HistGradientBoostingRegressor

April 2, 2023

0.1 #Introducción

TFM: Aplicación de ciencia de datos en el sector de producción animal para la predicción y explicación de óptimos en ganado porcino.

Titulo: **HistGradientBoostingRegressor**

Autor: **Jose Eduardo Cámara Gómez**

0.2 Importar paquetes

[1]:
```
# Importación de paquetes
import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
from matplotlib.pyplot import figure

from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler

from sklearn.ensemble import HistGradientBoostingRegressor

sns.set(style="darkgrid")
```

[2]:
```
from google.colab import files
# Cargamos el fichero del dataset gmd_02.csv
uploaded = files.upload()
# Leemos el fichero csv con los datos
df = pd.read_csv('gmd_02.csv', sep=';')
# Revisar la raza si se agrupan las razas con menos ocurrencias
```

```
agrupar_razas = {93 : 93, 85 : 93, 90 : 93, 95 : 93, 94 : 93, 82 : 93, 80 : 80,␣
 ↪96 : 80, 88 : 88, 0 : 0, 23 : 0,
                 84 : 0, 66 : 0, 18 : 0, 68 : 88, 7 : 7, 89 : 7, 65 : 7, 15 :␣
 ↪15, 97 : 7, 69 : 69, 81 : 81}
df.replace({'ct_raza' : agrupar_razas}, inplace=True)
df["bajas"] = df["NumBajas"] / (df["NumAnimales"] * df["DiasMedios"])
# Convertimos los tipos
df["ct_integra"] = df["ct_integra"].astype("category")
#df["ct_tipo"] = df["ct_tipo"].astype("category")
df["ct_raza"] = df["ct_raza"].astype("category")
df["ct_fase"] = df["ct_fase"].astype("category")
df['EntradaInicial']= pd.to_datetime(df['EntradaInicial'])
df['EntradaFinal']= pd.to_datetime(df['EntradaFinal'])
df["na_rega"] = df["na_rega"].astype("category")
df["NumBajas"] = df["NumBajas"].astype("int64")
df["gr_codpos"] = df["gr_codpos"].astype("category")
df["gr_poblacion"] = df["gr_poblacion"].astype("category")
df["na_nombre2"] = df["na_nombre2"].astype("category")

# Funcion para convertir en One Hot Encoding
def encode_and_bind(original_dataframe, feature_to_encode):
    dummies = pd.get_dummies(original_dataframe[[feature_to_encode]])
    res = pd.concat([original_dataframe, dummies], axis=1)
    res = res.drop([feature_to_encode], axis=1)
    return(res)
# Cargamos las variables objetivo y las usadas (15 variables seleccionadas, una␣
 ↪de ellas categórica con 8 valores).
y = df['GMD']
x0 = df[['ct_integra','ct_tipo', 'ct_raza', 'IncPeso', 'NumAnimales', 'na_rega',
         'PesoEntMedio', 'PesoRecMedio', 'bajas', 'GPS_Longitud', 'GPS_Latitud',
         'semanaEntrada', 'añoEntrada', 'PorcHembras', 'PiensoCerdaDia']]
features_to_encode = ['ct_raza']   # , 'na_rega']
x1 = x0.copy()
x1.drop(['ct_integra','na_rega'], inplace=True, axis=1)
for feature in features_to_encode:
    x1 = encode_and_bind(x1, feature)
```

```
<IPython.core.display.HTML object>
```

```
Saving gmd_02.csv to gmd_02.csv
```

[3]:
```
X_train, X_test, y_train, y_test = train_test_split(x1, y, test_size = 0.2,␣
 ↪random_state = 123)
## Vemos de escalar las variables para que no se vean influenciadas por los␣
 ↪outliers.
scaler = RobustScaler()
scaler.fit(X_train)
X_train_s = scaler.transform(X_train)
```

```
X_test_s = scaler.transform(X_test)
```

