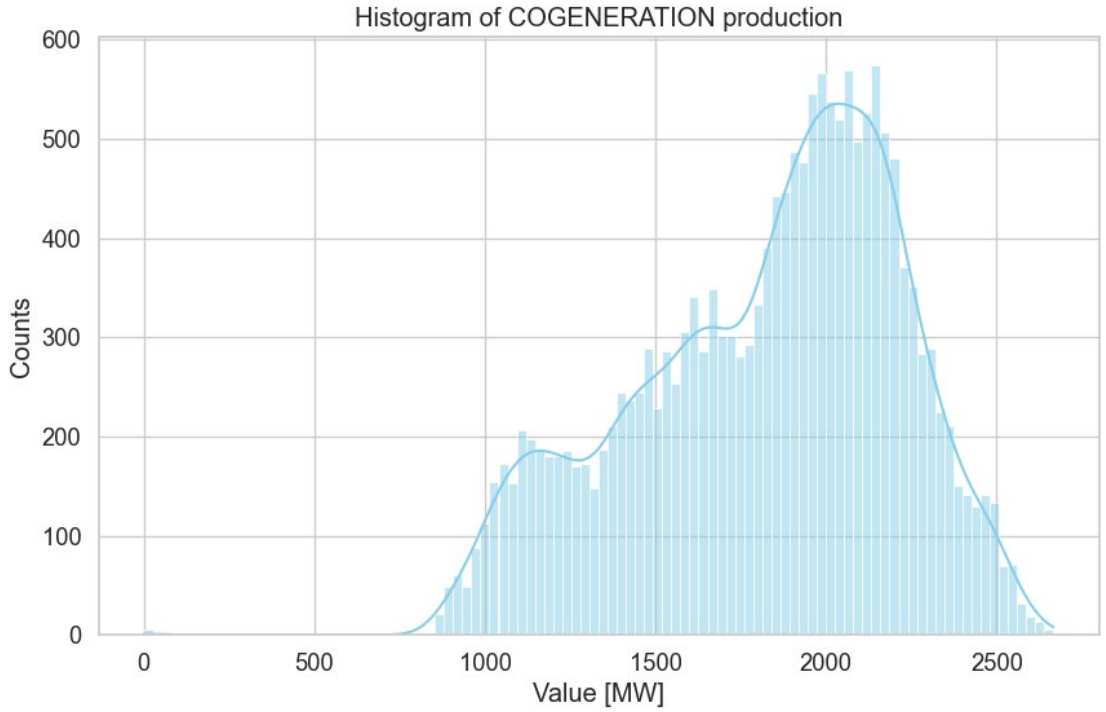

Figure 2.2.1.3. Histogram of Cogeneration Production

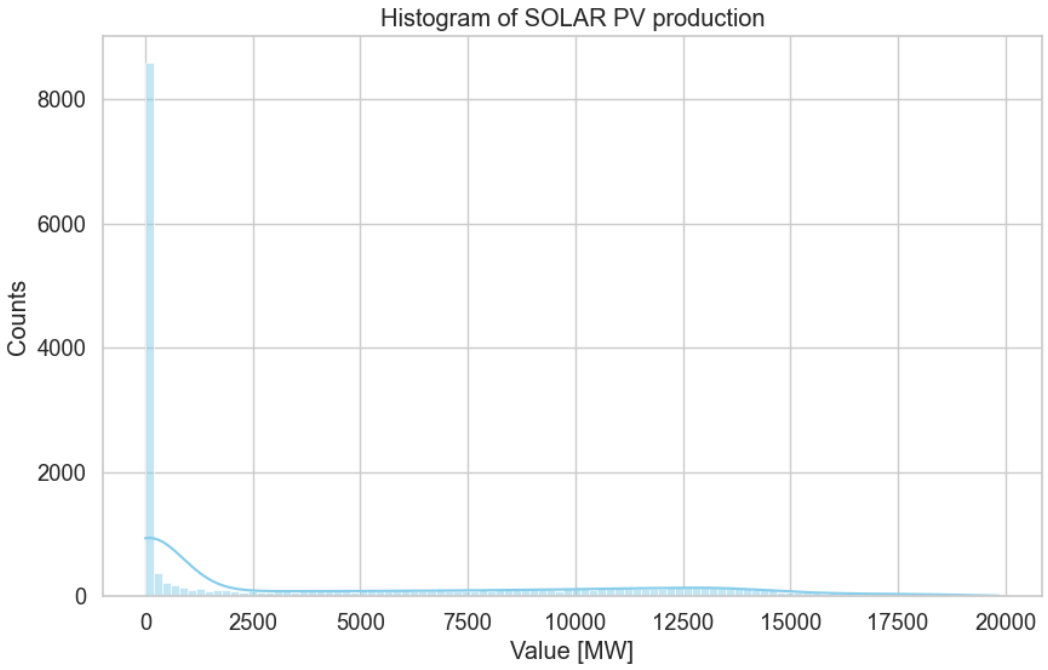

Figure 2.2.1.4. Histogram of Solar PV Generation

Solar PV generation is dependent on sunlight, and during the night or on cloudy days, the generation output can is close to zero or very low. Since the dataset includes periods when the sun isn't shining, the histogram shows a high count of zero generation values.

