



FACULTAD DE CIENCIAS

Memoria de Prácticas Externas

Grado en matemáticas

External Internship Report

Degree in Mathematics

Memoria de prácticas externas realizadas en

Evenbytes S.L

Alumno: Alberto Campuzano Solórzano
Doble Grado en Física y Matemáticas

Periodo: 16/06/2025 – 29/08/2025

Tipo de prácticas: Curriculares + Extracurriculares
Número de horas: 300 + 50

Tutor externo: Jacinto Pelayo, CEO
Tutor académico: Alicia Nieto Reyes

Santander, 18 de agosto de 2025

Resumen

Durante mis prácticas en Evenbytes S.L., una consultora tecnológica especializada en soluciones cloud y transformación digital, he desarrollado competencias, de manera principal, en ciencia de datos. En la fase inicial, centrada en la formación acerca del flujo de trabajo propio de la empresa, utilicé el *framework* Angular, el entorno de *Google Cloud Platform* (GCP) y el sistema de control de versiones *GitHub*. Posteriormente, participé en diversos proyectos, destacando especialmente uno orientado a la implementación de un modelo de predicción del precio de la electricidad y a la optimización de los procesos de fundición de unos hornos para la reducción del consumo energético. Esta experiencia me ha permitido integrar los conocimientos adquiridos en los grados de Matemáticas y Física, además de proporcionarme una visión más clara y aplicada del entorno profesional tecnológico.

Palabras clave: Ciencia de datos, modelo de predicción, machine learning, Google Cloud Platform.

Abstract

During my internship at Evenbytes S.L., a technology consulting firm specialized in *cloud* solutions and digital transformation, I primarily developed skills in data science. In the initial phase, focused on learning the company's workflow, I used the Angular *framework*, the *Google Cloud Platform* (GCP) environment, and the *GitHub* version control system. Subsequently, I participated in various projects, most notably one aimed at implementing a model for electricity price prediction and optimizing furnace casting processes to reduce energy consumption. This experience has allowed me to integrate the knowledge acquired during my Mathematics and Physics degrees, in addition to providing me with a clearer and more applied perspective of the professional tech environment.

Keywords: Data science, predictive model, machine learning, Google Cloud Platform.

Índice general

Siglas y acrónimos	4
Agradecimientos	5
1 Introducción	6
1.1 Elección	6
1.2 Objetivos	6
1.3 Centro	7
1.4 Práctica Externa	7
2 Descripción de las prácticas	8
2.1 Actividades desempeñadas	8
2.2 Formación	8
2.2.1 Angular	8
2.2.2 GCP	12
2.2.3 GitHub	13
2.3 Proyecto principal	14
2.3.1 Predicción de precios de la luz	14
2.3.1.1 Estudio y comprensión del mercado	14
2.3.1.2 Obtención y preparación de los datos	15
2.3.1.3 Análisis exploratorio	15
2.3.1.4 SARIMAX	17
2.3.1.5 Random Forest	20
2.3.1.6 XGBoost	21
2.3.1.7 TFT	24
2.3.1.8 Resultados	26
2.3.2 Modelo físico	28
2.3.2.1 Situación simplificada: Primera aproximación	28
2.3.2.2 Situación general	29
2.3.2.3 Traducción matemática	29
2.3.2.4 Implementación	30
2.3.3 Optimización de la superficie de enfriamiento	30
2.3.3.1 Tamaño de cajas	31
2.3.3.2 Chequeo de espacio	31
2.3.3.3 Distribución de superficie	31
2.3.3.4 Ejemplo de implementación	31
2.4 Relación entre formación recibida y actividades	33
2.5 Atención y asesoramiento recibido	33
3 Conclusiones	34

Índice de figuras

2.1	Formulario web: Página 1	9
2.2	Formulario web: Página 2	9
2.3	Formulario web: Página 3	10
2.4	Formulario web: Página 4	10
2.5	Segunda aplicación web con menú interactivo.	11
2.6	Fragmentos código entorno GCP.	12
2.7	Interfaz de GitHub en Visual Code	13
2.8	Figuras del análisis exploratorio.	16
2.9	ACF y PACF serie temporal original.	18
2.10	ACF y PACF serie temporal diferenciada $d = D = 1$	18
2.11	Predicción SARIMAX (2,1,1)x(1,1,1,24)	19
2.12	Predicción SARIMAX (1,1,1)x(1,1,1,24)	19
2.13	Predicción SARIMAX (2,1,0)x(1,1,0,24)	19
2.14	Predicción con Random Forest	21
2.15	Errores en predicción con Random Forest	21
2.16	Predicción con XGBoost	23
2.17	Errores en predicción con XGBoost	24
2.18	Arquitectura de TFT	25
2.20	Ejemplo distribución de superficie	32

Siglas y acrónimos

Como es habitual en documentos técnicos, a menudo se emplean siglas y acrónimos para abreviar términos específicos del área. Más aún, en el ámbito científico, el inglés predomina como lengua de trabajo por lo que muchos términos se encuentran en este idioma y la traducción al español no es siempre directa. A continuación se presenta una lista de los acrónimos más relevantes junto con su significado:

UC: Universidad de Cantabria

IA: Inteligencia artificial

TFT: Temporal Fusion Transformers (Transformadores de fusión temporal)

XGBoost: Extreme Gradient Boosting (Aumento extremo de gradiente)

SARIMAX: (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors) (Medias móviles integradas autorregresivas estacionales con regresores exógenos)

GCP: Google Cloud Platform (Plataforma Google Cloud)

TSF: Time series forecasting (Predicción de series temporales)

MAE: Mean absolute error (error absoluto promedio)

RMSE: Root mean squared error (raíz del error cuadrático promedio)

ACF: Autocorrelation function (función de autocorrelación)

PACF: Partial autocorrelation function (función de autocorrelación parcial)

API: Application Programming Interface (Interfaz de Programación de Aplicaciones)

PVPC: Precio de venta al pequeño consumidor

ESIOS: Estructura de Intercambio de Información del Operador del Sistema (Operador del Sistema Eléctrico Español)

Agradecimientos

Me gustaría agradecer a todas las personas de Evenbytes su amabilidad e integración, la plantilla de trabajo es como una gran familia y han hecho que este tiempo haya sido muy agradable. En particular, a Miguel y Pedro, quienes más cerca de mí han estado en ese inmejorable *equipo Data* y que me han enseñado día a día. Con ellos me he divertido, he aprendido y he cumplido uno de mis objetivos con estas prácticas, que era conocer de primera mano el mundo empresarial. Naturalmente, agradecer también a Jacinto Pelayo, CEO de la empresa, su confianza desde el primer momento para la incorporación a la empresa. Finalmente, a mi tutora académica, Alicia Nieto, por haberme ayudado en la organización y elaboración de esta memoria.

En último lugar, no puedo dejar pasar la oportunidad (ni me dejarían) de agradecer a mi familia su apoyo. Como es comprensible, el inicio en el mundo laboral no es sencillo y gracias a ellos me ha costado menos.

En resumen, estas prácticas no solo me han permitido aplicar conocimientos adquiridos a lo largo de estos cuatro años de formación, sino que también han sobrepasado las expectativas de conocer el mundo empresarial y me han proporcionado aquella visión que necesitaba para decidir mi futuro profesional.

Muchas gracias

Capítulo 1

Introducción

1.1. Elección

El proceso que llevó a la decisión de incorporarme a Evenbytes estuvo influenciado por diversos factores. En primer lugar, tuve la fortuna de que amigos míos, ya graduados de esta misma facultad, realizasen las prácticas en esta empresa, teniendo estos además diferentes perfiles, de manera que contaba de inicio con la ventaja de conocer la empresa de manera previa. Además, durante la feria de empleo realizada el 30 de abril de este mismo año pude asistir a los diferentes puestos que situaban las empresas asistentes para promocionarse y captar posibles futuros trabajadores. Dentro de estos grupos, Evenbytes fue la empresa que más llamó mi atención gracias a un proyecto que involucraba la implementación de predicciones estadísticas, modelado físico y optimización matemática. Fue por tanto la conjunción de un proyecto que aunase los conocimientos de las dos carreras en las que me estoy formando, con conocer de manera previa el buen ambiente de trabajo, la que hizo que me decantase por Evenbytes.

1.2. Objetivos

La decisión acerca del futuro laboral es una tarea complicada, y más aún cuando tu formación te otorga una plenitud de opciones. Además, me encontraba ante la indecisión de si continuar por el mundo académico o desligarme de lo *teórico* e insertarme en la aplicación de los conocimientos adquiridos. Ante esta situación, veía más que necesario tener un contacto directo con el mundo profesional para poder dilucidar cuál quería que fuera mi futuro. Entre las ramas de mi interés se encontraban principalmente el sector financiero y el tecnológico, ya que considero que ambas áreas ofrecen un campo de desarrollo muy amplio para los perfiles con formación en Física y Matemáticas.

Por tanto, buscar una empresa que encajase con estas características era indispensable. La mayor parte de las ofertas actuales, especialmente en ámbitos relacionados con el análisis de datos, la modelización estadística y el desarrollo de soluciones tecnológicas, se alinean con esta línea de interés. A la hora de seleccionar un centro que me aceptase, una de mis prioridades fue también el ambiente laboral, puesto que, en mi opinión, uno aprende más cuando se encuentra cómodo.

En este aspecto, Evenbytes me acogió de manera total, siendo un claro ejemplo de un entorno de

trabajo saludable en el que se valora la iniciativa, la capacidad de adaptación y la aportación personal al proyecto común. Considero que realizar las prácticas en una empresa con estas características me permitía no sólo adquirir experiencia práctica sino también contrastar en primera persona cómo se aplican competencias transversales como el pensamiento crítico, toma de decisiones en proyectos reales y la proactividad. Asimismo, esta oportunidad suponía un primer paso importante para explorar con mayor claridad qué caminos profesionales me gustaría recorrer en el futuro.

1.3. Centro

Evenbytes es una consultora tecnológica que impulsa la transformación digital de empresas mediante soluciones tecnológicas avanzadas. Con un recorrido de más de 10 años de experiencia, su objetivo es ayudar a las organizaciones a ser más eficientes, competitivas e innovadoras, adaptándose siempre a sus necesidades específicas.

Especializados en el desarrollo de software, infraestructuras en la nube y consultoría tecnológica, en Evenbytes se posiciona como un aliado estratégico para empresas de todos los tamaños, desde startups hasta grandes corporaciones. Son Google Cloud Partners y cuentan con experiencia consolidada en áreas como IA, Big Data e IoT, poniendo estas tecnologías al alcance de sus clientes para que puedan afrontar los retos de un entorno empresarial en constante evolución. Sus principales áreas de especialización incluyen el desarrollo de software personalizado, gestión de infraestructuras en la nube, automatización y optimización de procesos, Inteligencia artificial y Big Data e integración de sistemas y plataformas.

La empresa actualmente cuenta con 16 trabajadores y 4 estudiantes de prácticas. Entre los perfiles de los compañeros se encontraban principalmente el Grado universitario en Ingeniería Informática, Ingeniería de Telecomunicaciones, Grados superiores relacionados con la informática y Grado en Matemáticas. La actividad está dividida en áreas de gerencia, ventas, consultoría, desarrollo y ciencia de datos. Los 3 socios y principales responsables ocupan los cargos de CEO, CTO y COO.

