COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MASTER THESIS

MEDIUM-TERM ELECTRIC PRODUCTION FORECASTING USING PROBABILISTIC MACHINE LEARNING ALGORITHMS

Author: Teresa Carbo Espeja

Supervisors:

José Portela González

Jaime Pizarroso Gonzalo

Madrid

January 2025

A los mejores profes de Machine Learning

Contents

List of Figures

List of Tables

Abstract

This Master's Thesis explores the application of advanced probabilistic machine learning techniques for medium-term forecasting of electricity production, with a particular focus on wind power prediction. As the integration of renewable energy sources into the power grid increases, the need to develop accurate and reliable forecasting models arises to ensure the grid's stability and efficiency. This study focuses on two key methodologies: **Partial Least Squares Regression** (PLSR) for power prediction and **Gaussian Mixture Models** (GMM) for generation of scenarios. These two technologies are used to forecast monthly wind power production over a one-year horizon. A significant part of this research was devoted to comprehensive data preparation and exploratory data analysis (EDA), which are essential steps for developing effective forecasting models.

The data preparation phase involved transforming raw data into a usable format for model training. This included cleaning the data to handle missing values. Missing data were addressed through Lagrange interpolation to fill the gaps. Data normalization and scaling were performed to ensure that all variables contributed equally to the model, which facilitated the convergence of machine learning algorithms and improved their performance.

Exploratory Data Analysis (**EDA**) was conducted to uncover underlying patterns and relationships within the data. Through visualizations such as scatter plots and correlation matrices, the EDA process provided insights into the distribution and variability of the data, helping to identify key trends and correlations. This step also involved checking assumptions of normality and linearity, which are important for the validity of some machine learning techniques.

Regarding the methodology, **Partial Least Squares Regression** (PLSR) was used to address the problem of multicollinearity among predictor variables. PLSR reduces data dimensionality by focusing on latent variables, which capture the essential relationships between meteorological factors and wind power production. This method was particularly effective for handling highly correlated variables, making the model robust in its predictions.

Gaussian Mixture Models (GMM) were employed to model the probabilistic distribution of wind energy production. GMMs provide a flexible framework for representing data as a mixture of several Gaussian distributions, each with its own mean and covariance. This probabilistic approach allowed for comprehensive modeling of the variability and uncertainty inherent in wind energy production.

The findings of this research highlight the significant impact of meteorological factors on wind energy production. Variables such as wind speed, atmospheric pressure, and temperature emerged as important predictors. The combination of PLSR and GMM allowed for the development of a sophisticated forecasting model that outperformed traditional deterministic

approaches in terms of accuracy and reliability. The results demonstrate that probabilistic models offer robust and reliable forecasts for wind energy production.

This Master's Thesis makes a significant contribution to the field of renewable energy forecasting by presenting a robust and practical methodology for medium-term forecasting of wind energy production. The integration of comprehensive data preparation, detailed exploratory data analysis, and advanced machine learning techniques highlights the potential of probabilistic models to improve forecast accuracy and provide valuable insights for energy planners and decision-makers, thus supporting more informed decision-making in energy management.

Resumen

Esta trabajo de fin de máster explora la aplicación de técnicas avanzadas de aprendizaje automático probabilístico para la previsión a medio plazo de la producción de electricidad, con un enfoque particular en la predicción de la energía eólica. A medida que aumenta la integración de fuentes de energía renovable en la red eléctrica, surge la necesidad de desarrollar modelos de previsión precisos y fiables para garantizar la estabilidad y eficiencia de la red. Este estudio se centra en dos metodologías clave: **Regresión por Mínimos Cuadrados Parciales** (PLSR) para la predicción de la energía y **Modelos de Mezcla Gaussiana** (GMM) para la generación de escenarios. Estas dos tecnologías se utilizan para prever la producción mensual de energía eólica en un horizonte de un año. Una parte significativa de esta investigación se dedicó a la preparación integral de datos y al análisis exploratorio de datos (EDA), que son pasos imprescindibles para desarrollar modelos de previsión efectivos.

La fase de preparación de datos implicó transformar los datos brutos en un formato utilizable para el entrenamiento del modelo. Esto incluyó la limpieza de los datos para manejar los valores faltantes. Los datos faltantes se trataron mediante interpolación de Lagrange para llenar los vacíos. Se realizaron la normalización y el escalado de los datos para asegurar que todas las variables contribuyeran por igual al modelo, lo que facilitó la convergencia de los algoritmos de aprendizaje automático y mejoró su rendimiento.

Se llevó a cabo un Análisis Exploratorio de Datos (**EDA**) para descubrir patrones y relaciones subyacentes en los datos. A través de visualizaciones como gráficos de dispersión y matrices de correlación, el proceso de EDA proporcionó información sobre la distribución y variabilidad de los datos, ayudando a identificar tendencias y correlaciones clave. Este paso también involucró la verificación de supuestos de normalidad y linealidad, que son importantes para la validez de algunas técnicas de aprendizaje automático.

En cuanto a la metodología, se utilizó la **Regresión por Mínimos Cuadrados Parciales** (PLSR) para abordar el problema de la multicolinealidad entre las variables predictoras. PLSR reduce la dimensionalidad de los datos al centrarse en variables latentes, que capturan las relaciones esenciales entre los factores meteorológicos y la producción de energía eólica. Este método fue particularmente efectivo para manejar variables altamente correlacionadas, haciendo que el modelo sea robusto en sus predicciones.

Se emplearon **Modelos de Mezcla Gaussiana** (GMM) para modelar la distribución probabilística de la producción de energía eólica. Los GMM proporcionan un marco flexible para representar los datos como una mezcla de varias distribuciones gaussianas, cada una con su propia media y covarianza. Este enfoque probabilístico permitió una modelación integral de la variabilidad e incertidumbre inherentes a la producción de energía eólica, generando una gama de posibles resultados con sus probabilidades asociadas.

Los hallazgos de esta investigación resaltan el impacto significativo de los factores meteorológicos en la producción de energía eólica. Variables como la velocidad del viento, la presión atmosférica y la temperatura surgieron como predictores importantes. La combinación de PLSR y GMM permitió el desarrollo de un modelo de previsión que superó a los enfoques deterministas tradicionales en términos de precisión y fiabilidad. Los resultados demuestran que los modelos probabilísticos ofrecen previsiones robustas y fiables para la producción de energía eólica.

Esta trabajo de fin de máster hace una contribución significativa al campo de la previsión de energía renovable al presentar una metodología robusta y práctica para la previsión a medio plazo de la producción de energía eólica. La integración de una preparación de datos integral, un análisis exploratorio de datos detallado y técnicas avanzadas de aprendizaje automático destaca el potencial de los modelos probabilísticos para mejorar la precisión de las previsiones y proporcionar información valiosa para los planificadores y tomadores de decisiones en el ámbito energético, apoyando así una toma de decisiones más informada en la gestión de la energía.

1

Introduction

When you are inspired by some great purpose, some extraordinary project, all your thoughts break their bonds.

Patanjali (1^{th} Century BC)

This first chapter introduces the rationale behind this thesis as well as its main objectives. In addition, it provides the reader with a general overview of the organization and the outline of the dissertation in order to make it easier to follow.

1.1. Motivation

The motivation behind this master's thesis lies in the intricate dynamics of renewable energy integration into the existing electricity grid. As society increasingly pivots towards sustainable energy sources to combat climate change, understanding the complex interplay between renewable energy production and demand becomes essential.

At the core of the research is the probabilistic prediction of electricity (Zhang *et al.*, 2014) generated from the eolic renewable source. By exploring into the probabilistic nature of these predictions, the research aims to analyse and extract the trends that develop over time in production.

This exploration serves a dual purpose. Firstly, it offers insights into the inherent variability and uncertainty associated with renewable energy sources such as wind power (Xue *et al.*, 2014). Insights of this nature are indispensable for grid operators and policymakers who attempt to maintain grid stability and reliability in the face of fluctuating energy inputs.

Secondly, by figuring out the patterns and trends in electricity production, the research aims to find ways to optimize demand management. This information helps stakeholders design custom plans, like demand responsive programs or time-of-use pricing (Dillig *et al.*, 2016), that produce a tendency in consumers to use electricity when renewable energy is available.

Finally, the research aims to go beyond just academic study, applying the methodology developed in this thesis to a real use case (Korpas and Holen, 2006). By doing this, I want to

push us all towards a future with more sustainable and reliable energy, where we use renewable sources in smart ways to meet society's changing needs.

1.2. Thesis objectives

The primary objective of this thesis is to employ various Machine Learning techniques to predict the monthly energy production of wind (eolic) technology in 1 year horizon.

The secondary objectives in this thesis are the following:

- Evaluate the performance of different Machine Learning models in predicting the monthly energy production of wind technology using probabilistic predictions.
- Identify the most influential factors affecting the monthly energy production of wind technology (i.e. meteorological factors).
- Generate valid code for scenario generation to build a probabilistic model.

1.3. Dissertation outline

This dissertation consists of 6 chapters including this first introductory one.

In Chapter 2, we explore the background.

Diving into the data, Chapter 3 introduces how the data has been obtained and preprocessed.

The goal of Chapter 4 is to explain the EDA done in the research.

Chapter 5 involves the main models developed and their accuracy.

Finally, Chapter 6 provides the concluding remarks of the dissertation, summarizing the contributions and future developments.

2

Background

The journey of a thousand miles must begin with a single step.

Laozi (6^{th} Century BC)

This chapter aims to provide a comprehensive background to enhance the understanding of this Master's thesis. Firstly, it introduces fundamental electricity forecasting terminology, followed by an explanation of the concepts and definitions of the Probabilistic Machine Learning techniques employed during the thesis development.

2.1. Introduction

Electricity forecasting has emerged as a critical concern in recent years, attracting significant attention from major corporations. Recognizing its important role in various aspects such as saving funds and maximizing benefits, prominent entities are diligently pursuing the development of highly accurate predictive models. These models are meticulously crafted to predict both the production and consumption patterns of electricity with precision and reliability.

Terminology

Electricity forecasting involves predicting various aspects related to the supply and demand. Understanding fundamental terminology in this domain is essential for grasping the concepts discussed in the thesis.

- Load Forecasting: Predicting the amount of electricity consumption or demand over a certain period. This can include short-term forecasts (e.g., hourly or daily) or long-term forecasts (e.g., yearly).
- Generation Forecasting: Predicting the amount of electricity that will be produced by various generation sources, such as power plants, wind turbines, or solar panels.

- Probabilistic Prediction: Forecasting future outcomes while acknowledging and quantifying uncertainty. Unlike deterministic forecasts, which provide a single point estimate, probabilistic predictions provide a range of possible outcomes along with associated probabilities.
- Renewable energy technologies: Technologies that generate electricity from renewable sources, such as solar, wind, hydro, geothermal, and biomass. These technologies utilize natural processes to produce energy without depleting resources, offering a sustainable and environmentally friendly alternative to traditional fossil fuels.

Probabilistic Forecasting

Probabilistic forecasting holds significant importance in industries such as retail, consumer packaged goods (CPG), and quick-service restaurants (QSR), where accurately predicting critical parameters like demand is essential for success. Extending this concept to the domain of electricity business, the understanding and applying probabilistic prediction techniques in electricity forecasting enable businesses to make more informed decisions, mitigate risks, and enhance overall operational efficiency.

This mathematical concept addresses the inherent uncertainty present in various real-world problems by offering not just a single point estimate but a **range of potential outcomes alongside confidence levels**. These confidence levels quantify the certainty associated with a forecast. For instance, an 80% confidence level indicates an 80% probability that the actual energy production will fall within the forecasted range.

Quantiles serve as another measurement in this concept, representing specific points in a distribution that divide the data into equally sized groups. In demand forecasting, quantiles such as P75, P90, and P95 denote specific levels of confidence or probability. For instance, P90 signifies the 90th percentile of the demand distribution, indicating that 90% of the time, demand will be below this level.

To illustrate, point forecasting would state "we expect to sell 100 units," probabilistic demand forecasting might convey with "there is an 80% chance that sales will fall between 90 and 110 units.". Figure 2.1 shows a comparison between point forecasting and probabilistic forecasting.

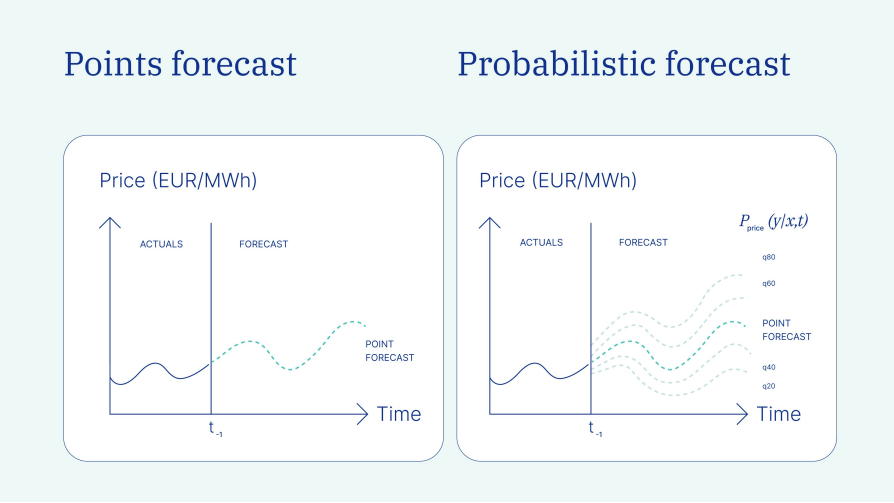


Figure 2.1. Probabilistic vs point forecasting. Source: Dexter energy.

Medium-Term Forecasting

Medium-term forecasting refers to predicting future events, trends, or outcomes over a time horizon typically ranging from several weeks to a few years. This type of forecasting is used in various fields such as economics, finance, meteorology, and energy planning to anticipate developments and make informed decisions. Because of this reason, it has been decided to use this type of forecast.

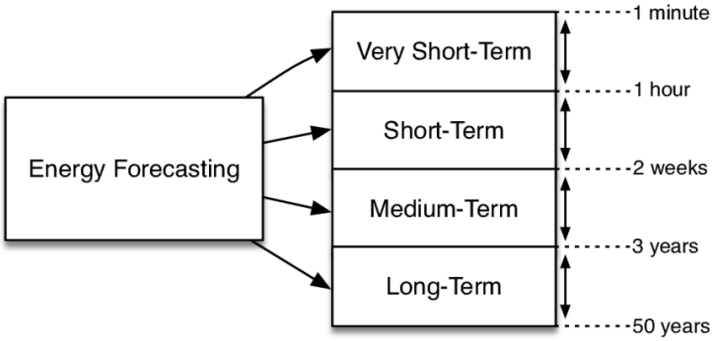


Figure 2.2. Forecasting types

The type of data granularity or aggregation used in medium-term forecasting can vary depending on the specific application and the level of detail required. However, in many cases, medium-term forecasting involves working with aggregated data that captures trends and patterns over relatively large time intervals, such as weeks, months, or quarters. This aggregated data allows forecasters to identify longer-term trends and make predictions at a broader scale, rather than focusing on day-to-day fluctuations or short-term variations.

To maintain precision, data will be collected on a **daily basis**. Following subsequent analysis, consideration will be given to aggregating the data into weekly, monthly, or quarterly intervals as necessary.

2.2. State of the Art

Wind energy has gained considerable attention as a renewable energy source due to its abundant reserves and wide distribution. However, the inherent instability of wind resources poses significant challenges for reliable power grid integration. Effective wind energy prediction is essential to mitigate these challenges. Over the years, numerous forecasting models have been developed, categorized primarily into deterministic and probabilistic methods (Alvarenga *et al.*, 2023).

Deterministic forecasting models provide specific point estimates of wind speed or power, whereas probabilistic forecasting models offer predictions in the form of probability distributions or intervals. Probabilistic models are particularly advantageous as they quantify the uncertainty in forecasts, providing more comprehensive information for decision-makers.

Deterministic models predict specific values based on historical data. They include physical models, statistical models, and machine learning models. Physical models, such as the Numerical Weather Prediction (NWP) models, use meteorological data to simulate wind conditions. Statistical models, including autoregressive and moving average models, rely on historical wind data to make predictions. Machine learning models, like artificial neural networks (ANNs)

and support vector machines (SVMs), have gained popularity due to their ability to capture complex patterns in wind data (Xie *et al.*, 2023).

Probabilistic forecasting methods provide a range of possible outcomes with associated probabilities. These methods can be broadly classified into parametric and non-parametric approaches (Alvarenga *et al.*, 2023). Parametric methods assume a specific distribution for the wind data, such as Gaussian or Weibull distributions, and estimate the parameters of these distributions. Non-parametric methods, such as kernel density estimation and quantile regression, do not assume any predefined distribution and are flexible in handling diverse wind data characteristics.

High-quality input data is crucial for developing accurate wind forecasting models. Common data sources include locally sensed wind speed and power data, NWP data, and exogenous data like meteorological and geographic information. Data processing techniques, such as data decomposition, dimensionality reduction, and data denoising, enhance the quality of input data by reducing noise and extracting relevant features.

The field of wind forecasting continues to evolve with advancements in machine learning and data processing techniques. Hybrid models that combine deterministic and probabilistic approaches are gaining traction for their ability to leverage the strengths of both methods. Additionally, integrating real-time data and improving the scalability of models for large-scale wind farms remain critical areas of research (Xie *et al.*, 2023).

3

Data acquisition and preprocessing

No great discovery was ever made without a bold guess.
Isaac Newton (1643—1727)

This chapter will cover the acquisition and preprocessing of data. It will discuss the sources of the data and the challenges faced in dealing with missing values, with a description on how the challenges have been solved.

3.1. Data acquisition and Data transformation

To forecast energy production for wind energy technology, meteorological data and historical energy production have been selected. Additionally, as previously mentioned, a separate model will be developed for each Autonomous Community in Spain. Consequently, energy production data for each Autonomous Community is required.

Meteorological data

The data has been acquired from Clima, 2023 on a daily basis, conveniently accessible through their website in **Excel** format. This daily data is disaggregated by weather stations, necessitating some data transformation to aggregate it by Autonomous Community. The provided data adheres to the following format:

Estación	Provincia	Temp. máx. (°C)	...	Precipitación 18-24h (mm)
Estaca de Bares	A Coruña	13.9 (13:10)	...	0,2
...	...	...	...	...
Bujaraloz	Zaragoza	14.1 (16:20)	...	0

Table 3.1. Daily meteorological data from select stations

Their columns represent the input meteorological data for the various models. Below is the explanation for each variable associated with each column:

1. **Station**: Name or identifier of the weather station.

2. **Province**: Location of the weather station, typically referring to the province or region where it is situated.

3. **Max Temperature (ºC)**: The highest recorded temperature in Celsius degrees for the given day.

4. **Min Temperature (ºC)**: The lowest recorded temperature in Celsius degrees for the given day.

5. **Avg Temperature (ºC)**: The average temperature in Celsius degrees for the given day, calculated from recorded temperatures over a specific period.

6. **Gust (km/h)**: The maximum gust of wind recorded in kilometers per hour for the given day.

7. **Max Wind Speed (km/h)**: The maximum sustained wind speed recorded in kilometers per hour for the given day.

8. **Precipitation 00-24h (mm)**: Total precipitation recorded in millimeters for the entire day (00:00 to 23:59).

9. **Precipitation 00-06h (mm)**: Precipitation recorded in millimeters from midnight to 6:00 AM.

10. **Precipitation 06-12h (mm)**: Precipitation recorded in millimeters from 6:00 AM to 12:00 PM (noon).

