

UNIVERSIDAD PONTIFICIA DE COMILLAS
ICAI

Master in Big Data. Technologies and Advanced Analytics.



COMILLAS
UNIVERSIDAD PONTIFICA

ICAI

ICADE

CIHS

**ENTREGA 1: REPORT
ENSEMBLE LEARNING
MACHINE LEARNING II ASSIGNMENT**

Alberto García Martín, 202309188, mbd46

Jorge Peralta Fernández-Revuelta, 202310326, mbd27

Juan López Segura, 202308780, mbd39

Ignacio Urretavizcaya Tato, 202316008, mbd37

13 de Abril de 2024

Fecha de Entrega: 15 de Abril de 2024

Contents

1	Introducción	2
2	EDA	2
2.1	Valores faltantes y conversiones	2
2.2	Outliers	2
2.3	Relaciones entre Variables	4
3	Análisis Previo	7
3.1	Clustering	7
3.2	PDF	8
3.3	Bootstrap	8
4	Modelos de ensamblaje	9
4.1	Árbol de Decisión	9
4.2	Bagged Tree	10
4.3	Random Forest	11
4.4	Boosting	13
4.5	AdaBoost	13
4.6	Gradient Boosting	15
4.7	Stacking	16
5	Comparación de modelos y conclusiones	17
6	Comprobación del Modelo Final	22
7	Propuestas de Mejora	24

1 Introducción

Este trabajo se basa en el estudio de un conjunto de datos dividido en dos dataframes: *DATOS_UTIL* y *DATOS_IRRAD*. Estos ficheros contienen, como su propio nombre indican, datos de irradiación solar y utilización en tramos horarios de 3 horas, donde cada registro es un día distinto y contiene cada uno de ellos 8 variables (una por cada tramo horario).

Es importante saber que la irradiación solar se conoce como la irradiación neta absorbida por la superficie terrestre, siendo ésta la suma de la radiación directa, difusa y reflejada. En este caso, la irradiación de la hora 0 es la acumulada de las horas 0,1 y 2. Por otro lado, la utilización solar fotovoltaica puede definirse como la generación solar horaria entre la potencia (siendo esta magnitud adimensional y acotada entre 0 y 1). Además, dicha utilización horaria del tramo 0 no es la suma de las horas 0,1 y 2, sino su media.

2 EDA

Antes de comenzar con la creación de los modelos, se realiza un preprocesamiento necesario para conocer los datos que se quieren modelizar. El primer paso se basa en la unión de ambos dataframes en un conjunto único, para posteriormente comenzar con los pasos recomendados.