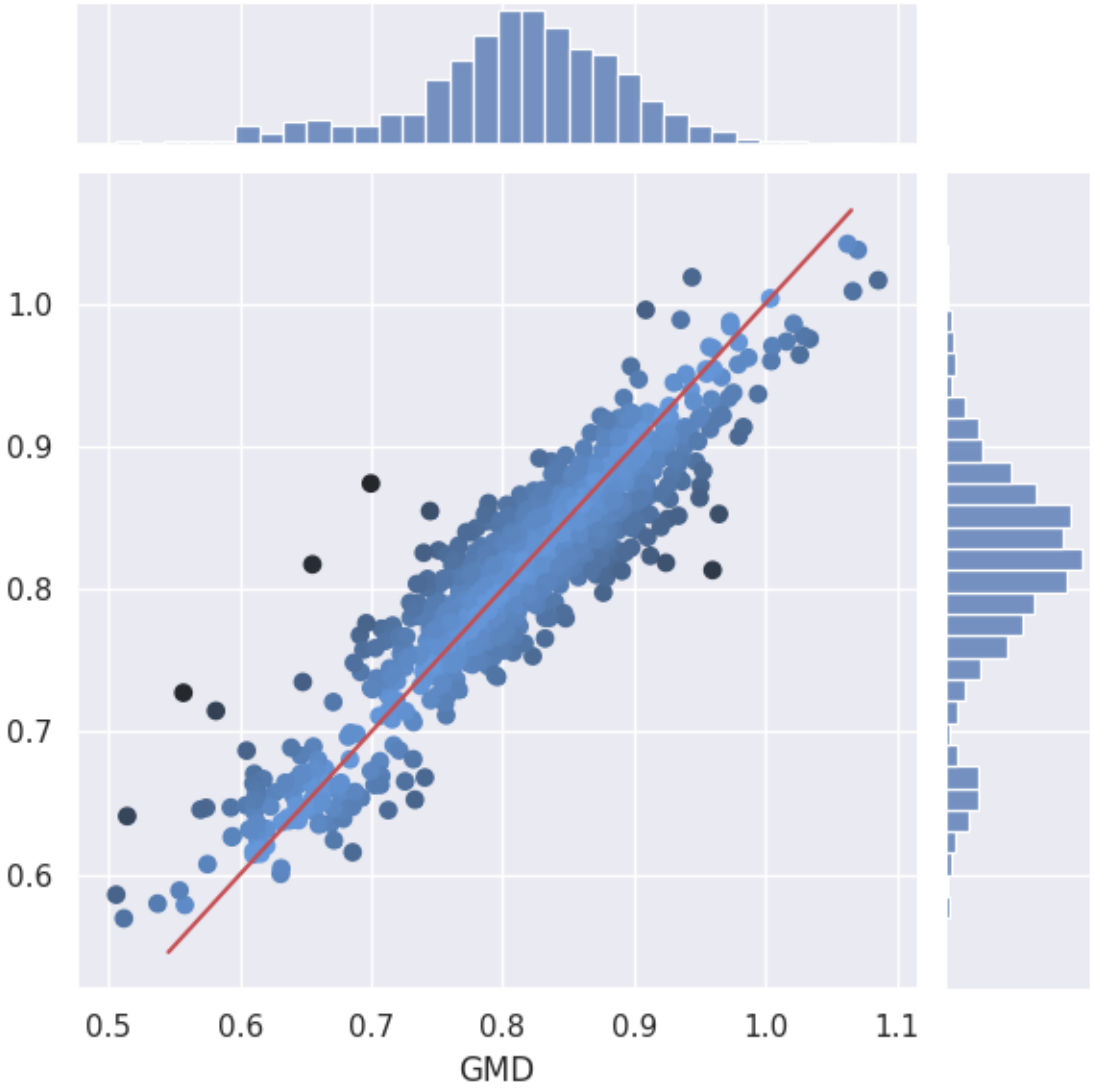
[4]:
```
# Lanzamos el modelo con los parámetros por defecto
modelo = HistGradientBoostingRegressor(
            loss = "squared_error",
            learning_rate=0.1,
            max_iter=100,
            max_leaf_nodes=31,
            random_state = 123
         )
modelo.fit(X_train_s, y_train)
```

[4]:
```
HistGradientBoostingRegressor(random_state=123)
```

[5]:
```
# Función para Graficar diferencias entre valor predicho y real en datos de␣
 ↪test del modelo pasado
def graficoDiferencias(modelo, X_test_s, y_test):
    y_pred = modelo.predict(X_test_s)
    diferencia = abs(y_pred - y_test)
    g = sns.jointplot(x=y_test, y=y_pred)
    # Draw a line of x=y
    x0, x1 = g.ax_joint.get_xlim()
    y0, y1 = g.ax_joint.get_ylim()
    lims = [max(x0, y0), min(x1, y1)]
    g.ax_joint.plot(lims, lims, '-r')
    g.ax_joint.scatter(x=y_test, y=y_pred, c=diferencia.values, cmap=sns.
 ↪dark_palette("#69d", reverse=True, as_cmap=True))
    plt.show()

# Graficar las diferencias
print('Score R2:',modelo.score(X_test_s, y_test))
graficoDiferencias(modelo, X_test_s, y_test)
```

```
Score R2: 0.8476511539323192
```