1.4. Práctica Externa

Las labores realizadas durante el tiempo de prácticas han sido diversas. Como bien es sabido, la formación tanto en matemáticas como en física dotan de una gran capacidad de polivalencia a la persona que la posee, por lo que los trabajos que he realizado han sido diversos. Previamente se ha mencionado que Evenbytes es una empresa que se encuentra en plena expansión dentro del sector de la transformación digital. Por ello, en las semanas iniciales de las prácticas llevé a cabo labores de formación relacionadas principalmente con esta rama: Programación en Angular, diseño de páginas web, entorno GCP, GitHub, etc. Tras haber realizado dichas tareas me incorporé en el equipo de *Data Science*. En este departamento fui incluido en varios proyectos que se describirán a continuación, en los que he aprendido diversos programas relacionados con la ciencia de datos como *BigQuery*, *Jupyter*, *Python* ... además de técnicas para el tratado de datos y predicciones como *SARIMAX* o *redes neuronales*. Adicionalmente, debido a las características de uno de los proyectos me vi involucrado en el modelado físico de un proceso termodinámico, por lo que también he puesto en práctica los conocimientos adquiridos en el grado en Física.

Capítulo 2

Descripción de las prácticas

2.1. Actividades desempeñadas

Durante mi periodo de prácticas me he visto involucrado en diversos proyectos y dado que las actividades desempeñadas han sido variadas, no es del todo posible crear un *log frame* que las recoja todas de manera organizada en el sentido temporal. Además, dichos proyectos no se han iniciado y finalizado durante mi estancia por lo que no se puede definir de una manera feaciente las fases en las que se encontraban. Por tanto, se diseccionará la descripción de las prácticas en dos partes. En primer lugar, una asociada a la formación, en la que se explicarán los conceptos básicos aprendidos en las primeras semanas de trabajo. A continuación, se expondrá el proyecto al que mayor tiempo he dedicado.

2.2. Formación

Mis primeros pasos en el aprendizaje de las herramientas de trabajo de Evenbytes está dominado por: *Angular*, *entorno GCP* y *GitHub*. Estas tres herramientas forman el pilar básico de la empresa. Angular es el framework principal utilizado para el desarrollo de aplicaciones web. Como *partners* de Google, el entorno GCP proporciona la infraestructura en la nube preferida por la empresa y GitHub que es un sistema de control de versiones ampliamente extendido.

2.2.1. Angular

Angular es un framework de desarrollo basado en *JavaScript* y *HTML*¹ para la creación de aplicaciones web dinámicas, que permite estructurar el código de manera modular y facilita la generación de interfaces interactivas. Naturalmente, una empresa dedicada en parte a la transformación digital requiere de personal capaz de implementar páginas web o aplicaciones para que los clientes hagan uso de ellas. Como ejemplo, uno de los productos de Evenbytes son los dispositivos de fichaje horario, por lo que los datos que se registran en el momento en el que un trabajador entra a su jornada

¹Técnicamente *TypeScript* pero esencialmente es lo mismo.

laboral, además de ser tratados, se mostrarán para que la empresa empleadora pueda manejarlos, filtrarlos, etc.

El periodo dedicado al aprendizaje de este ítem fue inferior a una semana pero aun así el poder conocer el trabajo que supone la creación e implementación de páginas y aplicaciones fue satisfactorio y pude desarrollar ciertos ejemplos, como los formularios que se muestran a continuación:

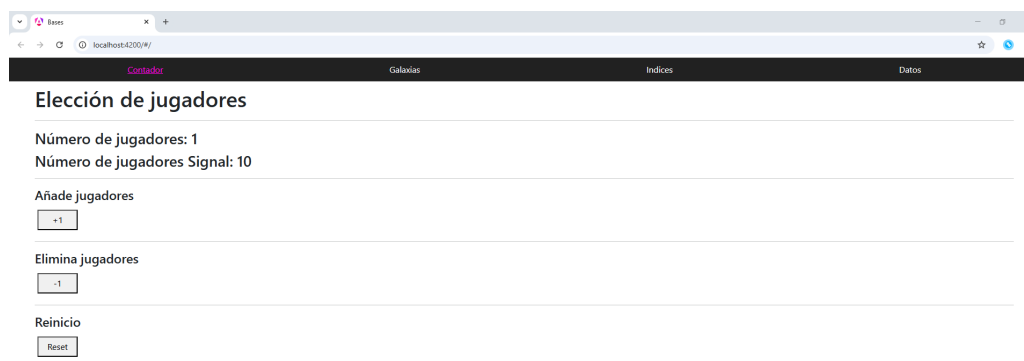


Figura 2.1: En este se pueden agregar y disminuir *jugadores* del entorno.

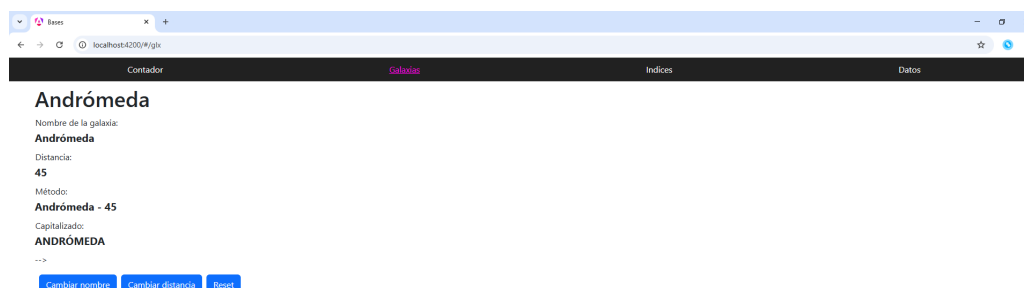


Figura 2.2: Formulario que permite variar los datos de una galaxia restaurarlos por los iniciales.

Figura 2.3: En este entorno se permite agregar elementos a la lista de manera general o condicionada.

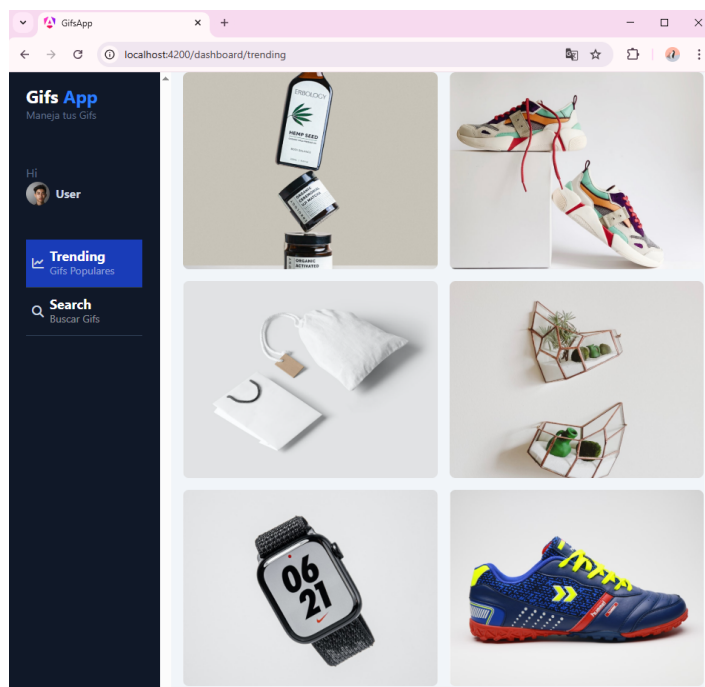
Figura 2.4: Formulario similar a la anterior pero en este caso permitía operar con los datos mostrándolos transformados.

En las figuras² se muestran capturas de la primera página desarrollada, en la que se tenían diversos formularios, con sus *URLs* específicas. Los datos, inventados naturalmente, hacían referencia al nombre de galaxias, distancia, origen e identificador. La funcionalidad de la página era la creación de formularios cuyos datos se agregaran a listas de manera general, condicionada (en este caso particular por el origen o la distancia). Además, se añadió la posibilidad de exponer los datos, cambiarlos o restaurarlos.

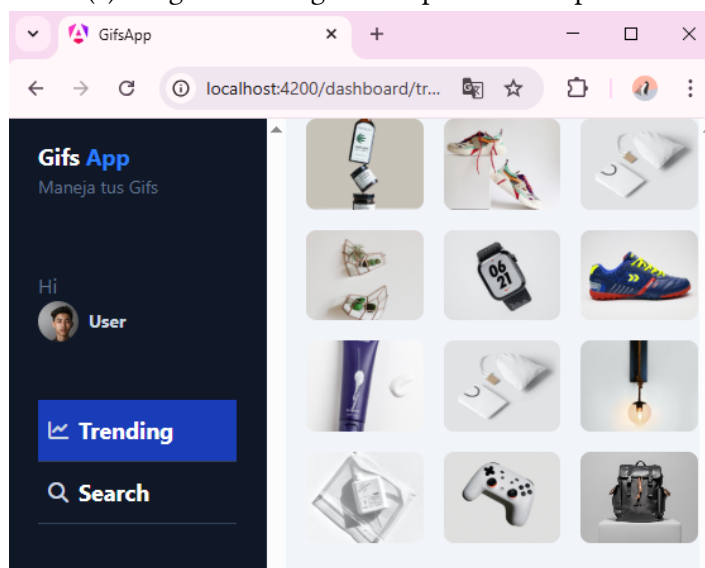
En segundo lugar, se procedió a la creación de una aplicación más elaborada, que tuviera un menú interactivo como capa principal y que registrara en él ciertos datos como el usuario o una foto de perfil.

²La primera de ellas se corresponde con una prueba en la que se analizaba el comportamiento de distintas formas de definir variables en Angular y no tiene relación con las siguientes.

El navegador se comportaba de manera reactiva, adaptándose al tamaño de la pantalla, tal y como se muestra en la Figura 2.5. A la izquierda de las imágenes puede apreciarse el menú lateral, en el que aparecen los ítems (en este caso, dos) que permiten acceder al resto de modalidades de la página.



(a) Imagen de navegador en pantalla completa.



(b) Imagen de navegador en pantalla reducida.

Figura 2.5: Segunda aplicación web con menú interactivo.

El repositorio de *gifs* incorporado en la aplicación se organizaba mediante una pestaña denominada *trending*, que permitía ordenarlos. En este caso, como es evidente, no se podía disponer de un registro de las consultas de los usuarios, por lo que las entradas se organizaban de manera alfabética.

2.2.2. GCP

Este entorno proporciona la infraestructura en la nube necesaria para desplegar aplicaciones, almacenar datos y ejecutar servicios de manera flexible y segura. La empresa Evenbytes es, como ya se ha mencionado, *partner* de Google, por lo que brinda la posibilidad de trabajar con la mayor parte de funcionalidades de esta plataforma. Trabajar en la nube facilita el acceso compartido, la escalabilidad de los recursos y la automatización de procesos, lo que resulta especialmente valioso en proyectos donde intervienen varios miembros del equipo o que requieren alta disponibilidad.

En mi caso particular, implementé una *Cloud Function* encargada de realizar el proceso de extracción de información de una página web (llamado *web scraping*) y la generación de un fichero con los resultados. La *Cloud Function* permite ejecutar fragmentos de código en respuesta a eventos o invocaciones programadas, sin necesidad accionar explícitamente el programa. Desde Python realicé el código para conectarse a la página de Wikipedia sobre cocientes intelectuales por país, extraer las tablas relevantes y convertir los datos en un archivo CSV. El código de la función se desplegó en GCP mediante las herramientas de línea de comandos y quedó configurado para ejecutarse de forma automática a intervalos regulares.

Para esta última tarea hice uso de Cloud Scheduler, un servicio que permite programar tareas basadas en cron. La configuración del Scheduler se gestionó a través de Terraform. Gracias a esta combinación, cada ejecución de la función generaba automáticamente un nuevo bucket³ de almacenamiento en Cloud Storage, en el que se guardaba el fichero CSV con los datos actualizados.

