Trabajo Fin de Máster

Máster en Ciencia de Datos e Ingeniería de Datos en la Nube

La Viña Wine Prediction
Caja Mar UniversityHack Datathon 2023

Autor: Pablo Moreira García

Tutor: Juan Carlos Alfaro Jiménez

Septiembre, 2023

Dedicado a mi familia y a todos aquellos que estuvieron a mi lado durante la elaboración de este proyecto y mi vida estudiantil
Dedicado a mi familia y a todos aquellos que me acompañaron a lo largo de este proyecto y durante mi trayecto académico. Su apoyo inquebrantable ha sido fundamental en este camino.

Declaración de Autoría

Yo, Pablo Moreira García con DNI 48150755W, declaro que soy el único autor del trabajo fin de grado titulado “La Viña Wine Prediction Caja Mar UniversityHack Datathon 2023” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 14 de Septiembre de 2023

Fdo: Pablo Moreira García

Resumen

Este proyecto pretende generar un modelo que permita predecir la producción de uva de un conjunto de viñedos meses antes de la cosecha facilitando la optimización de las diversas actividades y etapas del proceso agrario y mejorar la producción.

Para ello se ha desarrollado un proceso KDD (Knowledge Discovery in Databases) durante el cual se han preparado los datos, se han analizado y explorado y se han generado diferentes modelos de aprendizaje automático, validándolos y finalmente seleccionando un único modelo en base a su rendimiento, es decir, teniendo en cuenta la calidad de sus predicciones y el tiempo de ejecución necesario. Tras la selección del modelo se desarrolla su explicabilidad para una mejor adopción e integración del mismo.

Agradecimientos

Agradecimientos a mi familia por su apoyo durante todo mi viaje académico, no habría sido nada sin su amor y comprensión. También me gustaría agradecer a mis profesores dado que no habría llegado tan lejos sin su orientación y los conocimientos impartidos, su dedicación y pasión por la enseñanza dejaron una gran huella en mí. Por supuesto he de agradecer a mis amigos por su paciencia y comprensión durante los momentos más difíciles de este proyecto. También he de agradecerme a mí mismo por la dedicación, la perseverancia y el esfuerzo puesto en este proyecto para sacarlo adelante. Fue un viaje desafiante a la par que gratificante. Y, por último, pero no menos importante, quiero agradecer a María, por su comprensión y preocupación por mí.

Índice general

Capítulo 1 Introducción 1

1.1 Introducción 1

1.2 Objetivos 1

1.3 Competencias 2

1.4 Estructura del proyecto 2

Capítulo 2 Estado del Arte 4

2.1 Introducción 4

2.2 Problema Wine Prediction 4

2.2.1 Conjunto de datos TRAIN 4

2.2.2 Conjunto de datos METEO 5

2.2.3 Conjunto de datos ETO 6

2.3 Librerías, métodos y técnicas utilizadas 7

2.3.1 Preprocesamiento 7

2.3.2 Análisis exploratorio de datos 8

2.3.3 Modelado 9

2.4 Modelos Utilizados 10

2.4.1 Ridge 10

2.4.2 Arboles de regresión 10

2.4.3 Vecinos más cercanos 11

2.4.4 Random forest 11

2.4.5 Gradient boosting 12

2.4.6 Extreme gradient boosting 13

2.5 Evaluación de los resultados 13

2.5.1 Coeficiente de determinación (R Cuadrado) 14

2.5.2 Error absoluto medio (MAE) 14

2.5.3 Error cuadrático medio (MSE) 14

2.5.4 Raíz del error cuadrático medio (RMSE) 14

2.6 Explicabilidad 15

2.6.1 SHAP 15

Capítulo 3 Preparación, Análisis Exploratorio y Preprocesamiento de los Datos 17

3.1 Introducción 17

3.2 Conjunto de Datos TRAIN 17

3.2.1 Preparación de Datos 17

3.2.2 Análisis Exploratorio de Datos 18

3.2.3 Preprocesamiento de Datos 23

3.3 Conjuntos de Datos METEO y ETO 25

3.3.1 Preparación de Datos 25

3.3.2 Análisis Exploratorio de Datos 27

3.3.3 Preprocesamiento de Datos 29

3.4 Conjunto de Datos Final 30

Capítulo 4 Experimentos, Resultados y Explicabilidad 32

4.1 Introducción 32

4.2 Características del entorno 32

4.3 Modelos utilizados 32

4.4 Experimentos y Resultados 33

4.4.1 Experimento 1 34

4.4.2 Experimento 2 35

4.4.3 Experimento 3 36

4.4.4 Experimento 4 37

4.4.5 Experimento 5 38

4.4.6 Experimento 6 39

4.4.7 Experimento 7 39

4.4.8 Experimento 8 40

4.4.9 Selección de los mejores modelos 41

4.5 Explicabilidad 42

Capítulo 5 Conclusiones y Trabajo Futuro 45

5.1 Conclusiones 45

5.2 Trabajo Futuro 46

Bibliografía 47

Índice de figuras

Figura 1: Horas del día que contiene cada periodo 6
Figura 2: Ejemplo de predicción del modelo RIDGE 10
Figura 3: Ejemplo de Decision Tree Regressor 11
Figura 4: Funcionamiento del modelo KNeighborsRegressor 11
Figura 5: Funcionamiento del modelo RandomForestRegressor 12
Figura 6: Funcionamiento del modelo GradientBoostingRegressor 13
Figura 7: Mapa de calor de las variables del conjunto TRAIN 19
Figura 8: Pairgrid de scatterplots y kdeplots para mostrar la relación de unas variables con otras 19
Figura 9: Histplots de las variables del conjunto TRAIN 20
Figura 10: Gráficos de barras para la distribución de las variables VARIEDAD, MODO, TIPO y COLOR por campaña 21
Figura 11: Lineplots que describen la producción por campaña, teniendo en cuenta las variables VARIEDAD, TIPO, MODO y COLOR 22
Figura 12: Lineplot de la producción por campaña 22
Figura 13: Distribución de las variables preprocesadas 24
Figura 14: Heatmap de los conjuntos de datos METEO y ETO respectivamente 27
Figura 15: Ejemplo de los histplots que representan la distribución de las variables de ETO 28
Figura 16: Graficas visuales de la predicción de datos de Prophet 29
Figura 17: Mapas de calor mostrando la correlación entre las variables del conjunto de datos. El primero con el conjunto de datos completo, el segunda tras quitar variables repetidas y variables con varianza 0 y el ultimo tras quitar variables con alta correlación 31
Figura 18: Representación de los SHAP Values con Summary_Plot 42
Figura 19: Representación con Force_Plot, la superior es del conjunto completo de datos de Test y la inferior es de una única instancia. 43
Figura 20: Representación con decisión_plot de cómo va cambiando las predicciones del modelo al utilizar las distintas variables del conjunto completo de datos. 43
Figura 21: Representación con decisión_plot de cómo va cambiando la predicción del modelo al utilizar las distintas variables con una única instancia. 44
Figura 22: Representación con Dependence_plot los SHAP values de SUPERFICIE teniendo en cuenta MODO, TIPO y COLOR 44

Índice de tablas

Tabla 1: Tipo de las variables del conjunto TRAIN 17
Tabla 2: Diferentes estadísticas de las variables del conjunto TRAIN 18
Tabla 3: Numero de valores ausentes por campaña del conjunto TRAIN 18
Tabla 4: Diferentes estadísticas de las variables del conjunto METEO 25
Tabla 5: Numero de datos ausentes por variable y año del conjunto METEO 26
Tabla 6: Diferentes estadísticas de las variables del conjunto ETO 26
Tabla 7: Numero de datos ausentes por variable y año del conjunto ETO 27
Tabla 8: Resultados experimento 1 35
Tabla 9: Resultados experimento 2 36
Tabla 10: Medias ponderadas de los resultados experimento 3 37
Tabla 11: Medias ponderadas de los resultados experimento 4 38
Tabla 12: Medias ponderadas de los resultados experimento 5 38
Tabla 13: Medias ponderadas de los resultados experimento 6 39
Tabla 14: Medias ponderadas de los resultados experimento 7 40
Tabla 15: Medias ponderadas de los resultados experimento 8 41

Capítulo 1

Introducción

1.1 Introducción

Esta memoria presenta el desarrollo del proyecto presentado al Datathon de Caja Mar en el año 2023, en el cual se colabora con:

- La Cooperativa La Viña, bodega creada en 1945, cuando un grupo de 38 emprendedores se asociaron bajo una misma cooperativa reuniendo una superficie de cultivo de 47 hectáreas con una producción de 250.000 kg de uva. A lo largo de estos casi 80 años la actividad vitivinícola ha ganado importancia y la bodega ha ido creciendo en número de socios y producción.
- The Weather Company, an IBM Business, el clima cumple un papel fundamental en la inteligencia comercial, y esta empresa se enfoca en utilizar su conocimiento y tecnología para avanzar en la ciencia del pronóstico meteorológico preciso y ayudar a las personas en todas partes a tomar las mejores decisiones.

1.2 Objetivos

España es el tercer productor mundial de vino. Disponer de una previsión precisa de la producción en una campaña agrícola es cada vez más necesario de cara a optimizar todos los procesos de la cadena: recolección, traslado, procesado, almacenamiento y distribución.

Dado lo anterior y partiendo de amplios conjuntos de datos con histórico de producciones de los viñedos que conforman la cooperativa La Viña, así como histórico de la climatología de estos, el objetivo será crear el mejor modelo de predicción de producción de una campaña en base al

cual se pueda estimar la cosecha que dispondrá la cooperativa meses antes de la recolección [1].

1.3 Competencias

- Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.
- Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.
- Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes entornos inteligentes.
- Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

1.4 Estructura del proyecto

La memoria consta de los siguientes capítulos:

- **Introducción:** Explicación y los principales objetivos del problema sobre el que se realizó el Datathon y por consiguiente este proyecto.
- **Estado del Arte:** Introducción a los distintos conceptos clave para el desarrollo del proyecto como el problema al que nos enfrentamos, las librerías, modelos y otras técnicas que hemos utilizado, así como las métricas de evaluación y la explicabilidad del modelo final.
- **Preparación, Análisis Exploratorio y Preprocesamiento de datos:** En este capítulo se explicarán el proceso que se ha llevado a cabo paso a paso, desde el análisis exploratorio hasta la explicabilidad del modelo final.
- **Experimentos, Resultados y Explicabilidad:** En este se expondrán las características del entorno de ejecución, así como los resultados obtenidos tras ejecutar los diferentes experimentos.
- **Conclusiones:** Por último, el proyecto concluirá con una reflexión sobre el trabajo realizado y sobre el análisis basado en aprendizaje automático para resolver problemas

de minería de datos, además de proponer posibles líneas de trabajo que podrían ser utilizadas en un futuro.

Capítulo 2

Estado del Arte

2.1 Introducción

En este capítulo vamos a tratar el problema al que nos enfrentamos, librerías y técnicas utilizadas para el desarrollo del proyecto, así como los modelos con los que se han realizado los experimentos para alcanzar el resultado final, las métricas utilizadas para evaluar estos modelos. Por último, hablaremos de la explicabilidad y como realizarla con SHAP.

2.2 Problema Wine Prediction

Para el desarrollo del proyecto tenemos 3 conjuntos de datos: TRAIN, que contiene información histórica de las fincas que conforman la cooperativa La Viña; METEO, que dispone de información meteorológica de estaciones climatológicas de la zona a nivel horario de The Weather Company y ETO, que dispone de información meteorológica ampliada y transformada de las mismas estaciones agregada en franjas del día.

2.2.1 Conjunto de datos TRAIN

El primer conjunto de datos, TRAIN, contiene información anual del resultado de las campañas de la cooperativa La Viña, mostrando las siguientes variables:

- **CAMPAÑA:** Año de la campaña.
- **ID_FINCA:** Identificador de finca.
- **ID_ZONA:** Identificador de una zona con una tipología de suelo común.
- **ID_ESTACION:** Identificador de estación meteorológica.
- **ALTITUD:** Altitud media de la finca sobre el nivel del mar en metros.
- **VARIEDAD:** Código de variedad de la uva que se cultiva en la finca.
- **MODO:** Código del modo de cultivo.
- **TIPO:** Tipo de cultivo dentro de la variedad.

• **COLOR:** Identificador del color de la uva.

• **SUPERFICIE:** Superficie en hectáreas que ocupa la finca.

• **PRODUCCION:** Producción en kg. obtenida en la campaña.

Aunque la serie histórica de producción empieza en 2014, la información de la superficie de las fincas solo está disponible para las campañas 2020, 2021 y 2022. La producción de 2022 para cada finca es el valor a estimar.

2.2.2 Conjunto de datos METEO

El segundo conjunto de datos, METEO, contiene información horaria de estaciones climatológicas de The Weather Company de la zona de influencia en el periodo comprendido entre el 29-06-2015 y el 30-06-2022. Este dispone de las siguientes variables:

• **validTimeUtc:** Fecha en la que se tomó la muestra.

• **precip:** Volumen de lluvia en diferentes intervalos de tiempo, última hora, 6 últimas horas, 24 últimas horas, últimos 2 días, últimos 3 días, últimos 7 días, del mes en curso y del año en curso.

• **pressureChange:** Variación máxima en la presión atmosférica en las últimas 3 horas.

• **pressureMeanSeaLevel:** Diferencia barométrica respecto al nivel del mar.

• **relativeHumidity:** Humedad relativa.

• **snow:** Volumen de nieve en diferentes intervalos, última hora, 6 últimas horas, 24 últimas horas, últimos 2 días, últimos 3 días, últimos 7 días, del mes en curso, volumen de nieve trimestral (DIC-FEB, MAR-MAY, JUN-AGO, SEP-NOV) y del año en curso.

• **temperatura:** Temperatura ambiente a 2 metros del suelo.

• **temperatureChange24Hour:** Variación de temperatura respecto al día anterior.

• **temperatureMax24Hour:** Temperatura máxima últimas 24 horas.

• **temperatureMin24Hour:** Temperatura mínima últimas 24 horas.

• **temperatureDewPoint:** Punto de rocío, temperatura a la cual el aire debe ser enfriado a presión constante para alcanzar la saturación. El punto de rocío es también una medida indirecta de la humedad del aire.

• **temperatureFeelsLike:** Sensación térmica. Temperatura aparente resultante de combinación de la temperatura, la humedad y la velocidad del viento.

• **uvIndex:** Radiación ultravioleta categorizada: -2, -1 = No disponible, 0-2 = baja, 3-5 = moderada, 6-7 = alta, 8-10 = muy alta, 11-16 = extrema.

• **visibility:** Visibilidad horizontal desde la estación meteorológica, 999 equivale a ilimitada.

• **windDirection:** Dirección del viento en grados 0 – Norte, 90 – Este, 180 – Sur, 270 – Oeste.

• **windGust:** Velocidad máxima de ráfaga de viento registrada durante el período de observación.

• **windSpeed:** Fuerza del viento.

• **ID_ESTACION:** Identificador de la estación meteorológica.

2.2.3 Conjunto de datos ETO

El último conjunto de datos, ETO, contiene información agregada y transformada de las estaciones climatológicas de The Weather Company en el mismo periodo. Las agregaciones que tenemos son el mínimo, la media y el máximo de cada periodo, disponiendo también de los periodos DAY, DAYTIME, NIGHTTIME, MORNING, AFTERNOON, EVENING, OVERNIGHT para cada variable.

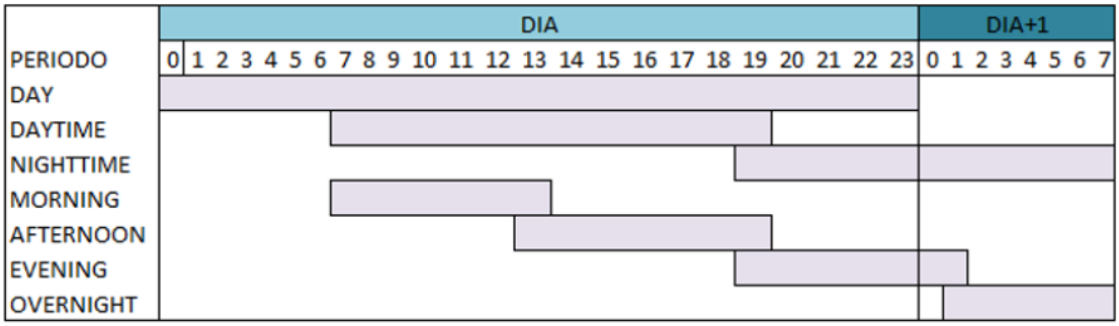

Figura 1: Horas del día que contiene cada periodo

Las variables que tenemos son similares a las del conjunto de datos METEO:

• **date**: Fecha de la muestra.

