Bu eğitimde, model performansının daha iyi ölçülmesi için çapraz doğrulamanın nasıl kullanılacağını öğreneceksiniz.

Introduction

Makine öğrenmesi yinelemeli bir süreçtir.

Hangi tahmin değişkenlerini kullanacağınız, hangi model türlerini seçeceğiniz, bu modellere hangi argümanları vereceğiniz gibi konularda kararlar alırsınız. Şimdiye kadar bu kararları, model kalitesini bir doğrulama (validation) veya tutma (holdout) seti ile ölçerek, verilere dayalı bir şekilde verdiniz.

Ancak bu yaklaşımın bazı dezavantajları vardır. Bunu görmek için 5000 satırlık bir veri setiniz olduğunu düşünün. Genellikle verilerin yaklaşık %20'sini, yani 1000 satırını doğrulama seti olarak ayırırsınız. Ancak bu, model skorlarının belirlenmesinde bir miktar rastgelelik bırakır. Yani, bir model 1000 satırlık bir set üzerinde iyi performans gösterse bile, farklı bir 1000 satırlık sette yanlış sonuçlar verebilir.

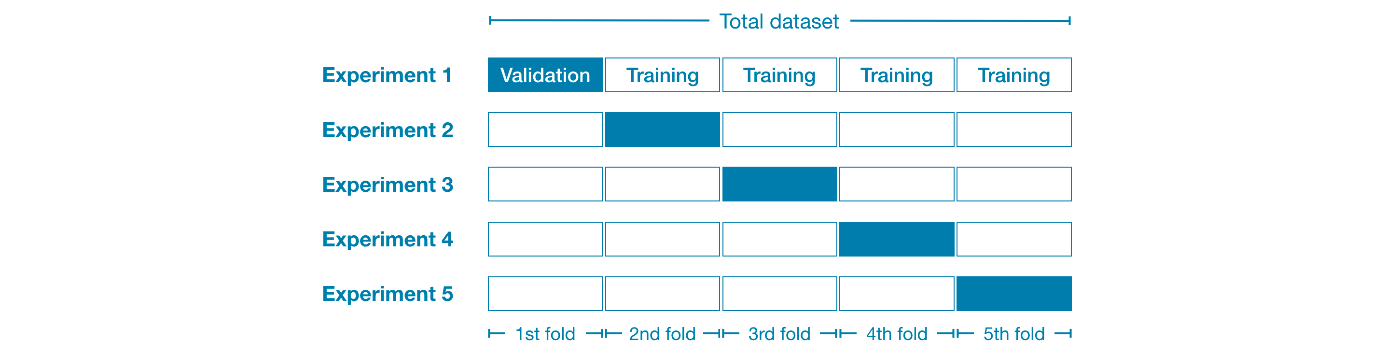
Uç bir örnek olarak, doğrulama setinizde sadece 1 satır veri olduğunu düşünebilirsiniz. Alternatif modelleri karşılaştırırsanız, tek bir veri noktası üzerinde hangi modelin en iyi tahmini yapacağı çoğunlukla şans eseri olacaktır!

Genel olarak, doğrulama seti ne kadar büyük olursa, model kalitesi ölçümümüzdeki rastgelelik (diğer adıyla "gürültü") o kadar az olur ve ölçüm o kadar güvenilir olur. Ne yazık ki, büyük bir doğrulama setine ancak eğitim verilerimizden satırları çıkararak sahip olabiliriz ve daha küçük eğitim veri setleri daha kötü modeller anlamına gelir!

# What is cross-validation?

Cross-validation'da (çapraz doğrulama), model kalitesini birden fazla kez ölçmek için modelleme sürecimizi verinin farklı alt kümeleri üzerinde çalıştırırız.

Örneğin, veriyi her biri tüm veri setinin %20'si olan 5 parçaya bölerek başlayabiliriz. Bu durumda, veriyi 5 "katmana" (fold) ayırdığımızı söyleriz.



Sonra, her bir katman için bir deney yürütürüz:

* **1. Deneyde**, ilk katmanı bir doğrulama (veya tutma) seti olarak kullanırız ve geri kalan her şeyi eğitim verisi olarak kullanırız. Bu bize, %20'lik bir tutma setine dayalı bir model kalitesi ölçümü verir.
* **2. Deneyde**, ikinci katmandaki verileri tutarız (ve ikinci katman hariç her şeyi modeli eğitmek için kullanırız). Tutulan set daha sonra model kalitesinin ikinci bir tahminini elde etmek için kullanılır.

Bu süreci, her katmanı bir kez tutma seti olarak kullanarak tekrarlarız. Tüm bunları birleştirdiğimizde, verinin %100'ü bir noktada tutma seti olarak kullanılmış olur ve veri setindeki tüm satırlara dayanan bir model kalitesi ölçümü elde ederiz (tüm satırları aynı anda kullanmasak bile).