2.1 Valores faltantes y conversiones

Se comienza con un estudio de valores faltantes, observando que no hay ninguno. No obstante, algunas variables no están bien codificadas, ya que el año, día de la semana, mes o día pueden ser más cómodas de manipular si se tratan como categóricas. Por ello, se realizan los cambios necesarios, llegando al resumen resultante:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2192 entries, 0 to 2191
Data columns (total 21 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   FECHA       2192 non-null   object 
 1   IRRADH00    2192 non-null   float64
 2   IRRADH03    2192 non-null   float64
 3   IRRADH06    2192 non-null   float64
 4   IRRADH09    2192 non-null   float64
 5   IRRADH12    2192 non-null   float64
 6   IRRADH15    2192 non-null   float64
 7   IRRADH18    2192 non-null   float64
 8   IRRADH21    2192 non-null   float64
 9   ANNO        2192 non-null   category
 10  MES          2192 non-null   category
 11  DIA          2192 non-null   category
 12  DIASEM      2192 non-null   category
 13  UTILH00    2192 non-null   float64
 14  UTILH03    2192 non-null   float64
 15  UTILH06    2192 non-null   float64
 16  UTILH09    2192 non-null   float64
 17  UTILH12    2192 non-null   float64
 18  UTILH15    2192 non-null   float64
 19  UTILH18    2192 non-null   float64
 20  UTILH21    2192 non-null   float64
dtypes: category(4), float64(16), object(1)
memory usage: 302.1+ KB
```

Figure 1: Tipos de Variables

2.2 Outliers

El siguiente paso se basa en un estudio de los histogramas de las variables junto a unas métricas básicas para detectar la presencia de outliers o comportamientos anómalos.

	IRRADH00	IRRADH03	IRRADH06	IRRADH09	IRRADH12	IRRADH15	IRRADH18	IRRADH21	UTILH00	UTILH03	UTILH06	UTILH09	UTILH12	UTILH15
count	2192.0	2192.0	2192.000000	2.192000e+03	2.192000e+03	2.192000e+03	2192.000000	2192.0	2192.000000	2192.000000	2192.000000	2192.000000	2192.000000	2192.000000
mean	0.0	0.0	94702.724066	1.011589e+06	1.716058e+06	1.089779e+06	133967.701842	0.0	0.000135	0.001774	0.093245	0.360505	0.414360	0.204008
std	0.0	0.0	148060.832994	6.771510e+05	8.377976e+05	6.809447e+05	199811.564332	0.0	0.000276	0.002380	0.079598	0.176818	0.187624	0.138319
min	0.0	0.0	0.000000	8.408046e+03	1.964414e+04	2.192887e+03	0.000000	0.0	0.000000	0.000000	0.000829	0.007645	0.008106	0.000682
25%	0.0	0.0	0.000000	4.461487e+05	1.051556e+06	4.936689e+05	0.000000	0.0	0.000000	0.000081	0.017445	0.212754	0.261524	0.076712
50%	0.0	0.0	0.000000	1.088099e+06	1.766789e+06	1.054568e+06	0.000000	0.0	0.000000	0.000826	0.077137	0.367105	0.441848	0.193826
75%	0.0	0.0	159331.900000	1.712944e+06	2.477452e+06	1.706886e+06	252328.225000	0.0	0.000188	0.002524	0.151881	0.518411	0.583739	0.329667
max	0.0	0.0	531755.500000	2.288140e+06	3.033988e+06	2.307160e+06	685027.800000	0.0	0.005681	0.016260	0.319156	0.695051	0.720680	0.474323

Figure 2: Resumen de variables Cuantitativas

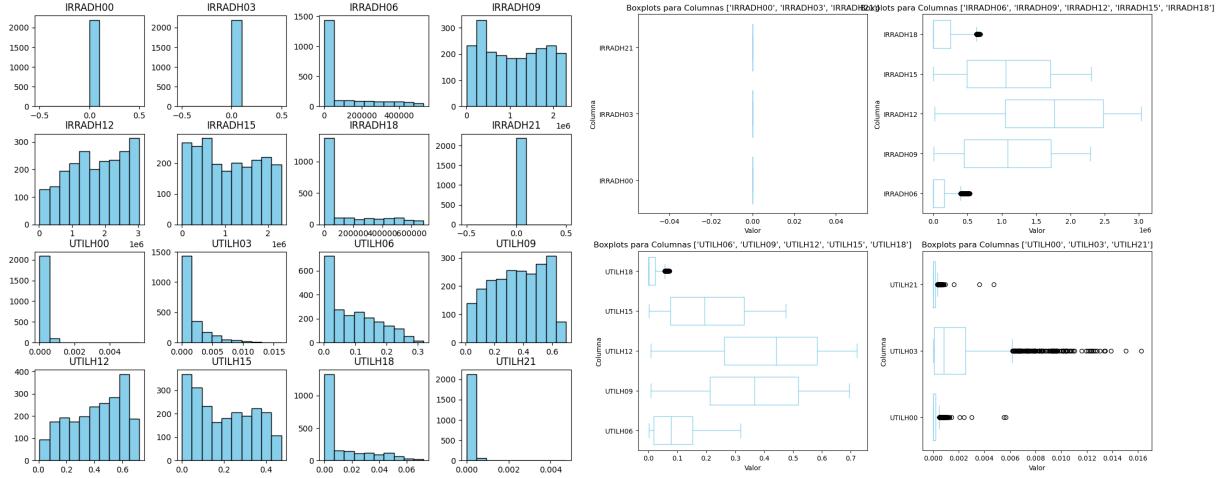


Figure 3: Histogramas

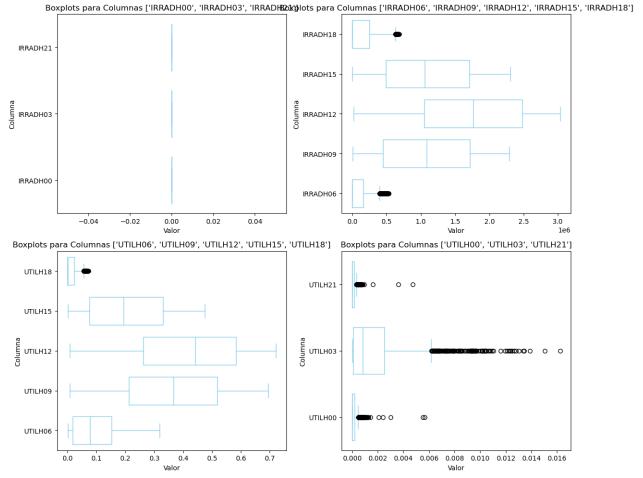


Figure 4: Gráficos de Cajas y Bigotes

Se puede observar que no hay valores extraños en nuestros datos. Claramente, para grupos horarios nocturnos los valores son 0 o muy cercanos a 0, y durante el día la distribución es similar para una agrupación horaria determinada tanto para la irradiación como para la utilización. Por otro lado, como era de esperar, los datos categóricos no presentan problemas y siguen la distribución esperada:

	FECHA	ANNO	MES	DIA	DIASEM
count		2192	2192	2192	2192
unique		2192	6	12	31
top	2015-01-01		2016	1	1
freq	1		366	186	72
					314

Figure 5: Descripción Variables Categóricas

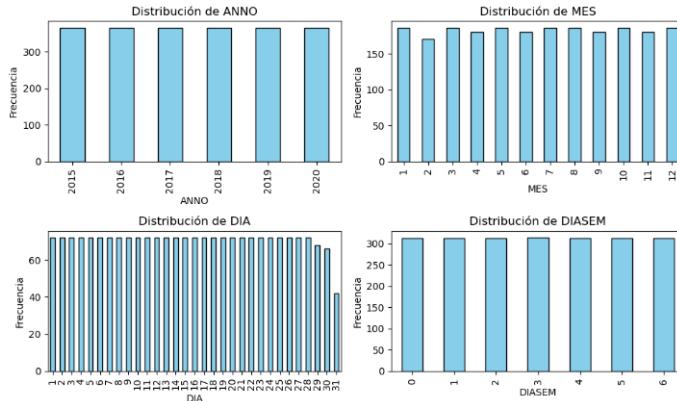


Figure 6: Gráficos de Barras

No obstante, tras este pequeño estudio, surge la necesidad de transformar nuestros datos debido a 2 problemas principales:

- Debemos predecir la utilización para todo grupo horario: Tal y como se encuentra ahora, tenemos 8 variables de salida (aunque se pudiesen agrupar en menos según un comportamiento similar).
- Una utilización no puede depender de una irradiación futura: Se deberían tener filtros que eviten que predictores de irradiación en ciertos grupos horarios se usen para predecir utilizaciones en grupos previos.

Debido a estos factores (entre otros) se crea una nueva variable categórica llamada grupo horario, y se condensan las 8 variables de cada tipo en 1 única, haciendo mucho más cómodo su manipulación. Tras esto y una nueva comprobación de valores faltantes y tipos de variables, se observa que todo funciona correctamente:

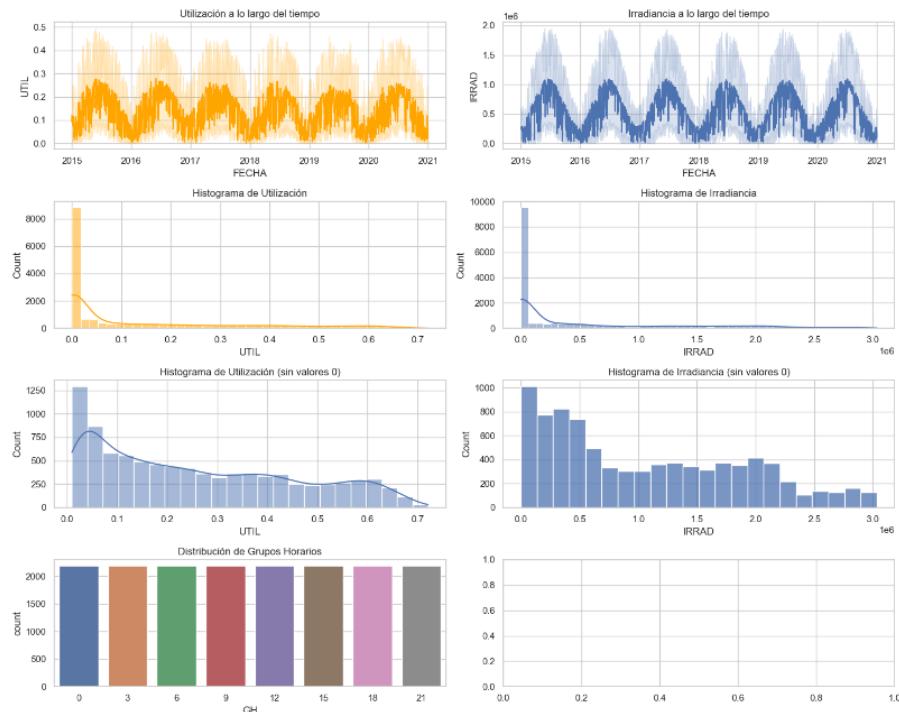


Figure 7: Gráficos Variados de Interés

2.3 Relaciones entre Variables

A continuación se estudian las relaciones entre variables:

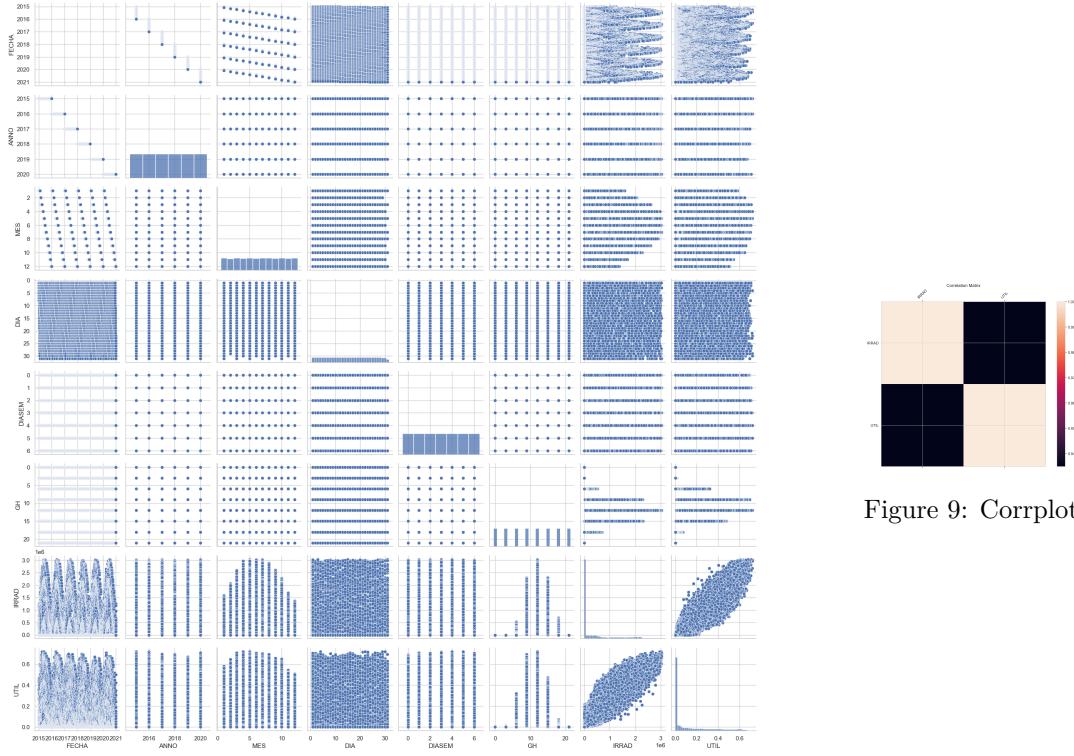


Figure 8: Gráfico de Relaciones entre Variables

Como se puede observar, las más correladas con nuestra respuesta son la irradiación, el mes y el grupo horario, tal y como nos dicta la lógica. El resto no gozan de ningún tipo de relación con la respuesta, tal y como se estudia en el notebook. Además, la correlación entre la irradiación y la utilización es muy alta, dando a entender que no se necesiten modelos muy complejos para obtener resultados decentes.

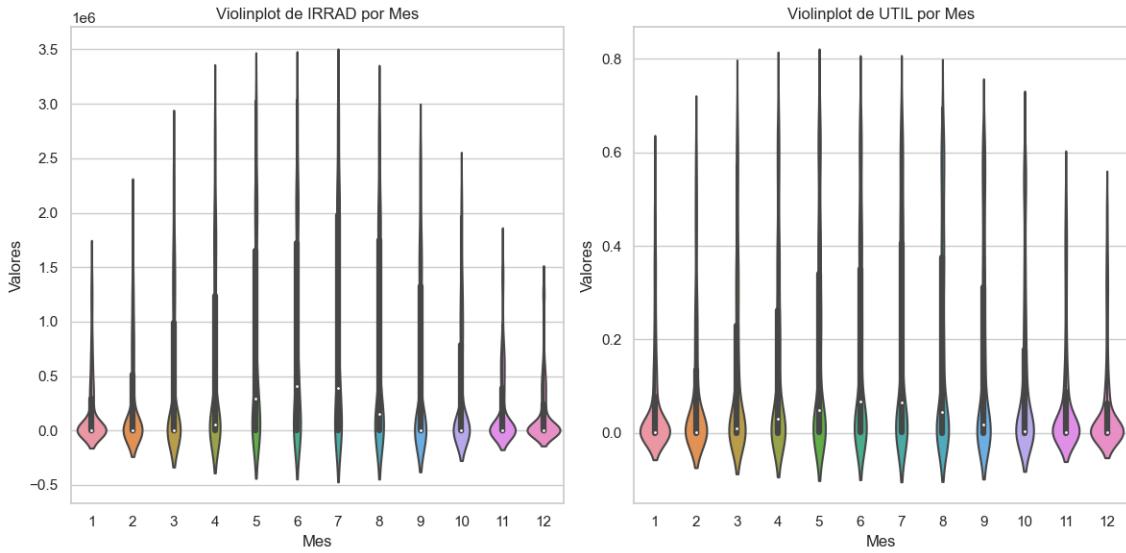


Figure 10: Violinplot con Mes

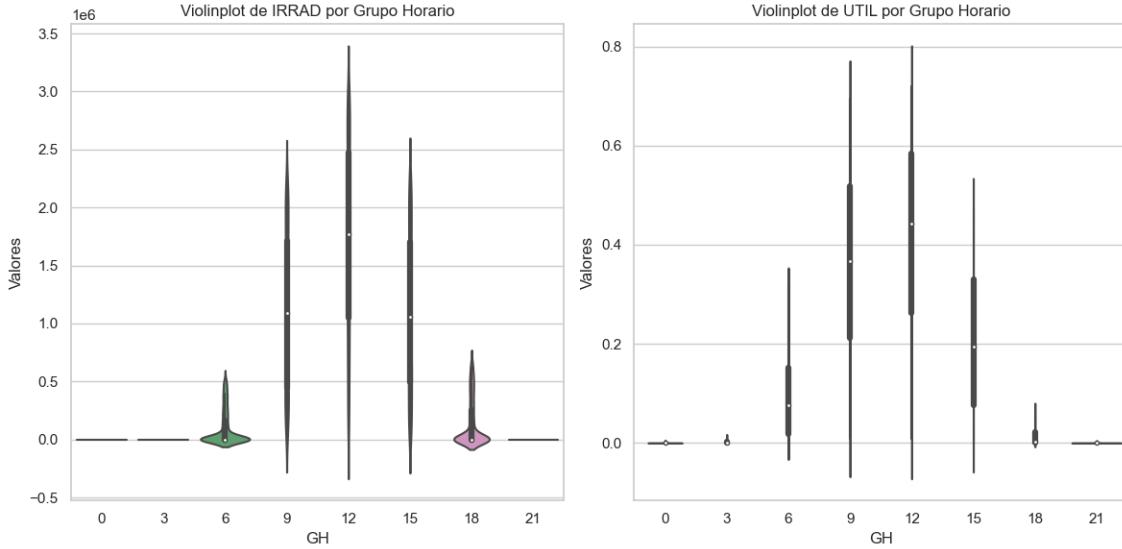


Figure 11: Violinplot con GH