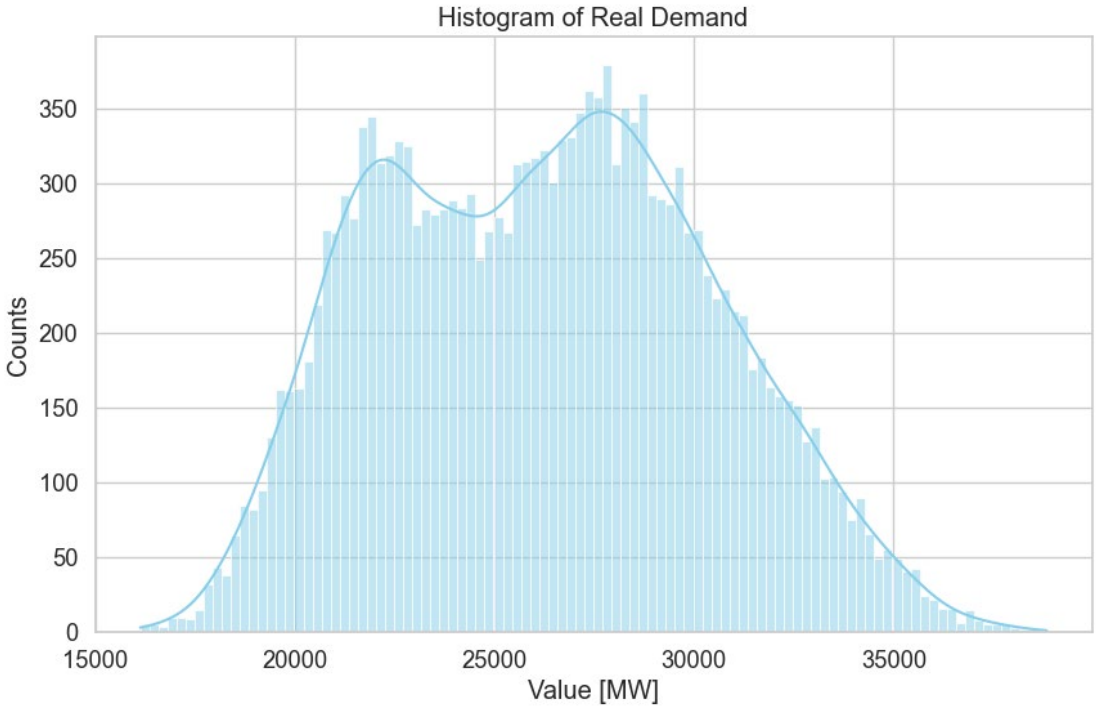

Figure 2.2.1.5. Histogram of Real Demand

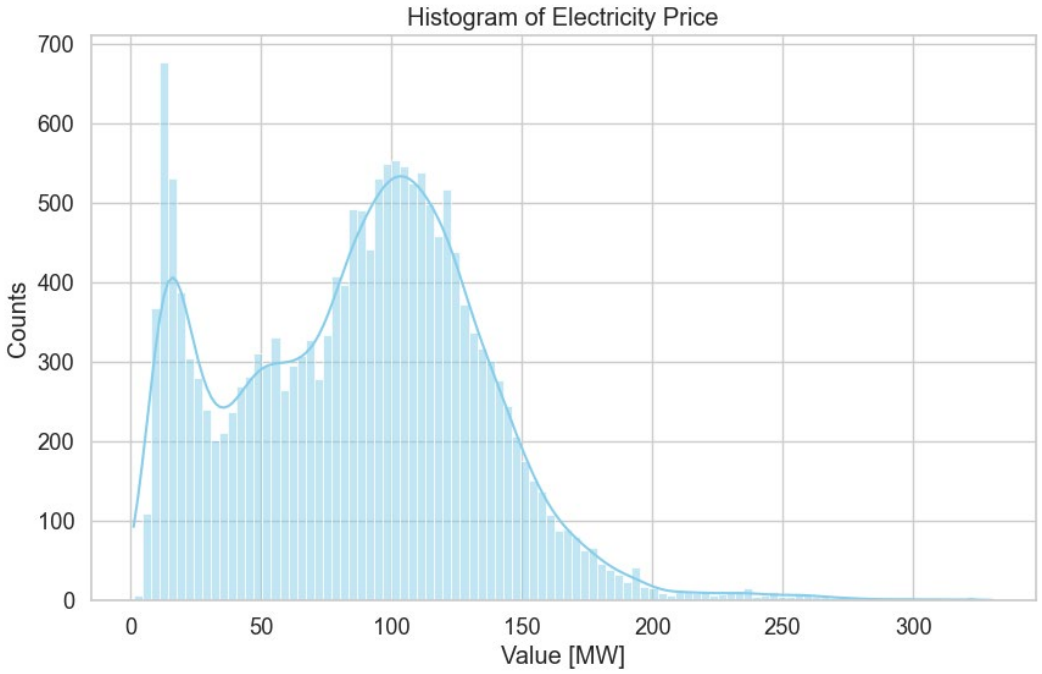

Figure 2.2.1.6. Histogram of Electricity Price

2.2.2. Density Plots

A **density plot**, also known as a kernel density estimate (KDE), is a way of visualizing the distribution of continuous data. It presents the probability density of the data rather than just frequency counts, as shown in a histogram.

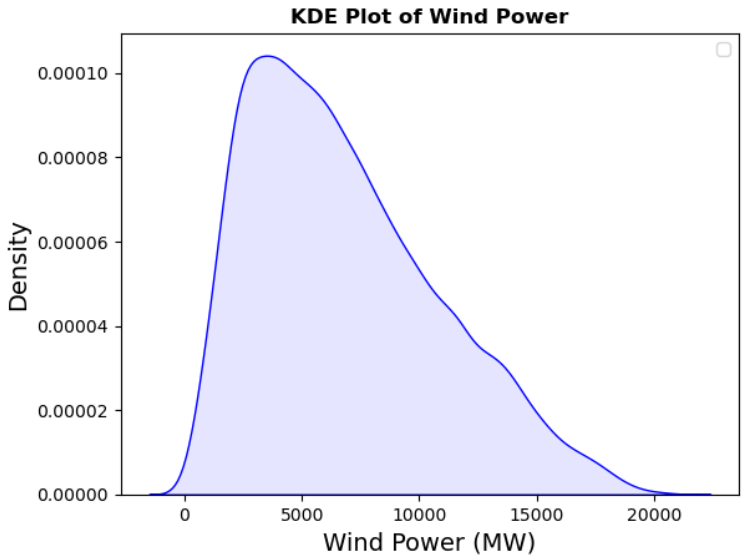

Figure 2.2.2.1. Wind Power

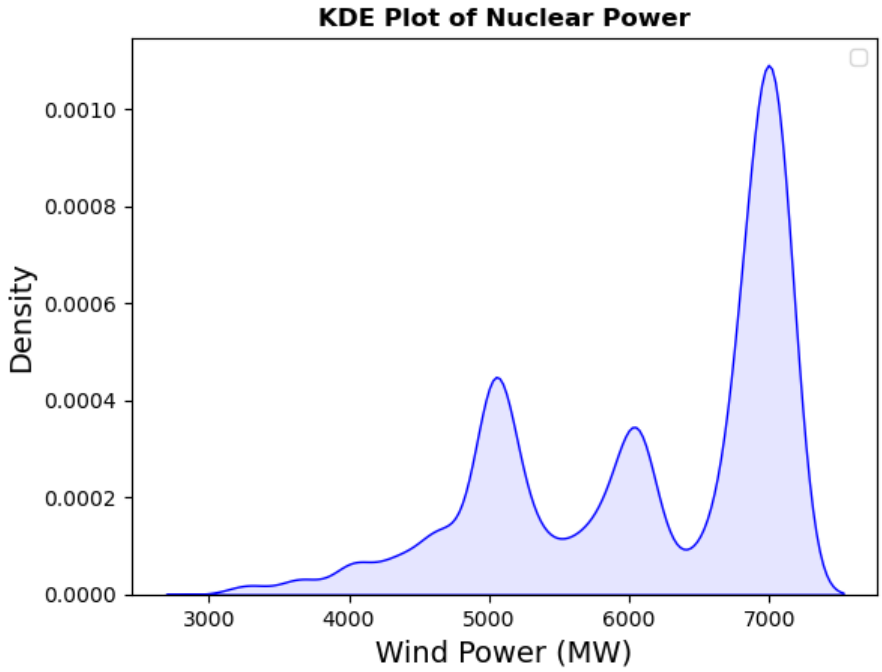

Figure 2.2.2.2. Nuclear Power

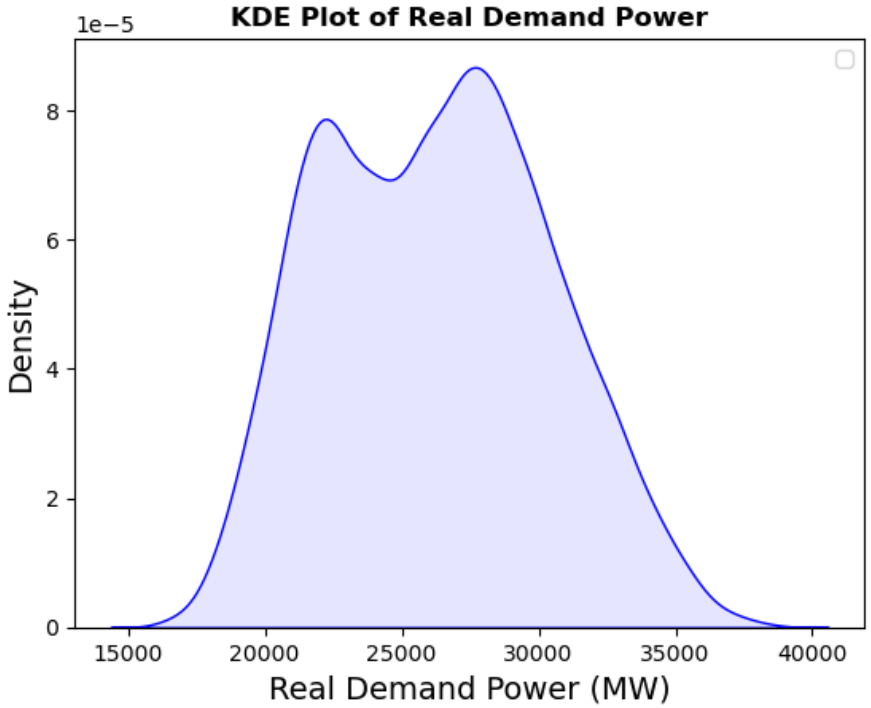

Figure 2.2.2.3. Real Demand

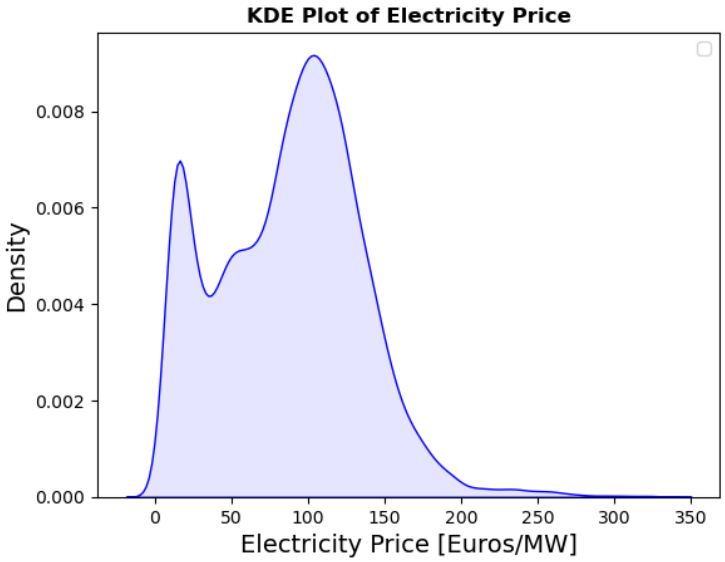

Figure 2.2.2.4. Electricity Price

2.2.3. Boxplots

A boxplot is a graphical tool used to summarize the distribution of a dataset, highlighting its spread and identifying potential outliers. The box represents the interquartile range (IQR), which covers the middle 50% of the data between the first quartile (Q1) and the third quartile (Q3). The whiskers extend to the smallest and largest values within 1.5 times the IQR. Data points that fall outside this range are considered outliers and are plotted as individual dots. Outliers can indicate unusual values, errors, or important variations in the data that may require further investigation.

