Boosting

Se hará un estudio utilizando el modelo de AdaBoost para regresión introducido por Freund y Schapire en 1995.
TODO Añadir la bibliografía Y. Freund, R. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.

Para ello se utilizará la función

```
class sklearn.ensemble.AdaBoostRegressor(base_estimator=None,
*, n_estimators=50, learning_rate=1.0,
loss='linear', random_state=None)
```

Bibliografía:

- Teoría: https://scikit-learn.org/stable/modules/ensemble.html#adaboost
- Implementación: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegresso

Los hiperparámetros que ajustaremos y utilizaremos de esta función son los siguientes.

- `base_estimator` objeto con el estimador base que utilizar, utilizares `DecisionTreeRegressor` inicialos con profundidad máxima de tres. (TODO, justificar, en el guión se nos indica que utilicemos estos.
- `learning_rate` haremos un estudio de cómo varía en función del larning rate.
- `loss` La función de pérdida para actualizar los pesos del boosting en cada iteración, será la linear. No tenemos ningún motivo para preferir uno frente a otro.
- `n_estimators` número de estimadores para el cual el boosting termina, en caso de encontrarse un ajuste perfector pararía antes.

Estudio preliminar

A partir de los datos normalizados y sin outliers probaremos una serie de parámeteros para comprobar cuáles dan los mejores resultados.

Realizaremos estimaciones con el número de estimadores y la tasa de aprendizaje, obteniendo los siguientes resultados:

Mejores parámetros: {'learning_rate': 0.1, 'n_estimators': 50} Con una R^2 de: 0.5862740782719961

Tabla 1: Estudio preliminar de hiperparámetros semilla fijada a dos.

Parámetros	R^2 medio	Des típ R^2	Ranking	t medio ajuste
n_estimators 50 learning_rate 0.1	0.5863	0.0202	1	1.1834
n_estimators 80 learning_rate 0.1	0.5861	0.0244	2	1.8518
n_estimators 60 learning_rate 0.1	0.5855	0.0236	3	1.3973
n_estimators 100 learning_rate 0.1	0.5833	0.0270	4	2.2807
n_estimators 100 learning_rate 0.01	0.5815	0.0079	5	2.3842
n_estimators 80 learning_rate 0.01	0.5813	0.0103	6	1.9088
n_estimators 50 learning_rate 0.01	0.5810	0.0113	7	1.1970
n_estimators 60 learning_rate 0.01	0.5804	0.0115	8	1.4396
n_estimators 50 learning_rate 0.001	0.5784	0.0108	9	1.2093
n_estimators 100 learning_rate 0.001	0.5777	0.0106	10	2.5904
n_estimators 80 learning_rate 0.001	0.5771	0.0110	11	1.9342
n_estimators 60 learning_rate 0.001	0.5762	0.0108	12	1.5566
n_estimators 50 learning_rate 1	0.4983	0.0475	13	1.0179
n_estimators 60 learning_rate 1	0.4941	0.0415	14	1.1876
n_estimators 80 learning_rate 1	0.4857	0.0444	15	1.5530
n_estimators 100 learning_rate 1	0.4756	0.0497	16	1.7192

De esto valores observamos que una tasa de aprendizaje de entre 0.01 y 0.1 parece ser la mejor opción independeientemente de número de estimadores.

Analicemos cómo varía entonces si cambiamos el conjunto de entrenamiento:

Tabla 2: Comparativa de mejores resultados en validación cruzada según el preprocesado de los datos

Datos de entrenamiento	Mejor error	Con parámetros
Sin normalizar con outliers	0.6223	{'learning_rate': 0.1, 'n_estimators': 50}
Normalizado con outliers	0.6199	{'learning_rate': 0.1, 'n_estimators': 50}
Sin normalizar sin outliers	0.5831	{'learning_rate': 0.1, 'n_estimators': 50}
Normalizado con outliers	0.5863	{'learning_rate': 0.1, 'n_estimators': 50}

Además analizando las distintas pruebas vemos que de manera general se mantiene esa mejora (ver sucesivas tablas de validación cruzada).