11. **Precipitation 12-18h (mm)**: Precipitation recorded in millimeters from 12:00 PM (noon) to 6:00 PM.

12. **Precipitation 18-24h (mm)**: Precipitation recorded in millimeters from 6:00 PM to midnight.

It has been observed that the meteorological data is more detailed than initially required, given that the data is provided on a per-station basis. Therefore, operations will be conducted to aggregate it by Autonomous Community. Initially, the aggregation of meteorological stations by province will be performed, involving various operations based on the variable. Subsequently, aggregation at the level of Autonomous Communities will be executed by computing averages across their respective provinces. To perform these aggregations, code 7.1 has been developed.

Variables	Aggregation by Province	Aggregation by AC
Maximum Temperature (ºC)	max	mean
Minimum Temperature (ºC)	min	mean
Mean Temperature (ºC)	mean	mean
Wind Gust (km/h)	max	mean
Maximum Wind Speed (km/h)	mean	mean
Precipitation 00-24h (mm)	mean	mean
Precipitation 00-06h (mm)	mean	mean
Precipitation 06-12h (mm)	mean	mean
Precipitation 12-18h (mm)	mean	mean
Precipitation 18-24h (mm)	mean	mean

Table 3.2. Aggregation Methods for Meteorological Variables

To finish with, this dataset contains certain missing data for some days. It is also necessary to identify the missing days in order to address them later. Missing data is observed for the following days:

- 2020: From March 5th to March 7th and the day July 26th.
- 2022: September 27th and September 28th.

To address the absence of data for these days, techniques will be implemented at a later stage.

Energy production data

The daily energy production data have been acquired through the eSios (de España (REE), 2023) website via API. However, obtaining energy production data disaggregated by Autonomous Community is not available. Consequently, an approximation is necessary to acquire this data, which will be explained later.

The initial step involves obtaining the **energy production** dissagregated by renewable technology in the whole Iberian Peninsula using the code 7.2, filtering by wind power technology.

The second step involves formulating the following hypothesis: a weighting will be conducted between the production of wind power technology and the installed capacity of each Autonomous Community for this technology. It is assumed that the higher the installed capacity, the greater the production. These weights will be calculated according to the following formula:

$$Weight_{i,j} = \frac{Installed\ capacity_{i,j}}{Total\ installed\ capacity_i} \tag{3.1}$$

where:

- i- index corresponding to the power technology type.
- j- index corresponding to the Autonomous Community.

therefore:

- $Installed\ capacity_{i,j}$: installed capacity of the i^{th} technology type in the j^{th} Autonomous Community. In this thesis, only wind power technology will be considered.

- $Total\ installed\ capacity_i$: installed capacity of the i^{th} technology in the Iberian Peninsula.

So finally, the daily energy production for the i^{th} technology in the j^{th} Autonomous Community will be calculated as follows:

$$Daily\ energy\ production_{i,j} = Weight_{i,j} \times Total\ energy\ production_i \tag{3.2}$$

Based on the aforementioned hypothesis, it is necessary to have some data related with the **installed capacity** for each technology. It's worth mentioning that the granularity of this capacity data in eSios is only disaggregated by month. The process will follow the same steps as code 7.2, although with slight modifications, resulting in code 7.3. Both codes use the API of this website to extract the necessary data, taking advantage of the disaggregation it offers.

After the data is obtained with the previous codes, the calculation of daily energy production by Autonomous Community and technology is performed with code referenced as 7.4. This code applies the formula previously mentioned.

3.2. Addressing Missing Values

As mentioned earlier, the meteorological data contains missing values. Various techniques are available for addressing this issue (Thomas and Rajabi, 2021), with the method utilized in this section being Lagrange interpolation (Manembu *et al.*, 2015).

Lagrange interpolation is a mathematical method used to estimate values between known data points. It works by constructing a polynomial function that passes through the given data points. This polynomial can then be used to approximate the value of the function at any point within the range of the data.

The key idea behind Lagrange interpolation is to construct a polynomial of degree n, where n is the number of data points minus one. This polynomial is uniquely determined by the given data points and can be written as a weighted sum of Lagrange basis polynomials. These basis polynomials are constructed in such a way that each one evaluates to 1 at its corresponding data point and 0 at all other data points, ensuring that the resulting polynomial passes through each data point.

Once the Lagrange polynomial is constructed, it can be used to estimate the value of the function at any point within the range of the data. This makes Lagrange interpolation a useful tool for filling in missing data points or generating smooth curves from discrete data.

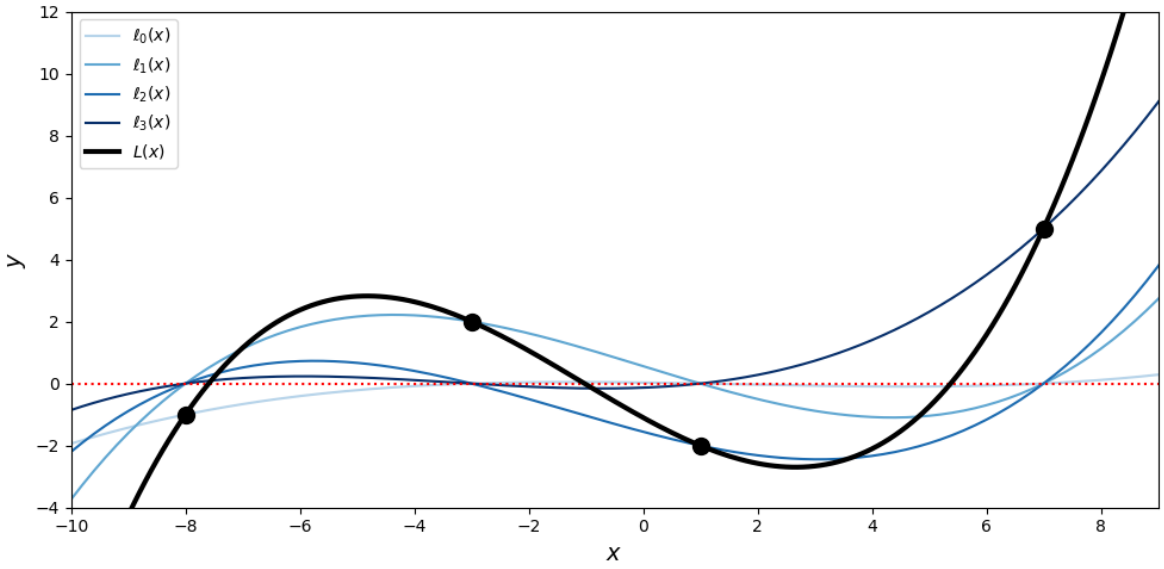


Figure 3.1. Different degrees of Lagrange interpolation

In the image above, with four given points, there's the option to interpolate passing through one of them (such as f_0, f_1, f_2 and f_3,) up to all four. The Lagrange interpolation of degree $n = 4$ precisely represents the function that passes through each of these points.

To tackle the issue of missing values, it has been determined to employ a second-degree interpolation ($n = 2$). The code utilized to address this problem is found in code 7.5.

4

Exploratory Data Analysis

Statistics is the grammar of science.
Karl Pearson (1857–1936)

This chapter explains the exploratory data analysis (EDA) process performed in the data acquired in the previous chapter. The EDA aims to understand distribution, correlations between variables, summarize information, and guide subsequent modeling for deeper analysis and informed decision-making.

4.1. EDA

Exploratory Data Analysis will be used in the collected meteorological data to understand its main characteristics, identify outliers, understand patterns and relationships, and extract insights. This process can help in the development of forecasting models, for example, choosing the optimal model based on the insights retrieved.

First of all, the mean, the standard deviation (std), and the quartiles of the meteorological data can be seen in the table below:

	Tmax	Tmin	Tmed	Rmed	Vmax	Pmed_00_24	Pmed_00_06	Pmed_06_12	Pmed_12_18	Pmed_18_24
count	32850.000000	32850.000000	32850.000000	32850.000000	32850.000000	32850.000000	32850.000000	32850.000000	32850.000000	32850.000000
mean	23.575466	6.020096	14.864414	51.638002	19.447863	2.025615	0.477405	0.464324	0.514524	0.569363
std	7.831106	5.048710	6.653464	17.064865	5.667196	4.983842	1.676711	1.558840	1.545667	1.675045
min	2.740000	0.000000	-3.626238	17.000000	6.625000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	17.400000	1.200000	9.644761	40.000000	15.520340	0.000000	0.000000	0.000000	0.000000	0.000000
50%	22.766667	5.133333	14.259944	48.000000	18.477283	0.067929	0.000000	0.003333	0.000000	0.000000
75%	29.600000	9.900000	19.945625	59.666667	22.333333	1.575856	0.112378	0.111111	0.203233	0.258873
max	46.200000	22.500000	33.962115	203.000000	57.251736	101.892857	69.435714	30.115789	43.247059	32.945714

Figure 4.1. Description of meteorological variables

As observed, the mean of “Tmax” in Spain is about 23.6ºC, corresponding with which one can expect. Regarding “Tmin” and “Tmed”, both are also as expected. The gust and the Max Wind Speed (“Vmax”), measured in Km/h, are within the range that can be expected. Regarding precipitation data, we can see that Spain is not a rainy country as the mean and the 50%

quartile of all the variables related to precipitation are near or equal to zero. Because of this characteristic of rain, a different model will be proposed in section 5.2 for rainy days, as it is expected that the relationship between energy generation and meteorological data changes substantially for a rainy day versus a not rainy day. This approach is further motivated by the precipitation histogram depicted in figure 4.2.

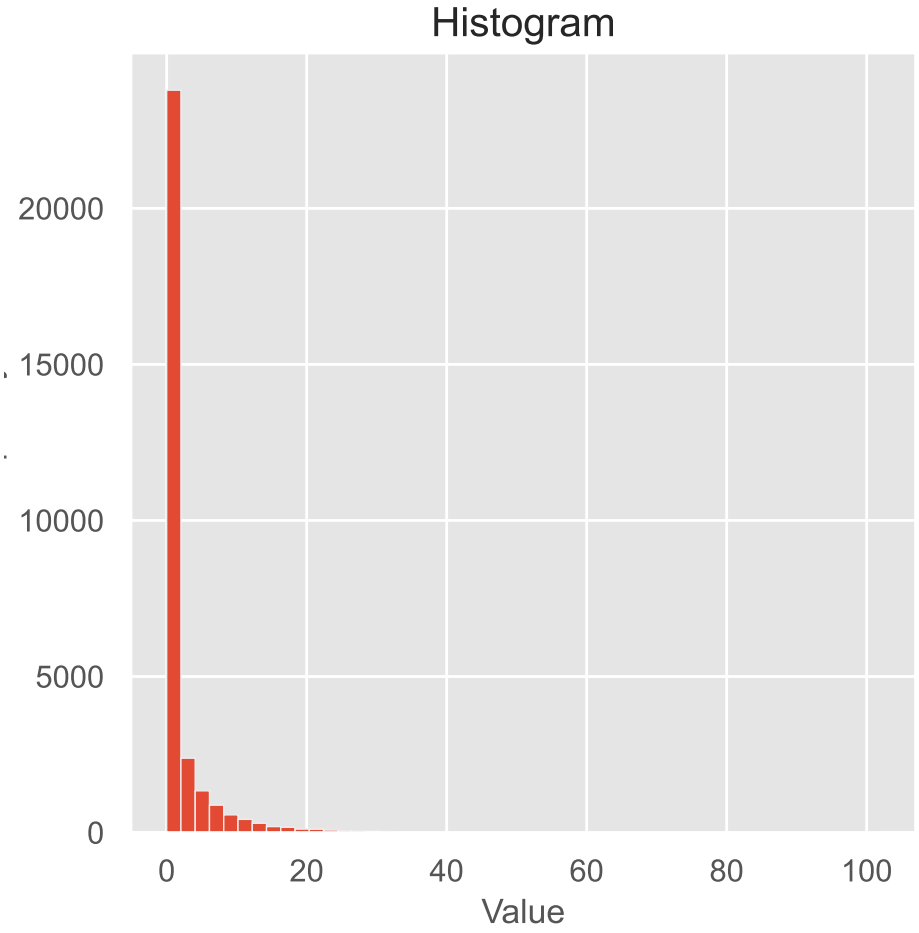


Figure 4.2. Histogram of the precipitation in Spain

To differentiate between rainy and non-rainy days, it is hypothesized that a rainfall of more than 1mm is needed for a day to be considered rainy.

It is going to be displayed the pairplot (figure 4.3 for understanding the relationships between multiple variables in a dataset.

- **Temperature Relationships (Tmax, Tmin, Tmed)**:
  - There is a strong positive correlation between Tmax and Tmed, and between Tmin and Tmed, as expected, since the average temperature is calculated from the maximum and minimum temperatures.
  - There is also a significant positive correlation between Tmax and Tmin.
- **Wind Relationships (Rmed, Vmax)**:

– There is a notable positive correlation between Rmed and Vmax, which is logical since higher gusts of wind tend to accompany higher maximum wind speeds.

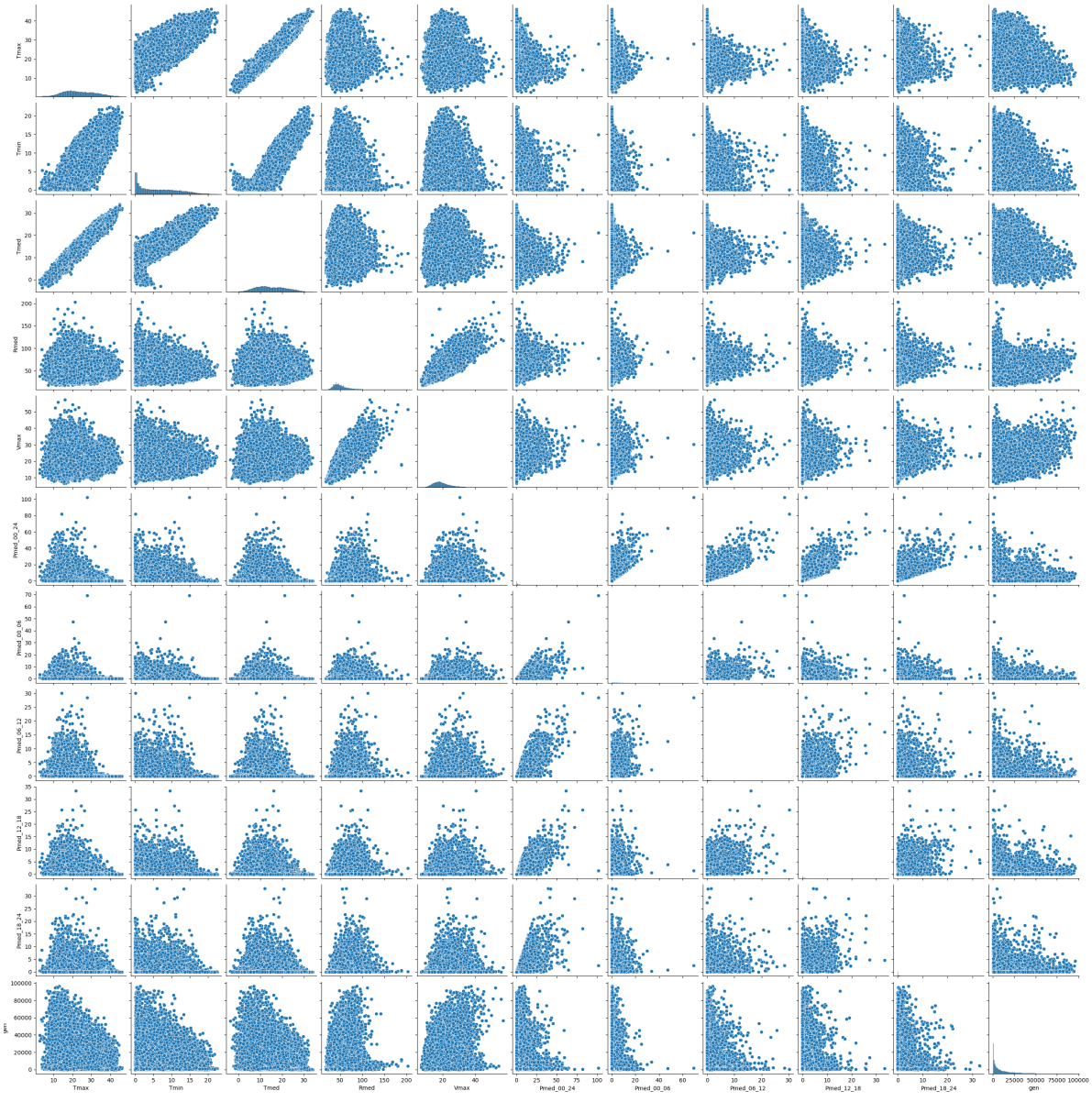


Figure 4.3. Pairplot of the data

- **Precipitation Relationships**:
  - The partial precipitation variables (Pmed_00_06, Pmed_06_12, Pmed_12_18, Pmed_18_24) are strongly correlated with each other and with the total daily precipitation (Pmed_00_24). This indicates that the daily values are a direct sum of the partial precipitation amounts.
- **Interactions between Temperature and Precipitation Variables**:
  - There are no strong correlations between temperature variables (Tmax, Tmin, Tmed) and precipitation variables (Pmed_00_24, Pmed_00_06, Pmed_06_12, Pmed_12_18, Pmed_18_24). This suggests that daily temperatures are not directly influenced by the amount of precipitation on the same day.

- **Interactions between Wind and Precipitation Variables**:
  - Similarly, there are no strong correlations between wind variables (Rmed, Vmax) and precipitation variables. This indicates that the amount of precipitation does not have a significant impact on the daily maximum wind gusts and speeds.
- **Distributions**:
  - The distributions of individual variables can be observed in the histograms on the diagonal of the pairplot. This provides information about the dispersion and central tendency of each variable.

In order to see the correlations between variables, the correlation matrix will be plotted below:

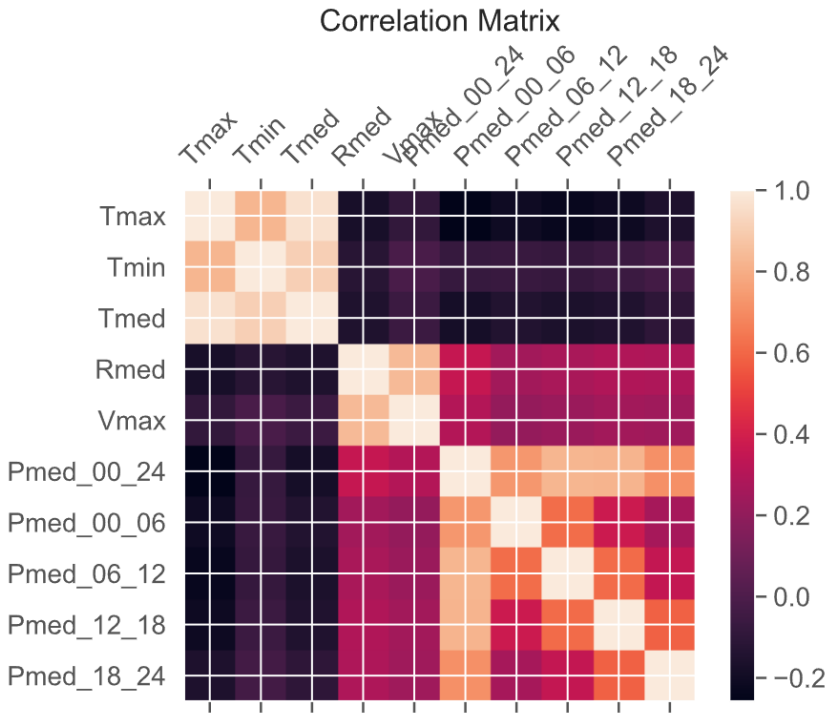


Figure 4.4. Correlation matrix of meteorological variables

1. High Positive Correlations:
   - **Tmax**, **Tmin** and **Tmed** have a high positive correlation (between 0.8 to 1). This indicates that when the maximum temperature of the day is high, it is likely that the minimum temperature and the average temperature are also high, which is expected.
2. Negative Correlations:
   - **Rmed** (Gust) and **Temperatures** (Tmax, Tmin, Tmed) have negative correlations. This suggests that as the temperature increases, the gust tends to be lower.
3. Precipitations:
   - **Pmed_00_24** (Total Day Precipitation) has significant positive correlations with **Pmed_00_06**, **Pmed_06_12**, **Pmed_12_18**, **Pmed_18_24**, which is natural since the total precipitation of the day is the sum of the precipitations in these time intervals.