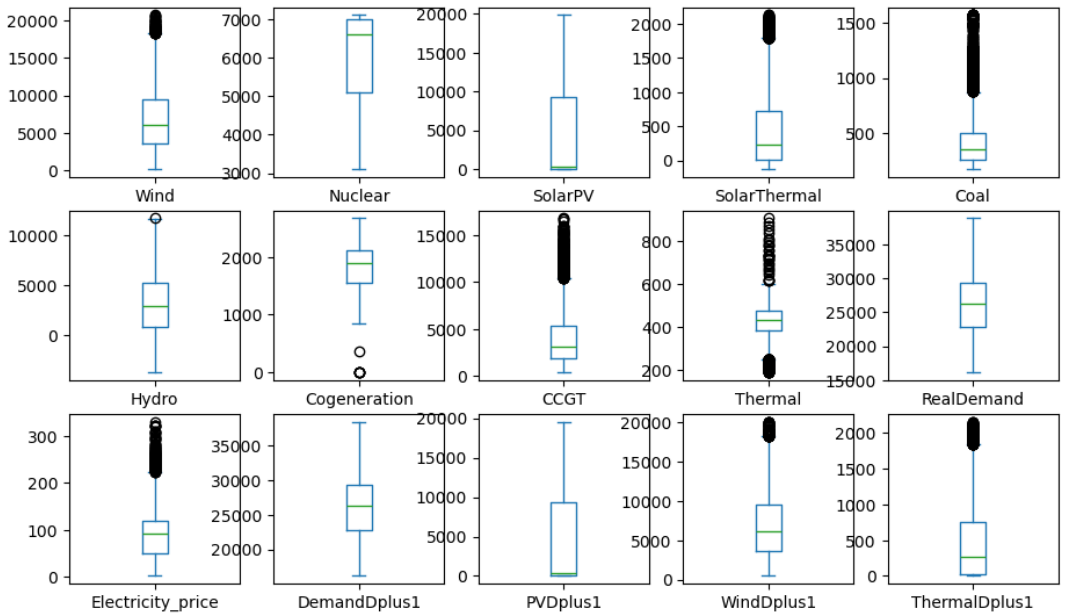

Figure 2.2.3.1. All Boxplots

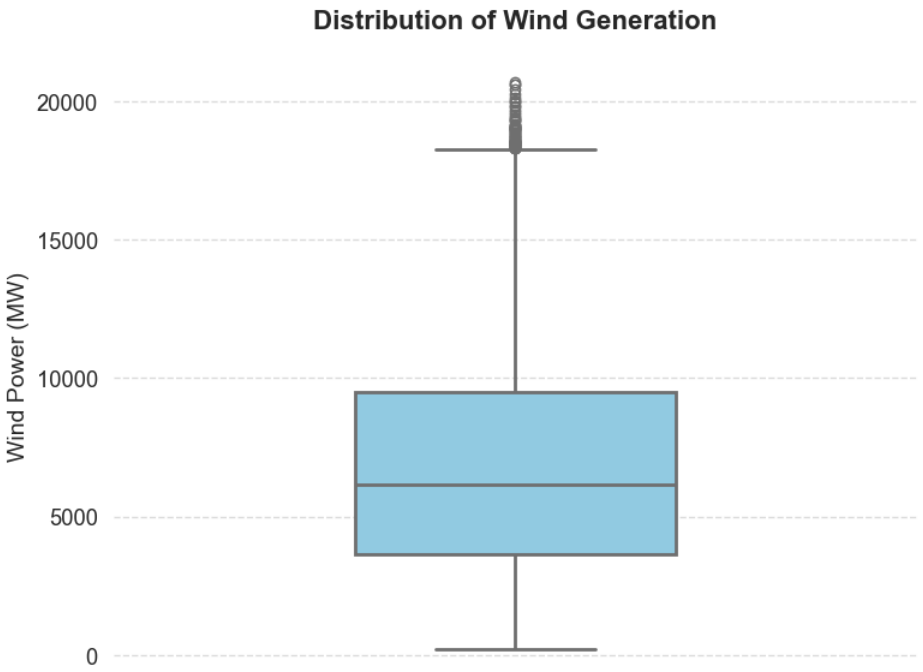

Figure 2.2.3.2. Wind Generation

Removing outliers in wind production data before training a machine learning model might not be a good idea because these "outliers" could represent extreme but real scenarios, such as unusually strong winds or equipment malfunctions. These situations can significantly impact energy production and might be crucial for the model to learn and predict rare but important events. By removing them, we risk losing valuable information that helps the model handle a broader range of real-world conditions, leading to less accurate or robust predictions.

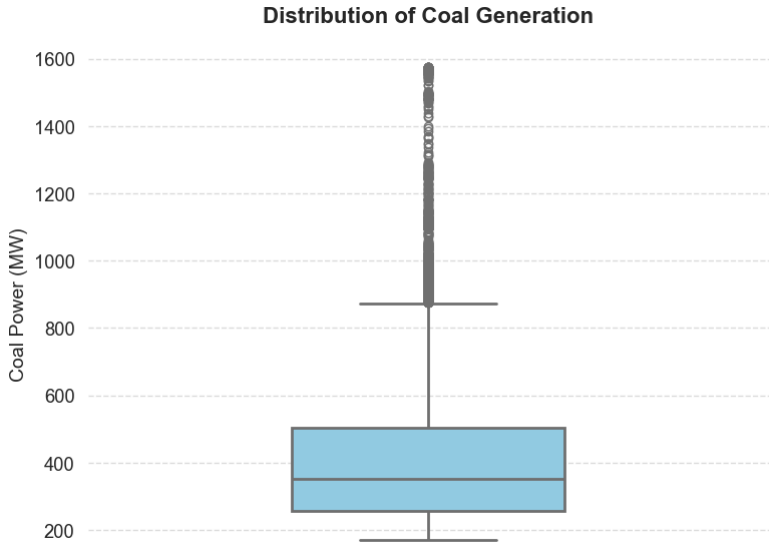

Figure 2.2.3.3. Coal Generation

2.3. Prepare the Data for Machine Learning Algorithm

It's time to prepare the data for our Machine Learning algorithm. Instead of just doing this manually (copying large amounts of code), we should write functions to do that, for several good reasons:

1. We'll be able to easily apply these transformations to any dataset, even when you get a new one in the future. (For using the already trained model in order to predict the price for tomorrow)
2. We'll build a collection of reusable transformation functions that can be helpful in future projects.
3. It will also allow us to experiment with different transformations and find the best combination for your model.

Functions used to prepare the data:

```
def weekend(dataset):
    dataset.loc[:,'day_of_week'] = dataset['DateTime'].dt.weekday
    dataset.loc[:,'hour'] = dataset['DateTime'].dt.hour
    return dataset
```

✓ 0.0s

Figure 2.3.1. Weekend

Day of the week or whether it's a weekend influence electricity price. On weekdays, electricity demand is typically higher because businesses, factories, and offices are running, along with residential consumption. This higher demand can lead to higher electricity prices, especially during peak hours (usually in the morning and evening).

```
def one_hot_encoding(dataset):
    dataset = pd.get_dummies(dataset, columns=['day_of_week'], prefix='day_of_week', dtype=float)
    dataset = pd.get_dummies(dataset, columns=['hour'], prefix='hour', dtype=float)
    return dataset
```

✓ 0.0s

Figure 2.3.2. One Hot Encoding

One-hot encoding is a method used to transform categorical data into a format that can be easily used by machine learning algorithms. It converts each category of a variable into a new binary (0 or 1) column. Instead of representing a category with a single number or string, each category gets its own column, where a "1" indicates the presence of that category, and "0" indicates its absence.

The idea behind this approach is to create a new dummy feature for each unique value in the nominal feature column. Here, we would convert the DateTime feature into new features: day_of_week and hour. Binary values can then be used to indicate the particular day for example, a Monday example can be encoded as day_of_week1=1, day_of_week2=0, day_of_week3=0, etc.

```python
def lags(dataset):
    dataset['Electricity Price Lag_1_day'] = dataset['Electricity_price'].shift(24)
    dataset['Electricity Price Lag_2_days'] = dataset['Electricity_price'].shift(48)
    dataset['Electricity Price Lag_7_days'] = dataset['Electricity_price'].shift(168)
    return dataset
```
✓ 0.0s

Figure 2.3.3. Lags Of Electricity Price

The lag of electricity price refers to using past electricity prices as features (inputs) in a machine learning model to predict future electricity prices. The price of electricity at any given time is often correlated with its past values, especially in markets where prices are based on demand-supply dynamics, weather conditions, and other factors that exhibit temporal patterns. If electricity prices were high yesterday due to high demand, they might remain elevated for a short period, so knowing the price from the previous day (lag) helps the model understand these temporal dependencies.

```python
def remove_nan(dataset):
    dataset.dropna(inplace=True)
    return dataset
```
✓ 0.0s

Figure 2.3.4. Remove Incomplete or Missing Data

Machine learning algorithms can not work with missing features.

```python
def holiday(dataset):
    es_holidays_2023 = holidays.Spain(years=2023, prov='CT')
    es_holidays_2024 = holidays.Spain(years=2024, prov='CT')
    all_holidays = {**es_holidays_2023, **es_holidays_2024}
    dataset['is_holiday'] = pd.Series(dataset.index.date, index=dataset.index).isin(all_holidays.keys()).astype(int)
    return dataset
```
✓ 0.0s

Figure 2.3.5. Is it a Holliday or Not?