A continuación se muestran fragmentos de código de la programación del *web scraping*, la planificación en Terraform y el resultado del CSV.

```
13 def web_scraping():
14     # Extraer los datos
15     filas = driver.find_elements(By.XPATH, '//*[@class = "wikitable sortable jquery-tablesorter"]//tbody/tr')
16     print(f"Numero de filas encontradas: {len(filas)}")
17     data = []
18
19     print("Comenzamos a buscar la información de las filas...")
20     # Las recorremos
21     for i, fila in enumerate(filas):
22         try:
23             # Tomo el elemento de la imagen
24             elemento_imagen = fila.find_element(By.XPATH, '//*[@img')
25             # Voy a capturar ahora el nombre del país
26             texto_completo = elemento_imagen.get_attribute('alt')
27             nombre = texto_completo.replace("Bandera de ", "")
28
29             # Ahora extraer los IQ
30             td_completos = fila.find_elements(By.TAG_NAME, 'td')
31             # IQ en tercera columna por lo que lo printeo directamente ahí
32             if len(td_completos) > 2:
33                 in = td_completos[2].text
34                 posicion = td_completos[0].text
35                 data.append({
36                     'Position': posicion,
37                     'NombrePaís': nombre,
38                     'IQ': in,
39                 })
40             else:
41                 print(f"La fila {i+1} no tiene suficientes columnas")
42         except Exception as error:
43             print(f"Error al procesar la fila {i+1}: {error}")
44
45     if driver:
46         driver.quit()
47     driver = None
48     print("Webdriver cerrado tras el scraping.")
49
50     # Creo el dataframe en pandas
51     df = pd.DataFrame(data)
52
53     if df.empty:
54         print("No se encontraron datos para exportar.")
55     # Asegurado: Retorno explícito para este caso
56     return "No se encontraron datos para exportar. El scraping pudo fallar.", 200
57
58     # Lo exporto a csv
59     csv_filename = 'iq_paises.csv'
60     csv_path = os.path.join('/tmp', csv_filename)
61     df.to_csv(csv_path, index=False, encoding='utf-8')
62     print(f"DataFrame guardado localmente en: {csv_path}")
```

(a) Fragmento del código de *web scraping* en Python.

```
1 provider "google" {
2   project = var.gcp_project_id
3   region  = var.gcp_region
4 }
5
6 # Obtener información del servicio de Cloud Run existente
7 data "google_cloud_run_service" "my_cloud_run_service" {
8   name     = var.cloud_run_service_name
9   location = var.gcp_region
10 }
11
12 # Crear el trabajo de Cloud Scheduler
13 resource "google_cloud_scheduler_job" "my_scheduler_job" {
14   name           = "invoke-iq-paises-alberto-daily"
15   description    = "Invoca el servicio de Cloud Run IQ Países diariamente"
16   schedule       = "0 9 * * 2,4"
17
18   http_target {
19     uri      = "https://iq-paises-998937693176.europe-west1.run.app"
20     http_method = "POST"
21   }
22
23   headers = {
24     "Content-Type" = "application/json"
25   }
26
27   oidc_token {
28     service_account_email = var.scheduler_service_account_email
29   }
30
31   time_zone = "Europe/Madrid"
32 }
```

(b) Ejemplo de configuración de Terraform para el *scheduler*.

Figura 2.6: Fragmentos código entorno GCP.

³Los buckets son contenedores que permiten almacenar objetos de manera segura y escalable. En este proyecto, cada bucket contenía los resultados de la extracción en formato CSV, de forma que quedaba un histórico de ejecuciones accesible a través de la consola de GCP o mediante las APIs correspondientes.

A modo de breve explicación de los códigos mostrados, para el caso del *web scraping* la idea principal es generar un navegador virtual al que se le manda a la página de la que queremos extraer los datos. De manera previa, se investiga el código HTML para saber la estructura de la página, de modo que creamos un bucle que busque los elementos que queremos y extraiga la información de ellos, es decir, si tenemos una estructura en la que todos los objetos tienen una propiedad llamada *nombre*, hacemos que nuestro código acceda a él para extraer todos los nombres de la página y guardarlos en un fichero.

Por otro lado, el scheduler es más sencillo, le introducimos los datos del proyecto, función a llamar... y definimos las órdenes bajo las que se ejecuta. En el código aparece en la línea 16 el fragmento `0 9 * * 2,4` que significa que se ejecute a las 9:00, *cualquier día* (numérico), *cualquier mes*, los *martes* y *jueves*.

2.2.3. GitHub

En último lugar estuvo el aprendizaje de GitHub, que actúa como sistema de control de versiones, permitiendo gestionar el historial de cambios en el código, coordinar el trabajo y asegurar que todas las contribuciones queden registradas de forma ordenada. Su uso es fundamental en un entorno profesional, ya que permite mantener la trazabilidad de los desarrollos, revisar modificaciones y garantizar la integridad de los proyectos a medida que evolucionan. Su estructura es muy sencilla y puede entenderse rápidamente pensando en grafos. Se crea una rama principal del proyecto y de ellas, cuando se quiere trabajar en una parte específica del mismo, salen hijas. De esta manera los cambios realizados se hacen fuera del proyecto principal, permitiendo guardar los cambios, avanzar únicamente en esa línea o abrir nuevas. Cuando una parte se da por revisada y acabada se unen ramas a la principal (padre), de manera que se va completando el trabajo. Los comandos básicos de Git son *commit*, que guarda los cambios realizados en el repositorio local; *pull*, que actualiza la copia local con las modificaciones del repositorio remoto; *push*, que sube los cambios locales al repositorio compartido; y *merge*, que combina diferentes ramas en una sola versión final.

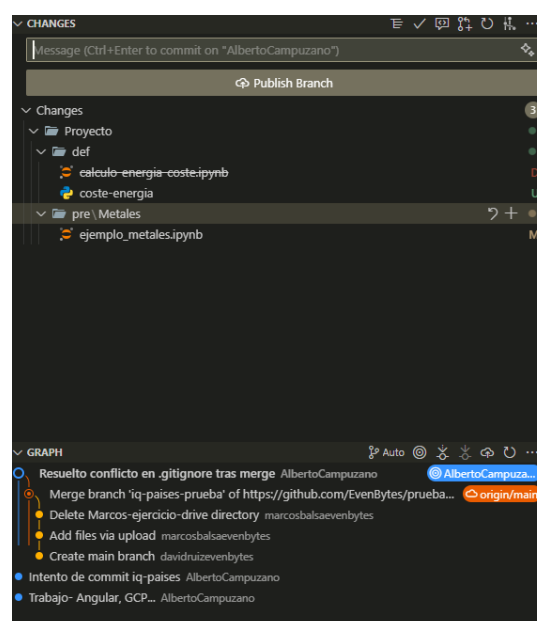


Figura 2.7: Imagen GitHub.

2.3. Proyecto principal

Este trabajo trataba de la optimización del gasto energético de una empresa que contaba con hornos de fundición. Para ello, se tuvieron que realizar predicciones del precio de la electricidad mediante modelos estadísticos. Posteriormente, se modelizó el proceso de fundición llevado a cabo en la fábrica y por último se trató de optimizar la selección de los materiales a fundir, dependiendo del espacio disponible en fábrica y el precio de la luz en cada tramo horario, estimado previamente.

2.3.1. Predicción de precios de la luz

Para llevar a cabo un modelo predictivo de los precios de la luz para una empresa hemos de realizar una serie de pasos. En primer lugar, como es evidente, debemos conocer y entender qué es lo que queremos predecir, bajo qué condiciones y qué variables afectan al precio de la luz, tanto naturales, como puede ser la demanda, la generación de ciertos tipos de energía, como aquellas relacionadas con la legislación vigente. Posteriormente debemos decidir cómo se extraeran los datos necesarios. Puesto que estamos hablando de un proyecto real, la empresa espera un proceso automatizado por lo que se debe tener en cuenta una extracción periódica de datos.

Una vez planteado el contexto en el que vamos a movernos, debemos decidir qué modelos probar, conociendo sus ventajas y desventajas. Mi elección, basada en búsquedas de desarrollos similares, fue optar por tres modelos:

SARIMAX: Modelo estadístico clásico.

Random Forest: Modelo de *machine learning* basado en árboles de decisión.

XGBoost: Modelo de *machine learning* basado en árboles de decisión, más avanzado que el anterior.

TFT: Modelo de *deep learning*, específico para series temporales.

Por último, por requisitos del proyecto, la predicción se realizaba a un mes vista, lo cual como es lógico, no es óptimo. Sin embargo, esto era algo fijo por lo que los resultados esperados son mejorables, recortando el horizonte de predicción y reentrenado los modelos con mayor asiduidad.

2.3.1.1. Estudio y comprensión del mercado

Dado que la empresa es una electrointensiva no se rige por el régimen general del pequeño consumidor (PVPC). Para la toma de datos de la variable objetivo, el precio de la luz, se consultó el precio del mercado SPOT diario, que es el que regula la venta de luz al por mayor.

Para calcular exactamente, dentro de lo que podemos entender como *exacto* en un modelo de predicción, el precio de la luz debemos tener en cuenta tanto el precio de venta como los peajes. Estos últimos son valor añadido a la factura de la luz que depende del tipo de empresa y de la hora de consumo. De cualquier forma, en lo que a nuestro modelo se refiere, únicamente nos interesa el precio del mercado SPOT diario, que viene dado de manera horaria.

2.3.1.2. Obtención y preparación de los datos

Los datos se obtuvieron a través de la API pública proporcionada por ESIOS (Sistema de Información del Operador del Sistema Eléctrico), que permite acceder a los precios horarios históricos, así como a otros indicadores del sistema eléctrico. De manera adicional al precio de la luz, y como se ha comentado previamente, se tomaron variables explicativas, es decir, que se cree que tienen relación y afectan al precio de la que queremos. En este caso, se tomó: La demanda real de consumo, la generación de energía eólica y solar.

En lo relativo a la preparación de los datos, salvando las diferencias en función de la implementación del modelo, el procedimiento fue similar: Una vez descargados los datos se formatean de manera que tengamos aquellos valores que queremos, es decir, el precio, demanda, etc. y la fecha en la que se dio. Además, para refinar nuestro modelo, se introdujeron variables temporales como el día de la semana, día del mes o mes del año. Un estudio mucho más exhaustivo puede hacerse incluyendo más variables o con más combinaciones. En [2] puede verse cómo incluyen el precio del gas o la emisión de CO₂ entre otros, llevándoles a mejores resultados, aunque con mayor trabajo.

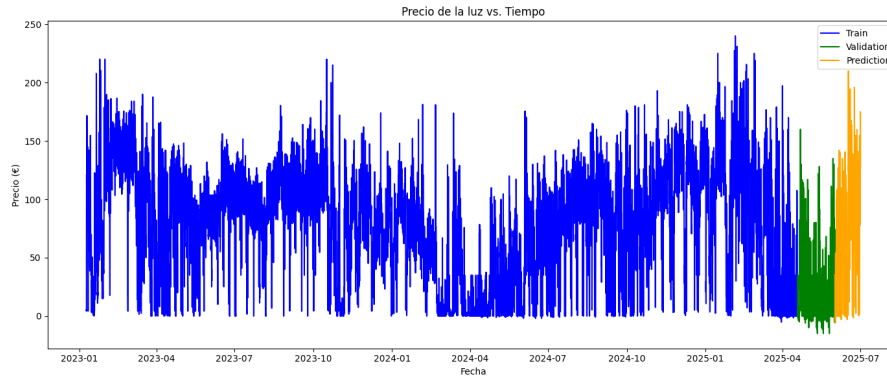
2.3.1.3. Análisis exploratorio

Este proceso consiste en la observación de los datos, su representación en diferentes formas con el fin de decidir características del modelo. El estudio consistió en graficar respecto del tiempo el histórico de datos, implementar una matriz de correlación para observar las relaciones entre las distintas variables y boxplot para estudiar el comportamiento de nuestra serie.