- Among the different precipitation intervals **Pmed_00_06**, **Pmed_06_12**, **Pmed_12_18**, and **Pmed_18_24** have significant correlations with each other, indicating that, if it rains in a period of the day, it will rain during more than one period.

Due to these high correlations between group of variables, it can be inferred that some variables would not provide any information to predict the wind energy production. Therefore, we need to make sure that the variables which carry the most information are selected as input variables in our models.

5

Models

Mathematics is the language God used
to write the world.
Galileo Galilei (1564–1642)

In this chapter, the modeling process is described in detail, focusing on the application of various regularization techniques—specifically Ridge, Lasso, and Partial Least Squares Regression (PLSR)—in contrast to traditional Linear Regression (LR). Besides, it will be explained the generation of scenarios using Gaussian Mixture Models (GMM) for probabilistic forecasting.

5.1. Introduction

This section introduces key concepts essential for understanding the approach to modeling wind energy production. Modeling involves creating a mathematical representation of a real-world process, in this case, the generation of wind energy. To build robust and reliable models, data is partitioned into training and testing sets. The training set is used to fit the model, while the testing set is utilized to evaluate its performance, ensuring that the model generalizes well to unseen data. For each community under study, a separate model is developed, tailored to its specific characteristics.

Linear regression serves as a fundamental technique in these models, aiming to predict a target variable based on one or more predictor variables by fitting a linear equation to the observed data. It is often used as a baseline model to establish initial metrics for comparison. By providing a straightforward and interpretable benchmark, linear regression helps assess the effectiveness of more complex models. However, linear regression can suffer from overfitting, particularly with complex datasets containing numerous predictors. To mitigate this issue, regularization methods are applied. Regularization introduces a penalty to the model's complexity, encouraging simpler models that generalize better to new data. Techniques such as Lasso, Ridge regression and Partial Least Square Regression are employed to improve the model's performance by preventing overfitting, thereby ensuring more reliable predictions across different communities.

5.2. Regularization

Regularization is a technique used in machine learning models to prevent overfitting and improve model generalization. It works by adding a penalty term based on the model parameters to the loss function that the model optimizes, which discourages the model from learning overly complex patterns in the training data that may not generalize well to unseen data.

With these properties, it is going to be used some regularization techniques in the context of understanding which variables (or features) are important to the model. These techniques offer several benefits:

1. **Feature Selection**: Some forms of regularization, like L1 regularization (Lasso), can result in sparse solutions where some feature coefficients are zero. This effectively performs feature selection, identifying which features are most important for the model.
2. **Multicollinearity**: Regularization can help in situations where you have multicollinearity (i.e., high correlations between predictor variables). In these cases, standard regression might distribute the effect of one variable across several correlated variables, finding spurious relationships between output and inputs.
3. **Interpretability**: By discouraging complex models (i.e., models with too many coefficients), regularization can help to make the model more interpretable. This can make it easier to understand the effect of individual variables on the model's predictions, identifying which are the most important variables to predict the outcome.
4. **Preventing Overfitting**: as stated previously, regularization helps to prevent overfitting by adding a cost to the loss function for large coefficients. This means that models are less likely to fit the noise in the training data and are more likely to find the signal.

The selected regularization techniques include **Ridge Regression**, **Lasso Regression** and **Partial Least Square Regression**. These techniques have been applied across all Autonomous Communities but for illustrative purposes, however, an illustrative analysis will be shown for only one of them, Galicia. Besides, a regular **Linear Regression** model will be trained in order to compare it with the chosen regularization techniques.

Ridge Regression

As stated before, regularization techniques add a penalty term to the model objective function. the objective function for Ridge regression is given by:

$$\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \alpha \sum_{j=1}^{p} \beta_j^2 \right\} \tag{5.1}$$

Where:

- y_i is the response for the i-th observation,
- β_0 is the intercept term,
- β_j is the coefficient for the j-th predictor,
- x_{ij} is the value of the j-th predictor for the i-th observation,
- n is the number of observations,

- p is the number of predictors,

- α is the tuning parameter that controls the strength of the penalty.

After training the regularization model, the development of the predictors proceeds as follows:

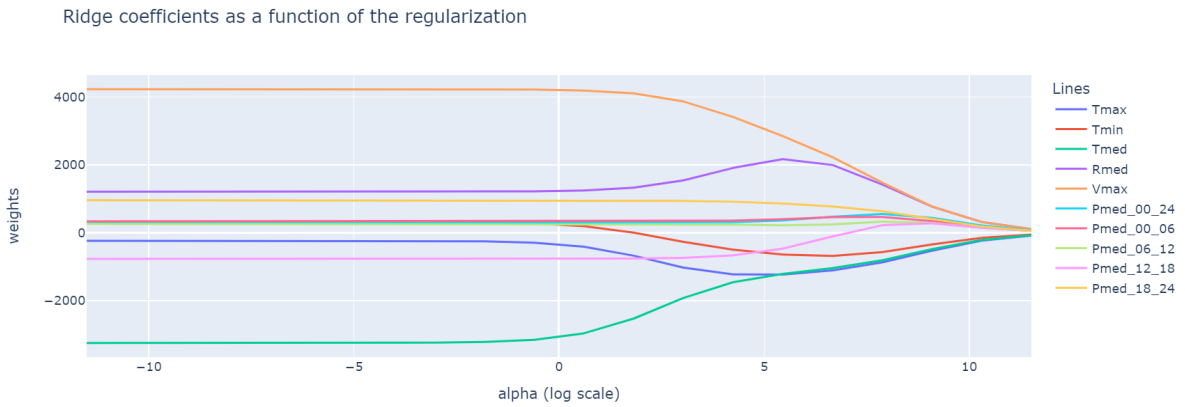


Figure 5.1. Ridge coefficients

The graph 5.1 shows the Ridge coefficients as a function of the regularization parameter α, on a logarithmic scale. On the x-axis is α and on the y-axis are the values of the coefficients (weights).

When α is very small (close to 0 or negative values), the regularization is minimal and the coefficients adjust almost freely to minimize the training error. We observe that some coefficients have very large values, indicating overfitting. As α increases, the regularization becomes stronger. The coefficients begin to reduce in magnitude, approaching zero, which reduces the complexity of the model and the risk of overfitting. With very large values of α, the coefficients approach zero. This can lead to a model that underfits the data, losing the ability to capture the relationship between the independent variables and the dependent variable.

As for the behavior of the variables, some variables like "Vmax" (maximum wind speed) start with very high coefficient values, but they drastically reduce as α increases. Other variables, like the different precipitation measurements ("Pmed_00_24", "Pmed_00_06", etc.), show less drastic changes but follow a decreasing trend with the increase of α.

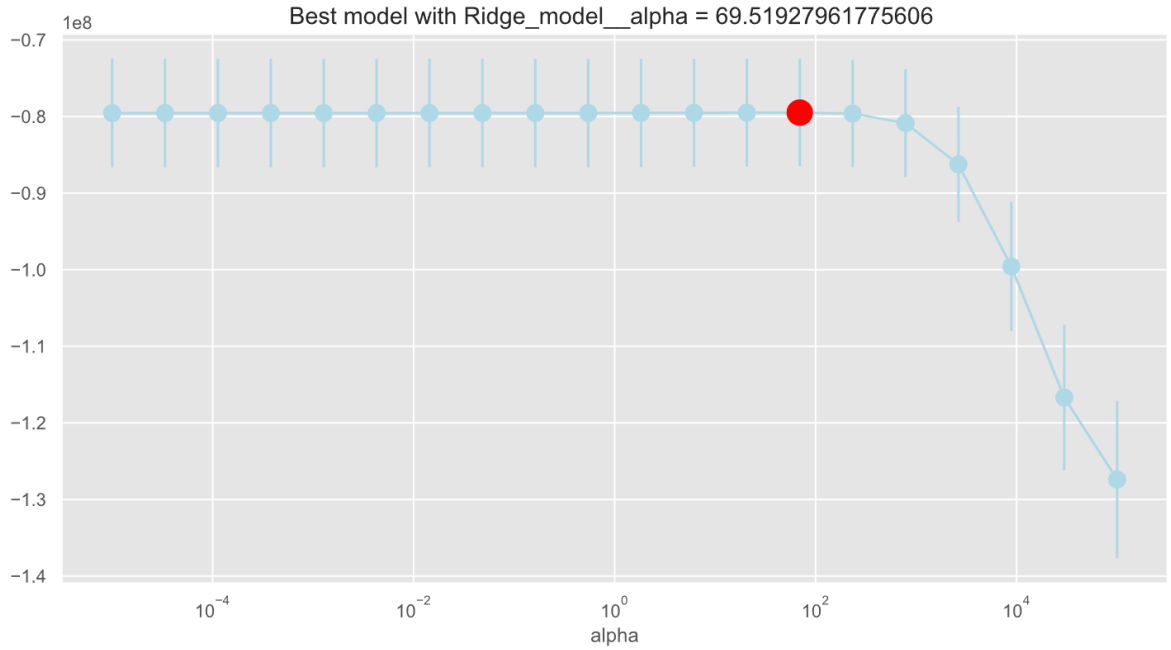


Figure 5.2. Alpha chosen for Ridge

It can be seen that the model has chosen an α of **69.5** with a training R2 of **0.411** and test R2 of **0.407**.

	Coefs	Selected
num__Tmax	-79.308400	True
num__Tmin	-50.080141	True
num__Tmed	-71.574398	True
num__Rmed	109.034026	True
num__Vmax	107.827917	True
num__Pmed_00_24	78.628981	True
num__Pmed_00_06	59.482198	True
num__Pmed_06_12	55.718488	True
num__Pmed_12_18	56.301411	True
num__Pmed_18_24	65.366272	True

Figure 5.3. Variables chosen for Ridge

It can be seen that with this type of regularization, there is no variable discarded.

Lasso Regression

Lasso Regression, also known as Least Absolute Shrinkage and Selection Operator, is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This is done by imposing a constraint on the model parameters

that causes regression coefficients for some variables to shrink towards zero. Variables with a regression coefficient equal to zero after the shrinkage process are excluded from the model.

The Lasso method performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients. This type of regularization can result in sparse models with few coefficients; Some coefficients can become zero and eliminated from the model. Larger penalties result in coefficient values closer to zero, which is the ideal for producing simpler models.

The objective function to minimize for Lasso Regression can be represented as:

$$\min_{\beta} \left\{ \frac{1}{2n} ||y - X\beta||_2^2 + \alpha ||\beta||_1 \right\} \tag{5.2}$$

In this equation:

- y is the output vector
- X is the input matrix
- β is the vector of coefficients
- $||.||_2$ denotes the L2 norm
- $||.||_1$ denotes the L1 norm
- α is the regularization parameter controlling the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage.

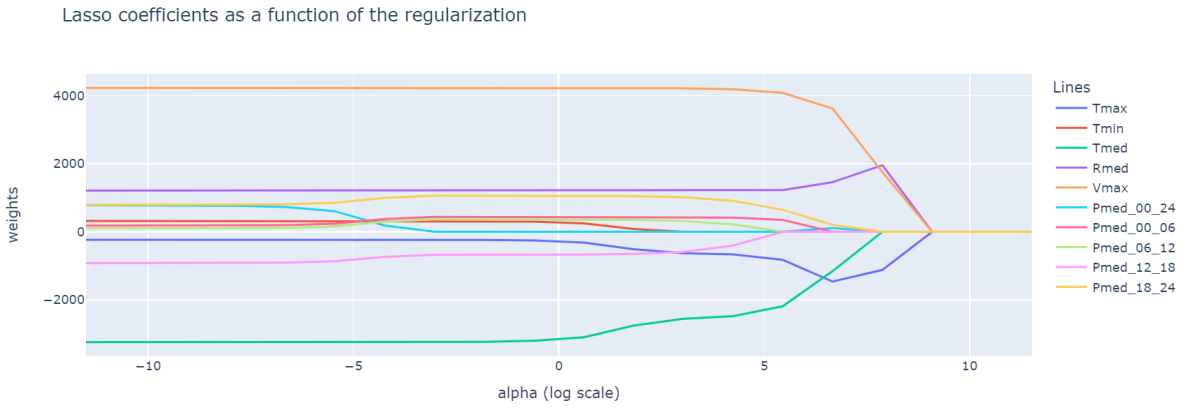


Figure 5.4. Lasso coefficients

The graph 5.4 shows the Lasso coefficients as a function of the regularization parameter α, also on a logarithmic scale. On the x-axis is α and on the y-axis are the values of the coefficients (weights).

For very small α (close to 0 or negative values), the behavior is similar to Ridge. The regularization is minimal and the coefficients adjust almost freely. As α increases, the coefficients begin to reduce. Unlike Ridge, Lasso can bring coefficients exactly to zero, effectively eliminating some variables from the model. With very large values of α, many coefficients are reduced to zero, resulting in a very simple model that likely underfits the data.

As for the behavior of the variables, the variable "Vmax" starts with very high coefficient values but reduces significantly with an increase in α. Several precipitation variables ("Pmed_00_24", "Pmed_00_06", etc.) and temperatures ("Tmax", "Tmin", "Tmed") also show reduction trends and some of them can reach zero.

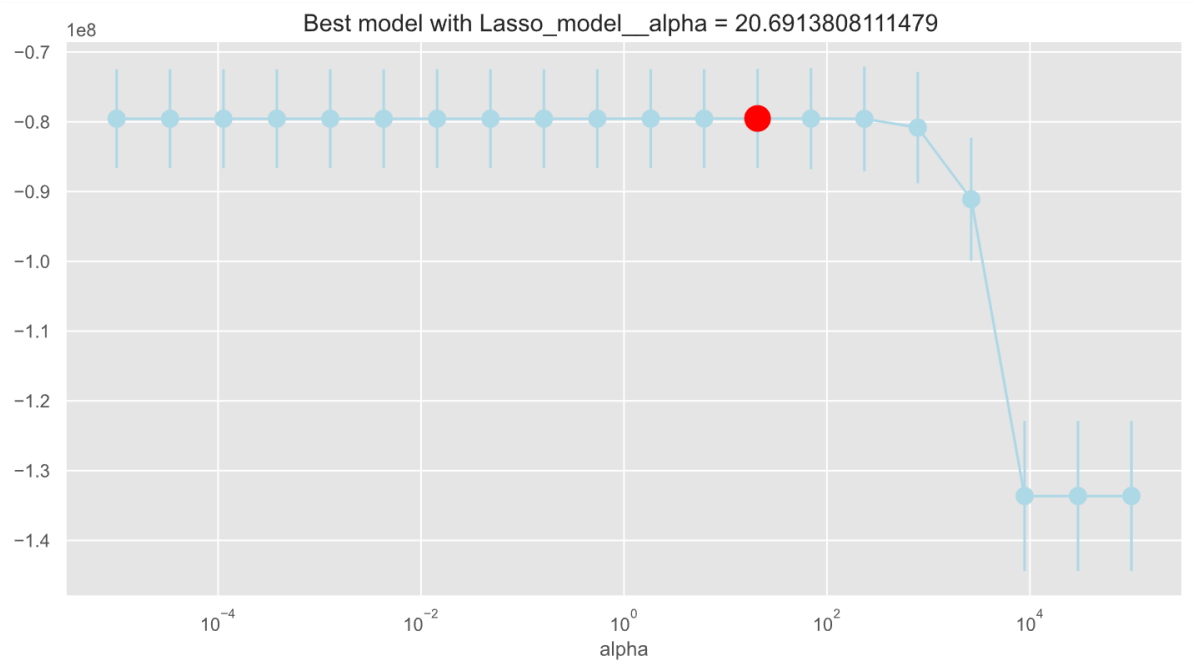


Figure 5.5. Alpha chosen for Lasso

It can be seen that the model has chosen an α of **20.69** with a training R2 of **0.412** and test R2 of **0.411**.

	Coefs	Selected
num__Tmax	-626.683649	True
num__Tmin	-0.000000	False
num__Tmed	-2555.532736	True
num__Rmed	1215.891596	True
num__Vmax	4214.111434	True
num__Pmed_00_24	0.000000	False
num__Pmed_00_06	428.014472	True
num__Pmed_06_12	319.841010	True
num__Pmed_12_18	-593.032606	True
num__Pmed_18_24	1016.192071	True

Figure 5.6. Variables chosen for Lasso

The variables chosen for Lasso Regression would be: "Tmax", "Tmed", "Rmed", "Vmax", and the precipitations by intervals of time.

Partial Least Square Regression

Partial Least Squares Regression (PLSR) is a statistical method that bears some relation to principal components regression; instead of finding hyperplanes of maximum variance between the response and independent variables, it finds a linear regression model by projecting the predicted variables and the observable variables to a new space.

The objective function of PLSR can be represented as:

$$\min_{w,c} \left\{ \sum_{i=1}^{n} \left(y_i - x_i^T wc\right)^2 + \alpha \left(||w||_2^2 + ||c||_2^2\right) \right\} \tag{5.3}$$

- x_i is the i-th observation of the predictor variables,
- y_i is the i-th observation of the response variable,
- w is the weight vector for the predictor variables,
- c is the weight for the response variable,
- $||.||_2$ denotes the L2 norm,
- α is the regularization parameter controlling the amount of shrinkage.

By solving this optimization problem, PLSR finds the weights w and c that minimize the sum of the squared errors between the observed and predicted response variables, subject to a penalty term that prevents overfitting.

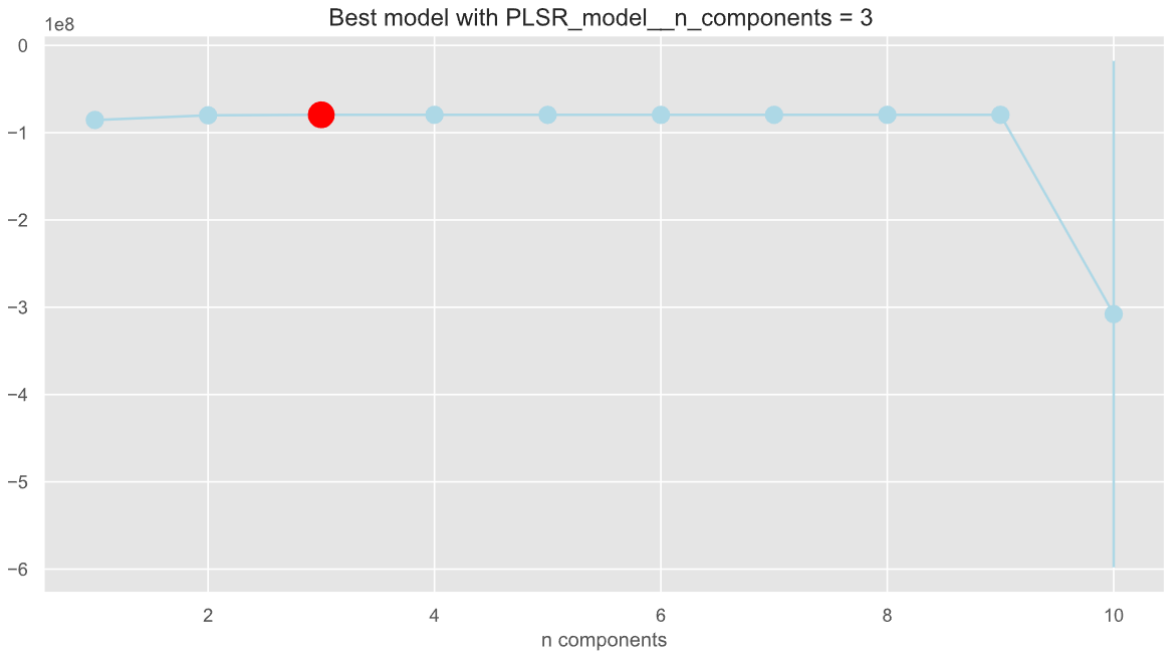


Figure 5.7. Number of components for PLSR

It can be seen that the model has chosen a number of components of $n = 3$ with a training R2 of **0.410** and test R2 of **0.411**.