Holidays influence electricity prices. On holidays, many businesses, factories, and offices are closed or operate at reduced capacity, leading to **lower overall demand** for electricity compared to normal weekdays. While residential demand might stay relatively stable, there can be slight increases or decreases depending on the holiday. For example, during Christmas or New Year, people may stay home more, using more electricity for heating, cooking, or entertainment, which could increase residential demand.

2.4. Irrelevant features

A machine learning system is only as good as the data it's trained on. If the training data lacks relevant features or contains too many irrelevant ones, the system won't learn effectively. That's why a important step in any successful machine learning project is creating a good set of features for training—this process is known as feature engineering.

Feature engineering involves three key steps:

1. Feature selection: Choosing the most useful features from the existing data to train the model.
2. Feature extraction: Combining or transforming existing features to create more meaningful ones. Dimensionality reduction techniques, like PCA, can help with this.
3. Creating new features: Gathering additional data to introduce entirely new features that improve the model.

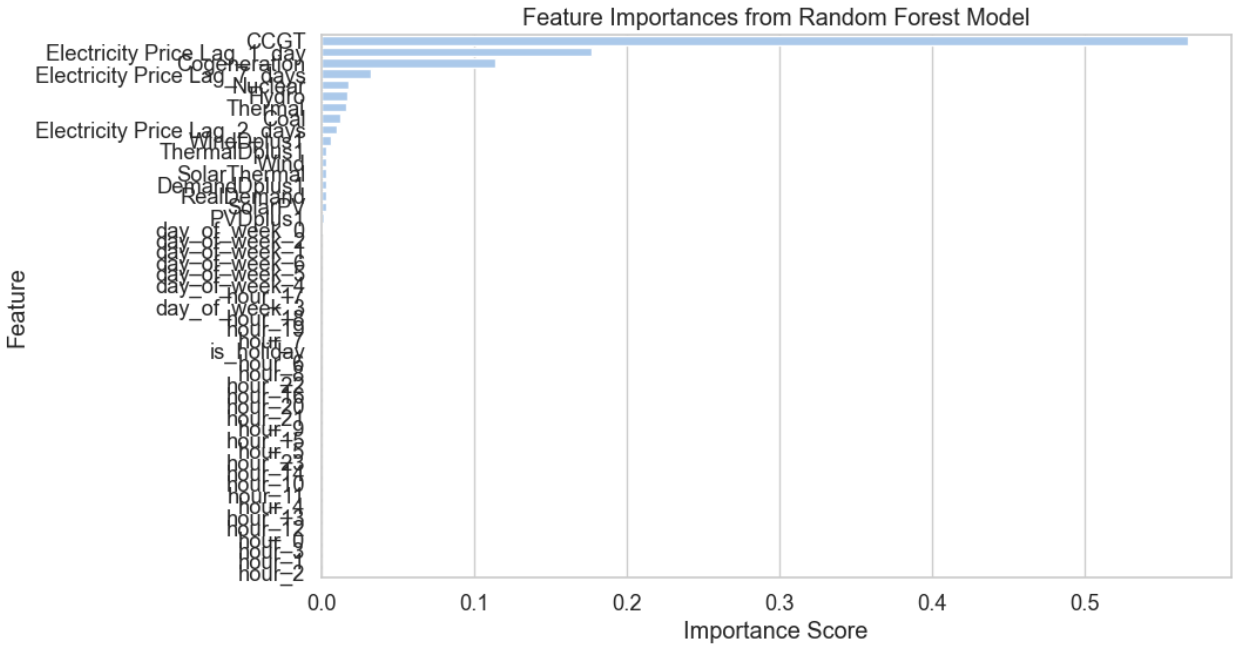

Figure 2.4.1. Most Important Features of our ML model

As we can seen the most relevant feature is Combined Cycle Gas Turbine. CCGT power plants significantly influence electricity prices due to their unique role in the energy market. Electricity markets often operate on a marginal pricing system, where the price is set by the last power plant (the marginal generator) needed to meet demand. CCGT plants often act as these marginal generators because they are flexible, efficient, and can quickly ramp up or down to balance supply and demand.

If gas prices increase, the cost of running CCGT plants rises, which directly drives up electricity prices. The price of natural gas is highly volatile, influenced by geopolitical events, supply disruptions, and seasonal demand. Since CCGT plants rely on natural gas, fluctuations in gas prices significantly impact electricity prices. When gas prices surge, CCGT generation becomes more expensive, raising the marginal cost of electricity.

2.5. Select and Train a Machine Learning Model

At last! We've defined the problem, gathered and explored the data, split it into a training set and test set, cleaned and prepared everything for our machine learning algorithms. Now, we're all set to choose and train a machine learning model! The good news is that thanks to all these previous steps, things are now going to be much simpler than we might think.

One way to assess the performance of your Decision Tree model is by using the train_test_split function to break the training set into a smaller training set and a validation set. We can then train the model on the smaller training set and evaluate it on the validation set. While this approach is simple and works well, it does require a bit of extra effort.

A more efficient alternative is to use K-fold cross-validation in Scikit-Learn. This method divides the training set into 5 random subsets, or "folds" (We are defining the number of folds). The model is then trained and evaluated 5 times, each time using a different fold for validation and the remaining 4 folds for training. This approach provides a more robust evaluation, and the result will be an array of 5 evaluation scores.

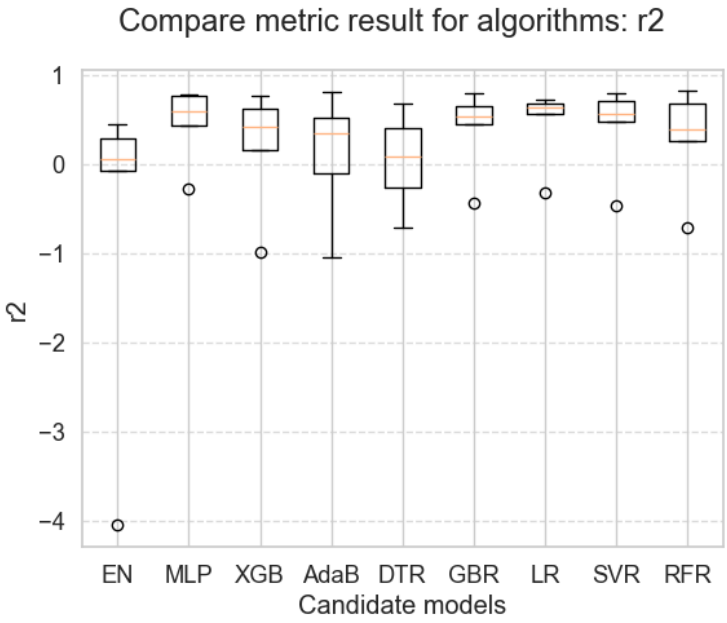

Figure 2.5.1. R2

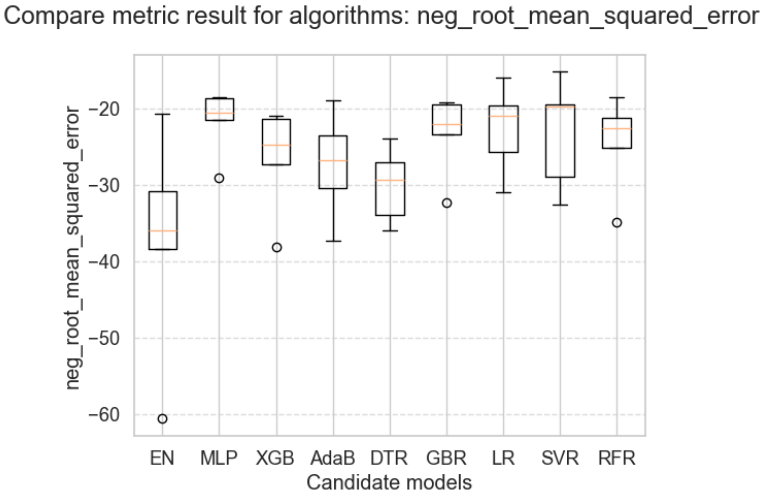

Figure 2.5.2. RMSE

We use negative RMSE (Root Mean Squared Error) when comparing different machine learning models primarily because of how the scoring mechanism works in some evaluation frameworks, like Scikit-Learn's model selection tools cross_val_score function expects a higher score to indicate a better model.

Random Forest look very promising! Before we dive too deep into Random Forests, it's a good idea to experiment with several other models from different types of machine learning algorithms—like different Support Vector Machines and so on.

2.6. Fine-Tune Our Model

Now that we've shortlisted some promising models, it's time to fine-tune them. One approach is to manually adjust the hyperparameters, tweaking them bit by bit until you find a good combination. However, this can be really time-consuming, and you might not have the time to explore many combinations.

