Universitat Autònoma de Barcelona

Facultat d'Economia i Empresa UAB

Treball de Final de Grau

Facultat d'Economia i Empresa

TÍTOL: Hacia un Futuro Energético Predecible: Aplicando Machine Learning para Prever los Precios de la Electricidad

AUTOR/A: *Carlos Romero Matarin*

TUTOR/A: Xavier Vilà

GRAU: Economia

DATA: *30/05/2023*

Resum

En aquest treball volem centrar-nos en l'anàlisi d'un component essencial en la societat moderna: el preu de l'electricitat. Amb l'objectiu de comprendre les complexitats del mercat elèctric i la seva dinàmica, l'estudi que hem realitzat ens proporcionarà una base teòrica sobre els articles i el mercat elèctric.

La investigació que hem dut a terme es basa en l'ús de diferents tècniques de Machine Learning per predir els preus en el mercat elèctric. Per poder realitzar els models, hem utilitzat diferents fonts de dades, incloent informació de mercat, dades meteorològiques i dades de generació d'energia. En aquest treball, explorem diferents models de Machine Learning, com models de regressió bayesiana, models de regressió regularitzats i xarxes neuronals, amb l'objectiu d'identificar el model més precís per la predicció de preus en el mercat energètic.

Hem centrat l'estudi en l'any 2019 per evitar les fluctuacions atípiques causades per la pandèmia de la COVID-19, però la metodologia i les tècniques de Machine Learning són flexibles i es podrien adaptar per incorporar dades addicionals en el futur.

L'objectiu principal d'aquest treball és aplicar tècniques de Machine Learning i tècniques estadístiques per predir el preu en el mercat elèctric, desenvolupant un sistema d'anàlisi utilitzant Python i Rstudio.

Com a avanç en aquest treball, s'ha trobat que el model de regressió bayesiana ha proporcionat les estimacions més precises, seguit del model de procés gaussià. Aquests models han demostrat una capacitat molt alta per capturar relacions no lineals i gestionar la incertesa de les dades, la qual cosa els converteix en eines útils per a la predicció de preus futurs.

S'espera que aquest treball pugui ajudar a la comparativa de diferents models de Machine Learning aplicats al mercat elèctric. Els resultats obtinguts poden servir com a referència per a investigadors i professionals interessats a utilitzar tècniques de Machine Learning per a l'anàlisi i predicció de preus en aquest sector.

Resumen

En este trabajo queremos centrados en el análisis de un componente esencial en la sociedad moderna: el precio de la electricidad. Con el objetivo de entender las complejidades del mercado eléctrico, el estudio que hemos realizado nos dejara una base teórica sobre los artículos y el del mercado eléctrico .

La investigación que hemos realizado se basa en el uso de diferentes técnicas de Machine Learning para predecir los precios en el mercado eléctrico. Par poder realizar los modelos hemos utilizado diferentes fuentes de datos, incluyendo información de mercado, datos meteorológicos y datos de generación de energía. En este trabajo exploremos diferentes modelos de Machine Learning, como modelos de regresión bayesiana, modelos de regresión regularizados y redes neuronales, con el objetivo de identificar el modelo más preciso para la predicción de precios en el mercado energético.

El estudio lo hemos centrado en el año 2019 para evitar las fluctuaciones atípicas causadas por la pandemia de COVID-19, pero la metodología y las técnicas de Machine Learning son flexibles y se podrían adaptar para incorporar datos adicionales en el futuro.

El objetivo principal de este trabajo es aplicar técnicas de Machine Learning y técnicas estadísticas para predecir el precio en el mercado eléctrico, desarrollando un sistema de análisis utilizando Python y Rstudio.

Como avance en este trabajo se ha encontrado que el modelo de regresión bayesiana ha proporcionado las estimaciones más precisas, seguido del modelo de proceso gaussiano. Estos modelos han demostrado una capacidad muy alta para capturar relaciones no lineales y manejar la incertidumbre de los datos, lo que los convierte en herramientas útiles para la predicción de precios futuros.

Se espera que este trabajo puede ayudar a la comparativa de diferentes modelos de Machine Learning aplicados al mercado eléctrico. Los resultados obtenidos pueden servir como referencia para investigadores y profesionales interesados en utilizar técnicas de Machine Learning para el análisis y predicción de precios en este sector.

Abstract

In this work, we want to focus on the analysis of an essential component in modern society: the price of electricity. With the aim of understanding the complexities of the electric market and its dynamics, the study we have carried out will provide us with a theoretical basis on the articles and the electricity market.

The research we have conducted is based on the use of different Machine Learning techniques to predict prices in the electricity market. To create the models, we have used different sources of data, including market information, weather data, and energy generation data. In this work, we explore different Machine Learning models, such as Bayesian regression models, regularized regression models, and neural networks, with the aim of identifying the most accurate model for price prediction in the energy market.

We have focused the study on the year 2019 to avoid atypical fluctuations caused by the COVID-19 pandemic, but the methodology and Machine Learning techniques are flexible and could be adapted to incorporate additional data in the future.

The main objective of this work is to apply Machine Learning techniques and statistical techniques to predict the price in the electricity market, developing an analysis system using Python and Rstudio.

As a progress in this work, it has been found that the Bayesian regression model has provided the most accurate estimates, followed by the Gaussian process model. These models have demonstrated a very high ability to capture non-linear relationships and manage data uncertainty, making them useful tools for future price prediction.

This work is expected to assist in the comparison of different Machine Learning models applied to the electricity market. The results obtained can serve as a reference for researchers and professionals interested in using Machine Learning techniques for the analysis and prediction of prices in this sector.

Acrónimos

DL: Deep Learning

MLP: Modelos lineales de Probabilidad

ML: Machine Learning

DNN: Deep Neural Network, redes neuronales profundas

LSTM: Long short-term memory networks, extensión de redes neuronales recurrentes

CNN: Convolutional Neural Networks

GWh: Giga watts hora

REE: Red Eléctrica de España

OMIE: Operador del Mercado Ibérico de Energía

AEMET: Agencia Estatal de Meteorología.

Contenido

Índice de Figuras

1 Introducción

Este trabajo surge como análisis de un componente importante para la sociedad actual, como es el precio de la electricidad. Podemos ver, que en los últimos años el sector energético ha tenido diferentes desafíos que solucionar, lo que ha provocado variaciones de los precios. Estos desafíos engloban desde la transición a fuentes de energía renovables, la implementación de nuevas tecnologías, nuevas regularizaciones, una demanda creciente, etc.

1.1 Contexto del trabajo

Para poder entender el contexto del trabajo debemos explicar primero la complejidad del entorno a analizar. El mercado eléctrico es un mercado muy dinámico en el cual participan múltiples agentes (generadores, distribuidores, comercializadores y consumidores) los cuales interactúan de manera individual como de manera conjunta entre ellos.

Otro tema que comentar es la importancia el recurso y los recursos a analizar, podemos ver la importancia de la electricidad y de los recursos que la generan en el funcionamiento de la sociedad y de la economía, actualmente la mayor parte de la empresa, por no decir el 100%, utilizan electricidad, por lo que podemos ver que su precio tiene impactos directos en la competencia de las empresas y en el bienestar de los consumidores.

Por otro lado, el análisis se realizará por medios de técnicas de machine learning, que podemos ver que en los últimos años se ha convertido en una herramienta cuotidiana de las empresas para el análisis de sus datos, en este caso nosotros utilizaremos técnicas para poder predecir los precios energéticos del mercado eléctrico, con la ventaja de ser rápidamente adaptables para agregar nuevos datos, estudios, etc. En este trabajo utilizaremos diferentes técnicas de regresiones como son la regresión línea, polinomial, etc. como también la creación de una pequeña red neuronal para el cálculo y la predicción de los precios. El trabajo también incluirá un análisis del preprocesamiento de datos y de las técnicas de selección de características utilizadas.

Como se comenta, se aplicarán diferentes técnicas de Machine Learning para predecir los precios en el mercado eléctrico. Para ello, se utilizarán diversas fuentes de datos, incluyendo información de mercado, datos meteorológicos y datos de generación de energía.

Me gustaría indicar que para el análisis del trabajo, se ha decidido centrarnos en el año 2019. Aunque es posible incluir más años en el análisis, se ha seleccionado 2019 por diferentes razones.

En primer lugar, el año 2019 es el año anterior a la pandemia de COVID-19, por lo que los datos no están afectados por las fluctuaciones atípicas de la situación global pudo haber causado en el mercado eléctrico (como demandas o generaciones atípicas). En

segundo lugar, los datos de 2019 están libremente completos y limpios, lo que facilita el análisis y la interpretación.

Pero es importante destacar que la metodología y las técnicas de machine learning utilizadas en este trabajo son flexibles y adaptables con la cual cosa nos permitirían incorporar datos adicionales en el futuro. Extrapolando lo dicho, podemos decir que si se necesitan, se pueden agregar más años al análisis, permitiendo así una visión más amplia y conseguir una mejor precisión de los precios del mercado eléctrico.

1.2 Antecedentes

En este apartado, se presenta una revisión de la literatura relacionada con el mercado energético y precios, con énfasis en el análisis de precios del mercado eléctrico. Se describirán los principales factores que influyen en la formación de precios, así como los modelos de predicción de precios más utilizados. Además, se presentarán las principales contribuciones y limitaciones de los trabajos previos en este ámbito.

Como un artículo de referencia podemos ver el articulo*" Electricity price forecasting: A review of the state-of-the-art with a look into the future. International Journal of Forecasting,"* de Rafal Weron. En este articulo el autor crea un marco en la predicción de precios de la electricidad utilizando diferentes técnicas de aprendizaje profundo y de machine learing. Este autor tiene diferentes artículos relacionados en el tema en este en concreto nos habla de los modelos DL (Deep learning) y explica:

- Integración del Mercado: Nos comenta que todos los modelos de DL tienen una característica en común, que es la integración del mercado, con esto intenta mejorar la precisión de la predicción, los diferentes modelos predicen simultáneamente los precios de la electricidad en varios mercados. Con el fin de que debido a la integración del mercado y a través de la multitarea, los modelos pueden aprender características más generales.

- Modelo DNN: El primer modelo que analiza es el de DL para predecir los precios del día siguiente es una red neuronal profunda dos capas ocultas. Es una simple extensión del MLP tradicional, y su entrada es un vector X con elementos x1,...,xn. Las capas ocultas tienen n1 y n2 neuronas respectivamente, y el vector de salida es p, que representa los precios que se pretenden prever.

- Modelo LSTM-DNN: El segundo modelo DL es un modelo híbrido que combina una red LSTM y una red DNN. La creación de esta estructura híbrida es incluir una capa recurrente que puede aprender y modelar las relaciones secuenciales en los datos de series de tiempo, así como una capa regular que puede aprender relaciones que dependen de datos no secuenciales. Con este objetivo en modelo, las entradas se dividen entre las que tienen datos de tiempo secuenciales, por ejemplo, precios de electricidad, y las que tienen datos regulares, como el día de la semana.

- Modelo GRU-DNN: El tercer modelo que nos presenta el autor de DL para predecir los precios, este modelo como el anterior es un modelo híbrido que combina una red GRU y una red DNN. Al igual que con la estructura híbrida LSTM-DNN, este modelo intenta incluir una capa que esté diseñada para datos secuenciales. Sin embargo, para reducir la carga computacional de la capa LSTM, se utiliza una capa GRU para modelar las secuencias de datos de tiempo.

- Modelo CNN: El cuarto modelo que nos presenta el auto de DL para predecir los precios es una red CNN. Al igual que en los dos casos anteriores, las entradas se dividen entre las que modelan datos secuenciales pasados y las que modelan información sobre el día. Pero en este caso, la separación es necesaria para agrupar datos con las mismas dimensiones como entradas para la misma CNN. En particular, los datos se dividen en dos partes, una colección secuencias de entrada usadas para los modelos híbridos y una nueva colección de vectores de entrada, donde cada valor representa alguna información futura de las 24 horas del día siguiente, como ejemplo la predicción del tiempo si las tenemos disponibles.

Otro documento pionero e importante en el campo de la predicción de precios es el libro *"Price forecasting using an integrated approach"* de Rafal Weron. Donde se nos expone diferentes temas y aspectos importantes del mercado eléctrico como:

- Orígenes del suministro eléctrico: El autor nos indica por medio del contexto histórico que en el inicio, las compañías eléctricas servían a áreas geográficas específicas, donde los consumidores estaban obligados a comprar su electricidad. Por lo que se consideró que la regulación centralizada era necesaria para garantizar la seguridad del suministro y la eficiencia de la producción.

- Desregulación y competencia: Pero el autor nos muestra que en las últimas dos décadas, la estructura del negocio eléctrico ha cambiado el mundo. Dese una situación monopolística hasta los mercados competitivos y desregulados, donde los consumidores, en teoría, son libres de elegir a su proveedor.

- Riesgos y volatilidad: Un tema importante en este trabajo es lo que el autor nos cometa sobre que en un mercado de energía competitivo, la electricidad se compra y vende a precios de mercado. Como consecuencia, el riesgo asumido por las empresas de servicios públicos, los productores de energía y los comercializadores ha aumentado sustancialmente.

- Análisis de series de tiempo: El autor nos da diferencia del análisis de muestras aleatorias de observaciones, el análisis de series temporales que nos muestra el autor se basa en la suposición de que los valores sucesivos representan mediciones consecutivas tomadas en intervalos de tiempo. Esto es importante para los datos del mercado de energía, ya que los precios al contado de la

electricidad, las cargas, las cifras de producción, etc., se toman 24 horas al día, 365 días al año.

Otro artículo importante para la predicción de precios de mercado es el articulo *"Electricity price forecasting on the day-ahead market using machine learning"* de Léonard Tschora . En este articulo el autor nos muestra diferentes técnicas de machine learning utilizadas para conseguir una predicción de precios del mercado eléctrico, entre los nos indica modelos como:

-Support Vector Machiness que son utilizados para la predicción de modelos multivariante, como modelos MiltiSVR, chainSvr.

-Métodos de regresiones de Random Forest que son modelos de árboles de decisiones con lo que poder crear subconjuntos de datos

-Métodos de conexiones Neuronales como se comenté en el artículo anterior métodos de Deep Learning.

Y un artículo que hemos utilizado pare la creación de este trabajo y como guía es el articulo *"Stock Price Prediction based on Multiple Regression Models"* de Y. Li. Donde nos expone:

-La importancia de la utilización de modelos OLS, que son modelos de mínimos cuadrados ordinarios (OLS por su forma en inglés), estos modelos son los utilizado normalmente como regresiones lineales, polinomiales, etc.

-Utilización del modelo de regresión de Ridg el cual podemos ver que en su estudio es uno de los más preciso para precios bajos y medios.

-Utilización de métodos como XGBoost que son modelos de árboles de decisiones, los cuales en su estudio nos comenta que no dan buenas predicciones.

1.3 Objetivos del trabajo

El objetivo principal del trabajo es realizar un estudio estadístico aplicando técnicas de machine learning y técnicas estadísticas para poder predecir el precio en el mercado eléctrico. Los objetivos específicos que hemos intentado cubrir en este trabajo son los siguientes:

- Recolección y análisis de datos del mercado eléctrico utilizando fuentes confiables y actualizadas.

- Crear y probar diferentes modelos de regresión que permitan estimar el precio en función de variables explicativas como la demanda, la oferta, el clima, etc.

- Desarrollar un sistema de análisis utilizando Python y Rstudio que contraste las hipótesis nula y alternativa sobre si el precio observado se ajusta al precio predicho por el modelo.

Con el fin de cumplir con estos objetivos, se han realizado las siguientes acciones:

- Se ha desarrollar un código en Python que tenga la habilidad de extraer los datos del mercado electico desde diferentes fuentes web y los almacene en un archivo local para su posterior análisis.

- Se aplicarán técnicas de corrección de datos para solucionar problemas de valores nulos y su relevancia para el análisis.

- Se utilizarán las librerías de Python y R para realizar diferentes modelos de regresión lineal y no lineal, con los que por diferentes indicadores podemos evaluar su fiabilidad, algunos de ellos son el coeficiente de determinación, el error cuadrático medio, etc.

2 Marco teórico

2.1 Mercado eléctrico

El mercado energético es uno de los sectores más importantes de una economía, y los precios de la energía son un dato crítico en la toma de decisiones de los consumidores, productores, políticos, etc. La energía eléctrica, en particular, es una fuente esencial para la mayoría de las actividades económicas y sociales. En este sentido, la predicción de precios en el mercado eléctrico se ha convertido en un área de gran interés para los investigadores y los actores del mercado.

La formación de precios en el mercado eléctrico es un proceso complejo que involucra múltiples factores, incluyendo:

-la oferta y la demanda de energía,

-la capacidad de generación,

-la variabilidad climática,

-las políticas gubernamentales,

-las restricciones de infraestructura,

-entre otros.

Podemos definir el mercado eléctrico como el sistema que permite la compraventa de energía eléctrica entre los agentes. Estos agentes pueden ser productores, consumidores, comercializadores o intermediarios. El mercado eléctrico se divide en dos grandes segmentos: el mercado mayorista y el mercado minorista.

Primero definiéremos el mercado mayorista como el mercado que se encarga de determinar el precio y la cantidad de energía que se intercambia entre los productores y los consumidores mayoristas (grandes industrias o distribuidoras). Este mercado se compone de varios submercados: el mercado diario, el mercado intradiario, el mercado de ajuste y el mercado de servicios auxiliares. Cada uno de estos submercados tiene sus propias reglas e intervalos temporales.

El mercado diario es el que fija el precio y la cantidad de energía para cada hora del día siguiente, mediante un proceso de unión con otros mercados europeos. El mercado intradiario permite ajustar las ofertas y demandas de energía a lo largo del día en curso, mediante subastas o negociaciones continuas. El mercado de ajuste gestiona las desviaciones entre la programación y la producción o consumo real de energía, mediante ofertas de balance. El mercado de servicios auxiliares proporciona los recursos necesarios para garantizar la seguridad y la calidad del suministro eléctrico, como la reserva, el control de frecuencia o el control de tensión.

Como segundo mercado definiremos el mercado minorista como el que se encarga de suministrar la energía eléctrica a los consumidores finales (hogares, comercios o pymes). Este mercado está definido por los contratos que establecen los consumidores con las comercializadoras, que pueden ser de dos tipos: el precio voluntario al pequeño consumidor (PVPC) o el precio libre. El PVPC es un precio regulado por el gobierno que refleja el coste de la energía en el mercado mayorista más unos peajes y cargos de la administración. Por otro lado, el precio libre es un precio pactado entre el consumidor y la comercializadora, que puede incluir descuentos, servicios adicionales o tarifas planas.

En este contexto, el uso de técnicas de análisis estadístico y de machine learning se ha convertido en una herramienta importante para la predicción de precios en el mercado eléctrico.

Por otro lado, es importante destacar que el mercado energético es altamente dinámico y volátil por lo que está sujeto a cambios y fluctuaciones constantes. Por esta razón, los modelos de predicción de precios deben ser capaces de adaptarse a los cambios en las condiciones del mercado y proporcionar resultados precisos y confiables en tiempo real.

En este sentido, el uso de técnicas de machine learning se ha vuelto cada vez más popular en el ámbito de la predicción de precios en el mercado energético. Estas técnicas tienen la capacidad de analizar grandes cantidades de datos históricos y de tiempo real, identificar patrones y tendencias en los datos y predecir los precios futuros en el mercado.

Entre las técnicas de machine learning más utilizadas en la predicción de precios en el mercado energético se incluyen el análisis de regresión, los árboles de decisión, las redes neuronales, y el aprendizaje profundo. Cada una de estas técnicas tiene sus propias ventajas y desventajas, y la elección de la técnica más adecuada dependerá de las características específicas del mercado y de los datos disponibles.

En conclusión, el mercado energético y los precios de la energía son temas de gran importancia para la economía global y para la toma de decisiones de los actores del mercado. La predicción de precios en el mercado eléctrico es un área de investigación moderna y en constante evolución, en la que se utilizan técnicas de análisis estadístico y machine learning para proporcionar predicciones precisas y confiables en tiempo real.

En el siguiente apartado, se describirán con más detalle las variables que influyen en la definición de precios en el mercado eléctrico y los modelos de predicción de precios que hemos utilizado.

2.2 Mercado eléctrico español

En España podemos considerar que el mercado español empieza desde 1997 con la liberalización del mercado eléctrico que conlleva a los contratos de compra y venta de electricidad en diferentes plazos como horas, días, semanadas, meses, etc.

Actualmente en el mercado podemos ver dos clases de agentes los agentes regulados por el estado, que son los agentes de operaciones de la red y la distribución de la electricidad que son controlador por la REE. Por otro lado, tenemos los agentes liberalizados como son la generación y la comercialización, algunas de las empresas más conocidas son Iberdrola, Naturgy, etc.

Como hemos expuesto anteriormente el mercado se divide en dos grandes mercados:

- Mercado minorista: el mercado de contratos de los consumidores con las comercializadores.
- Mercado Mayorista: en el caso de España el Mercado ibérico de la Electricidad, formado por España y Portugal. En este mercado podemos encontrar diferentes tipos de contratos.

2.3 Análisis de la demanda y la oferta del mercado

El análisis de la demanda y la oferta del mercado eléctrico lo hemos definido como el estudio de las características y el comportamiento de los agentes que participan en el sistema eléctrico, así como las condiciones técnicas, económicas y regulatorias que influyen en el equilibrio entre la generación y el consumo de electricidad.

2.3.1 Agentes del sistema eléctrico

Como hemos comentado anteriormente, los agentes del mercado eléctrico don todos aquellos que están relacionados con la producción, la transmisión, la distribución o el consumo de la electricidad. Estos agentes dependiendo del sistema en el que interactúen (sistemas monopolísticos, sistemas de libre mercado, etc.) pueden variar, pero de manera general tenemos los siguientes agentes:

- Generadores: Son los encargados de la producción eléctrica, podemos ver que en la mayoría de los países pueden ser desde empresas públicas que operan con centrales eléctricas o empresas privadas, productores independientes. Estos agentes están especializados en procesar recursos externos, las fuentes de energía (como la energía nuclear, la energía eólica, …) y producir la electricidad.
- Operadores de red: Son los encargados de la operación y el mantenimiento de las infraestructuras que permiten la transmisión y distribución eléctrica, es decir son los encargados del transporte eléctrico. En el caso de España es el REE, por lo que podemos decir que mayormente este agente es un ente público.
- Distribuidores: Son los encargados de transmitir la electricidad desde la red hasta los consumidores finales que pueden ser empresas, hogares, industrias, etc.

- Consumidores: Son los agentes finales, también llamados usuarios finales, ellos consumen la electricidad.
- Reguladores: Organismos encargados de la regulación y de los estándares del sistema eléctrico.

2.3.2 Equilibrio de consumo y generación

Entendemos como equilibrio que la cantidad de energía eléctrica generada sea igual a la cantidad de energía eléctrica consumida. Para poder entender mejor el equilibrio que tiene el mercado español en este apartado veremos un análisis de las dos partes.

Generación

La generación de energía en España, podemos ver que desde 1980 ha tenido una tendencia creciente, teniendo una pequeña recesión desde el 2008 hasta el 2020 coincidiendo con los años de crisis económica financiera y hasta después de la pandemia de covid-19.

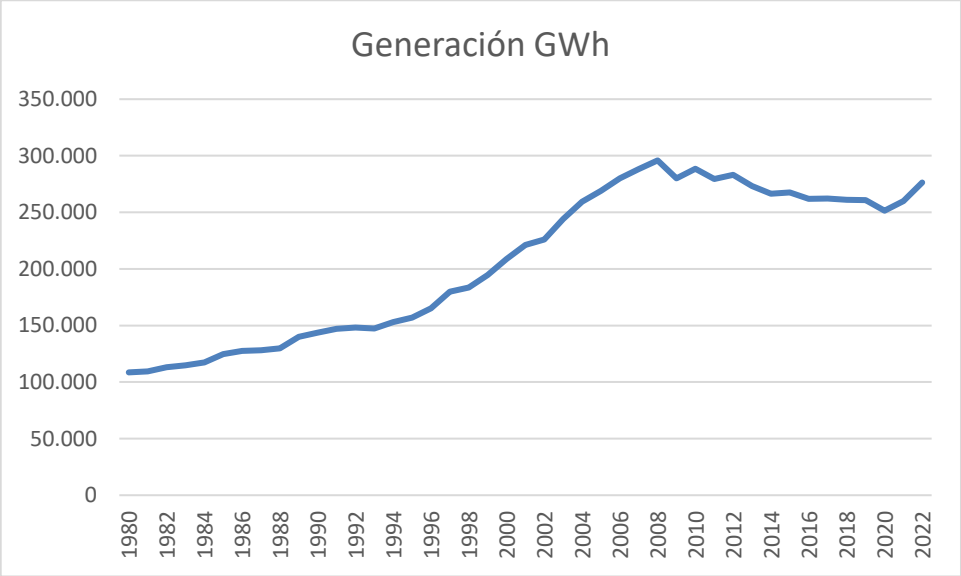


Figura 1. Gráfico de generación de energía eléctrica. (Elaboración propia)

Consumo

Podemos ver una tendencia similar a la del grafico de generación eléctrica, el consumo se dispara desde el 1980 hasta al inicio de la crisis d 2008 donde empieza una recesión.

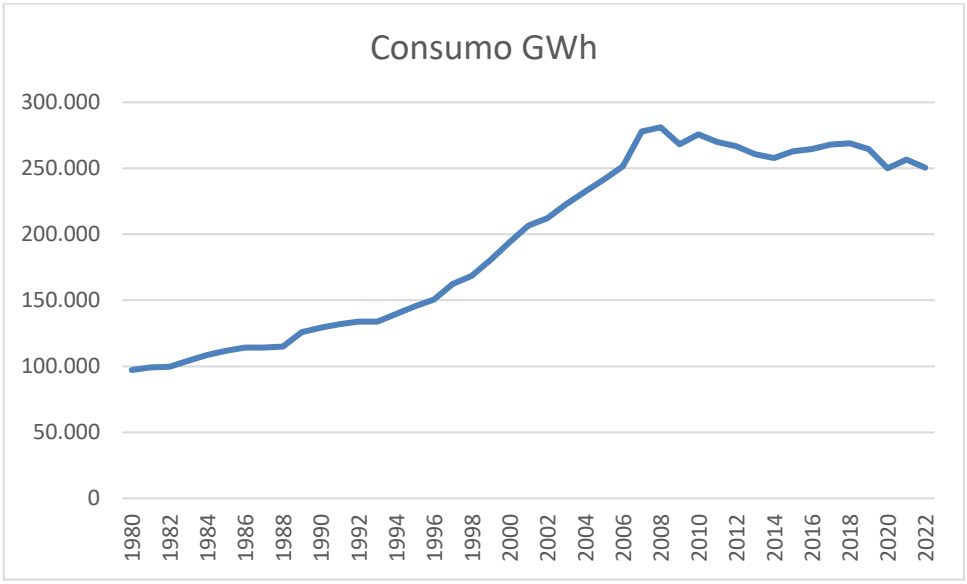


Figura 2. Gráfico de consumo de energía eléctrica. (Elaboración propia)

Equilibrio

De manera general uniendo los dos gráficos podemos observar que el mercado español no ha tenido déficit energético, por lo que ha mantenido un equilibrio energético, en la mayor parte de su historia, solo entre el 2016-2020 vemos un periodo de déficit, por lo que podemos decir que el mercado español mayormente genera más de los que se consume en el mercado nacional

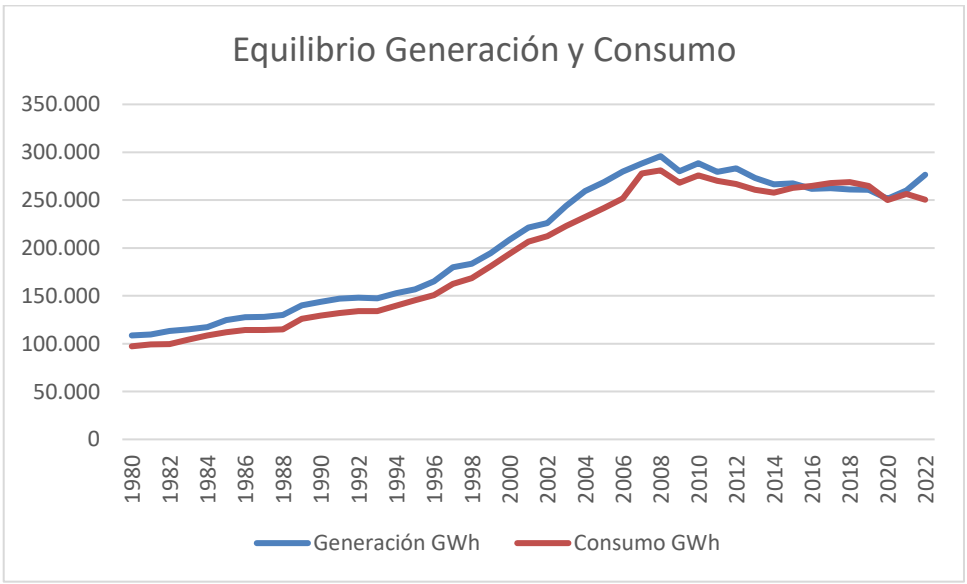


Figura 3. Gráfico de equilibrio entre el consumo y la generación de energía eléctrica. (Elaboración propia)

Nos gustaría destacar que este equilibrio energético es esencial para la estabilidad de la red eléctrica y la sostenibilidad a largo plazo. A medida que avanzamos hacia un futuro más verde y sostenible, será crucial continuar monitoreando y gestionando este equilibrio para garantizar que la generación de energía pueda satisfacer la demanda sin comprometer nuestros objetivos de sostenibilidad.

2.4 Machine Learning y su aplicación en el mercado energético

Podemos ver cada año un aumento de los diferentes métodos y algoritmos de machine learning o aprendizaje automático, podemos definir este concepto como un conjunto de diferentes procedimientos que permiten a los sistemas informáticos aprender sobre procesos simulando el comportamiento humano, estas técnicas se pueden clasificar en grandes términos por el tipo de aprendizaje:

- Aprendizaje supervisado: Se define a los métodos en los que se utilizan datos etiquetados en el cual conocemos los resultados, es decir son conocidos previamente al desarrollo del modelo.
- Aprendizaje no supervisado: Se define a los métodos en los cuales se utilizan datos no etiquetados con lo que no conocemos la estructura de los mismo, más bien estos tipos de modelos son utilizados para buscar datos importantes sin conocer las variables de salidas.
- Aprendizaje reforzado: Se define a los métodos en los cuales conocemos como "Deep Learning" o aprendizaje profundo, este tipo de modelo tiene como objetivo la construcción de modelos con un gran rendimiento teniendo resultados o recompensa en las diferentes interacciones. Estos modelos son utilizados para la predicción de movimientos como en el juego de ajedrez.

Por otro lado, podemos definir diferentes tipos de algoritmos según su construcción y su forma, los cuales se pueden definir en:

- Algoritmos de regresión: Son los algoritmos utilizados para la realización de tareas de regresión como el estudio de relaciones entre variables y la estimación de resultados.
- Algoritmos Bayesianos: Son los algoritmos que aplican el teorema de Bayes con lo que nos permite la clasificación de variables independientes como otros datos del conjunto de datos que nos permita la clasificación de estos.
- Algoritmos de agrupación: Son los algoritmos de aprendizaje no supervisado que organiza y categoriza los datos que no están etiquetados, es decir, se generan grupos y agrupaciones de los datos para representarlos con una variable.
- Algoritmos de árbol de decisión: Son los algoritmos que nos permite representar de manera estructurada las diferentes opciones basadas en criterios preestablecidos, podemos ver la generación de nodos que representan variables y ramas que se generan según el resultado de las pruebas ejecutadas.

Como hemos indicado el machine learning es una disciplina que tiene como objetivo desarrollar algoritmos capaces de aprender a partir de los datos y de mejorar su rendimiento con la experiencia. En el ámbito del mercado energético, el machine learning se ha convertido en una herramienta fundamental para la predicción de precios y la gestión de la demanda y la oferta de energía.