• **DewPoint**: Punto de rocío, temperatura a la cual el aire debe ser enfriado a presión constante para alcanzar la saturación. El punto de rocío es también una medida indirecta de la humedad del aire.

• **Evapotranspiration**: Evapotranspiración del cultivo de referencia. Esta es una tasa que da la cantidad de agua perdida por un cultivo de referencia.

• **FeelsLike**: Sensación térmica. Temperatura aparente resultante de combinación de la temperatura, la humedad y la velocidad del viento.

• **GlobalHorizontalIrrandiance**: Cantidad total de radiación solar recibida en una superficie horizontal.

• **Gust**: Velocidad máxima de ráfaga de viento registrada durante el período de observación.

• **MSL**: Presión barométrica.

• **precipAmount**: Volumen de lluvia por hora.

• **relativeHumidity**: Humedad relativa.

• **SnowAmount**: Volumen de nieve por hora.

• **Temperature**: Temperatura ambiente a 2 metros del suelo.

• **uvIndex**: Radiación ultravioleta: 0-2 = baja, 3-5 = moderada, 6-7 = alta, 8-10 = muy alta, 11-16 = extrema.

• **visibility**: Visibilidad horizontal desde la estación meteorológica, 999 equivale a ilimitada.

• **windSpeed**: Fuerza del viento.

• **ID_ESTACION**: Identificador de la estación meteorológica [1].

2.3 Librerías, métodos y técnicas utilizadas

Para el desarrollo del proyecto se han utilizado diferentes librerías, métodos y técnicas que se explicarán a continuación (únicamente las que son más interesantes).

2.3.1 Preprocesamiento

- **Numpy**: Es una de las principales librerías utilizadas para el cálculo numérico y el análisis de datos, especialmente para grandes volúmenes de datos. Gracias a esta librería apareció la clase de objetos llamada array, estructura que permite representar datos del mismo tipo organizada en forma de tabla o cuadrícula de distintas dimensiones, y la aparición de funciones muy eficientes para su manipulación.

  Antes de la aparición de Numpy se utilizaban las listas genéricas ofrecidas por Python, pero estos arrays tienen muchas mejoras, son más compactos y utilizar menos memoria y tienen una mejor velocidad de procesamiento. También tienen muchas más funcionalidades como la posibilidad de utilizar operaciones vectorizadas y la posibilidad de almacenar objetos de diferentes tipos [2].

- **Pandas**: Es una librería de Python de código abierto que se usa más ampliamente para problemas de ciencia de datos y aprendizaje automático. Esta construida sobre la librería Numpy, la cual nos facilita el uso de arrays multidimensionales.

  Pandas como una de las librerías de gestión de datos más populares, funciona bien con otros muchos módulos, por ejemplo, para trazar funciones de Matplotlib, entrenar algoritmos de aprendizaje automático en Scikit-Learn, etc. Normalmente se incluye en las distribuciones de Python.

  Pandas facilita la realización de las tareas más repetitivas asociadas a trabajar con datos, limpieza de datos, normalización, visualización, análisis estadístico, carga y almacenamiento de datos y mucho más. Todo esto gracias a la estructura de datos DataFrame, esta es una estructura con dos dimensiones en la cual se pueden almacenar datos de distintos tipos en columnas, es similar a una hoja de Excel, una tabla de SQL o un DataFrame de R [3].

- **Deterministic Regression Imputation**: Técnica en la que sustituimos los datos que faltan por los valores predichos en nuestro modelo de regresión y repetimos este proceso para cada variable que contenga datos ausentes.

  Un problema importante de este método es que reducimos la variabilidad a la variable imputada, puesto que sustituimos los datos que faltan por los resultados de la regresión, los valores predichos se sitúan a lo largo del hiperplano de regresión en el que la variable habría contenido en realidad algo de ruido o sesgo [4][5].
- **Stochastic Regression Imputation:** Para arreglar el problema de la Deterministic Regression Imputation se puede añadir incertidumbre a los valores de las variables imputadas, podemos añadir un poco de ruido distribuido normalmente con una media de cero y una varianza igual al error estándar de las estimaciones de regresión. Este método se denomina Stochastic Regression Imputation [4][5].
- **Prophet:** Es un software de código abierto publicado por el equipo de ciencia de datos básicos de Facebook.

  Este software se basa en un procedimiento para la estimación de datos de series temporales el cual utiliza un modelo aditivo donde las tendencias no lineales se ajustan de forma anual, semanal y diaria. Funciona de forma eficaz en las series temporales que tienen fuertes efectos estacionales y numerosas temporadas de datos. Este procedimiento suele controlar bien los datos atípicos y los cambios de tendencia [6].

2.3.2 Análisis exploratorio de datos

- **Matplotlib:** Matplotlib fue una de las primeras librerías que se desarrollaron para la visualización de datos y trazado de gráficos para el lenguaje Python, siendo clave para trabajar con librerías más avanzadas como Seaborn.

  Pyplot como parte de Matplotlib le permite ser una gran alternativa de código abierto a MATLAB, esta API es una colección de funciones y objetos que se encargan de hacer cambios en las figuras que se están dibujando, desde crear un área de trazado, a dibujar líneas en un área o decorar la gráfica con texto…

  Para organizar y estructurar la visualización de los gráficos, Matplotlib suele crear una figura que representa al gráfico, indicando el número de ejes y después mostrando la figura, esta técnica se conoce como Object-Oriented Style.

  Utiliza también Pandas, para la manipulación y el análisis de datos, y a diferencia de Numpy, Pandas no tiene una dependencia obligatoria con Matplotlib [7].
- **Seaborn:** Seaborn es una librería de Python de código abierto realizada sobre Matplotlib, proporcionándole una interfaz de alto nivel. Esta como muchas otras

librerías se utiliza para la visualización y el análisis exploratorio de datos, integrándose estrechamente con Pandas y estructuras de datos como los DataFrames.

Utilizando un conjunto de datos y un gráfico en específico como parámetros de entrada, Seaborn es capaz de realizar automáticamente un mapeado de los valores de los datos y asignarlos a atributos visuales, como color, tamaño, estilo. Aunque se generen automáticamente, los gráficos generados son fácilmente personalizables.

Esta librería puede ser utilizada durante todo el ciclo de vida de un proyecto, ya que produce gráficos completos a partir de una única llamada de función con un mínimo de argumentos, facilitando la creación de prototipos y el análisis exploratorio de los datos [8].

2.3.3 Modelado

- **Scikit-Learn:** Scikit-Learn es una de las principales librerías para Python utilizadas en el mundo de la ciencia de datos gracias a la gran variedad de algoritmos que contiene, clasificación, regresión, técnicas de agrupamiento/segmentación, reducción de dimensionalidad, etc. Aparte permite un uso compatible con otras librerías utilizadas en el sector como Numpy, Matplotlib, Pandas o Seaborn.

  Todo esto la convierte en una de las herramientas básicas y principales para programar y estructurar sistemas de análisis de datos y modelado [9].
- **Resampling con PCA:** El algoritmo utiliza PCA para generar datos sintéticos que son similares a los datos originales y luego ajusta esos datos para que cumplan con ciertas restricciones en las columnas especificadas antes de devolver el DataFrame aumentado. Esta técnica es útil para el aumento de datos en conjuntos de datos desequilibrados o para generar datos adicionales cuando la cantidad de datos originales es limitada.

  Para ser exactos, inicialmente utilizamos PCA para reducir la dimensionalidad del conjunto de datos utilizando el método fit_transform. Tras esto generamos instancias nuevas con el método resample de Scikit-Learn utilizando el conjunto de datos reducido e invertimos el proceso realizado por PCA con inverse_transform para tener de nuevo un conjunto de datos similar al que teníamos inicialmente, pero con un mayor número de instancias.
- **Deep Learning para tabular data augmentation:** Este algoritmo utiliza una red neuronal para aprender una representación de datos y luego genera datos sintéticos basados en esta representación. Tras esto, ajusta estos datos sintéticos para que cumplan con ciertas restricciones en las columnas especificadas antes de devolverlos

como un conjunto de datos aumentado. Esto es útil para el aumento de datos en conjuntos de datos tabulares y puede ayudar a mejorar el rendimiento de los modelos de aprendizaje automático cuando la cantidad de datos es limitada [10].

2.4 Modelos Utilizados

En este punto se explicará el funcionamiento de los modelos utilizados para la predicción final, se han utilizado desde unos más sencillos y transparentes a otros más complejos con la necesidad de uso de librerías para su explicabilidad y comprensión.

2.4.1 Ridge

El modelo Ridge, también conocido como regresión Ridge o regularización Tikhonov, es un enfoque de regresión que minimiza la función de pérdida de mínimos cuadrados lineales con regularización L2, la cual tiene el efecto de reducir de forma proporcional el valor de todos los coeficientes del modelo, pero sin que estos lleguen a cero. El grado de penalización está controlado por el hiperparámetro λ [11][12].

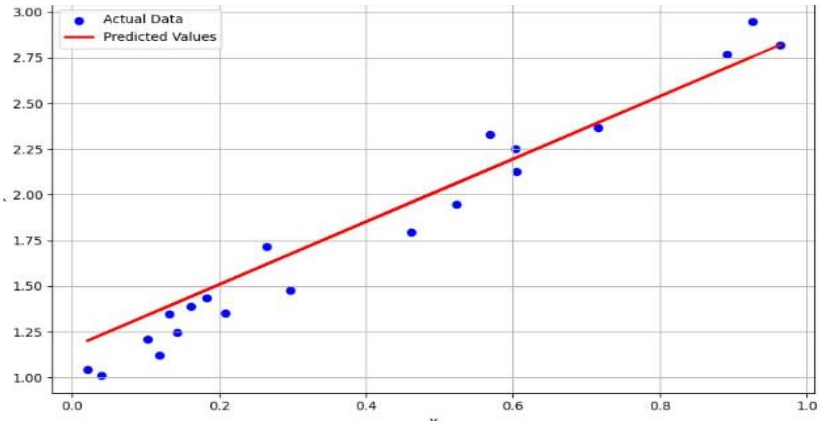

Figura 2: Ejemplo de predicción del modelo RIDGE

2.4.2 Arboles de regresión

Los árboles de regresión/clasificación fueron propuestos por Leo Breiman y son árboles de decisión que tienen como objetivo predecir la variable respuesta y en función de covariables. Los árboles de regresión son aquellos en los que la variable objetivo y es cuantitativa.

Un árbol de regresión consiste en hacer preguntas de tipo $¿x_k \leq c?$ para cada una de las covariables, de esta forma el espacio de las covariables es divido en hiper-rectángulos y todas las observaciones que queden dentro de un hiper-rectángulo tendrán el mismo valor estimado $\hat{y}$ [13][14].

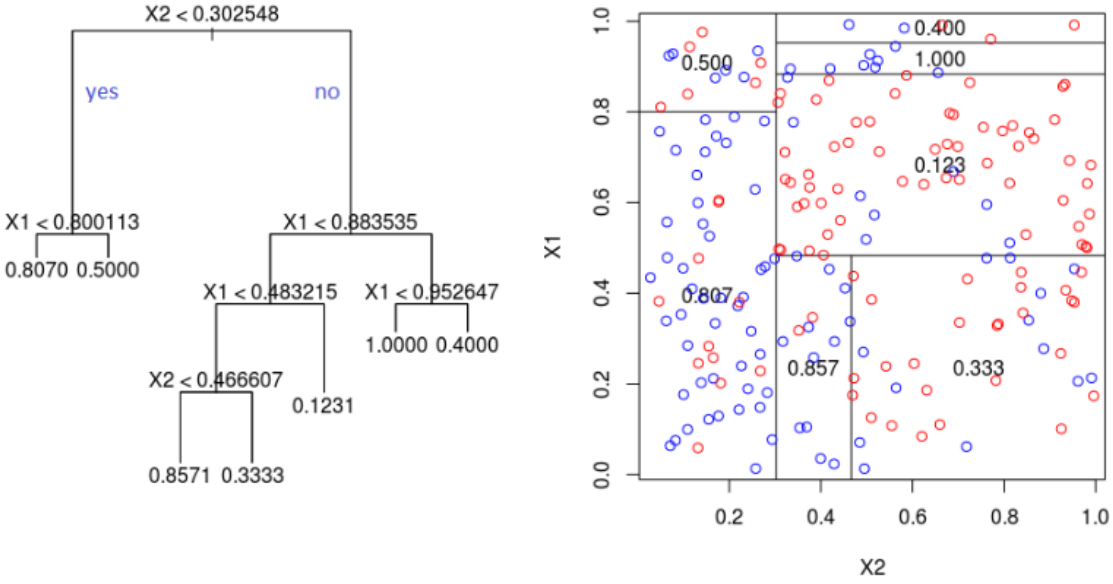

Figura 3: Ejemplo de Decision Tree Regressor

2.4.3 Vecinos más cercanos

El método de los vecinos más cercanos es un algoritmo intuitivo y no paramétrico que almacena todas las observaciones conocidas (conjunto de entrenamiento) y las utiliza para predecir resultados basándose en funciones de similitud o distancia.

La k indica cuántos vecinos se utilizan para realizar la predicción; cuando k > 1 el valor y puede calcularse como la media de todas las salidas de los k vecinos más cercanos (los vecinos se ponderan uniformemente) o éstos pueden ponderarse en función de su distancia [15][16].

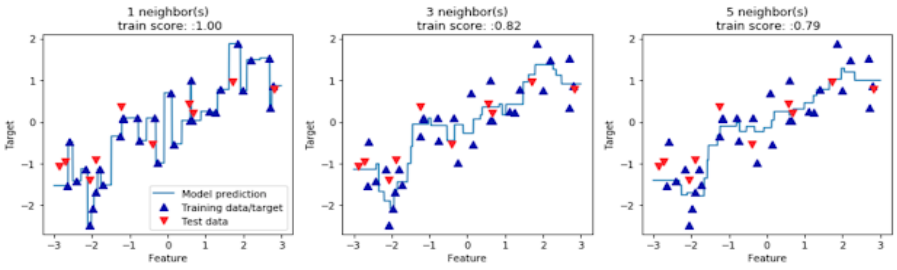

Figura 4: Funcionamiento del modelo KNeighborsRegressor

2.4.4 Random forest

Random forest es una técnica de regresión que combina el rendimiento de numerosos árboles de decisión para clasificar o predecir el valor de una variable. Cuando Random Forest recibe un vector de entrada, construye un número K de árboles de regresión y promedia los resultados. Para evitar la correlación de los diferentes árboles, Random Forest aumenta la diversidad de los árboles de varias maneras:

En primer lugar, en cada nodo de cada árbol, se considera un subconjunto aleatorio de atributos en lugar de utilizar todos los atributos disponibles. Esto significa que cada árbol se entrena solo con un conjunto limitado de características, lo que aumenta la diversidad y reduce la dependencia entre los árboles.

Además, Random Forest genera estos árboles a partir de diferentes subconjuntos de datos de entrenamiento. Este procedimiento implica crear varios conjuntos de datos de entrenamiento mediante un remuestreo aleatorio del conjunto de datos original con sustitución. En otras palabras, los datos se seleccionan aleatoriamente de la muestra de entrada para generar el siguiente subconjunto de entrenamiento. Esto asegura que cada árbol ve una versión ligeramente diferente de los datos, lo que nuevamente aumenta la diversidad y mejora el rendimiento del conjunto [17][18].

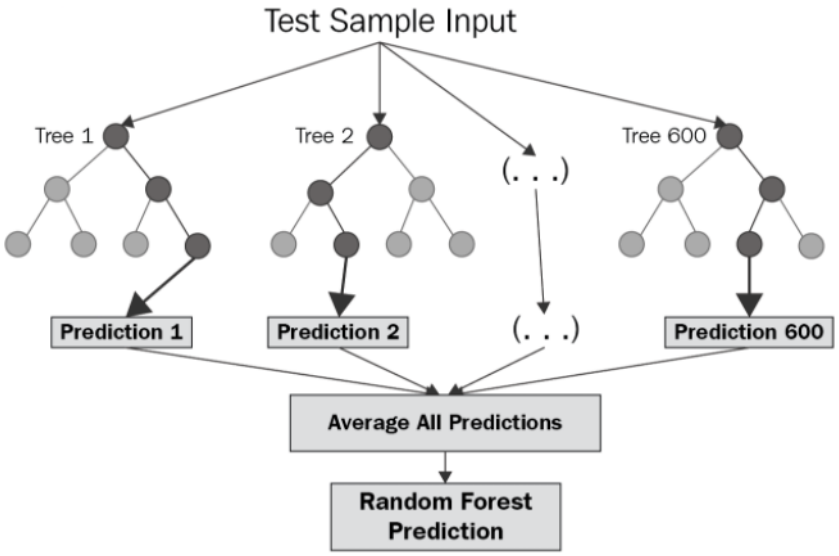

Figura 5: Funcionamiento del modelo RandomForestRegressor

2.4.5 Gradient boosting

Gradient Boosting es una generalización del algoritmo AdaBoost que permite emplear cualquier función de coste, siempre que esta sea diferenciable. La flexibilidad de este algoritmo ha hecho posible aplicar boosting a multitud de problemas (regresión, clasificación múltiple, etc.) convirtiéndolo en uno de los métodos de machine learning de mayor éxito.