A better option is to let Scikit-Learn's GridSearchCV do the heavy lifting for us. All we need to do is specify which hyperparameters we want to tune and what range of values you'd like it to try, and GridSearchCV will evaluate every possible combination using cross-validation.

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
[CV 1/5] END ..................n_estimators=100;, score=0.726 total time=   0.6s
[CV 2/5] END ..................n_estimators=100;, score=0.874 total time=   1.4s
[CV 3/5] END ..................n_estimators=100;, score=0.823 total time=   2.1s
[CV 4/5] END ..................n_estimators=100;, score=0.885 total time=   3.2s
[CV 5/5] END ..................n_estimators=100;, score=0.672 total time=   4.2s
[CV 1/5] END ..................n_estimators=500;, score=0.735 total time=   3.5s
[CV 2/5] END ..................n_estimators=500;, score=0.879 total time=   7.1s
[CV 3/5] END ..................n_estimators=500;, score=0.827 total time=  11.4s
[CV 4/5] END ..................n_estimators=500;, score=0.886 total time=  15.9s
[CV 5/5] END ..................n_estimators=500;, score=0.681 total time=  21.0s
Best result: 0.801496 using the following hyperparameters {'n_estimators': 500}
```

Figure 7. Adjusting HyperParameters

GridSearchCV works well when you're exploring a small number of hyperparameter combinations, but when the search space is large, *RandomizedSearchCV* is often a better choice. This class works similarly to *GridSearchCV*, but instead of evaluating every possible combination, it randomly selects a specific number of hyperparameter combinations to try.

There are two key advantages to using *RandomizedSearchCV*:

1. **Larger Exploration**: For example, if we let the randomized search run for 1,000 iterations, it will explore 1,000 different combinations of hyperparameters, giving you a broader range of options than a grid search with only a few values per hyperparameter.

2. **Better Control Over Resources**: We can control how much computing power we want to allocate to the search by simply setting the number of iterations. This makes it easier to manage your computational budget while still exploring a large hyperparameter space.

2.7. Evaluate Our System on the Test Set

After fine-tuning your models and achieving a satisfactory performance, it's time to evaluate the final model on the test set. There is nothing special about this process, we just get the predictors and the labels from our test set, and evaluate the final model on the test set!

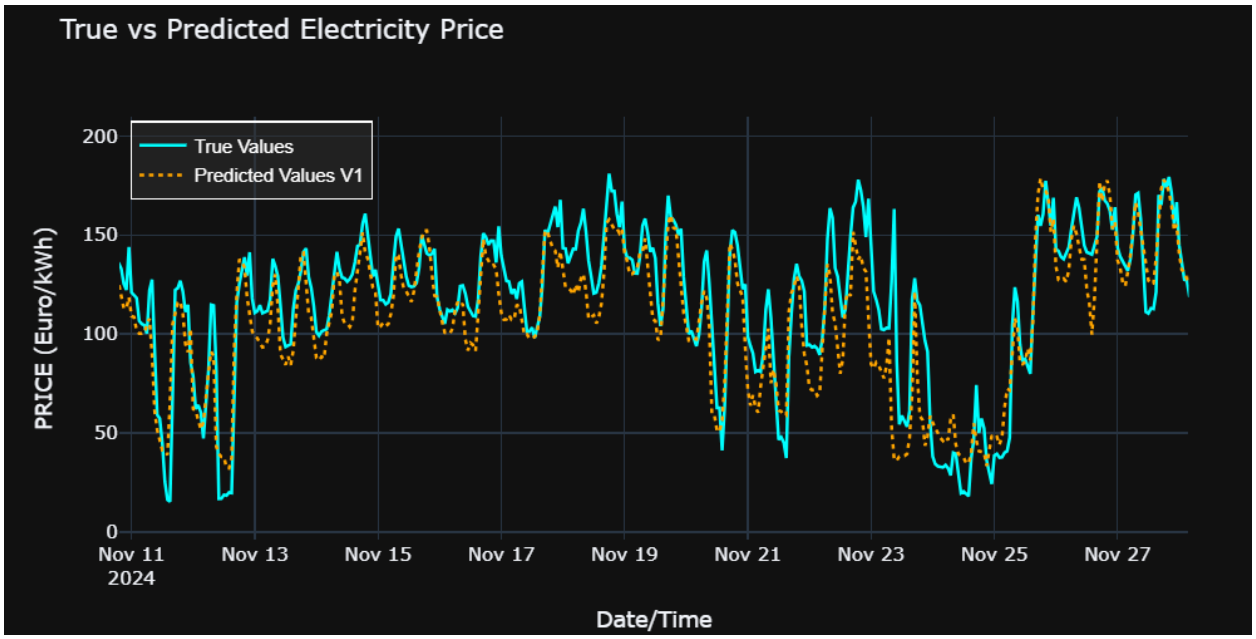

Figure 2.7.1. True VS Predicted Electricity Prices

2.8. Predicting Electricity Price for Tomorrow

Now that we have a trained machine learning model, we can use it to predict electricity prices for tomorrow. Thanks to the functions we've already created for data preparation, we can easily transform and process the new data as it arrives. These functions allow us to clean and format the input data just like we did during training. Once the data is ready, we can feed it into the trained model and get accurate predictions for the upcoming day's electricity prices. This setup ensures that our model can be used seamlessly for future predictions with minimal additional effort.

Figure 2.8.1. Real Electricity Price from ESIOS

We have two approaches to predict electricity prices using real data from ESIOS:

1. **Using 8 days of all available data**: In this approach, we use 8 days of all relevant data, including electricity prices, demand, weather, and any other available factors.
2. **Using 8 days of electricity price data and 1 day of other data**: In this approach, we still use the past 8 days of electricity price data, but we add only 1 day of other data, such as demand or forecasts. If some of the non-price data isn't available for the most recent day, we use the available data, which might mean using less than 1 full day of non-price data. This method is useful for accounting for electricity price lags while still keeping the data as complete as possible.