Debido a esta clara relación, se observa un scatterplot condicionado a estas variables categóricas, demostrando que la relación entre la irradiación y la utilización efectivamente cambia dependiendo especialmente del grupo horario en el que nos encontremos:

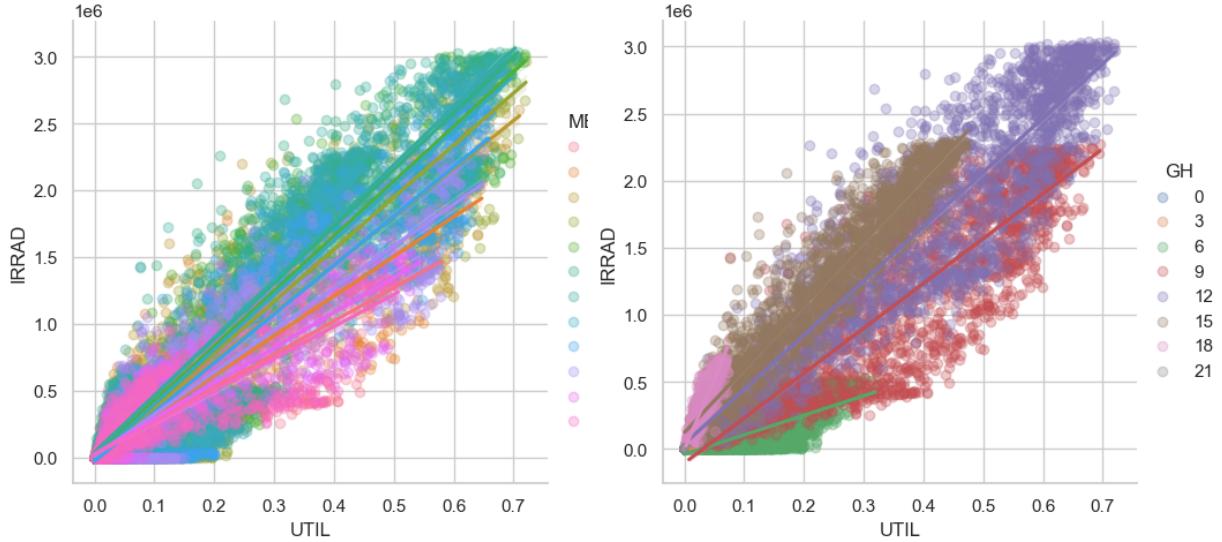


Figure 12: Scatterplots

Por último, dado que las clases están balanceadas (en el caso categórico, como ya se ha estudiado), se divide el conjunto de datos en entrenamiento y test, pero con una peculiaridad. Debido a la cantidad de datos, que es suficientemente grande, se plantea dividir el conjunto de datos en 3:

- **Train:** 60% de los datos (algo más de 10k). Se encuentran también los datos de validación usados para el k-fold cross validation utilizado en el grid search.
- **AdjTest:** Llamado adjusting test para ayudar en la elección y ligeros ajustes de los modelos, 20% de los datos. Una vez los modelos se encuentran ajustados, este subconjunto nos sirve para confirmar si el modelo tiene sobreajuste o subajuste, si es capaz de generalizar, etc. Con ello, escogemos nuestro modelo final.

- **Test:** Se trata del 20% restante de los datos. Este conjunto a priori no lo tendríamos, nos sirve para simular la entrada de datos nuevos y comprobar cómo lo haría el modelo final ante casos de predicción con nuevos datos (que es su caso de uso real). Sirve únicamente para ver si estábamos en lo cierto, o en qué hemos fallado, pero una vez se mira el desempeño del modelo en este conjunto de datos, no se puede volver a tocar nada del modelo.

Con esto termina el análisis exploratorio de los datos, y comienza la segunda etapa del trabajo.

3 Análisis Previo

A pesar de no ser solicitado, se realizan una serie de técnicas previas a los modelos de ensamblaje, de cara a un mejor entendimiento de los datos, y división de modelos si fuese necesario. En esta sección entrarían técnicas como PCA o ICA si fuesen necesarias (en este caso no lo son), pero se aplican para nuestro problema modelos de clustering básicos. Esto se hace para saber si nuestros datos se encuentran clasificados en distintos grupos que se comportan de forma diferente y, por tanto, necesitan modelos distintos para cada uno de los grupos.

3.1 Clustering

Primero, es necesario realizar One-Hot Encoding para las variables categóricas y crear un nuevo dataframe con ellas y eliminando la fecha de cara al correcto funcionamiento (la información de la fecha se encuentra en otras variables, realmente no es necesaria). Tras esto, se estandarizan y se procede a la creación de modelos de clustering según 3 técnicas:

- Hierárquico:

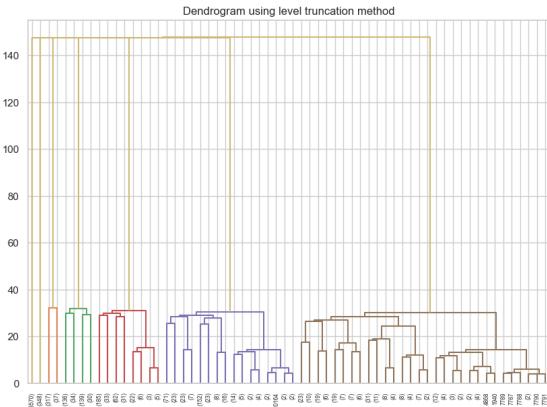


Figure 13: Dendograma

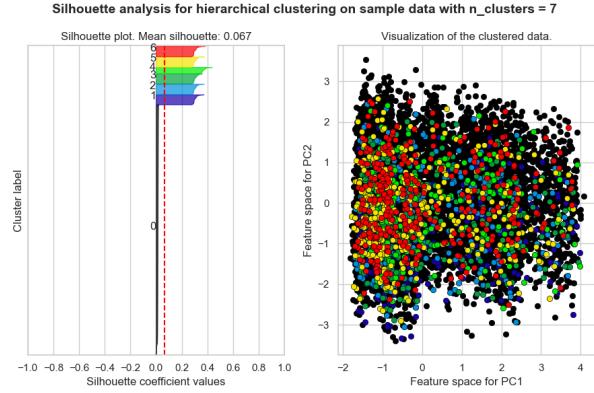


Figure 14: Análisis Silhouette

Se puede observar que, a pesar de que parecía muy claro el número de clústers a usar, los resultados son bastante pobres (al menos observándolos en un espacio de 2 componentes principales).

- K-means:

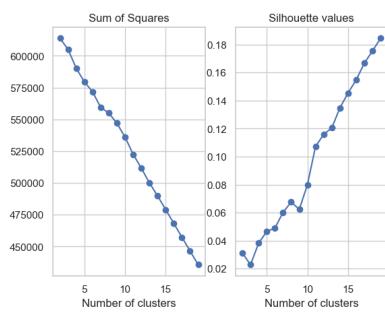


Figure 15: Suma de Cuadrados y Silhouette

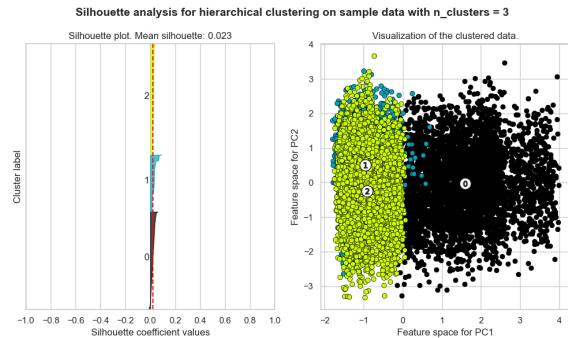


Figure 16: Análisis Silhouette

No parece que la división del conjunto de datos ayude. Parece haber un óptimo con 3, pero los resultados son muy pobres.

- Gaussian Mixture Models:

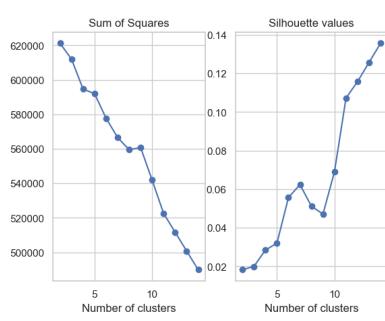


Figure 17: Suma de Cuadrados y Silhouette

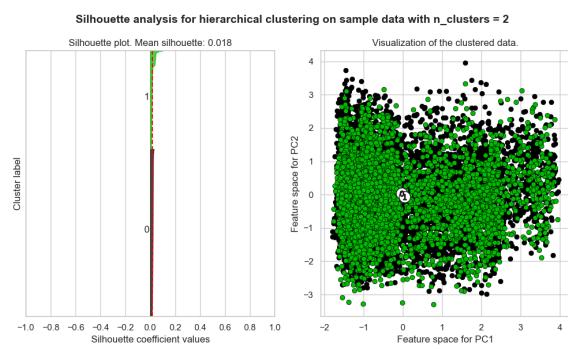


Figure 18: Análisis Silhouette

En este caso no hay un resultado bueno, y parece indicar que es mejor no usar clustering.

Como se puede ver fácilmente, no parece ser lógica la división de nuestro conjunto de datos en 2 o más subconjuntos, posiblemente debido a la gran simplicidad de dicho conjunto y a la gran cantidad de variables categóricas.

3.2 PDF

En este caso, hemos pensado que no nos aporta ninguna utilidad clara.

3.3 Bootstrap

Se plantea la posibilidad de emplear bootstrap una vez que hayamos seleccionado un conjunto final de modelos candidatos. La idea consiste en aplicar bootstrap a cada uno de estos modelos antes de tomar una decisión definitiva, con el propósito de obtener intervalos que nos proporcionen información sobre los rangos de error en los conjuntos de entrenamiento y prueba. Esto es especialmente relevante debido a que, aunque los modelos puedan ser similares en términos de rendimiento, algunos pueden presentar una variabilidad mayor que otros. Nos interesa identificar aquellos modelos con menor variabilidad. Sin embargo, este enfoque requiere realizar múltiples ejecuciones de bootstrap, lo cual demanda una capacidad computacional considerable que actualmente no poseemos.

Tras esto, se comienza con la creación de los modelos de predicción.

4 Modelos de ensamblaje

A continuación, se estudian diversas técnicas de ensamblaje vistas en clase, con un variado grado de complejidad, variabilidad y flexibilidad.

4.1 Árbol de Decisión

El Árbol de Regresión Estándar se establece como un modelo fundamental en el ámbito de la predicción del consumo de energía solar basándose en datos de irradiación solar. Este modelo permite una exploración detallada de cómo la irradiación directa juega un papel crucial en el aprovechamiento eficiente de la energía solar. Solo se han mostrado los primeros 4 cortes ya que la dimensión del árbol es de un tamaño considerable.