Una de las aplicaciones más importantes del machine learning en el mercado energético en especial en el mercado eléctrico es la predicción de precios de la energía eléctrica. Para ello, se utilizan técnicas de análisis estadístico y de machine learning para analizar grandes cantidades de datos histórico, identificar patrones y tendencias en los datos y predecir los precios futuros en el mercado.

3 Estado del Arte

3.1 Desarrollo/Metodología

En este trabajo hemos intentado plasma todos los pasos seguidos para el análisis de los datos, desde la extracción, hasta los resultados finales.

Las etapas que hemos seguido en toda la estructura para genera una base de datos con la que trabajar es:

- Extracción de datos: extraer los datos de fuentes fiables.
- Procesamiento de datos: Proceso de recolección de los datos extraídos y almacenamiento local.
- Tratamiento de datos: Tratamiento de datos atípicos, datos nulos, etc. Y eliminación de datos no necesarios.
- Unificación de datos: Unificar los diferentes datos obtenidos en una sola base de datos.
- Limpieza de datos: Eliminación de datos duplicados y datos innecesarios de la unificación.

Figura 4. Esquema de meteorología (Elaboración propia)

Siguiendo estos pasos, que saldrán mostrados en los diferentes códigos proporcionados en los anexos, hemos de conseguir tener una base de datos solida con la que poder realizar los diferentes modelos.

Para la realización de los diferentes modelos hemo utilizado una metodología similar en cada uno de ellos, con la finalidad de poder obtener algunos indicadores para poder comprobarlos.

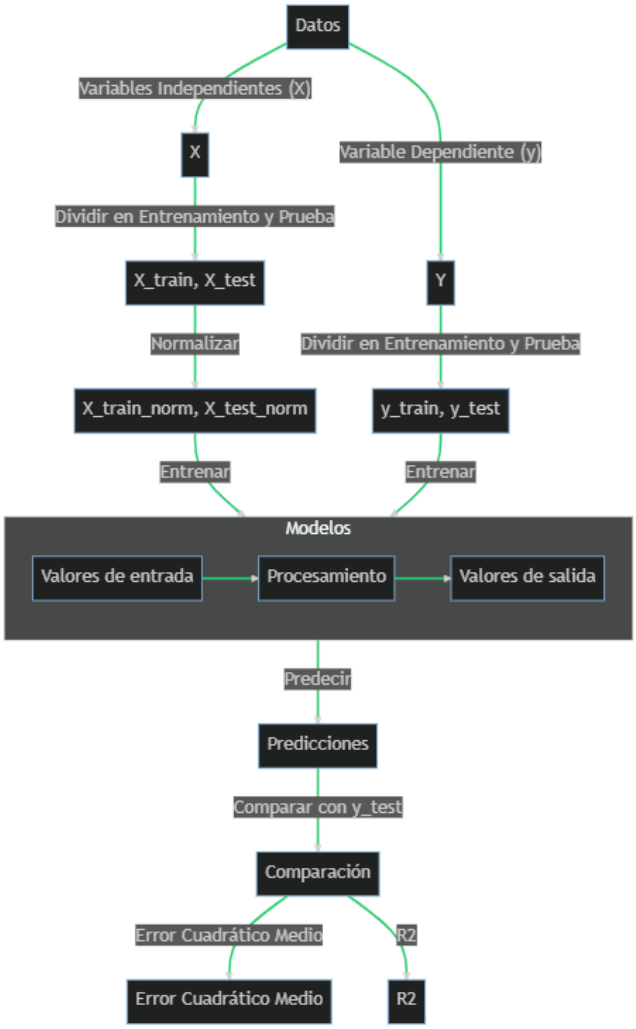


Figura 5. Esquema de pruebas y análisis de modelos (Elaboración propia)

Como idea general, hemos intentado reflejar el esquema de pasos que hemos realizado para la programación y la ejecución de los modelos. Primero podemos ver que hemos realizado una división de las variables para tener el conjunto de variables independientes y el vector de variable dependiente, en este caso el Precio, con esto utilizando diferentes herramientas de Python hemos dividido los datos en dos grupos, el primero que es el 80% de los datos son los datos de entrenamiento, los datos que utilizara el modelo para estimar los componentes necesario, y el 20% de los datos que son los datos de prueba que utilizaremos para poder saber la precisión del modelo.

Después de dividir los datos alimentamos el modelo con los datos de entrenamiento, y le indiquemos que queremos la predicción de datos según los datos de prueba. Con esto

podemos llegar a tener ya la predicción del modelo. Con los datos podemos buscar dos grandes indicadores que son el error cuadrático medio y la bondad de ajuste (R^2).

En el caso de la red neuronal podemos ver un funcionamiento similar que los demás modelos con la importancia de la capa interior o oculta de la red neuronal que cambia al ser un sistema de nodos y procesamiento de datos en el interior de ellos.

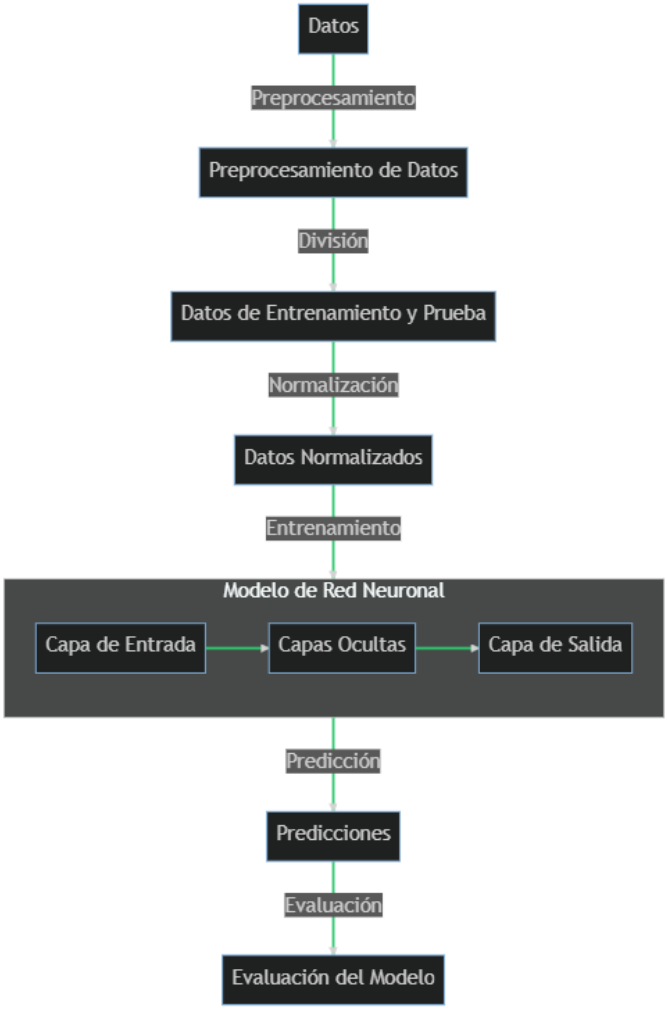


Figura 6. Esquema de pruebas y análisis de Red Neuronal (Elaboración propia)

3.2 Explicación de Python, R y Rstudio como herramienta de Trabajo

Para poder realizar el desarrollo del trabajo hemos escogió el lenguaje de programación Python, que podemos definirlo como un lenguaje de programación de alto nivel de tipo interpretado, actualmente este lenguaje de programación es utilizado para diferentes usos que abarcan desarrollo de aplicaciones web y de escritorio (desarrollo de software), control de microprocesadores, scripting, … y el que nos interesa en este trabajo el análisis de datos y machine learning. Python gracias a ser un lenguaje de alto nivel nos proporciona un desarrollo rápido y fácil de actualizar y modificar, por otro lado su comunidad y las instituciones han creado un gran abanico de librerías que nos ayuda y facilita el desarrollo.

Por otro lado para poder utilizar el lenguaje hemos decido utilizar el entorno de desarrollo (IDE) que nos proporciona Rstudio que podemos definirlo como un programa o herramienta gratuita que se ha desarrollado para realizar los desarrollos de manera más rápida y cómoda por medio de sus componentes como la consola, el editor de sintaxis, ejecución y depuración de código, … lo que nos realiza un sistema de gestión del espació de trabajo.

Este está especializado para la programación en el lenguaje R pero gracias la librería "reticulate" del lenguaje R nos permite ejecutar código escrito en Python en la consola de R.

Como hemos comentado Python dispone de un gran número de librerías y para el desarrollo de este trabajo utilizaremos las siguientes:

-Numpy, especializada en el cálculo y el análisis numérico.

-Pandas, que nos proporciona nuevas estructuras de datos, funciones de lectura y escritura de datos desde diferentes sistemas (base de datos SQL, csv, Excel, …) y se especializa en el desarrollo de herramientas de data science y machine learning.

-Matplotlib, que nos proporciona diferentes tipos de funciones para la generación de gráficos en dos dimensiones.

-Seaborn, que nos proporciona diferentes herramientas para la creación de gráficos estadísticos de manera más simple e inédita de matplotlib.

-Scipy, que nos proporciona herramientas y algoritmos para realizar métodos de optimización, algebra lineal, integración, procesamiento de datos, ….

-Sklearn, que nos proporciona diferentes algoritmos de machine learning de regresión, clasificación, clustering y reducción.

-Statsmodels, que nos proporciona diferentes herramientas de análisis y modelado estadístico (regresión lineal, modelos generalizados, …).

Tambien tenemos que comentar que en el desarrollo, se ha empleado el lenguaje de programación R, una herramienta para el análisis estadístico y la visualización de datos. R nos ha ofrecido una amplia gama de paquetes que permiten mejorar y personalizar los modelos estadísticos, así como crear gráficos detallados y atractivos.

3.3 Fuentes de datos y preprocesamiento

Para aplicar técnicas de machine learning a la predicción de precios en el mercado energético eléctrico, es necesario contar con una gran cantidad de datos históricos sobre los precios y el comportamiento del mercado. Estos datos pueden ser obtenidos de diversas fuentes, como los operadores del mercado energético, los proveedores de

energía, las empresas de análisis financiero y las agencias gubernamentales encargadas de regular el sector energético.

Una vez obtenidos los datos, es necesario realizar un proceso de preprocesamiento para asegurar su calidad y uniformidad. Este proceso incluye la limpieza y eliminación de datos erróneos o incompletos, la identificación y corrección de valores atípicos y la normalización de los datos para asegurar que sean comparables a lo largo del tiempo. También se pueden utilizar técnicas de interpolación o extrapolación para llenar los vacíos en los datos faltantes.

En este apartado explicaremos los datos y sus fuentes para poder hacer un análisis del mercado eléctrico y poder concluir un modelo donde poder explicar y predecir datos futuros.

Los datos que utilizaremos provienen de tres fuentes principales: OMIE, AEMET y REE.

- OMIE es el operador del mercado ibérico de electricidad, que publica los precios horarios del mercado diario e intradiario, así como la demanda y la oferta de energía.

- AEMET es la agencia estatal de meteorología, que proporciona información sobre las variables climáticas que pueden influir en el consumo y la generación de energía, como la temperatura, la humedad, la velocidad del viento y la radiación solar.

- REE es el operador del sistema eléctrico español, que gestiona el transporte y la distribución de la electricidad, así como el balance entre la oferta y la demanda. REE también ofrece datos sobre la producción de energía por tipo de tecnología, como hidráulica, eólica, solar o térmica.

3.4 Selección de variables y análisis exploratorio de datos

En este apartado queremos exponer que se han seleccionado las siguientes variables para realizar el análisis exploratorio de datos:

- Año, mes y día: son variables cuantitativas discretas que indican la fecha de cada observación. Estas variables son relevantes porque nos permiten estudiar la estacionalidad y la tendencia del precio de la electricidad. Además, son variables disponibles en el portal de datos abiertos del operador del mercado eléctrico (OMIE).

- tmed, tmin y tmax: son variables cuantitativas continuas que miden la temperatura media, mínima y máxima en grados Celsius. Estas variables son relevantes porque afectan a la demanda y a la generación de energía, especialmente la solar y la eólica. Además, son variables disponibles en

el portal de datos abiertos de la Agencia Estatal de Meteorología (AEMET).

- prec: es una variable cuantitativa continua que mide la precipitación en milímetros. Esta variable es relevante porque influye en la generación hidroeléctrica, que es una de las fuentes más baratas y flexibles. También es una variable disponible en el portal de datos abiertos de la AEMET.

- Demanda: es una variable cuantitativa continua que mide la demanda eléctrica en megavatios hora (GWh). Esta variable es relevante porque determina el equilibrio entre la oferta y la demanda en el mercado eléctrico, y por tanto el precio. Es una variable disponible en el portal de datos abiertos de Red Eléctrica de España (REE).

- Diferentes tipos de generación de energía: son variables cuantitativas continuas que miden la generación eléctrica en GWh de cada fuente de energía (eólica, solar, hidráulica, etc.). Estas variables son relevantes porque cada fuente tiene un coste y una capacidad diferentes de adaptarse a las variaciones de la demanda. Son variables disponibles en el portal de datos abiertos de REE.

- Generación total: es una variable cuantitativa continua que mide la generación eléctrica total en GWh. Esta variable es relevante porque debe ser igual a la demanda más las pérdidas en el transporte y distribución. Es una variable disponible en el portal de datos abiertos de REE.

- Precio: es una variable cuantitativa continua que mide el precio medio diario de la electricidad en euros. Esta variable es la dependiente o de estudio, ya que queremos analizar cómo influyen las demás variables en su comportamiento. Es una variable disponible en el portal de datos abiertos de OMIE.

La variable dependiente o de estudio en este trabajo es el precio, ya que queremos analizar cómo influyen las demás variables en su comportamiento.

3.5 Tratamiento de datos

En este apartado describiremos los procesos de tratamiento de datos que se ha realizado para obtener una base de datos limpia y adecuada para el análisis.

El primer tratamiento que podemos decir es la descarga de los datos desde la web hasta diferentes archivos locales, estos datos han sido descargados de manera manual o utilizando diferentes herramientas de conexión a servicios de internet, como APIs, para la descarga directa de los datos. En este trabajo la mayoría de los datos han tenido que

ser descargados de manera manual en el caso de los datos provenientes de OMIE y de REE, en el caso de los datos obtenidos de AEMET se han descargado utilizando una conexión a su API, la cual nos ha permitido la descarga de los datos de manera automática en formato json, un ejemplo de la conexión realizada es:

```
#DATOS DE BARCELONA
url = "https://opendata.aemet.es/opendata/sh/8e275d34"

response = requests.request("GET", url, headers=headers,
params=querystring)

datosJsonBarcelona = response.json()
```

Gracias a este código desarrollado en Python hemos podido obtenido los datos meteorológicos de la región de Barcelona, para tener una gran unión de los datos hemo elegido coger diferentes puntos específicos de España como Barcelona, Madrid, a Coruña y Navarra.

Más tarde todo los datos, precios, meteorológicos, estructurales y demanda han sido tratados para separar la fecha en tres campos diferentes Año, Mes y Día los cuales serán los campos que utilizaremos como claves primarias a la hora de realizar una unificación de los datos en una misma tabla.

En el caso de datos de precio y de meteorología utilizados estaban en divididos en datos horarios para todos los días del año, para poder tener un análisis más firme y consistente se ha decidido realizar la media de los datos ordenados por fecha, con esto tenemos los valores medios diarios de estos datos.

```
#agrupar por fecha y calcular la media de los datos
df = df.groupby('fecha').mean()
```

Con este código en Python, se ha realizado la agrupación de datos por el campo fecha y se han asignado como valor la media de los datos.

3.6 Modelos de Machine Learning utilizados

Como hemos explicado, existen diferentes tipos de modelos y algoritmos de machine learning, para nuestro estudio sobre predicción de precios de la electricidad utilizaremos diferentes algoritmos de los tipos de regresión y bayesianos que nos permiten realizar estimaciones y análisis de las variables que tenemos. Los algoritmos que utilizaremos son:

-Algoritmos de regresión

- Regresión Lineal

Como idea general podemos definir la regresión lineal como un modelo estadístico que se utiliza para crear modelos de relación entre una variable dependiente y una o más

variables independientes. En la regresión lineal, se ajusta una línea recta a los datos que se ajuste mejor a los patrones lineales en los datos.

En la regresión lineal, la relación entre una variable dependiente y las variables independientes x realiza o se crea un modelo utilizado una ecuación lineal. La forma matricial, la ecuación de regresión lineal se puede expresar de la siguiente manera:

$$Y = X\beta + \epsilon$$

Donde:

- Y es un vector de la variable dependiente del modelo.
- X es la matriz de las variables independientes. Cada columna de X representa una variable independiente, y cada fila representa una observación.
- β es el vector de los coeficientes de regresión. Cada elemento de β corresponde a la pendiente de la línea de regresión para la variable independiente correspondiente.
- ϵ es el vector de los errores de regresión. Cada elemento de ϵ representa la diferencia entre el valor observado de la variable dependiente y el valor predicho por la línea de regresión.

La solución de mínimos cuadrados para los coeficientes de regresión β se puede obtener resolviendo la siguiente ecuación:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

- Regresión Polinomial

Podemos definir la regresión polinomial como un modelo de regresión que se utiliza para modelar relaciones no lineales entre una variable dependiente y una o más variables independientes. En la regresión polinomial, se ajusta una curva polinómica a los datos en lugar de una línea recta.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \epsilon$$

Donde:

- y es la variable dependiente.
- x es la variable independiente.
- β0,β1,…,βn son los coeficientes de regresión.
- ϵ es el error de regresión.

- Regresión de Ridge

Podemos definir la regresión de ridge como un modelo regularizado que agrega un término de penalización a la función de pérdida del modelo para evitar el sobreajuste.

Esto permite controlar la complejidad del modelo y mejorar su capacidad de generalización.

Este método es útil cuando se tienen muchas variables explicativas que presentan una alta correlación entre ellas, lo que puede provocar problemas de multicolinealidad.

En el contexto del mercado eléctrico, es frecuente que las variables que afectan a los precios tengan una fuerte correlación, por lo que la regresión de Ridge es una buena alternativa para abordar esta situación. Como ejemplo en nuestro análisis, si queremos predecir el precio de la electricidad en función de la demanda, la temperatura, el viento y el tipo de combustible, podemos usar la regresión de Ridge para evitar que las variables que están relacionadas entre sí (como la demanda y la temperatura) influyan demasiado en el modelo y lo hagan inestable.

La ecuación para la regresión de Ridge es:

$$\min_{\beta}\{|y - X\beta|^2 + \lambda|\beta|^2\}$$

Donde:

- y es un vector de la variable dependiente.
- X es la matriz de las variables independientes.
- β es el vector de los coeficientes de regresión.
- λ es el parámetro de regularización que controla la cantidad de contracción de los coeficientes hacia cero. Un valor mayor de λ resultará en una mayor contracción, lo que puede ayudar a prevenir el sobreajuste.

La solución para los coeficientes de regresión en la regresión de Ridge es:

$$\beta = (X^{TX} + \lambda I)^{-1X^{Ty}}$$

Donde I es la matriz de identidad del mismo tamaño que X^TX.

-Algoritmos Bayesianos

- Regresión Bayesiana

Podemos definir la regresión bayesiana como el modelo que utiliza técnicas bayesianas para encontrar la distribución de probabilidad de los parámetros del modelo, en lugar de estimar valores puntuales como lo hacen otros modelos. Según los expertos la regresión bayesiana es útil para la predicción de precios en el mercado, ya que permite incorporar información previa sobre el comportamiento del mercado y obtener predicciones más precisas.

Esta regresión utiliza el teorema de Bayes para encontrar la distribución de probabilidad posterior de los parámetros del modelo. La regresión bayesiana proporciona una distribución de probabilidad completa para los parámetros. La ecuación para la regresión bayesiana es similar a la de la regresión lineal, pero en lugar de encontrar

valores específicos para los coeficientes, busca la distribución posterior de los coeficientes dados los datos:

$$p(\boldsymbol{\beta}|\boldsymbol{X},\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\beta})p(\boldsymbol{\beta})$$

Donde:

- p(β|X,y) es la distribución posterior de los coeficientes β dados los datos X y y.
- p(y|X,β) es la verosimilitud de los datos y dados los coeficientes β y los datos X.
- p(β) es la distribución a priori de los coeficientes β.

- Proceso Gaussiano

Podemos definir un proceso Gaussiano como un modelo bayesiano que nos permite modelar la distribución de probabilidad conjunta de la variable. Estos procesos son utilizados para la predicción de precios en el mercado eléctrico, ya que permiten modelar relaciones no lineales entre las variables y obtener predicciones precisas con incertidumbre.

De manera general definimos un proceso a través de su función de media)m(x) y su función de covarianza k(x,x′). En su forma más simple, un proceso Gaussiano se puede escribir como:

$$f(x) \sim \mathrm{GP}(m(x), k(x,x'))$$

Donde:

- f(x) es la función aleatoria que sigue un Proceso Gaussiano.
- GP denota el Proceso Gaussiano.
- m(x) es la función de media, que es la expectativa de f(x) en cada punto x.
- k(x,x′) es la función de covarianza, que mide cómo las salidas f(x) y f(x′) varían conjuntamente.

-Red neuronal Secuencial

En este trabajo para poder tener una mejor percepción de las nuevas técnicas estadísticas hemos desarrollado una red neuronal secuencial simple, esta red la podemos definir como un modelo de aprendizaje automática que aprende o utiliza datos de manera secuencial.

En palabras más técnicas, podemos decir que una red neuronal secuencial es un tipo de red neuronal artificial donde las conexiones entre los nodos forman una secuencia dirigida a lo largo de una dimensión temporal, en otras palabras, es que la información se mueve en una sola dirección a través de la red, desde la entrada hasta la salida, sin bucles. Esta red como hemos comentado está compuesta de diferentes capas de nodos, cada capa toma

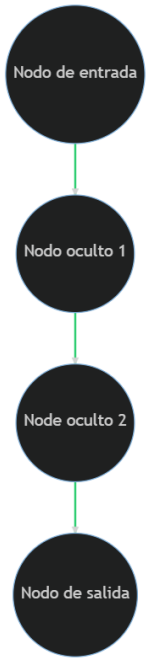


Figura 7. Esquema Red Neuronal (Elaboración propia)

la salida de la capa anterior realiza un procesamiento y envía el dato a la siguiente capa, este proceso finaliza hasta la capa de salida.

Un esquema rápido sería el siguiente, donde podemos ver u n nodo de entrada que recibe los datos y los enviar a los nodos ocultos que realizan un procesamiento y esto hasta llegar al nodo salida.

4 Resultados y discusiones

En este apartado haremos un pequeño resumen de los resultados obtenidos desde los diferentes modelos obtenidos.

Primero, nos centraremos en analizar diversos aspectos del mercado eléctrico durante el año 2019. Como con los precios de la electricidad, podemos ver que la tendencia de los precios diarios de la electricidad durante el año 2019 es una disminución general. Comenzamos el año con precios superiores a 60€/MWh y, a medida que avanzaba el año, estos precios experimentaron una caída, terminando el año con un precio inferior a 40€/MWh. Esta tendencia a la baja en los precios eléctricos es un aspecto significativo para considerar al evaluar el comportamiento del mercado eléctrico en 2019.

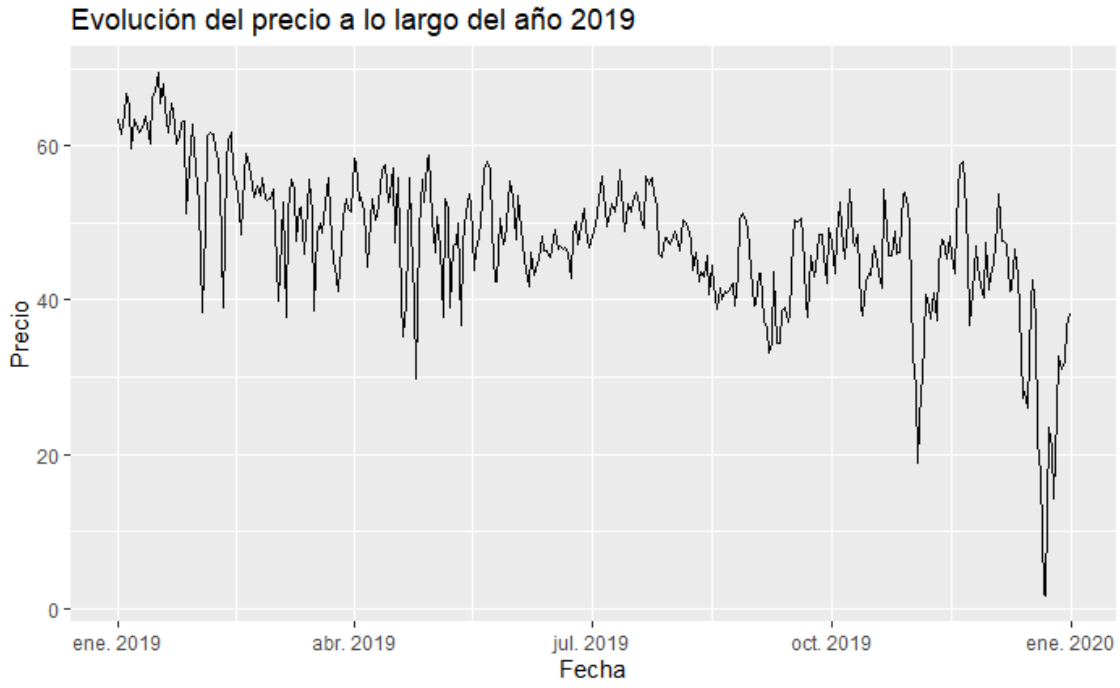


Figura 8. Evolución del precio eléctrico en 2019 (Elaboración propia)

En el caso de la estructura de la generación eléctrica, es notable cómo varían las diferentes fuentes de energía. Las más importantes son la energía nuclear, los ciclos combinados y la energía eólica. Es sorprendente la gran fluctuación de la energía eólica, que parece ser más prominente en ciertos días, mientras que la energía nuclear muestra una notable estabilidad a lo largo del año, manteniéndose dentro de un intervalo de

generación constante. Esto demuestra la diversidad y la complementariedad de las fuentes de energía en la generación eléctrica.

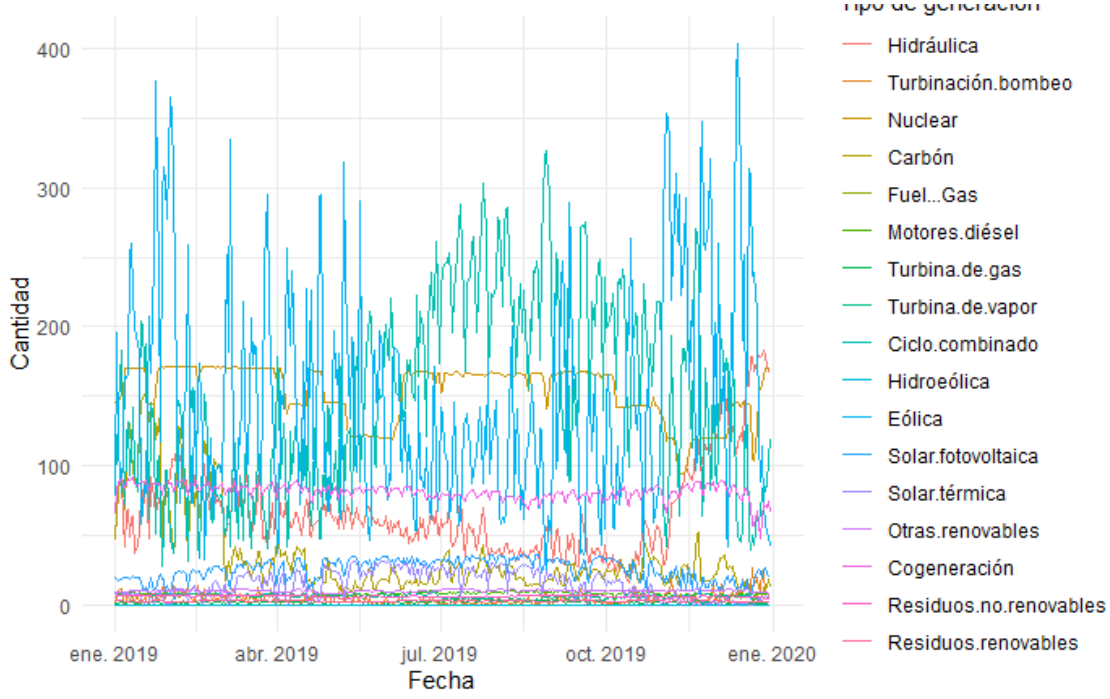


Figura 9. Evolución de la cantidad eléctrica generada por recursos en 2019 (Elaboración propia)

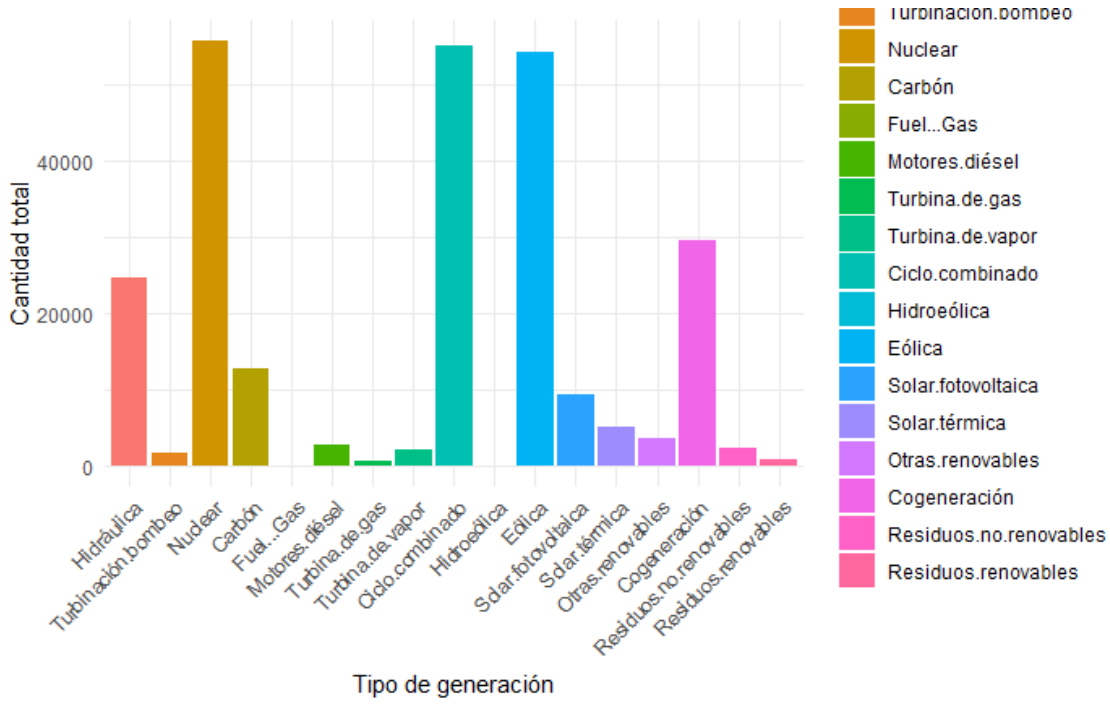


Figura 10. Suma de la cantidad eléctrica generada por recursos en 2019 (Elaboración propia)

A pesar de los avances en la generación de energía renovable, todavía podemos observar un uso significativo de fuentes de energía no renovables. La energía nuclear y

los ciclos combinados, que son fuentes de energía no renovables, siguen siendo predominantes en comparación con la energía eólica, que es una fuente de energía renovable. Esto subraya la necesidad de continuar trabajando hacia una mayor adopción y utilización de fuentes de energía renovables para lograr una matriz energética más sostenible y respetuosa con el medio ambiente.