Boosting se basa en ajustar secuencialmente múltiples modelos sencillos, llamados weak learners, de forma que cada modelo aprende de los errores del anterior, tratando de reducirlos, esto se logra ajustando los pesos de las muestras en cada iteración [19][20][21].

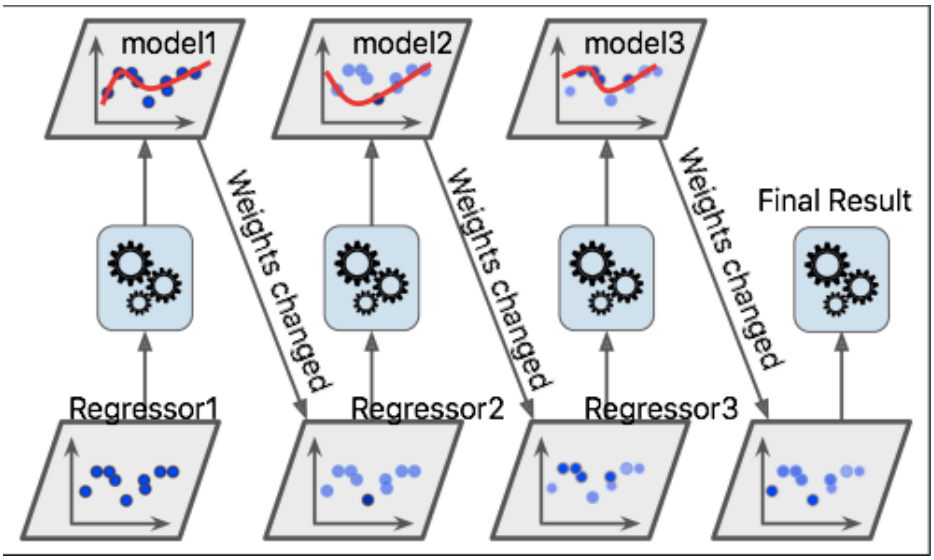

Figura 6: Funcionamiento del modelo GradientBoostingRegressor

2.4.6 Extreme gradient boosting

Este es un método para entrenar árboles de decisión más rápidos empleados en gradient boosting. La discretización y el binning (histogramas) pueden utilizarse para acelerar drásticamente el proceso de formación de árboles que se añaden a un conjunto y reducir la carga computacional. Esto se logra gracias a reducir el número de valores de las características de entrada continuas de decenas de miles a unos cientos. Y esta aproximación de los datos de entrada a menudo tiene poco impacto en la habilidad del modelo, si no la mejora. Esto permite que se puedan utilizar estructuras de datos eficientes para representar la subdivisión de los datos de entrada como los histogramas y el algoritmo de construcción del árbol se puede adaptar aún más para el uso eficiente de histogramas en la construcción de cada árbol.

La implementación extreme gradient boosting es órdenes de magnitud más rápido en comparación con la implementación gradient boosting regressor.

Para este modelo se ha utilizado dos implementaciones distintas dada por dos librerias, HistGradientBoostingRegressor desarrollada por Scikit-learn, y XGBRegressor desarrollada por XGBoost, la cual es una librería centrada en gradient boosting optimizada para ser más eficiente, flexible y portátil [22][23][24].

2.5 Evaluación de los resultados

En este punto vamos a comentar todas las métricas utilizadas para evaluar nuestros modelos, aunque la métrica realmente importante y con la que tuvimos que evaluar los modelos para la fase local del Datathon fue RMSE.

2.5.1 Coeficiente de determinación (R Cuadrado)

Es una medida que representa que tan bien describe el modelo de regresión los datos observados. Es decir, el porcentaje de varianza de la variable dependiente que el modelo es capaz de predecir a partir de las variables independientes. Cuando R-Cuadrado es 1 significa que existe una relación lineal perfecta entre x e y, cuando 0 < R-Cuadrado < 1 significa que la relación es más débil y no toda la variación de y queda explicada con la variación de x [25].

$$R^2 = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Donde $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ (media de los datos observados), $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$ (suma residual de cuadrados) y $\sum_{i=1}^{n}(y_i - \bar{y})^2$ (suma total de cuadrados) proporcional a la varianza de los datos.

2.5.2 Error absoluto medio (MAE)

Medida que muestra la media de los errores absolutos que existen entre las predicciones de nuestros modelos y el valor real, este error se le conoce como error absoluto o $\mathrm{L1}$. MAE es la media agregada de estos errores que nos permite reconocer el rendimiento de nuestro modelo sobre todo el conjunto de datos de una forma más clara [26].

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|$$

Donde $\hat{y}$ es la predicción de la instancia i, e y es el valor real de la instancia i.

2.5.3 Error cuadrático medio (MSE)

Al igual que con el MAE, el MSE es la media del error cuadrático que, como hemos dicho antes, existe entre las predicciones y los valores reales, a este error cuadrático se le conoce como $\mathrm{L2}$ [26].

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$$

Donde $\hat{y}$ es la predicción de la instancia i, e y es el valor real de la instancia i.

2.5.4 Raíz del error cuadrático medio (RMSE)

Esta medida es la raíz cuadrada del error que acabamos de comentar, esta nueva medida se utiliza para representar y entender mejor el error cuadrático medio, ya que la métrica que produce se encuentra en un rango de valores cercano al valor que se predice [26].

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2}$$

Donde $\hat{y}$ es la predicción de la instancia i, e y es el valor real de la instancia i.

2.6 Explicabilidad

Actualmente cuando realizamos modelos de aprendizaje automático, se suelen ver como una caja negra, cuyo funcionamiento es únicamente entendido por un grupo pequeño de personas y tienen una baja interpretabilidad. A la gente que utiliza estos modelos también les beneficiaría una explicación de como el modelo ha llegado a cierta predicción y que valores ha tomado en consideración para ello en mayor o menor medida.

Para solucionar esto cada vez se van implementando más herramientas dedicadas a la explicabilidad de los modelos y/o de los procesos de decisión basados en ellos.

Estas herramientas deben mostrar los datos que utilizan los modelos para sus predicciones, estimar bajo qué información o estimulo podría cambiar la predicción de un modelo, o simplificar los modelos hasta el punto de que los haga comprensibles para la audiencia que lo necesite.

Este campo de la explicabilidad de los modelos actualmente se le conoce como XAI, Explainable Artificial Intelligence, donde se tienen en cuenta tres factores fundamentales:

- La naturaleza del modelo a explicar, que puede hacerlo desde transparente como un árbol de decisión hasta totalmente opaco e ininteligible como un modelo de aprendizaje profundo.
- La audiencia objetivo ya que, dependiendo de esta, se buscará su explicabilidad con finalidades diferentes, desde entender mejor los modelos para extender sus capacidades a simplemente información en las implicaciones que tiene el uso de esta tecnología en cuestiones sensibles como la privacidad de los datos.
- La forma en la que dicha explicación se va a generar y va a ser presentada, ya que dependerá del grado de conocimiento de la audiencia y las posibilidades que brinde el modelo de ser explicado de una u otra forma, es decir, mostrar las explicaciones de una forma que una persona pueda entenderlo fácilmente sin perderse con tecnicismos.

2.6.1 SHAP

SHAP (SHapley Additive exPlanations) es un enfoque para explicar las predicciones de los modelos de machine learning. Se basa en la teoría de juegos y los valores de Shapley, que son un concepto fundamental en la teoría de juegos cooperativos. En resumen, SHAP proporciona

una forma coherente y sólida de descomponer el valor de una predicción de un modelo en contribuciones individuales de cada característica o atributo de entrada. Esto ayuda a comprender cómo cada característica contribuye a la predicción final de una manera justa y equitativa. A continuación, se explica cómo SHAP utiliza los valores de Shapley para la explicabilidad de los modelos de machine learning:

- Teoría de juegos y valores Shapley: Los valores de Shapley son un concepto de la teoría de juegos que se utiliza para asignar una contribución justa a cada jugador en un juego cooperativo. En el contexto de la explicabilidad de modelos, consideramos que las características del modelo son "jugadores" que contribuyen a la predicción del modelo. SHAP utiliza estos valores para determinar cuánto contribuye cada característica a una predicción específica.
- Exploración exhaustiva de todas las combinaciones posibles: SHAP considera todas las posibles combinaciones de características para calcular cómo cada característica contribuye al resultado. Esto implica calcular predicciones para todas las permutaciones posibles de características, lo cual es computacionalmente costoso, pero garantiza una asignación justa de contribuciones.
- Cálculo de contribuciones marginales: Para cada permutación, SHAP calcula la contribución marginal de cada característica al resultado. Estas contribuciones marginales se basan en cómo cambia la predicción cuando una característica se agrega o elimina de la combinación.
- Valores Shapley ponderados: SHAP combina las contribuciones marginales calculadas en todas las permutaciones posibles, utilizando los valores de Shapley para asignar un peso a cada contribución. Esto produce una descomposición del valor de la predicción en contribuciones individuales de cada característica.
- Visualización y comprensión: Finalmente, SHAP proporciona visualizaciones y gráficos que muestran cómo cada característica contribuye a una predicción específica. Estos gráficos son interpretativos y ayudan a los usuarios a comprender la importancia relativa de cada característica en una predicción.

En resumen, SHAP es una técnica que utiliza la teoría de juegos y los valores de Shapley para proporcionar explicaciones interpretables de las predicciones de los modelos de machine learning. Permite entender cómo cada característica influye en una predicción y proporciona una visión transparente de la toma de decisiones del modelo. Esto es especialmente útil en aplicaciones críticas donde la explicabilidad y la transparencia son fundamentales, como en la atención médica o las finanzas [27].

Capítulo 3

Preparación, Análisis Exploratorio y Preprocesamiento de los Datos

3.1 Introducción

En este capítulo explicaremos la preparación, análisis exploratorio y preprocesamiento que hemos realizado sobre nuestros conjuntos de datos por separado y el realizado sobre el conjunto de datos final.

3.2 Conjunto de Datos TRAIN

3.2.1 Preparación de Datos

Para el primer conjunto de datos lo primero de todo es cargar los datos y hacer una breve exploración viendo los datos y su tipo inicial.

ID	Columna	N.º de instancias	Tipo
0	CAMPAÑA	9601	Int64
1	IF_FINCA	9601	Int64
2	ID_ZONA	9601	Int64
3	ID_ESTACION	9601	Int64
4	ALTITUD	9547	Object
5	VARIEDAD	9601	Int64
6	MODO	9601	Int64
7	TIPO	9601	Int64
8	COLOR	9601	Int64
9	SUPERFICIE	9601	Float64
10	PRODUCCION	8526	Float64

Tabla 1: Tipo de las variables del conjunto TRAIN

Como se puede ver (Tabla 1: Tipo de las variables del conjunto TRAIN), la variable ALTITUD es cualitativa y la transformamos en numérica, sustituyendo los rangos por la media de estos y los valores únicos por estos mismos, pero de forma numérica en vez de forma nominal, para que nos sea más fácil trabajar con ella.

Tras esto visualizamos varias estadísticas sobre nuestros datos exceptuando los identificadores (Tabla 2: Diferentes estadísticas de las variables del conjunto TRAIN) y vemos que la variable SUPERFICIE tiene como valor mínimo 0.0, esto lo tomamos como un error de codificación de los valores ausentes y los codificamos como tal.

	Count	Mean	Std	Min	25%	50%	75%	Max
ALTITUD	9547	563.31	75.79	370.0	480.0	600.0	625.0	820.0
VARIEDAD	9601	45.48	23.81	4.0	17.0	52.0	59.0	94.0
MODO	9601	1.50	0.49	1.0	1.0	2.0	2.0	2.0
TIPO	9601	0.01	0.11	0.0	0.0	0.0	0.0	1.0
COLOR	9601	0.84	0.36	0.0	1.0	1.0	1.0	1.0
SUPERFICIE	9601	0.69	1.78	0.00072	0.0	0.0	0.56	26.85
PRODUCCION	8526	9209.19	13268.81	0.71	1940.0	4620.00	11137.50	177520.0

Tabla 2: Diferentes estadísticas de las variables del conjunto TRAIN

Tras estas modificaciones comprobamos la ausencia de datos de nuestro conjunto (Tabla 3: Numero de valores ausentes por campaña del conjunto TRAIN) y observamos que tenemos datos ausentes en las variables anteriormente tratadas, ALTITUD y SUPERFICIE, a parte de la variable objetivo cuyas instancias son sobre las que se quiere conocer la producción final.

CAMPAÑA	ALTITUD	SUPERFICIE	PRODUCCION
14	9	1148	0
15	9	1116	0
16	3	1079	0
17	4	1017	0
18	6	1061	0
19	6	1055	0
20	6	13	0
21	6	20	0
22	5	9	1075

Tabla 3: Numero de valores ausentes por campaña del conjunto TRAIN

3.2.2 Análisis Exploratorio de Datos

Inicialmente en el análisis exploratorio representamos los valores ausentes con un heatmap, a continuación, observamos la correlación de todas las variables exceptuando los identificadores

con un pairgrid de scatterplots y kdeplots (Figura 7) y un heatmap (Figura 8) para visualizarlo correctamente, donde podemos observar una alta correlación entre las variables SUPERFICIE y PRODUCCION.

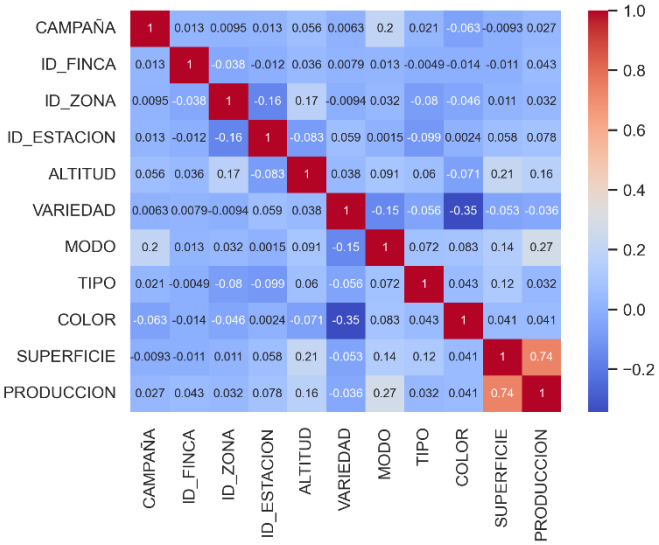

Figura 7: Mapa de calor de las variables del conjunto TRAIN

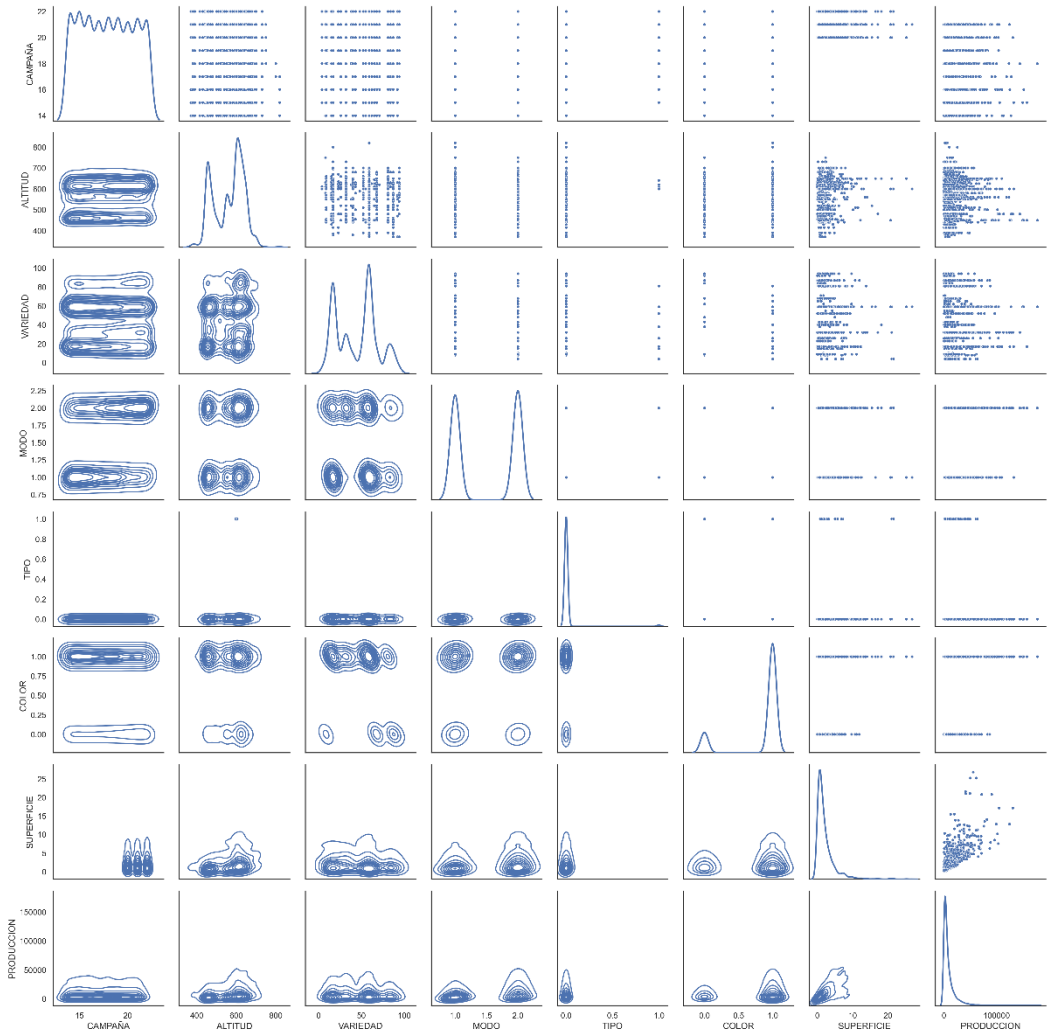

Figura 8: Pairgrid de scatterplots y kdeplots para mostrar la relación de unas variables con otras