[6]:
```
# Analizamos otros errores del método
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import max_error
y_pred = modelo.predict(X_test_s)
# Definimos la función con las métricas a mostrar
def mostrar_metricas(y_test, y_pred):
   print("Metr.\t Valor\t\t\t Descripción")
   print("R^2 \t", r2_score(y_test, y_pred), "\t (Coeficiente de␣
 ↪Determinación)")
   print("RMSE\t", mean_squared_error(y_test, y_pred, squared=True), "\t (Raíz␣
 ↪de error cuadrático medio)")
```

```
   print("MAE \t", mean_absolute_error(y_test, y_pred), "\t (Error absoluto␣
 ↪medio)")
   print("MAX \t", max_error(y_test, y_pred), "\t (Error Máximo)")
# Pedimos que muestre las métricas para el modelo de RandomForest
print("Métricas para RandomForest v1")
mostrar_metricas(y_test, y_pred)
```

```
Métricas para RandomForest v1
Metr.    Valor                  Descripción
R^2      0.8476511539323192     (Coeficiente de Determinación)
RMSE     0.0010916783631631634  (Raíz de error cuadrático medio)
MAE      0.02515203975776052    (Error absoluto medio)
MAX      0.17427550398957703    (Error Máximo)
```

1 Optimización de Hiperparámetros

[7]:
```
# Hacemos una optimización de los hiperparámetros básica para RandomForest y␣
 ↪vemos de probar de forma aleatoria con la combinación de 100 de estos␣
 ↪modelos a ver cuál es el que mejor ajusta el MSRE.
from sklearn.model_selection import RandomizedSearchCV
random_grid = {'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.2, 0.25, 0.3, 0.5],
               'max_iter': [10, 30, 50, 100, 200, 500, 1000],
               'max_leaf_nodes': [5, 10, 20, 30, 50, 100, 200]}
modelo_random = RandomizedSearchCV(scoring="neg_mean_squared_error", estimator␣
 ↪= modelo,
                                   param_distributions = random_grid, n_iter = 100,
                                   cv = 3, verbose=1, random_state=123, n_jobs = -1)
modelo_random.fit(X_train_s, y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

[7]:
```
RandomizedSearchCV(cv=3,
                   estimator=HistGradientBoostingRegressor(random_state=123),
                   n_iter=100, n_jobs=-1,
                   param_distributions={'learning_rate': [0.0001, 0.001, 0.01,
                                                          0.1, 0.2, 0.25, 0.3,
                                                          0.5],
                                       'max_iter': [10, 30, 50, 100, 200, 500,
                                                    1000],
                                       'max_leaf_nodes': [5, 10, 20, 30, 50,
                                                          100, 200]},
                   random_state=123, scoring='neg_mean_squared_error',
                   verbose=1)
```

[9]:
```
modelo_random.best_params_
```

[9]:
```
{'max_leaf_nodes': 5, 'max_iter': 1000, 'learning_rate': 0.1}
```

[8]:
```
print("Estimadores para Train")
mostrar_metricas(y_train, modelo_random.best_estimator_.predict(X_train_s))
print("")
print("Estimadores para Test")
mostrar_metricas(y_test, modelo_random.best_estimator_.predict(X_test_s))
```

```
Estimadores para Train
Metr.    Valor                  Descripción
R^2      0.9181345063853255     (Coeficiente de Determinación)
RMSE     0.0005632649816280457  (Raíz de error cuadrático medio)
MAE      0.01841218605638466    (Error absoluto medio)
MAX      0.1369394846401235     (Error Máximo)

Estimadores para Test
Metr.    Valor                  Descripción
R^2      0.8438046323594022     (Coeficiente de Determinación)
RMSE     0.0011192411867944516  (Raíz de error cuadrático medio)
MAE      0.02530588487320156    (Error absoluto medio)
MAX      0.17435190880309026    (Error Máximo)
```

Como se puede apreciar el método parece mejorar en entrenamiento, pero es únicamente sobreajustándose al modelo, ya que si vemos el error en datos de test este no ha mejorado nada, más bien ha empeorado ligeramente.

Intentamos ahora hacerlo con early_stopping. Este modelo permite ser lanzado con la opción de early stopping, y es esta buscar optimizar no los resultados para train si no para test (reserva un porcentaje de los datos de entrenamiento para realizar la validación con datos no usados). Podemos ponerle un número de iteaciones máximas muy grande pues se espera que acabe antes el proceso por no mejorar según las condiciones de early stopping.

[10]:
```
modelo2 = HistGradientBoostingRegressor(
            loss = "squared_error",
            learning_rate=0.1,
            max_iter=50000,
            max_leaf_nodes=31,
            early_stopping=True,
            scoring='loss',
            validation_fraction=0.1,
            n_iter_no_change=10,
            tol=1e-07,
            verbose=1,
            random_state = 123
         )
modelo2.fit(X_train_s, y_train)
# Mostramos los errores mediante este método
print("Estimadores para Train")
mostrar_metricas(y_train, modelo2.predict(X_train_s))
print("")
print("Estimadores para Test")
```

```
mostrar_metricas(y_test, modelo2.predict(X_test_s))
## Mostramos la gráfica del entrenamiento
graficoDiferencias(modelo2, X_test_s, y_test)
```