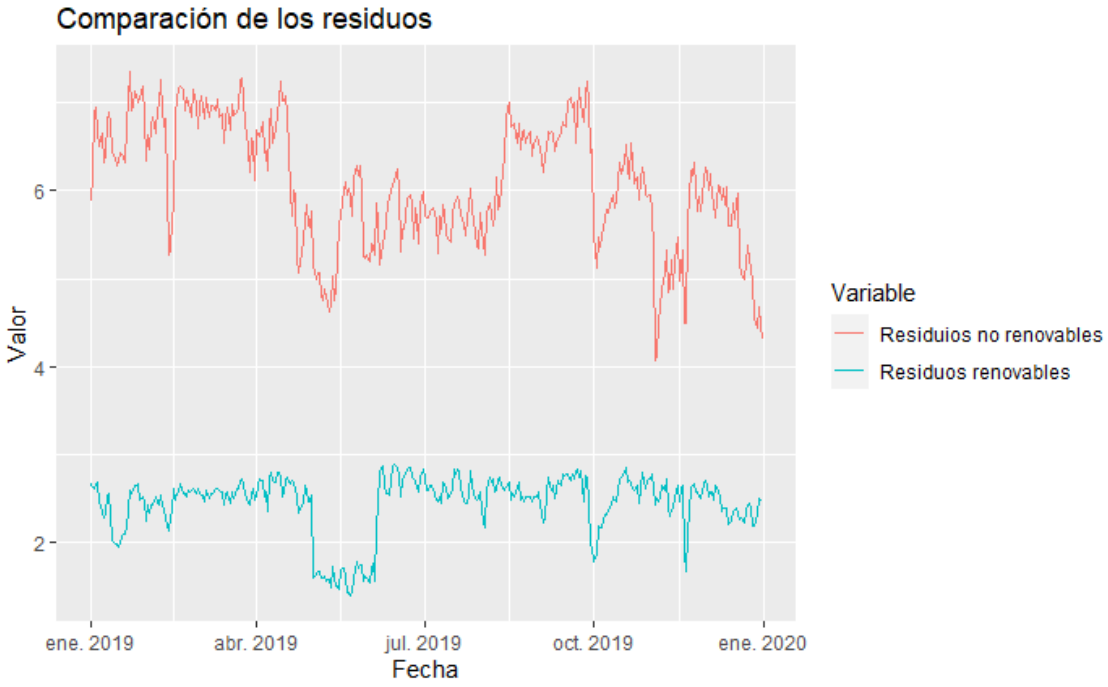


Figura 11. Gráfica comparación de residuos renovables y no renovables en 2019 (Elaboración propia)

Como hemos visto anteriormente el año 2019, se observa que la demanda y la generación de energía muestran tendencias similares, fluctuando de manera conjunta. Esto sugiere una estrecha relación entre ambas, lo que es esperable ya que la generación de energía se ajusta para satisfacer la demanda. Estas tendencias paralelas indican un equilibrio entre la oferta y la demanda de energía durante ese año, lo cual es un aspecto crucial para el funcionamiento eficiente del mercado eléctrico.

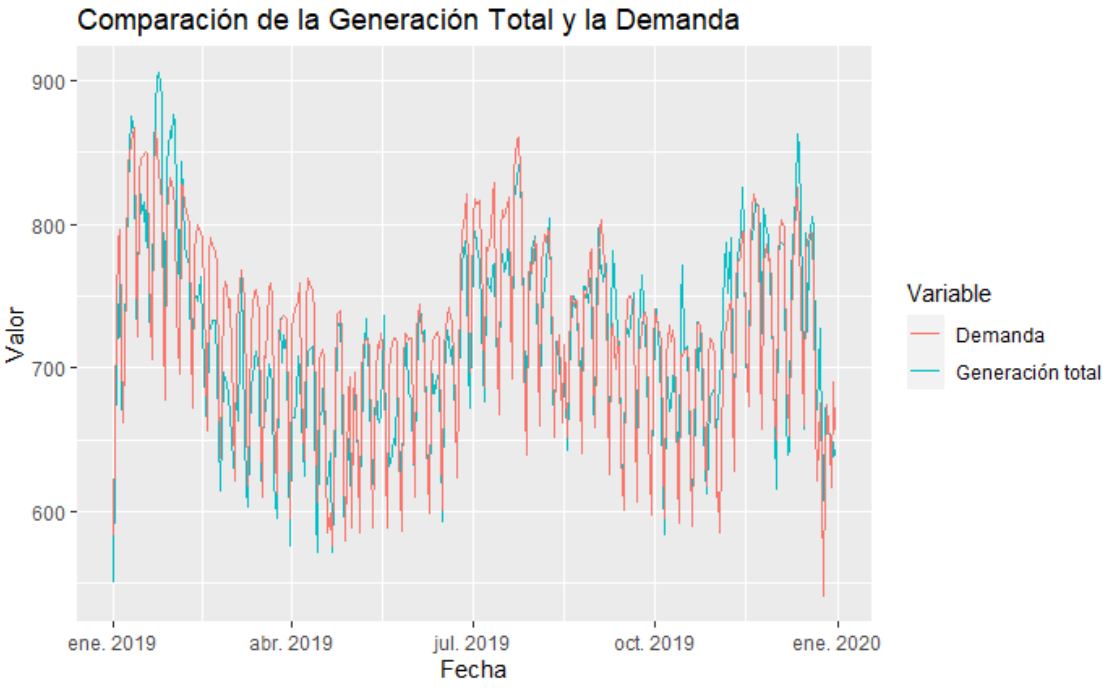


Figura 12. Gráfica de comparación de generación y demanda en 2019 (Elaboración propia)

Ahora que tenemos los conocimientos y el contexto general del año 20019, empecemos a revisar los modelos y sus predicciones.

Modelo de Regresión Lineal

Empezando con el modelo de regresión lineal, que ha sido desarrollado con el algoritmo de LinearRegression de la biblioteca scikit-learn para entrenar el modelo. Se separaron los datos en conjuntos de entrenamiento y prueba, y luego se ejecutó el entrenamiento del modelo.

Una vez entrenado el modelo, se realizaron predicciones utilizando los datos de prueba. Estas predicciones se compararon con los valores reales y se creó un dataframe para mostrar la comparación entre ambos. Un ejemplo de los datos predichos en comparación con los reales es:

```
LinearRegression()
          Real  predicción
0    47.430417   45.413879
1    50.403750   48.706848
2    55.286667   61.972473
3    51.112500   41.667786
4    48.639167   51.956848
..         ...         ...
68   47.886667   47.734192
69   43.298333   46.671692
70   51.290000   51.601379
71   46.567917   50.890442
72   29.685833   42.620911
```

```
Ecuación de la recta en forma matricial: y=  [-3.41062012e-01
2.07088079e+08  9.01785404e+00  3.36635359e-01
  7.13867152e+00  7.48930331e-02 -3.82704712e+00 -3.49538336e+00
  1.19435655e-02  4.44700542e+10  4.44700542e+10  4.44700542e+10
  4.44700542e+10  4.44715402e+10  4.44700542e+10  4.44700542e+10
  4.44700542e+10  4.44700542e+10  4.44700542e+10  4.44700542e+10
  4.44700542e+10  4.44700542e+10  4.44700542e+10  4.44700542e+10
  4.44700542e+10  4.44700542e+10 -4.44700542e+10] x + -
418110832287.86346
```

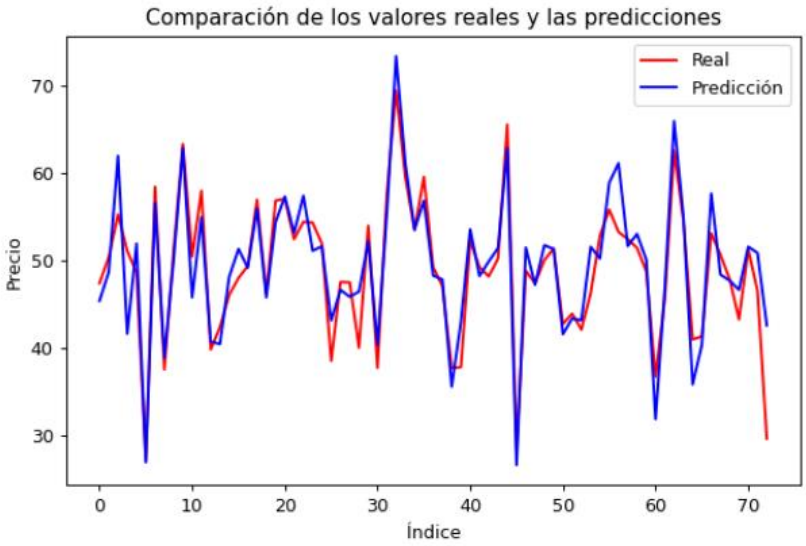


Figura 13. Gráfica de comparación de valores Reales y Predicciones del modelo de regresión lineal(Elaboración propia)

Además de la comparación visual, también se calcularon métricas para evaluar el desempeño del modelo. Se calculó el error cuadrático medio (mean_squared_error) y el coeficiente de determinación (R2) utilizando las funciones de la biblioteca scikit-learn, con datos:

```
Error cuadrático medio: 10.990799703756176
Coeficiente de determinación R²: 0.8275342522220949
```

Con todos los datos estudiados y lo sindicadores mostrados podemos decir que es este modelo explica de manera correcta los precios de la prueba que hemos realizado, por lo que podríamos decir que es un buen modelo para estimar precios.

Modelo de Regresión Polinomial

En modelo de regresión polinomial, sigue el mismo procedimiento que en el modelo lineal.

En primer lugar como en el caso del lineal , los datos se dividieron en conjuntos de entrenamiento y prueba utilizando la función train_test_split de la biblioteca scikit-learn. A continuación, se aplicó la transformación polinomial a las características de entrenamiento utilizando la clase PolynomialFeatures, lo que permitió buscar relaciones no lineales entre las variables.

El modelo de regresión lineal se entrenó utilizando las características transformadas y los valores objetivo de entrenamiento. Al tener lo resultados realizaron predicciones en

el conjunto de prueba se obtuvieron los valores predichos. Se compararon los valores reales y las predicciones en un dataframe para evaluar el desempeño del modelo. Un ejemplo de la comparación es:

LinearRegression()

	Real	predicción
0	47.430417	19.494815
1	50.403750	40.040921
2	55.286667	72.063824
3	51.112500	71.074858
4	48.639167	47.258414
..	...	...
68	47.886667	60.508590
69	43.298333	70.379488
70	51.290000	68.125634
71	46.567917	44.855378
72	29.685833	118.097974

Ecuación de la recta en forma matricial: y= [-4.75212196e-02 -1.39453977e-02 -2.80315543e-02 1.01353246e-02
2.50802879e-02 6.04027467e-04 -1.82006786e-03 3.65335134e-03
4.31723706e-03 -2.57522206e-03 -6.89831677e-04 2.02338872e-03
2.21927746e-03 -4.91842380e-03 -4.70875334e-03 -8.46729288e-04
1.09412615e-03 2.49263598e-03 -4.31916632e-03 -1.67076037e-03
2.43235074e-03 3.02428174e-03 2.48111554e-03 5.51633462e-03
-8.64757765e-04 1.16950476e-03 -3.59634763e-04 -2.76020435e-04
-1.06374036e-01 -2.58986699e-02 3.25715667e+00 2.29392527e-01
1.86645759e+00 -1.99755794e-02 -1.76719732e+00 -1.34349364e+00
-6.08281282e-02 -1.99636018e-01 -9.20998401e-02 -1.86818635e-01
-3.59853979e-01 -6.17999669e-02 -3.90938116e-02 -4.90149805e-02
-1.49629900e-01 1.32051004e-01 -1.93095667e-01 4.51713604e-02
5.75081000e-01 -6.94296510e-01 9.11013141e-02 8.00488637e-01
3.78830264e-01 -6.29496861e-02 1.06234100e-01 4.39026317e-05
7.19475531e-01 3.69824833e-02 -6.46175342e-01 9.54574372e-03
2.95998948e-01 3.17196463e-01 9.47623369e-04 -6.49660781e-02
-5.72583334e-02 -5.72914209e-02 -6.24944200e-02 -5.96199461e-01
-1.60223798e-01 -1.19843476e-01 -3.48516466e-02 -5.48198489e-02
1.51502616e-01 -5.54886520e-02 -3.88054852e-02 -7.26567320e-02
-1.10749347e-01 -7.29167442e-02 -5.94060664e-02 -8.48317643e-02
5.92779932e-02 4.05258727e-01 -3.87654630e+00 1.23853090e+01
6.35771667e-01 1.75733794e+01 7.85274955e+00 1.73030338e+00
6.00872273e+00 2.24317651e+00 5.46295449e+00 1.08652187e+01
-2.13247760e-03 2.55133730e+00 -1.18212011e-01 6.04901085e+00
-4.17728778e+00 2.14141859e-01 -1.56834675e+00 -1.81575985e+01
2.10992755e+01 -1.26375054e-01 -2.47158488e+01 -7.02455271e+00
-1.57568815e+00 -2.98371144e+00 -1.36942610e-01 7.07155600e+00
-4.57345948e-02 -2.77962974e+00 -3.15853259e+00 6.72207206e-02
1.70502110e-01 6.41803611e-02 1.15082658e-01 2.95644553e-01
-6.79712368e-03 -4.11286089e-01 4.46579873e-01 2.14509833e-01
-1.65021252e-01 8.73536044e-01 -8.74574347e-02 -5.38878407e-01
5.70044806e-01 -3.67480307e-01 -1.09222117e+00 -3.39004808e-03
-1.65575404e-01 -6.29371202e-02 -1.87916686e+00 -1.79743971e+01
-2.63857422e+00 1.30889065e+00 -9.12690662e-01 6.61562464e-01
4.07779724e+00 -4.61644138e+00 2.39807784e+00 -6.24960118e-03
1.67305403e+01 -7.01939967e+00 1.56054905e+00 -5.31858577e+00
-1.23002455e+01 -3.34601657e+00 8.85300242e+00 -6.15017230e-01
-1.96007323e+00 -8.07532055e+00 1.07368173e+01 3.43646027e+00
5.19709886e+00 5.57087572e-02 8.77976321e+00 9.09716540e+00
-1.52395461e-02 -1.38894388e+00 -1.41901127e+00 -1.35625639e+00
-1.32264373e+00 -7.36928241e-04 -1.41175163e+00 -2.46837450e+00
-4.56850829e-01 -1.35096784e+00 2.35560228e+01 -1.35267373e+00
-1.51702558e+00 -1.31140811e+00 -1.27613996e+00 -1.51583249e+00
-1.23807519e+00 -2.80484597e+00 1.36435843e+00 3.48624833e+00
-1.02851473e-01 4.69287423e-01 1.81638687e+00 -3.86667915e-01
4.30528647e+00 8.75818981e-01 -4.91295979e-03 -9.17734283e+00
2.91941589e+00 4.73490823e-01 4.71980673e+00 -2.27301877e+01
3.74347668e+00 -1.42771536e+00 1.83820368e+00 5.23320607e-01
6.93100512e+00 -2.86850208e+00 3.78153265e+00 -4.66908151e+00

```
 4.59499612e-02  4.89092476e-01 -1.48965350e-01 -1.54730373e+00
 2.65436141e+00 -9.17770829e-01 -7.58936331e-03 -5.11237513e+00
 3.98961255e+00 -4.95085138e-01  2.93362579e+00 -1.19521890e-01
 1.92437578e+00 -5.04453839e+00  1.16894858e+00  4.25790507e+00
 3.90755908e+00 -6.25082689e+00 -4.11019581e+00 -2.89975482e+00
-1.22488064e-03 -7.30195740e-02 -6.10531328e-02 -8.65703251e-02
-8.03707236e-02 -1.92416919e-01  9.31780711e-02 -2.79411076e-02
 1.40897940e-02 -7.64594391e-02  1.14512184e+00 -7.41467285e-02
-1.84166325e-02 -1.49792238e-01 -1.52329379e-02 -1.27910470e-01
-6.37527390e-02 -8.60527638e-02  7.64375495e-02  3.06975206e-01
-1.11605118e+00  3.27109013e-01  6.69683726e-01 -1.09914897e-02
 1.20004326e+00 -1.07577329e+00 -6.05765023e-01  3.43177439e-01
-3.20554330e+00  3.91481400e-01  2.61453085e-01  2.85233418e-01
-3.84414468e-01  2.73487789e-01  1.99248034e+00  7.44271950e-02
-2.39349077e-01 -1.34606910e+00 -1.50360718e+00 -1.07501082e+00
-4.60818584e-04  4.07758491e-01 -2.27053847e+00 -1.53719453e+00
-1.36512309e+00  1.81008468e+01 -1.31428878e+00 -1.54757187e+00
-1.50924217e+00 -2.32691994e+00 -1.62546569e+00  6.49015527e-01
 8.45360972e-01  1.46980862e+00  4.95075687e-04  3.84643253e-01
-4.65906083e-02  1.12794144e+00 -1.07610765e+00 -7.91716633e-01
 4.78232398e-02  5.61284675e-01  8.78974364e-02 -1.40763703e-01
-7.47673166e-03 -5.57207785e-01 -1.87310196e-01  1.92008270e+00
-6.89270399e-02  7.11882646e-02  3.64500894e-01 -4.99690729e-03
 9.72284031e-01 -7.62831000e-01 -4.72796588e-01  3.95462805e-01
-4.64533315e+00  4.41464769e-01  2.49141118e-01  3.49191694e-01
-4.98058478e-02  1.73493302e-01  2.16684149e+00  5.71270817e-01
-2.85187568e-01  2.98000236e-10 -2.28928892e-03 -5.39646350e-04
-1.11336068e-03 -4.72699295e-02 -2.66885481e-05 -4.00148761e-02
-8.85450028e-03 -5.78996138e-03 -2.94580096e-03 -2.18884673e-02
-1.76219234e-03 -7.45438488e-04 -1.96280007e-01 -1.53802302e+00
 3.60609270e+00 -6.10631901e-01  8.43909235e-01 -1.74158988e+01
 9.59386110e-01 -1.16661533e-01  1.45546855e+00  3.56590300e+00
 2.04579486e+00  1.19717248e+01 -9.41267821e+00 -9.33368353e-01
-3.17923222e+00 -1.68136988e+00 -1.20408381e+00  1.31155084e-01
-1.05275882e+00  4.42894265e-01 -1.06284306e+00  4.73344682e+00
-1.31915795e+00 -6.93383332e+00  1.39397360e+01  1.23246238e+00
-8.16625671e-01 -7.89459148e-01  2.37174782e+01 -7.63985126e-01
-1.60490044e+00 -4.38951750e-01 -3.25137952e+00 -1.63032293e+00
 2.76180283e+00 -1.06020335e+01  8.68046569e-01  3.77662237e-02
 1.05657575e+00  1.20898868e-01 -6.60451465e-02  2.57433767e-02
-6.64350825e-01 -1.07160709e-01  1.70038815e+00 -2.37169986e-01
 3.64764103e-02  3.06315567e-01 -1.81474863e-01  1.47412264e+00
 1.08642830e+00 -9.99526381e+00 -1.23073399e+00 -1.52811066e+01
 4.75637421e+00 -7.64829366e-01  8.34792429e-02  5.27188343e-03
 5.98343582e-02 -4.48566693e-01 -8.71397386e-02  1.82377609e+00
-1.68351226e-01 -1.33837334e-02  3.35480396e-01 -5.65832749e-01
-2.18853381e+00 -2.92102972e-01  2.26715286e+00  1.59859799e+00
 1.05305230e-01  6.01289500e-02  1.23505967e-01 -1.12830333e-02
 2.24196301e+00 -1.94928282e+00  1.45957463e-01 -2.34324487e+00
 1.32313625e+00 -5.12695447e+00  1.80859992e+01  4.96437940e-01
 2.89547964e-01  1.59328083e+00  1.12308843e+00  2.22164804e-01
 1.05553693e-01 -5.55366491e+00 -1.69592682e+00 -1.27744613e+01
 2.24298695e-01 -7.05922599e-02] x + -44.35328270803841
```

Se imprimieron los coeficientes y el intercepto del modelo, formando la ecuación de la recta en forma matricial para describir la relación entre las variables. Finalmente, se generó una gráfica para visualizar la comparación entre los valores reales y las predicciones, lo que facilita la comprensión del rendimiento del modelo.

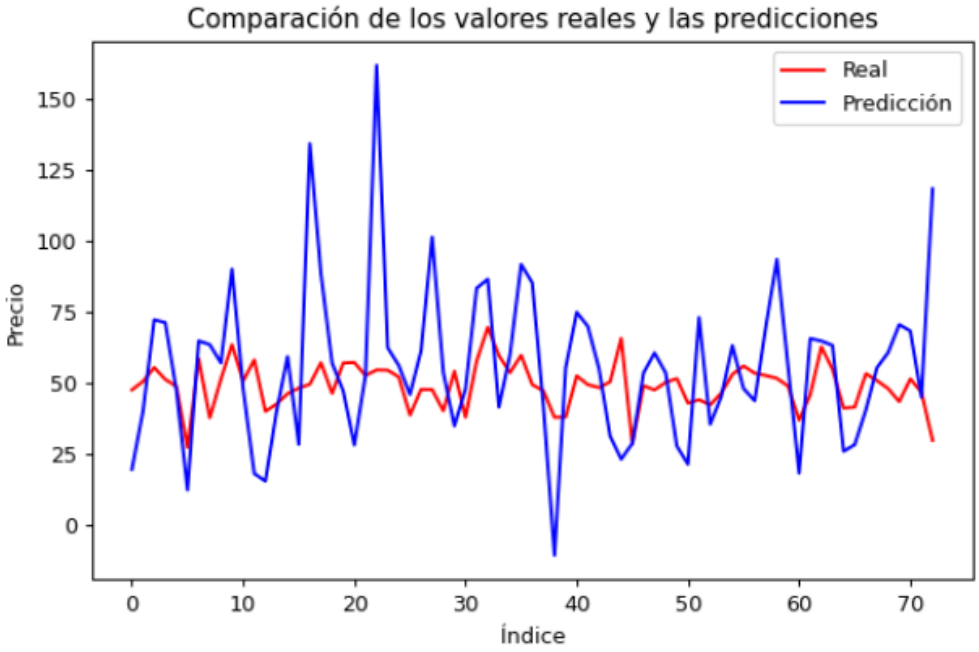


Figura 14.Gráfica de comparación de valores Reales y Predicciones del modelo de regresión Polinomial de grado 2 (Elaboración propia)

Además, se calcularon métricas de evaluación, como el error cuadrático medio y el coeficiente de determinación, para medir la precisión del modelo.

Este enfoque de regresión polinomial nos permite capturar relaciones más complejas entre las variables y obtener predicciones más precisas en comparación con un modelo de regresión lineal simple.

```
Error cuadrático medio: 750.3347439802386
Coeficiente de determinación R^2: -10.77412437605147
```

Podemos ver un coeficiente de determinación R^2 de -10.77412437605147 lo que nos indica que el modelo de regresión polinomial no se ajusta bien a los datos. Un valor negativo de R^2 sugiere que el modelo es peor que un modelo constante que simplemente predice la media de los valores observados. Podemos ver también que en el grafico los valores son extraños, por lo que podemos decir que no es un buen estimador de los precios eléctricos.

Por ese motivo y al tener un r^2 fuera del rango de 0 a 1 hemos decidió intentar arreglado cambiando el grado del polinomio para ver el comportamiento, en el caso de aumentar el polinomio a 3 tenemos:

```
Modelo de regresión polinomial
         Real  predicción
0   47.430417   38.913789
1   50.403750   53.067506
2   55.286667   56.294935
3   51.112500   30.891625
4   48.639167   52.297350
..        ...         ...
```

```
68  47.886667   50.736727
69  43.298333   44.491880
70  51.290000   53.055803
71  46.567917   48.388220
72  29.685833   48.445200
```

Ecuación de la recta en forma matricial: `y= [ 1.93477650e-07  4.07712299e-07 1.74603204e-07 ... -4.69288828e-09  2.49730428e-06 -8.50257841e-06] x + -1773.2894635938505`

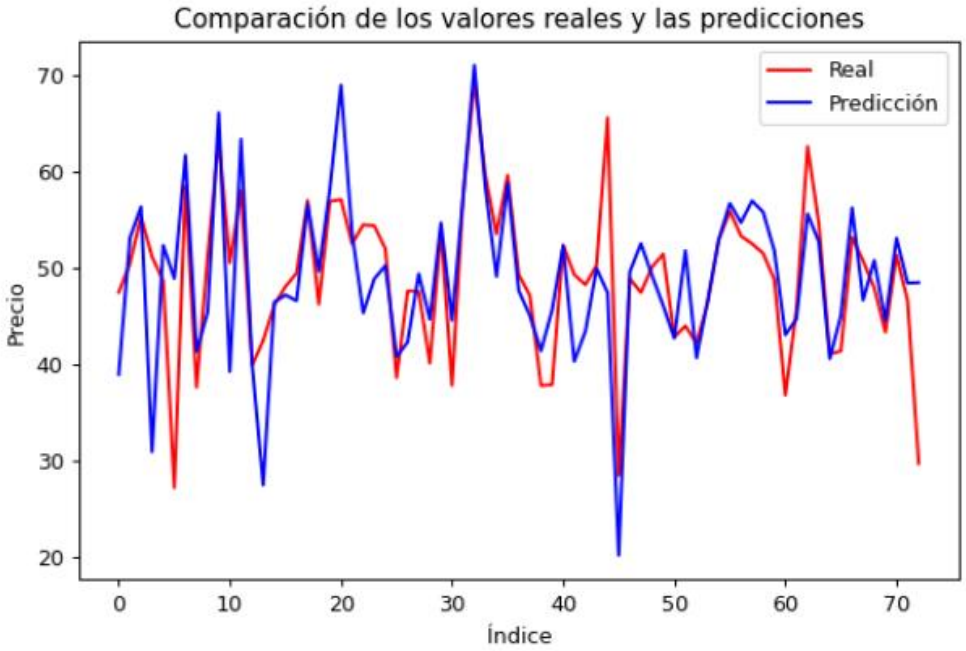


Figura 15. Gráfica de comparación de valores Reales y Predicciones del modelo de regresión Polinomial de grado 3 (Elaboración propia)

Y los valores de análisis:

```
Error cuadrático medio: 42.149858966013944
R2: 0.3385916274297611
```

En este caso podemos ver que ele r^2 si nos da un valor entre el 0 y el 1, con lo cual está dentro del rango teórico de análisis. Por lo que podemos decir que el modelo de ajusta a los datos correctamente al aumentar el grado del polinomio a 3, pero sigue siendo un r^2 bajo por lo que podemos decir que este modelo no es un buen modelo para predecir los precios eléctricos futuros.

Regresión de Ridge

Primero decir que el modelo de regresión de Ridge es un enfoque de regresión lineal regularizada que busca minimizar la suma de los errores al cuadrado y, al mismo tiempo, reducir la complejidad del modelo. Para estudiar el modelo hemos seguido los mimos pasos que en los modelos anteriores

En primer lugar, hemos dividido los datos en variables independientes (X) y la variable dependiente (y). A continuación, se separan los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split de sklearn.model_selection. En este caso, se

utiliza un tamaño de prueba del 20% y una semilla aleatoria de 10 para asegurar la reproducibilidad de los resultados.

Luego, se crea una instancia del modelo de regresión de Ridge con un parámetro de regularización (alfa) establecido en 0.05. Este parámetro controla la cantidad de penalización aplicada a los coeficientes del modelo. Un valor más alto de alfa conduce a una mayor regularización y, por lo tanto, a una mayor reducción de la complejidad del modelo.

A continuación, se entrena el modelo de regresión de Ridge utilizando los datos de entrenamiento (X_train y y_train). Durante el entrenamiento, el modelo ajusta los coeficientes del modelo para minimizar la suma de los errores al cuadrado, teniendo en cuenta la penalización establecida por el valor de alfa.

Una vez entrenado el modelo, se realizan predicciones en el conjunto de prueba utilizando el método predict del modelo de regresión de Ridge. Las predicciones se almacenan en la variable y_pred y para evaluar el desempeño del modelo, se comparan los valores reales de la variable objetivo (y_test) con las predicciones realizadas (y_pred). Esto se hace creando un dataframe que muestra los valores reales y las predicciones lado a lado., que podemos ver como ejemplo:

```
     Modelo de regresion de Ridge
Ridge(alpha=0.05)
          Real  predicción
0   47.112917   48.861822
1   53.027917   52.765802
2   29.685833   42.661427
3   63.319583   62.252826
4   51.894167   49.562713
..        ...         ...
68  17.433333   23.812179
69  51.887500   51.479868
70  54.637083   55.189383
71  40.071250   46.709283
72  47.886667   47.357318
```

Se a genera una gráfica para visualizar la comparación entre los valores reales y las predicciones. Esto nos permite tener una representación gráfica de cómo el modelo se ajusta a los datos.

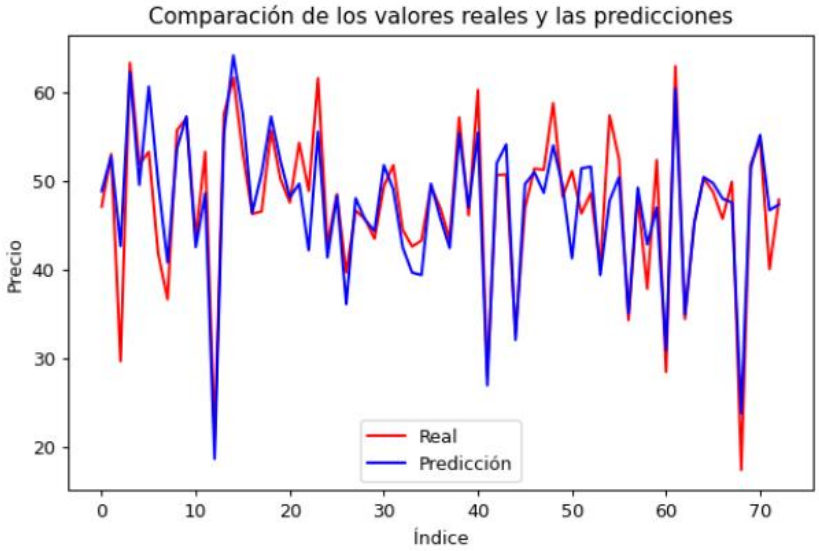


Figura 16. Gráfica de comparación de valores Reales y Predicciones del modelo de regresión Ridge (Elaboración propia)

Además, se calcula el error cuadrático medio utilizando la función mean_squared_error de sklearn.metrics. Este valor nos da una medida de la calidad de las predicciones del modelo, siendo deseable que sea lo más bajo posible.

```
Error cuadrático medio: 14.230008670672673
Coeficiente de determinación R^2: 0.8240006964116204
```

En resumen, el modelo de regresión de Ridge es una técnica útil para realizar regresiones lineales regularizadas. Permite reducir la complejidad del modelo y evitar el sobreajuste al introducir un término de penalización. Podemos ver como los datos de prueba son similares en los reales y en las predicciones por lo que podemos decir que es un buen modelo de análisis.

Regresión Bayesiana

El modelo de regresión bayesiana está enfocado en una regresión que utiliza métodos bayesianos para estimar los parámetros del modelo y realizar predicciones. A diferencia de los modelos que hemos visto antes la regresión bayesiana permite incorporar información previa o conocimiento experto en el proceso de inferencia, en este caso nosotros no informaremos de ninguna información anterior, cogiendo por defecto los datos del modelo.

En el código que hemos desarrollado sigue los mismos pasos que los anteriores, se implementa un modelo de regresión bayesiana utilizando la clase BayesianRidge del módulo sklearn.linear_model.

Primero, los datos se dividen en características (X) y la variable objetivo (y) utilizando la función train_test_split de sklearn.model_selection. Se utiliza un tamaño de prueba del 20% y una semilla aleatoria de 10 para asegurar la reproducibilidad de los

resultados. Después , se crea una instancia del modelo de regresión bayesiana mediante la inicialización de la clase BayesianRidge y se asigna a la variable bayesianRidge.