	Coefs	Selected
num__Tmax	-1210.002189	True
num__Tmin	-783.585443	True
num__Tmed	-1146.382254	True
num__Rmed	2678.255626	True
num__Vmax	2881.274576	True
num__Pmed_00_24	278.654764	True
num__Pmed_00_06	379.310568	True
num__Pmed_06_12	-49.988990	True
num__Pmed_12_18	-394.783888	True
num__Pmed_18_24	845.581987	True

Figure 5.8. Variables chosen for PLSR

It can also be seen that with this type of regularization, there is no variable discarded.

Comparison of the regularization techniques with Linear Regression

In Figure 5.9 it can be seen the negative Root Mean Squared Error (Neg RMSE) for different models including Ridge, Lasso, Linear Regression (LR1), and Partial Least Squares Regression (PLSR), providing a sense of their performance. Lower Neg RMSE values indicate better performance since the negative sign implies that higher absolute values are actually worse.

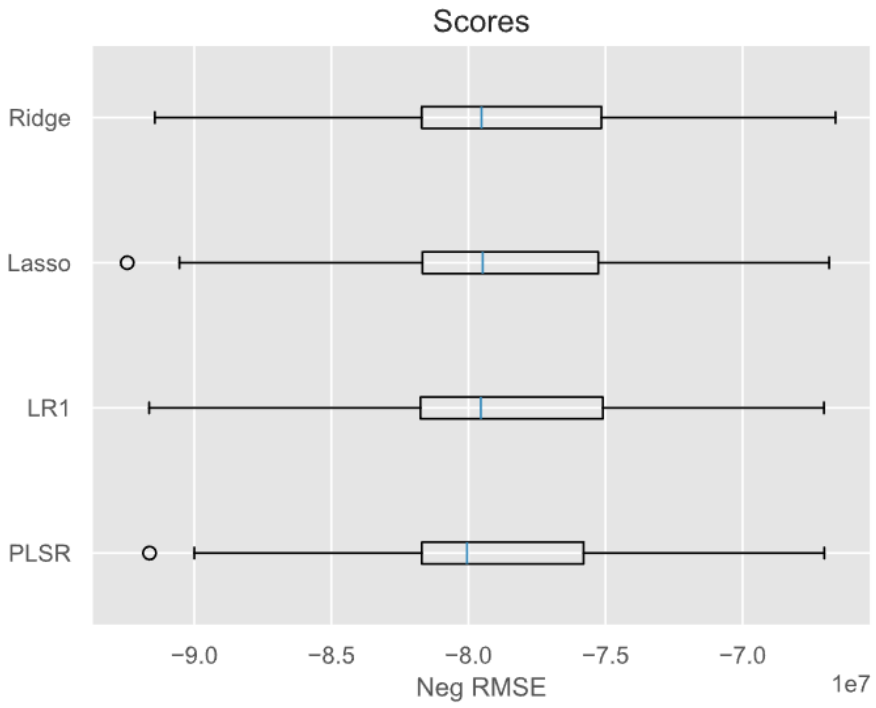


Figure 5.9. Comparison of Regularization Techniques

In the displayed graph, the Neg RMSE values are in the range of -9×10^7 to -7×10^7, suggesting variability in the performance of these models on the dataset. All four models—Ridge, Lasso, LR1, and PLSR—exhibit Neg RMSE values within a similar range, indicating that they perform comparably.

Specifically, the Ridge and LR1 models show slightly higher Neg RMSE values compared to Lasso and PLSR, indicating marginally poorer performance. However, the differences are relatively small, suggesting that while there are some variations, all models have a comparable predictive accuracy on this dataset. The PLSR model demonstrates the narrowest interquartile range, implying more consistent performance across different data subsets. Conversely, the Ridge and LR1 models have slightly wider distributions, indicating more variability in their performance.

Overall, while all four models demonstrate similar performance with Neg RMSE values tightly clustered, PLSR and Lasso slightly outperform Ridge and LR1 in terms of consistency and lower Neg RMSE values. For this reason, we will use **Linear Regression** as the baseline metric to compare the performance of **PLSR**. As previously mentioned, the models will be trained separating between rainy and non-rainy days to assess their effectiveness under different weather conditions.

5.3. Baseline metrics

It is believed that distinguishing between rainy and non-rainy days could potentially enhance the performance of the models. Therefore, separate Linear Regression models have been trained for rainy and non-rainy days, and their respective metrics have been calculated. After being trained with code 7.6, it can be seen in the figure 5.10, that when differentiating, the R^2 decreases for non-rainy days for each autonomous community. Even in Navarra, a model with a negative R^2 has been obtained.

Precipitación	CCAA	Mae TR	Mae TS	RMSE TR	RMSE TS	R2 TR	R2 TS	MAPE TR	MAPE TS
SI	Andalucia	7263.307	8095.878	8988.952	10098.916	0.495	0.29	0.47	0.582
NO	Andalucia	6782.119	7320.153	8573.27	9165.08	0.26	0.159	0.497	0.571
SI	Aragon	9767.472	10824.026	12164.957	13634.655	0.412	0.348	0.689	0.663
NO	Aragon	9155.997	8108.274	11627.225	10441.094	0.214	0.162	0.676	0.533
SI	Cantabria	70.82	70.837	88.402	89.619	0.456	0.443	0.435	0.416
NO	Cantabria	60.789	58.828	76.997	74.444	0.306	0.303	0.47	0.448
SI	Castilla la Mancha	8793.815	10037.265	10757.288	12491.583	0.512	0.354	0.496	0.701
NO	Castilla la Mancha	7919.634	8177.123	10031.027	10199.266	0.327	0.163	0.458	0.559
SI	Castilla y Leon	12519.606	12886.591	15692.991	15710.697	0.535	0.515	0.474	0.381
NO	Castilla y Leon	9966.878	11290.607	12615.155	14086.552	0.394	0.374	0.406	0.443
SI	Cataluna	2526.555	2799.102	3144.783	3469.083	0.461	0.288	0.475	0.556
NO	Cataluna	2576.401	2649.895	3273.696	3367.583	0.29	0.272	0.495	0.518
SI	Pais Vasco	318.66	342.885	396.945	433.292	0.433	0.333	0.428	0.547
NO	Pais Vasco	263.369	286.424	334.81	364.503	0.295	0.19	0.462	0.54
SI	Asturias	1250.996	1295.467	1586.782	1670.531	0.429	0.386	0.461	0.444
NO	Asturias	1030.415	1115.591	1317.341	1416.548	0.293	0.226	0.461	0.515
SI	Navarra	2798.177	3125.341	3481.695	3921.87	0.399	0.116	0.485	0.594
NO	Navarra	2279.888	2257.83	2942.641	2822.356	0.249	-0.011	0.501	0.548
SI	Comunidad Valenciana	2495.438	2558.956	3081.2	3189.756	0.37	0.271	0.559	0.554
NO	Comunidad Valenciana	2367.709	2599.071	3010.289	3261.552	0.348	0.309	0.473	0.512
SI	Extremadura	120.045	125.518	169.576	180.774	0.325	0.249	0.59	0.653
NO	Extremadura	82.649	77.915	114.94	112.534	0.25	0.186	0.506	0.475
SI	Galicia	7588.682	8053.854	9493.845	9862.924	0.461	0.428	0.421	0.445
NO	Galicia	6028.766	6276.93	7569.555	8217.152	0.3	0.268	0.443	0.424
SI	La Rioja	910.486	958.642	1121.182	1206.928	0.511	0.489	0.45	0.53
NO	La Rioja	801.346	823.098	1019.884	1043.977	0.315	0.308	0.454	0.492
SI	Region de Murcia	540.605	574.467	652.453	690.626	0.318	0.33	0.559	0.524
NO	Region de Murcia	528.054	565.074	668.302	697.84	0.312	0.324	0.502	0.536

Figure 5.10. Metrics in LR

There's a proposal to create a model with a lag of 1 year for when the model performs worse. A model with a lag of 1 year refers to a type of time series model where the value of a variable at a certain time (t) is influenced by the values of the same variable at previous times, in this case, one year prior ($t - 1$ year). In this case, the primary hypothesis is that the energy production for a given day is identical to that of the same day in the previous year.

To train the model, it has been developed the code 7.7. The results of this model are shown in figure 5.11. However, as seen in the table below, this idea is rejected because it performs worse, as all the R^2 are negatives.

ccaa	MAE	RMSE	R2	MAPE
Andalucia	196137523.033	14004.911	-0.686	0.714
Aragon	273008758.141	16522.977	-0.356	0.635
Cantabria	20264.133	142.352	-0.717	0.727
Castilla la Mancha	284100356.472	16855.277	-0.614	0.695
Castilla y Leon	627019753.56	25040.362	-0.662	0.704
Cataluna	27057281.204	5201.661	-0.687	0.715
Pais Vasco	388452.609	623.26	-0.711	0.723
Asturias	5901807.08	2429.364	-0.577	0.696
Navarra	25221751.812	5022.126	-0.569	0.679
Comunidad Valenciana	24440035.753	4943.686	-0.707	0.721
Extremadura	35601.779	188.684	-0.377	0.708
Galicia	229794051.923	15158.959	-0.672	0.708
La Rioja	3264050.024	1806.668	-0.717	0.727
Region de Murcia	1122327.782	1059.4	-0.717	0.727

Figure 5.11. Metrics in LR with lag

With all this, the metrics of this work for each autonomous community are established in figure 5.10.

5.4. Partial Least-Squares Regression

To train the model it has been needed the code 7.8. The parameters used in the model has been `'PLSR_model__n_components'`, and a grid of `'PLSR_model__n_components'`: {1, 2, 3, 4, 5, 6, 7, 8} has been employed. Depending on the autonomous community, the selected grid has varied, as the optimal one for each case is chosen.

After being trained with the code previously mentioned, the model shows the following metrics:

Precipitación	CCAA	Mae TR	Mae TS	RMSE TR	RMSE TS	R2 TR	R2 TS	MAPE TR	MAPE TS
SI	Andalucia	7871.032	8422.627	9837.753	10423.283	0.395	0.243	0.503	0.622
NO	Andalucia	7165.094	7441.503	8938.902	9277.002	0.195	0.138	0.527	0.571
SI	Aragon	10294.594	11795.053	12980.591	14737.117	0.33	0.238	0.714	0.741
NO	Aragon	9443.021	8316.698	11958.963	10583.955	0.169	0.139	0.701	0.557
SI	Cantabria	73.134	71.555	90.696	90.926	0.427	0.427	0.447	0.423
NO	Cantabria	62.101	60.764	78.514	75.861	0.278	0.277	0.482	0.458
SI	Castilla la Mancha	9568.639	10101.287	11767.998	12348.196	0.416	0.369	0.547	0.659
NO	Castilla la Mancha	8163.74	8201.978	10287.276	10127.156	0.292	0.175	0.47	0.553
SI	Castilla y Leon	12932.206	13150.08	16241.389	16152.171	0.502	0.488	0.489	0.402
NO	Castilla y Leon	10131.596	11352.47	12786.679	14103.775	0.377	0.373	0.413	0.446
SI	Cataluna	2685.811	2834.739	3300.157	3493.801	0.407	0.278	0.501	0.558
NO	Cataluna	2678.622	2766.77	3383.227	3472.85	0.241	0.225	0.516	0.538
SI	Pais Vasco	330.03	349.407	411.033	436.909	0.392	0.322	0.439	0.554
NO	Pais Vasco	268.985	286.715	342.066	360.452	0.264	0.208	0.474	0.535
SI	Asturias	1291.452	1284.75	1633.354	1659.024	0.395	0.394	0.475	0.429
NO	Asturias	1047.656	1106.884	1346.087	1409.916	0.262	0.233	0.472	0.518
SI	Navarra	2938.756	3084.161	3738.112	3872.317	0.307	0.139	0.513	0.594
NO	Navarra	2351.154	2231.167	3005.159	2760.421	0.217	0.033	0.516	0.548
SI	Comunidad Valenciana	2623.645	2482.378	3233.509	3081.27	0.306	0.32	0.592	0.537
NO	Comunidad Valenciana	2458.595	2668.793	3103.323	3343.978	0.307	0.274	0.488	0.526
SI	Extremadura	125.314	133.42	180.827	192.429	0.232	0.149	0.628	0.725
NO	Extremadura	87.191	78.026	119.294	111.54	0.192	0.201	0.535	0.473
SI	Galicia	7903.644	8000.13	9784.318	9832.281	0.428	0.432	0.439	0.43
NO	Galicia	6105.442	6446.141	7667.908	8402.77	0.282	0.235	0.452	0.435
SI	La Rioja	928.078	972.151	1151.67	1215.015	0.484	0.482	0.47	0.544
NO	La Rioja	809.641	830.93	1027.846	1046.561	0.304	0.305	0.459	0.496
SI	Region de Murcia	574.577	577.845	696.272	691.132	0.224	0.329	0.593	0.552
NO	Region de Murcia	552.844	583.427	694.041	721.138	0.258	0.278	0.522	0.551

Figure 5.12. Metrics in PLSR

As can be seen in the table, many R^2 values have reasonably improved, even leading to a positive R^2 for Navarra, which was previously negative. Therefore, this model seems to be useful for this dataset.

5.5. Generation of scenarios

Once it has been seen that the PLSR technique works well for the dataset, it is continued with the generation of scenarios.

To achieve this, it will be performed dimensionality reduction by using Principal Component Analysis (PCA), in order to facilitate the input space density estimation using a Gaussian Mixture Model (GMM). Subsequently, the generation of random scenarios would be produced by sampling from the GMM model, that shall had learnt the patterns in the data and would produce feasible synthetic input samples.

Dimensionality Reduction with PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms a set of possibly correlated variables into a set of linearly uncorrelated variables called principal components. The transformation is defined as follows:

1. Calculate the mean of the dataset

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

   where $\mathbf{x}_i$ is the data vector and n is the number of samples.

2. Subtract the mean

$$\mathbf{X}_{\text{centered}} = \mathbf{X} - \mu$$

3. Compute the covariance matrix

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}_{\text{centered}}^T\mathbf{X}_{\text{centered}}$$

4. Compute the eigenvalues and eigenvectors of the covariance matrix

$$\mathbf{C}\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

   where $\mathbf{v}_i$ are the eigenvectors and λ_i are the eigenvalues.

5. Select the top k eigenvectors corresponding to the top k largest eigenvalues to form the principal component matrix

$$\mathbf{W}_{\text{PCA}} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$$

6. Transform the original data

$$\mathbf{Z} = \mathbf{X}_{\text{centered}}\mathbf{W}_{\text{PCA}}$$

   where $\mathbf{Z}$ is the representation of the data in the principal component space.

With this technique, the variance captured by each component is as follows:

	Exp_variance	cum_Exp_variance
PC1	4.112382e-01	0.411238
PC2	2.529109e-01	0.664149
PC3	1.483095e-01	0.812459
PC4	8.500649e-02	0.897465
PC5	4.515749e-02	0.942623
PC6	2.612127e-02	0.968744
PC7	1.644053e-02	0.985184
PC8	1.344990e-02	0.998634
PC9	1.365747e-03	1.000000
PC10	9.384911e-33	1.000000

Figure 5.13. Variance captured by each component in PCA

It can be observed that only with 5 components, we can capture near to 95% of the variance of the original data.

After the PCA space has been constructed, it can be seen the projection of the data on PCA1 and PCA2, differentiating by color the autonomous community:

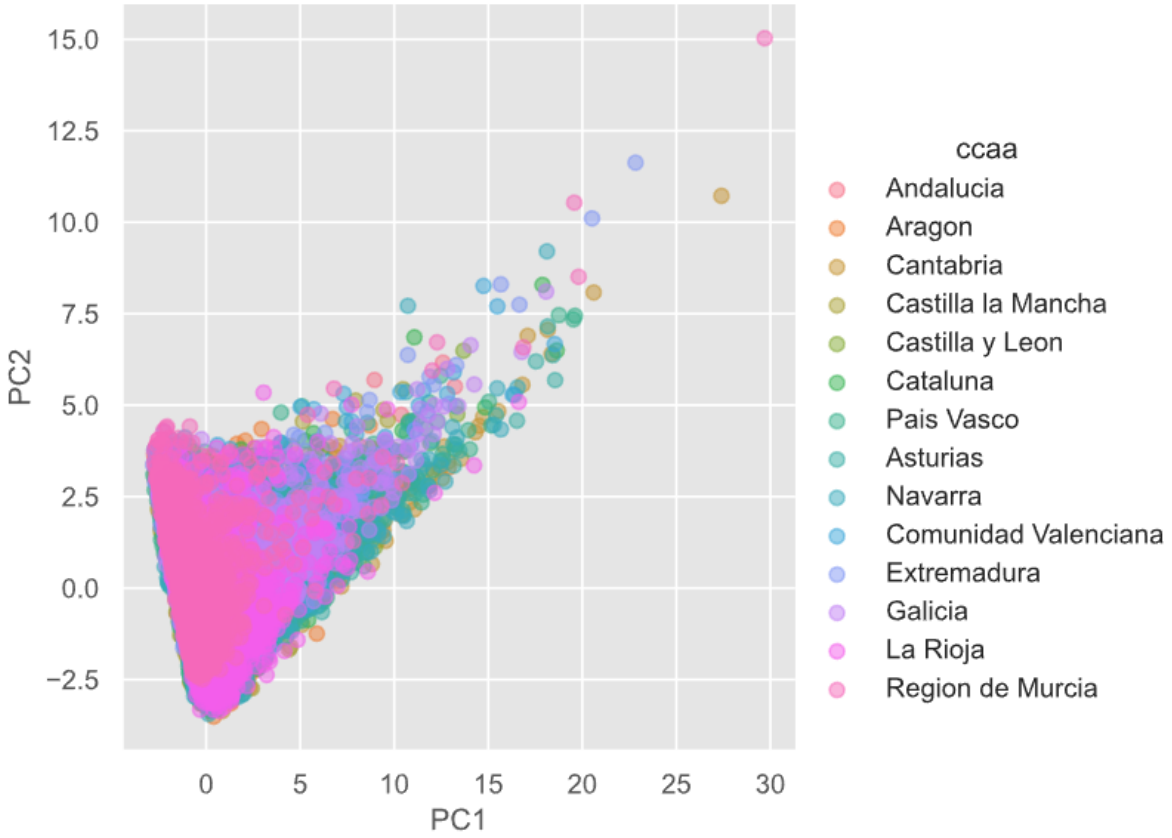


Figure 5.14. Data projection

To gain a deeper understanding of the PCA space, it is essential to plot at least the first three principal components, as shown in figure 5.15:

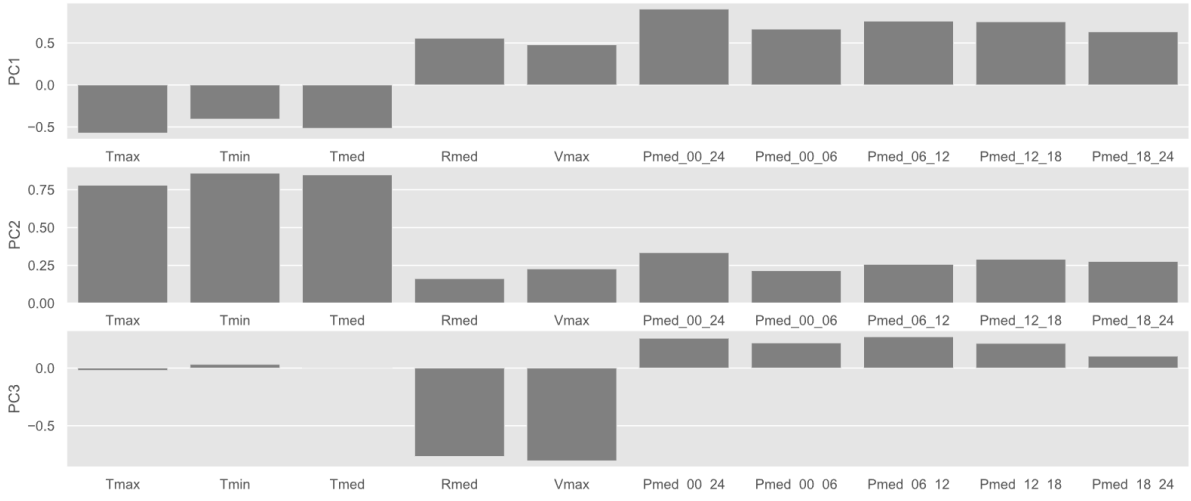


Figure 5.15. Interpretation of the first three PCA

PC1 is primarily a temperature-related component, where maximum temperature, minimum and mean temperatures influence it negatively. On the other hand, Wind speed and precipitation measurements contribute positively.