Tabla 3: Validación cruzada para data set sin normalizar con outliers

Parámetros	R^2 medio	Desviación tipica R^2	Ranking	tiempo medio ajuste
n_estimators 50 learning_rate 0.1	0.6223	0.0123	1	1.2234
n_estimators 100 learning_rate 0.1	0.6191	0.0110	2	2.3279
n_estimators 100 learning_rate 0.01	0.6167	0.0234	3	2.5144
n_estimators 50 learning_rate 0.01	0.6145	0.0251	4	1.2346

Tabla 4: Validación cruzada para data set normalizado con outliers

Parámetros	R^2 medio	Desviación tipica R^2	Ranking	tiempo medio ajuste
n_estimators 50 learning_rate 0.1	0.5863	0.0202	1	1.1715
n_estimators 100 learning_rate 0.1	0.5833	0.0270	2	2.1543
n_estimators 100 learning_rate 0.01	0.5815	0.0079	3	2.3857
n_estimators 50 learning_rate 0.01	0.5810	0.0113	4	1.2034

Tabla 5: Validación cruzada para data set sin normalizar sin outliers

Parámetros	R^2 medio	Desviación tipica R^2	Ranking	tiempo medio ajuste
n_estimators 50 learning_rate 0.1	0.5831	0.0204	1	1.1916
n_estimators 100 learning_rate 0.01	0.5831	0.0063	2	2.3928
n_estimators 50 learning_rate 0.01	0.5818	0.0079	3	1.2035
n_estimators 100 learning_rate 0.1	0.5778	0.0282	4	2.1643

Tabla 6: Normalizado con outliers

Parámetros	R^2 medio	Desviación tipica R^2	Ranking	tiempo medio
n_estimators 50 learning_rate 0.1	0.5829	0.0169	1.0000	1.5142
n_estimators 100 learning_rate 0.01	0.5824	0.0087	2.0000	3.0499
n_estimators 50 learning_rate 0.01	0.5811	0.0107	3.0000	1.5124
n_estimators 100 learning_rate 0.1	0.5801	0.0246	4.0000	2.7327

El hecho de que los mejores sean sin normalizar con outliers y normalizado con outliers, nos hacen plantearnos dos situaciones:
1. Se está produciendo sobreajuste.
2. El criterio de eliminación fue demasiado estricto e incluso habría que plantearse aumento de la dimensión.

Para comprobar estas hipótesis plantearemos los siguientes experimentos:

1. Estudio de la diferencia entre E_{in} y E_{val}

2. Aumentaremos la dimensión para ver si conseguimos mejor explicación.

1. Estudio de la difrencia entre E_{in} y E_{val} variando el número de estimadores.

Para formular este experimento se han reservado un 15 % de datos del conjunto de entrenamiento como evaluación. Hemos considerado este porcertaje frente al 20 % para tener más datos de entrenamiento.

Como podemos observar se produce un ligero sobreajuste.

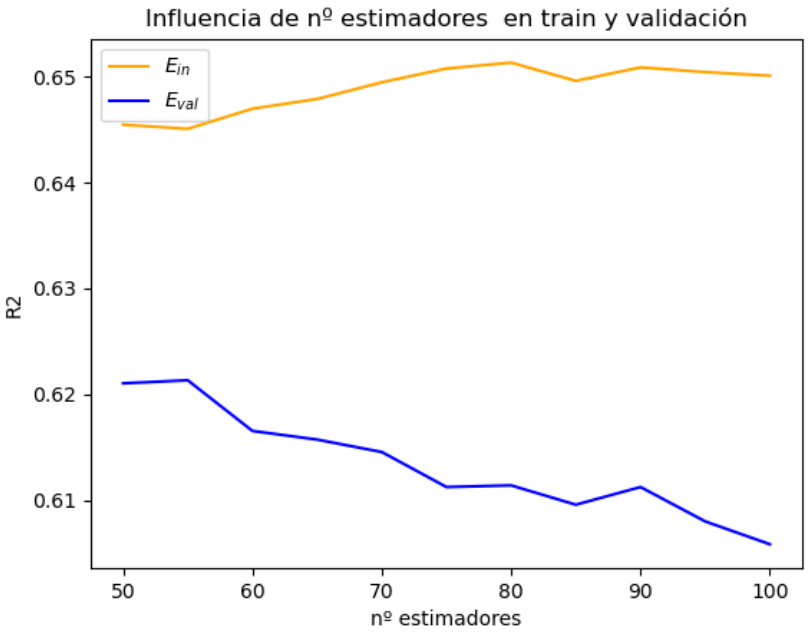

Figura 1: Comparativas E_{in} y E_{eval} en función del número de estimadores, con datos de entrenamiento normalizados sin outliers.