Luego, el modelo ejecuta su fase de entrenamiento utilizando los datos de entrenamiento (X_train y y_train) mediante el método fit. Durante el entrenamiento, el modelo estima los parámetros del modelo utilizando métodos bayesianos y se ajusta a los datos. Una vez entrenado el modelo, se realizan predicciones en el conjunto de prueba utilizando el método predict. Las predicciones se almacenan en la variable y_pred, y se puede observar los siguientes datos de predicción en comparación con los reales:

```
        Modelo de regresion Bayesiana
BayesianRidge()
          Real  predicción
0   47.112917   49.002203
1   53.027917   53.471326
2   29.685833   41.633746
3   63.319583   62.077755
4   51.894167   48.918363
..        ...         ...
68  17.433333   24.195787
69  51.887500   51.041717
70  54.637083   55.667718
71  40.071250   45.125887
72  47.886667   46.724344
```

Se genera una gráfica para visualizar la comparación entre los valores reales y las predicciones. Esto nos permite tener una representación gráfica de cómo el modelo se ajusta a los datos.

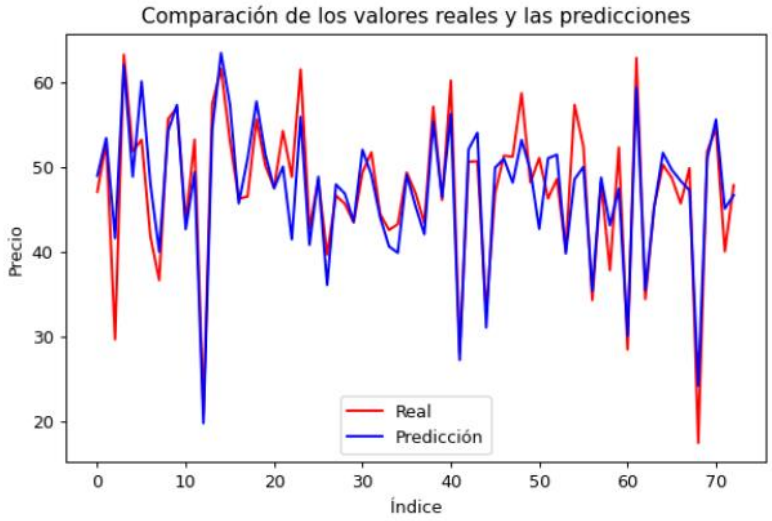


Figura 17. Gráfica de comparación de valores Reales y Predicciones del modelo de regresión Bayesiana (Elaboración propia)

Además, se calcula el error cuadrático medio utilizando la función mean_squared_error de sklearn.metrics:

```
Error cuadrático medio: 12.618771377931617
Coeficiente de determinación R^2: 0.8439287686989192
```

En resumen, el modelo de regresión bayesiana utiliza métodos bayesianos para realizar la regresión y obtener predicciones. Podemos ver, que el modelo en comparación con el anterior ha mejora , vemos una explicación mejor de los datos y una similitud en la gráfica que nos muestra que es un buen modelo para predecir precios.

Proceso Gaussiano

El modelo de Proceso Gaussiano es la técnica que utilizada en el aprendizaje automático para modelar y predecir valores basados en una distribución gaussiana. Para nuestro desarrollo hemos seguido el mismo proceso que los anteriores

En primer lugar, separamos los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split de la biblioteca sklearn.model_selection. Esto nos permite evaluar la capacidad de generalización del modelo en datos no vistos. En este caso, se utiliza un tamaño de prueba del 20% y se establece una semilla aleatoria para asegurar la reproducibilidad de los resultados.

después, definimos el kernel del proceso gaussiano que es el encargado de describir la relación entre las variables. En este modelo, utilizamos el kernel de producto puntual (DotProduct) combinado con un kernel de ruido blanco (WhiteKernel). El kernel de producto puntual modela las relaciones lineales entre las variables, mientras que el kernel de ruido blanco captura el ruido aleatorio presente en los datos.

después de configurar el modelo creamos una instancia del modelo de Proceso Gaussiano (GaussianProcessRegressor) utilizando el kernel definido y una semilla aleatoria. Y empezamos el entrenamiento del modelo utilizando los datos de entrenamiento (X_train e y_train). El modelo ajusta los parámetros del kernel para adaptarse a los datos y aprender la estructura subyacente. Después de entrenar el modelo, realizamos predicciones utilizando los datos de prueba (X_test). Estas predicciones nos permiten evaluar el rendimiento del modelo en datos no vistos, podemos ver que nos da:

```
Modelo de proceso Gaussiano
GaussianProcessRegressor(kernel=DotProduct(sigma_0=1
) + WhiteKernel(noise_level=1),
                         random_state=0)
         Real  predicción
0   47.112917   49.298285
1   53.027917   52.754925
2   29.685833   42.539162
3   63.319583   61.989203
4   51.894167   49.737571
..        ...         ...
68  17.433333   23.716447
69  51.887500   51.064063
70  54.637083   55.284260
71  40.071250   46.617064
72  47.886667   46.840777
```

Generamos una gráfica que muestra la comparación entre los valores reales y las predicciones del modelo. Esto nos permite visualizar cómo se ajusta el modelo a los datos y evaluar su desempeño.

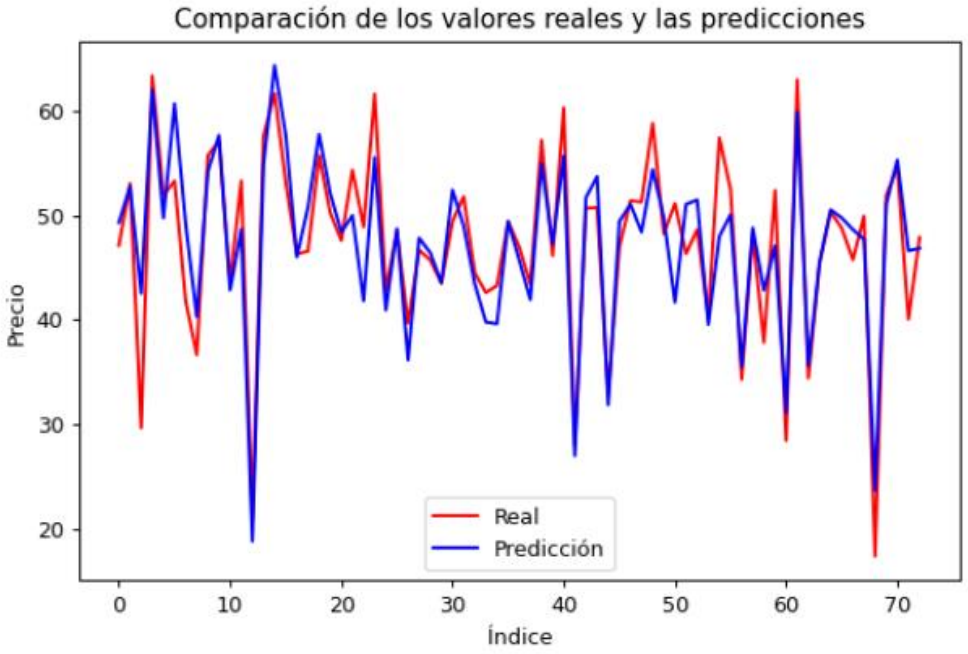


Figura 18. Gráfica de comparación de valores Reales y Predicciones del modelo de proceso gaussiano (Elaboración propia)

Calculamos el error cuadrático medio utilizando la función mean_squared_error de la biblioteca sklearn.metrics. Y calculamos el coeficiente de determinación R^2 utilizando el método score del modelo de Proceso Gaussiano.

```
Error cuadrático medio: 13.893275765409523
Coeficiente de determinación R^2: 0.8281654694762882
```

En resumen, el modelo de Proceso Gaussiano es una técnica poderosa para modelar y predecir valores basados en una distribución gaussiana. Se basa en la estimación de parámetros y utiliza un kernel para describir la relación entre las variables, se podría probar diferentes tipos de kernel para realizar más pruebas, en este caso podemos ver que el modelo predice correctamente los resultados por lo que podemos decir que es un buen modelo para predecir precios.

Red Neuronal

El modelo de Red Neuronal Secuencial que hemos escogido es una técnica de aprendizaje automático inspirada en el funcionamiento del cerebro humano. Vamos a explicar paso a paso su desarrollo, similar y siguiendo los principios que hemos indicado anteriormente, en este caso iremos observando el código desarrollado:

En primer lugar, hemos separamos los datos en variables independientes (X) y variable dependiente (y) utilizando la función iloc de la biblioteca pandas. A continuación, hemos dividido los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split de la biblioteca sklearn.model_selection. Esto nos permite evaluar el rendimiento del modelo en datos no vistos. En este caso, se utiliza un tamaño de prueba del 20% y se establece una semilla aleatoria para garantizar la reproducibilidad de los resultados.

```
# Datos x como variables independientes y y como variable dependiente
#y es la ultima columna
X = datos.iloc[:, :-1].values
#x es todo menos la ultima columna
y = datos.iloc[:, -1].values

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
```

Luego para evitar errores y por cómo funciona el sistema de red neuronal, hemos normalizamos los datos utilizando la técnica de estandarización mediante la clase StandardScaler de la biblioteca sklearn.preprocessing. Esto asegura que todas las variables tengan la misma escala y facilita el entrenamiento de la red neuronal.

```
# Normalizar los datos
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

después, hemos creamos el modelo de red neuronal utilizando la clase Sequential de la biblioteca keras. Este modelo se compone de capas conectadas entre ellas, es decir diferentes nodos de conexión. Utilizamos la función de activación 'relu' en las capas ocultas y no se aplica ninguna función de activación en la capa de salida, ya que estamos realizando una regresión.

```
# Crear la red neuronal
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(16, activation='relu'))
model.add(Dense(1))
```

Compilamos el modelo especificando el optimizador y la función de pérdida. En este caso, utilizamos el optimizador 'adam' y la pérdida se calcula mediante el error cuadrático medio. Definido esto entrenamos el modelo utilizando el método fit y los datos de entrenamiento. Especificamos el número de épocas y el tamaño del lote

(batch_size) para el entrenamiento. Durante el entrenamiento, el modelo ajusta los pesos de las conexiones entre las neuronas para minimizar la pérdida.

```
# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')

# Entrenar el modelo
model.fit(X_train, y_train, epochs=500, batch_size=32)
```

Después de entrenar el modelo, realizamos predicciones en el conjunto de prueba utilizando el método predict. Obtenemos las predicciones de los precios (y_pred) y los comparamos con los valores reales del conjunto de prueba.

```
# Predecir los precios en el conjunto de prueba
y_pred = model.predict(X_test)

# Comparar los resultados
df = pd.DataFrame({'Real': y_test, 'Prediccion': y_pred.flatten()})
print(df)
```

Y obtenemos los datos:

```
          Real  predicción
0   47.430417   42.481709
1   50.403750   47.687180
2   55.286667   58.875057
3   51.112500   46.308914
4   48.639167   48.217987
..        ...         ...
68  47.886667   53.619465
69  43.298333   42.763912
70  51.290000   53.245487
71  46.567917   48.200596
72  29.685833   37.093300
```

Calculamos el error cuadrático medio (MSE) utilizando la función mean_squared_error de la biblioteca sklearn.metrics. También calculamos el coeficiente de determinación R^2 utilizando la función r2_score de la biblioteca sklearn.metrics.

```
Error cuadrático medio: 14.299967124239492
R2: 0.7756073634534013
```

Generamos una gráfica que muestra la comparación entre los valores reales y las predicciones del modelo. Esto nos permite visualizar cómo se ajusta el modelo a los datos y evaluar su desempeño.

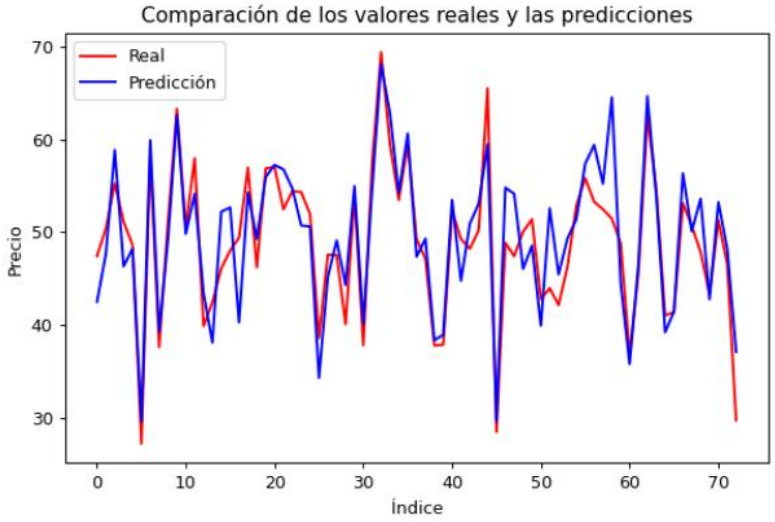


Figura 19. Gráfica de comparación de valores Reales y Predicciones del modelo de red neuronal secuencial (Elaboración propia)

En resumen, el modelo de Red Neuronal es una técnica poderosa para realizar predicciones en problemas de regresión. A través del entrenamiento y las predicciones, podemos evaluar la precisión del modelo utilizando medidas como el error cuadrático medio y el coeficiente de determinación R^2.

Es importante tener en cuenta que el desempeño de la Red Neuronal depende en gran medida de los datos utilizados para el entrenamiento, así como de otros parámetros como el número de repeticiones (epochs) y la arquitectura de la red. Por lo que la red Neuronal es un modelo flexible y adaptable que puede aprender de los datos para realizar predicciones precisas.

5 Conclusiones y trabajo futuros

5.1.1 Conclusiones y contribuciones

En este trabajo, se han evaluado diferentes modelos de Machine Learning para predecir los precios en el mercado eléctrico. Se ha encontrado que el modelo de regresión bayesiana ha logrado proporcionar las estimaciones más precisas, seguido del modelo de proceso gaussiano. Estos modelos han demostrado una capacidad muy alta para capturar relaciones no lineales y manejar la incertidumbre de los datos, lo que los convierte en herramientas útiles para la predicción de precios futuros.

Además, hemos podido observar que los modelos de regresión de Ridge y regresión lineal también han mostrado resultados significativos, aunque con un rendimiento ligeramente inferior en comparación con los modelos bayesianos y gaussianos. Estos modelos han sido capaces de capturar algunas tendencias y patrones en los datos, lo que demuestra su utilidad en el análisis y predicción de precios.

Creemos que este trabajo ha contribuido, a la comparativa detallada de diferentes modelos de Machine Learning aplicados al mercado eléctrico. Los resultados obtenidos pueden servir como referencia para investigadores y profesionales interesados en utilizar técnicas de Machine Learning para el análisis y predicción de precios en este sector.

En resumen, este trabajo ha demostrado y comprobado que los modelos de regresión bayesiana y proceso gaussiano son altamente efectivos en la predicción de precios en el mercado eléctrico, mientras que los modelos de regresión de Ridge y regresión lineal también han mostrado resultados significativos.

5.1.2 Trabajo futuro

En cuanto al trabajo o propuestas de futuro, se pueden considerar diversas direcciones. Una de las más comentadas por os profesionales actualmente incorporación de técnicas de Deep Learning, como redes neuronales convolucionales o recurrentes, para la predicción de precios en el mercado eléctrico.

Además, se puede explorar la aplicación de otros tipos de modelos, como Random Forests o Gradient Boosting, que combinan múltiples modelos para mejorar las predicciones.

Otro aspecto importante para considerar es la evaluación de los modelos en diferentes períodos de tiempo y en diferentes regiones geográficas. Esto permitiría evaluar mejor los modelos y ver si se pueden generalizar y determinar su aplicabilidad en contextos más amplios.

6 Bibliografía

- Alberca, A. S. (2020, 4 octubre). La librería Matplotlib. Aprende con Alf. https://aprendeconalf.es/docencia/python/manual/matplotlib/

- Alberca, A. S. (2022a, mayo 12). La librería Numpy. Aprende con Alf. https://aprendeconalf.es/docencia/python/manual/numpy/

- Alberca, A. S. (2022b, junio 14). La librería Pandas. Aprende con Alf. https://aprendeconalf.es/docencia/python/manual/pandas/

- Chacón, J. L. (2022, 17 agosto). Introducción a Pandas, la librería de Python para trabajar con datos. Profile Software Services. https://profile.es/blog/pandas-python/

- Che, J., & Wang, J. (2010). Short-term electricity prices forecasting based on support vector regression and Auto-regressive integrated moving average modeling. Energy Conversion and Management, 51(10), 1911-1917. https://doi.org/10.1016/j.enconman.2010.02.023

- Chiok, C. H. M. (2014). Modelos de regresión lineal con redes neuronales. Dialnet. https://dialnet.unirioja.es/servlet/articulo?codigo=7025156

- colaboradores de Wikipedia. (2023, 24 febrero). RStudio. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/RStudio

- Cruz, A., Muñoz, A., Zamora, J. C., & Espínola, R. (2011). The effect of wind generation and weekday on Spanish electricity spot price forecasting. Electric Power Systems Research, 81(10), 1924-1935. https://doi.org/10.1016/j.epsr.2011.06.002

- De Meteorología, A. E. (s. f.). AEMET OpenData. Agencia Estatal de Meteorología - AEMET. Gobierno de España. https://www.aemet.es/es/datos_abiertos/AEMET_OpenData

- Electricity Price Forecasting With Extreme Learning Machine and Bootstrapping. (2012, 1 noviembre). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/6184354

- Fernandez, R. (2019, 29 agosto). SciPy: Funciones principales. ▷ Cursos de Programación de 0 a Experto © Garantizados.

https://unipython.com/scipy-funciones-principales/

- Hu, Z., Yu, Y., Wang, Z., Sun, W., Gan, D., & Han, Z. (2004a). Price forecasting using an integrated approach. https://doi.org/10.1109/drpt.2004.1338463

- Scikit-Learn, herramienta básica para el Data Science en Python. (2019, 8 abril). Máster en Data Science. https://www.master-data-scientist.com/scikit-learn-data-science/

- Welcome to. (2023, 15 febrero). Python.org. https://www.python.org/

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

- Roman, V. (2021b, diciembre 9). Algoritmos Naive Bayes: Fundamentos e Implementación. Medium. https://medium.com/datos-y-ciencia/algoritmos-naive-bayes-fudamentos-e-implementaci%C3%B3n-4bcb24b307f
- Orlando. (2022, 21 septiembre). Machine Learning | Qué es, tipos, ejemplos y cómo implementarlo. GraphEverywhere. https://www.grapheverywhere.com/machine-learning-que-es-tipos-ejemplos-y-como-implementarlo/

- Tschora, L., Pierre, E., Plantevit, M., & Robardet, C. (2022). Electricity price forecasting on the day-ahead market using machine learning. Applied Energy, 313, 118752. https://doi.org/10.1016/j.apenergy.2022.118752
- Crypto-Currency price prediction using Decision Tree and Regression techniques. (2019, 1 abril). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8862585/
- Li, Y. (2023). Stock Price Prediction based on Multiple Regression Models. Highlights in Science, Engineering and Technology, 39, 657-662. https://doi.org/10.54097/hset.v39i.6622

7 Anexos

7.1.1 Programaciones realizadas para el análisis.

Adjunto a este trabajo está el código fuente para poder ejecutar el programa de manera directa, aquí adjuntamos el código principal que se ejecuta en un archivo RMD en rstudio:

Análisis de datos Precios electricidad

Programación realizada en lenguaje de programación Python y R, por Carlos Romero Matarin para el TFG de economía de la UAB

Importamos librerías necesarias para el tratamiento de datos de precios

```
import numpy as np
import pandas as pd
import os
import json
```

Variables de ruta del ordenador para obtener los datos:

```
ruta_precios_2019 = 'C:/Users/crome/Desktop/TFG_ECO/Precios_Electricidad/2019/'

ruta_climatica_2019 = 'C:/Users/crome/Desktop/TFG_ECO/datos_climaticos/'

ruta_Estructura_2019 =
'C:/Users/crome/Desktop/TFG_ECO/TFG_Entrega_Parcial_Carlos_Romero_1518680/estructura/
estructura_2019.xlsx'

ruta_Demanda_2019 =
'C:/Users/crome/Desktop/TFG_ECO/TFG_Entrega_Parcial_Carlos_Romero_1518680/Demanda/dem
anda_2019.xlsx'
```

Analizamos los datos de 2019, obtenidos desde la página oficial de OMIE (Operador del mercado diario eléctrico)

https://www.omie.es/es/file-access-list?parents%5B0%5D=/&parents%5B1%5D=Mercado%20Diario&parents%5B2%5D=1.%20Precios&dir=Precios%20horarios%20del%20mercado%20diario%20en%20Espa%C3%B1a&realdir=marginalpdbc

Tratamiento de datos:

```
#Analisis de datos 2019

# Importar todos los datos de diferentes archivos en una sola tabla

ruta = ruta_precios_2019

#numero total de archivos dentro de la carpeta
num_archivos = len(os.listdir(ruta))

#lista de todos los archivos dentro de la carpeta
lista_archivos = os.listdir(ruta)
```

```
#recorrer todos los archivos y guardarlos en una lista
lista_datos = []
for i in range(num_archivos):
    archivo = lista_archivos[i]

    #leer el archivo y guardar los datos en un dataframe
    datos = pd.read_csv(ruta + '/' + archivo, sep = ';', decimal = '.')

    lista_datos.append(datos)

#concatenar todos los archivos en un solo dataframe
datos = pd.concat(lista_datos)

#Eliminar filas con nada en la primera columna con indice 0
datos = datos.dropna(subset = [datos.columns[0]])

#Eliminar ultima columna
datos = datos.drop(datos.columns[-1], axis = 1)

#pasar a csv
datos.to_csv('2019.csv', sep = ';', decimal = '.')

# Importar datos de 2019
datos = pd.read_csv('2019.csv', sep = ';', decimal = '.')

#renombrar columnas
datos = datos.rename(columns = {'Unnamed: 0': 'Año', 'Unnamed: 1': 'Mes', 'Unnamed:
2': 'Dia', 'Unnamed: 3': 'Hora', 'Unnamed: 4': 'Precio'})

#eliminar ultima columna
datos = datos.drop(datos.columns[-1], axis = 1)

#datos de dolumas año, mes, dia y hora de decimal a entero
datos['Año'] = datos['Año'].astype(int)
datos['Mes'] = datos['Mes'].astype(int)
datos['Dia'] = datos['Dia'].astype(int)
datos['Hora'] = datos['Hora'].astype(int)

#eliminar columnas que hora
datos = datos.drop(datos.columns[3], axis = 1)

#agrupar por año, mes y dia
datos = datos.groupby(['Año', 'Mes', 'Dia']).mean()

#pasar a csv pepe.csv
datos.to_csv('datos_Media_Diario.csv', sep = ';', decimal = '.')

datos = pd.read_csv('datos_Media_Diario.csv', sep = ';', decimal = '.')
```

Traemos datos climáticos desde la web oficinal de AEMET, los datos han sido descargados por medio de la API que se puede obtener en la web, para realizar más rápido los cálculos hemos descargado los datos en una carpeta con archivos del tipo .json, los datos desde la API viene en formato aray json.

Tratamiento de datos:

```
#datos de barcelona
with open(ruta_climatica_2019+'datosBarcelona.json') as file:
    datosJsonBarcelona = json.load(file)

#convertir array json a dataframe
```

```
#{'fecha': '2019-01-01', 'indicativo': '0201D', 'nombre': 'BARCELONA',
# 'provincia': 'BARCELONA', 'altitud': '6', 'tmed': '11,4',
# 'prec': '0,0', 'tmin': '6,6', 'horatmin': '03:40', 'tmax': '16,2',
# 'horatmax': '12:30', 'dir': '01', 'velmedia': '3,1', 'racha': '6,4',
# 'horaracha': '07:30'}
dfBarcelona = pd.DataFrame(datosJsonBarcelona)

#datos de madrid
with open(ruta_climatica_2019+'datosMadrid.json') as file:
    datosJsonMadrid = json.load(file)

#convertir array json a dataframe
dfMadrid = pd.DataFrame(datosJsonMadrid)

#datos de sevilla
with open(ruta_climatica_2019+'datosSevilla.json') as file:
    datosJsonSevilla = json.load(file)

#convertir array json a dataframe
dfSevilla = pd.DataFrame(datosJsonSevilla)

#datos de a coruña
with open(ruta_climatica_2019+'datosCoruna.json') as file:
    datosJsonCoruna = json.load(file)

#convertir array json a dataframe
dfCoruna = pd.DataFrame(datosJsonCoruna)

#datos de navarra
with open(ruta_climatica_2019+'datosNavarra.json') as file:
    datosJsonNavarra = json.load(file)

#convertir array json a dataframe
dfNavarra = pd.DataFrame(datosJsonNavarra)

#unir los dataframes en uno solo
df = pd.concat([dfBarcelona, dfMadrid, dfSevilla, dfCoruna, dfNavarra],
ignore_index=True)

#eliminar columnas que no se van a utilizar
df = df.drop(['indicativo', 'altitud', 'horatmin', 'horatmax', 'dir', 'velmedia',
'racha', 'horaracha','presMax', 'horaPresMax','presMin','horaPresMin','sol'], axis=1)



#diferentes valores que puede tomar la columna prec
#print(df['prec'].unique())
# ['Ip' 'Acum' '9,9' '9,8' '9,6' '9,5' '9,4' '9,3' '9,1' '9,0' '80,9' '8,8'
#  '8,6' '8,5' '8,4' '8,2' '8,0' '7,8' '7,7' '7,6' '7,4' '7,3' '7,2' '7,0'
#  '6,8' '6,7' '6,6' '6,4' '6,3' '6,2' '6,1' '6,0' '58,3' '53,2' '50,0'
#  '5,9' '5,8' '5,7' '5,6' '5,4' '5,3' '5,2' '5,1' '48,6' '47,2' '42,2'
#  '42,0' '4,8' '4,7' '4,6' '4,4' '4,3' '4,2' '4,1' '4,0' '39,1' '38,5'
#  '38,4' '36,7' '36,6' '34,4' '32,4' '3,9' '3,8' '3,6' '3,5' '3,4' '3,3'
#  '3,2' '3,1' '3,0' '29,7' '29,6' '28,5' '28,2' '27,6' '27,3' '27,2' '26,9'
#  '26,8' '26,4' '25,6' '25,2' '24,9' '24,8' '24,0' '23,8' '23,2' '23,0'
#  '22,8' '22,6' '21,8' '21,6' '20,6' '20,0' '2,9' '2,8' '2,7' '2,6' '2,5'
#  '2,4' '2,3' '2,2' '2,1' '2,0' '18,6' '18,4' '18,3' '18,2' '18,1' '18,0'
#  '17,9' '17,8' '17,4' '17,3' '17,2' '17,0' '16,9' '16,8' '16,2' '15,8'
#  '15,6' '15,2' '15,0' '14,9' '14,6' '14,4' '13,9' '13,7' '13,4' '13,2'
#  '12,9' '12,8' '12,6' '12,4' '12,1' '12,0' '11,8' '11,6' '11,2' '11,1'
#  '10,9' '10,8' '10,6' '10,4' '10,2' '10,1' '10,0' '1,9' '1,8' '1,7' '1,6'
#  '1,5' '1,4' '1,3' '1,2' '1,1' '1,0' '0,9' '0,8' '0,7' '0,6' '0,5' '0,4'
#  '0,3' '0,2' '0,1' '0,0' nan]
```

```
#Convertir datos de la columna prec (ip, acum) a nan
df['prec'] = df['prec'].replace(['Ip', 'Acum'], np.nan)

#convertir , por . en los datos de la columna prec
df['prec'] = df['prec'].str.replace(',','.')

#convertir datos de la columna prec a float
df['prec'] = df['prec'].astype(float)

#convertir , por . en los datos de la columna tmin
df['tmin'] = df['tmin'].str.replace(',','.')

#convertir datos de la columna tmin a float
df['tmin'] = df['tmin'].astype(float)

#convertir , por . en los datos de la columna tmax
df['tmax'] = df['tmax'].str.replace(',','.')

#convertir datos de la columna tmax a float
df['tmax'] = df['tmax'].astype(float)

#convertir , por . en los datos de la columna tmed
df['tmed'] = df['tmed'].str.replace(',','.')

#convertir datos de la columna tmed a float
df['tmed'] = df['tmed'].astype(float)

#valores nan de la columna prec a la media de la columna
df['prec'] = df['prec'].fillna(df['prec'].mean())

#valores nan de la columna tmin a la media de la columna
df['tmin'] = df['tmin'].fillna(df['tmin'].mean())

#valores nan de la columna tmax a la media de la columna
df['tmax'] = df['tmax'].fillna(df['tmax'].mean())

#valores nan de la columna tmed a la media de la columna
df['tmed'] = df['tmed'].fillna(df['tmed'].mean())

#convertir datos de la columna fecha a datetime
df['fecha'] = pd.to_datetime(df['fecha'], format='%Y-%m-%d')

#convertir datos de la columna fecha a datetime
df['fecha'] = pd.to_datetime(df['fecha'], format='%Y-%m-%d')

#agrupar por fecha y calcular la media de los datos
df = df.groupby('fecha').mean()

#separar la columna fecha en dia, mes y año
df['Dia'] = df.index.day
df['Mes'] = df.index.month
df['Año'] = df.index.year

#convertir datos de la columna dia a int
df['Dia'] = df['Dia'].astype(int)

#convertir datos de la columna mes a int
df['Mes'] = df['Mes'].astype(int)

#convertir datos de la columna año a int
df['Año'] = df['Año'].astype(int)
```

```
DatosMeteorologicos = df
```

Traemos datos de estructura de la generación eléctrica diaria, estos datos están en formato csv

Tratamiento de datos:

```
#Lectura del fichero excel
df_Estructura_2019 = pd.read_excel(ruta_Estructura_2019)

# pasar filas a columnas
df_Estructura_2019 = df_Estructura_2019.transpose()

#primera fila y primera columna poner fecha
df_Estructura_2019.iloc[0,0] = 'Fecha'

#primera fila como cabecera
new_header = df_Estructura_2019.iloc[0] #
df_Estructura_2019 = df_Estructura_2019[1:]
df_Estructura_2019.columns = new_header

# sacar fecha de la primera columna y dividirla en año, mes y dia, la fecha esta en
01/ene/19
df_Estructura_2019['Año'] = df_Estructura_2019['Fecha'].str.split('/').str[2]
df_Estructura_2019['Mes'] = df_Estructura_2019['Fecha'].str.split('/').str[1]
df_Estructura_2019['Dia'] = df_Estructura_2019['Fecha'].str.split('/').str[0]

#convertir a entero
df_Estructura_2019['Año'] = df_Estructura_2019['Año'].astype(int)
df_Estructura_2019['Dia'] = df_Estructura_2019['Dia'].astype(int)

# pasar de mes ene, feb, mar, etc a numero
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['ene'], 1)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['feb'], 2)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['mar'], 3)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['abr'], 4)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['may'], 5)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['jun'], 6)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['jul'], 7)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['ago'], 8)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['sep'], 9)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['oct'], 10)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['nov'], 11)
df_Estructura_2019['Mes'] = df_Estructura_2019['Mes'].replace(['dic'], 12)

#año en formato YYYY no YY
df_Estructura_2019['Año'] = df_Estructura_2019['Año'] + 2000

#borrar columna fecha
df_Estructura_2019 = df_Estructura_2019.drop(['Fecha'], axis=1)

#valors nulos a 0
df_Estructura_2019 = df_Estructura_2019.fillna(0)

#renplazar - por 0
df_Estructura_2019 = df_Estructura_2019.replace('-', 0)

#convertir a float los valores de todas las columnas

#primero reemplazar , por . para poder convertir a float
```

```
df_Estructura_2019['Hidráulica'] = df_Estructura_2019['Hidráulica'].str.replace(',',
'.')
df_Estructura_2019['Turbinación bombeo'] = df_Estructura_2019['Turbinación
bombeo'].str.replace(',', '.')
df_Estructura_2019['Nuclear'] = df_Estructura_2019['Nuclear'].str.replace(',', '.')
df_Estructura_2019['Carbón'] = df_Estructura_2019['Carbón'].str.replace(',', '.')
df_Estructura_2019['Fuel + Gas'] = df_Estructura_2019['Fuel + Gas'].str.replace(',',
'.')
df_Estructura_2019['Motores diésel'] = df_Estructura_2019['Motores
diésel'].str.replace(',', '.')
df_Estructura_2019['Turbina de gas'] = df_Estructura_2019['Turbina de
gas'].str.replace(',', '.')
df_Estructura_2019['Turbina de vapor'] = df_Estructura_2019['Turbina de
vapor'].str.replace(',', '.')
df_Estructura_2019['Ciclo combinado'] = df_Estructura_2019['Ciclo
combinado'].str.replace(',', '.')
df_Estructura_2019['Hidroeólica'] =
df_Estructura_2019['Hidroeólica'].str.replace(',', '.')
df_Estructura_2019['Eólica'] = df_Estructura_2019['Eólica'].str.replace(',', '.')
df_Estructura_2019['Solar fotovoltaica'] = df_Estructura_2019['Solar
fotovoltaica'].str.replace(',', '.')
df_Estructura_2019['Solar térmica'] = df_Estructura_2019['Solar
térmica'].str.replace(',', '.')
df_Estructura_2019['Otras renovables'] = df_Estructura_2019['Otras
renovables'].str.replace(',', '.')
df_Estructura_2019['Cogeneración'] =
df_Estructura_2019['Cogeneración'].str.replace(',', '.')
df_Estructura_2019['Residuos no renovables'] = df_Estructura_2019['Residuos no
renovables'].str.replace(',', '.')
df_Estructura_2019['Residuos renovables'] = df_Estructura_2019['Residuos
renovables'].str.replace(',', '.')
df_Estructura_2019['Generación total'] = df_Estructura_2019['Generación
total'].str.replace(',', '.')