Para garantizar la efectividad y la precisión de los modelos presentados, se ha llevado a cabo un proceso meticuloso de optimización de hiperparámetros mediante la técnica de búsqueda en cuadrícula (GridSearch) aplicada al Árbol de Regresión Estándar. Este procedimiento ha permitido identificar una combinación óptima de hiperparámetros que maximizan la eficiencia del modelo en la tarea de predicción del consumo de energía solar. Los parámetros optimizados incluyen: un impurity-decrease de 0.0, lo cual indica que no se requerirá una disminución de impureza para realizar divisiones adicionales del nodo; un min-samples-leaf de 10, estableciendo el número mínimo de muestras que debe tener una hoja del árbol; y un min-samples-split de 23, especificando el número mínimo de muestras requeridas para dividir un nodo interno. La optimización de estos parámetros es crucial para el desempeño del modelo, ya que impacta directamente en la complejidad del árbol y, por ende, en su capacidad para generalizar bien a partir de los datos de entrenamiento sin caer en sobreajuste. Los resultados de esta búsqueda se puede ver en los siguientes gráficos.

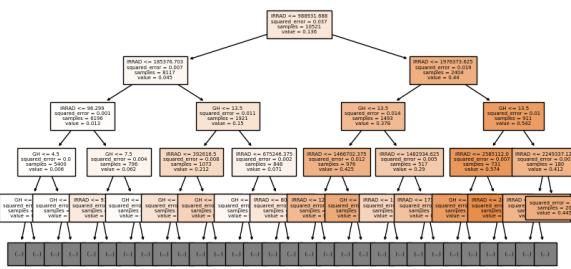


Figure 19: Primeros cortes del árbol

Como se puede observar en la figura del árbol de clasificación estándar, nuestro análisis destaca la preeminencia de la irradiación solar en las etapas iniciales de la decisión del modelo. Este factor es notoriamente predominante en los primeros cortes, lo cual indica que la muestra se divide primordialmente en función de los valores de irradiación.

Posteriormente, los parámetros óptimos identificados mediante GridSearch para el Árbol de Regresión Estándar han sido aplicados al estimador base tanto del modelo Bagging de Árboles de Regresión como del modelo Random Forest. Esta estrategia garantiza que los modelos de ensamble se construyan sobre la base de un estimador altamente optimizado, potenciando así su rendimiento en la predicción del consumo de energía solar. Al utilizar estos parámetros optimizados en los modelos de ensamble, se busca aprovechar las ventajas inherentes de cada técnica de ensamble, como la reducción de la varianza y el sobreajuste, manteniendo al mismo tiempo una estructura de modelo subyacente que ya ha demostrado ser eficaz para el conjunto de datos en cuestión.

Tras analizar las importancias de las variables en nuestro modelo de árbol de decisión estándar, los resultados confirmaron nuestras expectativas iniciales. Como anticipábamos, la irradiación solar emergió como la variable más significativa.

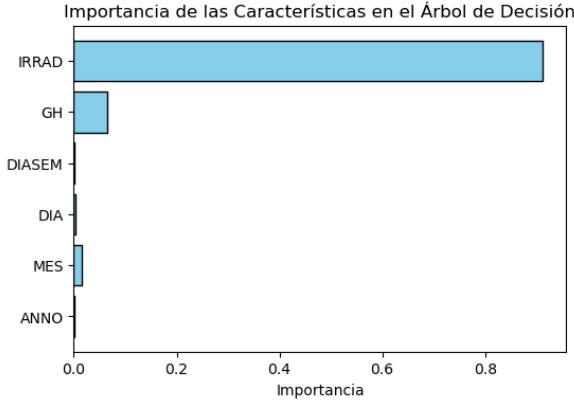


Figure 20: Importancia de variables en Arbol estandar

Los resultados obtenidos a partir del Árbol de Regresión Estándar destacan por su precisión, arrojando un Error Cuadrático Medio (MSE) de entrenamiento de aproximadamente 0.00099 y un MSE de prueba de cerca de 0.00145, complementados con un Error Absoluto Medio (MAE) de entrenamiento en torno a 0.016 y un MAE de prueba de aproximadamente 0.019. Estas métricas evidencian la alta precisión de las predicciones generadas por el modelo, demostrando su habilidad para ajustarse de manera efectiva a los datos de entrenamiento y, simultáneamente, mantener una buena capacidad de generalización ante nuevos conjuntos de datos. No obstante, la existencia de diferencias entre los errores de entrenamiento y prueba sugiere que aún existe espacio para mejorar la robustez del modelo ante variaciones en los datos.

La siguiente imagen muestra la distribución de las predicciones obtenidas por el árbol frente a los valores reales del conjunto AdjTest. Como se puede ver en la figura, la distribución se ajusta decentemente a lo que sería "ideal".

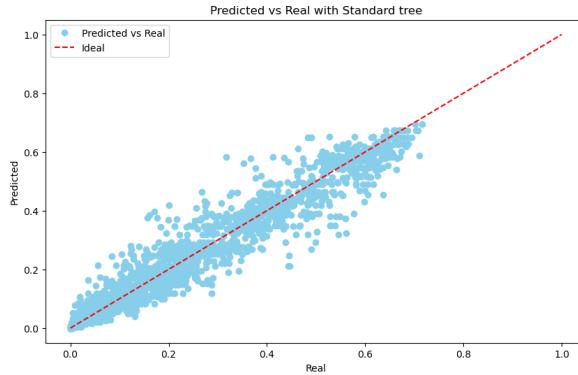


Figure 21: Predicciones del arbol estándar

4.2 Bagged Tree

La técnica de Bagging, cuando se aplica junto a los Árboles de Regresión, introduce un método de ensamblaje diseñado para incrementar la estabilidad y precisión de los modelos predictivos en la estimación del consumo de energía solar. Mediante la creación de múltiples árboles a partir de distintos subconjuntos de datos y la posterior combinación de sus predicciones, esta técnica aspira a disminuir la varianza y prevenir el sobreajuste, optimizando de este modo el aprovechamiento de la información proporcionada por la irradiación solar.

En el contexto de nuestro modelo de Bagged Tree, la evaluación de la importancia de las variables arrojó resultados alineados con nuestras predicciones previas. La variable de irradiación solar se destacó por encima de las demás, afirmando su posición como la más influyente en el modelo.

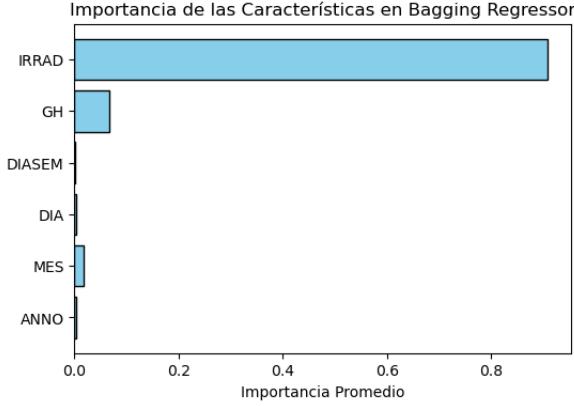


Figure 22: Importancia de variables en Bagged tree

Los resultados obtenidos con el modelo Bagging basado en Árboles de Regresión muestran una mejora significativa en términos de precisión, con un MSE de entrenamiento reducido a aproximadamente 0.00082 y un MSE de prueba de alrededor de 0.00120, además de presentar un MAE de entrenamiento de cerca de 0.015 y un MAE de prueba aproximadamente de 0.018. Estos indicadores resaltan la eficacia del Bagging para fortalecer la precisión de las predicciones, minimizando los errores y proporcionando un modelo más resistente a la variabilidad de los datos. La menor discrepancia observada entre los errores de entrenamiento y de prueba refleja igualmente una mejor capacidad de generalización del modelo.