PC2 represents a component primarily influenced by temperature variables, with maximum, minimum, and mean temperatures all positively contributing to it. It suggests a relationship

where higher temperature measurements are grouped together, distinguishing them from other variables.

PC3 is influenced significantly by wind speed and precipitation measurements. It shows a negative relationship with gust and wind speed, suggesting that it might be representing weather patterns where high precipitations are associated with lower gust and wind speed.

One can see this PCA as a sort of scenario classifier, where PC1 states the general behavior of every day in Spain, where days with heavier rain correlates with lower temperatures and greater wind speed. PC2 provides incremental information above PC1, differentiating those days where, for the same amount of rain, temperatures might be greater or lower than the expected by PC1. PC3 provides even more information, differentiating those days whose wind speed is much higher or lower than the expected with the information of PC1 and PC2. Therefore, a day with heavy rain, high temperature and no wind would have a high value of PC1, PC2 and PC3; while a day with almost rain, medium temperature and a little wind would have a null value of PC1, PC2 and PC3, and a day with no rain, cold and high wind would have negative values of PC1, PC2 and PC3.

The task remaining is to first, locate the real scenario in the PCA space and then generate believable scenarios in the neighborhood of the point. To do so, we would use a GMM model.

Space Partitioning with GMM

The Gaussian Mixture Model (GMM) is a probabilistic model that assumes the data is a mixture of several Gaussian distributions. The probability density function for a GMM with k components is given by:

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j)$$

where:

- π_j is the weight of the j-th component, with $\sum_{j=1}^{k} \pi_j = 1$.
- $\mathcal{N}(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j)$ is the multivariate normal distribution with mean μ_j and covariance matrix $\mathbf{\Sigma}_j$.

The parameters π_j, μ_j and $\mathbf{\Sigma}_j$ are estimated using the Expectation-Maximization (EM) algorithm, which iteratively maximizes the likelihood of the observed data given the Gaussian components.

Following the parameter estimation, the decision to utilize **six Gaussian distributions** in the Gaussian Mixture Model (GMM) was guided by two main factors: the Bayesian Information Criterion (**BIC**) and a three-dimensional visualization of the data using Principal Component Analysis (**PCA**). The BIC was employed to determine the number of Gaussian components, as it provides a measure that balances model complexity against the fit to the data. By evaluating models with varying numbers of Gaussian distributions, it was observed that the model incorporating six Gaussians resulted in the lowest BIC value, as it can be seen in figure 5.16. This indicated that six components offered an optimal compromise between adequately fitting the data and avoiding overfitting.

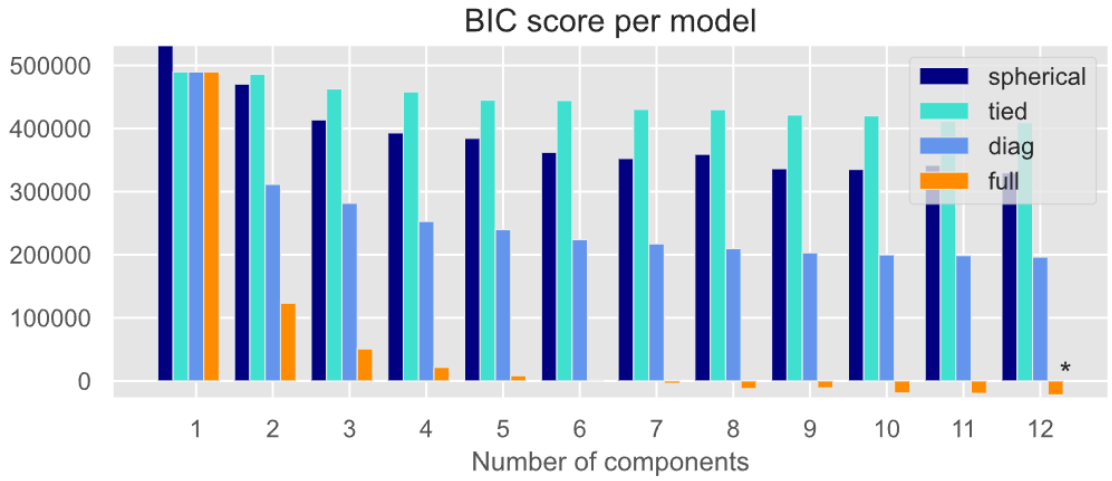


Figure 5.16. BIC

Additionally, a three-dimensional visualization of the data using PCA was conducted to further validate the choice of six Gaussian distributions. This method allowed for a visual inspection of how well the Gaussian components represented the data's density and structure in a reduced dimensional space. The visualization (represented in figures 5.17, 5.18 and 5.19) demonstrated that the use of six Gaussians effectively captured the data's clusters and distribution patterns, suggesting a good fit.

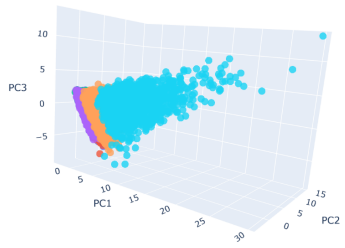


Figure 5.17. View 1

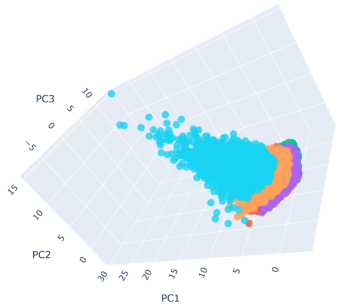


Figure 5.18. View 2

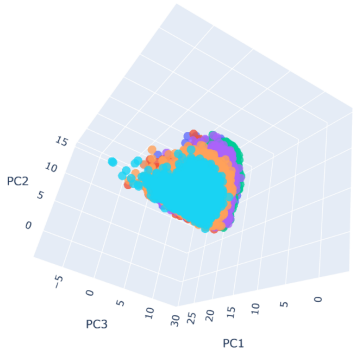


Figure 5.19. View 3

Thus, the selection of six Gaussian distributions was justified both by the statistical evidence from the BIC and the visual confirmation from the PCA-based analysis, ensuring an accurate and interpretable model of the data's underlying structure.

Generating Random Scenarios

Once the GMM is trained, we can generate new random scenarios by sampling from the Gaussian mixture.

The first hypothesis is that, we generate new random scenarios taking each day from the previous year. The process consists of the following steps:

1. Select each day of the previous year and pass it through the scaler and the transformer.
2. Calculate the membership probabilities for each point in relation to each GMM component. For example, day i has a probability π_j of being a member of the component j.

3. Scale those probabilities so that for each point i, the sum of the membership probabilities across all GMM components equals 1.
4. Generate scenarios using a weighted uniform distribution based on the membership probabilities, selecting points from each component according to these probabilities.

In the developed code, 1000 scenarios have been selected for each day.

Reversing the PCA Projection

To reverse the projection back to the original space, we use the inverse transformation of PCA. If $\mathbf{Z}_{\text{new}}$ are the new points generated in the reduced space:

$$\mathbf{X}_{\text{new}} = \mathbf{Z}_{\text{new}} \mathbf{W}_{\text{PCA}}^{T} + \mu$$

Prediction with PLSR

Finally, we apply Partial Least Squares Regression (PLSR) on the dataset projected back to the original space to make predictions. The PLSR model applied to the point will depend on the selected autonomous community and whether it rains or not.

PLSR finds the linear relationship between the input matrix $\mathbf{X}$ and the output matrix $\mathbf{Y}$.

The basic equation of PLSR is:

$$\mathbf{Y} = \mathbf{XB} + \mathbf{E}$$

where:

- $\mathbf{B}$ is the matrix of regression coefficients,
- $\mathbf{E}$ is the error term.

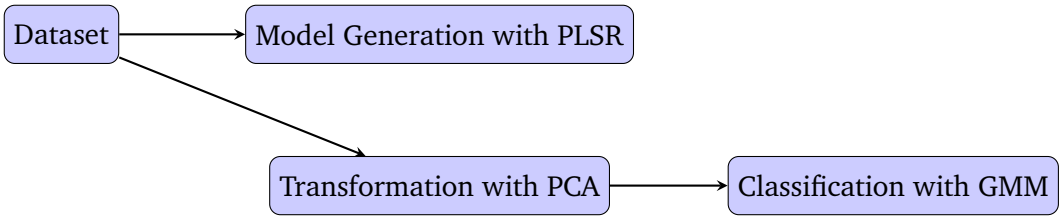
The PLSR algorithm finds $\mathbf{B}$ by maximizing the covariance between the projections of $\mathbf{X}$ and $\mathbf{Y}$.

All of this has been developed in code 7.9.

5.6. Final Methodology

After the mathematical explanation, the following diagram illustrates the methodology that has been followed:

Model generation:



Scenario generation and prediction:

To validate this methodology, it has been tested with various autonomous communities. For illustrative purposes, two examples will be presented to demonstrate its application in a scenarios involving *Galicia* and *Castilla y León*.

A scenario of $n = 1000$ simulations was created for each day following the previously mentioned procedure. A day is considered rainy when $'Pmed\_00\_24' \geq 1$, and non-rainy otherwise. To measure its accuracy, the global MAE and the mean RMSE was used. The following results were obtained:

	Galicia	***Castilla y León***
Global MAE	10.903,62	18.655,44
mean RMSE	13.772,22	24.215,17

Table 5.1. Summary statistics for *Galicia* and *Castilla y León*.

As observed, both metrics vary across the autonomous communities. The Global MAE and mean RMSE values are notably lower in Galicia compared to Castilla y León, indicating better predictive accuracy in the former region. This disparity could be attributed to regional differences in climate patterns, data distribution, or model performance specific to each area.

These results highlight the importance of considering regional characteristics when evaluating prediction models, as their accuracy may differ depending on the location. Further analysis could explore the factors driving these differences and assess whether model adjustments could improve performance in areas with higher error metrics.

The graphics below are presenting the actual production for *Galicia* and *Castilla y León* in black, in blue the predicted value by the PLSR based on the hypothesis that the following year behaves like the previous one, and the 50%, 75%, and 90% confidence intervals of the simulations.

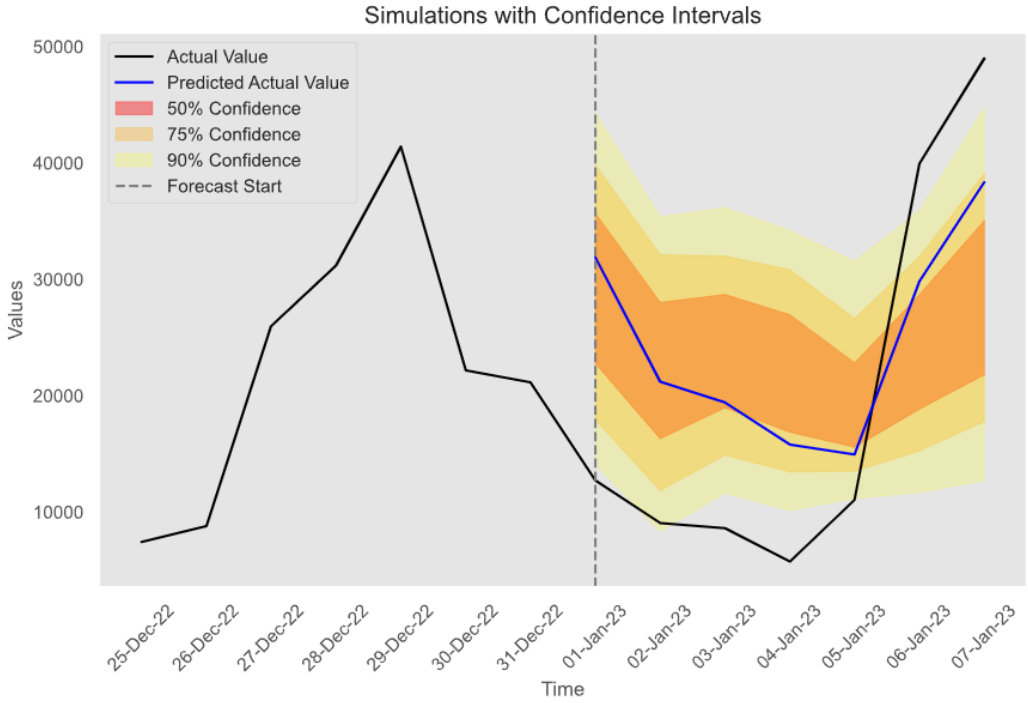


Figure 5.20. Confidence intervals for the energy production in *Galicia*

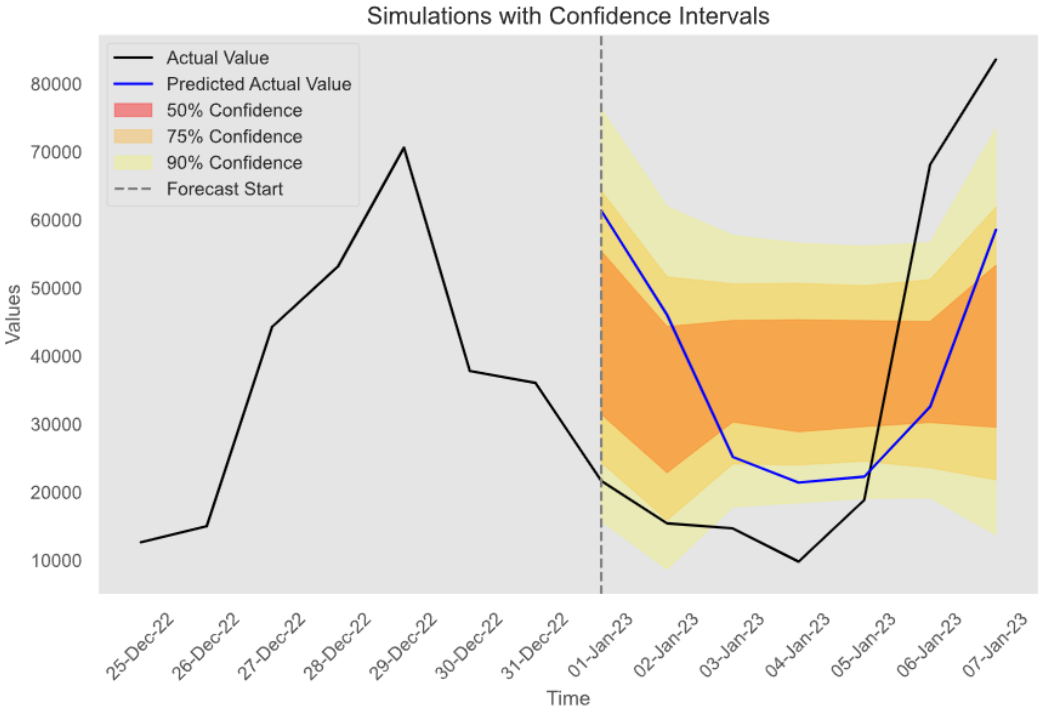


Figure 5.21. Confidence intervals for the energy production in *Castilla y León*

As observed, daily predictions are not sufficiently accurate. Therefore, it is proposed to perform aggregations, such as monthly aggregations, to improve the accuracy.

6

Conclusions

Success doesn't mean the absence of failures; it means the attainment of ultimate objectives. It means winning the war, not every battle.
Edwin Bliss (1912—2002)

This last chapter summarizes the developments of this dissertation. The main conclusions that can be drawn from the experiments carried out are set forth and the most original contributions are highlighted. Finally, the open issues that have not been tackled in the thesis, as well as possible future research lines, are discussed.

6.1. Summary and conclusions

In this thesis, we have explored the application of probabilistic machine learning techniques for medium-term forecasting of electric production, with a specific focus on wind energy. The increasing integration of renewable energy sources into the power grid necessitates reliable forecasting models to ensure stability and efficiency. Through the analysis of various machine learning models, we aimed to predict monthly wind energy production over a one-year horizon.

Our research demonstrated that probabilistic models, which provide a range of possible outcomes along with associated probabilities, are particularly effective in capturing the inherent uncertainty in renewable energy production. These models were evaluated and compared against traditional deterministic approaches, highlighting their superiority in terms of accuracy and reliability.

Key findings from our study include the identification of meteorological factors as critical predictors for wind energy production, and the successful application of probabilistic models such as PLSR, PCA and GMM in generating reliable forecasts. The comparative analysis with existing models confirmed that our proposed approach offers a new methodology to produce probability forecasts, making it a valuable tool for energy planners and policymakers.

However, prediction accuracy of the model is not sufficient and is left as future development the enhancement of the prediction model with date aggregation or data enhancement.

In conclusion, this thesis contributes to the field of renewable energy forecasting by presenting a robust and practical methodology for medium-term wind energy production forecasting. The findings underscore the importance of probabilistic forecasting in managing the complexities of renewable energy integration and pave the way for further advancements in this domain.

6.2. Original contributions

The original contributions of this thesis are manifold and highlight its significance in the field of renewable energy forecasting. Firstly, this work introduces a novel application of probabilistic machine learning techniques to the specific problem of medium-term wind energy production forecasting. Unlike traditional deterministic models, the probabilistic approach provides a comprehensive view of possible outcomes, accounting for the inherent uncertainty in wind energy generation.

Secondly, the thesis identifies key meteorological factors that significantly impact wind energy production, providing valuable insights for future research and model development. The integration of these factors into the forecasting models has proven to enhance the accuracy and reliability of the predictions.