.

Tabla 7: Comparativas E_{in} y E_{eval} en función del número de estimadores, con datos de entrenamiento normalizados sin outliers.

Nº estimadores	E_{in}	E_{eval}
50	0.6455	0.6210
55	0.6451	0.6213

Nº estimadores	E_{in}	E_{eval}
60	0.6470	0.6165
65	0.6479	0.6157
70	0.6495	0.6145
75	0.6508	0.6112
80	0.6514	0.6114
85	0.6496	0.6096
90	0.6509	0.6112
95	0.6505	0.6080
100	0.6501	0.6058

2. Aumento de la dimensión por transformaciones lineales

Hemos obtenido incluso peores resultados aumentado la dimensión por transformaciones cúbicas.

La cuadrática la mejora ligeramente.

Tabla 8: Comparativas aplicando transformaciones polinómicas

Transformación	Mejor R^2 medio	Hiperparámetros	Tiempo
Sin transformación	0.6186	n_estimators 50 learning_rate 0.1	1.2370
Polinomio grado 2	0.6215	n_estimators 50 learning_rate 0.1	2.3817
Polinomio grado 3	0.6178	n_estimators 50 learning_rate 0.1	3.7210

Hiperparámetros finales determinados

- Número de estimadores: 50
- Tasa de aprendizaje: 0.1
- Datos de entrenamiento con outliers y una transformación cuadrática.

¿Temos sufientes datos en este modelo?

Para el modelos

Vemos como era de esperar que al aumentar el número de datos de entrenamiento:
- El coeficiente de determinación dentro del entrenamiento disminuye. - El coeficiente de determinación dentro de evaluación aumenta (el modelo mejora).

Y de seguirse este comportamiento asintótico, con más datos pordríamos mejorar el modelo pero con un umbral de 0,7 de coeficiente de determinación.

.

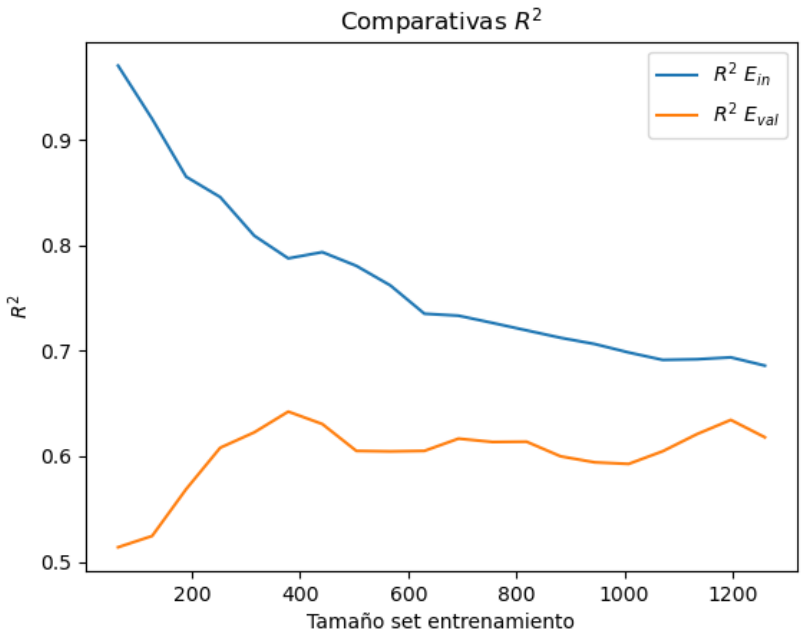

Figura 2: Comparativas E_{in} y E_{eval} en función del número de datos de entrenamiento, tabla conjunta