A continuación se muestra la distribución de las predicciones contra los valores reales del bagged tree. Al igual que con el arbol estándar, la distribución que se observa es buena.

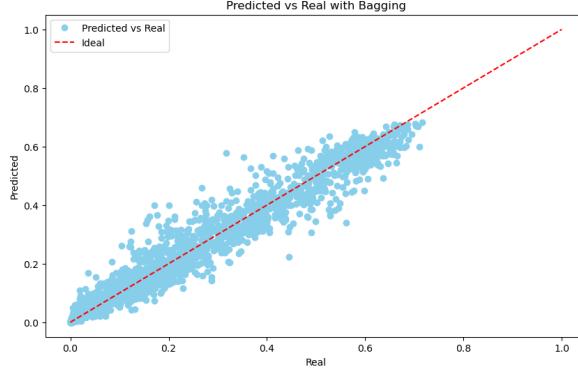


Figure 23: Predicciones del bagged tree

4.3 Random Forest

El modelo de Random Forest amplía la conceptualización del ensamblaje de árboles de decisión incorporando una selección aleatoria de características en cada punto de división, lo que fomenta una mayor diversidad entre los árboles individuales. Esta metodología no solo combate el sobreajuste sino que también mejora la habilidad del modelo para captar la complejidad de las relaciones entre la irradiación solar y el consumo de energía solar, destacando por su robustez y precisión en las predicciones.

El numero de estimadores óptimo para el random forest que se ha obtenido con un GridSearch por validación cruzada es de 140, como se puede ver en la siguiente imagen en la que con 140 estimadores se obtiene el error más bajo.

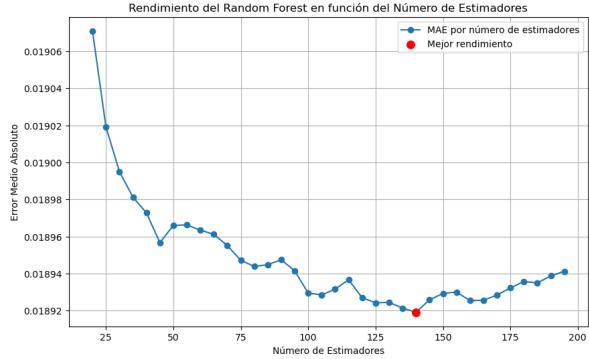


Figure 24: Busqueda de numero de estimadores optimo

La aplicación del modelo de Random Forest para la predicción del consumo de energía solar revela un MSE de entrenamiento de aproximadamente 0.00099 y un MSE de prueba de cerca de 0.00123, junto con un MAE de entrenamiento de alrededor de 0.017 y un MAE de prueba aproximado de 0.018. Estos resultados subrayan la capacidad del Random Forest para atenuar la varianza sin incrementar el sesgo de manera significativa, logrando un equilibrio óptimo entre el ajuste a los datos de entrenamiento y la capacidad de generalización. La mejora marginal en los errores de prueba, en comparación con el modelo estándar y el Bagging, indica que el Random Forest es particularmente eficaz en manejar la diversidad de los datos y en potenciar la precisión de las predicciones.

Al investigar la importancia de las variables dentro de nuestro modelo de Random Forest, los datos revelaron que la irradiación solar ostentaba la mayor importancia, coincidiendo plenamente con nuestras expectativas.

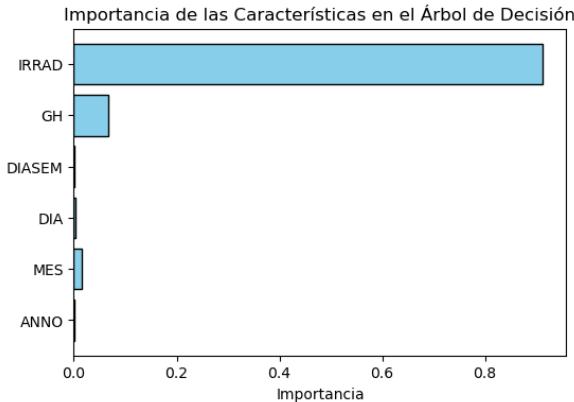


Figure 25: Importancia de variables en Random forest

En la siguiente imagen se puede ver de nuevo las predicciones contra los valores reales. Al igual que en el bagged tree y en el arbol estandar, el resultado de la distribucion es aceptable.

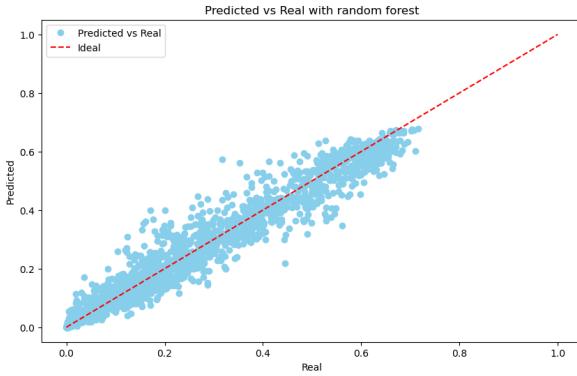


Figure 26: Predicciones del random forest

4.4 Boosting

Se van a probar algoritmos de Boosting para ver si se consigue una mejora respecto a los métodos previos. Para ello, se va a utilizar tanto AdaBoost como Gradient Boosting.

En métodos previos, se vio que el valor del parámetro de mínimo decreso de impureza que devolvía los modelos óptimos era 0, ya que de esta manera se conseguía llegar a árboles más complejos, lo cual parecía ser necesario para predecir la utilización en este conjunto de datos.

Por tanto, dicho parámetro es fijado a 0, y el estudio fundamentalmente se va a centrar en la optimización de 3 parámetros:

1. Máxima profundidad del estimador base(Árbol de decisión)
2. Número de estimadores
3. Learning rate

Debido a la naturaleza de los parámetros de número de estimadores y learning rate, se deberá buscar un trade-off entre ambos, de manera que al aumentar el número de estimadores, el learning rate probablemente deberá reducirse para evitar el sobreajuste.

También se debe tener en cuenta que, el número de estimadores que se ha podido probar ha sido limitado, debido al coste computacional de los modelos, por lo que, aunque en algunos casos la tendencia de mejora del error sugería que un aumento en el número de estimadores acompañado de una disminución del valor del learning rate llevaría a un mejor modelo, no ha sido posible llegar a entrenar modelos con más de 10000 árboles.

4.5 AdaBoost

Para el modelo de AdaBoost se probaron valores de máxima profundidad de 1 a 10, y en general, los modelos obtenidos siempre devolvían mejores predicciones para ***max_depth***=10.

Parece lógico que el modelo requiera de árboles más complejos para entrenar ya que, debido a la potencia de los ordenadores con que se ha entrenado, no se puede llegar al nº de estimadores necesario para que con árboles muy sencillos se alcance una precisión alta. En otras palabras, si se hubiera podido entrenar un modelo con 100000 árboles de profundidad máxima menor y un menor learning rate, es muy probable por las tendencias apreciadas de la función de error, que se hubiera conseguido una mejora en la precisión de test. Sin embargo, como se comentaba antes, esto no ha sido viable por temas de la capacidad computacional de nuestros ordenadores.

Una vez fijado `max_depth=10`, el estudio se reduce a conseguir el equilibrio adecuado entre los parámetros de n° de estimadores y learning rate. El resto de parámetros como `min_samples_split` o `min_samples_leaf` no son definidos, para dar la mayor libertad posible a los árboles entrenados.

Se prueban valores muy diversos de los parámetros de n° de estimadores y learning rate, llegando a los siguientes resultados:

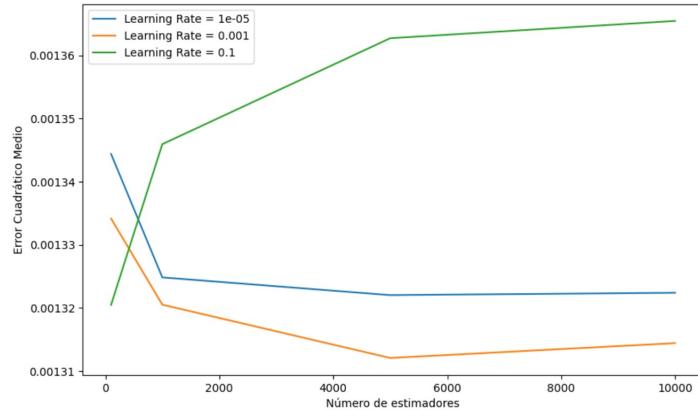


Figure 27: $MSE(N^o \text{ estimadores}, \text{Learning Rate})$ para AdaBoost

La figura muestra como, a partir de los 5000 estimadores, el error en general, se estabiliza, y además, se observa como, un valor intermedio del learning rate, 0.001 en este caso, devuelve los resultados óptimos.