Thirdly, the comparative analysis with existing models demonstrates the superior performance of the proposed probabilistic approach, showcasing its potential to improve decision-making processes in energy planning and management.

Finally, the practical application of the methodology to a real-world use case underlines its relevance and effectiveness, offering a scalable and adaptable solution for medium-term energy forecasting in various contexts.

6.3. Future work

The scope of this research opens up several avenues for future work, aimed at further refining and expanding the capabilities of probabilistic energy forecasting models.

1. Model Enhancement: Future studies could focus on improving the existing models by incorporating additional data sources, such as real-time weather forecasts, economic indicators, and grid operational data. Enhancing the models' predictive accuracy through advanced feature selection and model optimization techniques remains a key area for development.

2. Integration with Other Renewable Sources: Expanding the methodology to include other renewable energy sources, such as solar and hydro, could provide a more comprehensive forecasting tool. This integration would help in creating a unified model capable of predicting the combined output of various renewable sources.

3. Short-Term Forecasting: While this thesis focuses on medium-term forecasting, extending the approach to short-term horizons could offer significant benefits for operational planning and grid management. Developing models that can provide accurate forecasts on a daily or hourly basis could further enhance grid stability and efficiency.

4. Real-World Implementation: Applying the developed models in real-world scenarios, such as utility companies and energy market operations, would be an important next step. This practical implementation could provide valuable feedback and insights for model refinement and validation.

By addressing these areas, future research can build on the foundations laid by this thesis, driving further advancements in the field of renewable energy forecasting and contributing to a more sustainable and efficient energy future.

7

List of codes

Code 1

```
# Import libraries
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

sheet_name = 0

# Load the variables
galicia_provinces = ['A Coruña', 'Lugo', 'Ourense', 'Pontevedra']
asturias_provinces = ['Asturias']
cantabria_provinces = ['Cantabria']
pais_vasco_provinces = ['Bizkaia', 'Gipuzkoa', 'Araba/Álava']
navarra_provinces = ['Navarra']
aragon_provinces = ['Huesca', 'Zaragoza', 'Teruel']
cataluna_provinces = ['Girona', 'Lleida', 'Barcelona', 'Tarragona']
comunidad_valenciana_provinces = ['Castelló/Castellón', 'València/Valencia',
    ↪ 'Alacant/Alicante']
region_murcia_provinces = ['Murcia']
andalucia_provinces = ['Almería', 'Granada', 'Jaén', 'Córdoba', 'Málaga',
    ↪ 'Cádiz', 'Sevilla', 'Huelva']
extremadura_provinces = ['Cáceres', 'Badajoz']
castilla_leon_provinces = ['León', 'Zamora', 'Palencia', 'Valladolid', 'Ávila',
    ↪ 'Burgos', 'Segovia', 'Soria']
la_rioja_provinces = ['La Rioja']
castilla_mancha_provinces = ['Guadalajara', 'Cuenca', 'Albacete', 'Toledo',
    ↪ 'Ciudad Real']
madrid_provinces = ['Madrid']

# Create a dictionary with the data
Aemet = {'Dia': [], 'ccaa': [], 'Tmax': [], 'Tmin': [], 'Tmed': [], 'Rmed': [],
    ↪ 'Vmax':[], 'Pmed_00_24': [], 'Pmed_00_06': [], 'Pmed_06_12': [],
    ↪ 'Pmed_12_18': [], 'Pmed_18_24': []}
# Create the DataFrame
Aemet = pd.DataFrame(Aemet)
# Put the column names into the columns
Aemet.columns = ['Dia', 'ccaa', 'Tmax', 'Tmin', 'Tmed', 'Rmed', 'Vmax',
    ↪ 'Pmed_00_24', 'Pmed_00_06', 'Pmed_06_12', 'Pmed_12_18', 'Pmed_18_24']
```

```
31
32 # Create a dictionary with the data and create the DataFrame
33 Aemet_aux = pd.DataFrame(Aemet)
34 # Put the column names into the columns
35 Aemet_aux.columns = ['Dia', 'ccaa', 'Tmax', 'Tmin', 'Tmed', 'Rmed', 'Vmax',
       ↪ 'Pmed_00_24', 'Pmed_00_06', 'Pmed_06_12', 'Pmed_12_18', 'Pmed_18_24']
36
37 # Load the time data
38 start_date = datetime(2022, 9, 29) #change the date to the first day of the data
39
40 end_date = datetime(2023, 12, 31) #change the date to the last day of the data
41 date_range = pd.date_range(start=start_date, end=end_date, freq='D')
42
43 for date in date_range:
44     file_name = 'Aemet{}.xls'.format(date.strftime('%Y-%m-%d'))
45     xls_file = pd.ExcelFile(file_name)
46     df = xls_file.parse(sheet_name)
47
48     # Eliminate the first 3 rows
49     df = df.drop([0, 1, 2])
50
51     # Take the values of the first row
52     names = df.iloc[0]
53
54     # Eliminate the 1st row
55     df = df.iloc[1:]
56     df.columns = names #put the name to the colums
57     df = df.reset_index(drop=True)
58
59     #Eliminate the Nan values
60     df = df.dropna()
61
62     #Transform the columns to its type and clean the data
63     df["Temperatura máxima (ºC)"] = df["Temperatura máxima
       ↪ (ºC)"].str.extract(r"(\d+\.\d+)")
64     df["Temperatura mínima (ºC)"] = df["Temperatura mínima
       ↪ (ºC)"].str.extract(r"(\d+\.\d+)")
65     df["Velocidad máxima (km/h)"] = df["Velocidad máxima
       ↪ (km/h)"].str.extract(r"(\d+)")
66     df["Racha (km/h)"] = df["Racha (km/h)"].str.extract(r"(\d+)")
67     df["Velocidad máxima (km/h)"] = df["Velocidad máxima
       ↪ (km/h)"].str.extract(r"(\d+)")
68
69
70     df["Temperatura máxima (ºC)"] = df["Temperatura máxima (ºC)"].astype(float)
71     df["Temperatura mínima (ºC)"] = df["Temperatura mínima (ºC)"].astype(float)
72     df["Temperatura media (ºC)"] = df["Temperatura media (ºC)"].astype(float)
73     df["Precipitación 00-24h (mm)"] = df["Precipitación 00-24h
       ↪ (mm)"].astype(float)
74     df["Precipitación 00-06h (mm)"] = df["Precipitación 00-06h
       ↪ (mm)"].astype(float)
75     df["Precipitación 06-12h (mm)"] = df["Precipitación 06-12h
       ↪ (mm)"].astype(float)
76     df["Precipitación 12-18h (mm)"] = df["Precipitación 12-18h
       ↪ (mm)"].astype(float)
77     df["Precipitación 18-24h (mm)"] = df["Precipitación 18-24h
       ↪ (mm)"].astype(float)
78     df["Velocidad máxima (km/h)"] = df["Velocidad máxima (km/h)"].astype(float)
79     df["Racha (km/h)"] = df["Racha (km/h)"].astype(float)
```

```
80
81      #Select the unique values of the column Provincia
82      df = df[~df['Provincia'].isin(['Illes Balears', 'Las Palmas', 'Ceuta',
        ↪ 'Melilla', 'Santa Cruz de Tenerife'])]
83
84      galicia_df = df[df['Provincia'].isin(galicia_provinces)]
85      asturias_df = df[df['Provincia'].isin(asturias_provinces)]
86      cantabria_df = df[df['Provincia'].isin(cantabria_provinces)]
87      pais_vasco_df = df[df['Provincia'].isin(pais_vasco_provinces)]
88      navarra_df = df[df['Provincia'].isin(navarra_provinces)]
89      aragon_df = df[df['Provincia'].isin(aragon_provinces)]
90      cataluna_df = df[df['Provincia'].isin(cataluna_provinces)]
91      comunidad_valenciana_df =
        ↪ df[df['Provincia'].isin(comunidad_valenciana_provinces)]
92      region_murcia_df = df[df['Provincia'].isin(region_murcia_provinces)]
93      andalucia_df = df[df['Provincia'].isin(andalucia_provinces)]
94      extremadura_df = df[df['Provincia'].isin(extremadura_provinces)]
95      castilla_leon_df = df[df['Provincia'].isin(castilla_leon_provinces)]
96      la_rioja_df = df[df['Provincia'].isin(la_rioja_provinces)]
97      castilla_mancha_df = df[df['Provincia'].isin(castilla_mancha_provinces)]
98      madrid_df = df[df['Provincia'].isin(madrid_provinces)]
99
100     spain = [galicia_df, asturias_df, cantabria_df, pais_vasco_df, navarra_df,
        ↪ aragon_df, cataluna_df, comunidad_valenciana_df, region_murcia_df,
        ↪ andalucia_df, extremadura_df, castilla_leon_df, la_rioja_df,
        ↪ castilla_mancha_df, madrid_df]
101     comunidades_autonomas = ['Galicia', 'Asturias', 'Cantabria', 'Pais Vasco',
        ↪ 'Navarra', 'Aragon', 'Cataluna', 'Comunidad Valenciana', 'Region de
        ↪ Murcia', 'Andalucia', 'Extremadura', 'Castilla y Leon', 'La Rioja',
        ↪ 'Castilla la Mancha', 'Madrid']
102
103     Aemet_aux.drop(Aemet_aux.index, inplace=True)
104     cont = 0
105     for i in spain:
106         provincias = i['Provincia'].unique()
107         Tmax_list = []
108         Tmin_list = []
109         Tmed_list = []
110         Rmax_list = []
111         Vmed_list = []
112         Pmed_00_24_list = []
113         Pmed_00_06_list = []
114         Pmed_06_12_list = []
115         Pmed_12_18_list = []
116         Pmed_18_24_list = []
117
118         for j in provincias:
119             aux = i[i['Provincia'] == j]
120             Tmax = aux['Temperatura máxima (ºC)'].max()
121             Tmin = aux['Temperatura mínima (ºC)'].min()
122             Tmed = aux['Temperatura media (ºC)'].mean()
123             Rmax = aux['Racha (km/h)'].max()
124             Vmed = aux['Velocidad máxima (km/h)'].mean()
125             Pmed_00_24 = aux['Precipitación 00-24h (mm)'].mean()
126             Pmed_00_06 = aux['Precipitación 00-06h (mm)'].mean()
127             Pmed_06_12 = aux['Precipitación 06-12h (mm)'].mean()
128             Pmed_12_18 = aux['Precipitación 12-18h (mm)'].mean()
129             Pmed_18_24 = aux['Precipitación 18-24h (mm)'].mean()
130
```

```
            Tmax_list.append(Tmax)
            Tmin_list.append(Tmin)
            Tmed_list.append(Tmed)
            Rmax_list.append(Rmax)
            Vmed_list.append(Vmed)
            Pmed_00_24_list.append(Pmed_00_24)
            Pmed_00_06_list.append(Pmed_00_06)
            Pmed_06_12_list.append(Pmed_06_12)
            Pmed_12_18_list.append(Pmed_12_18)
            Pmed_18_24_list.append(Pmed_18_24)

        Aemet_aux = pd.concat([Aemet_aux, pd.DataFrame({'Dia':
    ↪ [date.strftime('%Y-%m-%d')],
        'ccaa': [comunidades_autonomas[cont]],
        'Tmax': [np.mean(Tmax_list)],
        'Tmin': [np.mean(Tmin_list)],
        'Tmed': [np.mean(Tmed_list)],
        'Rmed': [np.mean(Rmax_list)],
        'Vmax': [np.mean(Vmed_list)],
        'Pmed_00_24': [np.mean(Pmed_00_24_list)],
        'Pmed_00_06': [np.mean(Pmed_00_06_list)],
        'Pmed_06_12': [np.mean(Pmed_06_12_list)],
        'Pmed_12_18': [np.mean(Pmed_12_18_list)],
        'Pmed_18_24': [np.mean(Pmed_18_24_list)]})], ignore_index=False)
        cont += 1

    Aemet = pd.concat([Aemet, Aemet_aux], ignore_index=False)
Aemet.to_csv('tramox.csv', index=False)
```

Listing 7.1. Meteorological Aggregations by Autonomous Community

Code 2

```
import json
import requests
import csv
import pandas as pd
# Now we are going to obtain the energy production for la peninsula

sd = ['2018-01-02T00:00', '2019-01-02T00:00', '2020-01-02T00:00',
    ↪ '2021-01-02T00:00', '2022-01-02T00:00', '2023-01-02T00:00']
ed = ['2019-01-02T00:00', '2020-01-02T00:00', '2021-01-02T00:00',
    ↪ '2022-01-02T00:00', '2023-01-02T00:00', '2024-01-02T00:00']

def extract(attributes):
    return attributes['values']

def extract_value(title_list):
    return [item['value'] for item in title_list]

def extract_date(title_list):
    return [item['datetime'] for item in title_list]

data_list = []
for s, e in zip(sd, ed):
    url = f"https://apidatos.ree.es/es/datos/generacion/estructura-renovables?
    ↪ start_date={s}&end_date={e}&time_trunc=day&geo_trunc=electric_system&
    ↪ geo_limit=peninsular&geo_ids=8741"
```

```
    print(url)
    response = requests.get(url)
    print(response)
    data = response.json()
    print(data)

    df = pd.DataFrame(data['included'])
    df = df[['type', 'attributes']]
    df['title'] = df['attributes'].apply(extract)

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

    df['value'] = df['title'].apply(extract_value)
    df['date'] = df['title'].apply(extract_date)

    df.drop(columns=['title'], inplace=True)
    filtered_df = df[df['type'].isin(['Eólica'])]

    new_data = []
    for _, row in filtered_df.iterrows():
        for value, date in zip(row['value'], row['date']):
            new_data.append({'type': row['type'], 'value': value, 'date': date})

    new_df = pd.DataFrame(new_data)

    new_df['date'] = pd.to_datetime(new_df['date'], utc=True)
    new_df['Month'] = pd.to_datetime(new_df['date']).dt.month
    new_df['Year'] = pd.to_datetime(new_df['date']).dt.year

    data_list.append(new_df)

result_df = pd.concat(data_list)
result_df.to_csv('eolica.csv', index=False)
```

Listing 7.2. Request of the production of wind power energy

Code 3

```
import requests
import csv
import pandas as pd

#We are going to stract the potencias instaladas
ccaa = [ '4', '5', '6', '7', '8', '9', '10', '11', '13', '14', '15', '16', '17',
    ↪ '20', '21']
sd = ['2018-01-01T00:01', '2020-01-01T00:01', '2022-01-01T00:01']
ed = ['2020-01-01T00:00', '2022-01-01T00:00', '2024-01-01T00:00']

def extract(attributes):
    return attributes['values']

def extract_value(title_list):
    return [item['value'] for item in title_list]

def extract_date(title_list):
    return [item['datetime'] for item in title_list]

data_list = []
```

```
20
21 for c in ccaa:
22     for s, e in zip(sd, ed):
23         url = f"https://apidatos.ree.es/es/datos/generacion/potencia-instalada?
    ↪ start_date={s}&end_date={e}&time_trunc=month&geo_limit=ccaa&geo_ids={c}"
24         response = requests.get(url)
25         data = response.json()
26
27         df = pd.DataFrame(data['included'])
28         df = df[['type', 'attributes']]
29         df['title'] = df['attributes'].apply(extract)
30
31         df.drop(columns=['attributes'], inplace=True)
32
33         df['value'] = df['title'].apply(extract_value)
34         df['date'] = df['title'].apply(extract_date)
35
36         df.drop(columns=['title'], inplace=True)
37         filtered_df = df[df['type'].isin(['Eólica'])]
38
39         new_data = []
40         for _, row in filtered_df.iterrows():
41             for value, date in zip(row['value'], row['date']):
42                 new_data.append({'type': row['type'], 'value': value, 'date':
    ↪ date})
43
44         new_df = pd.DataFrame(new_data)
45
46         new_df['date'] = pd.to_datetime(new_df['date'], utc=True)
47         new_df['Month'] = pd.to_datetime(new_df['date']).dt.month
48         new_df['Year'] = pd.to_datetime(new_df['date']).dt.year
49
50         new_df['ccaa'] = c
51
52         data_list.append(new_df)
53
54 result_df = pd.concat(data_list)
55
56 #We have finished the request, I split the data in 3 different files to make it
    ↪ easier to work with them
57 pt_eo = result_df[result_df['type'] == 'Eólica']
58
59 pt_eo.to_csv('potencia_instalada_eo.csv', index=False)
```

Listing 7.3. Request of the data of installed capacity

Code 4

```
1 '''
2 With the previous df we can start to make the analysis. In the same date, we are
    ↪ going to
3 see the proportion of the different types of energy in the different regions.
4 '''
5 import pandas as pd
6
7 pt_inst_eo = pd.read_csv('potencia_instalada_eo.csv', sep=';', decimal=',')
8 pt_inst_eo_date = pt_inst_eo.groupby([ 'Year', 'Month']).sum('value')
9 pt_inst_eo_date.rename(columns={'value': 'total'}, inplace=True)
```

```
10
11 result_eo = pd.merge(pt_inst_eo, pt_inst_eo_date, on=['Year', 'Month'],
    ↪ how='left')
12 result_eo['percentage'] = result_eo['value'] / result_eo['total']
13 result_eo.drop(columns=['ccaa_y'], inplace=True)
14 result_eo.rename(columns={'ccaa_x': 'ccaa'}, inplace=True)
15 result_eo.to_csv('pot_inst_eo_ccaa.csv', index=False)
```