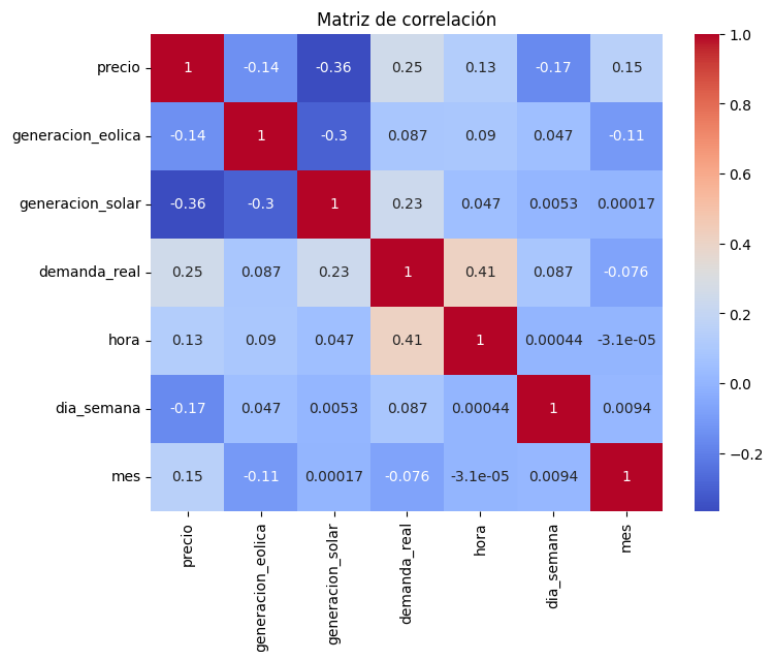
A partir de los datos obtenidos con las gráficas, como se puede observar en la figura figura 2.8a, se divide el histórico de datos en un conjunto de entrenamiento, otro de test y se deja uno de validación, en este caso de un mes, al igual que la predicción que queremos realizar, para comprobar el funcionamiento *real* de nuestro modelo. Este proceso es el que se ha utilizado en la SARIMAX, Random Forest y el XGBoost.

Observando la matriz de correlación (figura 2.8b) se aprecia que la variable con mayor correlación negativa con el precio es la generación solar ($r \approx -0,38$), lo que sugiere que un aumento en la producción solar tiende a asociarse con una disminución en el precio. La demanda real presenta una correlación positiva moderada ($r \approx 0,25$) con el precio, mientras que la generación eólica muestra una relación más débil y ligeramente negativa. Las variables temporales (hora, día de la semana, mes) tienen correlaciones bajas, lo que indica que, si bien influyen en el precio, lo hacen de forma más indirecta o no lineal.

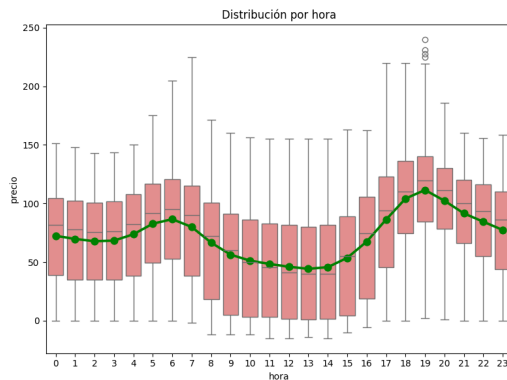
En cuanto a los boxplots, puede observarse cómo en el ámbito anual (figura figura 2.8d) los meses de abril y mayo concentran los precios más bajos, mientras que en febrero se registran picos más altos. También se aprecia que la variabilidad del precio es mayor en los meses de invierno y menor en primavera. En el análisis diario (figura figura 2.8c), se identifican claramente las denominadas *horas valle*, comprendidas aproximadamente entre las 10:00 y las 15:00, y las *horas pico*, situadas en el tramo de 18:00 a 20:00.



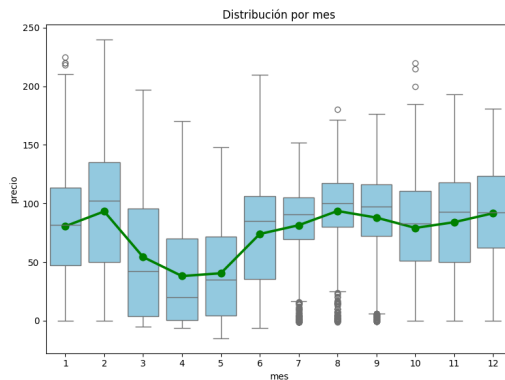
(a) Precio de la luz respecto del tiempo, en azul se ve el conjunto de entrenamiento (*train test*), en verde el conjunto de test (*test test*) y en amarillo el conjunto de validación (*validation test*).



(b) Matriz de correlación.



(c) Boxplot diario.



(d) Boxplot anual.

Figura 2.8: Figuras del análisis exploratorio.

2.3.1.4. SARIMAX

Este es un modelo estadístico clásico para la predicción de series temporales. Su fortaleza radica en su capacidad para modelar explícitamente tres componentes clave de la serie:

- **Tendencia (I - Integrado):** Se logra a través de la diferenciación para hacer que la serie sea estacionaria.
- **Estacionalidad (S - Seasonal):** Captura patrones que se repiten en intervalos fijos (por ejemplo, diarios o semanales).
- **Variables (X):** Permite incluir predictores externos, como la demanda o la generación de energía.

El proceso consiste en:

- **Identificación:** El modelo SARIMAX viene caracterizado por los valores $(p, d, q)x(P, D, Q, s)$ donde:
 - p : Orden del componente autorregresivo (AR) no estacional. Indica el número de rezagos de la variable endógena que se incluyen en el modelo.
 - d : Orden de diferenciación no estacional. Representa la cantidad de veces que se deben diferenciar los datos para hacer la serie estacionaria.
 - q : Orden del componente de media móvil (MA) no estacional. Indica el número de rezagos del error del pronóstico que se incluyen en el modelo.
 - P : Orden del componente autorregresivo (AR) estacional.
 - D : Orden de diferenciación estacional.
 - Q : Orden del componente de media móvil (MA) estacional.
 - s : Período de estacionalidad, en este caso vemos que $s = 24$ de manera natural al tratarse de precios horarios.
- **Estimación:** Una vez definidos los órdenes, se ajusta el modelo al conjunto de datos de entrenamiento. El modelo aprende los coeficientes que ponderan la importancia de los valores pasados, los errores de pronóstico pasados y las variables exógenas.

La predicción con SARIMAX es un proceso iterativo o recursivo:

- **Entrenamiento:** Al igual que con otros modelos, se reentrena un modelo SARIMAX final con su orden óptimo utilizando todos los datos históricos disponibles para maximizar la información aprendida.
- **Predicción:** Para predecir el horizonte futuro (ej. las próximas 24 horas), el modelo:
 - Predice el primer paso ($t + 1$) utilizando todos los datos históricos reales.
 - Para predecir el segundo paso ($t + 2$), utiliza los datos históricos y la predicción que acaba de hacer para $t+1$ como si fuera un dato real.

Este proceso se repite para cada punto en el horizonte de predicción.

En nuestro caso, para determinar los órdenes de los hiperparámetros del modelo, se analizan las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF). La ACF mide la correlación de la serie consigo misma en diferentes rezagos, mientras que la PACF mide esta misma correlación eliminando la influencia de los rezagos intermedios.

En las siguientes imágenes podemos ver los gráficos de ACF y PACF para la serie de precios de la luz, lo que nos permite identificar los patrones de estacionalidad y los órdenes de los modelos.

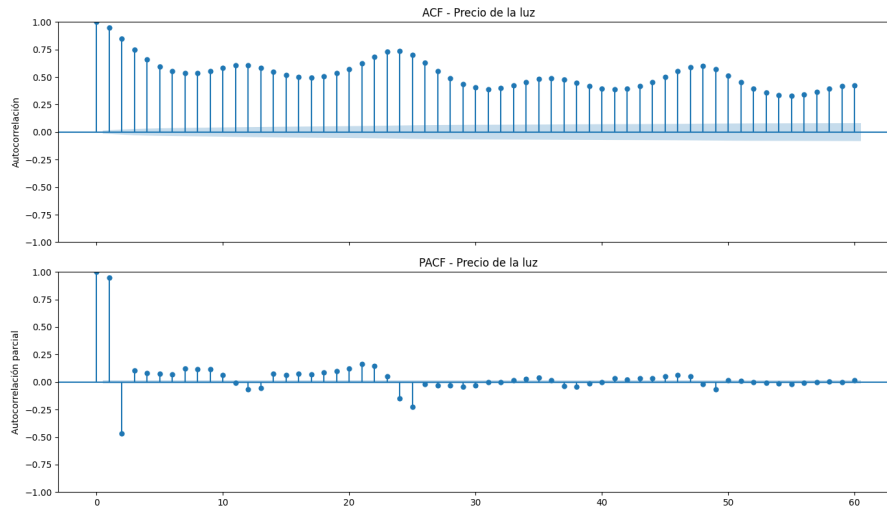


Figura 2.9: Podemos observar el ACF y el PACF de la serie temporal original, sin diferenciación. En ellos podemos ver que hay una clara estacionalidad alrededor del lag 24, lo que concuerda con el pensamiento de una periodicidad diaria, cada 24 horas. Además, que no haya un decaimiento fuerte hace pensar que requiera diferenciación.

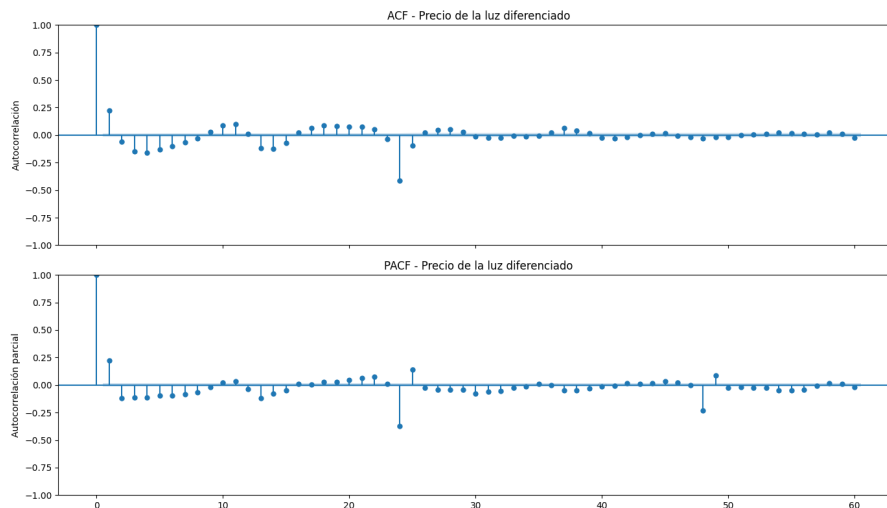


Figura 2.10: De nuevo, es visible la periodicidad en los picos con estacionalidad 24, sin embargo, esta vez se aprecia la caída en los picos, lo que sugiere haber encontrado ya la estacionariedad, además de que esto nos ayuda a determinar los posibles órdenes, como $p, P, Q, q \in \{0, 1, 2\}$.

Basado en el análisis de las funciones de autocorrelación y en una herramienta, *pmdarima*, que

permite probar y obtener los órdenes óptimos, se han seleccionado los siguientes tres modelos, donde se muestran las gráficas de predicción obtenidas:

■ **Modelo 1: SARIMAX(2,1,1)x(1,1,1,24):**

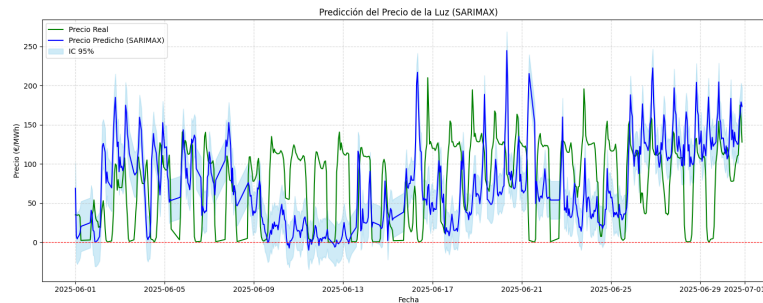


Figura 2.11: Predicción para junio con el SARIMAX de orden (2,1,1)x(1,1,1,24).