Both approaches use real electricity price data from ESIOS, but the second approach adjusts depending on the availability of additional non-price data.

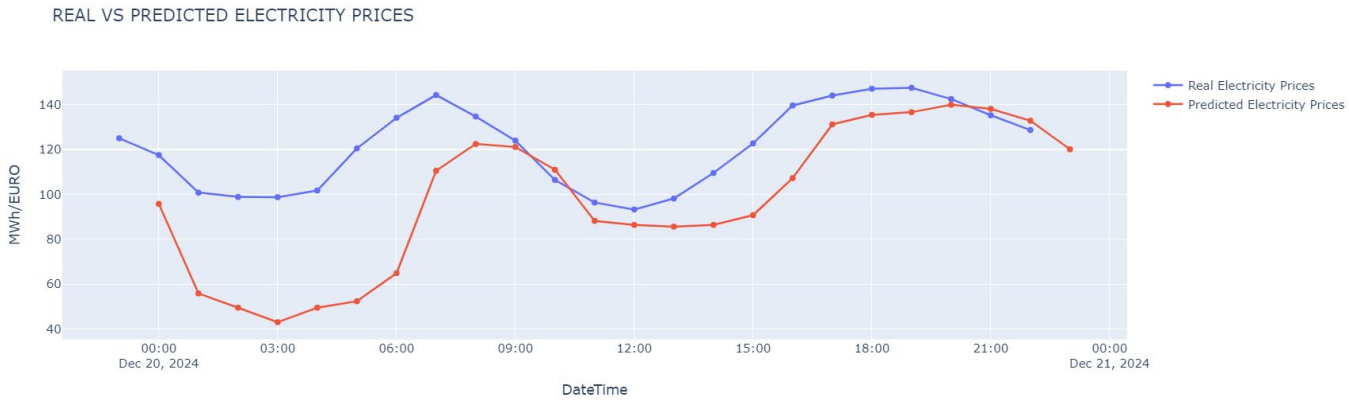

Figure 2.8.2. Real vs Predicted Electricity Prices for Tomorrow

3. Unsupervised Learning Techniques

While most machine learning applications today focus on supervised learning (which is where most investments are directed), the reality is that the majority of available data is actually unlabeled. In other words, we often have the input features (X), but we don't have the corresponding labels (y). Clustering (or cluster analysis) is a technique that allows us to find groups of similar objects that are more related to each other than to objects in other groups.

Imagine we want to create a system that takes pictures of items on a production line and detects which ones are defective. You could easily set up a system that automatically takes thousands of pictures every day, and within a few weeks, you'd have a large dataset. But here's the problem—there are no labels!

To train a standard binary classifier that can tell if an item is defective or not, you'd need to label each picture as "defective" or "normal." This task usually requires experts to manually review and label the images, which is time-consuming and expensive. As a result, we end up with a small labeled dataset, and your model's performance will likely be disappointing. Plus, if the product changes, you'd need to repeat the whole process.

Wouldn’t it be great if the algorithm could use the unlabeled data without requiring human labeling? That’s where *unsupervised learning* comes in.

Unsupervised learning tasks:

- **Clustering**: This groups similar instances together into clusters. It’s useful for data analysis, customer segmentation, recommender systems, search engines, image segmentation, and more.
- **Anomaly detection**: The goal here is to understand what "normal" data looks like, so you can detect outliers or abnormalities, such as defective items or new trends in a time series.
- **Density estimation**: This involves estimating the probability distribution of the data. It’s often used in anomaly detection, where instances in low-density areas are likely anomalies. It's also useful for data analysis and visualization.

3.1. Clustering

Since we’re analyzing electricity prices over the past few months, as we look at the data, we notice that the prices vary throughout the day and week, with certain patterns emerging on specific days or during particular seasons. We might not know exactly why some days have higher prices, but we can clearly see groups of days with similar price trends.

For instance, prices tend to be higher on weekdays during summer afternoons, while weekends or winter mornings may show lower prices. While we may need an expert to explain the exact reasons behind these price fluctuations, we don’t need to be an expert to identify these groups or patterns. This

is where *clustering* comes in: it's the task of grouping similar instances—in this case, similar price patterns—into clusters, helping us identify when prices are likely to follow a similar trend in the future.

3.1.1. Hierarchical (Agglomerative) clustering

Agglomerative clustering works by building a hierarchy of clusters from the bottom up. Imagine many tiny bubbles floating on water, each separate at first. Over time, these bubbles gradually join together to form larger clusters, until eventually, there's just one big group of bubbles.

In the same way, agglomerative clustering starts with each individual data point as its own cluster. Then, at each step, it merges the two closest clusters together. This process continues iteratively, with clusters getting larger and larger, until all the data points belong to a single cluster. The result is a tree-like structure called a dendrogram, which shows how clusters were merged at each step.

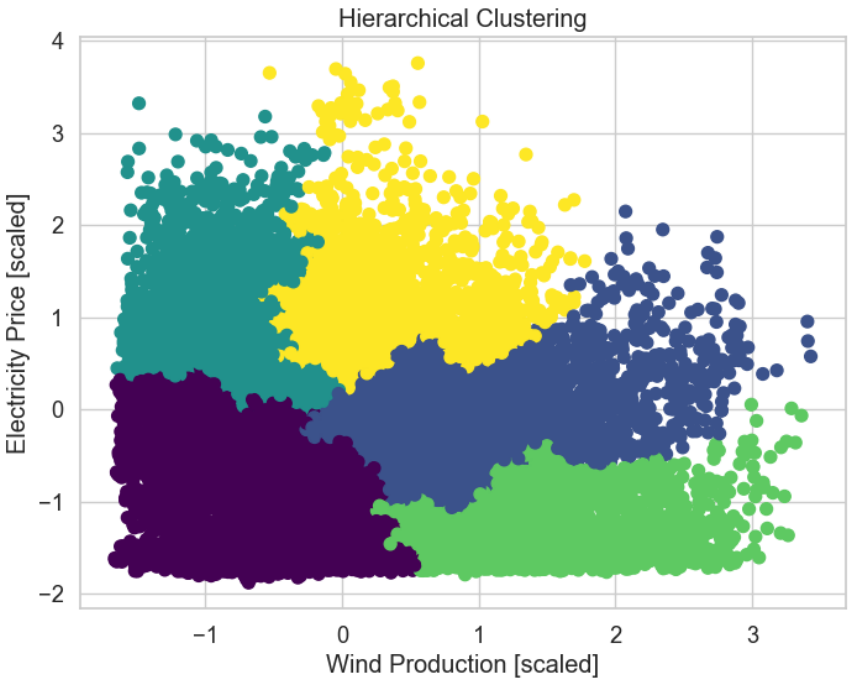

