Random Forests para regresión

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
```

In [2]:
```
data = pd.read_csv("../../Data-Sets/datasets/boston/Boston.csv")
data.head()
```

Out[2]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [3]:
```
colnames = data.columns.values.tolist()
predictors = colnames[:13]
target = colnames[13]
X = data[predictors]
Y = data[target]
```

In [27]:
```
forest = RandomForestRegressor(n_jobs=4, oob_score=True, n_estimators=10000)
forest.fit(X,Y)
```

Out[27]:
```
RandomForestRegressor(n_estimators=10000, n_jobs=4, oob_score=True)
```

Las variables significan:

n_jobs: Cantidad de procesos en paralelo que se ejecutarán por el ordenador.

oob_score: Variable binaria, en true significa que el modelo va a ser un modelo aleatorio para hacer las predicciones. oob significa (out of the box).

n_estimators: Cantidad de árboles que nuestro modelo va a tener. Pero se puede incrementar muchísimo.

In [28]:
```
data["rforest_pred"]=forest.oob_prediction_
data[["rforest_pred", "medv"]]
```

Out[28]:

	rforest_pred	medv
0	28.451683	24.0
1	22.575502	21.6
2	34.424230	34.7
3	34.781743	33.4
4	34.329139	36.2
...	...	...
501	24.343706	22.4

	rforest_pred	medv
502	19.031917	20.6
503	27.889505	23.9
504	26.208492	22.0
505	20.881828	11.9

506 rows × 2 columns

Como no se puede hacer validación cruzada, lo que se hace es calcular el error para cada uno de los valores, recordad que se eleva al cuadrado para potenciar los errores y que queden todos positivos, luego, se los divide por el total, calculando así el promedio.

In [29]:
```
data["rforest_error2"] = (data["rforest_pred"]-data["medv"])**2
sum(data["rforest_error2"])/len(data)
```

Out[29]:
```
10.264235643817102
```

Como vemos ha disminuído el error de 21 a 10, lo cual es muy bueno.

In [30]:
```
forest.oob_score_
```

Out[30]:
```
0.8784140060529393
```

Llega un punto que por más árboles que agreguemos, no vamos a mejorar el error, puesto que llegamos a un límite del modelo.

In [ ]:
```

```