#convertir a float
df_Estructura_2019['Hidráulica'] = df_Estructura_2019['Hidráulica'].astype(float)
df_Estructura_2019['Turbinación bombeo'] = df_Estructura_2019['Turbinación
bombeo'].astype(float)
df_Estructura_2019['Nuclear'] = df_Estructura_2019['Nuclear'].astype(float)
df_Estructura_2019['Carbón'] = df_Estructura_2019['Carbón'].astype(float)
df_Estructura_2019['Fuel + Gas'] = df_Estructura_2019['Fuel + Gas'].astype(float)
df_Estructura_2019['Motores diésel'] = df_Estructura_2019['Motores
diésel'].astype(float)
df_Estructura_2019['Turbina de gas'] = df_Estructura_2019['Turbina de
gas'].astype(float)
df_Estructura_2019['Turbina de vapor'] = df_Estructura_2019['Turbina de
vapor'].astype(float)
df_Estructura_2019['Ciclo combinado'] = df_Estructura_2019['Ciclo
combinado'].astype(float)
df_Estructura_2019['Hidroeólica'] = df_Estructura_2019['Hidroeólica'].astype(float)
df_Estructura_2019['Eólica'] = df_Estructura_2019['Eólica'].astype(float)
df_Estructura_2019['Solar fotovoltaica'] = df_Estructura_2019['Solar
fotovoltaica'].astype(float)
df_Estructura_2019['Solar térmica'] = df_Estructura_2019['Solar
térmica'].astype(float)
df_Estructura_2019['Otras renovables'] = df_Estructura_2019['Otras
renovables'].astype(float)
df_Estructura_2019['Cogeneración'] = df_Estructura_2019['Cogeneración'].astype(float)
df_Estructura_2019['Residuos no renovables'] = df_Estructura_2019['Residuos no
renovables'].astype(float)
df_Estructura_2019['Residuos renovables'] = df_Estructura_2019['Residuos
renovables'].astype(float)
df_Estructura_2019['Generación total'] = df_Estructura_2019['Generación
total'].astype(float)

#valores nulos a 0
```

```
df_Estructura_2019 = df_Estructura_2019.fillna(0)

DatosEstructura = df_Estructura_2019
```

Traemos datos de la demanda eléctrica diaria, estos datos están en formato csv

Tratamiento de datos:

```
#Lectura del fichero excel
df_Demanda_2019 = pd.read_excel(ruta_Demanda_2019)

#    Column1     Column2     Column3     Column4     Column5     Column6  ...
Column361          Column362   Column363          Column364   Column365
Column366
# 0            01/ene/19   02/ene/19   03/ene/19   04/ene/19   05/ene/19  ...
26/dic/19          27/dic/19   28/dic/19          29/dic/19   30/dic/19
31/dic/19
# 1  Demanda  582,949806  742,199407  787,764963

#Eliminacion de la primera columna
df_Demanda_2019 = df_Demanda_2019.drop(df_Demanda_2019.columns[0], axis = 1)

#convertir columanas en filas
df_Demanda_2019 = df_Demanda_2019.transpose()

#renombrar columnas, columna 1 es la fecha y columna 2 es la demanda
df_Demanda_2019 = df_Demanda_2019.rename(columns = {0: 'Fecha', 1: 'Demanda'})

# sacar fecha de la primera columna y dividirla en año, mes y dia, la fecha esta en
01/ene/19
df_Demanda_2019['Año'] = df_Demanda_2019['Fecha'].str.split('/').str[2]
df_Demanda_2019['Mes'] = df_Demanda_2019['Fecha'].str.split('/').str[1]
df_Demanda_2019['Dia'] = df_Demanda_2019['Fecha'].str.split('/').str[0]

# convertir a entero
df_Demanda_2019['Año'] = df_Demanda_2019['Año'].astype(int)
df_Demanda_2019['Dia'] = df_Demanda_2019['Dia'].astype(int)

# pasar de mes ene, feb, mar, etc a numero
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['ene'], 1)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['feb'], 2)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['mar'], 3)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['abr'], 4)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['may'], 5)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['jun'], 6)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['jul'], 7)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['ago'], 8)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['sep'], 9)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['oct'], 10)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['nov'], 11)
df_Demanda_2019['Mes'] = df_Demanda_2019['Mes'].replace(['dic'], 12)

#año en formato YYYY no YY
df_Demanda_2019['Año'] = df_Demanda_2019['Año'] + 2000

#pasar demanda de string a float
df_Demanda_2019['Demanda'] = df_Demanda_2019['Demanda'].str.replace(',', '.')
df_Demanda_2019['Demanda'] = df_Demanda_2019['Demanda'].astype(float)

#eliminar columna fecha
df_Demanda_2019 = df_Demanda_2019.drop(['Fecha'], axis = 1)

DatosDemanda = df_Demanda_2019
```

Unimos los dos datos por los datos comunes Año, Mes, Día:

```
#Unir los datos de los precios con los datos de la meteorologia
datos = pd.merge(datos, DatosMeteorologicos, on = ['Año', 'Mes', 'Dia'])

#poner columna precio como ultima columna
precio = datos['Precio']
datos = datos.drop(['Precio'], axis = 1)
datos['Precio'] = precio

#pasar a csv
datos.to_csv('datos_Media_Diario_Climatico.csv', sep = ';', decimal = '.')

#Leer datos
datos = pd.read_csv('datos_Media_Diario_Climatico.csv', sep = ';', decimal = '.')

#Unir los datros de los precios con los datos de la demanda
datos = pd.merge(datos, DatosDemanda, on = ['Año', 'Mes', 'Dia'])

#poner columna precio como ultima columna
precio = datos['Precio']
datos = datos.drop(['Precio'], axis = 1)
datos['Precio'] = precio

#pasar a csv
datos.to_csv('datos_Media_Diario_Climatico_Demanda.csv', sep = ';', decimal = '.')
datos = pd.read_csv('datos_Media_Diario_Climatico_Demanda.csv', sep = ';', decimal =
'.')

#Unir los datos de los precios con los datos de la estructura
datos = pd.merge(datos, DatosEstructura, on = ['Año', 'Mes', 'Dia'])

#poner columna precio como ultima columna
precio = datos['Precio']
datos = datos.drop(['Precio'], axis = 1)
datos['Precio'] = precio

#pasar a csv
datos.to_csv('datos_Media_Diario_Climatico_Demanda_Estructura.csv', sep = ';',
decimal = '.')
datos = pd.read_csv('datos_Media_Diario_Climatico_Demanda_Estructura.csv', sep = ';',
decimal = '.')

#ELIMINAR COLUMNAS QUE NO SE VAN A USAR, las tres primeras
datos = datos.drop(datos.columns[0], axis = 1)
datos = datos.drop(datos.columns[0], axis = 1)
datos = datos.drop(datos.columns[0], axis = 1)

#pasar a csv
datos.to_csv('datos_Media_Diario_Climatico_Demanda_Estructura.csv', sep = ';',
decimal = '.')

#Datos finales
datos = pd.read_csv('datos_Media_Diario_Climatico_Demanda_Estructura.csv', sep = ';',
decimal = '.')



print(datos)
```

```
##      Unnamed: 0   Año  Mes  ...  Residuos renovables  Generación total     Precio
## 0             0  2019    1  ...            2.661146        550.497196  63.454583
## 1             1  2019    1  ...            2.627507        715.146012  61.585417
## 2             2  2019    1  ...            2.605261        725.064340  63.961667
```

```
## 3             3  2019    1  ...             2.633564        764.578695  66.835833
## 4             4  2019    1  ...             2.678894        679.245801  65.328750
## ..          ...   ...  ...  ...                  ...               ...        ...
## 360         360  2019   12  ...             2.185158        673.647652  32.721667
## 361         361  2019   12  ...             2.301042        647.891005  31.090000
## 362         362  2019   12  ...             2.488827        626.099179  31.794583
## 363         363  2019   12  ...             2.467757        647.997859  36.726667
## 364         364  2019   12  ...             2.472558        638.019268  38.275833
##
## [365 rows x 28 columns]
```

Obtenida la tabla principal de estudio donde tenemos los datos:

-Año -Mes -Día -tmed: temperatura media -tmin: temperatura mínima -tmax: temperatura máxima -prec: precipitaciones -Demanda -… -Diferentes Tipos de generación energía (eólica, …) -… -Generación total -Precio: precio diario (la media de todos los precios horarios del día)

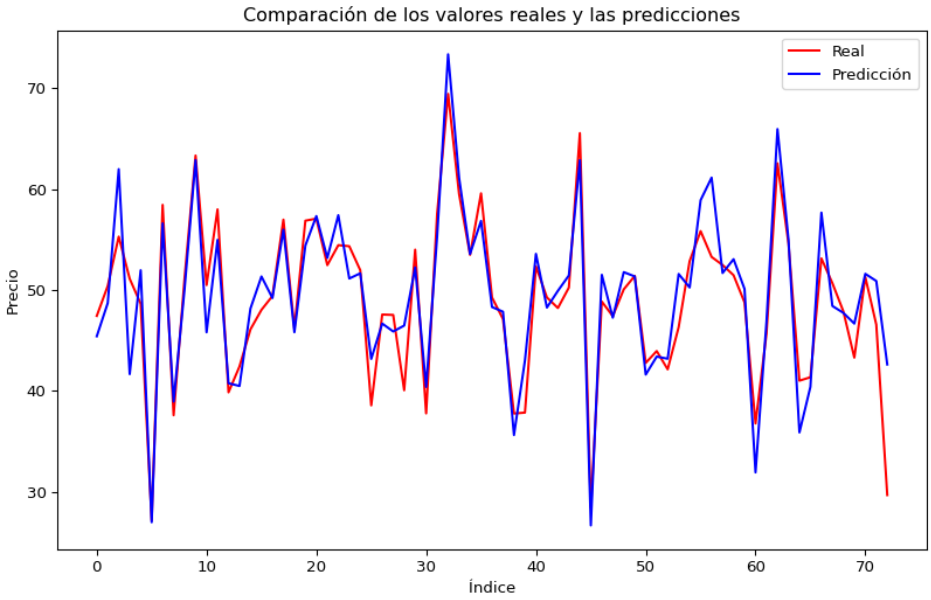
Podemos empezar el análisis de la regresión lineal:

```
#Modelo de regresion lineal
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import statsmodels.api as sm



#separar datos de entrenamiento y de prueba por el 80% y 20%
X = datos.iloc[:, :-1].values

#datos de la columna precio es la utlima columna
y = datos.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

#entrenar el modelo
regressor = LinearRegression()
regressor.fit(X_train, y_train)

#predecir los resultados

## LinearRegression()

y_pred = regressor.predict(X_test)

#comparar los resultados
df = pd.DataFrame({'Real': y_test, 'Prediccion': y_pred})
print(df)

##          Real  Prediccion
## 0   47.430417   45.413879
## 1   50.403750   48.706848
## 2   55.286667   61.972473
## 3   51.112500   41.667786
## 4   48.639167   51.956848
## ..        ...         ...
## 68  47.886667   47.734192
## 69  43.298333   46.671692
## 70  51.290000   51.601379
## 71  46.567917   50.890442
## 72  29.685833   42.620911
##
## [73 rows x 2 columns]
```

```
print('Resultado del modelo de regresion lineal')
#Error cuadratico medio

## Resultado del modelo de regresion lineal

from sklearn import metrics
print('Error cuadratico medio:', metrics.mean_squared_error(y_test, y_pred))

# Calcular el R^2

## Error cuadratico medio: 10.990799703756176

r2 = regressor.score(X_test, y_test)
# Imprimir el R^2
print('Coeficiente de determinación R^2:', r2)

#Imprimir coeficientes

## Coeficiente de determinación R^2: 0.8275342522220949

print('Coeficientes: ', regressor.coef_)
#Imprimir intercepto

## Coeficientes:  [-3.41062012e-01  2.07088079e+08  9.01785404e+00  3.36635359e-01
##   7.13867152e+00  7.48930331e-02 -3.82704712e+00 -3.49538336e+00
##   1.19435655e-02  4.44700542e+10  4.44700542e+10  4.44700542e+10
##   4.44700542e+10  4.44715402e+10  4.44700542e+10  4.44700542e+10
##   4.44700542e+10  4.44700542e+10  4.44700542e+10  4.44700542e+10
##   4.44700542e+10  4.44700542e+10  4.44700542e+10  4.44700542e+10
##   4.44700542e+10  4.44700542e+10 -4.44700542e+10]

print('Intercepto: ', regressor.intercept_)
#Imprimir ecuacion de la recta

## Intercepto:  -418110832287.86346

print('Ecuacion de la recta en forma matricial: y= ', regressor.coef_, 'x +',
regressor.intercept_)

# Añadir una constante a las variables independientes

## Ecuacion de la recta en forma matricial: y=  [-3.41062012e-01  2.07088079e+08
9.01785404e+00  3.36635359e-01
##   7.13867152e+00  7.48930331e-02 -3.82704712e+00 -3.49538336e+00
##   1.19435655e-02  4.44700542e+10  4.44700542e+10  4.44700542e+10
##   4.44700542e+10  4.44715402e+10  4.44700542e+10  4.44700542e+10
##   4.44700542e+10  4.44700542e+10  4.44700542e+10  4.44700542e+10
##   4.44700542e+10  4.44700542e+10  4.44700542e+10  4.44700542e+10
##   4.44700542e+10  4.44700542e+10 -4.44700542e+10] x + -418110832287.86346

X2 = sm.add_constant(X_train)

# Crear un nuevo modelo con statsmodels
model = sm.OLS(y_train, X2)

# Ajustar el modelo
results = model.fit()

# Imprimir el resumen del modelo
print(results.summary())

#Grafica de predicciones

##                             OLS Regression Results
## ==============================================================================
```

```
## Dep. Variable:                      y   R-squared:                       0.896
## Model:                            OLS   Adj. R-squared:                  0.886
## Method:                 Least Squares   F-statistic:                     91.56
## Date:                vi., 26 may. 2023   Prob (F-statistic):          1.99e-115
## Time:                        21:44:37   Log-Likelihood:                -744.46
## No. Observations:                 292   AIC:                             1541.
## Df Residuals:                     266   BIC:                             1637.
## Df Model:                          25
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## x1            -0.3411      0.487     -0.701      0.484      -1.299       0.617
## const          0.0050      0.009      0.555      0.579      -0.013       0.023
## x2             9.0191     14.845      0.608      0.544     -20.210      38.248
## x3             0.3365      0.484      0.695      0.487      -0.616       1.289
## x4             7.1628     12.755      0.562      0.575     -17.950      32.276
## x5             0.0771      0.079      0.981      0.328      -0.078       0.232
## x6            -3.8403      6.385     -0.601      0.548     -16.412       8.731
## x7            -3.5044      6.379     -0.549      0.583     -16.065       9.056
## x8             0.0118      0.011      1.062      0.289      -0.010       0.034
## x9         -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x10        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x11        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x12        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x13         1.404e+06    1.3e+06      1.084      0.279   -1.15e+06    3.95e+06
## x14        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x15        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x16        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x17        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x18         -8.26e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x19        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x20        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x21        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x22        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x23        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x24        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x25        -8.259e+04   7.62e+04     -1.084      0.279   -2.33e+05    6.74e+04
## x26         8.259e+04   7.62e+04      1.084      0.279   -6.74e+04    2.33e+05
## ==============================================================================
## Omnibus:                       20.089   Durbin-Watson:                   2.119
## Prob(Omnibus):                  0.000   Jarque-Bera (JB):               64.916
## Skew:                          -0.079   Prob(JB):                     8.01e-15
## Kurtosis:                       5.304   Cond. No.                     1.16e+16
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
## [2] The smallest eigenvalue is 1.14e-23. This might indicate that there are
## strong multicollinearity problems or that the design matrix is singular.

import matplotlib.pyplot as plt

# Crear una figura
plt.figure(figsize=(10,6))

# Crear un gráfico de línea con los datos reales
plt.plot(range(len(y_test)), y_test, color = 'red', label = 'Real')

# Crear un gráfico de línea con las predicciones
plt.plot(range(len(y_pred)), y_pred, color = 'blue', label = 'Predicción')

# Añadir títulos y etiquetas
plt.title('Comparación de los valores reales y las predicciones')
```

```
plt.xlabel('Índice')
plt.ylabel('Precio')
plt.legend()

# Mostrar la gráfica
plt.show()
```



Estudio de la regresión polinomial

```
#Modelo de regresion polinomial
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score


#separar datos de entrenamiento y de prueba por el 80% y 20%
X = datos.iloc[:, :-1].values

#datos de la columna precio es la utlima columna
y = datos.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

#entrenar el modelo
poly_reg = PolynomialFeatures(degree = 2)
X_poly = poly_reg.fit_transform(X_train)
regressor = LinearRegression()
regressor.fit(X_poly, y_train)

#predecir los resultados

## LinearRegression()

y_pred = regressor.predict(poly_reg.fit_transform(X_test))

#comparar los resultados
df = pd.DataFrame({'Real': y_test, 'Prediccion': y_pred})
print(df)
```

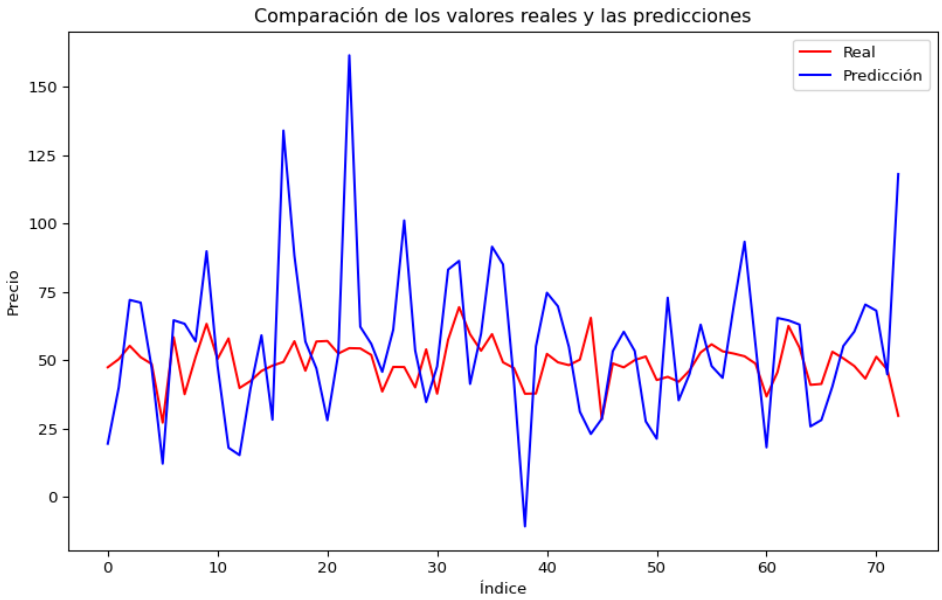
```
##          Real  Prediccion
## 0   47.430417   19.494815
## 1   50.403750   40.040921
## 2   55.286667   72.063824
## 3   51.112500   71.074858
## 4   48.639167   47.258414
## ..        ...         ...
## 68  47.886667   60.508590
## 69  43.298333   70.379488
## 70  51.290000   68.125634
## 71  46.567917   44.855378
## 72  29.685833  118.097974
##
## [73 rows x 2 columns]

print('Resultados de la regresion polinomial')
#Error cuadratico medio

## Resultados de la regresion polinomial

from sklearn import metrics
print('Error cuadratico medio:', metrics.mean_squared_error(y_test, y_pred))

# Calcular el R^2

## Error cuadratico medio: 750.3347439802386

r2 = r2_score(y_test, y_pred)
# Imprimir el R^2
print('Coeficiente de determinación R^2:', r2)

#Imprimir coeficientes

## Coeficiente de determinación R^2: -10.77412437605147

print('Coeficientes: ', regressor.coef_)
#Imprimir intercepto

## Coeficientes:  [-4.75212196e-02 -1.39453977e-02 -2.80315543e-02  1.01353246e-02
##   2.50802879e-02  6.04027467e-04 -1.82006786e-03  3.65335134e-03
##   4.31723706e-03 -2.57522206e-03 -6.89831677e-04  2.02338872e-03
##   2.21927746e-03 -4.91842380e-03 -4.70875334e-03 -8.46729288e-04
##   1.09412615e-03  2.49263598e-03 -4.31916632e-03 -1.67076037e-03
##   2.43235074e-03  3.02428174e-03  2.48111554e-03  5.51633462e-03
##  -8.64757765e-04  1.16950476e-03 -3.59634763e-04 -2.76020435e-04
##  -1.06374036e-01 -2.58986699e-02  3.25715667e+00  2.29392527e-01
##   1.86645759e+00 -1.99755794e-02 -1.76719732e+00 -1.34349364e+00
##  -6.08281282e-02 -1.99636018e-01 -9.20998401e-02 -1.86818635e-01
##  -3.59853979e-01 -6.17999669e-02 -3.90938116e-02 -4.90149805e-02
##  -1.49629900e-01  1.32051004e-01 -1.93095667e-01  4.51713604e-02
##   5.75081000e-01 -6.94296510e-01  9.11013141e-02  8.00488637e-01
##   3.78830264e-01 -6.29496861e-02  1.06234100e-01  4.39026317e-05
##   7.19475531e-01  3.69824833e-02 -6.46175342e-01  9.54574372e-03
##   2.95998948e-01  3.17196463e-01  9.47623369e-04 -6.49660781e-02
##  -5.72583334e-02 -5.72914209e-02 -6.24944200e-02 -5.96199461e-01
##  -1.60223798e-01 -1.19843476e-01 -3.48516466e-02 -5.48198489e-02
##   1.51502616e-01 -5.54886520e-02 -3.88054852e-02 -7.26567320e-02
##  -1.10749347e-01 -7.29167442e-02 -5.94060664e-02 -8.48317643e-02
##   5.92779932e-02  4.05258727e-01 -3.87654630e+00  1.23853090e+01
##   6.35771667e-01  1.75733794e+01  7.85274955e+00  1.73030338e+00
##   6.00872273e+00  2.24317651e+00  5.46295449e+00  1.08652187e+01
##  -2.13247760e-03  2.55133730e+00 -1.18212011e-01  6.04901085e+00
##  -4.17728778e+00  2.14141859e-01 -1.56834675e+00 -1.81575985e+01
##   2.10992755e+01 -1.26375054e-01 -2.47158488e+01 -7.02455271e+00
##  -1.57568815e+00 -2.98371144e+00 -1.36942610e-01  7.07155600e+00
##  -4.57345948e-02 -2.77962974e+00 -3.15853259e+00  6.72207206e-02
```

```
##   1.70502110e-01  6.41803611e-02  1.15082658e-01  2.95644553e-01
##  -6.79712368e-03 -4.11286089e-01  4.46579873e-01  2.14509833e-01
##  -1.65021252e-01  8.73536044e-01 -8.74574347e-02 -5.38878407e-01
##   5.70044806e-01 -3.67480307e-01 -1.09222117e+00 -3.39004808e-03
##  -1.65575404e-01 -6.29371202e-02 -1.87916686e+00 -1.79743971e+01
##  -2.63857422e+00  1.30889065e+00 -9.12690662e-01  6.61562464e-01
##   4.07779724e+00 -4.61644138e+00  2.39807784e+00 -6.24960118e-03
##   1.67305403e+01 -7.01939967e+00  1.56054905e+00 -5.31858577e+00
##  -1.23002455e+01 -3.34601657e+00  8.85300242e+00 -6.15017230e-01
##  -1.96007323e+00 -8.07532055e+00  1.07368173e+01  3.43646027e+00
##   5.19709886e+00  5.57087572e-02  8.77976321e+00  9.09716540e+00
##  -1.52395461e-02 -1.38894388e+00 -1.41901127e+00 -1.35625639e+00
##  -1.32264373e+00 -7.36928241e-04 -1.41175163e+00 -2.46837450e+00
##  -4.56850829e-01 -1.35096784e+00  2.35560228e+01 -1.35267373e+00
##  -1.51702558e+00 -1.31140811e+00 -1.27613996e+00 -1.51583249e+00
##  -1.23807519e+00 -2.80484597e+00  1.36435843e+00  3.48624833e+00
##  -1.02851473e-01  4.69287423e-01  1.81638687e+00 -3.86667915e-01
##   4.30528647e+00  8.75818981e-01 -4.91295979e-03 -9.17734283e+00
##   2.91941589e+00  4.73490823e-01  4.71980673e+00 -2.27301877e+01
##   3.74347668e+00 -1.42771536e+00  1.83820368e+00  5.23320607e-01
##   6.93100512e+00 -2.86850208e+00  3.78153265e+00 -4.66908151e+00
##   4.59499612e-02  4.89092476e-01 -1.48965350e-01 -1.54730373e+00
##   2.65436141e+00 -9.17770829e-01 -7.58936331e-03 -5.11237513e+00
##   3.98961255e+00 -4.95085138e-01  2.93362579e+00 -1.19521890e-01
##   1.92437578e+00 -5.04453839e+00  1.16894858e+00  4.25790507e+00
##   3.90755908e+00 -6.25082689e+00 -4.11019581e+00 -2.89975482e+00
##  -1.22488064e-03 -7.30195740e-02 -6.10531328e-02 -8.65703251e-02
##  -8.03707236e-02 -1.92416919e-01  9.31780711e-02 -2.79411076e-02
##   1.40897940e-02 -7.64594391e-02  1.14512184e+00 -7.41467285e-02
##  -1.84166325e-02 -1.49792238e-01 -1.52329379e-02 -1.27910470e-01
##  -6.37527390e-02 -8.60527638e-02  7.64375495e-02  3.06975206e-01
##  -1.11605118e+00  3.27109013e-01  6.69683726e-01 -1.09914897e-02
##   1.20004326e+00 -1.07577329e+00 -6.05765023e-01  3.43177439e-01
##  -3.20554330e+00  3.91481400e-01  2.61453085e-01  2.85233418e-01
##  -3.84414468e-01  2.73487789e-01  1.99248034e+00  7.44271950e-02
##  -2.39349077e-01 -1.34606910e+00 -1.50360718e+00 -1.07501082e+00
##  -4.60818584e-04  4.07758491e-01 -2.27053847e+00 -1.53719453e+00
##  -1.36512309e+00  1.81008468e+01 -1.31428878e+00 -1.54757187e+00
##  -1.50924217e+00 -2.32691994e+00 -1.62546569e+00  6.49015527e-01
##   8.45360972e-01  1.46980862e+00  4.95075687e-04  3.84643253e-01
##  -4.65906083e-02  1.12794144e+00 -1.07610765e+00 -7.91716633e-01
##   4.78232398e-02  5.61284675e-01  8.78974364e-02 -1.40763703e-01
##  -7.47673166e-03 -5.57207785e-01 -1.87310196e-01  1.92008270e+00
##  -6.89270399e-02  7.11882646e-02  3.64500894e-01 -4.99690729e-03
##   9.72284031e-01 -7.62831000e-01 -4.72796588e-01  3.95462805e-01
##  -4.64533315e+00  4.41464769e-01  2.49141118e-01  3.49191694e-01
##  -4.98058478e-02  1.73493302e-01  2.16684149e+00  5.71270817e-01
##  -2.85187568e-01  2.98000236e-10 -2.28928892e-03 -5.39646350e-04
##  -1.11336068e-03 -4.72699295e-02 -2.66885481e-05 -4.00148761e-02
##  -8.85450028e-03 -5.78996138e-03 -2.94580096e-03 -2.18884673e-02
##  -1.76219234e-03 -7.45438488e-04 -1.96280007e-01 -1.53802302e+00
##   3.60609270e+00 -6.10631901e-01  8.43909235e-01 -1.74158988e+01
##   9.59386110e-01 -1.16661533e-01  1.45546855e+00  3.56590300e+00
##   2.04579486e+00  1.19717248e+01 -9.41267821e+00 -9.33368353e-01
##  -3.17923222e+00 -1.68136988e+00 -1.20408381e+00  1.31155084e-01
##  -1.05275882e+00  4.42894265e-01 -1.06284306e+00  4.73344682e+00
##  -1.31915795e+00 -6.93383332e+00  1.39397360e+01  1.23246238e+00
##  -8.16625671e-01 -7.89459148e-01  2.37174782e+01 -7.63985126e-01
##  -1.60490044e+00 -4.38951750e-01 -3.25137952e+00 -1.63032293e+00
##   2.76180283e+00 -1.06020335e+01  8.68046569e-01  3.77662237e-02
##   1.05657575e+00  1.20898868e-01 -6.60451465e-02  2.57433767e-02
##  -6.64350825e-01 -1.07160709e-01  1.70038815e+00 -2.37169986e-01
##   3.64764103e-02  3.06315567e-01 -1.81474863e-01  1.47412264e+00
##   1.08642830e+00 -9.99526381e+00 -1.23073399e+00 -1.52811066e+01
##   4.75637421e+00 -7.64829366e-01  8.34792429e-02  5.27188343e-03
##   5.98343582e-02 -4.48566693e-01 -8.71397386e-02  1.82377609e+00
```

```
##  -1.68351226e-01 -1.33837334e-02  3.35480396e-01 -5.65832749e-01
##  -2.18853381e+00 -2.92102972e-01  2.26715286e+00  1.59859799e+00
##   1.05305230e-01  6.01289500e-02  1.23505967e-01 -1.12830333e-02
##   2.24196301e+00 -1.94928282e+00  1.45957463e-01 -2.34324487e+00
##   1.32313625e+00 -5.12695447e+00  1.80859992e+01  4.96437940e-01
##   2.89547964e-01  1.59328083e+00  1.12308843e+00  2.22164804e-01
##   1.05553693e-01 -5.55366491e+00 -1.69592682e+00 -1.27744613e+01
##   2.24298695e-01 -7.05922599e-02]

print('Intercepto: ', regressor.intercept_)
#Imprimir ecuacion de la recta

## Intercepto:  -44.35328270803841

print('Ecuacion de la recta en forma matricial: y= ', regressor.coef_, 'x +',
regressor.intercept_)