A continuación, visualizamos la distribución de todas las variables e igualmente excluyendo los identificadores, con histplots para ver la distribución tanto de las variables continuas como de las categóricas (Figura 9) donde vemos que la variable binaria MODO esta más equilibrada que las variables binarias TIPO y COLOR donde se puede observar un claro desbalanceo. En cuanto a las variables VARIEDAD y ALTITUD se ve que existen ciertos casos más comunes que el resto. Por último, las variables PRODUCCION y SUPERFICIE tiene una forma bastante parecida con una gran asimetría hacia la izquierda.

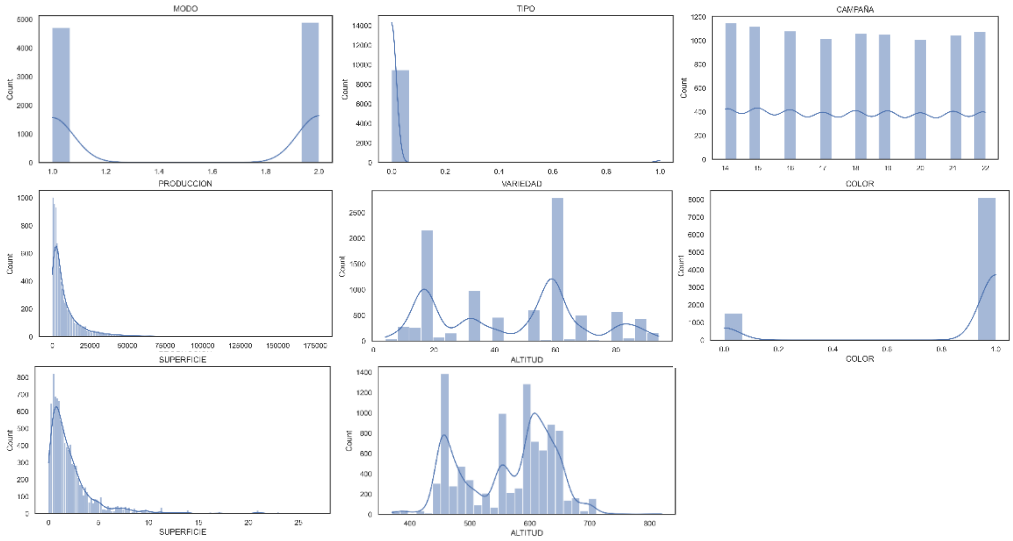

Figura 9: Histplots de las variables del conjunto TRAIN

También utilizamos gráficos de barras (Figura 10) con las variables categóricas por campañas para observar un claro desbalanceo en algunas de ellas, como puede ser el caso de COLOR y TIPO y también observamos como cambia con el avance de las temporadas la tendencia de MODO es decir como inicialmente se utilizaba mayoritariamente el modo 1 pero que el modo 2 va aumentando hasta que en las últimas campañas es el modo 2 el más utilizado. Por último, también se puede ver que la variedad 59 es la más cultivada en todas las campañas.

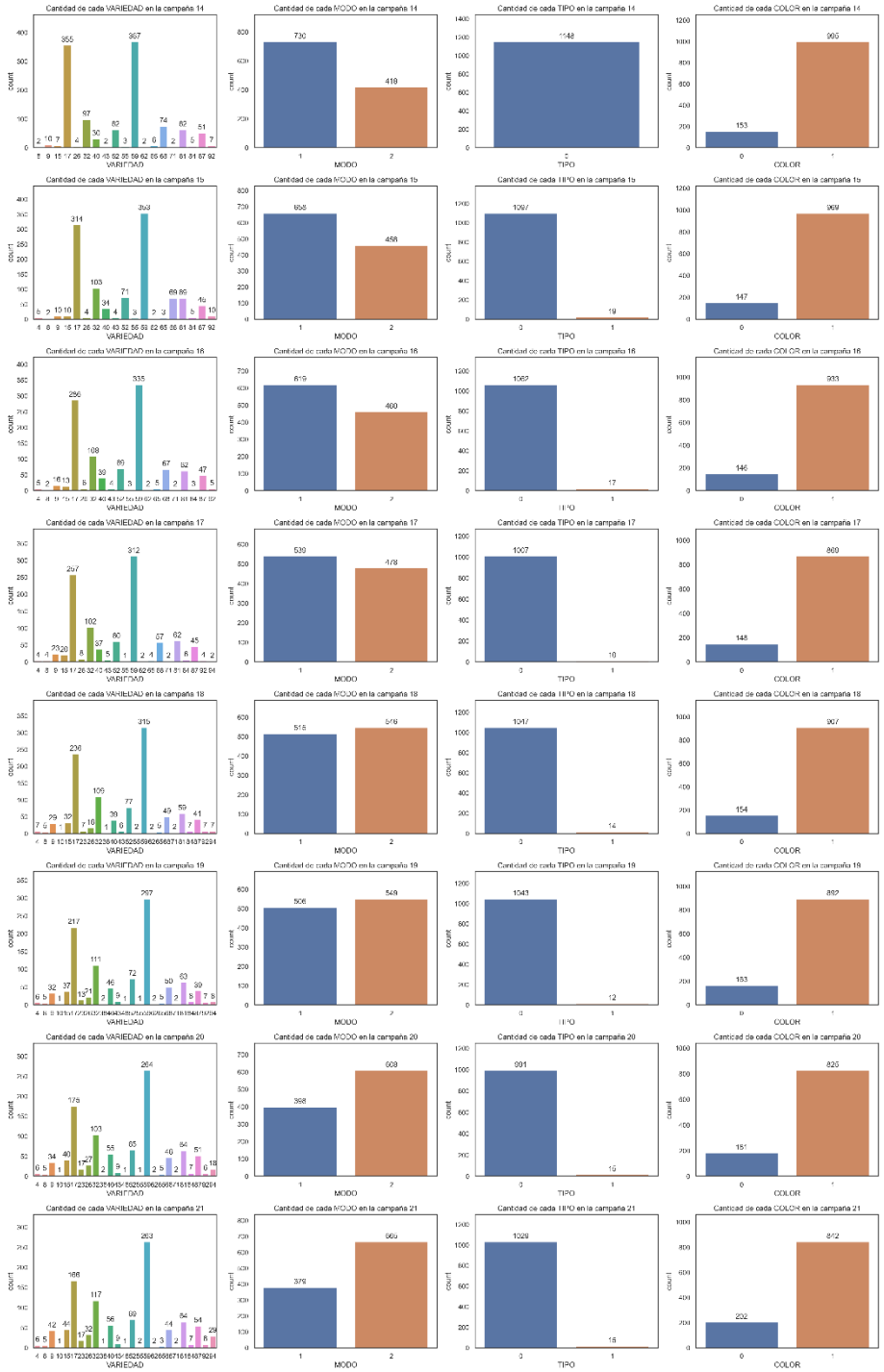

Figura 10: Gráficos de barras para la distribución de las variables VARIEDAD, MODO, TIPO y COLOR por campaña

Para terminar con la visualización de este conjunto de datos utilizamos lineplots visualizando así la producción a lo largo de las campañas. Generamos una gráfica dependiendo de cada una de las variables que hemos visualizado antes (Figura 11), donde se puede ver claramente el desbalanceo de las mismas y como la producción es mayor en las clases mayoritarias de esas variables. Finalmente representamos la producción a lo largo de las campañas de forma general

(Figura 12), viendo como hubo un desplome de la producción en la campaña 17 y un gran aumento en la campaña 18.

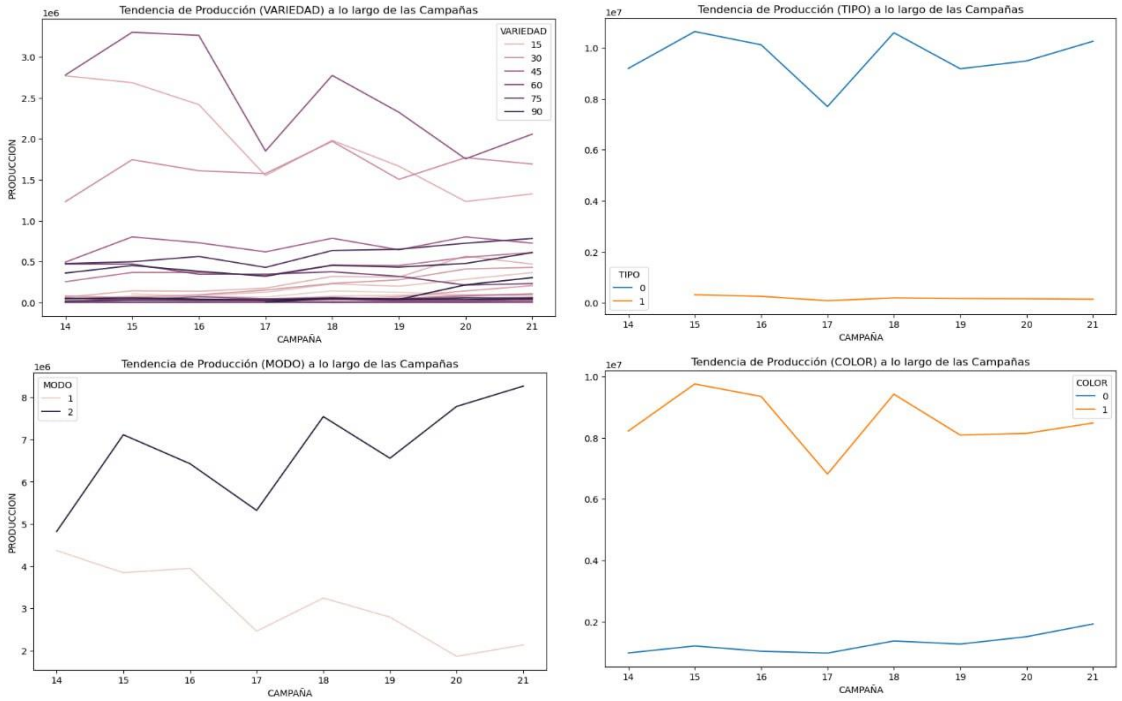

Figura 11: Lineplots que describen la producción por campaña, teniendo en cuenta las variables VARIEDAD, TIPO, MODO y COLOR

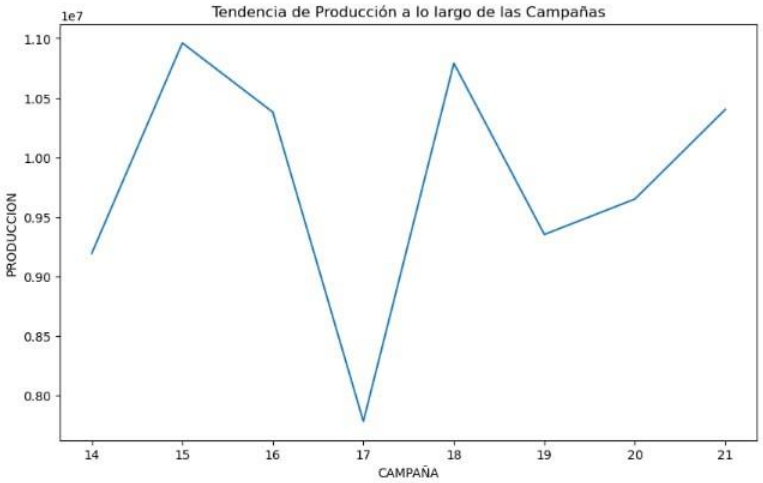

Figura 12: Lineplot de la producción por campaña

3.2.3 Preprocesamiento de Datos

Tras la visualización y análisis exploratorio de este conjunto de datos vamos a realizamos varias técnicas de preprocesado para la imputación de datos ausentes:

- Para la imputación de datos para la variable ALTITUD utilizando la mediana de los casos con el mismo ID_ESTACION, separando el conjunto de datos por campaña para no utilizar datos de campañas del "futuro", para campañas del "pasado".
- Para imputar los datos de la variable SUPERFICIE (la más problemática con 6518 datos ausentes), utilizamos la combinación de las variables ID_FINCA, VARIEDAD, MODO, TIPO y COLOR. Si tenemos ciertas instancias donde sí que tenemos la información sobre la superficie y coinciden estas variables con instancias donde no tenemos la información, imputamos la media de las instancias que sí que la tengan, si son más de una. En caso de que solo coincida una instancia se imputa el mismo valor y en los casos donde no existan coincidencias se deja sin imputar para tratarlo más adelante. Este proceso lo repetimos con varias combinaciones (las que más coincidencias generan):
    - ID_Finca, VARIEDAD, MODO, TIPO, COLOR
    - ID_Finca, VARIEDAD, TIPO, COLOR
    - ID_Finca, TIPO, COLOR
    - ID_Finca, TIPO, MODO
    - ID_Finca, MODO, COLOR
    - ID_Finca, TIPO
    - ID_Finca, COLOR
    - ID_Finca

  Los datos que se imputan son datos del pasado utilizando datos del futuro, esto es una clara fuga de datos, pero dado que no existe información de la superficie para ninguna de las 6 primeras campañas, no encontramos otra alternativa.
- Por último, se realizan varios experimentos para la imputación de los valores ausentes restantes de la variable SUPERFICIE utilizando diversas técnicas:
    - Imputación aleatoria
    - Imputación con la media
    - Imputación con la moda
    - Imputación con la mediana
    - Deterministic regression imputation
    - Stochastic regression imputation

En cada experimento también se hace por separado las imputaciones de cada campaña para que, al igual que antes, no utilizar datos futuros con los pasados. Después de cada experimento comprobamos visualizando con histplots la distribución de la variable antes y después de la imputación para asegurarnos de que se ha hecho correctamente y no se han generado valores anómalos. Tras esto suplimos con todos los valores ausentes de la variable SUPERFICIE seleccionando el resultado de la técnica stochastic regression imputation.

Finalmente hacemos una visualización de las variables (Figura 13) modificadas para comprobar su distribución y ver que no se han generado datos anómalos.