Listing 7.4. Weighting of the installed capacity

Code 5

```
1  import pandas as pd
2  from scipy.interpolate import lagrange
3
4  tramo1 = pd.read_csv('tramo1.csv')
5  tramo2 = pd.read_csv('tramo2.csv')
6  tramo3 = pd.read_csv('tramo3.csv')
7  tramo4 = pd.read_csv('tramo4.csv')
8
9  ccaa = pd.unique(tramo1['ccaa'])
10 column_names = tramo4.columns
11 column_names = column_names.drop('Dia')
12 column_names = column_names.drop('ccaa')
13
14 #TRAMO 1
15 AEMET = pd.concat([tramo1], axis=0)
16 missing_val = ['2020-03-05', '2020-03-06', '2020-03-07']
17 aux = pd.DataFrame(columns = tramo1.columns)
18
19 for i in missing_val:
20     for j in ccaa:
21         new_rows = pd.DataFrame({'Dia': [i], 'ccaa': [j]})
22         aux = pd.concat([aux, new_rows], ignore_index=True)
23
24 AEMET = pd.concat([AEMET, aux], axis=0)
25
26 # TRAMO 2
27 AEMET = pd.concat([AEMET, tramo2], axis=0)
28 missing_val = ['2020-07-26']
29 aux = pd.DataFrame(columns = tramo1.columns)
30
31 for i in missing_val:
32     for j in ccaa:
33         new_rows = pd.DataFrame({'Dia': [i], 'ccaa': [j]})
34         aux = pd.concat([aux, new_rows], ignore_index=True)
35
36 AEMET = pd.concat([AEMET, aux], axis=0)
37
38 # TRAMO 3
39 AEMET = pd.concat([AEMET, tramo3], axis=0)
40 missing_val = ['2022-09-27', '2022-09-28']
41 aux = pd.DataFrame(columns = tramo1.columns)
42
43 for i in missing_val:
44     for j in ccaa:
45         new_rows = pd.DataFrame({'Dia': [i], 'ccaa': [j]})
46         aux = pd.concat([aux, new_rows], ignore_index=True)
```

```
47
48 AEMET = pd.concat([AEMET, aux], axis=0)
49 AEMET = pd.concat([AEMET, tramo4], axis=0)
50
51 #Recuperar datos - 2020: 03/05, 03/06, 03/07  con una interpolacion lineal entre
       ↪ 03/04 y 03/08
52 inicio = tramo1.loc[tramo1['Dia'] == '2020-03-04']
53 fin = tramo2.loc[tramo2['Dia'] == '2020-03-08']
54
55 x= [1,5]
56
57 for i in ccaa:
58     filtered_row_i = inicio[inicio['ccaa'] == i]
59     filtered_row_f = fin[fin['ccaa'] == i]
60
61     filtered_row_i.drop('Dia', axis=1, inplace=True)
62     filtered_row_i.drop('ccaa', axis=1, inplace=True)
63
64     filtered_row_f.drop('Dia', axis=1, inplace=True)
65     filtered_row_f.drop('ccaa', axis=1, inplace=True)
66
67
68     for j in column_names:
69         y = [filtered_row_i[j].values[0], filtered_row_f[j].values[0]]
70         p = lagrange(x, y)
71         AEMET.loc[(AEMET['Dia'] == '2020-03-05') & (AEMET['ccaa'] == i), j] = p(2)
72         AEMET.loc[(AEMET['Dia'] == '2020-03-06') & (AEMET['ccaa'] == i), j] = p(3)
73         AEMET.loc[(AEMET['Dia'] == '2020-03-07') & (AEMET['ccaa'] == i), j] = p(4)
74
75 AEMET.to_csv('AEMET.csv', index=False)
76
77 #Recuperar datos - 2020: 07/26  con una interpolacion lineal entre 07/25 y 07/27
78 inicio = tramo2.loc[tramo2['Dia'] == '2020-07-25']
79 fin = tramo3.loc[tramo3['Dia'] == '2020-07-27']
80
81 x= [1,3]
82
83 for i in ccaa:
84     filtered_row_i = inicio[inicio['ccaa'] == i]
85     filtered_row_f = fin[fin['ccaa'] == i]
86
87     filtered_row_i.drop('Dia', axis=1, inplace=True)
88     filtered_row_i.drop('ccaa', axis=1, inplace=True)
89
90     filtered_row_f.drop('Dia', axis=1, inplace=True)
91     filtered_row_f.drop('ccaa', axis=1, inplace=True)
92
93
94     for j in column_names:
95         y = [filtered_row_i[j].values[0], filtered_row_f[j].values[0]]
96         p = lagrange(x, y)
97         AEMET.loc[(AEMET['Dia'] == '2020-07-26') & (AEMET['ccaa'] == i), j] = p(2)
98
99 #Recuperar datos - 2020: 09/27, 09/28  con una interpolacion lineal entre 09/26 y
       ↪ 09/29
100 inicio = tramo3.loc[tramo3['Dia'] == '2022-09-26']
101 fin = tramo4.loc[tramo4['Dia'] == '2022-09-29']
102
103 x= [1,4]
```

```

for i in ccaa:
    filtered_row_i = inicio[inicio['ccaa'] == i]
    filtered_row_f = fin[fin['ccaa'] == i]

    filtered_row_i.drop('Dia', axis=1, inplace=True)
    filtered_row_i.drop('ccaa', axis=1, inplace=True)

    filtered_row_f.drop('Dia', axis=1, inplace=True)
    filtered_row_f.drop('ccaa', axis=1, inplace=True)


    for j in column_names:
        y = [filtered_row_i[j].values[0], filtered_row_f[j].values[0]]
        p = lagrange(x, y)
        AEMET.loc[(AEMET['Dia'] == '2022-09-27') & (AEMET['ccaa'] == i), j] = p(2)
        AEMET.loc[(AEMET['Dia'] == '2022-09-28') & (AEMET['ccaa'] == i), j] = p(3)

AEMET.to_csv('AEMET.csv', index=False)
```

Listing 7.5. Lagrange interpolation for missing values in meteorological data

Code 6

```
### Load libraries ###
import pandas as pd

# plotting libraries
import matplotlib.pyplot as plt

# Data management libraries
import numpy as np # linear algebra

# # Machine learning libraries
from sklearn.model_selection import train_test_split, GridSearchCV,
    ↪ cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error,
    ↪ mean_absolute_percentage_error

# others
from mltools.regression_tools import LinearRegressor
import math

# Load data
meteo = pd.read_csv("AEMET.csv")

# Remove missing
meteo.dropna(inplace=True)

## Add day of the week
meteo['Dia'] = pd.to_datetime(meteo['Dia'])
meteo['day_of_week'] = meteo['Dia'].dt.day_name()

### Convert necessary variable to factor
meteo.day_of_week = meteo.day_of_week.astype('category')
```

```
33
34 ## Month
35 meteo['month'] = meteo['Dia'].dt.month
36 meteo.month = meteo.month.astype('category')
37 dataset = meteo.copy()
38 gen_eo = pd.read_csv('gen_eo_ccaa.csv')
39
40 gen_eo = gen_eo.replace(4, "Andalucia")
41 gen_eo = gen_eo.replace(5, "Aragon")
42 gen_eo = gen_eo.replace(6, "Cantabria")
43 gen_eo = gen_eo.replace(7, "Castilla la Mancha")
44 gen_eo = gen_eo.replace(8, "Castilla y Leon")
45 gen_eo = gen_eo.replace(9, "Cataluna")
46 gen_eo = gen_eo.replace(10, "Pais Vasco")
47 gen_eo = gen_eo.replace(11, "Asturias")
48 gen_eo = gen_eo.replace(8744, "Comunidad de Ceuta")
49 gen_eo = gen_eo.replace(8745, "Comunidad de Melilla")
50 gen_eo = gen_eo.replace(13, "Madrid")
51 gen_eo = gen_eo.replace(14, "Navarra")
52 gen_eo = gen_eo.replace(15, "Comunidad Valenciana")
53 gen_eo = gen_eo.replace(16, "Extremadura")
54 gen_eo = gen_eo.replace(17, "Galicia")
55 gen_eo = gen_eo.replace(8743, "Islas Baleares")
56 gen_eo = gen_eo.replace(8742, "Islas Canarias")
57 gen_eo = gen_eo.replace(20, "La Rioja")
58 gen_eo = gen_eo.replace(21, "Region de Murcia")
59
60 gen_eo['date'] = pd.to_datetime(gen_eo['date'])
61
62 dataset.rename(columns={'Dia': 'date'}, inplace=True)
63
64 eolica = pd.merge(gen_eo, dataset, on=['date', 'ccaa'], how='inner')
65
66 INPUTS = ['Tmax', 'Tmin', 'Tmed', 'Rmed', 'Vmax', 'Pmed_00_24', 'Pmed_00_06',
    ↪ 'Pmed_06_12', 'Pmed_12_18', 'Pmed_18_24']
67 OUTPUT = 'gen'
68 errores = pd.DataFrame()
69 for i in eolica.ccaa.unique():
70     lluviosos = eolica[(eolica.ccaa == i) & (eolica.Pmed_00_24 >= 1)]
71     X = lluviosos[INPUTS]
72     y = lluviosos[OUTPUT]
73
74     # Split
75     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    ↪ random_state=0) #seed for replication
76     ## Create dataset to store model predictions
77
78     dfTR_eval = X_train.copy()
79     dfTR_eval['gen'] = y_train
80     dfTS_eval = X_test.copy()
81     dfTS_eval['gen'] = y_test
82
83     ## Inputs of the model. Change accordingly to perform variable selection
84     INPUTS_LR_NUM =
    ↪ X_train.select_dtypes(include=['int64','float64']).columns.values.tolist()
85     INPUTS_LR_CAT =
    ↪ X_train.select_dtypes(include=['category']).columns.values.tolist()
86     INPUTS_LR = INPUTS_LR_NUM + INPUTS_LR_CAT
87
```

```
    # Prepare the numeric variables by imputing by scaling
    numeric_transformer = Pipeline(steps=[('scaler', StandardScaler())])

    # Prepare the categorical variables by encoding the categories
    categorical_transformer = Pipeline(steps=[('onehot',
    ↪ OneHotEncoder(handle_unknown='ignore'))])

    # Create a preprocessor to perform the steps defined above
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', numeric_transformer, INPUTS_LR_NUM),
            ('cat', categorical_transformer, INPUTS_LR_CAT)
            ])

    param = {}
    pipe = Pipeline([
                ('preprocessor', preprocessor),
                ('LR_model', LinearRegressor())
            ])
    # We use Grid Search Cross Validation to find the best parameter for the
    ↪ model in the grid defined
    nFolds = 10
    LR_fit = GridSearchCV(estimator=pipe, # Structure of the model to use
                        param_grid=param, # Defined grid to search in
                        n_jobs=-1, # Number of cores to use (parallelize)
                        scoring='neg_mean_squared_error', # RMSE
    ↪ https://scikit-learn.org/stable/modules/model_evaluation.html
                        cv=nFolds) # Number of Folds
    LR_fit.fit(X_train[INPUTS_LR], y_train) # Search in grid

    LR_fit.best_estimator_['LR_model'].summary(LR_fit
    ↪ .best_estimator_['preprocessor'].get_feature_names_out())  #information
    ↪ about the model trained
    dfTR_eval['LR_pred'] = LR_fit.predict(X_train)
    dfTS_eval['LR_pred'] = LR_fit.predict(X_test)

    #Training and test MAE - Mean Absolute error
    MAE_TR_LL = mean_absolute_error(dfTR_eval['gen'], dfTR_eval['LR_pred'])
    MAE_TS_LL = mean_absolute_error(dfTS_eval['gen'], dfTS_eval['LR_pred'])
    #Training and test RMSE - Root Mean Square Error
    RMSE_TR_LL = math.sqrt(mean_squared_error(dfTR_eval['gen'],
    ↪ dfTR_eval['LR_pred']))
    RMSE_TS_LL = math.sqrt(mean_squared_error(dfTS_eval['gen'],
    ↪ dfTS_eval['LR_pred']))
    #Training and test r^2
    R2_TR_LL = r2_score(dfTR_eval['gen'], dfTR_eval['LR_pred'])
    R2_TS_LL = r2_score(dfTS_eval['gen'], dfTS_eval['LR_pred'])
    #Training and test r^2
    MAPE_TR_LL = mean_absolute_percentage_error(dfTR_eval['gen'],
    ↪ dfTR_eval['LR_pred'])
    MAPE_TS_LL = mean_absolute_percentage_error(dfTS_eval['gen'],
    ↪ dfTS_eval['LR_pred'])


    No_lluviosos = eolica[(eolica.ccaa == i) & (eolica.Pmed_00_24 < 1)]
    X = No_lluviosos[INPUTS]
    y = No_lluviosos[OUTPUT]

    # Split
```

```
138     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
        ↪ #percentage of test data
139     random_state=0) #seed for replication
140     ## Create dataset to store model predictions
141
142     dfTR_eval = X_train.copy()
143     dfTR_eval['gen'] = y_train
144     dfTS_eval = X_test.copy()
145     dfTS_eval['gen'] = y_test
146
147     ## Inputs of the model. Change accordingly to perform variable selection
148     INPUTS_LR_NUM =
        ↪ X_train.select_dtypes(include=['int64','float64']).columns.values.tolist()
149     INPUTS_LR_CAT =
        ↪ X_train.select_dtypes(include=['category']).columns.values.tolist()
150     INPUTS_LR = INPUTS_LR_NUM + INPUTS_LR_CAT
151
152     # Prepare the numeric variables by imputing by scaling
153     numeric_transformer = Pipeline(steps=[('scaler', StandardScaler())])
154
155     # Prepare the categorical variables by encoding the categories
156     categorical_transformer = Pipeline(steps=[('onehot',
        ↪ OneHotEncoder(handle_unknown='ignore'))])
157
158     # Create a preprocessor to perform the steps defined above
159     preprocessor = ColumnTransformer(
160         transformers=[
161             ('num', numeric_transformer, INPUTS_LR_NUM),
162             ('cat', categorical_transformer, INPUTS_LR_CAT)
163             ])
164
165     param = {}
166     pipe = Pipeline([
167                 ('preprocessor', preprocessor),
168                 ('LR_model', LinearRegressor())
169             ])
170     # We use Grid Search Cross Validation to find the best parameter for the
        ↪ model in the grid defined
171     nFolds = 10
172     LR_fit = GridSearchCV(estimator=pipe, # Structure of the model to use
173                         param_grid=param, # Defined grid to search in
174                         n_jobs=-1, # Number of cores to use (parallelize)
175                         scoring='neg_mean_squared_error', # RMSE
        ↪ https://scikit-learn.org/stable/modules/model_evaluation.html
176                         cv=nFolds) # Number of Folds
177     LR_fit.fit(X_train[INPUTS_LR], y_train) # Search in grid
178
179     LR_fit.best_estimator_['LR_model'].summary(LR_fit
        ↪ .best_estimator_['preprocessor'].get_feature_names_out())  #information
        ↪ about the model trained
180     dfTR_eval['LR_pred'] = LR_fit.predict(X_train)
181     dfTS_eval['LR_pred'] = LR_fit.predict(X_test)
182     #Training and test MAE - Mean Absolute error
183     MAE_TR_NLL = mean_absolute_error(dfTR_eval['gen'], dfTR_eval['LR_pred'])
184     MAE_TS_NLL = mean_absolute_error(dfTS_eval['gen'], dfTS_eval['LR_pred'])
185     #Training and test RMSE - Root Mean Square Error
186     RMSE_TR_NLL = math.sqrt(mean_squared_error(dfTR_eval['gen'],
        ↪ dfTR_eval['LR_pred']))
```

```
    RMSE_TS_NLL = math.sqrt(mean_squared_error(dfTS_eval['gen'],
    ↪ dfTS_eval['LR_pred']))
    #Training and test r^2
    R2_TR_NLL = r2_score(dfTR_eval['gen'], dfTR_eval['LR_pred'])
    R2_TS_NLL = r2_score(dfTS_eval['gen'], dfTS_eval['LR_pred'])
    #Mape
    MAPE_TR_NLL = mean_absolute_percentage_error(dfTR_eval['gen'],
    ↪ dfTR_eval['LR_pred'])
    MAPE_TS_NLL = mean_absolute_percentage_error(dfTS_eval['gen'],
    ↪ dfTS_eval['LR_pred'])

    data = {
    'Precipitación': ['SI', 'NO'],
    'CCAA': [i, i],
    'Mae TR': [MAE_TR_LL, MAE_TR_NLL],
    'Mae TS': [MAE_TS_LL, MAE_TS_NLL],
    'RMSE TR': [RMSE_TR_LL, RMSE_TR_NLL],
    'RMSE TS': [RMSE_TS_LL, RMSE_TS_NLL],
    'R2 TR': [R2_TR_LL, R2_TR_NLL],
    'R2 TS': [R2_TS_LL, R2_TS_NLL],
    'MAPE TR': [MAPE_TR_LL, MAPE_TR_NLL],
    'MAPE TS': [MAPE_TS_LL, MAPE_TS_NLL],
    }

    # Crear el DataFrame
    df = pd.DataFrame(data)

    errores = pd.concat([errores, df], ignore_index=True)

errores = errores.round(3)

# Configurar la figura
fig, ax = plt.subplots(figsize=(10, 4))  # Ajustar el tamaño de la figura según
    ↪ sea necesario
ax.axis('tight')
ax.axis('off')

# Crear la tabla
table = ax.table(cellText=errores.values, colLabels=errores.columns,
    ↪ cellLoc='center', loc='center')
table.auto_set_font_size(False)
table.set_fontsize(9)

for key, cell in table.get_celld().items():
    if key[0] == 0:
        cell.set_fontsize(9)  # Ajustar el tamaño de la fuente de la cabecera
        cell.set_text_props(weight='bold')

col_idx = errores.columns.get_loc('CCAA')
for key, cell in table.get_celld().items():
    if key[1] == col_idx:
        cell.set_width(0.18)

col_idx = errores.columns.get_loc('Precipitación')
for key, cell in table.get_celld().items():
    if key[1] == col_idx:
        cell.set_width(0.12)

# Save the figure
```

```
plt.savefig('errores.png', bbox_inches='tight', dpi=300)

# Mostrar la figura
plt.show()
```

Listing 7.6. Linear Regression model for rainy and non-rainy days

Code 7

```
errores = errores.round(3)

# Configurar la figura
fig, ax = plt.subplots(figsize=(10, 4))  # Ajustar el tamaño de la figura según sea necesario
ax.axis('tight')
ax.axis('off')

# Crear la tabla
table = ax.table(cellText=errores.values, colLabels=errores.columns, cellLoc='center', loc='center')
table.auto_set_font_size(False)
table.set_fontsize(9)

for key, cell in table.get_celld().items():
    if key[0] == 0:
        cell.set_fontsize(9)  # Ajustar el tamaño de la fuente de la cabecera
        cell.set_text_props(weight='bold')

col_idx = errores.columns.get_loc('CCAA')
for key, cell in table.get_celld().items():
    if key[1] == col_idx:
        cell.set_width(0.18)

col_idx = errores.columns.get_loc('Precipitación')
for key, cell in table.get_celld().items():
    if key[1] == col_idx:
        cell.set_width(0.12)

# Guardar la figura
plt.savefig('errores.png', bbox_inches='tight', dpi=300)

# Mostrar la figura
plt.show()

errores_lag = errores_lag.round(3)

# Configurar la figura
fig, ax = plt.subplots(figsize=(10, 4))  # Ajustar el tamaño de la figura según sea necesario
ax.axis('tight')
ax.axis('off')

# Crear la tabla
table = ax.table(cellText=errores_lag.values, colLabels=errores_lag.columns, cellLoc='center', loc='center')
table.auto_set_font_size(False)
table.set_fontsize(9)

```

```
for key, cell in table.get_celld().items():
    if key[0] == 0:
        cell.set_fontsize(9)  # Ajustar el tamaño de la fuente de la cabecera
        cell.set_text_props(weight='bold')

col_idx = errores_lag.columns.get_loc('ccaa')
for key, cell in table.get_celld().items():
    if key[1] == col_idx:
        cell.set_width(0.18)

# Ajustar el ancho de las columnas 'R2' y 'MAPE'
r2_idx = errores_lag.columns.get_loc('R2')
mape_idx = errores_lag.columns.get_loc('MAPE')

for key, cell in table.get_celld().items():
    if key[1] == r2_idx or key[1] == mape_idx:
        cell.set_width(0.1)  # Ajustar el ancho según sea necesario

rmse_idx = errores_lag.columns.get_loc('RMSE')
for key, cell in table.get_celld().items():
    if key[1] == rmse_idx:
        cell.set_width(0.1)

plt.savefig('errores_lag.png', bbox_inches='tight', dpi=300)

plt.show()
```