#Grafica de predicciones

## Ecuacion de la recta en forma matricial: y=  [-4.75212196e-02 -1.39453977e-02 -
2.80315543e-02  1.01353246e-02
##   2.50802879e-02  6.04027467e-04 -1.82006786e-03  3.65335134e-03
##   4.31723706e-03 -2.57522206e-03 -6.89831677e-04  2.02338872e-03
##   2.21927746e-03 -4.91842380e-03 -4.70875334e-03 -8.46729288e-04
##   1.09412615e-03  2.49263598e-03 -4.31916632e-03 -1.67076037e-03
##   2.43235074e-03  3.02428174e-03  2.48111554e-03  5.51633462e-03
##  -8.64757765e-04  1.16950476e-03 -3.59634763e-04 -2.76020435e-04
##  -1.06374036e-01 -2.58986699e-02  3.25715667e+00  2.29392527e-01
##   1.86645759e+00 -1.99755794e-02 -1.76719732e+00 -1.34349364e+00
##  -6.08281282e-02 -1.99636018e-01 -9.20998401e-02 -1.86818635e-01
##  -3.59853979e-01 -6.17999669e-02 -3.90938116e-02 -4.90149805e-02
##  -1.49629900e-01  1.32051004e-01 -1.93095667e-01  4.51713604e-02
##   5.75081000e-01 -6.94296510e-01  9.11013141e-02  8.00488637e-01
##   3.78830264e-01 -6.29496861e-02  1.06234100e-01  4.39026317e-05
##   7.19475531e-01  3.69824833e-02 -6.46175342e-01  9.54574372e-03
##   2.95998948e-01  3.17196463e-01  9.47623369e-04 -6.49660781e-02
##  -5.72583334e-02 -5.72914209e-02 -6.24944200e-02 -5.96199461e-01
##  -1.60223798e-01 -1.19843476e-01 -3.48516466e-02 -5.48198489e-02
##   1.51502616e-01 -5.54886520e-02 -3.88054852e-02 -7.26567320e-02
##  -1.10749347e-01 -7.29167442e-02 -5.94060664e-02 -8.48317643e-02
##   5.92779932e-02  4.05258727e-01 -3.87654630e+00  1.23853090e+01
##   6.35771667e-01  1.75733794e+01  7.85274955e+00  1.73030338e+00
##   6.00872273e+00  2.24317651e+00  5.46295449e+00  1.08652187e+01
##  -2.13247760e-03  2.55133730e+00 -1.18212011e-01  6.04901085e+00
##  -4.17728778e+00  2.14141859e-01 -1.56834675e+00 -1.81575985e+01
##   2.10992755e+01 -1.26375054e-01 -2.47158488e+01 -7.02455271e+00
##  -1.57568815e+00 -2.98371144e+00 -1.36942610e-01  7.07155600e+00
##  -4.57345948e-02 -2.77962974e+00 -3.15853259e+00  6.72207206e-02
##   1.70502110e-01  6.41803611e-02  1.15082658e-01  2.95644553e-01
##  -6.79712368e-03 -4.11286089e-01  4.46579873e-01  2.14509833e-01
##  -1.65021252e-01  8.73536044e-01 -8.74574347e-02 -5.38878407e-01
##   5.70044806e-01 -3.67480307e-01 -1.09222117e+00 -3.39004808e-03
##  -1.65575404e-01 -6.29371202e-02 -1.87916686e+00 -1.79743971e+01
##  -2.63857422e+00  1.30889065e+00 -9.12690662e-01  6.61562464e-01
##   4.07779724e+00 -4.61644138e+00  2.39807784e+00 -6.24960118e-03
##   1.67305403e+01 -7.01939967e+00  1.56054905e+00 -5.31858577e+00
##  -1.23002455e+01 -3.34601657e+00  8.85300242e+00 -6.15017230e-01
##  -1.96007323e+00 -8.07532055e+00  1.07368173e+01  3.43646027e+00
##   5.19709886e+00  5.57087572e-02  8.77976321e+00  9.09716540e+00
##  -1.52395461e-02 -1.38894388e+00 -1.41901127e+00 -1.35625639e+00
##  -1.32264373e+00 -7.36928241e-04 -1.41175163e+00 -2.46837450e+00
##  -4.56850829e-01 -1.35096784e+00  2.35560228e+01 -1.35267373e+00
##  -1.51702558e+00 -1.31140811e+00 -1.27613996e+00 -1.51583249e+00
##  -1.23807519e+00 -2.80484597e+00  1.36435843e+00  3.48624833e+00
##  -1.02851473e-01  4.69287423e-01  1.81638687e+00 -3.86667915e-01
##   4.30528647e+00  8.75818981e-01 -4.91295979e-03 -9.17734283e+00
```

```
##   2.91941589e+00  4.73490823e-01  4.71980673e+00 -2.27301877e+01
##   3.74347668e+00 -1.42771536e+00  1.83820368e+00  5.23320607e-01
##   6.93100512e+00 -2.86850208e+00  3.78153265e+00 -4.66908151e+00
##   4.59499612e-02  4.89092476e-01 -1.48965350e-01 -1.54730373e+00
##   2.65436141e+00 -9.17770829e-01 -7.58936331e-03 -5.11237513e+00
##   3.98961255e+00 -4.95085138e-01  2.93362579e+00 -1.19521890e-01
##   1.92437578e+00 -5.04453839e+00  1.16894858e+00  4.25790507e+00
##   3.90755908e+00 -6.25082689e+00 -4.11019581e+00 -2.89975482e+00
##  -1.22488064e-03 -7.30195740e-02 -6.10531328e-02 -8.65703251e-02
##  -8.03707236e-02 -1.92416919e-01  9.31780711e-02 -2.79411076e-02
##   1.40897940e-02 -7.64594391e-02  1.14512184e+00 -7.41467285e-02
##  -1.84166325e-02 -1.49792238e-01 -1.52329379e-02 -1.27910470e-01
##  -6.37527390e-02 -8.60527638e-02  7.64375495e-02  3.06975206e-01
##  -1.11605118e+00  3.27109013e-01  6.69683726e-01 -1.09914897e-02
##   1.20004326e+00 -1.07577329e+00 -6.05765023e-01  3.43177439e-01
##  -3.20554330e+00  3.91481400e-01  2.61453085e-01  2.85233418e-01
##  -3.84414468e-01  2.73487789e-01  1.99248034e+00  7.44271950e-02
##  -2.39349077e-01 -1.34606910e+00 -1.50360718e+00 -1.07501082e+00
##  -4.60818584e-04  4.07758491e-01 -2.27053847e+00 -1.53719453e+00
##  -1.36512309e+00  1.81008468e+01 -1.31428878e+00 -1.54757187e+00
##  -1.50924217e+00 -2.32691994e+00 -1.62546569e+00  6.49015527e-01
##   8.45360972e-01  1.46980862e+00  4.95075687e-04  3.84643253e-01
##  -4.65906083e-02  1.12794144e+00 -1.07610765e+00 -7.91716633e-01
##   4.78232398e-02  5.61284675e-01  8.78974364e-02 -1.40763703e-01
##  -7.47673166e-03 -5.57207785e-01 -1.87310196e-01  1.92008270e+00
##  -6.89270399e-02  7.11882646e-02  3.64500894e-01 -4.99690729e-03
##   9.72284031e-01 -7.62831000e-01 -4.72796588e-01  3.95462805e-01
##  -4.64533315e+00  4.41464769e-01  2.49141118e-01  3.49191694e-01
##  -4.98058478e-02  1.73493302e-01  2.16684149e+00  5.71270817e-01
##  -2.85187568e-01  2.98000236e-10 -2.28928892e-03 -5.39646350e-04
##  -1.11336068e-03 -4.72699295e-02 -2.66885481e-05 -4.00148761e-02
##  -8.85450028e-03 -5.78996138e-03 -2.94580096e-03 -2.18884673e-02
##  -1.76219234e-03 -7.45438488e-04 -1.96280007e-01 -1.53802302e+00
##   3.60609270e+00 -6.10631901e-01  8.43909235e-01 -1.74158988e+01
##   9.59386110e-01 -1.16661533e-01  1.45546855e+00  3.56590300e+00
##   2.04579486e+00  1.19717248e+01 -9.41267821e+00 -9.33368353e-01
##  -3.17923222e+00 -1.68136988e+00 -1.20408381e+00  1.31155084e-01
##  -1.05275882e+00  4.42894265e-01 -1.06284306e+00  4.73344682e+00
##  -1.31915795e+00 -6.93383332e+00  1.39397360e+01  1.23246238e+00
##  -8.16625671e-01 -7.89459148e-01  2.37174782e+01 -7.63985126e-01
##  -1.60490044e+00 -4.38951750e-01 -3.25137952e+00 -1.63032293e+00
##   2.76180283e+00 -1.06020335e+01  8.68046569e-01  3.77662237e-02
##   1.05657575e+00  1.20898868e-01 -6.60451465e-02  2.57433767e-02
##  -6.64350825e-01 -1.07160709e-01  1.70038815e+00 -2.37169986e-01
##   3.64764103e-02  3.06315567e-01 -1.81474863e-01  1.47412264e+00
##   1.08642830e+00 -9.99526381e+00 -1.23073399e+00 -1.52811066e+01
##   4.75637421e+00 -7.64829366e-01  8.34792429e-02  5.27188343e-03
##   5.98343582e-02 -4.48566693e-01 -8.71397386e-02  1.82377609e+00
##  -1.68351226e-01 -1.33837334e-02  3.35480396e-01 -5.65832749e-01
##  -2.18853381e+00 -2.92102972e-01  2.26715286e+00  1.59859799e+00
##   1.05305230e-01  6.01289500e-02  1.23505967e-01 -1.12830333e-02
##   2.24196301e+00 -1.94928282e+00  1.45957463e-01 -2.34324487e+00
##   1.32313625e+00 -5.12695447e+00  1.80859992e+01  4.96437940e-01
##   2.89547964e-01  1.59328083e+00  1.12308843e+00  2.22164804e-01
##   1.05553693e-01 -5.55366491e+00 -1.69592682e+00 -1.27744613e+01
##   2.24298695e-01 -7.05922599e-02] x + -44.35328270803841

import matplotlib.pyplot as plt

# Crear una figura
plt.figure(figsize=(10,6))

# Crear un gráfico de línea con los datos reales
plt.plot(range(len(y_test)), y_test, color = 'red', label = 'Real')
```

```
# Crear un gráfico de línea con las predicciones
plt.plot(range(len(y_pred)), y_pred, color = 'blue', label = 'Predicción')

# Añadir títulos y etiquetas
plt.title('Comparación de los valores reales y las predicciones')
plt.xlabel('Índice')
plt.ylabel('Precio')
plt.legend()

# Mostrar la gráfica
plt.show()
```

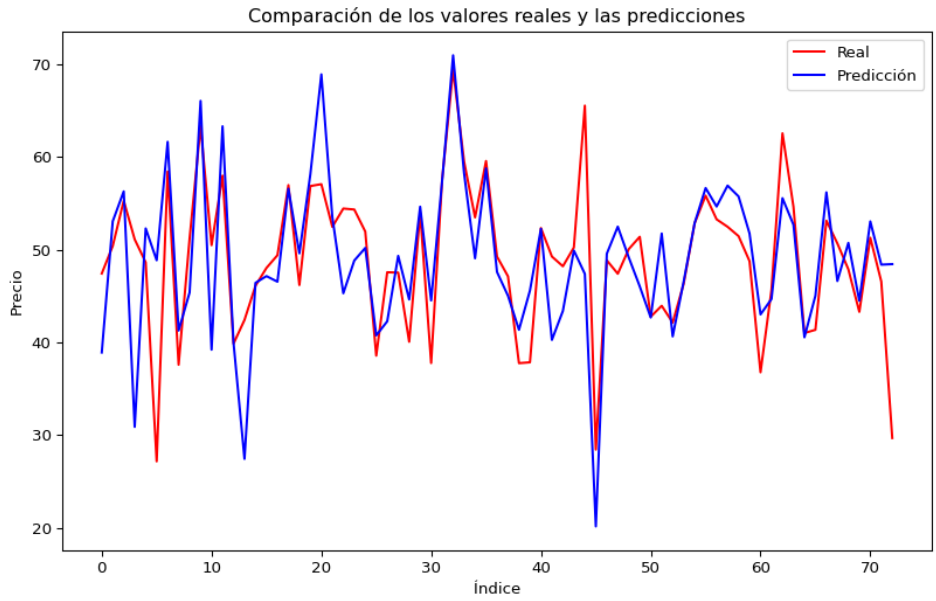


Regresión Polinomial de grado 3:

```
#Modelo de regresion polinomial de grado 3

print('Modelo de regresion polinomial')

## Modelo de regresion polinomial

from sklearn.preprocessing import PolynomialFeatures

#separar datos de entrenamiento y de prueba por el 80% y 20%
X = datos.iloc[:, :-1].values

#datos de la columna precio es la utlima columna
y = datos.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

#entrenar el modelo
poly_reg = PolynomialFeatures(degree = 3)
X_poly = poly_reg.fit_transform(X_train)
regressor = LinearRegression()
regressor.fit(X_poly, y_train)

#predecir los resultados
```

```
## LinearRegression()

y_pred = regressor.predict(poly_reg.fit_transform(X_test))

#comparar los resultados
df = pd.DataFrame({'Real': y_test, 'Prediccion': y_pred})
print(df)

##          Real  Prediccion
## 0   47.430417   38.913789
## 1   50.403750   53.067506
## 2   55.286667   56.294935
## 3   51.112500   30.891625
## 4   48.639167   52.297350
## ..        ...         ...
## 68  47.886667   50.736727
## 69  43.298333   44.491880
## 70  51.290000   53.055803
## 71  46.567917   48.388220
## 72  29.685833   48.445200
##
## [73 rows x 2 columns]

print('Resultados de la regresion polinomial')
#Error cuadratico medio

## Resultados de la regresion polinomial

from sklearn import metrics
print('Error cuadratico medio:', metrics.mean_squared_error(y_test, y_pred))

#R2

## Error cuadratico medio: 42.149858966013944

print('R2:', metrics.r2_score(y_test, y_pred))

#Imprimir coeficientes

## R2: 0.3385916274297611

print('Coeficientes: ', regressor.coef_)
#Imprimir intercepto

## Coeficientes:  [ 1.93477650e-07  4.07712299e-07  1.74603204e-07 ... -4.69288828e-
09
##   2.49730428e-06 -8.50257841e-06]

print('Intercepto: ', regressor.intercept_)
#Imprimir ecuacion de la recta

## Intercepto:  -1773.2894635938505

print('Ecuacion de la recta en forma matricial: y= ', regressor.coef_, 'x +',
regressor.intercept_)

#Grafica de predicciones

## Ecuacion de la recta en forma matricial: y=  [ 1.93477650e-07  4.07712299e-07
1.74603204e-07 ... -4.69288828e-09
##   2.49730428e-06 -8.50257841e-06] x + -1773.2894635938505

import matplotlib.pyplot as plt

# Crear una figura
plt.figure(figsize=(10,6))
```

```python
# Crear un gráfico de línea con los datos reales
plt.plot(range(len(y_test)), y_test, color = 'red', label = 'Real')

# Crear un gráfico de línea con las predicciones
plt.plot(range(len(y_pred)), y_pred, color = 'blue', label = 'Predicción')

# Añadir títulos y etiquetas
plt.title('Comparación de los valores reales y las predicciones')
plt.xlabel('Índice')
plt.ylabel('Precio')
plt.legend()

# Mostrar la gráfica
plt.show()
```



Ejemplo de regresión lineal en R:

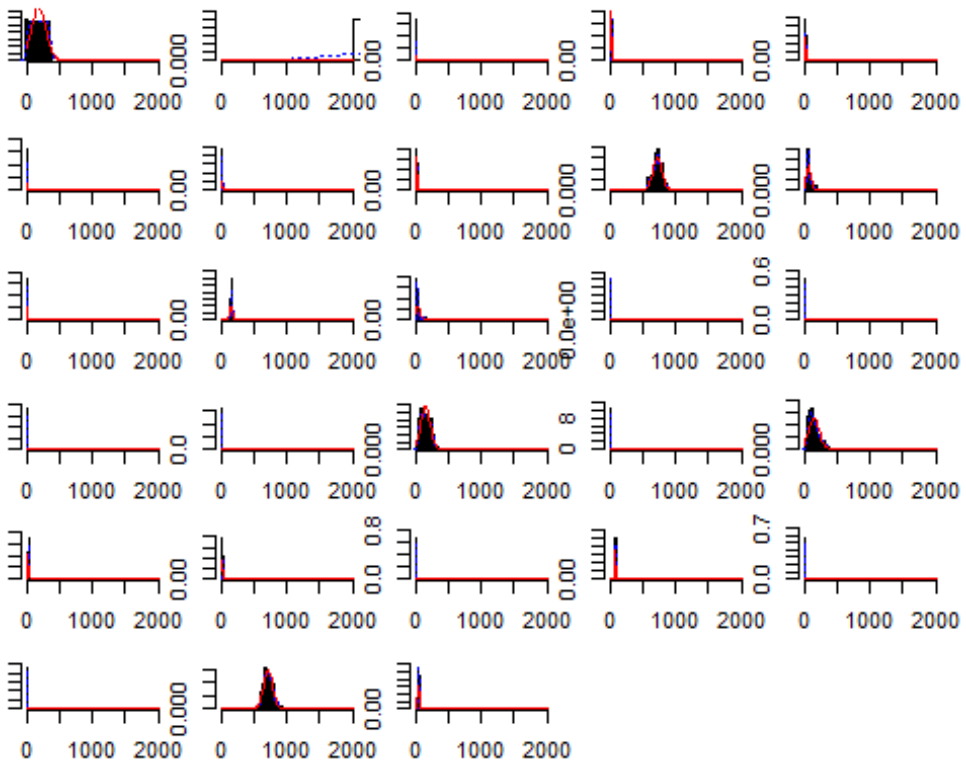
```r
datos_r <- read.csv('datos_Media_Diario_Climatico_Demanda_Estructura.csv', sep=';')

names(datos_r)
```

```
##  [1] "X"                     "Año"                   "Mes"
##  [4] "Dia"                   "tmed"                  "prec"
##  [7] "tmin"                  "tmax"                  "Demanda"
## [10] "Hidráulica"            "Turbinación.bombeo"    "Nuclear"
## [13] "Carbón"                "Fuel...Gas"            "Motores.diésel"
## [16] "Turbina.de.gas"        "Turbina.de.vapor"      "Ciclo.combinado"
## [19] "Hidroeólica"           "Eólica"                "Solar.fotovoltaica"
## [22] "Solar.térmica"         "Otras.renovables"      "Cogeneración"
## [25] "Residuos.no.renovables" "Residuos.renovables"  "Generación.total"
## [28] "Precio"
```

```r
#pairs(datos_r)

#cor(x = datos_r, method = "pearson")

library(psych)
```

```
## Warning: package 'psych' was built under R version 4.2.3
```

```
multi.hist(x = datos_r, dcol = c("blue", "red"), dlty = c("dotted", "solid"),
           main = "")
```



```
modelo <- lm(Precio ~ Año + Mes + Dia + tmed +
               prec + tmin + tmax + Demanda + Hidráulica + Turbinación.bombeo +
Nuclear + Carbón + Fuel...Gas
               + Motores.diésel + Turbina.de.gas + Turbina.de.vapor + Ciclo.combinado
+ Hidroeólica + Eólica + Solar.fotovoltaica
                + Solar.térmica + Otras.renovables + Cogeneración +
Residuos.no.renovables + Residuos.renovables + Generación.total
               , data = datos_r )
summary(modelo)

##
## Call:
## lm(formula = Precio ~ Año + Mes + Dia + tmed + prec + tmin +
##     tmax + Demanda + Hidráulica + Turbinación.bombeo + Nuclear +
##     Carbón + Fuel...Gas + Motores.diésel + Turbina.de.gas +
##     Turbina.de.vapor + Ciclo.combinado + Hidroeólica + Eólica +
##     Solar.fotovoltaica + Solar.térmica + Otras.renovables +
##     Cogeneración + Residuos.no.renovables + Residuos.renovables +
##     Generación.total, data = datos_r)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.408  -1.842   0.157   1.850  13.403
##
## Coefficients: (2 not defined because of singularities)
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             2.206e+01  5.878e+00   3.753 0.000206 ***
## Año                            NA         NA      NA       NA
## Mes                    -1.471e+00  1.201e-01 -12.247  < 2e-16 ***
## Dia                    -1.303e-02  2.069e-02  -0.630 0.529338
## tmed                    1.218e+01  1.117e+01   1.090 0.276415
## prec                    6.701e-02  6.762e-02   0.991 0.322443
## tmin                   -6.278e+00  5.588e+00  -1.123 0.262087
## tmax                   -6.109e+00  5.596e+00  -1.092 0.275705
## Demanda                 2.833e-03  9.470e-03   0.299 0.765028
```

```
## Hidráulica             -1.176e-02  1.333e-02  -0.882 0.378305
## Turbinación.bombeo     -2.800e-01  5.609e-02  -4.993 9.54e-07 ***
## Nuclear                -3.267e-02  1.576e-02  -2.073 0.038883 *
## Carbón                  4.423e-02  1.239e-02   3.570 0.000409 ***
## Fuel...Gas              1.093e+06  1.243e+06   0.879 0.380153
## Motores.diésel         -8.890e-01  3.374e-01  -2.634 0.008812 **
## Turbina.de.gas          1.838e-01  4.342e-01   0.423 0.672290
## Turbina.de.vapor       -2.766e-01  2.230e-01  -1.240 0.215682
## Ciclo.combinado         5.684e-02  9.771e-03   5.817 1.38e-08 ***
## Hidroeólica            -1.292e+01  7.170e+00  -1.803 0.072345 .
## Eólica                 -2.569e-02  6.977e-03  -3.682 0.000269 ***
## Solar.fotovoltaica     -5.121e-03  7.025e-02  -0.073 0.941935
## Solar.térmica          -9.129e-02  5.134e-02  -1.778 0.076284 .
## Otras.renovables        2.301e-01  2.686e-01   0.857 0.392273
## Cogeneración            5.789e-01  7.149e-02   8.097 1.01e-14 ***
## Residuos.no.renovables -6.311e-01  3.547e-01  -1.779 0.076114 .
## Residuos.renovables     7.146e-01  6.702e-01   1.066 0.287112
## Generación.total               NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.213 on 340 degrees of freedom
## Multiple R-squared:  0.8892, Adjusted R-squared:  0.8814
## F-statistic: 113.7 on 24 and 340 DF,  p-value: < 2.2e-16
```

```
step(object = modelo, direction = "both", trace = 1)
```

```
## Start:  AIC=876.19
## Precio ~ Año + Mes + Dia + tmed + prec + tmin + tmax + Demanda +
##     Hidráulica + Turbinación.bombeo + Nuclear + Carbón + Fuel...Gas +
##     Motores.diésel + Turbina.de.gas + Turbina.de.vapor + Ciclo.combinado +
##     Hidroeólica + Eólica + Solar.fotovoltaica + Solar.térmica +
##     Otras.renovables + Cogeneración + Residuos.no.renovables +
##     Residuos.renovables + Generación.total
##
##
## Step:  AIC=876.19
## Precio ~ Año + Mes + Dia + tmed + prec + tmin + tmax + Demanda +
##     Hidráulica + Turbinación.bombeo + Nuclear + Carbón + Fuel...Gas +
##     Motores.diésel + Turbina.de.gas + Turbina.de.vapor + Ciclo.combinado +
##     Hidroeólica + Eólica + Solar.fotovoltaica + Solar.térmica +
##     Otras.renovables + Cogeneración + Residuos.no.renovables +
##     Residuos.renovables
##
##
## Step:  AIC=876.19
## Precio ~ Mes + Dia + tmed + prec + tmin + tmax + Demanda + Hidráulica +
##     Turbinación.bombeo + Nuclear + Carbón + Fuel...Gas + Motores.diésel +
##     Turbina.de.gas + Turbina.de.vapor + Ciclo.combinado + Hidroeólica +
##     Eólica + Solar.fotovoltaica + Solar.térmica + Otras.renovables +
##     Cogeneración + Residuos.no.renovables + Residuos.renovables
##
##                         Df Sum of Sq    RSS     AIC
## - Solar.fotovoltaica     1      0.05 3510.3  874.19
## - Demanda                1      0.92 3511.1  874.29
## - Turbina.de.gas         1      1.85 3512.1  874.38
## - Dia                    1      4.09 3514.3  874.61
## - Otras.renovables       1      7.58 3517.8  874.98
## - Fuel...Gas             1      7.97 3518.2  875.02
## - Hidráulica             1      8.03 3518.2  875.02
## - prec                   1     10.14 3520.3  875.24
## - Residuos.renovables    1     11.74 3521.9  875.41
## - tmed                   1     12.27 3522.5  875.46
## - tmax                   1     12.31 3522.5  875.47
## - tmin                   1     13.03 3523.2  875.54
## - Turbina.de.vapor       1     15.88 3526.1  875.84
```

```
## <none>                                3510.2  876.19
## - Solar.térmica           1     32.64 3542.8  877.57
## - Residuos.no.renovables  1     32.68 3542.9  877.57
## - Hidroeólica             1     33.54 3543.7  877.66
## - Nuclear                 1     44.39 3554.6  878.78
## - Motores.diésel          1     71.65 3581.9  881.57
## - Carbón                  1    131.56 3641.8  887.62
## - Eólica                  1    139.99 3650.2  888.46
## - Turbinación.bombeo      1    257.34 3767.5  900.01
## - Ciclo.combinado         1    349.40 3859.6  908.82
## - Cogeneración            1    676.94 4187.1  938.55
## - Mes                     1   1548.60 5058.8 1007.58
##
## Step:  AIC=874.19
## Precio ~ Mes + Dia + tmed + prec + tmin + tmax + Demanda + Hidráulica +
##     Turbinación.bombeo + Nuclear + Carbón + Fuel...Gas + Motores.diésel +
##     Turbina.de.gas + Turbina.de.vapor + Ciclo.combinado + Hidroeólica +
##     Eólica + Solar.térmica + Otras.renovables + Cogeneración +
##     Residuos.no.renovables + Residuos.renovables
##
##                          Df Sum of Sq    RSS     AIC
## - Demanda                 1      0.88 3511.1  872.29
## - Turbina.de.gas          1      1.81 3512.1  872.38
## - Dia                     1      4.13 3514.4  872.62
## - Otras.renovables        1      7.56 3517.8  872.98
## - Fuel...Gas              1      8.05 3518.3  873.03
## - Hidráulica              1      8.47 3518.7  873.07
## - prec                    1     11.30 3521.6  873.37
## - Residuos.renovables     1     11.71 3522.0  873.41
## - tmed                    1     12.22 3522.5  873.46
## - tmax                    1     12.26 3522.5  873.47
## - tmin                    1     12.97 3523.2  873.54
## - Turbina.de.vapor        1     15.83 3526.1  873.84
## <none>                                3510.3  874.19
## - Residuos.no.renovables  1     32.62 3542.9  875.57
## - Hidroeólica             1     33.57 3543.8  875.67
## + Solar.fotovoltaica      1      0.05 3510.2  876.19
## + Generación.total        1      0.05 3510.2  876.19
## - Nuclear                 1     44.53 3554.8  876.80
## - Motores.diésel          1     71.63 3581.9  879.57
## - Solar.térmica           1     74.64 3584.9  879.87
## - Carbón                  1    132.74 3643.0  885.74
## - Eólica                  1    142.06 3652.3  886.68
## - Turbinación.bombeo      1    262.77 3773.0  898.54
## - Ciclo.combinado         1    353.88 3864.1  907.25
## - Cogeneración            1    687.41 4197.7  937.47
## - Mes                     1   1701.73 5212.0 1016.47
##
## Step:  AIC=872.29
## Precio ~ Mes + Dia + tmed + prec + tmin + tmax + Hidráulica +
##     Turbinación.bombeo + Nuclear + Carbón + Fuel...Gas + Motores.diésel +
##     Turbina.de.gas + Turbina.de.vapor + Ciclo.combinado + Hidroeólica +
##     Eólica + Solar.térmica + Otras.renovables + Cogeneración +
##     Residuos.no.renovables + Residuos.renovables
##
##                          Df Sum of Sq    RSS     AIC
## - Turbina.de.gas          1      3.18 3514.3  870.62
## - Dia                     1      5.03 3516.2  870.81
## - Otras.renovables        1      7.95 3519.1  871.11
## - Fuel...Gas              1      8.50 3519.6  871.17
## - Hidráulica              1     11.26 3522.4  871.46
## - prec                    1     11.40 3522.5  871.47
## - Residuos.renovables     1     11.74 3522.9  871.51
## - tmed                    1     12.11 3523.3  871.54
## - tmax                    1     12.14 3523.3  871.55
## - tmin                    1     12.87 3524.0  871.62
```

```
## - Turbina.de.vapor        1     15.54 3526.7  871.90
## <none>                              3511.1  872.29
## - Hidroeólica             1     33.11 3544.3  873.71
## - Residuos.no.renovables  1     33.30 3544.4  873.73
## + Demanda                 1      0.88 3510.3  874.19
## + Generación.total        1      0.01 3511.1  874.29
## + Solar.fotovoltaica      1      0.01 3511.1  874.29
## - Nuclear                 1     47.80 3558.9  875.22
## - Motores.diésel          1     70.80 3581.9  877.57
## - Solar.térmica           1     73.76 3584.9  877.88
## - Carbón                  1    210.96 3722.1  891.58
## - Turbinación.bombeo      1    262.97 3774.1  896.65
## - Eólica                  1    303.40 3814.5  900.54
## - Ciclo.combinado         1    725.80 4236.9  938.87
## - Cogeneración            1    999.49 4510.6  961.72
## - Mes                     1   1763.77 5274.9 1018.85
##
## Step:  AIC=870.62
## Precio ~ Mes + Dia + tmed + prec + tmin + tmax + Hidráulica +
##     Turbinación.bombeo + Nuclear + Carbón + Fuel...Gas + Motores.diésel +
##     Turbina.de.vapor + Ciclo.combinado + Hidroeólica + Eólica +
##     Solar.térmica + Otras.renovables + Cogeneración + Residuos.no.renovables +
##     Residuos.renovables
##
##                          Df Sum of Sq    RSS     AIC
## - Dia                     1      4.46 3518.8  869.08
## - Otras.renovables        1      8.15 3522.5  869.46
## - Fuel...Gas              1      8.35 3522.7  869.48
## - Hidráulica              1      9.18 3523.5  869.57
## - Residuos.renovables     1     11.55 3525.9  869.81
## - tmed                    1     11.61 3525.9  869.82
## - tmax                    1     11.62 3525.9  869.82
## - prec                    1     11.82 3526.1  869.84
## - tmin                    1     12.35 3526.7  869.90
## - Turbina.de.vapor        1     14.91 3529.2  870.16
## <none>                              3514.3  870.62
## - Hidroeólica             1     30.62 3544.9  871.78
## - Residuos.no.renovables  1     33.42 3547.7  872.07
## + Turbina.de.gas          1      3.18 3511.1  872.29
## + Demanda                 1      2.25 3512.1  872.38
## + Generación.total        1      0.15 3514.2  872.60
## + Solar.fotovoltaica      1      0.01 3514.3  872.62
## - Nuclear                 1     44.78 3559.1  873.24
## - Motores.diésel          1     67.65 3582.0  875.58
## - Solar.térmica           1     72.25 3586.6  876.05
## - Carbón                  1    207.78 3722.1  889.58
## - Turbinación.bombeo      1    264.13 3778.4  895.07
## - Eólica                  1    317.10 3831.4  900.15
## - Ciclo.combinado         1    864.53 4378.8  948.89
## - Cogeneración            1   1036.10 4550.4  962.92
## - Mes                     1   1793.34 5307.7 1019.11
##
## Step:  AIC=869.08
## Precio ~ Mes + tmed + prec + tmin + tmax + Hidráulica + Turbinación.bombeo +
##     Nuclear + Carbón + Fuel...Gas + Motores.diésel + Turbina.de.vapor +
##     Ciclo.combinado + Hidroeólica + Eólica + Solar.térmica +
##     Otras.renovables + Cogeneración + Residuos.no.renovables +
##     Residuos.renovables
##
##                          Df Sum of Sq    RSS     AIC
## - Fuel...Gas              1      7.51 3526.3  867.86
## - Otras.renovables        1      8.97 3527.7  868.01
## - Residuos.renovables     1     11.16 3529.9  868.24
## - prec                    1     11.52 3530.3  868.27
## - Hidráulica              1     12.12 3530.9  868.33
## - tmed                    1     12.58 3531.4  868.38
```