■ **Modelo 2: SARIMAX(1,1,1)x(1,1,1,24):**

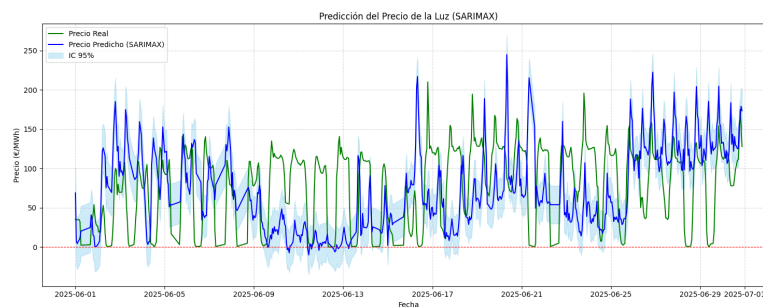


Figura 2.12: Predicción para junio con el SARIMAX de orden (1,1,1)x(1,1,1,24).

■ **Modelo 3: SARIMAX(2,1,0)x(1,1,0,24):**

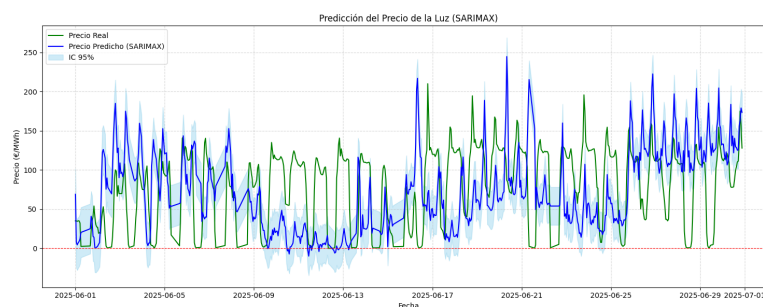


Figura 2.13: Predicción para junio con el SARIMAX de orden (2,1,0)x(1,1,0,24).

Puede apreciarse a simple vista que para un horizonte de predicción de un mes este método no es fiable. El modelo SARIMAX supone relaciones lineales y la volatilidad y complejidad del precio de la luz, además en un intervalo de tiempo tan grande como un mes no reporta predicciones satisfactorias.

2.3.1.5. Random Forest

Este modelo, también basado en árboles de decisión, destaca por su sencillez de implementación y su gran versatilidad para distintos tipos de problemas, aunque, como ocurre habitualmente en modelos de este tipo, presenta una menor interpretabilidad sobre el funcionamiento interno. Su fortaleza radica en el uso de un enfoque de ensamblado (*ensemble*) mediante *bagging*, que combina múltiples árboles de decisión entrenados de manera independiente a partir de bootstrap y predictores. Este procedimiento permite reducir la varianza del modelo y mejorar la capacidad de generalización, incluso sin requerir un ajuste tan fino como otros métodos más complejos. Entre las características más relevantes de Random Forest:

- **Robustez frente al *overfitting*:** El uso de múltiples árboles entrenados con subconjuntos aleatorios ayuda a evitar que el modelo memorice el conjunto de entrenamiento, reduciendo el riesgo de sobreajuste.
- **Capacidad de manejo de datos ruidosos:** El promedio (en regresión) o el voto mayoritario (en clasificación) suaviza el impacto de observaciones atípicas.
- **Manejo de variables de distinta naturaleza:** Puede trabajar tanto con variables numéricas como categóricas sin necesidad de una normalización previa.
- **Importancia de variables:** Aunque la interpretación global es limitada, proporciona métricas de importancia relativa para cada predictor, lo que facilita la identificación de los más influyentes.

En nuestro caso, la implementación con la librería *scikit-learn* permite optimizar el modelo mediante herramientas como *TimeSeriesSplit* y *GridSearchCV*, especialmente útiles en problemas de series temporales. De nuevo, nos enfrentamos a dos retos principales: la selección de hiperparámetros y la evaluación de errores.

Para la evaluación, *TimeSeriesSplit* divide el conjunto de entrenamiento en particiones cronológicas, evitando la ruptura de la secuencia temporal y permitiendo calcular métricas como MAE, RMSE o R^2 de forma más realista.

En cuanto a la búsqueda de los hiperparámetros óptimos (*hyperparameter tuning*), *GridSearchCV* prueba combinaciones dentro de los rangos definidos y nos devuelve la configuración con mejor rendimiento según una métrica objetivo, como el RMSE. En las siguientes gráficas mostramos los resultados obtenidos en uno de los entrenamientos junto a sus errores:

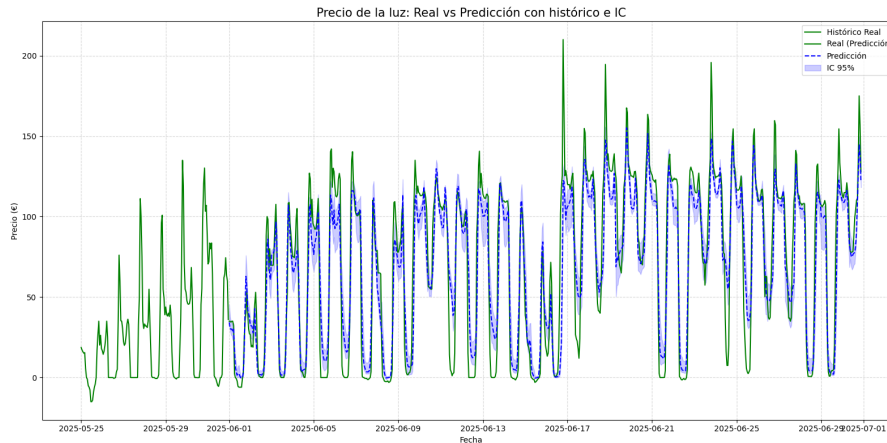


Figura 2.14: Predicción realizada para el mes de junio de 2025 con un entrenamiento sobre el histórico completo de datos.

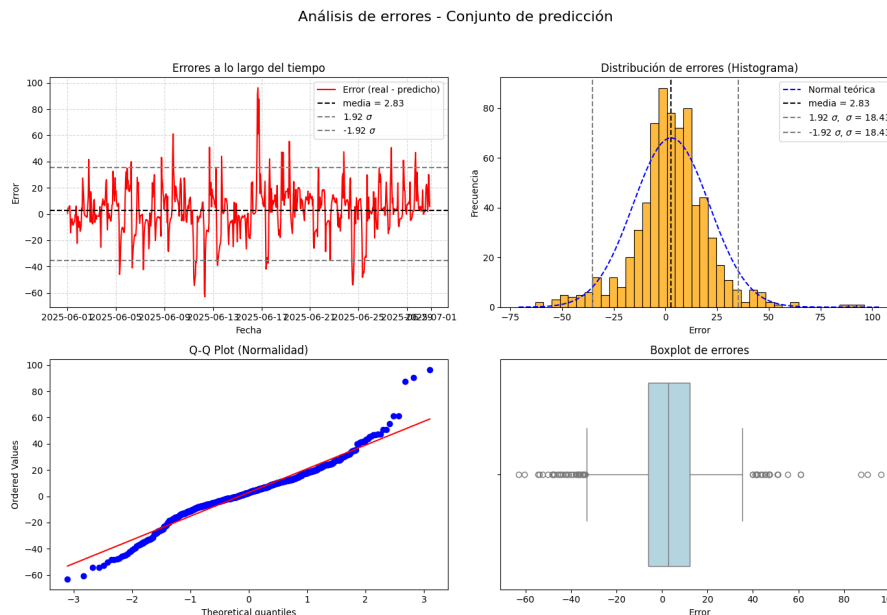


Figura 2.15: Errores del modelo Random Forest, donde los QQ-plots y el histograma muestran una distribución simétrica, sin colas pesadas pero no completamente normal.

2.3.1.6. XGBoost

Este modelo, basado en árboles de decisión, también es relativamente sencillo de implementar aunque, como es habitual en este tipo de modelos, presenta menor interpretabilidad respecto a lo que ocurre dentro del algoritmo. Su robustez radica en la capacidad de aprender patrones complejos mediante un enfoque de ensamblado (*ensemble*) y mejora por aproximación secuencial de errores. Además, se ve reforzado por técnicas de regularización y control del *overfitting*, lo que permite obtener modelos generalizables y eficientes en predicción. Entre las características más destacadas de XGBoost:

- **Regularización integrada:** A diferencia de otros modelos basados en árboles, XGBoost incluye regularización L_1 y L_2 como parte de la función de coste, permitiendo controlar la complejidad

del modelo y reducir el riesgo de *overfitting*.

- **Manejo de valores nulos:** Tiene un tratamiento nativo para valores faltantes, aprendiendo automáticamente la mejor dirección en los nodos de decisión.
- **Velocidad y eficiencia:** Permite la construcción de árboles de manera paralela. Además, utiliza una aproximación de segundo orden (Taylor) sobre la función de pérdida, incorporando tanto el gradiente como el hessiano. Esto no solo acelera la convergencia del modelo, sino que también permite trabajar con funciones de pérdida más generales, si bien lo más común es trabajar con el error cuadrático medio.
- **Importancia de variables:** Aunque la interpretabilidad del modelo global es limitada, permite extraer métricas de importancia relativa de cada predictor.

El entrenamiento se realiza mediante un enfoque de *boosting*:

- **Inicialización:** Se parte de una predicción base (por ejemplo, la media de la variable a predecir).
- **Aprendizaje secuencial:** Cada árbol posterior intenta corregir los errores residuales cometidos por el conjunto acumulado hasta el momento.
- **Optimización de la función de pérdida:** Se utiliza una aproximación de segundo orden (gradiente y hessiano) para mejorar la precisión de la optimización.
- **Regularización:** Durante el proceso se penaliza la complejidad del modelo (profundidad de los árboles, número de nodos) para evitar *overfitting*.

La predicción con XGBoost, al igual que en modelos como SARIMAX, se lleva a cabo de forma secuencial.

Aunque la implementación de este modelo puede parecer más sencilla, es importante comprender las distintas configuraciones que se pueden introducir. La librería *scikit-learn* proporciona herramientas muy útiles para optimizar modelos como XGBoost, en particular *TimeSeriesSplit* y *GridSearchCV*. En el proceso de entrenamiento, nos enfrentamos a dos desafíos principales: la selección de los hiperparámetros óptimos y la evaluación de los errores.

Para la segunda problemática, *TimeSeriesSplit* nos permite dividir nuestro conjunto de entrenamiento en particiones cronológicas, lo que es esencial para las series de tiempo, ya que una división aleatoria rompería la secuencia temporal. De esta manera, podemos obtener múltiples métricas de error (como MAE, RMSE y R^2) a lo largo del conjunto de datos y promediarlas para una evaluación más fiable.

Para la elección de los hiperparámetros del modelo, conocida como *hyperparameter tuning*, utilizamos *GridSearchCV*. Esta herramienta prueba sistemáticamente diferentes combinaciones de hiperparámetros dentro de los rangos que le proporcionamos, devolviéndonos la configuración que produce el mejor rendimiento. En esencia, y manera un poco más informal, prueba todas las configuraciones en un rango y nos dice cuál es la mejor en función de uno de los parámetros como puede ser el RMSE.