Listing 7.7. Model with lag

Code 8

```
### Load libraries ###
import pandas as pd

# plotting libraries
import matplotlib.pyplot as plt

# Data management libraries
import numpy as np # linear algebra

# # Machine learning libraries
from sklearn.model_selection import train_test_split, GridSearchCV,
    ↪ cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error,
    ↪ mean_absolute_percentage_error

# others
from mltools.regression_tools import LinearRegressor
import math
from sklearn.cross_decomposition import PLSRegression

# Load data
meteo = pd.read_csv("AEMET.csv")

# Remove missing
```

```
meteo.dropna(inplace=True)

## Add day of the week
meteo['Dia'] = pd.to_datetime(meteo['Dia'])
meteo['day_of_week'] = meteo['Dia'].dt.day_name()

### Convert necessary variable to factor
meteo.day_of_week = meteo.day_of_week.astype('category')

## Month
meteo['month'] = meteo['Dia'].dt.month
meteo.month = meteo.month.astype('category')
dataset = meteo.copy()
gen_eo = pd.read_csv('gen_eo_ccaa.csv')

gen_eo = gen_eo.replace(4, "Andalucia")
gen_eo = gen_eo.replace(5, "Aragon")
gen_eo = gen_eo.replace(6, "Cantabria")
gen_eo = gen_eo.replace(7, "Castilla la Mancha")
gen_eo = gen_eo.replace(8, "Castilla y Leon")
gen_eo = gen_eo.replace(9, "Cataluna")
gen_eo = gen_eo.replace(10, "Pais Vasco")
gen_eo = gen_eo.replace(11, "Asturias")
gen_eo = gen_eo.replace(8744, "Comunidad de Ceuta")
gen_eo = gen_eo.replace(8745, "Comunidad de Melilla")
gen_eo = gen_eo.replace(13, "Madrid")
gen_eo = gen_eo.replace(14, "Navarra")
gen_eo = gen_eo.replace(15, "Comunidad Valenciana")
gen_eo = gen_eo.replace(16, "Extremadura")
gen_eo = gen_eo.replace(17, "Galicia")
gen_eo = gen_eo.replace(8743, "Islas Baleares")
gen_eo = gen_eo.replace(8742, "Islas Canarias")
gen_eo = gen_eo.replace(20, "La Rioja")
gen_eo = gen_eo.replace(21, "Region de Murcia")

gen_eo['date'] = pd.to_datetime(gen_eo['date'])

dataset.rename(columns={'Dia': 'date'}, inplace=True)

eolica = pd.merge(gen_eo, dataset, on=['date', 'ccaa'], how='inner')

INPUTS = ['Tmax', 'Tmin', 'Tmed', 'Rmed', 'Vmax', 'Pmed_00_24', 'Pmed_00_06',
    ↪ 'Pmed_06_12', 'Pmed_12_18', 'Pmed_18_24']
OUTPUT = 'gen'
errores = pd.DataFrame()
for i in eolica.ccaa.unique():
    lluviosos = eolica[(eolica.ccaa == i) & (eolica.Pmed_00_24 >= 1)]
    X = lluviosos[INPUTS]
    y = lluviosos[OUTPUT]

    # Split
    X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                        test_size=0.2,
    ↪ #percentage of test data
                                                        random_state=0) #seed for
    ↪ replication
    ## Create dataset to store model predictions

    dfTR_eval = X_train.copy()
```

```
    dfTR_eval['gen'] = y_train
    dfTS_eval = X_test.copy()
    dfTS_eval['gen'] = y_test

    ## Inputs of the model. Change accordingly to perform variable selection
    INPUTS_LR_NUM =
    ↪ X_train.select_dtypes(include=['int64','float64']).columns.values.tolist()
    INPUTS_LR_CAT =
    ↪ X_train.select_dtypes(include=['category']).columns.values.tolist()
    INPUTS_LR = INPUTS_LR_NUM + INPUTS_LR_CAT

    # Prepare the numeric variables by imputing by scaling
    numeric_transformer = Pipeline(steps=[('scaler', StandardScaler())])

    # Prepare the categorical variables by encoding the categories
    categorical_transformer = Pipeline(steps=[('onehot',
    ↪ OneHotEncoder(handle_unknown='ignore'))])

    # Create a preprocessor to perform the steps defined above
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', numeric_transformer, INPUTS_LR_NUM),
            ('cat', categorical_transformer, INPUTS_LR_CAT)
            ])

    param = {'PLSR_model__n_components': [1,2,3,4,5,6,7,8]}
    pipe = Pipeline([
                ('preprocessor', preprocessor),
                ('PLSR_model', PLSRegression())
            ])
    # We use Grid Search Cross Validation to find the best parameter for the
    ↪ model in the grid defined
    nFolds = 10
    PLSR_fit = GridSearchCV(estimator=pipe, # Structure of the model to use
                        param_grid=param, # Defined grid to search in
                        n_jobs=-1, # Number of cores to use (parallelize)
                        scoring='neg_mean_squared_error', # RMSE
    ↪ https://scikit-learn.org/stable/modules/model_evaluation.html
                        cv=nFolds) # Number of Folds
    PLSR_fit.fit(X_train[INPUTS_LR], y_train) # Search in grid

    # PLSR_fit.best_estimator_['PLSR_model'].summary(PLSR_fit
    ↪ .best_estimator_['preprocessor'].get_feature_names_out())  #information
    ↪ about the model trained
    dfTR_eval['LR_pred'] = PLSR_fit.predict(X_train)
    dfTS_eval['LR_pred'] = PLSR_fit.predict(X_test)

    #Training and test MAE - Mean Absolute error
    MAE_TR_LL = mean_absolute_error(dfTR_eval['gen'], dfTR_eval['LR_pred'])
    MAE_TS_LL = mean_absolute_error(dfTS_eval['gen'], dfTS_eval['LR_pred'])
    #Training and test RMSE - Root Mean Square Error
    RMSE_TR_LL = math.sqrt(mean_squared_error(dfTR_eval['gen'],
    ↪ dfTR_eval['LR_pred']))
    RMSE_TS_LL = math.sqrt(mean_squared_error(dfTS_eval['gen'],
    ↪ dfTS_eval['LR_pred']))
    #Training and test r^2
    R2_TR_LL = r2_score(dfTR_eval['gen'], dfTR_eval['LR_pred'])
    R2_TS_LL = r2_score(dfTS_eval['gen'], dfTS_eval['LR_pred'])
    #Training and test r^2
```

```
132     MAPE_TR_LL = mean_absolute_percentage_error(dfTR_eval['gen'],
        ↪ dfTR_eval['LR_pred'])
133     MAPE_TS_LL = mean_absolute_percentage_error(dfTS_eval['gen'],
        ↪ dfTS_eval['LR_pred'])
134
135
136     No_lluviosos = eolica[(eolica.ccaa == i) & (eolica.Pmed_00_24 < 1)]
137     X = No_lluviosos[INPUTS]
138     y = No_lluviosos[OUTPUT]
139
140     # Split
141     X_train, X_test, y_train, y_test = train_test_split(X, y,
142                                                         test_size=0.2,
        ↪ #percentage of test data
143                                                         random_state=0) #seed for
        ↪ replication
144     ## Create dataset to store model predictions
145
146     dfTR_eval = X_train.copy()
147     dfTR_eval['gen'] = y_train
148     dfTS_eval = X_test.copy()
149     dfTS_eval['gen'] = y_test
150
151     ## Inputs of the model. Change accordingly to perform variable selection
152     INPUTS_LR_NUM =
        ↪ X_train.select_dtypes(include=['int64','float64']).columns.values.tolist()
153     INPUTS_LR_CAT =
        ↪ X_train.select_dtypes(include=['category']).columns.values.tolist()
154     INPUTS_LR = INPUTS_LR_NUM + INPUTS_LR_CAT
155
156     # Prepare the numeric variables by imputing by scaling
157     numeric_transformer = Pipeline(steps=[('scaler', StandardScaler())])
158
159     # Prepare the categorical variables by encoding the categories
160     categorical_transformer = Pipeline(steps=[('onehot',
        ↪ OneHotEncoder(handle_unknown='ignore'))])
161
162     # Create a preprocessor to perform the steps defined above
163     preprocessor = ColumnTransformer(
164         transformers=[
165             ('num', numeric_transformer, INPUTS_LR_NUM),
166             ('cat', categorical_transformer, INPUTS_LR_CAT)
167             ])
168
169     param = {'PLSR_model__n_components': [1,2,3,4,5,6,7,8]}
170     pipe = Pipeline([
171                 ('preprocessor', preprocessor),
172                 ('PLSR_model', PLSRegression())
173             ])
174     # We use Grid Search Cross Validation to find the best parameter for the
        ↪ model in the grid defined
175     nFolds = 10
176     PLSR_fit = GridSearchCV(estimator=pipe, # Structure of the model to use
177                         param_grid=param, # Defined grid to search in
178                         n_jobs=-1, # Number of cores to use (parallelize)
179                         scoring='neg_mean_squared_error', # RMSE
        ↪ https://scikit-learn.org/stable/modules/model_evaluation.html
180                         cv=nFolds) # Number of Folds
181     PLSR_fit.fit(X_train[INPUTS_LR], y_train) # Search in grid
```

```
182
183     # PLSR_fit.best_estimator_['PLSR_model'].summary(PLSR_fit
        ↪ .best_estimator_['preprocessor'].get_feature_names_out())  #information
        ↪ about the model trained
184     dfTR_eval['LR_pred'] = PLSR_fit.predict(X_train)
185     dfTS_eval['LR_pred'] = PLSR_fit.predict(X_test)
186     #Training and test MAE - Mean Absolute error
187     MAE_TR_NLL = mean_absolute_error(dfTR_eval['gen'], dfTR_eval['LR_pred'])
188     MAE_TS_NLL = mean_absolute_error(dfTS_eval['gen'], dfTS_eval['LR_pred'])
189     #Training and test RMSE - Root Mean Square Error
190     RMSE_TR_NLL = math.sqrt(mean_squared_error(dfTR_eval['gen'],
        ↪ dfTR_eval['LR_pred']))
191     RMSE_TS_NLL = math.sqrt(mean_squared_error(dfTS_eval['gen'],
        ↪ dfTS_eval['LR_pred']))
192     #Training and test r^2
193     R2_TR_NLL = r2_score(dfTR_eval['gen'], dfTR_eval['LR_pred'])
194     R2_TS_NLL = r2_score(dfTS_eval['gen'], dfTS_eval['LR_pred'])
195     #Mape
196     MAPE_TR_NLL = mean_absolute_percentage_error(dfTR_eval['gen'],
        ↪ dfTR_eval['LR_pred'])
197     MAPE_TS_NLL = mean_absolute_percentage_error(dfTS_eval['gen'],
        ↪ dfTS_eval['LR_pred'])
198
199     data = {
200     'Precipitación': ['SI', 'NO'],
201     'CCAA': [i, i],
202     'Mae TR': [MAE_TR_LL, MAE_TR_NLL],
203     'Mae TS': [MAE_TS_LL, MAE_TS_NLL],
204     'RMSE TR': [RMSE_TR_LL, RMSE_TR_NLL],
205     'RMSE TS': [RMSE_TS_LL, RMSE_TS_NLL],
206     'R2 TR': [R2_TR_LL, R2_TR_NLL],
207     'R2 TS': [R2_TS_LL, R2_TS_NLL],
208     'MAPE TR': [MAPE_TR_LL, MAPE_TR_NLL],
209     'MAPE TS': [MAPE_TS_LL, MAPE_TS_NLL],
210     }
211
212     # Crear el DataFrame
213     df = pd.DataFrame(data)
214
215     errores = pd.concat([errores, df], ignore_index=True)
216
217 errores = errores.round(3)
218
219 # Configurar la figura
220 fig, ax = plt.subplots(figsize=(10, 4))  # Ajustar el tamaño de la figura según
        ↪ sea necesario
221 ax.axis('tight')
222 ax.axis('off')
223
224 # Crear la tabla
225 table = ax.table(cellText=errores.values, colLabels=errores.columns,
        ↪ cellLoc='center', loc='center')
226 table.auto_set_font_size(False)
227 table.set_fontsize(9)
228
229 for key, cell in table.get_celld().items():
230     if key[0] == 0:
231         cell.set_fontsize(9)  # Ajustar el tamaño de la fuente de la cabecera
232         cell.set_text_props(weight='bold')
```

```

col_idx = errores.columns.get_loc('CCAA')
for key, cell in table.get_celld().items():
    if key[1] == col_idx:
        cell.set_width(0.18)

col_idx = errores.columns.get_loc('Precipitación')
for key, cell in table.get_celld().items():
    if key[1] == col_idx:
        cell.set_width(0.12)

# Save the figure
plt.savefig('errores.png', bbox_inches='tight', dpi=300)

plt.show()
```

Listing 7.8. PLSR model

Code 9

```
### Load libraries ###

# interactive plotting
%matplotlib inline
%config InlineBackend.figure_format = 'svg'

# plotting libraries
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
sns.set()
plt.style.use('ggplot')
plt.rcParams.update({'figure.figsize': (10, 7), 'figure.dpi': 120})

# Data management libraries
import itertools
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Machine learning libraries
from sklearn.decomposition import PCA, FastICA
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KernelDensity
import sklearn.mixture as mixture
from scipy import linalg
import math

# Other
from mltools import unsupervised_tools as UT
import joblib

# Cargar modelos de PCA, GMM y scaler
pca = joblib.load('pca_model.pkl')
gmm = joblib.load('gmm_model.pkl')
scaler = joblib.load('scaler_model.pkl')
PLSR_ll = joblib.load('PLSR_fit_ll_Galicia.pkl') # Cambiar esto en el futuro en
    ↪ funcion de ccaa
```

```
PLSR_nll = joblib.load('PLSR_fit_nll_Galicia.pkl') # Cambiar esto en el futuro en funcion de ccaa

# Cargar datos de prueba y nos quedamos con ccaa = Galicia, date = 2023-01-01 hasta 2023-12-31
data = pd.read_csv('eolica.csv', index_col=0)
data = data[(data['date'] >= '2023-01-01') & (data['date'] <= '2023-12-31')]
data = data[data['ccaa'] == 'Galicia']
data = data.drop(columns=['ccaa'])
n_dias = data.shape[0]
gen = data['gen']

# Seleccionar solo las columnas que fueron usadas durante el entrenamiento
numeric_inputs = ['Tmax','Tmin','Tmed','Rmed','Vmax','Pmed_00_24','Pmed_00_06','Pmed_06_12', 'Pmed_12_18','Pmed_18_24']
data = data[numeric_inputs]
data = data.reset_index(drop=True)

# Obtener scores de cada gausiana del gmm en base al punto, primeramente se pasa por el pca y el scaler
data_tr = scaler.transform(data)
data_tr = pca.transform(data_tr)

# Dado un punto:
# 1. Calcular las probabilidades de pertenencia para cada punto a cada componente del GMM
#   1.1 Obtener la probabilidad de pertenencia de cada punto a cada componente del GMM
scores_per_component = gmm.predict_proba(data_tr) * gmm.weights_

#   1.2 Escalar la probabilidad de pertenencia para que sumen 1
scores_per_component = scores_per_component / scores_per_component.sum(axis=1, keepdims=True)

# 2. Generar escenarios utilizando una distribucion uniforme ponderada por las probabilidades de pertenencia
#   2.1 Utilizar la funcion numpy.random.choice para seleccionar un componente para cada escenario
scenario_size = 1000
matrix_nsmpl_comp_day = np.empty((scenario_size, len(gmm.weights_)-1, 0))

for i in range(len(scores_per_component)): # Va iterando en cada dia
    #   2.2 La funcion numpy.random.choice estara ponderada en base a las probabilidades de pertenencia caluladas en el paso 1
    selected_components = np.random.choice(len(gmm.weights_), size=scenario_size, p=scores_per_component[i])

    generated_data = []
    #   2.3 Generar un escenario para cada componente seleccionado
    for component in selected_components:
            # Media y covarianza del componente seleccionado
            mean = gmm.means_[component]
            cov = gmm.covariances_[component]

            # Generar un punto desde la distribución gaussiana
            sample = np.random.multivariate_normal(mean, cov)
            generated_data.append(sample)

```

```
85        # Dimensiones: 1000x5x1
86
87      # Concatenar el vector a la matriz en la tercera dimensión
88      generated_data = np.array(generated_data)
89      generated_data = generated_data[:, :, np.newaxis]
90      matrix_nsmpl_comp_day = np.concatenate((matrix_nsmpl_comp_day,
        ↪ generated_data), axis=2)
91
92  #   2.4 Hacer la transformacion inversa y el escalado inverso para obtener los
        ↪ valores reales de los escenarios
93  numeric_inputs =
        ↪ ['Tmax','Tmin','Tmed','Rmed','Vmax','Pmed_00_24','Pmed_00_06','Pmed_06_12',
        ↪ 'Pmed_12_18','Pmed_18_24']
94
95  # Inverse transform the scenarios for each day, Duda: como hacer un dataframe de
        ↪ tres dimensiones
96  predictions = np.empty((n_dias, scenario_size))
97
98  for i in range(matrix_nsmpl_comp_day.shape[2]):
99    x_scenarios = pca.inverse_transform(matrix_nsmpl_comp_day[:,:,i])
100   x_scenarios = pd.DataFrame(scaler.inverse_transform(x_scenarios),
        ↪ columns=numeric_inputs)
101
102   # 3. Predecir las respuestas para los nuevos escenarios usando los modelos PLSR
103   y_pred_ll = np.empty((0, 1))
104   y_pred_nll = np.empty((0, 1))
105
106   x_escenarios_ll = x_scenarios[x_scenarios['Pmed_00_24'] >= 1]
107   x_escenarios_nll = x_scenarios[x_scenarios['Pmed_00_24'] < 1]
108
109   if x_escenarios_ll.shape[0] > 0:
110     y_pred_ll = PLSR_ll.predict(x_escenarios_ll)
111
112   if x_escenarios_nll.shape[0] > 0:
113     y_pred_nll = PLSR_nll.predict(x_escenarios_nll)
114
115   predictions[i] = np.concatenate((y_pred_ll, y_pred_nll), axis=0).flatten()
116
117 pred = np.empty((n_dias))
118
119 for i, row in data.iterrows():
120   x = row.to_frame().T
121   if x['Pmed_00_24'].values >= 1:
122     pred[i] = PLSR_ll.predict(x)
123   else:
124     pred[i] = PLSR_nll.predict(x)
125
126 predictions = np.vstack((pred, predictions.T)).T
```

Listing 7.9. Generation of scenarios

References

[AHL23] R. Alvarenga, H. Herbaux, and L. Linguet, "On the added value of state-of-the-art probabilistic forecasting methods applied to the optimal scheduling of a pv power plant with batteries", *Energies*, vol. 16, no. 18, 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/18/6543.

[Cli23] D. Clima, *Datos clima: Información meteorológica y climática*, https://datosclima.es/, Accessed: 2023-06-13, 2023.

[dEsp23] R. E. de España (REE), *Esios: Información del sistema eléctrico*, https://www.esios.ree.es/es, Accessed: 2023-06-13, 2023.

[DJK16] M. Dillig, M. Jung, and J. Karl, "The impact of renewables on electricity prices in germany–an estimation based on historic spot prices in the years 2011–2013", *Renewable and Sustainable Energy Reviews*, vol. 57, pp. 7–15, 2016.

[KH06] M. Korpas and A. T. Holen, "Operation planning of hydrogen storage connected to wind power operating in a power market", *IEEE transactions on energy conversion*, vol. 21, no. 3, pp. 742–749, 2006.

[MKW15] P. Manembu, A. Kewo, and B. Welang, "Missing data solution of electricity consumption based on lagrange interpolation case study: Intelligensia data monitoring", in *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*, IEEE, 2015, pp. 511–516.

[TR21] T. Thomas and E. Rajabi, "A systematic review of machine learning-based missing value imputation techniques", *Data Technologies and Applications*, vol. 55, no. 4, pp. 558–585, 2021.

[Xie+23] Y. Xie, C. Li, M. Li, F. Liu, and M. Taukenova, "An overview of deterministic and probabilistic forecasting methods of wind energy", *iScience*, vol. 26, no. 1, p. 105 804, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2589004222020776.

[Xue+14] Y. Xue *et al.*, "A review on impacts of wind power uncertainties on power systems", in *Zhongguo Dianji Gongcheng Xuebao/Proceedings of the Chinese Society of Electrical Engineering*, vol. 34, 2014, pp. 5029–5040.

[ZWW14] Y. Zhang, J. Wang, and X. Wang, "Review on probabilistic forecasting of wind power generation", *Renewable and Sustainable Energy Reviews*, vol. 32, pp. 255–270, 2014.