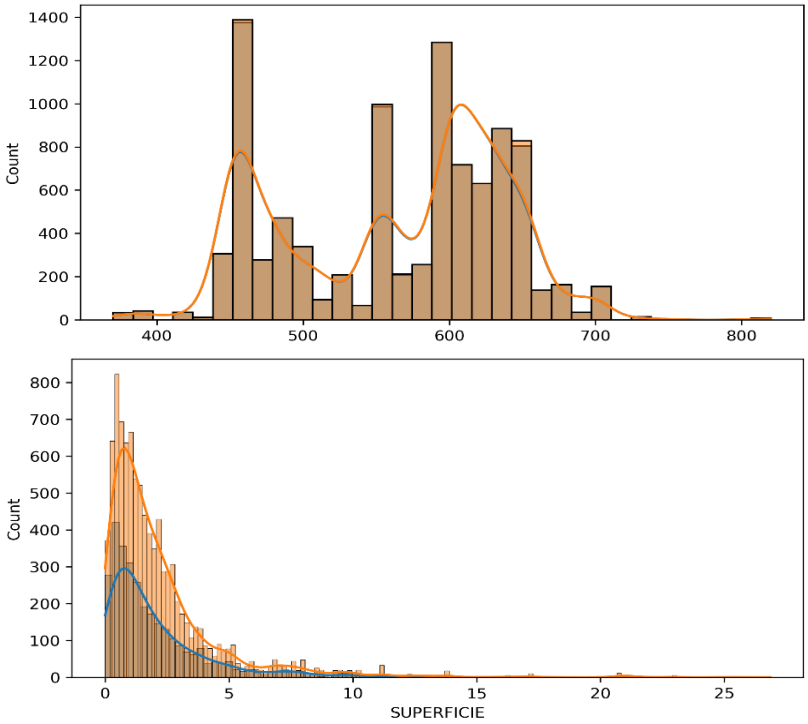

Figura 13: Distribución de las variables preprocesadas

Como se puede observar (Figura 13), la distribución de las variables antes y después de la imputación es similar, sin detectar valores anómalos, viendo en la variable SUPERFICIE esa asimetría hacia la izquierda, y en cuanto a la variable ALTURA, se ve como la mayoría de instancias tienen una altura cercana a los 450, 550 y 600 metros.

Finalmente guardamos el conjunto de datos final en un archivo ".csv" para su uso futuro.

3.3 Conjuntos de Datos METEO y ETO

3.3.1 Preparación de Datos

Para el conjunto de datos METEO, inicialmente cargamos los datos y al igual que antes una pequeña exploración, viendo los datos y su tipo, observando que el tipo de todas las variables es float64, exceptuando validTimeUtc de tipo datetime64 e ID_ESTACION que es de tipo int64. Tras esto transformamos la columna validTimeUtc en 4 columnas distintas con el año, el mes, el día y la hora y codificamos los valores "-2" y "-1" de la variable uvIndex como valores ausentes que es a lo que hacen referencia. Mantenemos únicamente las columnas interesantes, es decir las que tienen datos horarios y mostramos unas estadísticas sobre ellos (Tabla 4: Diferentes estadísticas de las variables del conjunto METEO.

	Count	Mean	Std	Min	25%	50%	75%	Max
precip	1223640	0.066	0.58	0.0	0.0	0.0	0.0	60.3
pressureC	1223640	0.0042	1.12	-6.2	-0.8	0.0	-0.8	5.7
pressureM	867520	1017.78	6.36	988.6	1014.0	1017.5	1021.8	1037.0
relativeH	1223660	66.91	22.12	6.5	49.6	68.4	86.0	100
snow	1223640	0.00073	0.035	0.0	0.0	0.0	0.0	5.0
temperature	1223640	15.42	8.01	-7.9	9.3	14.7	20.9	43.2
temperatureC	1223280	-0.047	1.50	-10.4	-0.8	0.1	0.8	8.9
temperatureM	1223560	21.73	7.94	0.1	15.1	21.0	28.4	43.2
temperaturem	1223560	1.006	6.08	-7.9	5.3	9.7	15.3	28.0
temperatureD	1223640	8.22	6.03	-17.6	4.1	8.2	12.9	23.3
temperatureF	1223640	14.74	8.80	-11.3	7.9	14.2	20.9	43.2
uvIndex	1223640	1.35	2.35	0.0	0.0	0.0	2.0	10.0
visibility	1223640	12.86	2.87	0.1	12.11	13.48	14.58	16.09
windDirection	867520	184.34	101.35	0.0	70.0	210.0	270.0	350.0
windSpeed	1223660	10.49	6.85	0.0	5.4	8.6	14.0	58.7
ID_ESTACION	1223660	9.50	5.76	0.0	4.75	9.5	14.25	19.0

Tabla 4: Diferentes estadísticas de las variables del conjunto METEO

Para finalizar la preparación de este conjunto de datos, buscamos la existencia de datos ausentes (Tabla 5: Numero de datos ausentes por variable y año del conjunto METEO. En este caso exceptuando las variables relativeHumidity y windSpeed el resto de las variables que hemos mantenido presentan valores ausentes.

año	precip	...	visibility	pressureM	temperatureC	temperatureM	temperaturem	wind
15	0	...	0	87760	140	100	100	87760
16	20	...	20	174020	20	0	0	174020
17	0	...	0	94360	100	0	0	94360

18	0	…	0	0	60	0	0	0
19	0	…	0	0	60	0	0	0
20	0	…	0	0	0	0	0	0
21	0	…	0	0	0	0	0	0
22	0	…	0	0	0	0	0	0

Tabla 5: Numero de datos ausentes por variable y año del conjunto METEO

Para el conjunto de datos ETO, el proceso es muy parecido al de METEO. Cargamos el conjunto de datos, transformamos la variable date en tres columnas diferentes que representan el día, mes y año de cada instancia y mantenemos únicamente las variables que contengan los datos diarios, es decir las que tienen en su nombre LocalDay. A continuación, mostramos el tipo de variables que tenemos, y vemos que la mayoría de variables son del tipo float64, excepto algunos máximos y mínimos que son de tipo int64. También mostramos algunos estadísticos (Tabla 6: Diferentes estadísticas de las variables del conjunto ETO, donde no observamos nada inusual.

	Count	**Mean**	**Std**	**Min**	**25%**	**50%**	**75%**	**Max**
ID_ESTACION	51180	9.50	5.76	0.0	4.75	9.50	14.25	19.00
DewpointLocalDayAvg	51180	281.33	5.77	261.0	277.0	281.2	286.0	295.0
DewpointLocalDayMax	51180	283.78	5.69	264.0	280.0	283.7	288.3	296.4
DewpointLocalDayMin	51180	278.84	6.04	256.0	274.7	279.0	283.3	294.0
…	…	…	…	…	…	…	…	…
WindSpeedLocalDayAvg	51180	2.91	1.50	0.5	1.90	2.50	3.40	12.90
WindSpeedLocalDayMax	51180	4.81	2.03	1.0	3.40	4.40	5.70	16.30
WindSpeedLocalDayMin	51180	1.32	1.08	0.0	0.60	1.0	1.60	10.80

Tabla 6: Diferentes estadísticas de las variables del conjunto ETO

Por último, comprobamos la existencia de datos ausentes (Tabla 7: Numero de datos ausentes por variable y año del conjunto ETO, estos se encuentran en las variables EvapotranspirationLocalDayAvg, EvapotranspirationLocalDayMax, EvapotranspirationLocalDayMin, GlobalHorizontalIrradianceLocalDayAvg, GlobalHorizontalIrradianceLocalDayMax, GlobalHorizontalIrradianceLocalDayMin, GustLocalDayAvg, GustLocalDayMax, GustLocalDayMin, MSLPLocalDayAvg, MSLPLocalDayMax, MSLPLocalDayMin.

Año	Evapo Avg	Evapo Max	Evapo Min	Global Avg	Global Max	Global Min	Gust Avg	Gust Max	Gust Min	MSLP Avg	MSLP Max	MSLP Min
15	3720	3720	3720	3720	3720	3720	2540	2540	2540	3720	3720	3720
16	7320	7320	7320	7320	7320	7320	5095	5095	5095	7320	7320	7320
17	7300	7300	7300	7300	7300	7300	5777	5777	5777	3940	3940	3940
18	7300	7300	7300	7300	7300	7300	5720	5720	5720	0	0	0

19	7300	7300	7300	7300	7300	7300	4787	4787	4787	0	0	0
20	1260	1260	1260	1260	1260	1260	5101	5101	5101	0	0	0
21	380	380	380	380	380	380	5239	5239	5239	0	0	0
22	0	0	0	0	0	0	2506	2506	2506	0	0	0

Tabla 7: Numero de datos ausentes por variable y año del conjunto ETO

3.3.2 Análisis Exploratorio de Datos

Para ambos conjuntos de datos, METEO y ETO, se realiza un proceso similar al realizado con el conjunto de datos TRAIN, primero se observan las variables que tiene valores ausentes con un heatmap, tras lo cual observamos la correlación de todas las variables (utilizando únicamente las medias/modas de cada variable en el caso de ETO) con un pairgrid de scatterplots y kdeplots y un heatmap (Figura 14) para visualizarlo correctamente. En estos se puede ver claramente una alta correlación entre las variables que muestran información sobre la temperatura, sensación térmica y radiación ultravioleta.

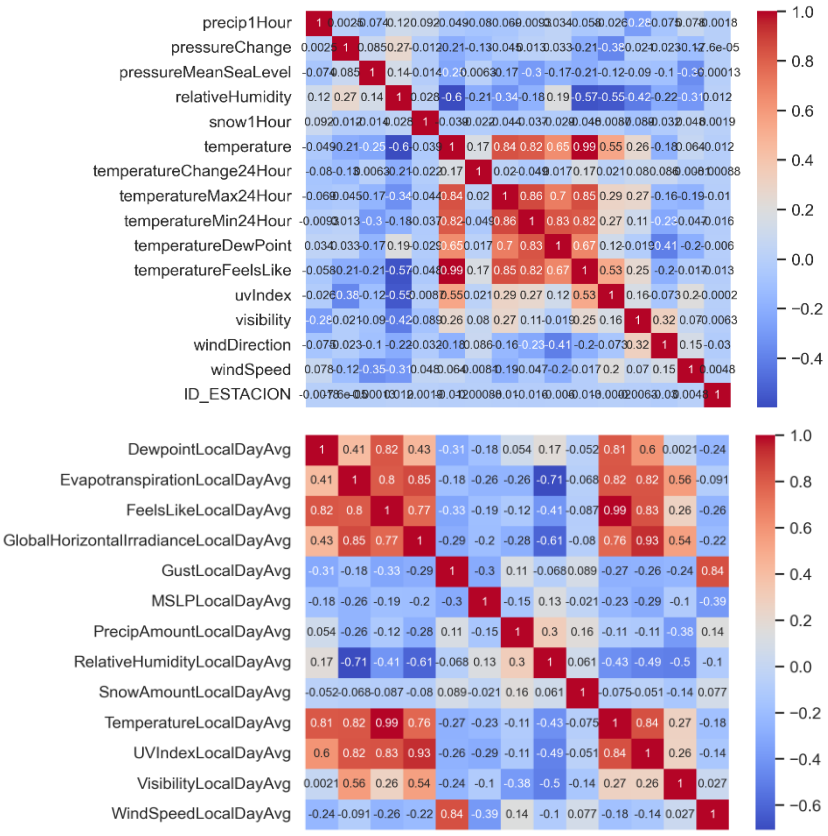

Figura 14: Heatmap de los conjuntos de datos METEO y ETO respectivamente

Por último, se muestra la distribución de todas las variables y medidas también con histplots para poder observar los cambios que existen entre las distintas medidas, mínimo, máximo, media, etc. (Figura 15). En la mayoría de casos tienen una distribución similar las distintas

medidas para una variable, aun así, sí que se ven diferencias en algunos casos como en relativeHumidity.

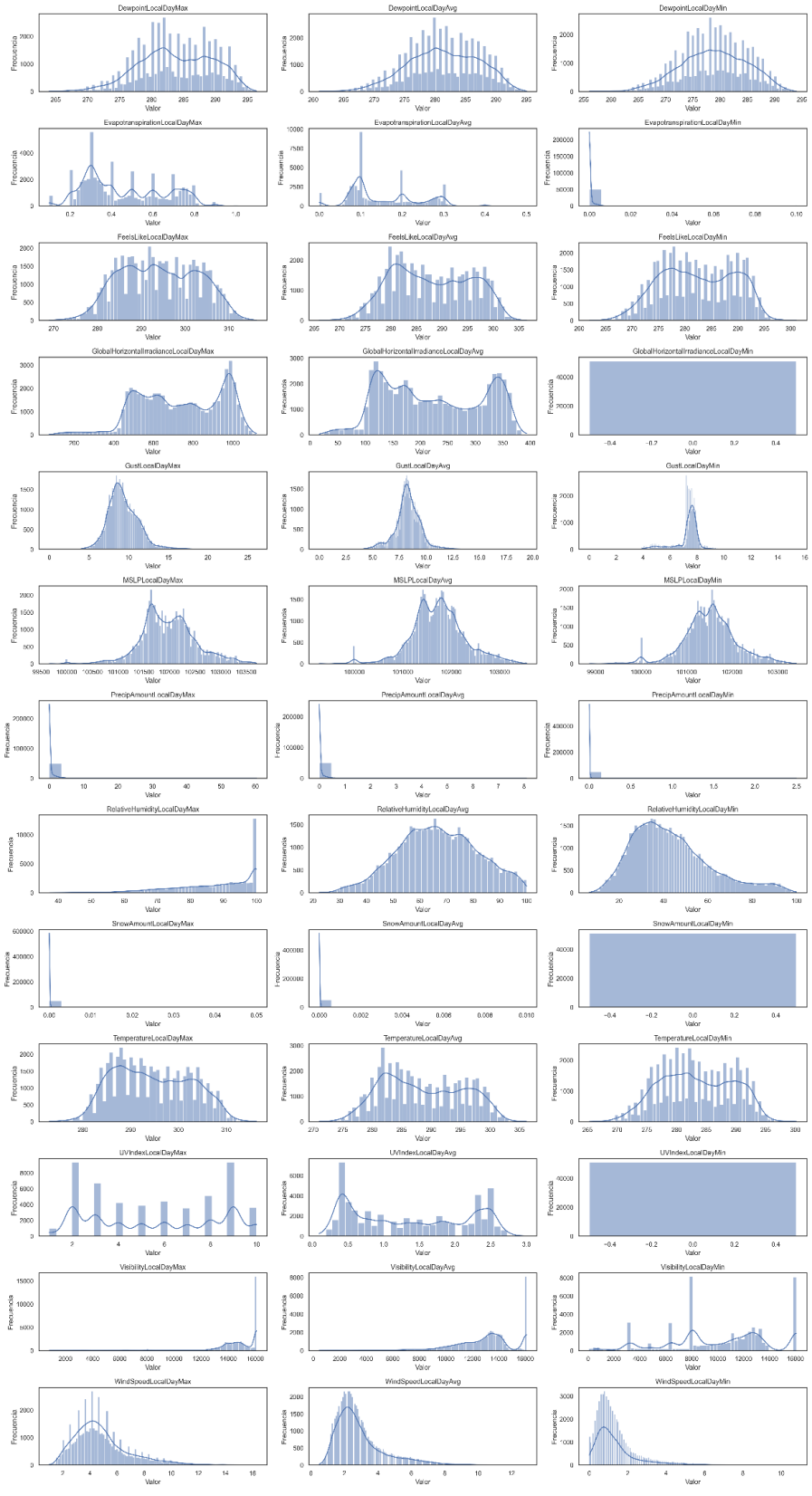

Figura 15: Ejemplo de los histplots que representan la distribución de las variables de ETO

3.3.3 Preprocesamiento de Datos

Dado que en estos conjuntos de datos tenemos variables temporales para conocer hora, día, mes y año, vamos a tratarlo como una serie temporal y a predecir e imputar los datos ausentes con Prophet (Figura 16), el modelo de Meta para la previsión de datos. El único problema que tiene este modelo es que únicamente predice datos futuros, pero realmente los datos ausentes de nuestros conjuntos de datos son del pasado, por lo que para solventar este problema se decide probar su funcionamiento invirtiendo las fechas de los conjuntos de datos completos y así realmente los datos ausentes que tendríamos serían los de un intervalo de tiempo futuro y obteniendo así unas buenas predicciones por parte de Prophet e imputando todos los datos ausentes.