Los resultados se presentarán al final de la sección junto a las métricas del siguiente modelo, para de esta manera, facilitar la comparación entre ambos. Sin embargo, es importante señalar que el modelo óptimo presentado parece estar algo sobreajustado ya que, el MSE de train es alrededor de 3 veces menor que el de validación. Para paliar este efecto, se ha probado a incluir un valor muy pequeño(órdenes de 10^{-6}) para el parámetro `min_impurity_decrease` que controla la complejidad del árbol, pero no se ha llegado a obtener ningún tipo de mejora en las métricas de test.

Por lo tanto, finalmente se considera como modelo de Ada Boost óptimo encontrado el cual tiene los parámetros:

- **Learning Rate=0.001**
- **Número de estimadores=5000**
- **Máxima profundidad de los árboles=10**

El resto de los parámetros de los árboles son fijados automáticamente para dar la mayor libertad posible en el entrenamiento.

En cuanto al gráfico de valores predichos por el modelo y valores reales, se muestra a continuación:

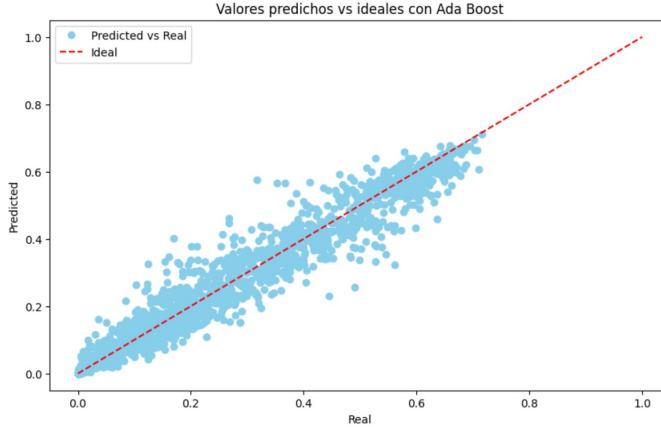


Figure 28: Valores predichos vs reales para AdaBoost

4.6 Gradient Boosting

Se va a desarrollar un estudio completamente análogo, pero esta vez utilizando modelos de Gradient Boosting usando el MSE como función de pérdida a optimizar. Cabe destacar que, los tiempos de computación requeridos para el entrenamiento de los modelos de Gradient Boosting es considerablemente menor a los de AdaBoost, por lo que el parámetro del nº de estimadores se ha podido aumentar algo más sin que los tiempos de computación se vuelvan excesivos.

Una vez más, se lleva a cabo un estudio inicial que nos permite fijar un valor para el parámetro de ***max_depth***, y una vez fijado dicho parámetro, se busca usando Cross-Validation, el par de parámetros óptimos para el nº de estimadores y el learning rate.

A partir de una primera búsqueda se decide fijar el parámetro de máxima profundidad de los árboles a 5, ya que, además de que el valor 10 devolvió un modelo algo sobreajustado para AdaBoost, tal y como se mostrará posteriormente en los resultados, probando diferentes modelos, el valor de 5 devolvió los mejores errores de Cross-Validation de entre todos los valores que se consideraban lógicos para ser probados(valores entre 1 y 10).

Por tanto, se procede a hacer una búsqueda más exhaustiva de los parámetros de learning rate y número de estimadores, probando en un inicio, valores muy similares a los testeados para AdaBoost.

Los resultados obtenidos son los siguientes:

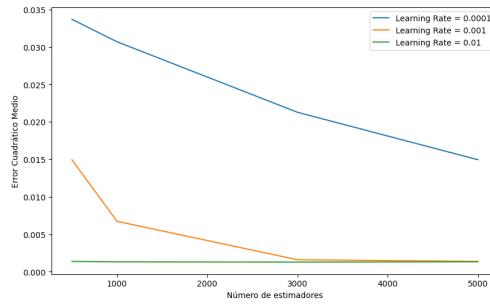


Figure 29: MSE(Learning Rate,Nº estimadores) en Gradient Boosting

El gráfico muestra como mayores valores del learning rate devuelven modelos más precisos, sin embargo, la tendencia decreciente del error para learning rate 0.0001 nos indica que simplemente, no se ha probado con un número de estimadores suficientemente grande. Por tanto, el estudio se continúa entrenando modelos con learning rate 0.0001 y hasta 50000 estimadores para ver si se mejora el modelo óptimo obtenido hasta ahora.

Los resultados obtenidos fueron mejores a los ya vistos, por lo que finalmente el modelo óptimo de Gradient Boosting es aquel con los siguientes parámetros:

- **Learning Rate=0.0001**
- **Número de estimadores=50000**
- **Máxima profundidad de los árboles=5**

El gráfico de valores predichos vs reales es:

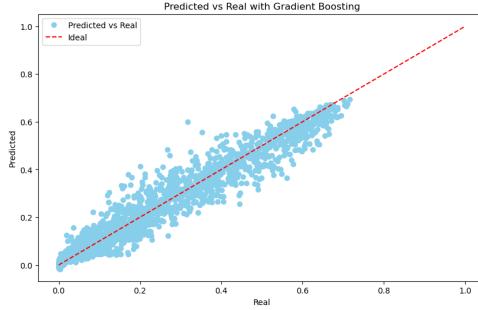


Figure 30: Valores predichos vs reales para Gradient Boosting

Observación(XGBoost): Se consideró probar modelos de XGBoost que nos permitieran optimizar el entrenamiento de los modelos, y de esta manera haber llegado a probar con un mayor n° de estimadores y un menor learning rate. Sin embargo, la librería **xgboost**, con la cual se debe implementar este algoritmo en Python, no está optimizada para el entrenamiento con variables categóricas en la actualidad, por lo que esta opción se desechó.

	MSE Entrenamiento	MSE Validación	MAE Entrenamiento	MAE Validación
AdaBoost	0.000440	0.001205	0.011047	0.017753
Gradient Boosting	0.001108	0.001275	0.018925	0.019748

Table 1: MAE y MSE de los modelos de Gradient Boosting y AdaBoost

Si bien el modelo de AdaBoost devuelve mejores resultados de test, también es cierto que presenta una cantidad considerable de sobreajuste, por lo que dependiendo del riesgo que se decida tomar a la hora de hacer las predicciones será más adecuado usar AdaBoost, si se quieren asumir más riesgos, o Gradient Boosting en caso de que se quiera optar por un modelo más estable.

4.7 Stacking

El modelo de stacking involucra la combinación de las predicciones de múltiples modelos base para generar una predicción final. Esta técnica ofrece flexibilidad al permitir la inclusión de diversos modelos de aprendizaje automático como componentes base. En este estudio, evaluamos la eficacia del stacking mediante la combinación de modelos base tales como árboles de decisión simples y modelos polinómicos, así como modelos de boosting y redes neuronales, con el objetivo de mejorar la capacidad predictiva en un contexto específico.

Inicialmente, implementamos el stacking combinando un árbol de decisión simple con un modelo polinómico. Evaluamos el desempeño de esta combinación en términos de error cuadrático medio (MSE) y error absoluto medio (MAE) tanto en conjuntos de entrenamiento como de prueba. Posteriormente, exploramos la combinación de un modelo de boosting con una red neuronal, nuevamente utilizando el stacking. Se registraron los MSE y MAE correspondientes para cada caso.

