Introduction

Karar ağaçları sizi zor bir kararla baş başa bırakır. Çok sayıda yaprağı olan derin bir ağaç **aşırı uyum** gösterir, çünkü her tahmin sadece yaprağındaki birkaç evden gelen geçmiş verilere dayanır. Ancak az yapraklı sığ bir ağaç, ham verilerdeki birçok ayrımı yakalayamadığı için kötü performans gösterir.

Günümüzün en gelişmiş modelleme teknikleri bile aşırı uyum ve eksik uyum arasındaki bu gerilimle karşı karşıyadır. Ancak, birçok model daha iyi performansa yol açabilecek akıllı fikirlere sahiptir. Bir örnek olarak **rastgele ormanı (random forest)** inceleyeceğiz.

Rastgele orman, birçok ağaç kullanır ve her bir bileşen ağacının tahminlerinin ortalamasını alarak bir tahmin yapar. Genellikle tek bir karar ağacına göre çok daha iyi bir tahmini doğruluğa sahiptir ve varsayılan parametrelerle iyi çalışır. Modellemeye devam ederseniz, daha da iyi performansa sahip daha fazla model öğrenebilirsiniz, ancak bunların birçoğu doğru parametreleri ayarlama konusunda hassastır.

# Example

Verileri yüklemek için kodu birkaç kez gördünüz. Veri yükleme işleminin sonunda aşağıdaki değişkenlere sahip olacağız:

* train\_X
* val\_X
* train\_y
* val\_y

import pandas as pd

*# Load data*

melbourne\_file\_path = '../input/melbourne-housing-snapshot/melb\_data.csv'

melbourne\_data = pd.read\_csv(melbourne\_file\_path)

*# Filter rows with missing values*

melbourne\_data = melbourne\_data.dropna(axis=0)

*# Choose target and features*

y = melbourne\_data.Price

melbourne\_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea',

'YearBuilt', 'Lattitude', 'Longtitude']

X = melbourne\_data[melbourne\_features]

from sklearn.model\_selection import train\_test\_split

*# split data into training and validation data, for both features and target*

*# The split is based on a random number generator. Supplying a numeric value to*

*# the random\_state argument guarantees we get the same split every time we*

*# run this script.*

train\_X, val\_X, train\_y, val\_y = train\_test\_split(X, y,random\_state = 0)

Scikit-learn'da rastgele orman modelini, karar ağacı modelini oluşturduğumuza benzer bir şekilde inşa ediyoruz; bu sefer **DecisionTreeRegressor** sınıfı yerine **RandomForestRegressor** sınıfını kullanıyoruz.

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error

forest\_model = RandomForestRegressor(random\_state=1)

forest\_model.fit(train\_X, train\_y)

melb\_preds = forest\_model.predict(val\_X)

print(mean\_absolute\_error(val\_y, melb\_preds))

191669.7536453626

# Conclusion

Büyük ihtimalle daha da iyileştirme için yer vardır, ancak bu, 250.000 olan en iyi karar ağacı hatasına göre büyük bir gelişmedir. Tıpkı tek bir karar ağacının maksimum derinliğini değiştirdiğimiz gibi, Rastgele Orman modelinin performansını değiştirebileceğiniz parametreler vardır. Ancak **Rastgele Orman** modellerinin en iyi özelliklerinden biri, bu ayarlamaları yapmadan bile genellikle makul bir şekilde iyi çalışmasıdır.