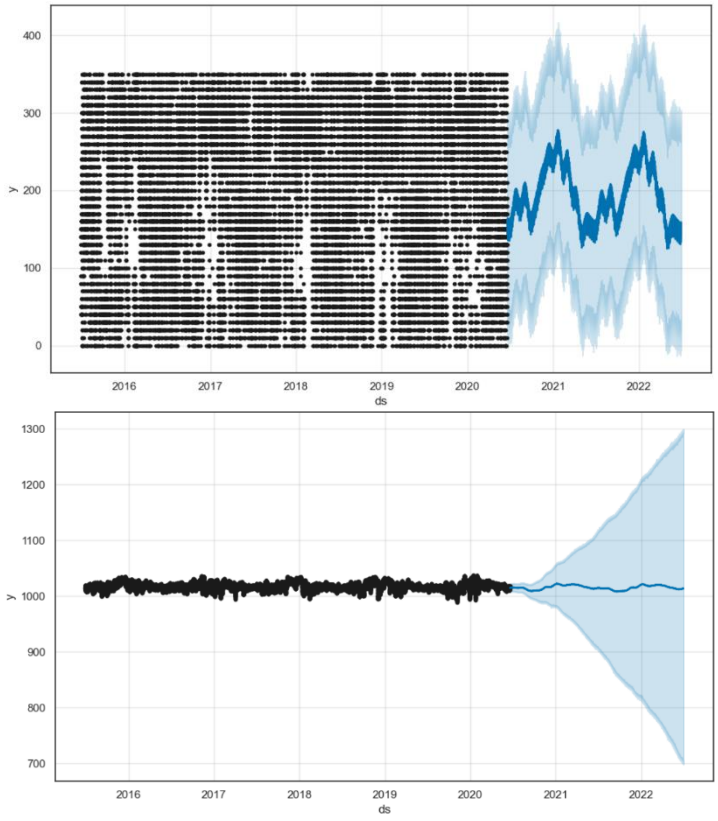

Figura 16: Graficas visuales de la predicción de datos de Prophet

Tras esto comprobamos la distribución de las variables de los conjuntos de datos con las nuevas predicciones como se hizo con el conjunto de datos TRAIN utilizando histogramas y vemos que generalmente son correctas. Los casos que debemos corregir en METEO son con la variable uvIndex ya que esta variable realmente es categórica y solo debe poder tomar valores enteros del 0 al 10, y como las predicciones de Prophet generan valores que no encajan se decide sustituirlos por la moda de valores de uvIndex que ha tenido en ese mismo mes a esa misma hora. También se deben modificar los valores mínimos de otras muchas de variables, ya que en

numerosas ocasiones generaba valores negativos en variables que únicamente podrían ser positivas, por lo que decidimos sustituir estos valores anómalos por los valores mínimos que tenía cada variable antes de utilizar Prophet. Este mismo caso se realiza con el conjunto de datos ETO donde debemos sustituir los mínimos de nuevo para no generar datos anómalos.
Tras realizar todo esto y a causa del tamaño del conjunto de datos METEO (1223660 instancias) se decide reducirlo, en vez de tener instancias por cada hora a tener instancias diarias calculando los siguientes datos para cada variable, el mínimo, el máximo, la media y mediana o la moda, y en algunos casos calculamos también el sumatorio.
En el caso del conjunto de datos ETO se han utilizado datos diarios desde un principio, por lo que tras la imputación y preprocesado necesario este conjunto de datos tiene un número de instancias razonable, 51140 instancias.

3.4 Conjunto de Datos Final

Para finalizar unimos los dos conjuntos de datos climáticos METEO y ETO utilizando como claves comunes ID_ESTACION, Day, Month, Year, y añadimos la variable Campaña que coincide con los últimos dos dígitos de cada año, 16, 17, 18, etc. Tras esto borramos las instancias referentes a las campañas 14 y 15 ya que no tenemos datos de las campañas completas.
Una vez tenemos el conjunto de datos climatológicos completo se aprecia que tenemos datos repetidos ya que algunas medidas que calculamos en su momento ya venían dadas en el conjunto ETO de datos agregados. Por lo que descartamos las variables que están repetidas manteniendo únicamente una de ellas. Tras esto calculamos la varianza y la desviación estándar de todas las variables y descartamos las que tienen varianza 0 y desviación estándar 0 ya que no proporcionan información a los modelos. Finalmente calculamos la tabla de correlación de nuestro conjunto de datos y la visualizamos con un mapa de calor, descartando las variables con una correlación superior al 80% de correlación (Figura 17).
Para no generar un conjunto de datos tan grande agrupamos por mes y “pivotamos” la tabla manteniendo como índices la campaña y la estación, generando una nueva variable por cada mes, para cada una de las variables que ya teníamos, reduciendo la cantidad de instancias, pero aumentando el número de variables predictoras. A continuación, renombramos las variables con nombres más descriptivos y unimos este conjunto de datos al conjunto TRAIN eliminando previamente los datos de las campañas 14 y 15 como hemos hecho antes con el resto de conjuntos de datos. Tras esto eliminamos todas las columnas referentes a los meses julio, agosto, septiembre, octubre, noviembre, diciembre como se explica en la descripción del

problema, “por ello se debe tener en cuenta que no se pueden usar datos meteorológicos posteriores al 30 de junio de cada año para estimar la producción del mismo”.

Por último, creamos un pipeline con las técnicas de preprocesado que necesitamos aplicar al conjunto de datos previamente a utilizarlo en cada modelo, simplemente dos pasos:

- Borrar la columna Campaña para que no exista dependencia de los resultados con la campaña de la instancia.
- Aplicar OneHotEncoder a las variables categóricas generando así una columna por cada categoría de cada variable, pasando de tener un conjunto de datos de 149 columnas a uno de 1294, estas variables son los identificadores, la variedad, el modo, el tipo y el color.

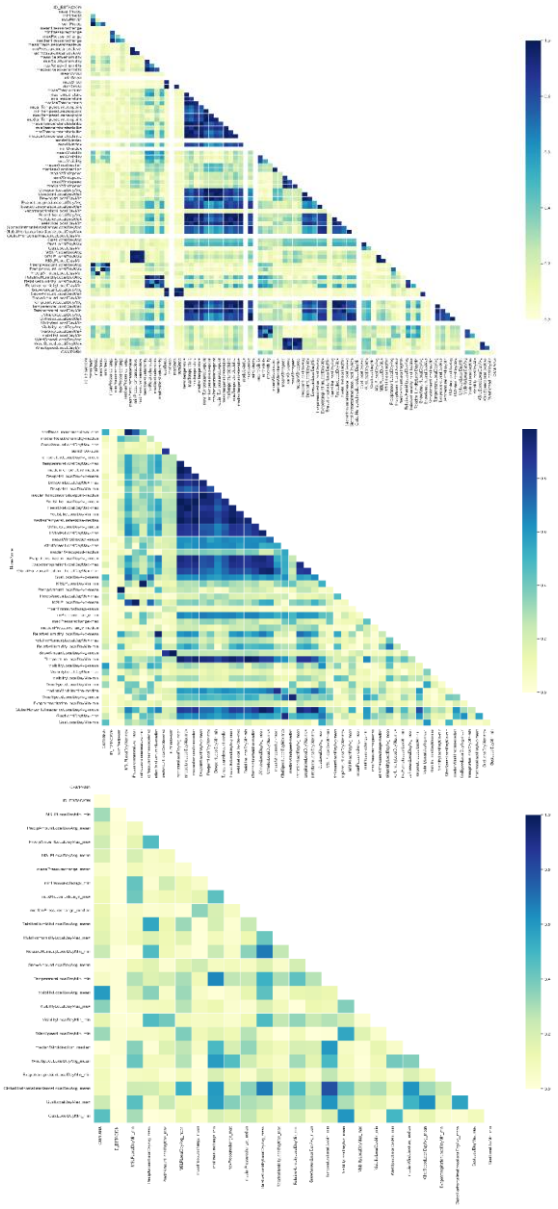

Figura 17: Mapas de calor mostrando la correlación entre las variables del conjunto de datos. El primero con el conjunto de datos completo, el segunda tras quitar variables repetidas y variables con varianza 0 y el ultimo tras quitar variables con alta correlación.

Capítulo 4

Experimentos, Resultados y Explicabilidad

4.1 Introducción

A continuación, vamos a mostrar los resultados de los experimentos realizados, los modelos utilizados y sus tiempos de ejecución. Los resultados mostrados, en este caso RMSE y R cuadrado son los obtenidos con el conjunto de Test y los tiempos son los utilizados para todo el proceso llevado a cabo con cada modelo.

4.2 Características del entorno

El entorno en el que se ha ejecutado la libreta para obtener los resultados mostrados más adelante tiene las siguientes características:

- Sistema Operativo: Windows 11 Pro versión 22H2 de 64 bits
- Procesador: 13th Gen Intel(R) Core(TM) i7-13700K 3.40 GHz de 16 núcleos y 24 hilos
- Gráfica: Gigabyte GeForce RTX 4070Ti Aero OC 12GB GDDR6X DLSS3
- Almacenamiento: RAM: 32,0 GB DDR5 6200MHz SSD: 2TB Gen4 PCIe x4 NVMe M.2

4.3 Modelos utilizados

Los modelos utilizados en todos los experimentos son los desarrollados en el punto 2.4, en el Capítulo 2 del documento. Estos son:

- Ridge
- KNeighbors Regressor
- DecisionTree Regressor
- RandomForest Regressor
- GradientBoosting Regressor
- HistGradientBoosting Regressor
- XGB Regressor

Para todos los modelos y en todos los experimentos hemos seguido el mismo proceso:

- Creación del pipeline con el preprocesamiento descrito antes, añadimos en el caso del modelo de los vecinos más cercanos un paso de escalado de las variables con MinMaxScaler, y el modelo a utilizar.
- Selección de los mejores hiperparámetros para cada modelo entre los definidos en parameters con RandomizedSearch el cual nos da una gran mejora respecto al tiempo que necesita para seleccionar los mejores parámetros frente a GridSearch.
- Tras esto con cada modelo y sus mejores hiperparámetros llamamos a la función show_results la cual nos da los resultados con los datos de entrenamiento, la media de una validación cruzada de cinco particiones también con los datos de entrenamiento, y los resultados con los datos de test. Tanto de los datos de entrenamiento como con los de testeo obtenemos los errores absolutos medios (MAE), los errores cuadráticos medios (MSE) y la raíz de los errores cuadráticos medios (RMSE) aunque la forma de evaluar los modelos lo explicaremos más adelante.

4.4 Experimentos y Resultados

Todos los experimentos están desarrollados sobre el mismo conjunto de datos, los experimentos 1, 3, 4 y 5 utilizan la misma división del conjunto de datos, las campañas de la 16 a la 19 para Train y la 21 y la 22 para Test. En los experimentos 2, 6, 7 y 8 la división de Train y Test se realiza de forma aleatoria utilizando el mismo tamaño del conjunto de Test y el mismo generador aleatorio.

Para mostrar los resultados de los experimentos 3,4,5,6,7,8 se ha utilizado las medias ponderadas de cada resultado y la suma de los tiempos.

$$RMSE_{Total} = \frac{\sum_{i=1}^{X} RMSE_{Estacion\ i} \times N\ Instancias_{Estacion\ i}}{\sum_{i=1}^{X} N\ Instancias_{Estacion\ i}}$$

$$R^2{}_{Total} = \frac{\sum_{i=1}^{X} R^2{}_{Estacion\ i} \times N\ Instancias_{Estacion\ i}}{\sum_{i=1}^{X} N\ Instancias_{Estacion\ i}}$$

$$Tiempo_{Total} = \sum_{i=1}^{X} Tiempo_{Estacion\ i}$$

4.4.1 Experimento 1

En el primer experimento, se dividió el conjunto de datos en conjuntos de Train y Test teniendo en cuenta la campaña en la que se tomaron las medidas de cada instancia. Esta división por campaña se realizó para evitar una fuga de datos entrenando el modelo con datos del futuro para predicciones del pasado, asegurando que los datos de prueba sean independientes de los datos de entrenamiento. Luego, se entrenaron y evaluaron los modelos de regresión comentados anteriormente utilizando las métricas de rendimiento R cuadrado y RMSE, también se tienen en cuenta los tiempos de la ejecución completa para cada modelo.

ID	Modelo	Parámetros	Subconjunto	R cuadrado	RMSE	Tiempo (s)
Ridge_All	Ridge	ERROR	Train	ERROR	ERROR	2370
			Validación			
			Test			
KNN_All	KNeighbors Regressor	Weights: 'uniform'	Train	0,495996404	9343,803235	543,3114185
		N_neighbors: 7	Validación	0,112279041		
		Metric: 'euclidean'	Test	0,503011002	9542,907751	
DT_All	DecisionTree Regressor	Min_samples_split: 2	Train	0,745453521	6640,341487	218,8076203
		Criterion: 'squared_error'	Validación	0,575989036		
		Min_Samples_leaf: 10	Test	0,470666674	9848,542353	
		Cpp_alpha: 0.0				
		Max_depth: 50				
RF_All	RandomForest Regressor	N_estimators: 100	Train	0,870124416	4743,191114	7472,282191
		Max_depth: 15	Validación	0,685333958		
		Min_samples_split: 2	Test	0,641341313	8106,773054	
		Criterion: 'poisson'				
		Min_Samples_leaf: 3				
		Cpp_alpha: 0.3				
GB_All	GradientBoosting Regressor	'modelo__n_estimators':2000	Train	0,964389377	2483,685798	18002,28846
		'modelo__min_samples_split':2	Validación	0,733115693		
		'modelo__min_samples_leaf': 10	Test	0,656137194	7937,796111	
		'modelo__max_depth': 50				
		'modelo__loss': 'huber'				
		'modelo__learning_rate': 0.1				
		'modelo__criterion': 'friedman_mse'				
		'modelo__ccp_alpha': 0.1				
		'modelo__alpha': 0.95				
HGB_All	**HistGradientBoosting Regressor**	**'modelo__min_samples_leaf':3**	**Train**	**0,8816419**	**4527,994178**	**2839,482412**
		'modelo__max_leaf_nodes':15	**Validación**	**0,722573025**		
		'modelo__max_iter':100	**Test**	**0,61046274**	**8448,543421**	
		'modelo__max_depth':20				
		'modelo__max_bins':155				
		'modelo__loss':'squared_error'				
		'modelo__learning_rate':0.1				
		'modelo__l2_regularization':0.2				
XGB_All	XGB Regressor	'modelo__subsample': 0.8	Train	0,958476276	2681,976777	3354,16223
		'modelo__objective':'reg:tweedie'	Validación	0,689447411		
		'modelo__n_estimators': 500	Test	0,378699642	10669,84573	
		'modelo__min_child_weight': 5				
		'modelo__max_depth': 4				

		'modelo__gamma': 0.5 'modelo__eval_metric': 'rmse' 'modelo__eta': 0.4 'modelo__colsample_bytree': 0.6 'modelo__booster': 'gbtree'				

Tabla 8: Resultados experimento 1

En la Tabla 8: Resultados experimento 1 se puede observar como el modelo que mejores resultados obtiene es Gradient Boosting, pero que tiene un tiempo de ejecución excesivamente largo, por lo que vamos a seleccionar como mejor modelo Histogram Gradient Boosting, ya que estos resultados no empeoran en gran medida, pero el tiempo se reduce un 85% con respecto al mejor modelo.

4.4.2 Experimento 2

En el experimento 2 se utiliza el conjunto de datos completo dividiendo Train y Test de forma aleatoria. Para comprobar cómo afectaría en los modelos que no hubiera una condición temporal. Luego, se entrenaron y evaluaron los modelos de regresión comentados anteriormente utilizando las métricas de rendimiento R cuadrado y RMSE, también se tienen en cuenta los tiempos de ejecución.