# When should you use cross-validation?

Çapraz doğrulama, model kalitesinin daha doğru bir ölçümünü sağlar ve bu, özellikle birçok modelleme kararı alıyorsanız önemlidir. Ancak, birden fazla model (her katman için bir tane) tahmin ettiği için çalışması daha uzun sürebilir.

Peki, bu avantaj ve dezavantajlar göz önüne alındığında, her bir yaklaşımı ne zaman kullanmalısınız?

* **Küçük veri setleri için**, ek işlem yükü büyük bir sorun değilse, **çapraz doğrulama** çalıştırmalısınız.
* **Daha büyük veri setleri için**, tek bir doğrulama seti yeterlidir. Kodunuz daha hızlı çalışır ve verinin bir kısmını tekrar tekrar tutma seti olarak kullanmaya pek gerek kalmayacak kadar veriye sahip olabilirsiniz.

Büyük ve küçük bir veri setini birbirinden ayırmak için basit bir eşik yoktur. Ancak modelinizin çalışması birkaç dakika veya daha az sürüyorsa, muhtemelen çapraz doğrulamaya geçmeye değer.

Alternatif olarak, çapraz doğrulamayı çalıştırıp her bir deneyin skorlarının birbirine yakın olup olmadığına bakabilirsiniz. Eğer her deney aynı sonuçları veriyorsa, tek bir doğrulama seti muhtemelen yeterlidir.

# Example

Önceki derstekiyle aynı verilerle çalışacağız. Giriş verilerini X'e, çıkış verilerini ise y'ye yükleyeceğiz.

import pandas as pd

*# Read the data*

data = pd.read\_csv('../input/melbourne-housing-snapshot/melb\_data.csv')

*# Select subset of predictors*

cols\_to\_use = ['Rooms', 'Distance', 'Landsize', 'BuildingArea', 'YearBuilt']

X = data[cols\_to\_use]

*# Select target*

y = data.Price

Daha sonra, eksik değerleri doldurmak için bir atama aracı (imputer) ve tahmin yapmak için bir rastgele orman modeli kullanan bir işlem hattı (pipeline) tanımlıyoruz.

Pipelinelar olmadan çapraz doğrulama yapmak **mümkün olsa da**, oldukça zordur! Bir pipeline kullanmak, kodu oldukça anlaşılır hale getirecektir.

from sklearn.ensemble import RandomForestRegressor

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

my\_pipeline = Pipeline(steps=[('preprocessor', SimpleImputer()),

('model', RandomForestRegressor(n\_estimators=50,

random\_state=0))

])

Çapraz doğrulama puanlarını scikit-learn'deki cross\_val\_score() fonksiyonuyla elde ediyoruz. Katlama sayısını cv parametresiyle ayarlıyoruz.

from sklearn.model\_selection import cross\_val\_score

*# Multiply by -1 since sklearn calculates \*negative\* MAE*

scores = -1 \* cross\_val\_score(my\_pipeline, X, y,

cv=5,

scoring='neg\_mean\_absolute\_error')

print("MAE scores:**\n**", scores)

MAE scores:

[301628.7893587 303164.4782723 287298.331666 236061.84754543

260383.45111427]

**scoring** parametresi, raporlanacak model kalitesi ölçütünü seçer: bu örnekte **negatif ortalama mutlak hatayı (MAE)** seçtik. scikit-learn'ün belgeleri, [seçeneklerin bir listesini](https://www.google.com/search?q=https://scikit-learn.org/stable/modules/model_evaluation.html%23scoring-parameter) gösterir.

Negatif MAE belirtmemiz biraz şaşırtıcı olabilir. Scikit-learn'ün, tüm metriklerin yüksek bir sayının daha iyi olduğu şekilde tanımlandığı bir kuralı vardır. Burada negatif değerler kullanılması, bu kurala tutarlı kalmalarını sağlar, ancak negatif MAE başka yerlerde neredeyse hiç duyulmamış bir durumdur.

Alternatif modelleri karşılaştırmak için genellikle tek bir model kalitesi ölçütü isteriz. Bu nedenle, deneyler arasında ortalamayı alırız.

print("Average MAE score (across experiments):")

print(scores.mean())

Average MAE score (across experiments):

277707.3795913405

# Conclusion

Çapraz doğrulama kullanmak, model kalitesi için çok daha iyi bir ölçüm sunar ve ayrıca kodumuzu düzenleme gibi ek bir fayda sağlar: Artık ayrı eğitim ve doğrulama setlerini takip etmemize gerek kalmadığına dikkat edin. Bu yüzden, özellikle küçük veri setleri için iyi bir gelişmedir!