A continuación, se muestra un fragmento de código que implementa la validación cruzada temporal y la búsqueda de hiperparámetros con *GridSearchCV*.

```

1 param_grid = {
2     'n_estimators': [300, 325, 350, 375, 400, 425, 450],
3     'learning_rate': [0.02, 0.03, 0.04, 0.05, 0.06, 0.07],
4     'max_depth': [2, 3, 4, 5]
5 }
6
7 tscv = TimeSeriesSplit(n_splits=5)
8
9 xgb = XGBRegressor(random_state=42, reg_alpha=5, reg_lambda=1)
10
11 grid_search = GridSearchCV(
12     estimator=xgb,
13     param_grid=param_grid,
14     scoring='neg_mean_absolute_error',
15     cv=tscv,
16     n_jobs=-1,
17     verbose=2
18 )
19
20 print("-----")
21 print("Iniciando búsqueda de hiperparámetros con GridSearchCV...")
22 print("-----")
23 grid_search.fit(X_train, Y_train)
24 print("<----- VALORES DE LOS ERRORES ----->")
25 print(f"Mejores hiperparámetros encontrados: {grid_search.best_params_}")
26 print(f"Mejor MAE con validación cruzada: {-grid_search.best_score_:.2f}")
27 print("-----")
28 print(f"R²: {r2_score(Y_test, Y_test_pred):.4f}")

```

Listing 2.1: Búsqueda óptima de hiperparámetros y validación cruzada en el modelo XGBoost.

En el código anterior, los hiperparámetros que se optimizan son *n_estimators*, *learning_rate* y *max_depth*.

- **n_estimators**: Representa el número de árboles de decisión que el modelo construye de forma secuencial. Un mayor número de estimadores puede mejorar la capacidad predictiva del modelo, pero también aumenta el riesgo de *overfitting* y el tiempo de cálculo.
- **learning_rate**: Es un factor de ponderación que controla el peso que tiene cada nuevo árbol de decisión en la predicción final. Un valor más bajo hace que el modelo aprenda de forma más gradual, reduciendo el riesgo de *overfitting*.
- **max_depth**: Limita la profundidad máxima de cada árbol de decisión. Es un hiperparámetro crucial para controlar la complejidad del modelo. Una profundidad más pequeña ayuda a prevenir el *overfitting* al simplificar los árboles.

En las siguientes gráficas mostramos los resultados obtenidos en uno de los entrenamientos junto a sus errores:

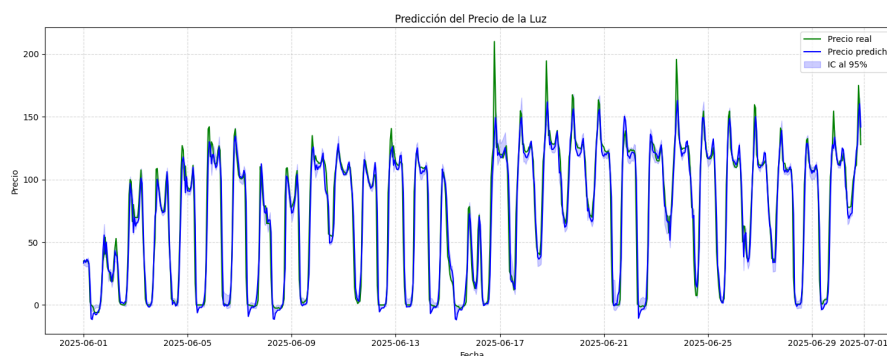


Figura 2.16: Se observa una predicción realizada para el mes de junio de 2025 con un entrenamiento del histórico de datos completos.

Análisis de errores - Conjunto de predicción

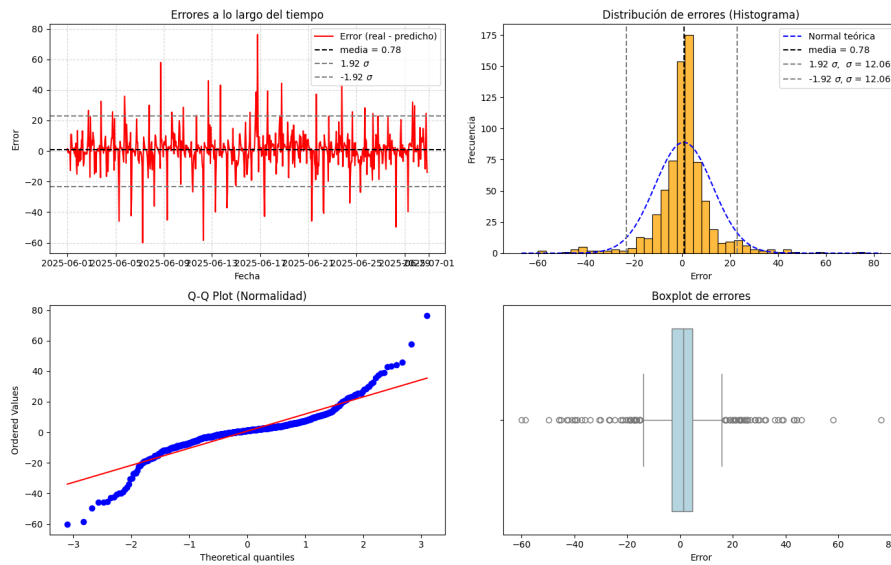


Figura 2.17: Errores del modelo XGBoost, en el que destacan los QQ-plots y el histograma para caracterizar el comportamiento de los mismos. Puede verse en ellos que se distribuyen de forma simétrica, sin colas pesadas aunque no completamente normal.

2.3.1.7. TFT

El Temporal Fusion Transformer (TFT) es una arquitectura de deep learning diseñada específicamente para el forecasting de series temporales multi-horizonte. Combina mecanismos de atención (propios de los Transformers) con redes neuronales recurrentes y capas de gating. Sus características clave son:

- **Manejo de Múltiples Tipos de Variables:** Integra de forma nativa variables estáticas, entradas conocidas en el futuro y variables observadas en el pasado (ej. precio, demanda).
- **Aprendizaje de Patrones a Múltiples Escalas:** Puede identificar patrones temporales tanto a corto como a largo plazo simultáneamente.

El entrenamiento consiste en:

- **Preparación de los Datos:** Los datos se estructuran en "ventanas" de entrada de una longitud fija para predecir una "ventana" de salida (ej. las próximas 24 horas).
- **Ajuste del Modelo:** Se entrena una única red neuronal que aprende las interacciones entre todas las variables a lo largo del tiempo.

La predicción con TFT se realiza en un solo paso:

- **Reentrenamiento:** Se ajusta el modelo final con todos los datos históricos disponibles.

- **Predicción Multi-Horizonte Directa:** Para generar la predicción, se le proporciona al modelo la última ventana de datos históricos disponibles. En una única pasada hacia adelante (forward pass), el modelo genera directamente todas las predicciones para el horizonte completo.

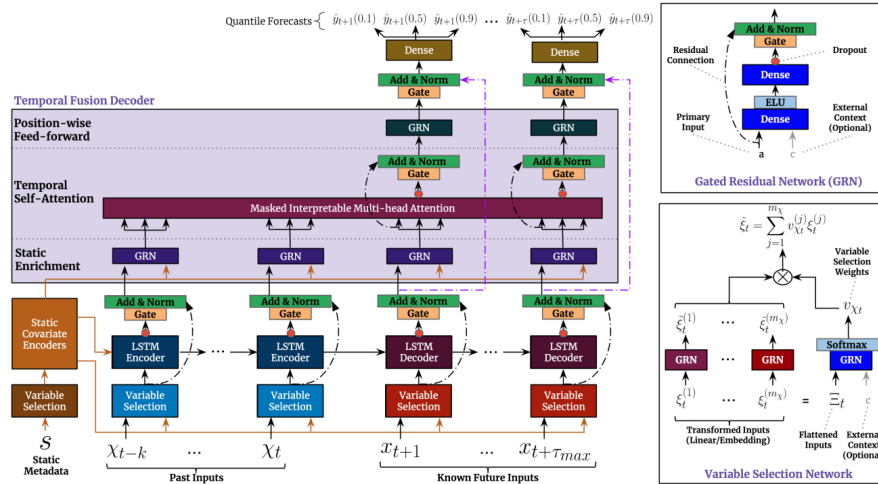


Figura 2.18: Arquitectura del TFT [3].

Respecto a la decisión de hiperparámetros de nuestro modelo TFT es más complicada. Este tipo de modelos son más exigentes en el formato de entrada de los datos y más sensible a las configuraciones, si bien lo compensa con un gran rendimiento. Por ello no me adentraré mucho en ello:

- **learning rate:** Este hiperparámetro, como se ha mencionado en el XGBoost, controla el tamaño del paso en cada iteración. Valores más altos aceleran la convergencia pero puede reportar resultados peores a costa de ellos. Por otro lado, un valor extremadamente bajo hace que la convergencia sea más lenta y puede dar lugar a que en el proceso de optimización de la función de pérdida encuentre un mínimo local y no obtenga el resultado óptimo.
- **input chunk length:** Define el número de pasos de tiempo pasados que el modelo utiliza como entrada para hacer una predicción. En nuestro caso, hicimos uso de tres meses de datos, es decir, $24 \times 7 \times 4 \times 3$ pasos de tiempo.
- **output chunk length:** Determina el número de pasos de tiempo futuros que el modelo debe predecir. Como se mencionó anteriormente, la predicción se realizó para un mes, lo que significa $24 \times 7 \times 4$ pasos.
- **hidden size:** Se refiere a la dimensión de las capas ocultas que forman la red. En este caso, un valor demasiado alto puede elevar la complejidad del modelo, llevando a *overfitting*.
- **attention heads:** Define el número de capas de atención del modelo, es decir, nos permite regular la *atención* que el TFT va a prestar a los diferentes componentes de la serie temporal de manera simultánea. Este hiperparámetro es clave para capturar las relación a largo plazo en nuestra serie.
- **dropout:** Para este tipo de modelos, un valor entre 0.1-0.3 suele ser lo estándar. Este hiperparámetro es fundamental para la regularización, previniendo el *overfitting*. Consiste en desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento.

En nuestro caso particular, muestro a continuación parte del código utilizado en el que se ve cómo se refina la configuración:

```

1  study = optimize_hyperparameters(
2      train_dataloader=train_dataloader,
3      val_dataloader=val_dataloader,
4      model_path="optuna_tft",
5      n_trials=500,
6      max_epochs=100,
7      gradient_clip_val_range=(0.01, 1.0),
8      hidden_size_range=(4, 128),
9      hidden_continuous_size_range=(4, 64),
10     attention_head_size_range=(1, 4),
11     learning_rate_range=(0.001, 0.3),
12     dropout_range=(0.1, 0.3),
13     trainer_kwargs=dict(
14         accelerator="gpu",
15         devices=1
16     ),
17     use_learning_rate_finder=False,
18     verbose=True
19 )
20
21 with open("tft_study_results.pkl", "wb") as fout:
22     pickle.dump(study, fout)
23
24 best_params = study.best_trial.params
25 print("<----- MEJORES HIPERPARÁMETROS ENCONTRADOS ----->")
26 print(best_params)
27
28 early_stop_callback = EarlyStopping(monitor="val_loss", min_delta=1e-4, patience=10, verbose=False,
29     mode="min")
30 lr_monitor = LearningRateMonitor()
31 checkpoint_callback = ModelCheckpoint(
32     dirpath="checkpoints_tft_CONbusqueda", # Cambiar carpeta cuando ejecutemos la búsqueda
33     filename="tft-epoch{epoch:02d}-val_loss{val_loss:.2f}",
34     auto_insert_metric_name=False,
35     monitor="val_loss",
36     mode="min",
37     save_top_k=1,
38     save_last=True
39 )
40
41 trainer = Trainer(
42     max_epochs=50,
43     accelerator="gpu",
44     devices=1,
45     gradient_clip_val=0.1,
46     callbacks=[lr_monitor, early_stop_callback, checkpoint_callback],
47 )
48
49 tft = TemporalFusionTransformer.from_dataset(
50     training_dataset,
51     learning_rate=best_params['learning_rate'],
52     hidden_size=best_params['hidden_size'],
53     attention_head_size=best_params['attention_head_size'],
54     dropout=best_params['dropout'],
55     hidden_continuous_size=best_params['hidden_continuous_size'],
56     output_size=7,
57     loss=QuantileLoss(),
58 )
59
60 print(f"Número de parámetros del modelo: {tft.size() / 1e3:.1f}k")

```

Listing 2.2: Código de implementación del modelo TFT.