La combinación de un árbol de decisión simple y un modelo polinómico mediante stacking resultó en un MSE de entrenamiento de 0.0012298384943441247 y un MSE de prueba de 0.0017803233807022357, con

un MAE de entrenamiento de 0.023792830360222554 y un MAE de prueba de 0.027339933455514095. Estos resultados mostraron un desempeño inferior en comparación con el uso exclusivo de un árbol de decisión simple, sugiriendo una posible falta de complejidad en el problema o la presencia de ruido en los datos.

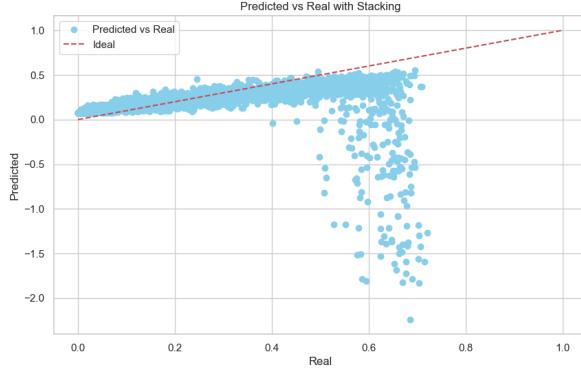


Figure 31: Stacking Simple Decision Tree

Por otro lado, la combinación del modelo elegido de boosting con una red neuronal mediante stacking produjo un MSE de entrenamiento de 0.0006879343681195563 y un MSE de prueba de 0.0017545699280656626, con un MAE de entrenamiento de 0.01667710791960603 y un MAE de prueba de 0.02447564563087598. Estos resultados, aunque mejoraron en comparación con la combinación anterior, aún mostraron un desempeño inferior al del modelo de boosting individual, lo que sugiere posibles limitaciones en la complejidad del problema o la presencia de ruido.

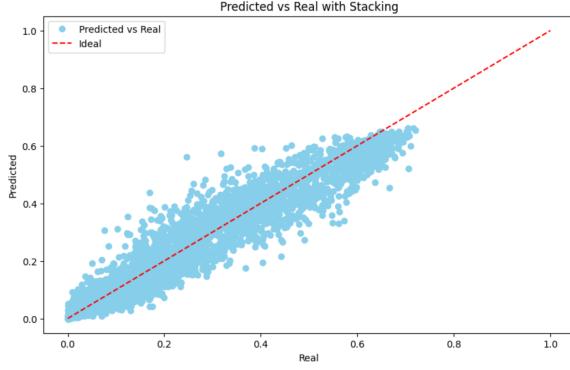


Figure 32: Stacking Boosting y MLP

Durante el entrenamiento de la red neuronal, se experimentó con diferentes configuraciones de neuronas. Se observó que el modelo mejoraba su rendimiento a medida que se incrementaba el número de neuronas. La configuración final seleccionada fue (4000, 100, 100, 100, 100, 4000). Sin embargo, para lograr un mejor desempeño de la red, habría sido apropiado seleccionar un número aún mayor de neuronas. Esta decisión no se tomó debido a limitaciones en la capacidad computacional.

Los resultados obtenidos sugieren que el uso de stacking puede no ser beneficioso en todos los casos, especialmente cuando los modelos base son relativamente simples o cuando el problema en cuestión no es lo suficientemente complejo como para justificar la combinación de modelos. Al estar entrenando el modelo generando ruido en los datos, lo que puede afectar negativamente el desempeño del stacking.

5 Comparación de modelos y conclusiones

La última sección se basa en la selección de los modelos principales y su comparación, de cara a la extracción de conclusiones y elección final.

En nuestro caso se han seleccionado como modelos finales el árbol de decisión, bagging y gradient boosting. Random Forest se ha descartado debido a que, a pesar de que a priori en nuestro caso de uso particular, donde claramente hay 1 variable predominante para la predicción de la respuesta, random forest debería funcionar mejor que bagging, no lo cumple. La hipótesis que se ha formulado a raíz de este hecho es la siguiente:

Random Forest suele ser mejor que Bagging en la mayoría de casos teóricamente hablando, pero es especialmente mejor cuando nos encontramos con una cantidad suficiente de predictores, donde hay 1 que es más significativo que el resto, pero luego se tienen una serie de predictores también relevantes y muy correlados entre sí. De esta forma, random forest te permite crear árboles decorrelados sin aumentar mucho la varianza, y los resultados son mejores. No obstante, en nuestro caso todo indica que hay 1 variable que es esencial y es la que se encarga de prácticamente todos los cortes. Por tanto, en los cortes de cada árbol de random forest donde no se haya escogido, la salida va a ser peor que cualquiera de los árboles de Bagging, de ahí las diferencias entre ambos modelos.

Por tanto, se escoge Bagging al ser un enfoque correcto para nuestro problema y permitir reducir la varianza frente a un árbol de decisión. Aún así, se escoge el árbol de decisión también debido a que las diferencias en los errores no son muy grandes y es el único modelo fácilmente interpretable. Por otro lado, boosting tampoco queda seleccionado debido a su alto coste computacional y la obtención de resultados similares a gradient boosting, el cual sabemos que es un algoritmo de aproximación rápido, lo que permite reducir un poco el coste computacional. Por último, los resultados de stacking no son muy prometedores, debido a que la complejidad necesaria para obtener errores decentes es demasiado elevada, no siendo por tanto rentable frente a algoritmos más sencillos. A continuación comparamos el RMSE negativo para cada modelo mediante validación cruzada, observamos los resultados de diversas métricas (R^2 , MAE y MSE) y exponemos gráficas tanto de los residuos de test, como del análisis de residuos para cada modelo.

- **single_reg_tree_fit (Árbol de Regresión Simple):**
 - **Criterion:** squared_error
 - **Min_samples_leaf:** 10
 - **Min_samples_split:** 23
 - **Min_impurity_decrease:** 0.0
 - **Random_state:** 0
- **bt_fit (BaggingRegressor con Árbol de Decisión Base):**
 - **N_estimators:** 140 (Número de estimadores en el ensamble)
 - **Base_estimator (Árbol de Decisión Base):**
 - * **Criterion:** squared_error
 - * **Min_samples_leaf:** 10
 - * **Min_samples_split:** 23
 - * **Min_impurity_decrease:** 0.0
 - * **Random_state:** 0
- **gb2_fit (Gradient Boosting Regressor):**
 - **Loss:** squared_error
 - **N_estimators:** 50000
 - **Learning_rate:** 0.0001