Figure 3.1.1.1. Wind/EP

In this scatter plot, we observe the results of hierarchical clustering applied to scaled wind production (x-axis) and scaled electricity price (y-axis). The data points are grouped into **five clusters**, each represented by a different color:

- **Yellow Cluster**: Located in the top-center. High prices and lower wind production. Suggests that low wind production is associated with higher electricity prices, likely due to supply shortages.
- **Cyan Cluster**: Moderate prices and low wind production. This area likely reflects average market conditions with balanced wind production and pricing.

- **Dark Purple Cluster**: Located in the bottom-left. Low prices and low wind production. Represents stable low prices despite low wind production, possibly influenced by other energy sources or low demand.
- **Blue Cluster**: Moderate prices with slightly higher wind production. This area likely reflects average market conditions with balanced wind production and pricing.
- **Green Cluster**: Located in the bottom-right. Low prices and high wind production. Indicates that high wind production contributes to lower electricity prices, likely due to surplus energy availability.

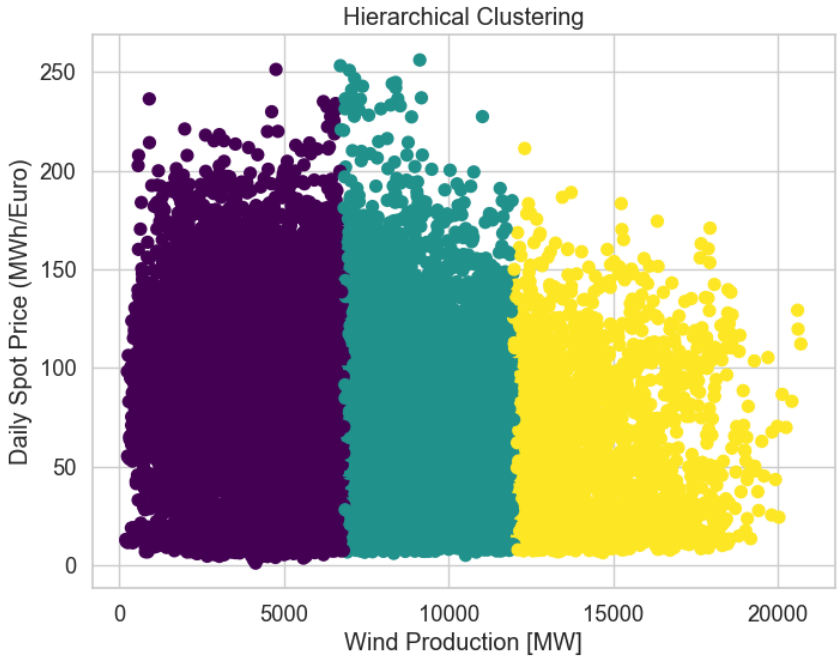

Figure 3.1.1.2. Wind/EP Non-Scaled

This scatter plot depicts hierarchical clustering results on nuclear production (MW) and daily spot price (MWh/Euro). The data is segmented into three clusters, each represented by a unique color (yellow, green, and purple).

The clusters are **horizontally aligned**, showing clear divisions in nuclear production levels:

- **Yellow Cluster**: Higher wind production (12000–20000 MW).
- **Green Cluster**: 5500–12000 MW.
- **Purple Cluster**: Low nuclear production (0–7000 MW)

While nuclear production determines horizontal positioning, the vertical spread (daily spot price) remains somewhat similar across clusters, with lower prices (below 50 €/MWh) being more frequent in higher production cluster (yellow). Higher wind production tends to stabilize prices, keeping them relatively low.

3.1.2. K-Means

The k-means algorithm is widely used because it's both simple to implement and computationally efficient compared to many other clustering methods. This combination of ease and speed makes it especially popular for clustering tasks. K-means falls into the category of prototype-based clustering, where each cluster is represented by a "prototype." For data with continuous features, this prototype is usually the centroid, which is the average position of all points within the cluster.

One key strength of k-means is its ability to identify clusters that are roughly spherical in shape. However, a major limitation is that you must specify the number of clusters (k) beforehand. If you choose an inappropriate value for k, the clustering results might be poor.

To address this, there are techniques like the elbow method and silhouette analysis that help evaluate the quality of clustering results and guide you in selecting the optimal number of clusters.

Now, the next question is, *how do we measure similarity between objects*? We can define similarity as the opposite of distance, and a commonly used distance for clustering examples with continuous features is the squared Euclidean distance between two points, *x* and *y*, in *m*-dimensional space:

$$d(\boldsymbol{x}, \boldsymbol{y})^2 = \sum_{j=1}^{m} \left(x_j - y_j\right)^2 = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$$

One of the main challenges in unsupervised learning is that we do not know the definitive answer. The elbow method is a useful graphical technique to estimate the optimal number of clusters (k) for a clustering task, based on the within-cluster SSE (Sum of Squared Errors). In general, as k increases, the distortion (or SSE) will decrease because the data points are closer to their assigned centroids. However, this reduction in distortion isn't always meaningful—after a certain point, adding more clusters gives only minimal improvements. The idea behind the elbow method is to find the point where the distortion stops decreasing significantly and begins to level off. This point is often referred to as the "elbow" because the plot of distortion versus k typically forms a curve that resembles a bent arm. The optimal number of clusters is where this "bend" or elbow occurs.