2.3.1.8. Resultados

Para la elección de lo que podemos entender como *mejor modelo* se han comparado tres métricas: MAE, RMSE y R^2 . Una vez entrenados los modelos sobre el conjunto de entrenamiento se realiza

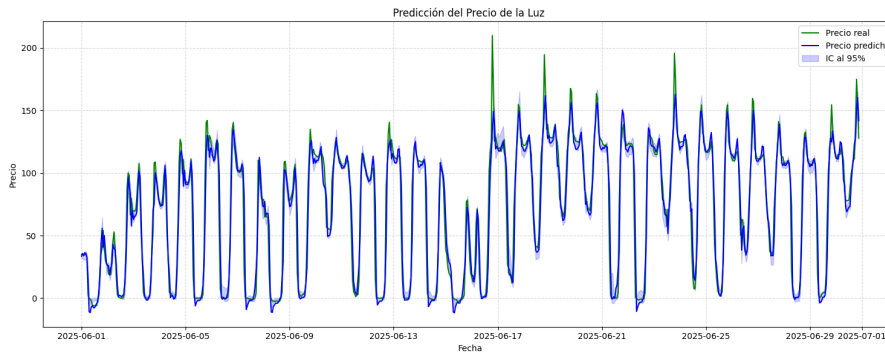
la prueba sobre el *test set* y posteriormente validamos los resultados sobre otro conjunto, del que conocemos ya los valores reales, es decir, entrenamos nuestro modelo para conocer como predice sobre los datos que le hemos introducido y después probamos sobre un conjunto, a priori de variables desconocidas, comprobando así si hemos conseguido generalizar bien el comportamiento de la serie temporal. En la tabla siguiente se muestran los valores mencionados:

Modelo	Test set			Predicción		
	MAE	RMSE	R ²	MAE	RMSE	R ²
SARIMAX(2,1,1)x(1,1,1,24)	47.26	61.64	-4.93	54.68	66.83	-1.07
SARIMAX(1,1,1)x(1,1,1,24)	47.11	61.77	-4.96	55.00	67.16	-1.09
SARIMAX(2,1,0)x(1,1,0,24)	47.29	61.90	-4.98	55.25	67.53	-1.11
XGBoost 70/30 + RM precio	7.30	11.08	0.96	7.43	12.23	0.94
XGBoost 90/10 + RM precio	6.33	10.78	0.92	6.75	11.14	0.95
XGBoost 90/10 + RM todo	7.67	11.60	0.95	7.74	12.94	0.94
RF (SIN búsqueda)	5.17	8.63	0.92	6.75	11.71	0.94
RF (CON búsqueda)	10.09	14.38	0.76	13.36	18.64	0.87
TFT (SIN búsqueda)	15.16	21.10	0.34	45.63	56.12	-0.19
TFT (CON búsqueda NO completa)	19.43	24.10	0.52	25.87	30.19	0.45

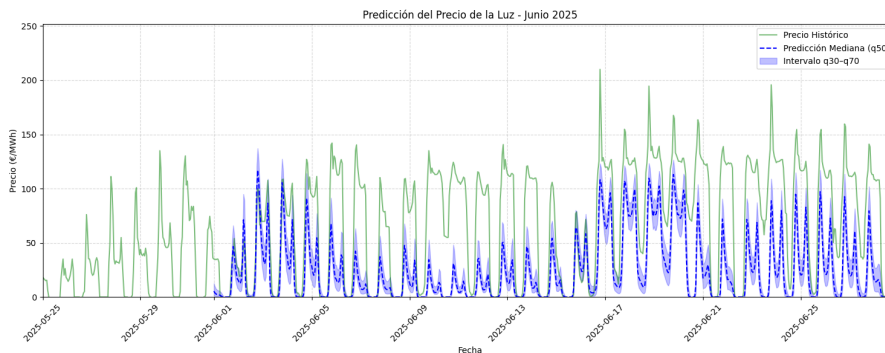
Tabla 2.1: Resultados por modelos para cada conjunto de datos.

A la vista de los resultados de la tabla 2.1 podemos observar que el modelo *SARIMAX* es el que peores métricas muestra, no consiguiendo captar de manera general el comportamiento de la serie temporal. Cabe destacar en este punto que también se trata del modelo que al que menos tiempo he invertido ya que era consciente que los otros iban a reportar mejores resultados de manera más sencilla y rápida. Por otro lado, los resultados de los modelos *XGBoost* y *RandomForest* son parejos y muestran un buen rendimiento, siendo el *XGBoost* ligeramente mejor.

En último lugar, para el TFT no he llegado a conocer los resultados finales. La implementación de la búsqueda de hiperparámetros la realicé, tal y como se ha mostrado en el Apartado 2.3.1.7, pero debido al gran tiempo que requiere no pude realizar el análisis de este modelo. Sin embargo, este es un modelo más complejo. Se trata de una adaptación de los *Transformers* (modelos de deep learning utilizados en el procesamiento de lenguaje natural como GPT) a series temporales, por lo que de él se esperan unos mejores resultados, comparables a los del *XGBoost*.



(a) Resultados para el modelo *XGBoost*.



(b) Resultados preliminares TFT, sin búsqueda exhaustiva de hiperparámetros.

En conclusión, mi recomendación y sensación, dentro del contexto del proyecto, es que el modelo *XGBoost* es la mejor elección en la relación de tiempo y resultados. El interés principal recaía en el comportamiento y la elección del instante en el que llevar a cabo las fundiciones y no tanto un precio muy exacto. Por lo tanto, el modelo *XGBoost* es el que mejor se adapta a las necesidades del proyecto, siendo capaz de predecir con una precisión aceptable y con un tiempo de entrenamiento razonable.

2.3.2. Modelo físico

Con el fin de explicar con algo de detalle cómo se modela un proceso de este tipo, voy a dar una introducción teórica del proceso, para traducirlo después a términos matemáticos. Como es comprensible, no se hicieron todos estos cálculos y las suposiciones y situaciones de la práctica no requieren tanto tecnicismo. Por esto, en la parte final de esta sección explicaré brevemente qué es lo que podría resultar útil de este planteamiento de cara a su implementación para una empresa.

2.3.2.1. Situación simplificada: Primera aproximación

Dado un metal conocido (X) de temperatura de fusión T_f , calor específico en los estados sólido y líquido, c_{sol} y c_{liq} respectivamente, entonces a la hora de conocer cuánta energía será necesaria tendremos que tener en cuenta:

- I El metal X a una temperatura inicial T_0 debe ser calentado hasta la temperatura de fusión T_f , esto dependerá del *calor específico* del material en el estado *sólido*, que viene a ser el *coste energético* para aumentar la temperatura del mismo.

- II Una vez alcanzado dicho objetivo el material cambia de estado de la materia requiriendo una nueva cantidad de calor dependiente del *calor latente* del material, denominado como L_{s-1} .
- III Finalmente supondremos que seguiremos elevando la temperatura por lo que, de manera análoga al paso I, tendremos que tener en cuenta esta energía. Llamaremos a este límite T_c

Para realizar dichos cálculos, haremos uso de las fórmulas de calor para calor específico y transiciones de fase.

$$Q_{pf} = m \cdot c_{sol} \cdot (T_f - T_0). \quad (2.1)$$

$$Q_f = m \cdot L. \quad (2.2)$$

$$Q_e = m \cdot c_{liq} \cdot (T_c - T_f). \quad (2.3)$$

De este modo, $Q = Q_{pf} + Q_f + Q_e$. Finalmente, deberíamos tener en cuenta que los hornos no son ideales y tenemos una cierta pérdida de energía. Este hecho, cuantificado por el rendimiento $\eta \in [0, 1]$, provoca un aumento en el calor total necesario: $Q^{tot} = Q/\eta$.

2.3.2.2. Situación general

Nos situamos ahora ante el problema general, en el que partimos de temperaturas iniciales cercanas a la ambiente ($T_0 \simeq 25^\circ\text{C}$) mientras que llegamos a una temperatura final, T_f . En este caso tendremos que aplicar la formulación general:

$$Q = m \cdot \int_{T_a}^{T_b} c(T) dT, \quad (2.4)$$

donde m es la masa del compuesto X y $c(T)$ es la función del calor específico con la temperatura. Para poder operar con esta expresión sería necesario consultar datos tabulados sobre el material y una vez obtenidos los datos de c para distintas temperaturas, interpolamos para obtener una función con la que calcular, de manera aproximada, el calor en la Ecuación (2.4). De este modo llegaríamos a una situación similar a la descrita en la Apartado 2.3.2.1, obteniendo Q^{tot} . En ambos casos, como se va a tener desarrollo polinómico de $c(T)$, podemos calcular

$$Q = m \cdot \int_{T_a}^{T_b} c(T) dT = m \cdot \int_{T_a}^{T_b} [a_0 + a_1 T + \dots + a_n T^n] dT = \quad (2.5)$$

$$= m \cdot \left[a_0 T + \frac{a_1}{2} T^2 + \dots + \frac{a_n}{n+1} T^{n+1} \right] \Big|_{T_a}^{T_b} = m \cdot p(T_a, T_b), \quad (2.6)$$

donde sustituyendo obtenemos el valor deseado. Para el primer caso, tenemos que T_a coincide con la temperatura *ambiente* de la pieza mientras que T_b con la de fusión del material X. Por otro lado, para la etapa final, la temperatura de fusión coincide con la inicial y la final que habíamos denominado T_f .

2.3.2.3. Traducción matemática

La situación anterior debe ser convenientemente planteada como restricciones, de cara a la resolución del problema de optimización con el que nos encontramos. Aunando lo descrito en la Apartado 2.3.2.2 podemos escribir que el calor Q necesario para la pieza X es:

$$Q = Q_{pf} + Q_f + Q_e + Q' = m \cdot (p_I + L + p_{III}) + Q', \quad (2.7)$$

Finalmente, debería considerarse el rendimiento η del horno. Podríamos suponer que este valor es aproximadamente constante de manera que $\min \eta f(x) = \eta \min f(x)$, pudiendo prescindir de este valor. Si no fuera así deberíamos formularlo como una función e introducirlo en nuestro problema con sus correspondientes restricciones. Ya por último, se ha obtenido el calor Q con unidad (J) *julios*, mientras que habitualmente el coste eléctrico se da en €/kWh por lo que deberemos realizar la debida conversión. Esta, al igual que pasa con el rendimiento, influye en el valor numérico final pero no en el problema a optimizar.

2.3.2.4. Implementación

Como se ha mencionado, el trasfondo teórico de un proceso como este es, en muchos casos, innecesario para dar una solución en el ámbito profesional, si bien es lo técnicamente correcto ⁴. Sin embargo, entender este planteamiento es importante debido a que la solución que se le otorga a una empresa está fundamentada en ello.

Por lo general, no solo en este proyecto, sino también en la investigación realizada, lo más demandado en procesos de este tipo son las *curvas de potencia* o respecto de otras magnitudes como el calor, temperatura... Del modelo planteado anteriormente podemos extraer ciertas conclusiones.

Es habitual que las empresas no tengan acceso a un histórico muy detallado del proceso, por lo que en ocasiones contamos con medidas escasas, sin embargo, podemos observar como la potencia ($P = Q/t$) crece de manera proporcional al calentamiento y depende directamente de la masa y las propiedades térmicas del material. Por ejemplo, una curva de potencia típica muestra cómo varía la demanda energética a lo largo del ciclo de fundición: inicialmente, la potencia requerida es mayor durante el calentamiento rápido, disminuyendo una vez alcanzada la temperatura de fusión y estabilizándose durante el mantenimiento térmico (una etapa muy común en fundiciones por ejemplo para eliminar impurezas).

De igual modo, las curvas de temperatura permiten visualizar el perfil térmico de la pieza, identificando los puntos críticos del proceso (como el paso por la temperatura de fusión). Estas curvas son útiles para ajustar los hiperparámetros del horno, optimizar los tiempos de ciclo y prever posibles incidencias.