```
Binning 0.001 GB of training data: 0.022 s
Binning 0.000 GB of validation data: 0.001 s
Fitting gradient boosted rounds:
[1/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00293, val loss:
0.00315, in 0.021s
[2/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00253, val loss:
0.00277, in 0.011s
[3/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00220, val loss:
0.00247, in 0.010s
[4/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00194, val loss:
0.00222, in 0.011s
[5/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00172, val loss:
0.00200, in 0.013s
[6/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00154, val loss:
0.00183, in 0.011s
[7/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00139, val loss:
0.00168, in 0.012s
[8/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00127, val loss:
0.00156, in 0.012s
[9/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00116, val loss:
0.00146, in 0.012s
[10/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00107, val loss:
0.00138, in 0.012s
[11/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00100, val loss:
0.00132, in 0.011s
[12/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00094, val loss:
0.00126, in 0.011s
[13/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00088, val loss:
0.00120, in 0.012s
[14/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00084, val loss:
0.00116, in 0.012s
[15/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00080, val loss:
0.00112, in 0.011s
[16/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00076, val loss:
0.00109, in 0.011s
[17/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00073, val loss:
0.00106, in 0.011s
[18/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00071, val loss:
0.00104, in 0.022s
[19/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00068, val loss:
0.00101, in 0.017s
[20/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00066, val loss:
0.00099, in 0.010s
[21/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00064, val loss:
```

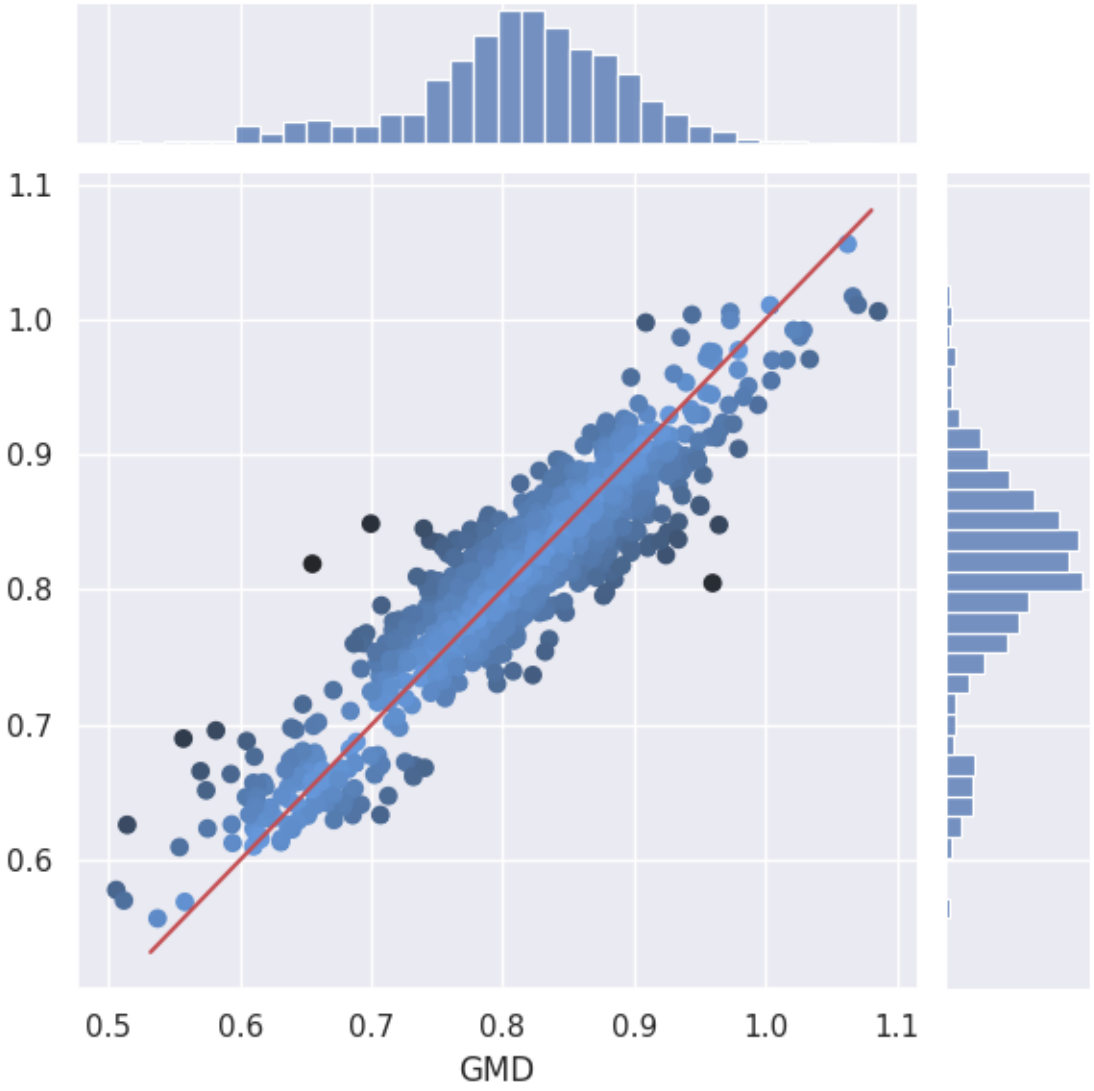
```
0.00097, in 0.010s
[22/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00063, val loss:
0.00096, in 0.014s
[23/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00061, val loss:
0.00094, in 0.011s
[24/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00059, val loss:
0.00093, in 0.010s
[25/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00058, val loss:
0.00091, in 0.013s
[26/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00057, val loss:
0.00090, in 0.012s
[27/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00055, val loss:
0.00089, in 0.010s
[28/50000] 1 tree, 31 leaves, max depth = 7, train loss: 0.00054, val loss:
0.00088, in 0.010s
[29/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00053, val loss:
0.00087, in 0.026s
[30/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00052, val loss:
0.00086, in 0.012s
[31/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00051, val loss:
0.00085, in 0.011s
[32/50000] 1 tree, 31 leaves, max depth = 14, train loss: 0.00050, val loss:
0.00084, in 0.011s
[33/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00049, val loss:
0.00084, in 0.010s
[34/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00049, val loss:
0.00083, in 0.014s
[35/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00048, val loss:
0.00083, in 0.011s
[36/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00047, val loss:
0.00082, in 0.010s
[37/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00046, val loss:
0.00081, in 0.013s
[38/50000] 1 tree, 31 leaves, max depth = 7, train loss: 0.00046, val loss:
0.00081, in 0.009s
[39/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00045, val loss:
0.00080, in 0.011s
[40/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00044, val loss:
0.00080, in 0.009s
[41/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00044, val loss:
0.00079, in 0.013s
[42/50000] 1 tree, 31 leaves, max depth = 14, train loss: 0.00043, val loss:
0.00079, in 0.010s
[43/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00043, val loss:
0.00079, in 0.011s
[44/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00042, val loss:
0.00078, in 0.013s
[45/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00042, val loss:
```