```
## - tmax                    1     12.59 3531.4  868.38
## - tmin                    1     13.36 3532.1  868.46
## - Turbina.de.vapor        1     15.21 3534.0  868.65
## <none>                                3518.8  869.08
## - Hidroeólica             1     28.00 3546.8  869.97
## + Dia                     1      4.46 3514.3  870.62
## + Demanda                 1      3.21 3515.6  870.75
## + Turbina.de.gas          1      2.60 3516.2  870.81
## - Residuos.no.renovables  1     37.62 3556.4  870.96
## + Generación.total        1      0.10 3518.7  871.07
## + Solar.fotovoltaica      1      0.00 3518.8  871.08
## - Nuclear                 1     44.87 3563.6  871.70
## - Motores.diésel          1     65.10 3583.9  873.77
## - Solar.térmica           1     72.43 3591.2  874.52
## - Carbón                  1    208.12 3726.9  888.05
## - Turbinación.bombeo      1    261.77 3780.5  893.27
## - Eólica                  1    319.42 3838.2  898.79
## - Ciclo.combinado         1    860.10 4378.9  946.90
## - Cogeneración            1   1047.00 4565.8  962.15
## - Mes                     1   1789.37 5308.1 1017.14
##
## Step:  AIC=867.86
## Precio ~ Mes + tmed + prec + tmin + tmax + Hidráulica + Turbinación.bombeo +
##     Nuclear + Carbón + Motores.diésel + Turbina.de.vapor +
##     Ciclo.combinado + Hidroeólica + Eólica + Solar.térmica +
##     Otras.renovables + Cogeneración + Residuos.no.renovables +
##     Residuos.renovables
##
##                          Df Sum of Sq    RSS     AIC
## - Otras.renovables        1      8.13 3534.4  866.70
## - prec                    1     11.31 3537.6  867.03
## - Hidráulica              1     11.76 3538.0  867.07
## - tmax                    1     12.50 3538.8  867.15
## - tmed                    1     12.50 3538.8  867.15
## - Residuos.renovables     1     13.28 3539.6  867.23
## - tmin                    1     13.31 3539.6  867.23
## - Turbina.de.vapor        1     14.63 3540.9  867.37
## <none>                                3526.3  867.86
## - Hidroeólica             1     27.19 3553.5  868.66
## + Fuel...Gas              1      7.51 3518.8  869.08
## + Demanda                 1      3.75 3522.5  869.47
## + Dia                     1      3.62 3522.7  869.48
## + Turbina.de.gas          1      2.52 3523.8  869.60
## - Residuos.no.renovables  1     37.68 3564.0  869.74
## + Generación.total        1      0.05 3526.2  869.85
## + Solar.fotovoltaica      1      0.00 3526.3  869.86
## - Nuclear                 1     45.18 3571.5  870.50
## - Motores.diésel          1     66.28 3592.6  872.65
## - Solar.térmica           1     69.53 3595.8  872.98
## - Carbón                  1    210.14 3736.4  886.99
## - Turbinación.bombeo      1    257.40 3783.7  891.57
## - Eólica                  1    317.56 3843.8  897.33
## - Ciclo.combinado         1    865.11 4391.4  945.94
## - Cogeneración            1   1048.28 4574.6  960.86
## - Mes                     1   1784.62 5310.9 1015.33
##
## Step:  AIC=866.7
## Precio ~ Mes + tmed + prec + tmin + tmax + Hidráulica + Turbinación.bombeo +
##     Nuclear + Carbón + Motores.diésel + Turbina.de.vapor +
##     Ciclo.combinado + Hidroeólica + Eólica + Solar.térmica +
##     Cogeneración + Residuos.no.renovables + Residuos.renovables
##
##                          Df Sum of Sq    RSS     AIC
## - Hidráulica              1      9.69 3544.1  865.70
## - prec                    1     10.25 3544.7  865.76
## - tmax                    1     11.77 3546.2  865.91
```

```
## - tmed                    1     11.79 3546.2  865.91
## - tmin                    1     12.58 3547.0  866.00
## - Residuos.renovables     1     15.72 3550.1  866.32
## - Turbina.de.vapor        1     17.29 3551.7  866.48
## <none>                                3534.4  866.70
## - Hidroeólica             1     25.06 3559.5  867.28
## + Otras.renovables        1      8.13 3526.3  867.86
## + Fuel...Gas              1      6.67 3527.7  868.01
## - Residuos.no.renovables  1     32.75 3567.2  868.07
## + Demanda                 1      4.61 3529.8  868.22
## + Dia                     1      4.36 3530.1  868.25
## + Turbina.de.gas          1      2.66 3531.8  868.42
## + Generación.total        1      0.92 3533.5  868.60
## + Solar.fotovoltaica      1      0.00 3534.4  868.70
## - Nuclear                 1     43.38 3577.8  869.15
## - Motores.diésel          1     67.85 3602.3  871.64
## - Solar.térmica           1     69.72 3604.1  871.83
## - Carbón                  1    226.29 3760.7  887.35
## - Turbinación.bombeo      1    260.97 3795.4  890.70
## - Eólica                  1    309.75 3844.2  895.36
## - Ciclo.combinado         1    887.56 4422.0  946.47
## - Cogeneración            1   1107.36 4641.8  964.18
## - Mes                     1   1790.16 5324.6 1014.27
##
## Step:  AIC=865.7
## Precio ~ Mes + tmed + prec + tmin + tmax + Turbinación.bombeo +
##     Nuclear + Carbón + Motores.diésel + Turbina.de.vapor +
##     Ciclo.combinado + Hidroeólica + Eólica + Solar.térmica +
##     Cogeneración + Residuos.no.renovables + Residuos.renovables
##
##                           Df Sum of Sq    RSS     AIC
## - prec                     1      9.99 3554.1  864.73
## - tmax                     1     12.46 3556.6  864.98
## - tmed                     1     12.55 3556.7  864.99
## - tmin                     1     13.31 3557.4  865.07
## - Residuos.renovables      1     16.86 3561.0  865.43
## <none>                                3544.1  865.70
## - Turbina.de.vapor         1     20.58 3564.7  865.81
## - Hidroeólica              1     26.13 3570.2  866.38
## - Residuos.no.renovables   1     28.63 3572.7  866.64
## + Hidráulica               1      9.69 3534.4  866.70
## + Generación.total         1      8.81 3535.3  866.79
## + Dia                      1      6.83 3537.3  866.99
## + Fuel...Gas               1      6.47 3537.6  867.03
## + Otras.renovables         1      6.06 3538.0  867.07
## + Solar.fotovoltaica       1      1.06 3543.0  867.59
## + Demanda                  1      0.98 3543.1  867.60
## + Turbina.de.gas           1      0.67 3543.4  867.63
## - Nuclear                  1     46.06 3590.2  868.41
## - Motores.diésel           1     67.32 3611.4  870.57
## - Solar.térmica            1     73.18 3617.3  871.16
## - Carbón                   1    219.50 3763.6  885.63
## - Eólica                   1    300.05 3844.2  893.36
## - Turbinación.bombeo       1    338.93 3883.0  897.03
## - Ciclo.combinado          1    896.66 4440.8  946.02
## - Cogeneración             1   1128.79 4672.9  964.62
## - Mes                      1   2036.48 5580.6 1029.41
##
## Step:  AIC=864.73
## Precio ~ Mes + tmed + tmin + tmax + Turbinación.bombeo + Nuclear +
##     Carbón + Motores.diésel + Turbina.de.vapor + Ciclo.combinado +
##     Hidroeólica + Eólica + Solar.térmica + Cogeneración +
##     Residuos.no.renovables + Residuos.renovables
##
##                           Df Sum of Sq    RSS     AIC
## - tmax                     1     14.36 3568.5  864.20
```

```
## - tmed                    1     14.41 3568.5  864.20
## - tmin                    1     15.16 3569.3  864.28
## - Residuos.renovables     1     15.90 3570.0  864.35
## <none>                                3554.1  864.73
## - Turbina.de.vapor        1     19.54 3573.6  864.73
## - Hidroeólica             1     24.37 3578.5  865.22
## - Residuos.no.renovables  1     26.25 3580.3  865.41
## + prec                    1      9.99 3544.1  865.70
## + Hidráulica              1      9.43 3544.7  865.76
## + Generación.total        1      9.16 3544.9  865.78
## + Dia                     1      6.41 3547.7  866.07
## + Fuel...Gas              1      6.33 3547.8  866.07
## + Otras.renovables        1      5.19 3548.9  866.19
## + Turbina.de.gas          1      0.86 3553.2  866.64
## + Demanda                 1      0.85 3553.3  866.64
## + Solar.fotovoltaica      1      0.05 3554.0  866.72
## - Nuclear                 1     50.21 3604.3  867.84
## - Motores.diésel          1     64.95 3619.0  869.33
## - Solar.térmica           1     84.06 3638.2  871.26
## - Carbón                  1    215.26 3769.4  884.19
## - Eólica                  1    290.56 3844.7  891.41
## - Turbinación.bombeo      1    338.75 3892.8  895.95
## - Ciclo.combinado         1    900.42 4454.5  945.15
## - Cogeneración            1   1132.05 4686.2  963.65
## - Mes                     1   2030.79 5584.9 1027.69
##
## Step:  AIC=864.2
## Precio ~ Mes + tmed + tmin + Turbinación.bombeo + Nuclear +
##     Carbón + Motores.diésel + Turbina.de.vapor + Ciclo.combinado +
##     Hidroeólica + Eólica + Solar.térmica + Cogeneración +
##     Residuos.no.renovables + Residuos.renovables
##
##                          Df Sum of Sq    RSS     AIC
## - tmed                    1      0.10 3568.6  862.21
## - tmin                    1      6.27 3574.7  862.84
## - Residuos.renovables     1     14.67 3583.1  863.69
## <none>                                3568.5  864.20
## - Turbina.de.vapor        1     22.89 3591.3  864.53
## + tmax                    1     14.36 3554.1  864.73
## + prec                    1     11.89 3556.6  864.98
## - Hidroeólica             1     28.58 3597.0  865.11
## - Residuos.no.renovables  1     28.88 3597.3  865.14
## + Hidráulica              1     10.14 3558.3  865.16
## + Generación.total        1      9.88 3558.6  865.19
## + Dia                     1      7.65 3560.8  865.41
## + Fuel...Gas              1      6.26 3562.2  865.56
## + Otras.renovables        1      4.43 3564.0  865.74
## + Demanda                 1      1.14 3567.3  866.08
## + Turbina.de.gas          1      0.53 3567.9  866.14
## + Solar.fotovoltaica      1      0.09 3568.4  866.19
## - Nuclear                 1     46.34 3614.8  866.91
## - Motores.diésel          1     73.80 3642.3  869.67
## - Solar.térmica           1     86.12 3654.6  870.90
## - Carbón                  1    222.57 3791.0  884.28
## - Eólica                  1    282.55 3851.0  890.01
## - Turbinación.bombeo      1    336.77 3905.2  895.11
## - Ciclo.combinado         1    898.40 4466.9  944.16
## - Cogeneración            1   1161.60 4730.1  965.06
## - Mes                     1   2016.94 5585.4 1025.73
##
## Step:  AIC=862.21
## Precio ~ Mes + tmin + Turbinación.bombeo + Nuclear + Carbón +
##     Motores.diésel + Turbina.de.vapor + Ciclo.combinado + Hidroeólica +
##     Eólica + Solar.térmica + Cogeneración + Residuos.no.renovables +
##     Residuos.renovables
##
```

```
##                           Df Sum of Sq    RSS     AIC
## - Residuos.renovables      1     14.58 3583.1  861.70
## <none>                                 3568.6  862.21
## - Turbina.de.vapor         1     22.80 3591.4  862.53
## + prec                     1     11.63 3556.9  863.02
## - Hidroeólica              1     28.55 3597.1  863.12
## + Hidráulica               1     10.23 3558.3  863.16
## - Residuos.no.renovables   1     29.09 3597.6  863.17
## + Generación.total         1      9.98 3558.6  863.19
## + Dia                      1      7.66 3560.9  863.42
## + Fuel...Gas               1      6.30 3562.3  863.56
## + Otras.renovables         1      4.47 3564.1  863.75
## + Demanda                  1      1.15 3567.4  864.09
## + Turbina.de.gas           1      0.55 3568.0  864.15
## + Solar.fotovoltaica       1      0.16 3568.4  864.19
## + tmed                     1      0.10 3568.5  864.20
## + tmax                     1      0.06 3568.5  864.20
## - Nuclear                  1     46.24 3614.8  864.91
## - tmin                     1     47.35 3615.9  865.02
## - Motores.diésel           1     73.81 3642.4  867.68
## - Solar.térmica            1    102.29 3670.8  870.52
## - Carbón                   1    222.55 3791.1  882.29
## - Eólica                   1    314.15 3882.7  891.00
## - Turbinación.bombeo       1    337.69 3906.3  893.21
## - Ciclo.combinado          1    898.30 4466.9  942.16
## - Cogeneración             1   1169.39 4738.0  963.66
## - Mes                      1   2017.12 5585.7 1023.74
##
## Step:  AIC=861.7
## Precio ~ Mes + tmin + Turbinación.bombeo + Nuclear + Carbón +
##     Motores.diésel + Turbina.de.vapor + Ciclo.combinado + Hidroeólica +
##     Eólica + Solar.térmica + Cogeneración + Residuos.no.renovables
##
##                           Df Sum of Sq    RSS     AIC
## - Residuos.no.renovables   1     18.21 3601.4  861.55
## - Turbina.de.vapor         1     19.39 3602.5  861.67
## <none>                                 3583.1  861.70
## - Hidroeólica              1     24.68 3607.8  862.20
## + Residuos.renovables      1     14.58 3568.6  862.21
## + Hidráulica               1     10.92 3572.2  862.58
## + prec                     1     10.78 3572.4  862.60
## + Generación.total         1     10.35 3572.8  862.64
## + Fuel...Gas               1      8.20 3574.9  862.86
## + Dia                      1      7.18 3576.0  862.96
## + Otras.renovables         1      6.17 3577.0  863.07
## - Nuclear                  1     34.13 3617.3  863.16
## + Demanda                  1      1.37 3581.8  863.56
## + Turbina.de.gas           1      0.40 3582.7  863.66
## + Solar.fotovoltaica       1      0.14 3583.0  863.68
## + tmax                     1      0.04 3583.1  863.69
## + tmed                     1      0.01 3583.1  863.69
## - tmin                     1     47.32 3630.5  864.48
## - Motores.diésel           1     70.87 3654.0  866.84
## - Solar.térmica            1    117.32 3700.5  871.46
## - Carbón                   1    216.15 3799.3  881.08
## - Eólica                   1    315.97 3899.1  890.54
## - Turbinación.bombeo       1    357.81 3941.0  894.44
## - Ciclo.combinado          1    891.06 4474.2  940.76
## - Cogeneración             1   1165.94 4749.1  962.52
## - Mes                      1   2077.40 5660.5 1026.60
##
## Step:  AIC=861.55
## Precio ~ Mes + tmin + Turbinación.bombeo + Nuclear + Carbón +
##     Motores.diésel + Turbina.de.vapor + Ciclo.combinado + Hidroeólica +
##     Eólica + Solar.térmica + Cogeneración
##
```

```
##                          Df Sum of Sq    RSS     AIC
## - Turbina.de.vapor        1     14.51 3615.9  861.01
## <none>                                3601.4  861.55
## + Residuos.no.renovables  1     18.21 3583.1  861.70
## - Hidroeólica             1     25.91 3627.3  862.16
## + Dia                     1     10.27 3591.1  862.50
## + prec                    1      9.39 3592.0  862.59
## + Fuel...Gas              1      7.49 3593.9  862.79
## + Generación.total        1      6.70 3594.7  862.87
## + Hidráulica              1      6.22 3595.1  862.91
## + Residuos.renovables     1      3.70 3597.6  863.17
## + Otras.renovables        1      2.22 3599.1  863.32
## + tmax                    1      0.56 3600.8  863.49
## + Turbina.de.gas          1      0.55 3600.8  863.49
## + tmed                    1      0.44 3600.9  863.50
## + Demanda                 1      0.21 3601.1  863.52
## + Solar.fotovoltaica      1      0.02 3601.3  863.54
## - Nuclear                 1     51.19 3652.5  864.70
## - tmin                    1     54.52 3655.9  865.03
## - Motores.diésel          1     79.95 3681.3  867.56
## - Solar.térmica           1    103.79 3705.1  869.92
## - Carbón                  1    214.18 3815.5  880.63
## - Eólica                  1    299.02 3900.4  888.66
## - Turbinación.bombeo      1    351.41 3952.8  893.53
## - Ciclo.combinado         1    982.77 4584.1  947.62
## - Cogeneración            1   1188.88 4790.2  963.67
## - Mes                     1   2062.23 5663.6 1024.80
##
## Step:  AIC=861.01
## Precio ~ Mes + tmin + Turbinación.bombeo + Nuclear + Carbón +
##     Motores.diésel + Ciclo.combinado + Hidroeólica + Eólica +
##     Solar.térmica + Cogeneración
##
##                          Df Sum of Sq    RSS     AIC
## - Hidroeólica             1     13.70 3629.6  860.39
## <none>                                3615.9  861.01
## + Turbina.de.vapor        1     14.51 3601.4  861.55
## + Residuos.no.renovables  1     13.32 3602.5  861.67
## + Dia                     1     11.18 3604.7  861.88
## + Generación.total        1      9.93 3605.9  862.01
## + prec                    1      9.08 3606.8  862.10
## + Hidráulica              1      8.76 3607.1  862.13
## + Fuel...Gas              1      6.57 3609.3  862.35
## + Otras.renovables        1      3.63 3612.2  862.65
## + Residuos.renovables     1      3.13 3612.7  862.70
## + Demanda                 1      0.99 3614.9  862.91
## + tmax                    1      0.91 3614.9  862.92
## + tmed                    1      0.75 3615.1  862.94
## + Turbina.de.gas          1      0.17 3615.7  863.00
## + Solar.fotovoltaica      1      0.13 3615.7  863.00
## - Nuclear                 1     43.93 3659.8  863.42
## - tmin                    1     57.84 3673.7  864.81
## - Motores.diésel          1     79.96 3695.8  867.00
## - Solar.térmica           1     96.86 3712.7  868.66
## - Carbón                  1    240.37 3856.2  882.50
## - Eólica                  1    295.10 3911.0  887.65
## - Turbinación.bombeo      1    383.99 3999.8  895.85
## - Ciclo.combinado         1    988.43 4604.3  947.22
## - Cogeneración            1   1402.45 5018.3  978.65
## - Mes                     1   2191.17 5807.0 1031.93
##
## Step:  AIC=860.39
## Precio ~ Mes + tmin + Turbinación.bombeo + Nuclear + Carbón +
##     Motores.diésel + Ciclo.combinado + Eólica + Solar.térmica +
##     Cogeneración
##
```

```
##                          Df Sum of Sq    RSS     AIC
## <none>                                  3629.6  860.39
## + Residuos.no.renovables  1     16.41 3613.2  860.74
## + Hidroeólica             1     13.70 3615.9  861.01
## + Generación.total        1      8.93 3620.6  861.49
## + prec                    1      8.04 3621.5  861.58
## + Hidráulica              1      7.89 3621.7  861.60
## + Dia                     1      7.37 3622.2  861.65
## + Fuel...Gas              1      6.26 3623.3  861.76
## + Turbina.de.vapor        1      2.30 3627.3  862.16
## + Residuos.renovables     1      1.92 3627.6  862.20
## + Otras.renovables        1      1.54 3628.0  862.24
## + Demanda                 1      1.35 3628.2  862.26
## + tmax                    1      0.71 3628.8  862.32
## + tmed                    1      0.56 3629.0  862.34
## + Solar.fotovoltaica      1      0.04 3629.5  862.39
## + Turbina.de.gas          1      0.02 3629.5  862.39
## - Nuclear                 1     60.68 3690.2  864.45
## - tmin                    1     69.21 3698.8  865.29
## - Motores.diésel          1     69.85 3699.4  865.35
## - Solar.térmica           1    128.79 3758.3  871.12
## - Carbón                  1    227.77 3857.3  880.61
## - Turbinación.bombeo      1    375.92 4005.5  894.37
## - Eólica                  1    414.98 4044.5  897.91
## - Ciclo.combinado         1   1076.26 4705.8  953.18
## - Cogeneración            1   1489.02 5118.6  983.87
## - Mes                     1   2234.21 5863.8 1033.48

##
## Call:
## lm(formula = Precio ~ Mes + tmin + Turbinación.bombeo + Nuclear +
##     Carbón + Motores.diésel + Ciclo.combinado + Eólica + Solar.térmica +
##     Cogeneración, data = datos_r)
##
## Coefficients:
##       (Intercept)                Mes               tmin  Turbinación.bombeo
##          18.01839            -1.40170            -0.19253            -0.30387
##           Nuclear              Carbón      Motores.diésel     Ciclo.combinado
##          -0.02829             0.04546            -0.67433             0.05948
##            Eólica       Solar.térmica        Cogeneración
##          -0.02169            -0.10055             0.56920
```

```
modelo_Corregido <- lm(formula = Precio ~ Mes + tmin + Hidráulica +
Turbinación.bombeo +
    Nuclear + Motores.diésel + Turbina.de.vapor + Hidroeólica +
    Eólica + Solar.térmica + Residuos.no.renovables + Generación.total +
    Cogeneración, data = datos_r)

summary(modelo_Corregido)
```

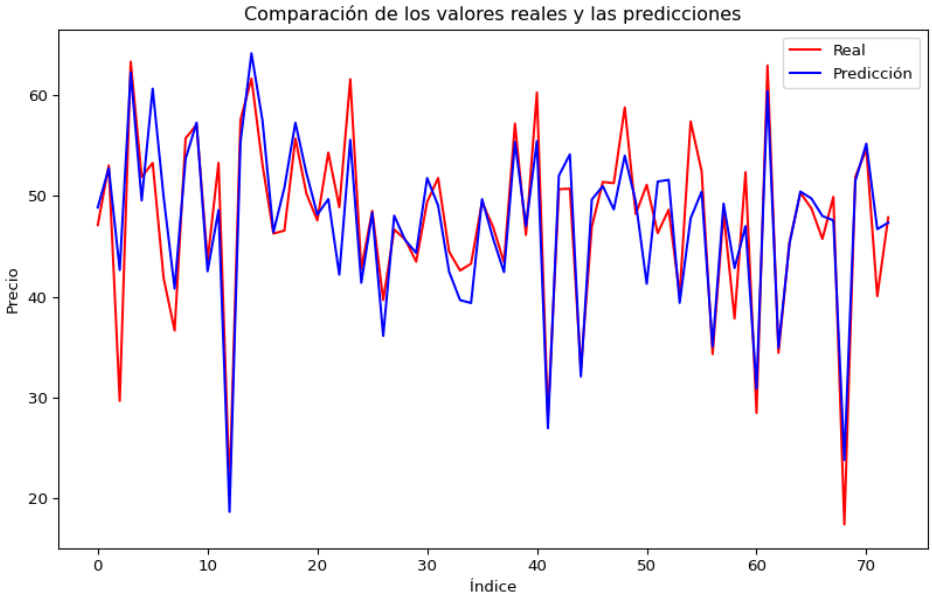
```
##
## Call:
## lm(formula = Precio ~ Mes + tmin + Hidráulica + Turbinación.bombeo +
##     Nuclear + Motores.diésel + Turbina.de.vapor + Hidroeólica +
##     Eólica + Solar.térmica + Residuos.no.renovables + Generación.total +
##     Cogeneración, data = datos_r)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.5763  -1.9923   0.1056   1.8002  15.0542
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          18.397796   4.853480   3.791 0.000177 ***
## Mes                  -1.386641   0.087014 -15.936  < 2e-16 ***
## tmin                 -0.141321   0.071314  -1.982 0.048297 *
```

```
## Hidráulica             -0.061864   0.009217  -6.712 7.71e-11 ***
## Turbinación.bombeo     -0.336367   0.054174  -6.209 1.51e-09 ***
## Nuclear                -0.082546   0.012440  -6.635 1.23e-10 ***
## Motores.diésel         -0.886523   0.323362  -2.742 0.006428 **
## Turbina.de.vapor       -0.302838   0.214537  -1.412 0.158956
## Hidroeólica            -9.834231   6.805524  -1.445 0.149340
## Eólica                 -0.077680   0.003535 -21.975  < 2e-16 ***
## Solar.térmica          -0.186394   0.026726  -6.974 1.53e-11 ***
## Residuos.no.renovables -0.596729   0.310578  -1.921 0.055497 .
## Generación.total        0.055204   0.005004  11.031  < 2e-16 ***
## Cogeneración            0.568807   0.060019   9.477  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.198 on 351 degrees of freedom
## Multiple R-squared:  0.8867, Adjusted R-squared:  0.8825
## F-statistic: 211.2 on 13 and 351 DF,  p-value: < 2.2e-16
```

Regresión de Ridge

```
print('Modelo de regresion de Ridge')

## Modelo de regresion de Ridge

from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split

#separar datos de entrenamiento y de prueba por el 80% y 20%
X = datos.iloc[:, :-1].values

#datos de la columna precio es la utlima columna
y = datos.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 10)

#entrenar el modelo
ridgeReg = Ridge(alpha=0.05)

ridgeReg.fit(X_train,y_train)

#predecir los resultados

## Ridge(alpha=0.05)

y_pred = ridgeReg.predict(X_test)

#comparar los resultados
df = pd.DataFrame({'Real': y_test, 'Prediccion': y_pred})

print(df)

#error cuadratico medio

##          Real  Prediccion
## 0   47.112917   48.861822
## 1   53.027917   52.765802
## 2   29.685833   42.661427
## 3   63.319583   62.252826
## 4   51.894167   49.562713
## ..        ...         ...
## 68  17.433333   23.812179
## 69  51.887500   51.479868
## 70  54.637083   55.189383
```

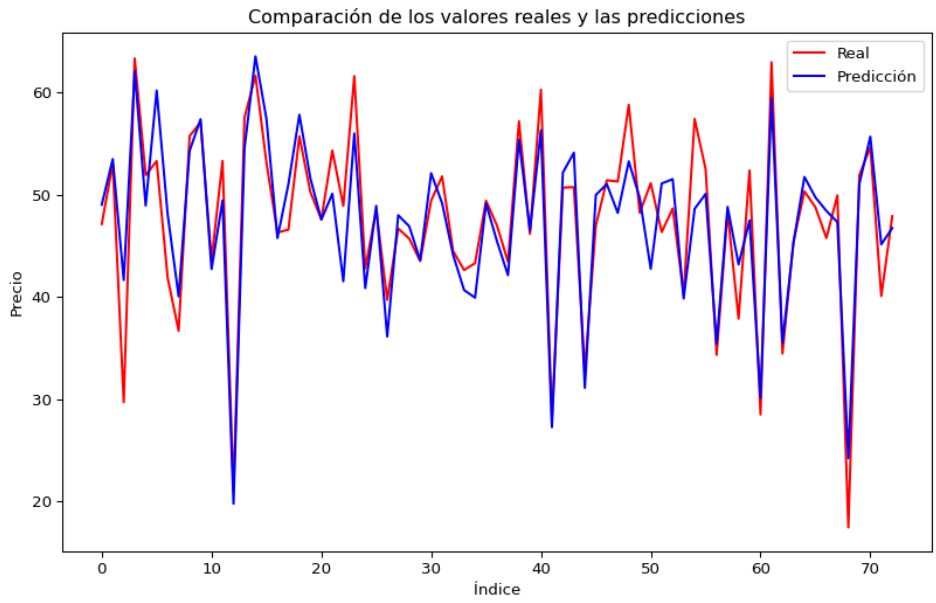
```
## 71  40.071250   46.709283
## 72  47.886667   47.357318
##
## [73 rows x 2 columns]

from sklearn import metrics
print('Error cuadratico medio:', metrics.mean_squared_error(y_test, y_pred))

# Calcular el R^2

## Error cuadratico medio: 14.230008670672673

r2 = ridgeReg.score(X_test, y_test)
# Imprimir el R^2
print('Coeficiente de determinación R^2:', r2)

#Grafica de predicciones

## Coeficiente de determinación R^2: 0.8240006964116204

import matplotlib.pyplot as plt

# Crear una figura
plt.figure(figsize=(10,6))

# Crear un gráfico de línea con los datos reales
plt.plot(range(len(y_test)), y_test, color = 'red', label = 'Real')

# Crear un gráfico de línea con las predicciones
plt.plot(range(len(y_pred)), y_pred, color = 'blue', label = 'Predicción')

# Añadir títulos y etiquetas
plt.title('Comparación de los valores reales y las predicciones')
plt.xlabel('Índice')
plt.ylabel('Precio')
plt.legend()

# Mostrar la gráfica
plt.show()
```



Regresión Bayesiana

```
print('Modelo de regresion Bayesiana')

## Modelo de regresion Bayesiana

from sklearn.linear_model import BayesianRidge
from sklearn.model_selection import train_test_split

#separar datos de entrenamiento y de prueba por el 80% y 20%
X = datos.iloc[:, :-1].values

#datos de la columna precio es la utlima columna
y = datos.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 10)

#entrenar el modelo
bayesianRidge = BayesianRidge()

bayesianRidge.fit(X_train,y_train)

#predecir los resultados

## BayesianRidge()

y_pred = bayesianRidge.predict(X_test)

#comparar los resultados
df = pd.DataFrame({'Real': y_test, 'Prediccion': y_pred})

print(df)

#error cuadratico medio

##          Real  Prediccion
## 0   47.112917   49.002203
## 1   53.027917   53.471326
## 2   29.685833   41.633746
## 3   63.319583   62.077755
## 4   51.894167   48.918363
## ..        ...         ...
## 68  17.433333   24.195787
## 69  51.887500   51.041717
## 70  54.637083   55.667718
## 71  40.071250   45.125887
## 72  47.886667   46.724344
##
## [73 rows x 2 columns]

from sklearn import metrics

#pasar y_pred de arraylike a numpy array
y_pred = np.asarray(y_pred)

print('Error cuadratico medio:', metrics.mean_squared_error(y_test, y_pred))

# Calcular el R^2

## Error cuadratico medio: 12.618771377931617
```

```
r2 = bayesianRidge.score(X_test, y_test)
# Imprimir el R^2
print('Coeficiente de determinación R^2:', r2)

#Grafica de predicciones

## Coeficiente de determinación R^2: 0.8439287686989192

import matplotlib.pyplot as plt

# Crear una figura
plt.figure(figsize=(10,6))

# Crear un gráfico de línea con los datos reales
plt.plot(range(len(y_test)), y_test, color = 'red', label = 'Real')

# Crear un gráfico de línea con las predicciones
plt.plot(range(len(y_pred)), y_pred, color = 'blue', label = 'Predicción')

# Añadir títulos y etiquetas
plt.title('Comparación de los valores reales y las predicciones')
plt.xlabel('Índice')
plt.ylabel('Precio')
plt.legend()