 - **Max_depth:** 5
 - **Random_state:** 0

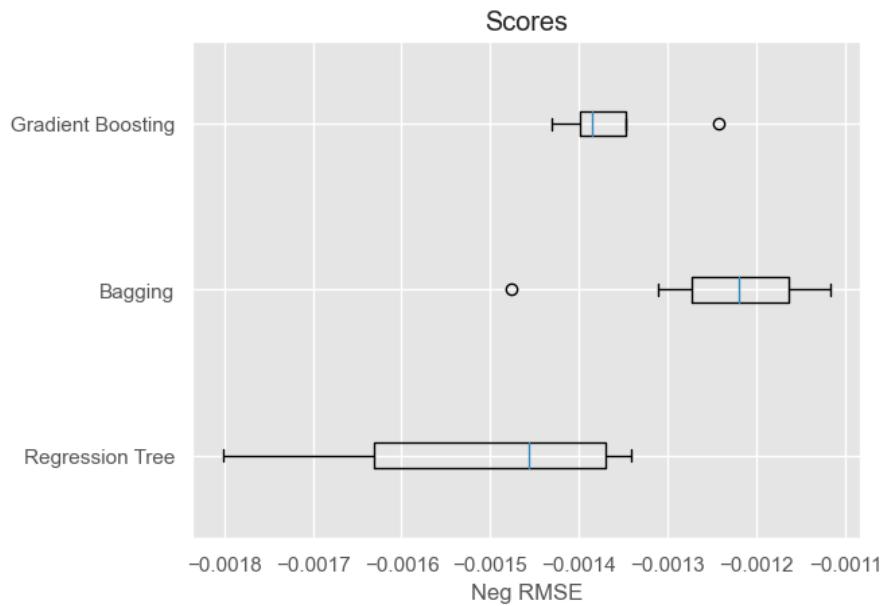


Figure 33: Comparación RMSE negativo

	MSE Entrenamiento	MSE Validación	MAE Entrenamiento	MAE Validación
single_reg_tree.fit	0.000949	0.001470	0.015840	0.019419
bt.fit	0.000781	0.001184	0.014561	0.017666
gb2.fit	0.001108	0.001276	0.018926	0.019749

Table 2: MAE y MSE del árbol, bagging y gradient boosting

	R2 Entrenamiento	R2 Validación
single_reg_tree.fit	0.974356	0.959810
bt.fit	0.978884	0.967639
gb2.fit	0.970046	0.965121

Table 3: R^2 del árbol, bagging y gradient boosting

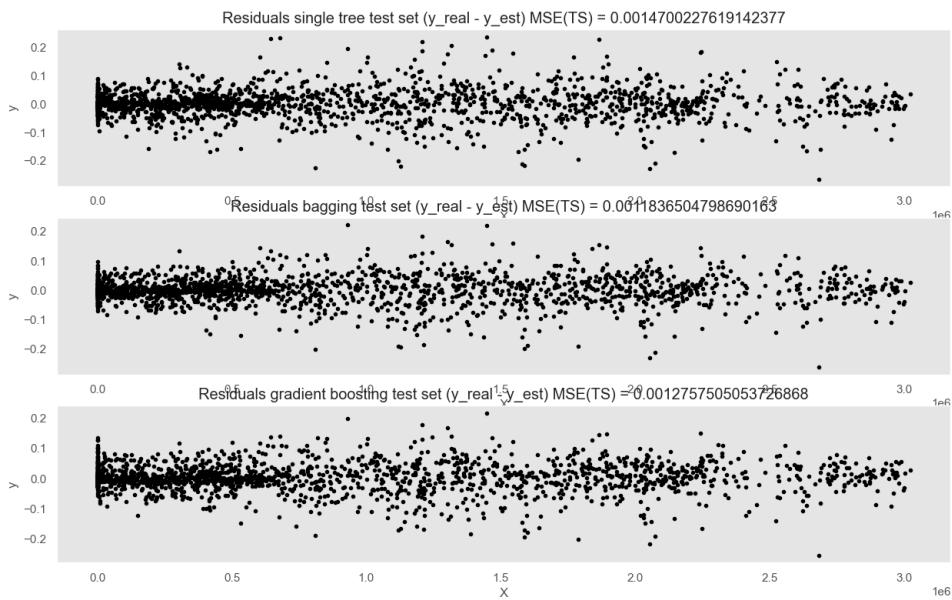


Figure 34: Residuos de Test

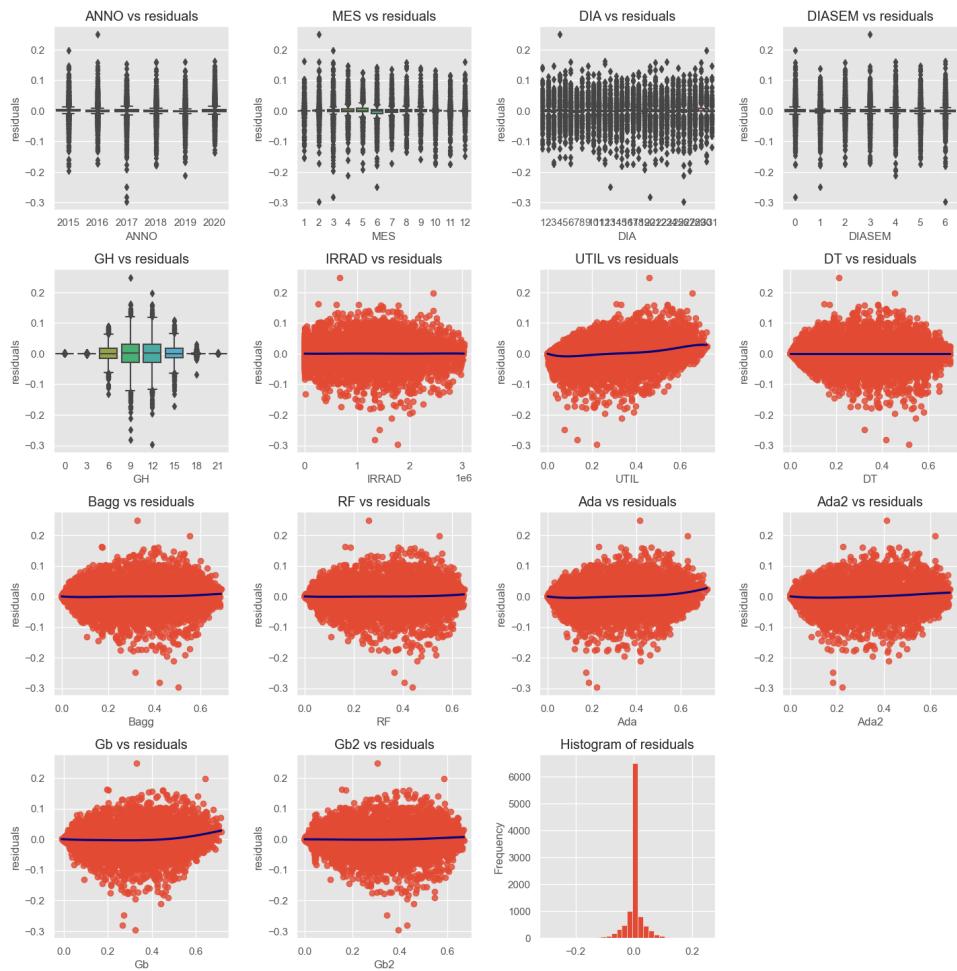


Figure 35: Análisis de residuos

Además, tal y como se comentaba anteriormente, se hace un intento de aplicación de bootstrap para 10 muestras, de cara a indicar la metodología a seguir si se dispusiese de mayor potencia.

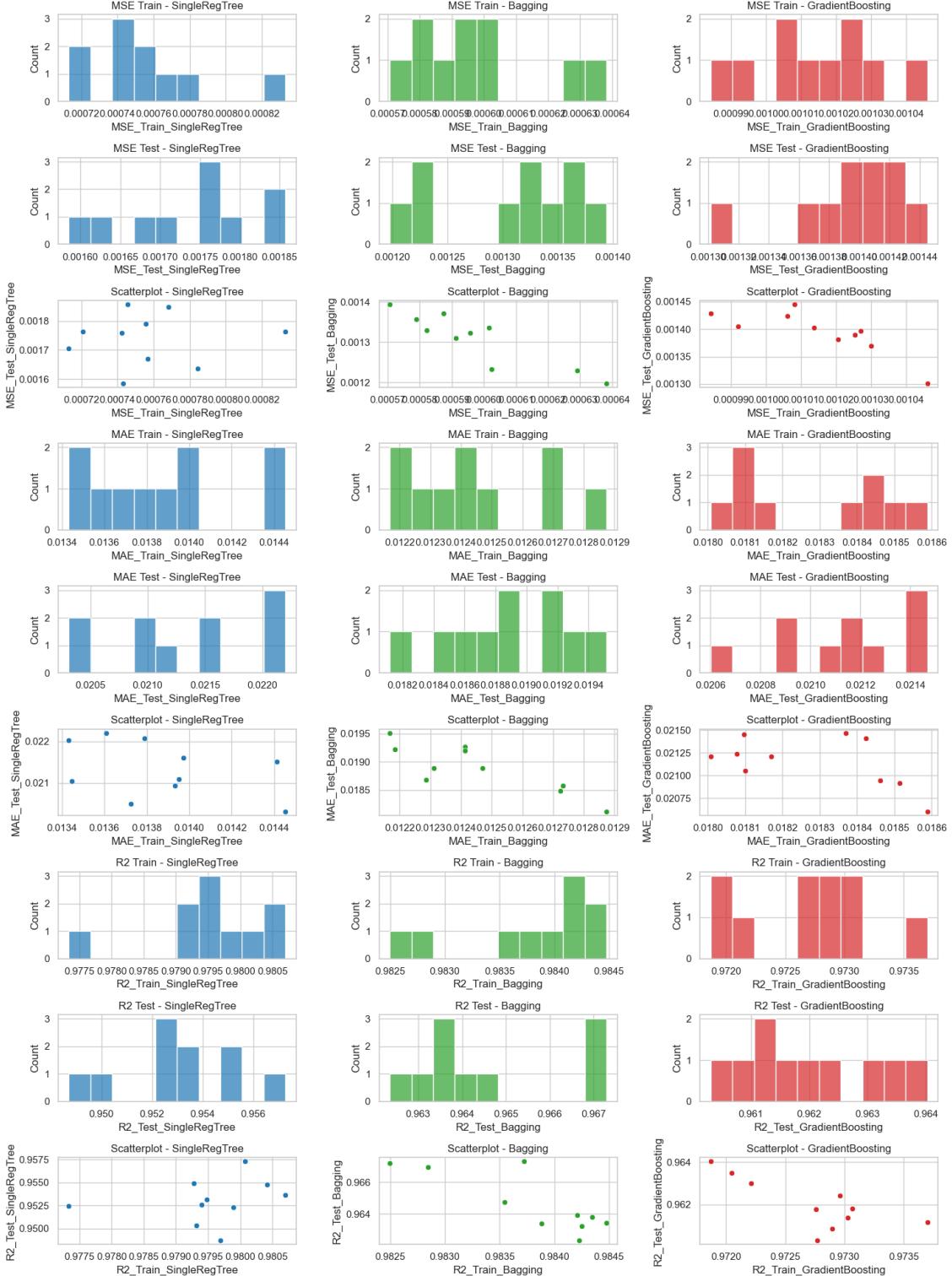


Figure 36: Bootstrap para métricas

Como se puede observar en los resultados (de bootstrap no se pueden extraer conclusiones sólidas debido a la falta de ejecuciones), en cuanto al comportamiento de los errores no hay una diferencia perceptible. Todos ellos se encuentran centrados, con un comportamiento análogo y siendo menores

cuando el valor del eje horizontal (la irradiación) es cercano a 0, como es lógico. Es cierto que nuestra respuesta con respecto a los residuos puede no encontrarse completamente centrada en 0 para los valores más altos de utilización, posiblemente debido al límite de las placas, donde nos encontramos con un límite superior que no se puede sobrepasar. Aún así, los resultados son bastante buenos para los modelos, haciendo que sea necesario fijarse en los resultados de las métricas.

- Resultados de métricas: Bagging.
- Buen ratio de métricas-interpretabilidad: Árbol de Decisión.
- Más consistencia train-test: Gradient Boosting.

Tras esto, se descarta Gradient Boosting debido a que los resultados son peores, a pesar de que sean más consistentes entre entrenamiento y test, y si quisiésemos mejorar estos resultados es posible que incurriésemos en un mayor riesgo de sobreajuste, algo que no ocurre en otros modelos de ensamble como Bagging. Además, también es relevante el coste computacional, siendo Bagging más de 30 veces más rápido en cualquier ejecución.

Por último, para decidir entre los modelos restantes, hay argumentos a favor y en contra de cada uno de ellos, como ya se ha comentado (al igual que para la mayoría de modelos entrenados, casi todos han tenido resultados muy similares y bastante buenos). En nuestro caso se ha decidido una solución híbrida en la que, mientras que el modelo final va a ser Bagging debido a la reducción de varianza y los mejores resultados frente al árbol, el Árbol de Decisión también va a ser escogido a modo de indicador para que, si alguien quiere entender el modelo que realiza las predicciones, pueda basarse en el árbol (al fin y al cabo los árboles de bagging son muy similares a ése en su mayoría) para entender el funcionamiento y los cortes del modelo. A continuación, se utiliza el conjunto de test guardado durante todo el proceso para comprobar si realmente nuestro modelo era una buena elección, o si nos hemos equivocado y nuestras conclusiones eran erróneas (se va a entrenar el árbol, bagging, random forest y gradient boosting, con los hiperparámetros óptimos, en el 80% de los datos, y se observarán las métricas de entrenamiento y del test real, sabiendo que nuestro modelo final es Bagging).

6 Comprobación del Modelo Final

	MSE Entrenamiento	MSE Validación	MAE Entrenamiento	MAE Validación
single_reg_tree_fit	0.001233	0.001802	0.017573	0.022247
bt_fit	0.000928	0.001346	0.015462	0.019535
rf_fit	0.001317	0.001764	0.020130	0.023356
gb2_fit	0.001175	0.001454	0.019588	0.021537

Table 4: MAE y MSE del árbol, bagging, random forest y gradient boosting

	R2 Entrenamiento	R2 Validación
single_reg_tree_fit	0.966675	0.952363
bt_fit	0.974915	0.964411
rf_fit	0.964395	0.953371
gb2_fit	0.968232	0.961568

Table 5: R^2 del árbol, bagging, random forest y gradient boosting

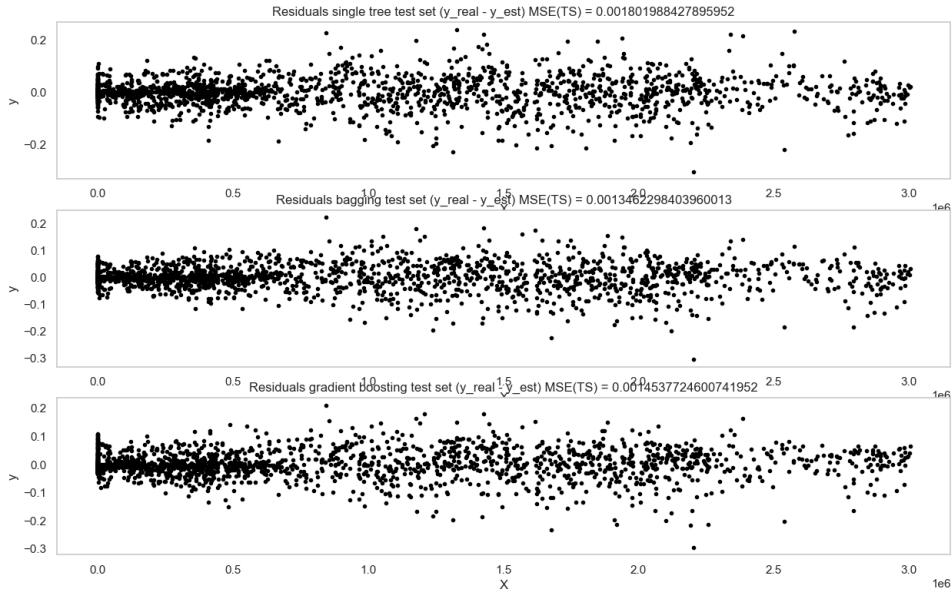


Figure 37: Residuos de Test

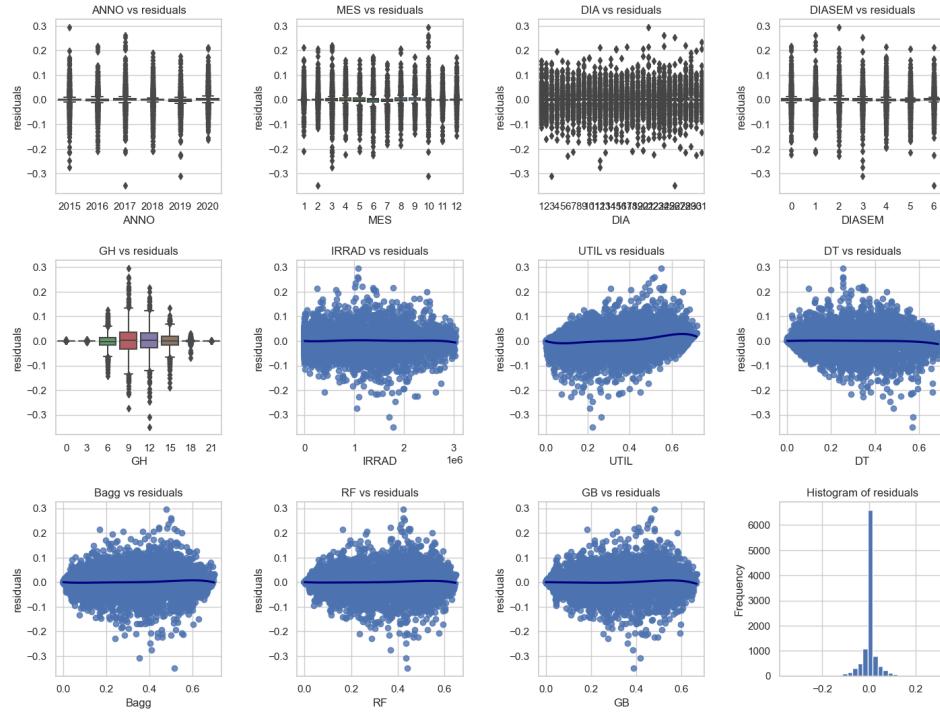


Figure 38: Análisis de residuos

Como se puede observar, la elección ha sido la acertada. No hay diferencias entre las métricas de entrenamiento y de test mayores a las del resto de modelos, a excepción de Gradient Boosting, sus valores son los más bajos, los residuos se comportan correctamente, etc.

De esta forma, el modelo final es el escogido anteriormente: Bagging.

7 Propuestas de Mejora

A pesar de todo el trabajo realizado, sin duda se podrían realizar diversas mejoras a lo largo de todo el modelo:

- La adición de instantes anteriores de irradiación y utilización como predictores, aumentando la complejidad del modelo y el número de predictores significativos.
- La ejecución de más modelos de ensamble:
 - Stacking con modelos más complejos.
 - Gradient Boosting con mayor grid search, XGBoost, LightGBM, XGBoost2, etc.
 - Stacking de modelos de ensambles.
- La aplicación de bootstrap para la comparación de los modelos y la selección.

Con todas estas mejoras, los modelos podrían ser mucho más precisos, aunque los obtenidos son bastante decentes. Tras este último apartado de autocritica, el artículo queda terminado.