```
0.00078, in 0.010s
[46/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00041, val loss:
0.00078, in 0.013s
[47/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00040, val loss:
0.00077, in 0.015s
[48/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00040, val loss:
0.00077, in 0.010s
[49/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00039, val loss:
0.00077, in 0.010s
[50/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00039, val loss:
0.00077, in 0.014s
[51/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00038, val loss:
0.00077, in 0.011s
[52/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00038, val loss:
0.00077, in 0.013s
[53/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00038, val loss:
0.00077, in 0.012s
[54/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00037, val loss:
0.00077, in 0.009s
[55/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00037, val loss:
0.00076, in 0.011s
[56/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00036, val loss:
0.00076, in 0.014s
[57/50000] 1 tree, 31 leaves, max depth = 15, train loss: 0.00036, val loss:
0.00076, in 0.012s
[58/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00036, val loss:
0.00075, in 0.010s
[59/50000] 1 tree, 31 leaves, max depth = 16, train loss: 0.00035, val loss:
0.00075, in 0.010s
[60/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00035, val loss:
0.00076, in 0.014s
[61/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00035, val loss:
0.00075, in 0.011s
[62/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00034, val loss:
0.00075, in 0.010s
[63/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00034, val loss:
0.00075, in 0.009s
[64/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00034, val loss:
0.00075, in 0.018s
[65/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00033, val loss:
0.00075, in 0.009s
[66/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00033, val loss:
0.00075, in 0.010s
[67/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00033, val loss:
0.00074, in 0.014s
[68/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00033, val loss:
0.00074, in 0.011s
[69/50000] 1 tree, 31 leaves, max depth = 15, train loss: 0.00032, val loss:
```

```
0.00074, in 0.012s
[70/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00032, val loss:
0.00074, in 0.011s
[71/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00032, val loss:
0.00074, in 0.010s
[72/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00031, val loss:
0.00074, in 0.012s
[73/50000] 1 tree, 31 leaves, max depth = 15, train loss: 0.00031, val loss:
0.00074, in 0.020s
[74/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00031, val loss:
0.00074, in 0.010s
[75/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00031, val loss:
0.00074, in 0.016s
[76/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00030, val loss:
0.00074, in 0.013s
[77/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00030, val loss:
0.00073, in 0.010s
[78/50000] 1 tree, 31 leaves, max depth = 14, train loss: 0.00030, val loss:
0.00073, in 0.008s
[79/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00030, val loss:
0.00073, in 0.012s
[80/50000] 1 tree, 31 leaves, max depth = 14, train loss: 0.00029, val loss:
0.00073, in 0.009s
[81/50000] 1 tree, 31 leaves, max depth = 14, train loss: 0.00029, val loss:
0.00073, in 0.016s
[82/50000] 1 tree, 31 leaves, max depth = 16, train loss: 0.00029, val loss:
0.00073, in 0.018s
[83/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00029, val loss:
0.00073, in 0.011s
[84/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00029, val loss:
0.00073, in 0.010s
[85/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00028, val loss:
0.00073, in 0.009s
[86/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00028, val loss:
0.00073, in 0.008s
[87/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00028, val loss:
0.00073, in 0.008s
[88/50000] 1 tree, 31 leaves, max depth = 14, train loss: 0.00028, val loss:
0.00073, in 0.013s
[89/50000] 1 tree, 31 leaves, max depth = 9, train loss: 0.00027, val loss:
0.00073, in 0.010s
[90/50000] 1 tree, 31 leaves, max depth = 10, train loss: 0.00027, val loss:
0.00073, in 0.010s
[91/50000] 1 tree, 31 leaves, max depth = 12, train loss: 0.00027, val loss:
0.00073, in 0.009s
[92/50000] 1 tree, 31 leaves, max depth = 8, train loss: 0.00027, val loss:
0.00073, in 0.011s
[93/50000] 1 tree, 31 leaves, max depth = 13, train loss: 0.00027, val loss:
```