By plotting the distortion for a range of k values, it becomes easier to visually identify where the improvement in clustering performance starts to slow down. This helps determine the most suitable value for k.

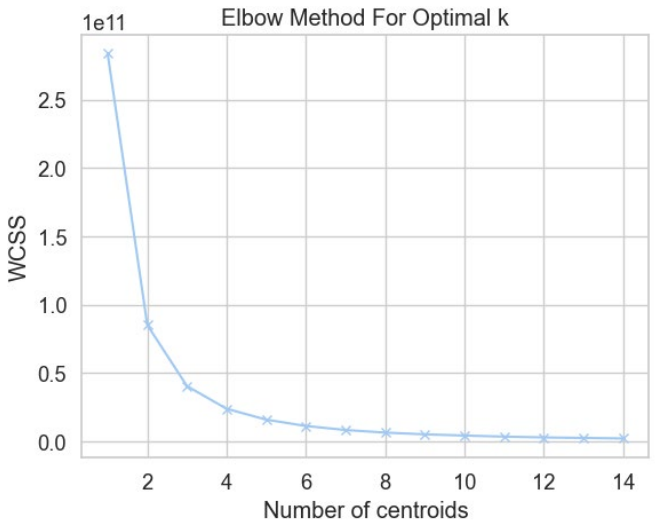

Figure 3.1.2.1. Elbow Method

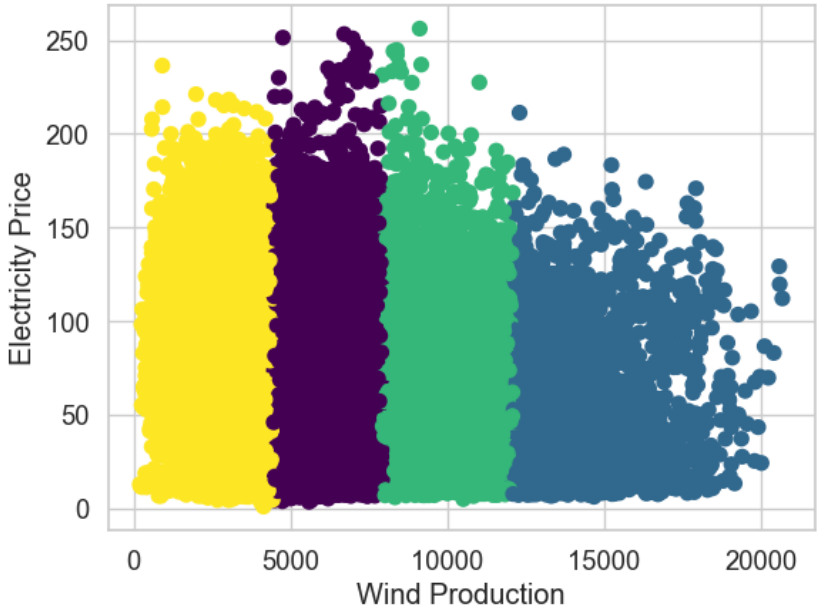

Figure 3.1.2.2. Using the optimal number of clusters

4. Dimensionality Reduction

Many machine learning problems involve working with thousands or even millions of features for each training instance. This can not only make the training process very slow, but it can also make it harder to find a good solution, which is commonly known as the curse of dimensionality.

The good news is that in real-world problems, it's often possible to reduce the number of features significantly, making the problem more manageable and improving the model's performance.

Dimensionality reduction isn't just helpful for speeding up training—it's also incredibly valuable for data visualization. By reducing a high-dimensional dataset to just two or three dimensions, you can plot the data and create a simplified, visual representation that can reveal important patterns, like clusters or trends, that might otherwise be hidden.

Beyond helping you analyze the data, visualization plays a crucial role in communicating your findings, especially to people who aren't data scientists. For decision-makers and stakeholders, a clear and intuitive visual can make complex results far easier to understand and act upon. This ability to "see the data" often bridges the gap between technical analysis and actionable business decisions.

4.1. Principal Component Analysis

Principal Component Analysis (PCA) is the most widely used algorithm for dimensionality reduction. Its popularity comes from its simplicity, efficiency, and ability to effectively transform high-dimensional data into a lower-dimensional space while preserving as much of the data's variability as possible.

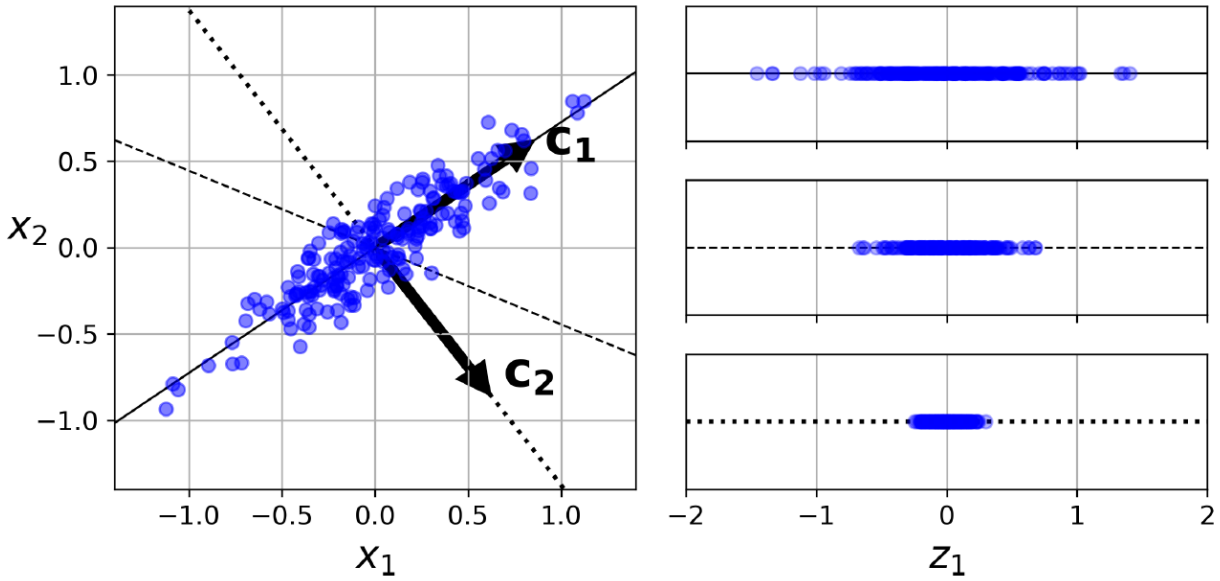

Figure 4.1. Selecting the subspace onto which to project

Before we can reduce the dimensions of our training set, you first need to choose the right direction to project it onto. For example, imagine a simple 2D dataset. If we try to project the data onto different lines (axes), we'll get different results. Some lines preserve more of the data's original structure, while others lose important information.

In this case, the best choice is the line that keeps the most variation in the data, because it retains more of the important information. Another way to think about it is that this line minimizes the average distance between the original data points and their projection on the line. This basic idea is the foundation of Principal Component Analysis (PCA), a method used to reduce the number of features while preserving as much information as possible.

4.1.1. Testing Clustering Using PCA

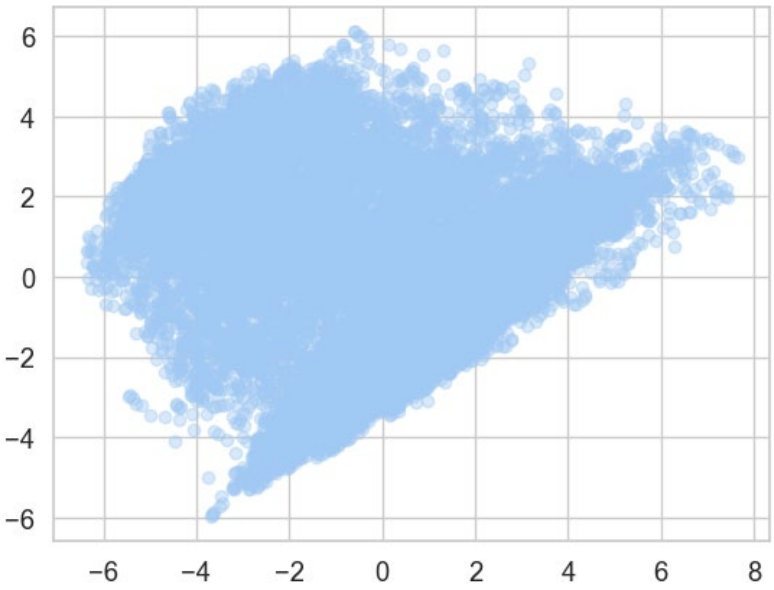

Figure 4.1.1.1. Using PCA we reduced number of Columns to 2

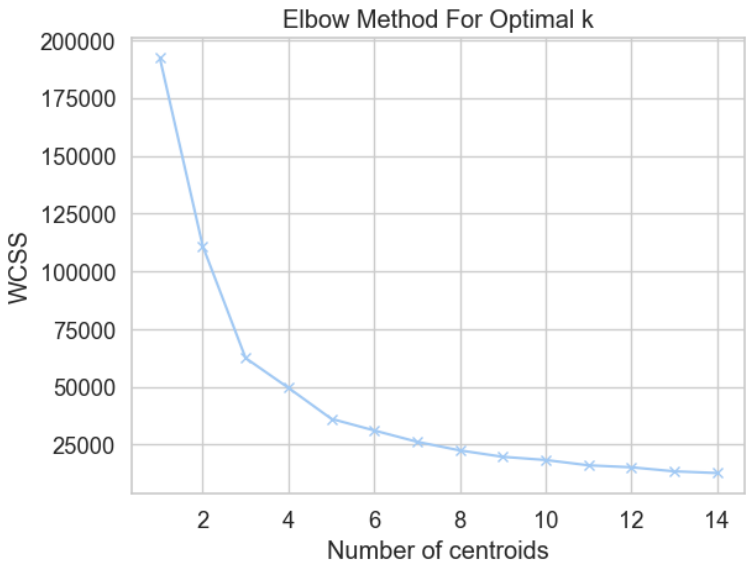

Figure 4.1.1.2. Finding Optimal Number of Clusters

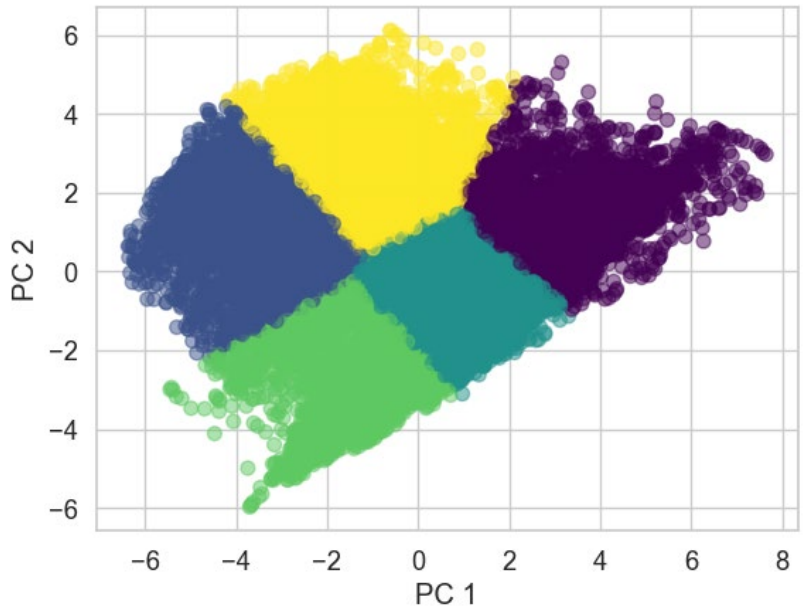

Figure 4.1.1.3. Clustering PCA

4.1.2. Testing Regression Using PCA

Benefits of using PCA components are less data preparation and almost 5 times faster training of the model. But results are a lot worse. RMSE = 22.6191 compared to RMSE = 16.75 when not using PCA.

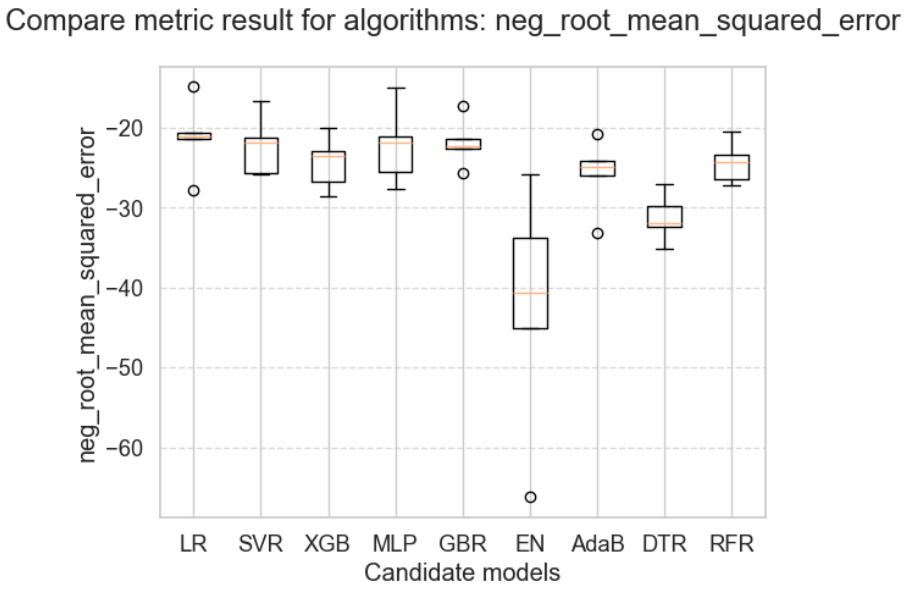

Figure 4.1.1.2. RMSE

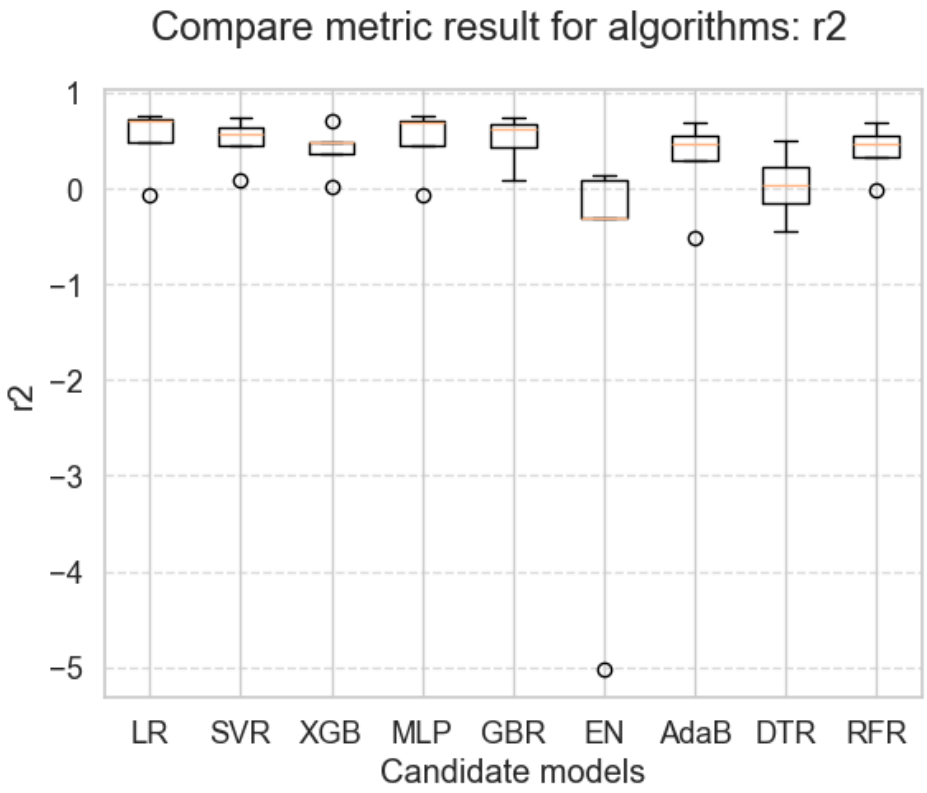

Figure 4.1.1.3. R2

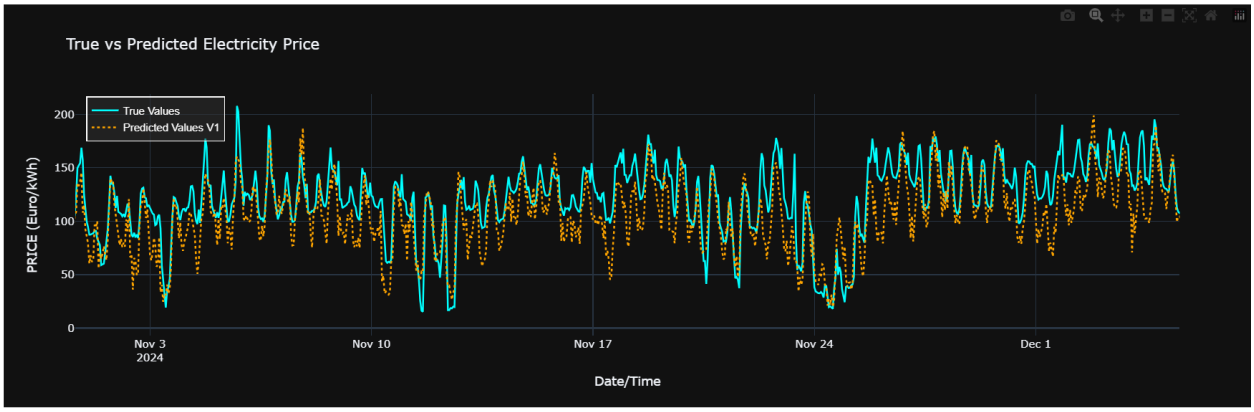

Figure 4.1.1.4. Results