ID	Modelo	Parámetros	Subconjunto	R cuadrado	RMSE	Tiempo (s)
Ridge_All_Rand	Ridge	ERROR	Train	ERROR	ERROR	2370
			Validación			
			Test			
KNN_All_Rand	KNeighbors Regressor	Weights: 'uniform' N_neighbors: 7 Metric: 'euclidean'	Train	0,473864138	9117,540002	553,9834478
			Validación	0,230203519		
			Test	0,286331361	12354,94776	
DT_All_Rand	DecisionTree Regressor	'modelo__min_samples_split': 2 'modelo__min_samples_leaf': 10 'modelo__max_depth': 30 'modelo__criterion': 'absolute_error' 'modelo__ccp_alpha': 0.3	Train	0,742771939	6375,11314	260,1581664
			Validación	0,546538807		
			Test	0,632936473	8860,601883	
RF_All_Rand	RandomForest Regressor	'modelo__n_estimators': 50 'modelo__min_samples_split': 11 'modelo__min_samples_leaf': 3 'modelo__max_depth': 30 'modelo__criterion': 'poisson' 'modelo__ccp_alpha': 0.2	Train	0,859458734	4712,275315	7903,214629
			Validación	0,640109358		
			Test	0,688297355	8165,125251	
GB_All_Rand	GradientBoosting Regressor	'modelo__n_estimators': 2000 'modelo__min_samples_split': 7 'modelo__min_samples_leaf': 3 'modelo__max_depth': 7 'modelo__loss': 'huber' 'modelo__learning_rate': 0.1 'modelo__criterion': 'squared_error' 'modelo__ccp_alpha': 0.1 'modelo__alpha': 0.3	Train	0,930725599	3308,377938	16465,80848
			Validación	0,675977153		
			Test	0,736482119	7507,539954	

HGB_All_Rand	**HistGradientBoosting Regressor**	**'modelo__min_samples_leaf': 3 'modelo__max_leaf_nodes': 15 'modelo__max_iter': 100 'modelo__max_depth': 20 'modelo__max_bins': 155 'modelo__loss': 'squared_error' 'modelo__learning_rate': 0.1 'modelo__l2_regularization': 0.2**	**Train**	**0,880038932**	**4353,60364**	**3067,292613**
			Validación	**0,64664769**		
			Test	**0,724839215**	**7671,598708**	
XGB_All_Rand	XGB Regressor	'modelo__subsample': 0.8 'modelo__objective': 'reg:tweedie' 'modelo__n_estimators': 500 'modelo__min_child_weight': 5 'modelo__max_depth': 4 'modelo__gamma': 0.5 'modelo__eval_metric': 'rmse' 'modelo__eta': 0.4 'modelo__colsample_bytree': 0.6 'modelo__booster': 'gbtree'	Train	0,959814918	2519,771474	3521,691074
			Validación	0,647352723		
			Test	0,68942766	8150,307491	

Tabla 9: Resultados experimento 2

Como se puede observar en ambos experimentos sucede lo mismo, Histogram Gradient Boosting obtiene un resultado un tanto peor pero el tiempo es mucho mejor que Gradient Boosting, en este experimento los resultados son mejores que en el experimento 1.

Ridge obtiene unos pésimos resultados y en ocasiones falla, debido a la dimensionalidad del conjunto de datos provocando un fallo de convergencia, esto sucede tanto en el experimento 1 como en el experimento 2.

4.4.3 Experimento 3

Conjunto de datos por estación meteorológica, únicamente las necesarias para la predicción del conjunto de datos de la campaña 22, es decir, las estaciones 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18 y 19 dividiendo Train y Test teniendo en cuenta las campañas de cuando se tomaron las medidas. Tanto con este como con el experimento 6, nos damos cuenta de que hay estaciones con muy pocas instancias, por lo que en el resto de experimentos utilizamos técnicas para generar datos de forma artificial, los experimentos 4 y 7 con PCA y los 5 y 8 con Deep Learning.

ID	Modelo	Parámetros	Subconjunto	R cuadrado	RMSE	Tiempo (s)
Ridge_Est	Ridge	Los parámetros de cada modelo por zona se pueden ver en la ejecución de la libreta.	Train	0,75773636	5476,381304	742,940449
			Validación	0,56297525		
			Test	0,595575136	7593,881113	
KNN_Est	KNeighbors Regressor	…	Train	0,964405263	389,6103298	31,99488091
			Validación	-0,03043512		
			Test	0,104069038	11541,12596	
DT_Est		…	Train	0,654034892	6360,159422	55,29191828

	DecisionTree Regressor		Validación	0,505088279		
			Test	0,500579876	8334,823271	
RF_Est	RandomForest Regressor	…	Train	0,809080567	4853,702335	2260,532574
			Validación	0,61740193		
			Test	0,663959667	6970,085963	
GB_Est	GradientBoosting Regressor	…	Train	0,921608811	2983,212323	3904,934054
			Validación	0,633111367		
			Test	0,716007241	6327,323304	
HGB_Est	HistGradientBoosting Regressor	…	Train	0,860471358	4028,685641	2010,15395
			Validación	0,617590966		
			Test	0,658449468	7085,53151	
XGB_Est	**XGB Regressor**	**…**	**Train**	**0,987397352**	**688,8494895**	**1180,289382**
			Validación	**0,648822979**		
			Test	**0,665187982**	**6893,555119**	

Tabla 10: Medias ponderadas de los resultados experimento 3

Como se puede observar en la Tabla 10 el modelo de los vecinos más cercanos sigue siendo el peor con una clara diferencia, aunque con el conjunto de entrenamiento obtiene buenos resultados, este se sobre ajusta a los datos y finalmente con el de test obtiene unos muy malos, en este experimento el mejor modelo es XGBoost con un tiempo de ejecución bastante bueno, y unos resultados ligeramente peores a Gradient Boosting.

4.4.4 Experimento 4

Conjunto de datos por estación meteorológica (únicamente las necesarias para la predicción del conjunto de datos de la campaña 22) dividiendo Train y Test teniendo en cuenta las campañas de cuando se tomaron las medidas. Los casos en los no tenemos gran cantidad de datos disponibles aplicamos resampling con PCA, aumentando 10 veces el número de instancias de las estaciones que tienen menos de 100.

ID	Modelo	Parámetros	Subconjunto	R cuadrado	RMSE	Tiempo (s)
Ridge_Est_PCA	Ridge	Los parámetros de cada modelo por zona se pueden ver en la ejecución de la libreta.	Train	0,745929496	5346,828102	844,2394974
			Validación	0,591197372		
			Test	0,486641405	8209,326863	
KNN_Est_PCA	KNeighbors Regressor	…	Train	0,987369646	113,6736922	34,59799719
			Validación	0,183503738		
			Test	0,053629725	12269,61962	
DT_Est_PCA	DecisionTree Regressor	…	Train	0,730040271	5133,18696	62,61298943
			Validación	0,587777732		
			Test	0,483535829	8859,656241	
RF_Est_PCA	RandomForest Regressor	…	Train	0,853603905	3903,309802	2646,651676
			Validación	0,686107156		
			Test	0,595394422	7724,847993	
GB_Est_PCA	GradientBoosting Regressor	…	Train	0,943119676	2162,941126	4262,120562
			Validación	0,71305473		
			Test	0,633837064	7287,137013	
HGB_Est_PCA		…	Train	0,892092386	3165,866066	2285,990815

	HistGradientBoosting Regressor		Validación	0,711111741		
			Test	0,53612753	8175,805893	
XGB_Est_PCA	**XGB Regressor**	**…**	**Train**	**0,992745537**	**527,1119348**	**1292,322551**
			Validación	**0,728398532**		
			Test	**0,575958059**	**7737,778056**	

Tabla 11: Medias ponderadas de los resultados experimento 4

Como se puede ver en la Tabla 11 el modelo que obtiene el mejor RMSE es Random Forest mientras que el que obtiene mejor R cuadrado es Gradient Boosting, pero teniendo en cuenta los tiempos de ejecución y el pequeño empeoramiento de los resultados, se decide seleccionar como mejor modelo a XGBoost de nuevo.

4.4.5 Experimento 5

Conjunto de datos por estación meteorológica (únicamente las necesarias para la predicción del conjunto de datos de la campaña 22) dividiendo Train y Test teniendo en cuenta las campañas de cuando se tomaron las medidas. Los casos en los no tenemos gran cantidad de datos disponibles aplicamos data augmentation con Deep Learning, aumentando el número de instancias de las estaciones que tienen menos de 100 instancias añadiendo 500 más.

ID	Modelo	Parámetros	Subconjunto	R cuadrado	RMSE	Tiempo (s)
Ridge_Est_DL	Ridge	Los parámetros de cada modelo por zona se pueden ver en la ejecución de la libreta.	Train	0,699373438	4560,12873	880,2336285
			Validación	0,13531015		
			Test	0,558334887	6250,964769	
KNN_Est_DL	KNeighbors Regressor	…	Train	0,988937801	99,56306325	33,97420406
			Validación	-0,023348436		
			Test	0,287738574	8759,296498	
DT_Est_DL	DecisionTree Regressor	…	Train	0,662544612	4874,693594	72,92201614
			Validación	0,26512452		
			Test	0,502847517	6483,691249	
RF_Est_DL	RandomForest Regressor	…	Train	0,791800695	3818,263683	6021,243702
			Validación	0,445240483		
			Test	0,63848251	5573,574853	
GB_Est_DL	GradientBoosting Regressor	…	Train	0,887022119	2075,45815	10078,18464
			Validación	0,423250262		
			Test	0,71136767	4957,155344	
HGB_Est_DL	HistGradientBoosting Regressor	…	Train	0,87246826	2840,444456	2421,060263
			Validación	0,412824961		
			Test	0,682783948	5329,393806	
XGB_Est_DL	**XGB Regressor**	**…**	**Train**	**0,954930687**	**580,3400763**	**1773,371649**
			Validación	**0,387084383**		
			Test	**0,697037812**	**5314,67912**	

Tabla 12: Medias ponderadas de los resultados experimento 5

En este experimento al igual que en los demás elegimos XGBoost como el mejor modelo por su buen tiempo de ejecución y buenos resultados, aunque el mejor si nos fijáramos únicamente en

los resultados seguiría siendo Gradient Boosting como se puede ver en la Tabla 12: Medias ponderadas de los resultados experimento 5.

4.4.6 Experimento 6

Conjunto de datos por estación meteorológica (únicamente las necesarias para la predicción del conjunto de datos de la campaña 22) dividiendo Train y Test de forma aleatoria.

ID	Modelo	Parámetros	Subconjunto	R cuadrado	RMSE	Tiempo (s)
Ridge_Est_Rand	Ridge	Los parámetros de cada modelo por zona se pueden ver en la ejecución de la libreta.	Train	0,794407247	5292,385603	762,1687655
			Validación	0,604053496		
			Test	0,600751192	7137,956497	
KNN_Est_Rand	KNeighbors Regressor	…	Train	0,093838065	11202,51793	31,26375175
			Validación	0,053492094		
			Test	0,05003887	11617,86507	
DT_Est_Rand	DecisionTree Regressor	…	Train	0,733211347	6053,405986	54,18977714
			Validación	0,488426029		
			Test	0,573532323	7465,732752	
RF_Est_Rand	RandomForest Regressor	…	Train	0,836701101	4837,436605	2480,820943
			Validación	0,59475161		
			Test	0,632386128	7001,345272	
GB_Est_Rand	GradientBoosting Regressor	…	Train	0,932298445	3004,765424	3831,141417
			Validación	0,609467775		
			Test	0,654156135	6749,323052	
HGB_Est_Rand	HistGradientBoosting Regressor	…	Train	0,915651303	3349,851236	1913,651748
			Validación	0,64036446		
			Test	0,665535819	6661,989474	
XGB_Est_Rand	**XGB Regressor**	**…**	**Train**	**0,972013993**	**1028,329918**	**1292,693055**
			Validación	**0,617113916**		
			Test	**0,693361025**	**6302,343829**	

Tabla 13: Medias ponderadas de los resultados experimento 6

Si comparamos los resultados de la Tabla 10: Medias ponderadas de los resultados experimento 3 con los de este experimento expuestos en la Tabla 13: Medias ponderadas de los resultados experimento 6 se puede observar como algunos modelos mejoran, pero tampoco de una forma exagerada. En este caso XGBoost obtiene las mejores puntuaciones y el mejor tiempo de ejecución.

4.4.7 Experimento 7

Conjunto de datos por estación meteorológica (únicamente las necesarias para la predicción del conjunto de datos de la campaña 22) dividiendo Train y Test de forma aleatoria. Los casos en los no tenemos gran cantidad de datos disponibles aplicamos resampling con PCA.

ID	Modelo	Parámetros	Subconjunto	R cuadrado	RMSE	Tiempo (s)
Ridge_Est_PCA_Rand	Ridge	Los parámetros de cada modelo por zona se pueden ver en la ejecución de la libreta.	Train	0,793596996	5224,129107	851,7702038
			Validación	0,65718925		
			Test	0,632522267	6803,472904	
KNN_Est_PCA_Rand		…	Train	0,275963156	8925,904559	32,70323372

	KNeighbors Regressor		Validación	0,259128977		
			Test	0,24513901	9375,765939	
DT_Est_PCA_Rand	DecisionTree Regressor	…	Train	0,796121505	4931,424499	58,97139859
			Validación	0,601630797		
			Test	0,667888823	6121,221041	
RF_Est_PCA_Rand	RandomForest Regressor	…	Train	0,870312655	4001,583788	2796,043677
			Validación	0,690014629		
			Test	0,713443141	5758,874968	
GB_Est_PCA_Rand	GradientBoosting Regressor	…	Train	0,94635055	2434,989188	4160,973998
			Validación	0,710282825		
			Test	0,733236214	5310,262992	
HGB_Est_PCA_Rand	HistGradientBoosting Regressor	…	Train	0,933207379	2694,945971	2095,443973
			Validación	0,730153201		
			Test	0,739243749	5282,386382	
XGB_Est_PCA_Rand	**XGB Regressor**	**…**	**Train**	**0,980228602**	**804,6871429**	**1386,070027**
			Validación	**0,708820511**		
			Test	**0,761574855**	**5003,885115**	

Tabla 14: Medias ponderadas de los resultados experimento 7

En este experimento con la Tabla 14 podemos observar que XGBoost vuelve a ser el mejor modelo de entre todos los evaluados.

4.4.8 Experimento 8

Conjunto de datos por estación meteorológica (únicamente las necesarias para la predicción del conjunto de datos de la campaña 22) dividiendo Train y Test de forma aleatoria. Los casos en los no tenemos gran cantidad de datos disponibles aplicamos data augmentation con Deep Learning.

ID	Modelo	Parámetros	Subconjunto	R cuadrado	RMSE	Tiempo (s)
Ridge_Est_DL_Rand	Ridge	Los parámetros de cada modelo por zona se pueden ver en la ejecución de la libreta.	Train	0,699097949	4372,288847	894,2269268
			Validación	0,387952898		
			Test	0,329368331	6034,157337	
KNN_Est_DL_Rand	KNeighbors Regressor	…	Train	0,128018366	8547,675238	33,24745631
			Validación	0,048477157		
			Test	0,027076481	9091,948421	
DT_Est_DL_Rand	DecisionTree Regressor	…	Train	0,544521914	5018,496899	72,05663919
			Validación	0,333619256		
			Test	0,410120969	6131,639099	
RF_Est_DL_Rand	RandomForest Regressor	…	Train	0,703693939	3925,059415	6196,258917
			Validación	0,450011753		
			Test	0,508497371	5597,908537	
GB_Est_DL_Rand	GradientBoosting Regressor	…	Train	0,756629039	2739,504984	11167,5906
			Validación	0,453572307		
			Test	0,512497876	5516,449275	
HGB_Est_DL_Rand	HistGradientBoosting Regressor	…	Train	0,790857321	2968,853875	2440,246402
			Validación	0,499046657		
			Test	0,493101862	5580,725152	
XGB_Est_DL_Rand	**XGB Regressor**	**…**	**Train**	**-6,3859E+12**	**1418682784**	

				(Error estación 13) 0,947023361	(Error estacón 13) 930,2249013	1939,302732 (Error estacón 13) 1809,137234
			Validación	-447,222949 (Error estación 13) 0,500085965		
			Test	-6,7570E+12 (Error estación 13) 0,537983359	1609593314 (Error estación 13) 5475,814873	

Tabla 15: Medias ponderadas de los resultados experimento 8

En este ultimo experimento hay que destacar la aparición de un error con el modelo XGBoost en la estación 13 no se sabe de una forma segura la causa del error debido a que ha funcionado correctamente con el resto de estaciones y en el resto de experimentos, pero posiblemente sea un error de la librería ya que con esta estación no se modifica nada con respecto al resto o al resto de experimentos.

En la tabla mostramos los resultados teniendo en cuenta los del fallo (realmente no es un fallo pero obtiene unas puntuaciones realmente malas cosa que solo podría ser causado por un error) y sin tener en cuenta estas puntuaciones, y si nos fijamos en las obtenidas sin el fallo vemos que el modelo vuelve a ser el mejor obteniendo un muy buen tiempo de ejecución y unas muy buenas puntuaciones.

4.4.9 Selección de los mejores modelos

Como se puede comprobar los modelos que generalmente obtienen los peores resultados son Ridge, KNeighbors Regressor y DecisionTree Regressor, aunque este último no obtiene tan malos resultados como los otros, pero en comparación con los ensembles tienen unos tiempos de ejecución muy cortos pese a los malos resultados.

Por otro lado, los ensembres obtienen unos mejores resultados, pero claro con unos tiempos de ejecución mucho más elevados, RandomForest Regressor y GradientBoosting Regressor son los que tienen unos altos tiempos de ejecución y quedan también descartados pese a que suelen tener los mejores resultados.

Los tiempos de ejecución se deben tener en cuenta ya que en su momento también fue un requisito del Datathon "será necesario evaluar el consumo energético (en base al tiempo de computación) vs precisión del modelo, usando el resultado de dicha evaluación como una de las variables a tener en cuenta en la elección del modelo ganador".