```
0.00073, in 0.010s
[94/50000] 1 tree, 31 leaves, max depth = 15, train loss: 0.00026, val loss:
0.00073, in 0.012s
[95/50000] 1 tree, 31 leaves, max depth = 11, train loss: 0.00026, val loss:
0.00073, in 0.012s
Fit 95 trees in 1.279 s, (2945 total leaves)
Time spent computing histograms: 0.202s
Time spent finding best splits:  0.117s
Time spent applying splits:      0.121s
Time spent predicting:           0.009s
Estimadores para Train
Metr.    Valor                   Descripción
R^2      0.9101464221217427      (Coeficiente de Determinación)
RMSE     0.0006182259662542193   (Raíz de error cuadrático medio)
MAE      0.01879705055225904     (Error absoluto medio)
MAX      0.18431448005700957     (Error Máximo)

Estimadores para Test
Metr.    Valor                   Descripción
R^2      0.842183850570004       (Coeficiente de Determinación)
RMSE     0.001130855140274013    (Raíz de error cuadrático medio)
MAE      0.025811140466109116    (Error absoluto medio)
MAX      0.16397140370451635     (Error Máximo)
```



Como se aprecia de todas las iteraciones permitidas ha parado tras 95 por no estar ya mejorando el error en validación, que se había estancado en 0.00073, a pesar que el error en train ya había bajado a 0.00026.

Vemos de lanzar la búsqueda de los mejores hiperparametros usando este ñultimo modo con una búsqueda aleatoria de hiperparémtros, y mostrando la gráfica de entrenamiento.

[11]:
```
from sklearn.model_selection import RandomizedSearchCV
random_grid = {'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.2, 0.25, 0.3, 0.5,␣
 ↪1],
               'max_iter': [200, 500, 1000, 5000],
               'max_leaf_nodes': [5, 10, 20, 30, 50, 100, 200],
               'early_stopping': [True],
               'validation_fraction': [0.1, 0.15, 0.2],
```

```
                'n_iter_no_change': [10, 15],
                'tol': [1e-07]
                }
modelo_random = RandomizedSearchCV(scoring="neg_mean_squared_error", estimator␣
 ↪= modelo2,
                                   param_distributions = random_grid, n_iter = 200,
                                   cv = 3, verbose=1, random_state=123, n_jobs = -1)
modelo_random.fit(X_train_s, y_train)

## Mostramos los resultados
print("Estimadores para Train")
mostrar_metricas(y_train, modelo_random.best_estimator_.predict(X_train_s))
print("")
print("Estimadores para Test")
mostrar_metricas(y_test, modelo_random.best_estimator_.predict(X_test_s))

## Mostramos la gráfica del entrenamiento
def mostrar_entrenamiento(modelo):
    # Muestro gráfica de accuracy y losses
    plt.style.use("ggplot")
    fig,ax = plt.subplots()
    filas = len(modelo.validation_score_)
    ax.plot(np.arange(0, filas), modelo.train_score_, label="train_loss",␣
 ↪color="red")
    ax.plot(np.arange(0, filas), modelo.validation_score_, label="val_loss",␣
 ↪color="orange")
    # Pongo en un eje secundario la Precisión, porque la escala no está en los␣
 ↪mismos rangos
    #ax2=ax.twinx()
    #ax2.plot(np.arange(0, filas), H.history["accuracy"], label="train_acc",␣
 ↪color="blue")
    #ax2.plot(np.arange(0, filas), H.history["val_accuracy"], label="val_acc",␣
 ↪color="green")
    ax.set_title("Training/Validation Loss and Accuracy")
    ax.set_xlabel("Epoch #")
    ax.set_ylabel("Loss", color="red")
    #ax.set_ylim(0., np.max(H.history["loss"]))
    #ax2.set_ylabel("Accuracy", color="blue")
    #ax2.set_ylim(0., 1.)
    ax.legend(loc="upper right")
    #ax2.legend(loc="lower right")
mostrar_entrenamiento(modelo_random.best_estimator_)
```

