Bir model oluşturdunuz, peki ne kadar iyi? 🤔

Bu derste, modelinizin kalitesini ölçmek için **model doğrulamasını (model validation)** kullanmayı öğreneceksiniz. Model kalitesini ölçmek, modellerinizi yinelemeli olarak iyileştirmenin anahtarıdır.

What is Model Validation

Modelinizi neredeyse her zaman değerlendirmeniz gerekecektir. Çoğu (hepsi olmasa da) uygulamada, model kalitesinin ilgili ölçüsü **tahmin doğruluğudur**. Diğer bir deyişle, modelin tahminleri gerçekte olanlara yakın olacak mı?

Birçok kişi tahmin doğruluğunu ölçerken büyük bir hata yapar. **Eğitim verileriyle** tahminler yapar ve bu tahminleri eğitim verilerindeki hedef değerlerle karşılaştırır. Bu yaklaşımla ilgili sorunu ve bunu nasıl çözeceğinizi birazdan göreceksiniz, ancak önce bunu nasıl yapacağımızı düşünelim.

İlk olarak, model kalitesini anlaşılır bir şekilde özetlemeniz gerekir. 10.000 ev için tahmin edilen ve gerçek ev değerlerini karşılaştırırsanız, muhtemelen iyi ve kötü tahminlerin bir karışımını bulacaksınız. 10.000 tahmin edilen ve gerçek değerden oluşan bir listeye bakmak anlamsız olacaktır. Bunu tek bir ölçüde özetlememiz gerekiyor.

Model kalitesini özetlemek için birçok metrik vardır, ancak **Ortalama Mutlak Hata (Mean Absolute Error - MAE)** adı verilen bir metrikle başlayacağız. Bu metriği son kelimesinden, yani "hata"dan başlayarak inceleyelim.

Her ev için tahmin hatası:

error=actual−predicted

Eğer bir evin maliyeti **150.000 dolar** iken, siz **100.000 dolar** tahmin ettiyseniz, hata **50.000 dolar** olur.

**Ortalama Mutlak Hata (MAE)** metriğinde, her bir hatanın mutlak değerini alırız. Bu, her hatayı pozitif bir sayıya dönüştürür. Daha sonra bu mutlak hataların ortalamasını alırız. Bu, model kalitemizin ölçüsüdür. Basit bir ifadeyle şöyle denilebilir:

"Ortalama olarak, tahminlerimiz yaklaşık **X** kadar sapmaktadır."

MAE'yi hesaplamak için öncelikle bir modele ihtiyacımız var. Bu, aşağıdaki gizli bir hücrede oluşturulmuştur; "kod" düğmesine tıklayarak inceleyebilirsiniz.

*# Data Loading Code Hidden Here*

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 price values*

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

*# Choose target and features*

y = filtered\_melbourne\_data.Price

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

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

X = filtered\_melbourne\_data[melbourne\_features]

from sklearn.tree import DecisionTreeRegressor

*# Define model*

melbourne\_model = DecisionTreeRegressor()

*# Fit model*

melbourne\_model.fit(X, y)

DecisionTreeRegressor()

Bir modelimiz olduğunda, ortalama mutlak hatayı şu şekilde hesaplarız:

from sklearn.metrics import mean\_absolute\_error

predicted\_home\_prices = melbourne\_model.predict(X)

mean\_absolute\_error(y, predicted\_home\_prices)

434.71594577146544

# The Problem with "In-Sample" Scores[¶](https://www.kaggle.com/code/dansbecker/model-validation" \l "The-Problem-with-%22In-Sample%22-Scores" \t "_self)

Hesapladığımız bu ölçüye **"örnek içi" (in-sample) skor** diyebiliriz. Hem modeli oluşturmak hem de değerlendirmek için tek bir ev "örneği" kullandık. Neden kötü olduğuna bakalım:

### Neden "Örnek İçi" Skor Kötüdür? 👎

Şunu hayal edin: Büyük bir emlak piyasasında kapı renginin ev fiyatıyla hiçbir ilgisi yok. Ancak, modeli oluşturmak için kullandığınız veri örneğinde, yeşil kapılı tüm evler çok pahalıydı. Modelin görevi ev fiyatlarını tahmin eden kalıpları bulmak olduğundan, bu kalıbı görecek ve yeşil kapılı evler için her zaman yüksek fiyatlar tahmin edecektir.

Bu kalıp **eğitim verilerinden** türetildiği için, model eğitim verilerinde doğru görünecektir.

Ancak bu kalıp, model yeni veriler gördüğünde geçerli olmazsa, model pratikte kullanıldığında çok yanlış olacaktır.

### Çözüm: Doğrulama Verileri (Validation Data) 🚀

Modellerin pratik değeri, yeni veriler üzerinde tahminler yapmaktan geldiği için, performansı **modeli oluşturmak için kullanılmayan veriler üzerinde** ölçeriz. Bunu yapmanın en basit yolu, model oluşturma sürecinden bazı verileri hariç tutmak ve daha sonra bu verileri, modelin daha önce görmediği verilerdeki doğruluğunu test etmek için kullanmaktır. Bu verilere **doğrulama verileri** denir.

# Coding It

Scikit-learn kütüphanesinde, veriyi iki parçaya ayırmak için **train\_test\_split** adında bir fonksiyon bulunur. Bu verilerin bir kısmını modeli eğiticerken **eğitim verisi** olarak kullanacağız, diğer kısmını ise **doğrulama verisi** olarak **ortalama mutlak hatayı (mean\_absolute\_error)** hesaplamak için kullanacağız.

## İşte kod:

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)

*# Define model*

melbourne\_model = DecisionTreeRegressor()

*# Fit model*

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

*# get predicted prices on validation data*

val\_predictions = melbourne\_model.predict(val\_X)

print(mean\_absolute\_error(val\_y, val\_predictions))

265806.91478373145

# Wow

İç örnek (in-sample) verileriniz için ortalama mutlak hatanız yaklaşık **500 dolardı**. Örnek dışı (out-of-sample) verilerde ise bu rakam **250.000 dolardan fazlaydı**.

Bu, neredeyse tamamen doğru olan bir model ile çoğu pratik amaç için kullanılamaz olan bir model arasındaki farktır. Referans olarak, doğrulama verilerindeki ortalama ev değeri **1.1 milyon dolardır**. Yani yeni verilerdeki hata, ortalama ev değerinin yaklaşık dörtte biridir.

## Bu modeli geliştirmenin birçok yolu vardır; örneğin daha iyi özellikler veya farklı model türleri bulmak için denemeler yapmak gibi.