# Mostrar la gráfica
plt.show()
```

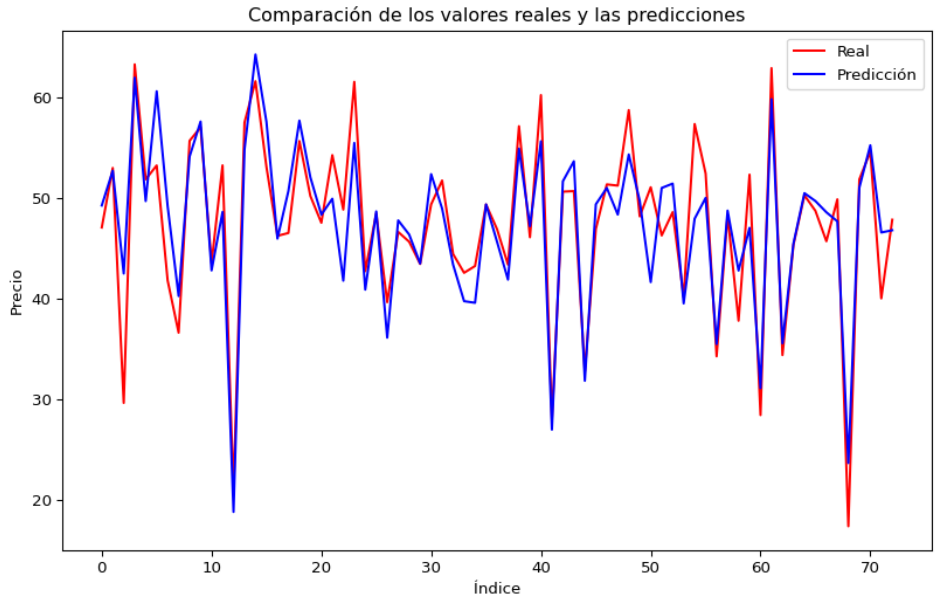


Proceso Gaussiano

```
print('Modelo de proceso Gaussiano')

## Modelo de proceso Gaussiano

from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import DotProduct, WhiteKernel
from sklearn.model_selection import train_test_split

#separar datos de entrenamiento y de prueba por el 80% y 20%
X = datos.iloc[:, :-1].values
```

```
#datos de la columna precio es la utlima columna
y = datos.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 10)

#entrenar el modelo
kernel = DotProduct() + WhiteKernel()
gaussianProcess = GaussianProcessRegressor(kernel=kernel, random_state=0)

gaussianProcess.fit(X_train,y_train)

#predecir los resultados

## GaussianProcessRegressor(kernel=DotProduct(sigma_0=1) +
WhiteKernel(noise_level=1),
##                          random_state=0)

y_pred = gaussianProcess.predict(X_test)

#comparar los resultados
df = pd.DataFrame({'Real': y_test, 'Prediccion': y_pred})

print(df)

#Error cuadratico medio

##          Real  Prediccion
## 0   47.112917   49.298285
## 1   53.027917   52.754925
## 2   29.685833   42.539162
## 3   63.319583   61.989203
## 4   51.894167   49.737571
## ..        ...         ...
## 68  17.433333   23.716447
## 69  51.887500   51.064063
## 70  54.637083   55.284260
## 71  40.071250   46.617064
## 72  47.886667   46.840777
##
## [73 rows x 2 columns]

from sklearn import metrics
print('Error cuadratico medio:', metrics.mean_squared_error(y_test, y_pred))

# Calcular el R^2

## Error cuadratico medio: 13.893275765409523

r2 = gaussianProcess.score(X_test, y_test)
# Imprimir el R^2
print('Coeficiente de determinación R^2:', r2)

#Grafica de predicciones

## Coeficiente de determinación R^2: 0.8281654694762882

import matplotlib.pyplot as plt

# Crear una figura
plt.figure(figsize=(10,6))

# Crear un gráfico de línea con los datos reales
```

```python
plt.plot(range(len(y_test)), y_test, color = 'red', label = 'Real')

# Crear un gráfico de línea con las predicciones
plt.plot(range(len(y_pred)), y_pred, color = 'blue', label = 'Predicción')

# Añadir títulos y etiquetas
plt.title('Comparación de los valores reales y las predicciones')
plt.xlabel('Índice')
plt.ylabel('Precio')
plt.legend()

# Mostrar la gráfica
plt.show()
```



Regresión Polinomial en R:

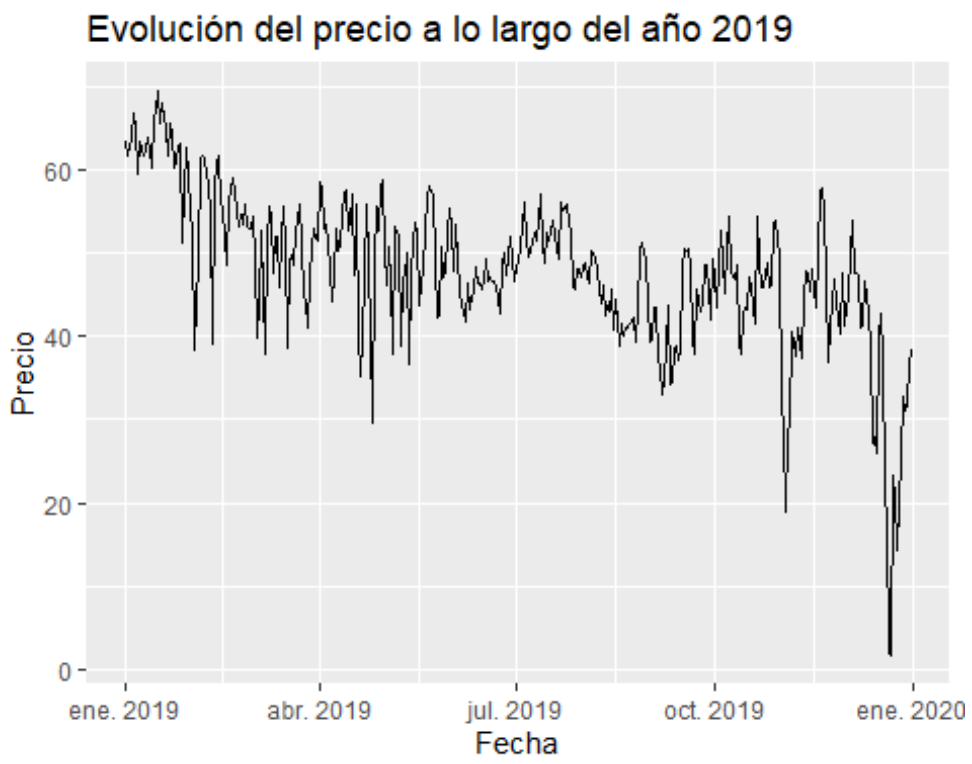
```r
# Cargar el paquete glmnet
library(Matrix)
library(glmnet)

## Warning: package 'glmnet' was built under R version 4.2.3

## Loaded glmnet 4.1-7

# Asumiendo que tus datos están en un data.frame llamado datos_r y Precio es tu
variable dependiente
# Primero, debes convertir tus variables independientes a una matriz
x <- model.matrix(Precio ~ ., datos_r)[,-1]

# Luego, convierte tu variable dependiente a un vector
y <- datos_r$Precio

# Ajustar el modelo de regresión de Ridge utilizando validación cruzada para
encontrar el mejor valor de lambda
modelo_ridge <- cv.glmnet(x, y, alpha = 0)

# Imprimir un resumen del modelo
print(modelo_ridge)

##
## Call:  cv.glmnet(x = x, y = y, alpha = 0)
```

```
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.6732   100   11.83 1.681      26
## 1se 2.7175    85   13.29 2.021      26

# Para obtener los coeficientes del modelo en el valor óptimo de lambda
coef(modelo_ridge, s = modelo_ridge$lambda.min)

## 28 x 1 sparse Matrix of class "dgCMatrix"
##                                    s1
## (Intercept)             1.906779e+01
## X                      -1.953465e-02
## Año                     .
## Mes                    -5.871310e-01
## Dia                     8.281675e-03
## tmed                   -6.720952e-02
## prec                    2.901773e-02
## tmin                   -1.160713e-01
## tmax                   -2.888342e-02
## Demanda                 1.182919e-02
## Hidráulica             -2.380293e-02
## Turbinación.bombeo     -2.840716e-01
## Nuclear                -2.983595e-02
## Carbón                  4.738807e-02
## Fuel...Gas              9.475576e+05
## Motores.diésel         -7.391247e-01
## Turbina.de.gas          1.568975e-01
## Turbina.de.vapor        4.879551e-02
## Ciclo.combinado         3.930460e-02
## Hidroeólica            -5.878772e+00
## Eólica                 -3.130362e-02
## Solar.fotovoltaica     -3.318845e-02
## Solar.térmica          -4.193642e-02
## Otras.renovables        2.149505e-01
## Cogeneración            4.985291e-01
## Residuos.no.renovables -3.009033e-01
## Residuos.renovables     9.307027e-02
## Generación.total        1.294382e-03
```

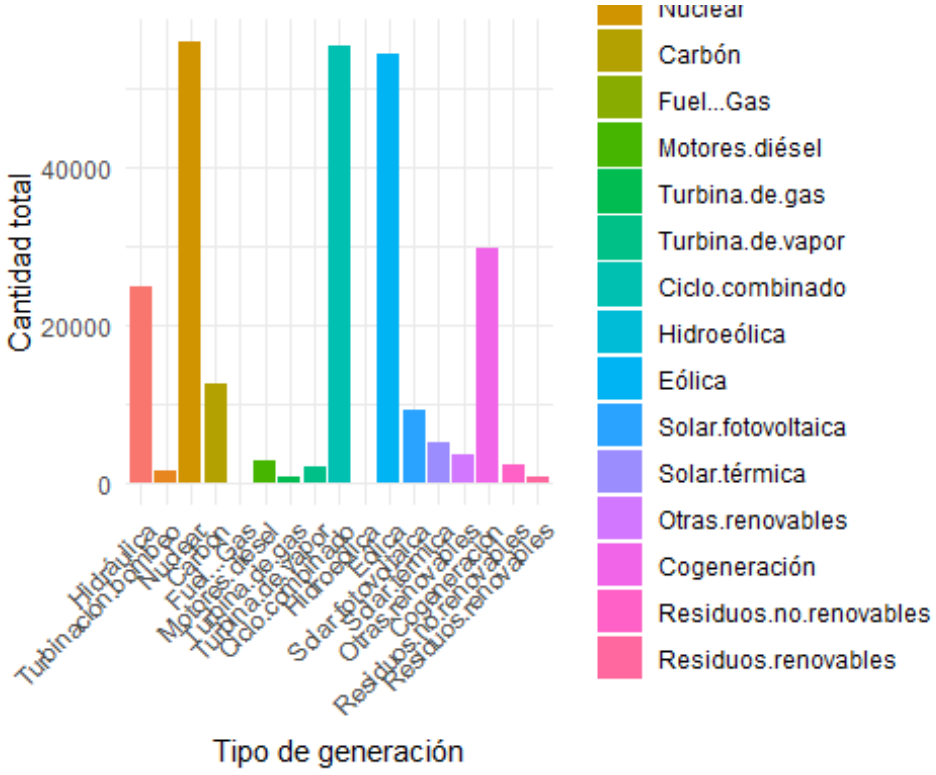
Graficos con R:

```
# Cargar la librería necesaria
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.2.3

##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha

library(reshape2)



# Crear una nueva columna "fecha" combinando año, mes y día
datos_r$fecha <- as.Date(paste(datos_r$Año, datos_r$Mes, datos_r$Dia, sep="-"),
format="%Y-%m-%d")

# Crear el gráfico
ggplot(data=datos_r, aes(x=fecha, y=Precio)) +
    geom_line() +
```

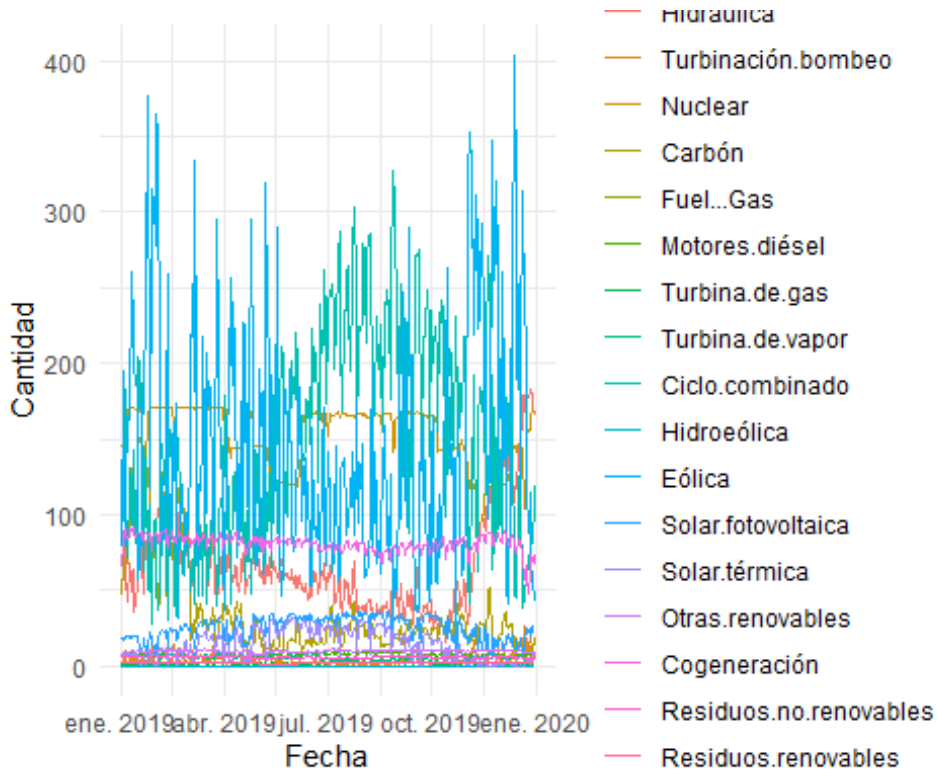
```
    labs(title="Evolución del precio a lo largo del año 2019",
         x="Fecha",
         y="Precio")
```



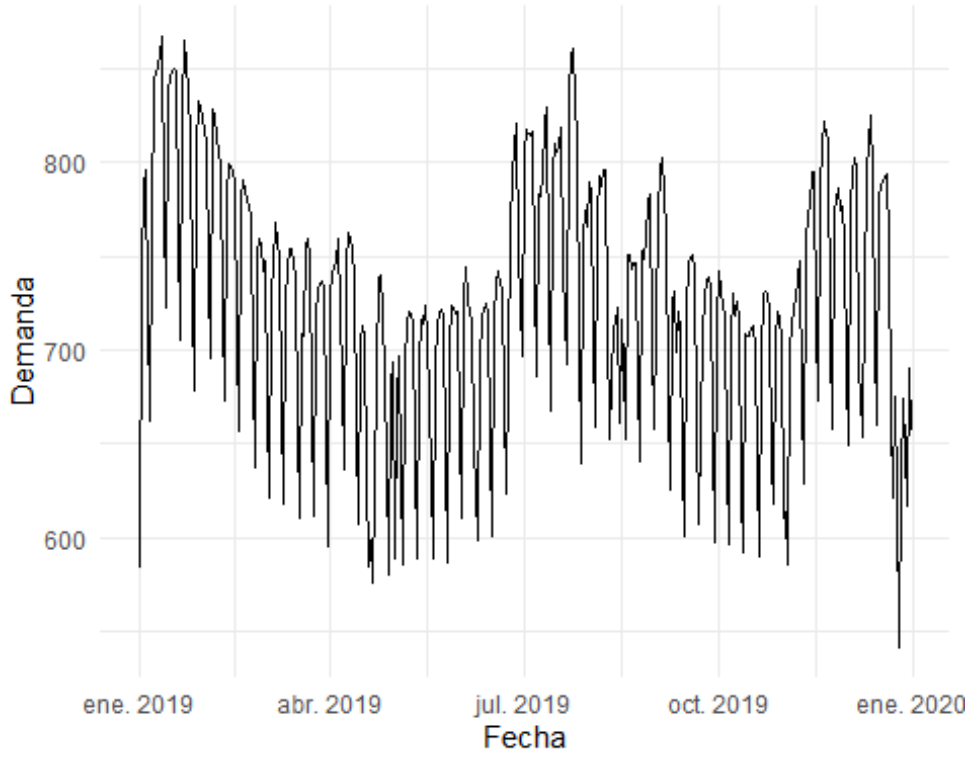

```
# Cambiar el formato del dataframe de ancho a largo
dataframe_largo <- melt(datos_r,
                        measure.vars = c("Hidráulica", "Turbinación.bombeo",
"Nuclear",
                                         "Carbón", "Fuel...Gas", "Motores.diésel",
                                         "Turbina.de.gas", "Turbina.de.vapor",
                                         "Ciclo.combinado", "Hidroeólica", "Eólica",
                                         "Solar.fotovoltaica", "Solar.térmica",
                                         "Otras.renovables", "Cogeneración",
                                         "Residuos.no.renovables",
"Residuos.renovables"))

# Calcular la suma total para cada tipo de generación de energía
sumas <- aggregate(value ~ variable, dataframe_largo, sum)

# Crear el gráfico de barras
ggplot(data = sumas, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity") +
  labs(x = "Tipo de generación", y = "Cantidad total", fill = "Tipo de generación") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Cambiar el formato del dataframe de ancho a largo
dataframe_largo <- melt(datos_r, id.vars = "fecha",
                        measure.vars = c("Hidráulica", "Turbinación.bombeo",
"Nuclear",
                                         "Carbón", "Fuel...Gas", "Motores.diésel",
                                         "Turbina.de.gas", "Turbina.de.vapor",
                                         "Ciclo.combinado", "Hidroeólica", "Eólica",
                                         "Solar.fotovoltaica", "Solar.térmica",
                                         "Otras.renovables", "Cogeneración",
                                         "Residuos.no.renovables",
"Residuos.renovables"))

# Crear el gráfico de líneas
ggplot(data = dataframe_largo, aes(x = fecha, y = value, color = variable)) +
  geom_line() +
  labs(x = "Fecha", y = "Cantidad", color = "Tipo de generación") +
  theme_minimal()
```

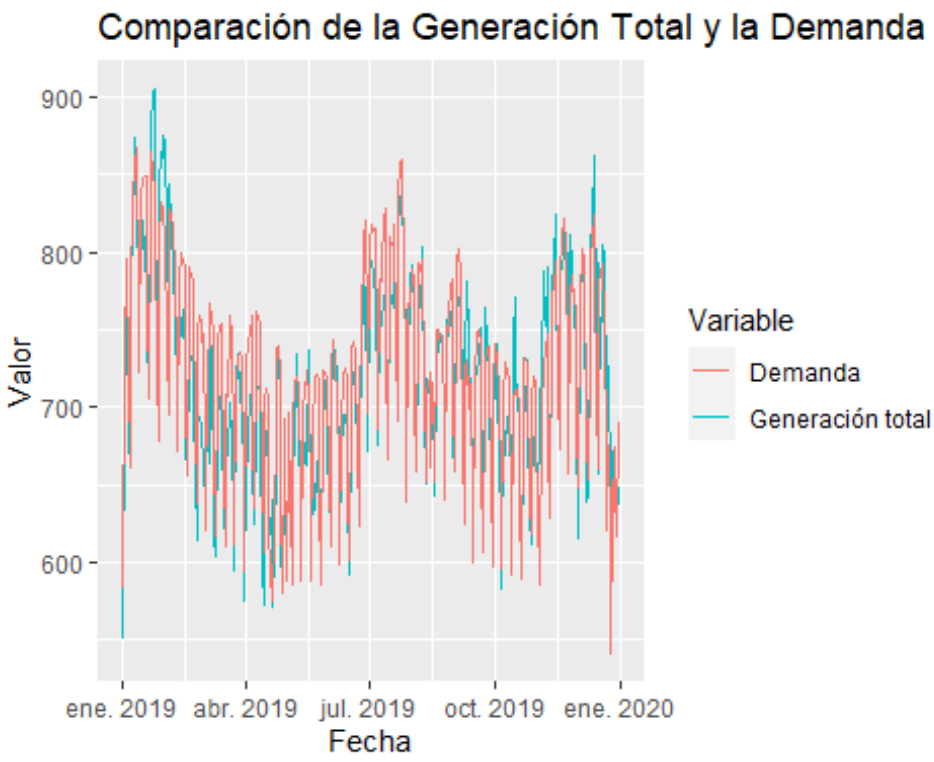



```r
# Crear el gráfico de líneas
ggplot(data = datos_r, aes(x = fecha, y = Demanda)) +
  geom_line() +
  labs(x = "Fecha", y = "Demanda") +
  theme_minimal()
```

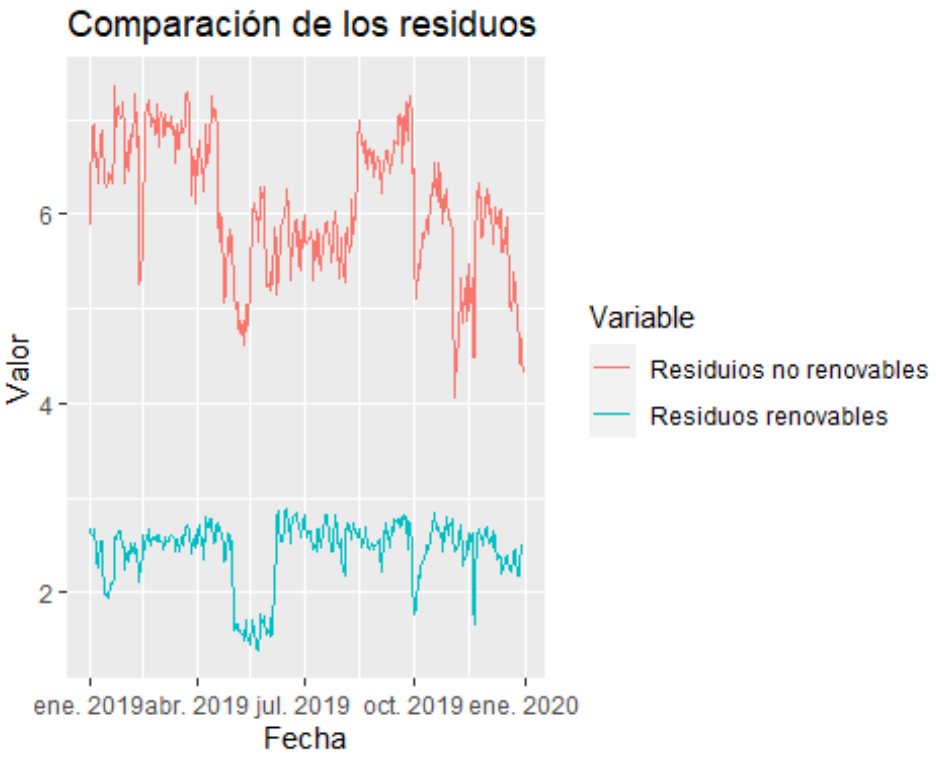



```r
# Crear un gráfico de líneas
ggplot(datos_r, aes(x = fecha)) +
  geom_line(aes(y = Generación.total, color = "Generación total")) +
```

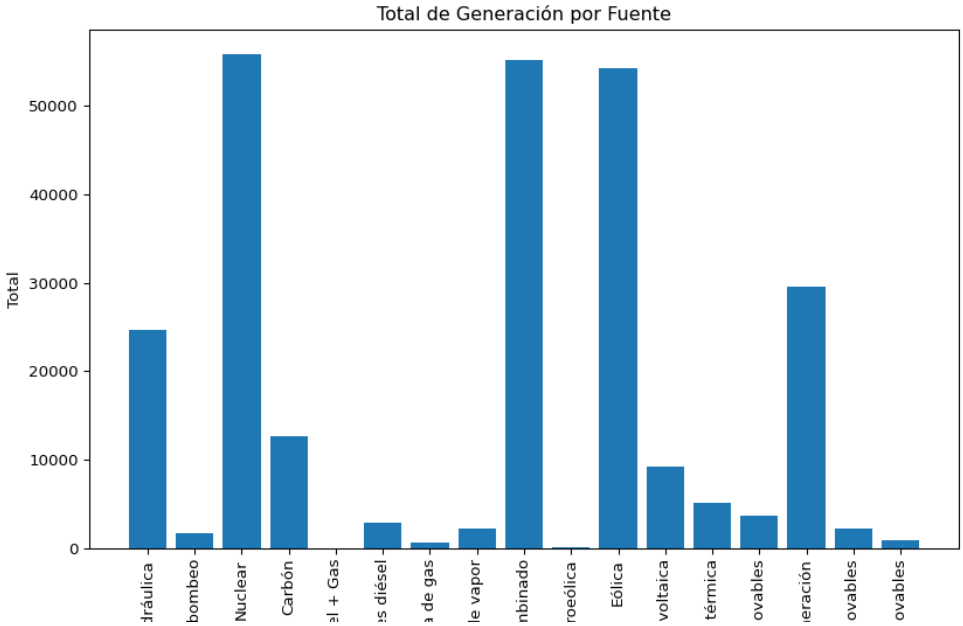
```
  geom_line(aes(y = Demanda, color = "Demanda")) +
  labs(x = "Fecha", y = "Valor", color = "Variable") +
  ggtitle("Comparación de la Generación Total y la Demanda")
```




```
# Crear un gráfico de líneas
ggplot(datos_r, aes(x = fecha)) +
  geom_line(aes(y = Residuos.no.renovables, color = "Residuios no renovables")) +
  geom_line(aes(y = Residuos.renovables, color = "Residuos renovables")) +
  labs(x = "Fecha", y = "Valor", color = "Variable") +
  ggtitle("Comparación de los residuos")
```




```
# Importar las bibliotecas necesarias
import pandas as pd
import matplotlib.pyplot as plt

# Asumiendo que tus datos están en un DataFrame de pandas llamado datos_r
# Calcular los totales de cada columna
totales = datos[["Hidráulica", "Turbinación bombeo", "Nuclear",
                 "Carbón", "Fuel + Gas", "Motores diésel",
                 "Turbina de gas", "Turbina de vapor",
                 "Ciclo combinado", "Hidroeólica", "Eólica",
                 "Solar fotovoltaica", "Solar térmica",
                 "Otras renovables", "Cogeneración",
                 "Residuos no renovables", "Residuos renovables"]].sum()

# Crear un gráfico de barras de los totales
plt.bar(totales.index, totales.values)

## <BarContainer object of 17 artists>

plt.title("Total de Generación por Fuente")
plt.ylabel("Total")
plt.xlabel("Fuente")
plt.xticks(rotation=90)

## ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16], [Text(0, 0, ''),
Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''),
Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''),
Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''),
Text(0, 0, '')])

plt.show()
```



Red Neuronal:

```
#-------------------------------------------------------------------------------
---------Modelo de redes neuronales

#Modelo de redes neuronales con funcion de activacion Sigmoide
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from sklearn import metrics

# crear modelo de red neuronal

# Datos x como variables independientes y y como variable dependiente
#y es la ultima columna
X = datos.iloc[:, :-1].values
#x es todo menos la ultima columna
y = datos.iloc[:, -1].values

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

# Normalizar los datos
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Crear la red neuronal
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(16, activation='relu'))
model.add(Dense(1))

# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')
```

```python
# Entrenar el modelo
model.fit(X_train, y_train, epochs=500, batch_size=32)

# Predecir los precios en el conjunto de prueba
```

```
## Epoch 1/500
##
##  1/10 [==>...........................] - ETA: 6s - loss: 2436.5823
## 10/10 [==============================] - 1s 2ms/step - loss: 2328.9729
## Epoch 2/500
##
…
##
##  1/10 [==>...........................] - ETA: 0s - loss: 1.6789
## 10/10 [==============================] - 0s 1ms/step - loss: 1.7111
## Epoch 500/500
##
##  1/10 [==>...........................] - ETA: 0s - loss: 1.1290
## 10/10 [==============================] - 0s 2ms/step - loss: 1.6987
## <keras.callbacks.History object at 0x00000231038345E0>
```

```python
y_pred = model.predict(X_test)

# Comparar los resultados
```

```
##
## 1/3 [=========>....................] - ETA: 0s
## 3/3 [==============================] - 0s 1ms/step
```

```python
df = pd.DataFrame({'Real': y_test, 'Prediccion': y_pred.flatten()})
print(df)

# Error cuadratico medio
```

```
##          Real  Prediccion
## 0   47.430417   45.985874
## 1   50.403750   47.803173
## 2   55.286667   57.428246
## 3   51.112500   43.907757
## 4   48.639167   49.263790
## ..        ...         ...
## 68  47.886667   55.493351
## 69  43.298333   42.639668
## 70  51.290000   54.528111
## 71  46.567917   47.088360
## 72  29.685833   39.721157
##
## [73 rows x 2 columns]
```

```python
from sklearn import metrics
print('Error cuadratico medio:', metrics.mean_squared_error(y_test, y_pred))

# R2
```

```
## Error cuadratico medio: 19.754664091225035
```

```python
print('R2:', metrics.r2_score(y_test, y_pred))

# Calcular el coeficiente de determinación R^2
```

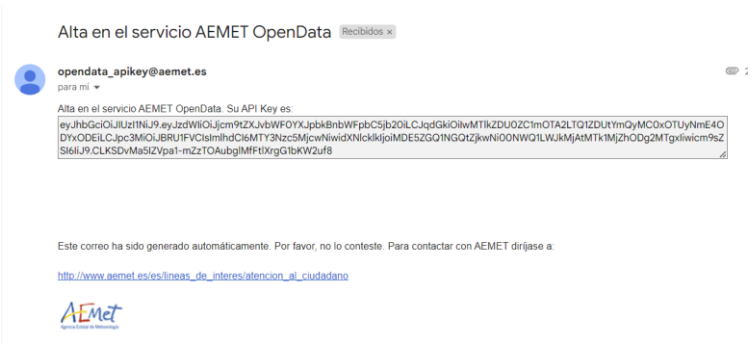
```
## R2: 0.6900131922675206
```

```python
r2 = metrics.r2_score(y_test, y_pred)
```

```
# Imprimir el resultado
print('Coeficiente de determinación R^2:', r2)

#Grafica de predicciones

## Coeficiente de determinación R^2: 0.6900131922675206

import matplotlib.pyplot as plt

# Crear una figura
plt.figure(figsize=(10,6))

# Crear un gráfico de línea con los datos reales
plt.plot(range(len(y_test)), y_test, color = 'red', label = 'Real')

# Crear un gráfico de línea con las predicciones
plt.plot(range(len(y_pred)), y_pred, color = 'blue', label = 'Predicción')

# Añadir títulos y etiquetas
plt.title('Comparación de los valores reales y las predicciones')
plt.xlabel('Índice')
plt.ylabel('Precio')
plt.legend()

# Mostrar la gráfica
plt.show()


#-------------------------------------------------------------------------------
---------
```

7.1.2 API Aemet:

Ejemplo de conexión API con credenciales obtenidas por correo electrónico:


Alta en el servicio AEMET OpenData Recibidos ×

opendata_apikey@aemet.es
para mí

Alta en el servicio AEMET OpenData. Su API Key es:

eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJjcm9tZXJvbWF0YXJpbkBnbWFpbC5jb20iLCJqdGkiOiIwMTlkZDU0ZC1mOTA2LTQ1ZDUtYmQyMC0xOTUyNmE4ODYxODEiLCJpc3MiOiJBRU1FVCIsImlhdCI6MTY3Nzc5MjcwNiwidXNlcklkIjoiMDE5ZGQ1NGQtZjkwNi00NWQ1LWJkMjAtMTk1MjZhODg2MTgxIiwicm9sZSI6IiJ9.CLKSDvMa5IZVpa1-mZzTOAubglMfFtlXrgG1bKW2uf8

Este correo ha sido generado automáticamente. Por favor, no lo conteste. Para contactar con AEMET diríjase a:

http://www.aemet.es/es/lineas_de_interes/atencion_al_ciudadano

AEMET

```
querystring =
{"api_key":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJjcm9tZXJvbWF0YXJpbkBnbWFpbC5j
b20iLCJqdGkiOiIwMTlkZDU0ZC1mOTA2LTQ1ZDUtYmQyMC0xOTUyNmE4ODYxODEiLCJpc3MiO
iJBRU1FVCIsImlhdCI6MTY3Nzc5MjcwNiwidXNlcklkIjoiMDE5ZGQ1NGQtZjkwNi00NWQ1LW
JkMjAtMTk1MjZhODg2MTgxIiwicm9sZSI6IiJ9.CLKSDvMa5IZVpa1-
mZzTOAubglMfFtlXrgG1bKW2uf8"}

    headers = {
        'cache-control': "no-cache"
        }

    #DATOS DE BARCELONA

    url = "https://opendata.aemet.es/opendata/sh/8e275d34"

    response = requests.request("GET", url, headers=headers,
params=querystring)

    print(response.text)

    datosJsonBarcelona = response.json()
```