```
Fitting 3 folds for each of 200 candidates, totalling 600 fits
Binning 0.001 GB of training data: 0.015 s
Binning 0.000 GB of validation data: 0.001 s
Fitting gradient boosted rounds:
```

```
[1/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00295, val loss:
0.00301, in 0.008s
[2/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00256, val loss:
0.00265, in 0.009s
[3/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00224, val loss:
0.00237, in 0.010s
[4/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00197, val loss:
0.00212, in 0.009s
[5/200] 1 tree, 20 leaves, max depth = 6, train loss: 0.00176, val loss:
0.00193, in 0.009s
[6/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00158, val loss:
0.00177, in 0.008s
[7/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00144, val loss:
0.00164, in 0.008s
[8/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00131, val loss:
0.00153, in 0.008s
[9/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00121, val loss:
0.00144, in 0.008s
[10/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00113, val loss:
0.00137, in 0.010s
[11/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00105, val loss:
0.00130, in 0.009s
[12/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00100, val loss:
0.00125, in 0.009s
[13/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00094, val loss:
0.00120, in 0.008s
[14/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00090, val loss:
0.00117, in 0.008s
[15/200] 1 tree, 20 leaves, max depth = 6, train loss: 0.00086, val loss:
0.00113, in 0.008s
[16/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00083, val loss:
0.00110, in 0.009s
[17/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00080, val loss:
0.00107, in 0.008s
[18/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00077, val loss:
0.00105, in 0.008s
[19/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00075, val loss:
0.00104, in 0.008s
[20/200] 1 tree, 20 leaves, max depth = 6, train loss: 0.00073, val loss:
0.00102, in 0.008s
[21/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00071, val loss:
0.00100, in 0.008s
[22/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00069, val loss:
0.00099, in 0.011s
[23/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00068, val loss:
0.00098, in 0.008s
[24/200] 1 tree, 20 leaves, max depth = 6, train loss: 0.00066, val loss:
0.00096, in 0.010s
```

```
[25/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00065, val loss:
0.00095, in 0.008s
[26/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00064, val loss:
0.00094, in 0.010s
[27/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00063, val loss:
0.00093, in 0.007s
[28/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00062, val loss:
0.00092, in 0.007s
[29/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00061, val loss:
0.00091, in 0.007s
[30/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00060, val loss:
0.00090, in 0.007s
[31/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00059, val loss:
0.00089, in 0.011s
[32/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00058, val loss:
0.00089, in 0.009s
[33/200] 1 tree, 20 leaves, max depth = 6, train loss: 0.00057, val loss:
0.00088, in 0.009s
[34/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00056, val loss:
0.00088, in 0.011s
[35/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00056, val loss:
0.00087, in 0.009s
[36/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00055, val loss:
0.00087, in 0.009s
[37/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00054, val loss:
0.00086, in 0.012s
[38/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00054, val loss:
0.00085, in 0.008s
[39/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00053, val loss:
0.00085, in 0.008s
[40/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00052, val loss:
0.00085, in 0.007s
[41/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00052, val loss:
0.00084, in 0.011s
[42/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00051, val loss:
0.00084, in 0.008s
[43/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00051, val loss:
0.00084, in 0.008s
[44/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00050, val loss:
0.00083, in 0.007s
[45/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00049, val loss:
0.00083, in 0.011s
[46/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00049, val loss:
0.00082, in 0.008s
[47/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00048, val loss:
0.00082, in 0.009s
[48/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00048, val loss:
0.00082, in 0.009s
```

```
[49/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00047, val loss:
0.00081, in 0.010s
[50/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00047, val loss:
0.00081, in 0.009s
[51/200] 1 tree, 20 leaves, max depth = 12, train loss: 0.00047, val loss:
0.00081, in 0.010s
[52/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00046, val loss:
0.00080, in 0.008s
[53/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00045, val loss:
0.00080, in 0.007s
[54/200] 1 tree, 20 leaves, max depth = 12, train loss: 0.00045, val loss:
0.00080, in 0.008s
[55/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00045, val loss:
0.00080, in 0.008s
[56/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00044, val loss:
0.00080, in 0.008s
[57/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00044, val loss:
0.00080, in 0.007s
[58/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00043, val loss:
0.00079, in 0.007s
[59/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00043, val loss:
0.00079, in 0.007s
[60/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00043, val loss:
0.00079, in 0.010s
[61/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00042, val loss:
0.00079, in 0.011s
[62/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00042, val loss:
0.00079, in 0.007s
[63/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00042, val loss:
0.00079, in 0.008s
[64/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00041, val loss:
0.00079, in 0.011s
[65/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00041, val loss:
0.00079, in 0.008s
[66/200] 1 tree, 20 leaves, max depth = 12, train loss: 0.00041, val loss:
0.00078, in 0.008s
[67/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00040, val loss:
0.00078, in 0.008s
[68/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00040, val loss:
0.00078, in 0.007s
[69/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00040, val loss:
0.00078, in 0.008s
[70/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00040, val loss:
0.00078, in 0.008s
[71/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00039, val loss:
0.00078, in 0.011s
[72/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00039, val loss:
0.00077, in 0.008s
```