2.3.3. Optimización de la superficie de enfriamiento

En último lugar, nos encontramos con el problema que aborda la distribución de las piezas metálicas obtenidas de la colada en una superficie para que enfríen. Claramente, el objetivo es maximizar la cantidad de piezas depositadas en la misma de manera simultánea. Además de las condiciones lógicas de tamaño, se cuenta con otras restricciones como son que las piezas no son apilables (de ahí que se trate de un problema de superficie y no de volumen) y es necesario dejar unos márgenes de seguridad entre piezas. En la resolución de este problema participé en la propuesta de resolución pero no he llegado a implementar completamente ésta por lo que voy a detallar el flujo de trabajo. De manera anterior a este proceso se supone que se ha realizado la predicción de los precios de la luz y con ello una propuesta de planificación, de manera que ahora sería el momento de comprobar cómo se va a distribuir en la nave.

⁴Evidentemente, para llevarlo al grado más óptimo habría que introducir pérdidas de calor, más factores disipativos que puedan afectar al proceso, etc.

2.3.3.1. Tamaño de cajas

En función del diámetro de las piezas fundidas se asignan cajas, de forma cuadrada o rectangular, rellenas de arena en las que la pieza reposará hasta la siguiente fase del proceso

2.3.3.2. Chequeo de espacio

Una vez definidas las cajas, se comprueba si el área disponible es suficiente para albergar la carga de trabajo suficiente.

- En caso negativo se rechaza la propuesta, mostrando datos como la superficie necesaria y ciertas combinaciones de piezas que permitirían depositar el material, por si el cliente quisiera aceptar la contrapropuesta y modificar la planificación.
- En caso positivo, se procede a la siguiente fase.

Hay que remarcar aquí que, como se ha mencionado anteriormente, entre las cajas debe existir un margen de seguridad. Para incluir esto en el modelo se implementa en la función del área de cada pieza, es decir, si llamamos A al área de la superficie de trabajo necesaria, de manera que $A = \sum A_i$ donde $i \in \{\text{piezas}\}$, entonces para cada pieza tendríamos que tener en cuenta que $A_i = (a + \frac{d}{2}) \cdot (b + \frac{d}{2})$, con a, b los lados de la caja y d el margen de seguridad.

2.3.3.3. Distribución de superficie

Una vez validada la propuesta, se procede a distribuir las cajas. Para ello se tienen que tener en cuenta la forma de la superficie, dado que hay que diferenciar entre la capacidad de almacenaje y la capacidad de distribución, me explico: Podríamos tener una superficie rectangular con una gran número de cajas que dividan mi área en dos zonas, por ejemplo de 2 m^2 , entonces no podríamos introducir una pieza de 4 m^2 aunque quisiéramos. Matemáticamente hablando deberíamos ver cómo de conexas es nuestra superficie.

Una vez superada esta parte, se procede a distribuir las piezas, colocando las más grandes en primer lugar y se van comprobando de manera combinatoria las posibilidades del resto. Esta forma de actuar, a fuerza bruta, se eligió porque ciertamente no se contaba con un gran número de piezas por lo que la complejidad de un problema como este, que crece como $n!$ si n es el número de piezas, no sería un gran problema en la práctica. Si contásemos con un mayor número de cajas se debería optar por un algoritmo mucho más refinado que disminuyera la cantidad de prueba-error.

2.3.3.4. Ejemplo de implementación

Como se ha mencionado, no he llegado a implementar completamente este proceso, sin embargo, si realicé una pequeña parte de la implementación, que se muestra a continuación. En este caso, se trata de un ejemplo sencillo en el que se comprueba si una superficie rectangular es capaz de albergar una serie de cajas con unas dimensiones concretas y un margen de seguridad.

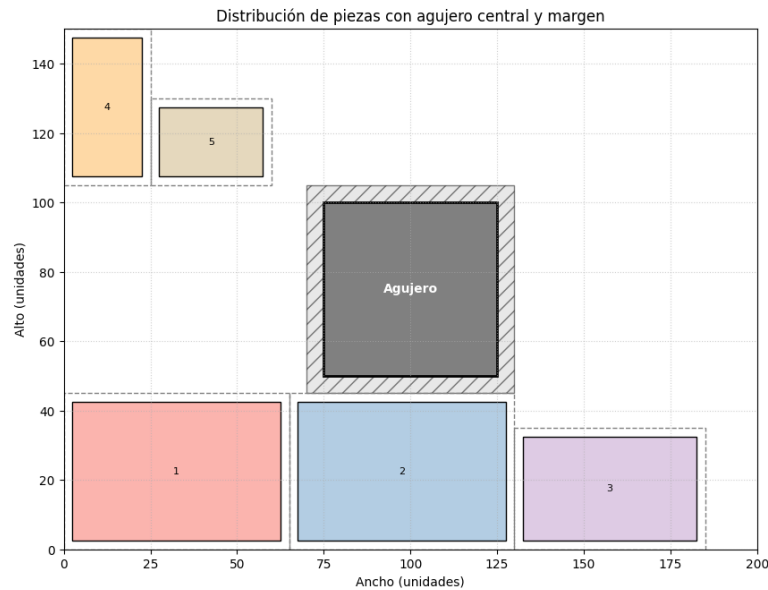


Figura 2.20: Ejemplo de la distribución de las cajas en la superficie de enfriamiento. Para ello se hizo uso de la librería *rectpack*, que permite realizar el problema de la mochila en 2D, o el *bin packing* con rectángulos (y por tanto cuadrados).

```

1  packer = newPacker(pack_algo=GuillotineBafSas, rotation=True)
2  for rect in piezas_sorted:
3      packer.add_rect(*rect)
4  for w, h in sub_bins:
5      if w > 0 and h > 0:
6          packer.add_bin(w, h)
7  packer.pack()
8  fig, ax = plt.subplots(figsize=(10, 8))
9  colors = plt.cm.get_cmap('Pastel1', len(piezas))
10 coordenadas_finales = []
11 ax.add_patch(plt.Rectangle((hx, hy), hw, hh, facecolor='lightgray', edgecolor='black', lw=1, alpha=0
12 .5, hatch='//'))
13 ax.add_patch(plt.Rectangle((hole_x, hole_y), hole_w, hole_h, facecolor='gray', edgecolor='black', lw
14 =2))
15 ax.text(hole_x + hole_w/2, hole_y + hole_h/2, "Agujero", ha='center', va='center', color='white',
16 weight='bold')
17 piece_idx = 0
18 for bin_idx, abin in enumerate(packer):
19     bin_x, bin_y = sub_bin_positions[bin_idx]
20     for rect in abin:
21         x_margin, y_margin, w_margin, h_margin = rect.x + bin_x, rect.y + bin_y, rect.width, rect.
22         height
23         piece_w = w_margin - margin
24         piece_h = h_margin - margin
25         piece_x = x_margin + margin / 2
26         piece_y = y_margin + margin / 2
27         coordenadas_finales.append({
28             "pieza": piece_idx + 1, "x": piece_x, "y": piece_y,
29             "ancho": piece_w, "alto": piece_h
30         })
31     ax.add_patch(plt.Rectangle((x_margin, y_margin), w_margin, h_margin,
32         facecolor='none', edgecolor='gray', lw=1, linestyle='--'))
33     ax.add_patch(plt.Rectangle((piece_x, piece_y), piece_w, piece_h,
34         facecolor=colors(piece_idx), edgecolor='black', lw=1))
35     ax.text(piece_x + piece_w/2, piece_y + piece_h/2, f'{piece_idx+1}',
36         ha='center', va='center', fontsize=8, color='black')
37     piece_idx += 1
38 ax.set_xlim(0, container_width)
39 ax.set_ylim(0, container_height)
40 ax.set_aspect('equal', adjustable='box')
41 ax.set_title("Distribución de piezas con agujero central y margen")
42 plt.xlabel("Ancho (unidades)")
43 plt.ylabel("Alto (unidades)")
44 plt.grid(True, linestyle=':', alpha=0.6)

```

Listing 2.3: Código de implementación del problema de distribución de piezas.

2.4. Relación entre formación recibida y actividades

Con salvedad de la etapa de formación, creo que ha quedado de manifiesto la íntima relación entre los conocimientos adquiridos durante el grado en matemática, y con menor relevancia en física, y las tareas desempeñadas durante este tiempo. Intentando particularizar en asignaturas: Programación, impartida durante el segundo cuatrimestre del primer curso, las asignaturas de estadística, principalmente Inferencia estadística, del segundo cuatrimestre de tercero (cuarto en mi caso debido a la distribución del Doble Grado) y en cuanto a física, claramente, Termodinámica.

Aunque a priori pueda parecer que no hay mucha relación entre modelos de redes neuronales, machine learning y deep learning, una vez investigas y te adentras en los mecanismos que hacen que estas tecnologías (bases de la IA) funcionen, rápidamente observas que la estadística son clave en estas áreas. Esta característica, que quizás queda algo más alejada del público general, es uno de los mayores atractivos para personas con formación en matemáticas que se sienten atraídas por una de la que probablemente sea la revolución del siglo: La IA.

2.5. Atención y asesoramiento recibido

Durante mi estancia en la empresa el trato y atención han sido inmejorables. En este tiempo me he encontrado bajo la supervisión de mis dos compañeros del equipo de Data, ambos analistas de datos, si bien cabe remarcar que el apoyo global de la plantilla ha estado presente.

En todo el periodo he estado presente y participado activamente en reuniones de proyectos, tomas de decisión junto a los responsables (CEO, CTO y COO), de manera que además del aprendizaje técnico he sido capaz de conocer el funcionamiento de situaciones habituales en el entorno laboral.

Capítulo 3

Conclusiones

El programa de prácticas llevado a cabo estos dos meses ha sido excelente, más que satisfactorio. Como se ha comentado en la introducción y varias veces a lo largo de esta memoria, mi principal interés estaba en el acercamiento al mundo laboral, de manera que pudiera conocer más de cerca el sector. En estos dos meses y medio me he adentrado dentro de un mundo que desconocía completamente más allá del interés superficial que suscita la IA y sus aplicaciones a situaciones de mercado.

Todas las habilidades, tanto técnicas como personales, adquiridas en este tiempo serán de una enorme utilidad para toda la vida. Además, el aprendizaje llevado a cabo en tareas como el trabajo en equipo, los plazos de entrega, las decisiones en proyectos y el trato con clientes han permitido que tenga una primera toma de contacto con el mundo profesional.

A modo de resumen y para finalizar: Las primeras semanas, en las que mi formación estuvo enfocada en un ámbito que desconocía totalmente y mi posterior inclusión en un grupo de trabajo ajeno tanto a la formación como a mi conocimiento, supusieron unas semanas de dureza ya que todo era nuevo. Progresivamente fui aprendiendo y adaptandome al flujo de trabajo, viendome incluido en diversos proyectos, entre ellos el principal expuesto aquí, de gran dificultad tanto en lo técnico como en su implementación para el cliente. Sin embargo, esta situación me ha enseñado a trabajar de manera autónoma, documentarme y asumir el liderazgo para la toma de decisiones, confiando en mi trabajo y conocimientos, de manera que he podido sacar adelante el trabajo.

Bibliografía

- [1] A. Campuzano Solórzano, *Repositorio GitHub de la memoria*, <https://github.com/Alcamsod/Memoria>, 2025.
- [2] N. Vega Muñoz, *Previsión del Precio SPOT en el Mercado Eléctrico Español con Redes Neuronales*, 2024. dirección: https://oa.upm.es/82722/3/TFG_NICOLAS_VEGA_M.pdf.
- [3] B. Lim, S. O. Arik, N. Loeff y T. Pfister, *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting*, 2020. arXiv: 1912.09363 [stat.ML]. dirección: <https://arxiv.org/abs/1912.09363>.