Por último, la elección del modelo final está entre XGB Regressor e HistGradientBoosting Regressor, ya que, aunque los resultados están muy parejos con RandomForest Regressor y GradientBoosting Regressor los tiempos de ejecución son mucho menores por lo que tienen un mejor rendimiento. Entre los dos finalistas vemos que cada uno es mejor en un tipo de experimentos, quiero decir, se puede ver que XGB Regressor es mejor en los experimentos donde se ha dividido el conjunto de datos por estaciones, tanto las puntuaciones como los tiempos de ejecución, mientras que HistGradientBoosting es mejor cuando se enfrenta al conjunto completo de datos y también obtiene un mejor tiempo de ejecución. Entre estos viendo el RMSE vemos que ambos modelos obtienen un mejor resultado en los casos donde se tiene en cuenta las estaciones, por lo que finalmente tenemos como elegido el modelo XGB Regressor.

4.5 Explicabilidad

La explicabilidad con SHAP se ha realizado únicamente del que se ha considerado mejor modelo, en nuestro caso XGB Regressor. Como el modelo seleccionado para explicar utilizando SHAP está basado en árboles, vamos a utilizar el TreeExplainer ya que este es un método rápido y exacto para estimar los valores SHAP de los modelos de árboles y ensembles de árboles, bajo varios supuestos posibles sobre la dependencia de las características.

Inicialmente representamos gracias a los SHAP values el impacto de cada variable en el modelo con un Summary_plot utilizando la media del valor absoluto en formato de barras y simplemente los SHAP values con otro Summary_plot (Figura 18) y observando cómo es la variables que hace referencia a la superficie la que más impacto tiene en las predicciones seguida del modo de cultivo utilizado, el resto de variables vemos como también tienen algo de importancia, pero mucha menos en comparación a estas.

Figura 18: Representación de los SHAP Values con

Tras lo cual hemos utilizado Force_Plot para mostrar como modifican la salida del modelo las distintas variables, se puede visualizar con el conjunto completo de datos o con una única instancia (Figura 19).

Figura 19: Representación con Force_Plot, la superior es del conjunto completo de datos de Test y la inferior es de una única instancia.

Se puede ver como las variables que están en rojo aumentan el valor de las predicciones, mientras que las que están en azul lo disminuyen, en este caso la superficie el modo lo que hace es aumentar la predicción, mientras que la variedad es la variable que más la disminuye. A continuación, mostramos de una forma más visual como el modelo va tomando las decisiones con el impacto de las variables gracias a los decision_plot, que al igual de antes se puede mostrar del conjunto completo de datos o de una única instancia (**¡Error! No se encuentra el origen de la referencia.**).

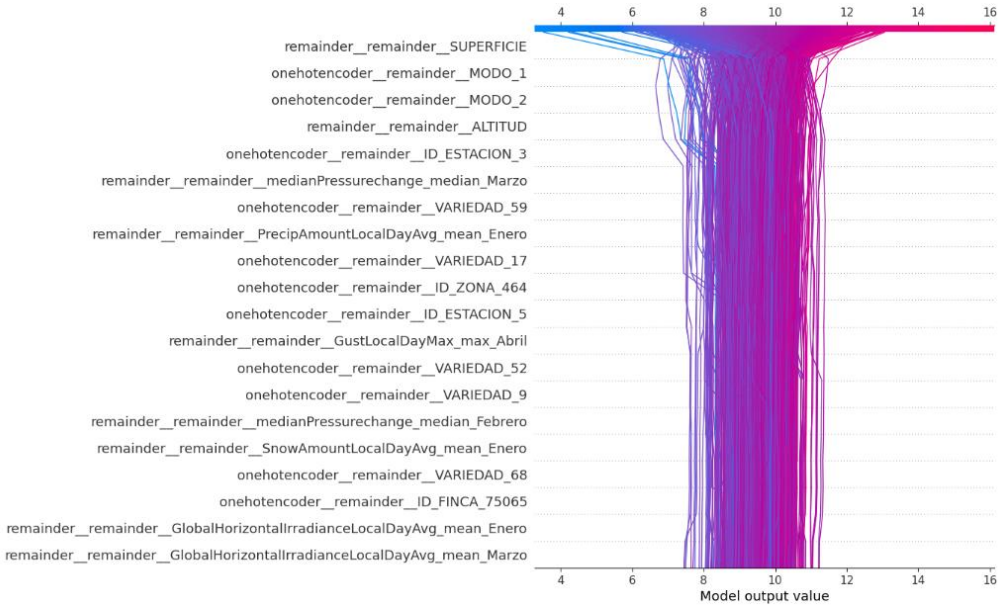

Figura 20: Representación con decisión_plot de cómo va cambiando las predicciones del modelo al utilizar las distintas variables del conjunto completo de datos.

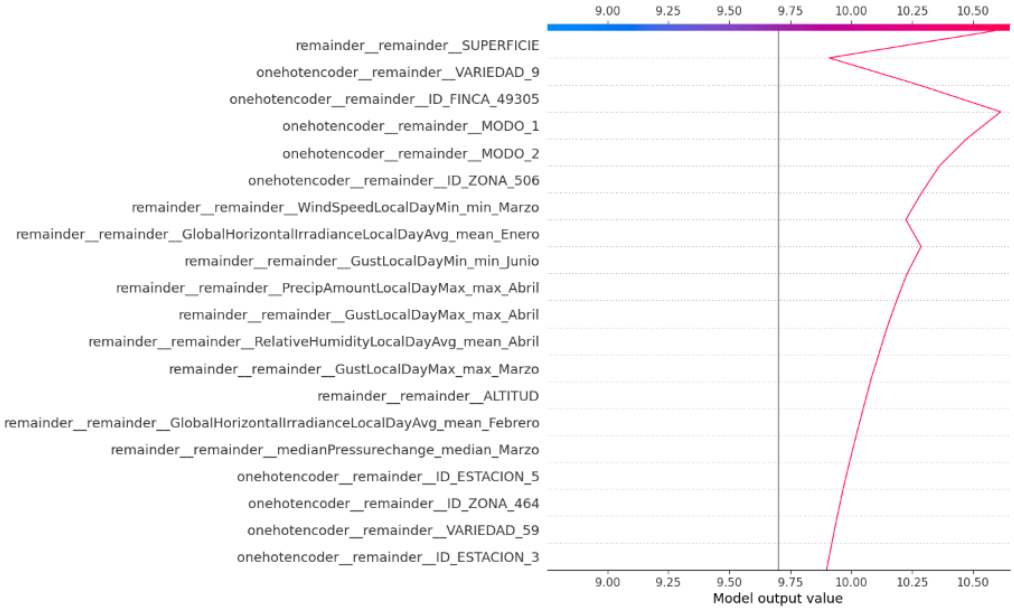

Figura 21: Representación con decisión_plot de cómo va cambiando la predicción del modelo al utilizar las distintas variables con una única instancia.

Como se puede observar, la variable que más impacto tiene en el modelo es SUPERFICIE por lo que vamos a utilizar Dependence_plot (Figura 22) para observar la distribución de los SHAP values para esta variable, pero diferenciando el color dependiendo de los valores que toman las variables binarias MODO, TIPO y COLOR. Se puede ver como aumentan los SHAP Values conforme aumenta la superficie, pero las otras variables se distribuyen de una forma genérica sin tener un patrón visible.

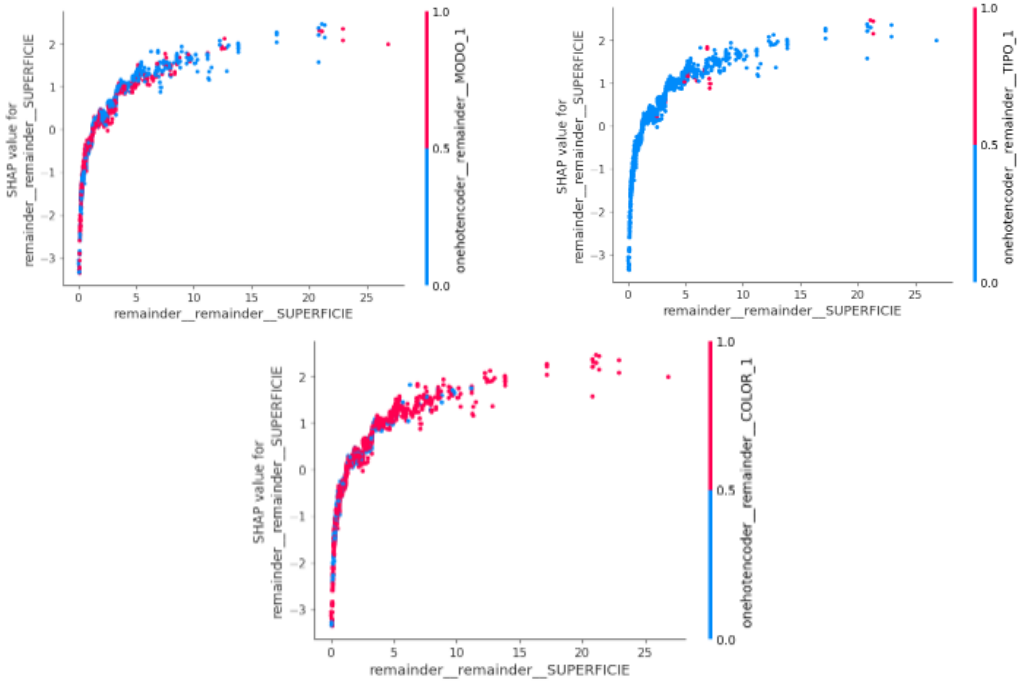

Figura 22: Representación con Dependence_plot los SHAP values de SUPERFICIE teniendo en cuenta MODO, TIPO y COLOR

Capítulo 5

Conclusiones y Trabajo Futuro

5.1 Conclusiones

El objetivo de este proyecto era el desarrollo y creación de un modelo de machine learning para la predicción de la producción, el cual se ha podido desarrollar correctamente a lo largo de este proyecto con unos resultados bastante buenos. A continuación, se resumen las conclusiones clave:

- **Exploración y Preprocesamiento de Datos:** Iniciamos nuestro proyecto con una exhaustiva exploración de los datos, lo que nos permitió comprender mejor la naturaleza de las características y la distribución de las mismas y de la producción de vinos.

  Realizamos un sólido preprocesamiento de datos que incluyó la eliminación de variables con alta correlación, la codificación de binarización de variables categóricas, etc.
- **Selección y Entrenamiento de Modelos:** Evaluamos múltiples algoritmos de regresión, desde modelos lineales como la regresión lineal hasta modelos más avanzados como Random Forest, Gradient Boosting y XGBoost.

  Utilizamos técnicas como la validación cruzada y la búsqueda de hiperparámetros para ajustar nuestros modelos y optimizar su rendimiento. Observamos que los

modelos basados en árboles, como Random Forest y Gradient Boosting, superaron a los modelos lineales.

- **Evaluación del Rendimiento:** Medimos el rendimiento de nuestros modelos utilizando métricas como el RMSE y R-cuadrado. Obtuvimos resultados prometedores con un RMSE bajo y valores de R-cuadrado que indican una buena capacidad de predicción. Sin embargo, también observamos que algunos modelos, como Gradient Boosting o XGBoost, lograron un mejor rendimiento que otros.
- **Importancia de las Características:** Utilizamos técnicas de análisis de importancia de características para identificar las variables más influyentes en nuestras predicciones. Esto nos permitió comprender qué características tienen el mayor impacto en producción del vino.
- **Conclusiones Generales:** En general, nuestro proyecto ha demostrado que es posible predecir la producción de los vinos con un buen nivel de precisión utilizando técnicas de aprendizaje automático.

  La elección del modelo adecuado y la selección de características influyen significativamente en la calidad de las predicciones.

  Las técnicas de análisis de datos y visualización desempeñaron un papel crucial en la comprensión de los datos y la toma de decisiones.

5.2 Trabajo Futuro

Para futuros trabajos, podríamos considerar la inclusión de más datos, así como la expansión de nuestro conjunto de características para mejorar aún más el rendimiento del modelo.

También podríamos explorar técnicas avanzadas de aprendizaje profundo para abordar aspectos más complejos para la predicción de la producción del vino.

Bibliografía

[1] *Datathon Cajamar UniversityHack 2023*. (n.d.). https://www.cajamardatalab.com/datathon-cajamar-universityhack-2023/reto-lavina/

[2] *NUMPY User Guide — NUMPY V1.25 Manual*. (s. f.). https://numpy.org/doc/stable/user/index.html

[3] *Pandas documentation — Pandas 2.1.0 documentation*. (s. f.-b). https://pandas.pydata.org/docs/index.html

[4] Gelman, A., & Hill, J. (2007). *Data analysis using regression and Multilevel/Hierarchical models*. Cambridge University Press.

[5] Enders, C. K. (2010). *Applied missing data analysis*. *New York: Guilford. http://hsta559s12.pbworks.com/w/file/fetch/52112520/enders.applied*

[6] Taylor SJ, Letham B. (2017). *Forecasting at scale. PeerJ Preprints 5:e3190v2* https://doi.org/10.7287/peerj.preprints.3190v2

[7] *MatPlotLib: a 2D Graphics environment*. (2007, 1 junio). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4160265

[8] Waskom, M. (2021). Seaborn: Statistical Data Visualization. *Journal of open source software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

[9] Pedregosa, F. (2011). *SciKit-Learn: Machine Learning in Python*. https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[10] Von Lschmiddey, D. S. B. (2022, 23 abril). Deep tabular augmentation for regression tasks. *Data Science Blog von lschmiddey*. https://lschmiddey.github.io/fastpages_/2022/04/23/DataAugmentation_for_Regression_Tasks.ipynb.html

[11] *Regularización Ridge, Lasso y Elastic Net con Python y Scikitlearn*. (n.d.). https://cienciadedatos.net/documentos/py14-ridge-lasso-elastic-net-python

[12] Rifkin, R., & Lippert, R. A. (2007). Notes on Regularized Least Squares. *Computer Science and Artificial Intelligence Laboratory Technical Report*. http://cbcl.mit.edu/publications/ps/MIT-CSAIL-TR-2007-025.pdf

[13] Loh, W. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews-Data Mining and Knowledge Discovery*, *1*(1), 14–23. https://doi.org/10.1002/widm.8

[14] Breiman, L. (2017b). *Classification and regression trees*. Routledge. https://doi.org/10.1201/9781315139470

[15] *k-Nearest Neighbor Regressors Optimized by using Random Search*. (2018, November 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8661399

[16] 류충상, 송재희, 손채봉, 이상이, & 김은수. (2017). Nearest Neighbor algorithm. En Springer eBooks (p. 1421). https://doi.org/10.1007/978-3-319-17885-1_100841

[17] Rodriguez-Galiano, V., Sanchez-Castillo, M., Chica-Olmo, M., & Chica-Rivas, M. (2015). *Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines*. *Ore Geology Reviews*, *71*, 804–818. https://doi.org/10.1016/j.oregeorev.2015.01.001

[18] Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5-32. https://doi.org/10.1023/a:1010933404324

[19] *Greedy function approximation: a gradient boosting machine on JSTOR*. (n.d.). https://www.jstor.org/stable/2699986?casa_token=rUXlsXhKmsIAAAAA%3AiO9-XXhTMFEw9Z7BuFq41JhmQH7ggBgAQ4JSwbk1y6-18ne342lAwt3yhUE-KnVARpDrL-oAszhl2eOAwSQKnG1HcqeBq0vTQo-NLbRof_GUzRdI_xGq&seq=2

[20] *Gradient Boosting con Python* by Joaquín Amat Rodrigo, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/py09_gradient_boosting_python.html

[21] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, *29*(5). https://doi.org/10.1214/aos/1013203451

[22] Gayathri, R., Rani, S. U., Čepová, L., Raj, C. J., & Kalita, K. (2022). *A comparative analysis of machine learning models in prediction of mortar compressive strength. Processes*, *10*(7), 1387. https://doi.org/10.3390/pr10071387

[23] Ke, G. (2017). *LightGBM: a highly efficient gradient boosting decision tree*. https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[24] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* https://doi.org/10.1145/2939672.2939785

[25] Palma, R. P. (2022, July 31). *Análisis crítico del coeficiente de determinación (R2), como indicador de la calidad de modelos lineales y no lineales*. http://www.revistas.espol.edu.ec/index.php/matematica/article/view/1037

[26] Smith-Miles, K. (2011). International Encyclopedia of Statistical Science. In *Springer eBooks*. https://doi.org/10.1007/978-3-642-04898-2

[27] Lundberg, S. M. (2017). *A unified approach to interpreting model predictions*. https://proceedings.neurips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html