```
[73/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00039, val loss:
0.00077, in 0.008s
[74/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00039, val loss:
0.00077, in 0.007s
[75/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00038, val loss:
0.00077, in 0.012s
[76/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00038, val loss:
0.00077, in 0.008s
[77/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00038, val loss:
0.00077, in 0.009s
[78/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00038, val loss:
0.00077, in 0.009s
[79/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00037, val loss:
0.00077, in 0.008s
[80/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00037, val loss:
0.00077, in 0.008s
[81/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00037, val loss:
0.00077, in 0.008s
[82/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00037, val loss:
0.00077, in 0.008s
[83/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00037, val loss:
0.00077, in 0.007s
[84/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00036, val loss:
0.00077, in 0.011s
[85/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00036, val loss:
0.00077, in 0.009s
[86/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00036, val loss:
0.00076, in 0.007s
[87/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00036, val loss:
0.00076, in 0.008s
[88/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00035, val loss:
0.00076, in 0.012s
[89/200] 1 tree, 20 leaves, max depth = 14, train loss: 0.00035, val loss:
0.00076, in 0.008s
[90/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00035, val loss:
0.00075, in 0.008s
[91/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00035, val loss:
0.00075, in 0.008s
[92/200] 1 tree, 20 leaves, max depth = 6, train loss: 0.00035, val loss:
0.00075, in 0.009s
[93/200] 1 tree, 20 leaves, max depth = 12, train loss: 0.00034, val loss:
0.00076, in 0.008s
[94/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00034, val loss:
0.00075, in 0.008s
[95/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00034, val loss:
0.00075, in 0.009s
[96/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00034, val loss:
0.00075, in 0.008s
```

```
[97/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00034, val loss:
0.00075, in 0.008s
[98/200] 1 tree, 20 leaves, max depth = 7, train loss: 0.00033, val loss:
0.00075, in 0.010s
[99/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00033, val loss:
0.00075, in 0.007s
[100/200] 1 tree, 20 leaves, max depth = 12, train loss: 0.00033, val loss:
0.00075, in 0.021s
[101/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00033, val loss:
0.00075, in 0.009s
[102/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00033, val loss:
0.00075, in 0.012s
[103/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00033, val loss:
0.00075, in 0.011s
[104/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00032, val loss:
0.00075, in 0.014s
[105/200] 1 tree, 20 leaves, max depth = 12, train loss: 0.00032, val loss:
0.00075, in 0.008s
[106/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00032, val loss:
0.00075, in 0.007s
[107/200] 1 tree, 20 leaves, max depth = 12, train loss: 0.00032, val loss:
0.00075, in 0.006s
[108/200] 1 tree, 20 leaves, max depth = 9, train loss: 0.00032, val loss:
0.00075, in 0.010s
[109/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00031, val loss:
0.00075, in 0.009s
[110/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00031, val loss:
0.00075, in 0.009s
[111/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00031, val loss:
0.00075, in 0.007s
[112/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00031, val loss:
0.00075, in 0.011s
[113/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00031, val loss:
0.00075, in 0.009s
[114/200] 1 tree, 20 leaves, max depth = 11, train loss: 0.00031, val loss:
0.00075, in 0.008s
[115/200] 1 tree, 20 leaves, max depth = 10, train loss: 0.00031, val loss:
0.00075, in 0.008s
[116/200] 1 tree, 20 leaves, max depth = 8, train loss: 0.00031, val loss:
0.00075, in 0.007s
Fit 116 trees in 1.206 s, (2320 total leaves)
Time spent computing histograms: 0.126s
Time spent finding best splits:  0.110s
Time spent applying splits:      0.088s
Time spent predicting:           0.010s
Estimadores para Train
Metr.    Valor                   Descripción
R^2      0.8920809102796025      (Coeficiente de Determinación)
```

```
RMSE      0.0007425233930035075    (Raíz de error cuadrático medio)
MAE       0.020471776688240143     (Error absoluto medio)
MAX       0.22465169947440788      (Error Máximo)

Estimadores para Test
Metr.     Valor                    Descripción
R^2       0.8378583629080998       (Coeficiente de Determinación)
RMSE      0.00116185006680291      (Raíz de error cuadrático medio)
MAE       0.025875958889140085     (Error absoluto medio)
MAX       0.18115629775007847      (Error Máximo)
```

[12]:
```
modelo_random.best_params_
```

[12]:
```
{'validation_fraction': 0.15,
 'tol': 1e-07,
 'n_iter_no_change': 10,
 'max_leaf_nodes': 20,
 'max_iter': 200,
 'learning_rate': 0.1,
 'early_stopping': True}
```