Welcome to Feature Engineering!

Bu kursta, harika bir makine öğrenmesi modeli oluşturmanın en önemli adımlarından biri olan **özellik mühendisliği** hakkında bilgi edineceksiniz. Şunları nasıl yapacağınızı öğreneceksiniz:

* **Karşılıklı bilgi (mutual information)** ile en önemli özelliklerin hangileri olduğunu belirlemek
* Birkaç gerçek dünya problem alanında yeni özellikler oluşturmak
* Yüksek kardinaliteli kategorik verileri bir **hedef kodlama (target encoding)** ile kodlamak
* **K-ortalamalar kümeleme (k-means clustering)** ile segmentasyon özellikleri yaratmak
* Bir veri kümesinin varyasyonunu **temel bileşen analizi (principal component analysis)** ile özelliklere ayrıştırmak

Uygulamalı alıştırmalar, tüm bu teknikleri **House Prices Getting Started** yarışmasına bir gönderim yapmak için uygulayan eksiksiz bir not defteri (notebook) oluşturacak. Bu kursu tamamladıktan sonra, performansınızı daha da artırmak için kullanabileceğiniz birçok fikre sahip olacaksınız.

Hazır mısınız? Başlayalım!

# The Goal of Feature Engineering

Özellik mühendisliğinin amacı, verilerinizi eldeki soruna daha uygun hale getirmektir.

Sıcaklık endeksi ve rüzgar soğukluğu gibi "hissedilen sıcaklık" ölçümlerini düşünün. Bu nicelikler, doğrudan ölçebildiğimiz hava sıcaklığı, nem ve rüzgar hızı gibi şeylere dayanarak insanlar tarafından algılanan sıcaklığı ölçmeye çalışır. Hissedilen bir sıcaklığı, aslında önemsediğimiz şeye, yani dışarının gerçekte nasıl hissedildiğine dair gözlemlenen veriyi daha alakalı hale getirme girişimi olan bir tür **özellik mühendisliği**nin sonucu olarak düşünebilirsiniz.

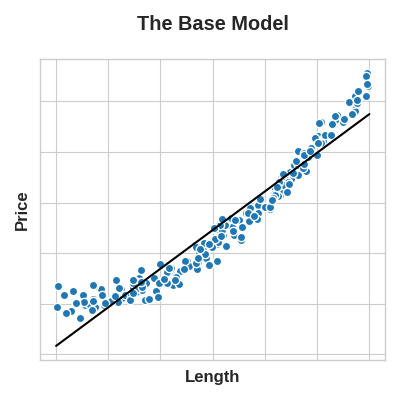
Özellik mühendisliğini şu amaçlarla yapabilirsiniz:

* Bir modelin tahminsel performansını artırmak
* Hesaplama veya veri ihtiyaçlarını azaltmak
* Sonuçların yorumlanabilirliğini geliştirmek

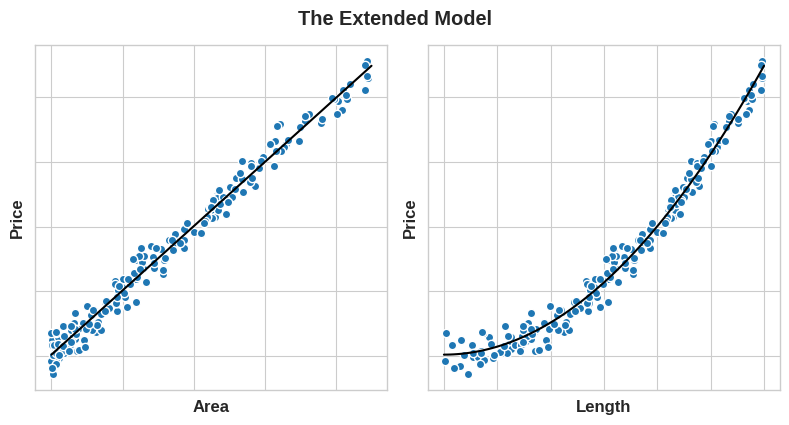
# A Guiding Principle of Feature Engineering

Bir özelliğin faydalı olması için, modelinizin öğrenebileceği bir ilişkiye sahip olması gerekir. Örneğin, **doğrusal modeller** yalnızca doğrusal ilişkileri öğrenebilir. Bu nedenle, doğrusal bir model kullanırken amacınız, özelliklerin hedefle olan ilişkisini doğrusal hale getirmek için **özellikleri dönüştürmektir**.

Buradaki ana fikir, bir özelliğe uyguladığınız dönüşümün aslında modelin kendisinin bir parçası haline gelmesidir. Diyelim ki, kare arsaların fiyatını bir kenarının uzunluğundan tahmin etmeye çalışıyorsunuz. Doğrusal bir modeli doğrudan Uzunluk'a uygulamak zayıf sonuçlar verir, çünkü ilişki doğrusal değildir.



Ancak Uzunluk özelliğini kare alarak 'Alan'ı elde edersek, doğrusal bir ilişki oluştururuz. Özellik kümesine Alan eklemek, bu doğrusal modelin artık bir parabole uyabileceği anlamına gelir. Bir başka deyişle, bir özelliği karelemek, doğrusal modele karelenmiş özelliklere uyma yeteneği kazandırdı.



Burada, özellik mühendisliğine yatırılan zamanın neden bu kadar yüksek bir geri dönüş sağlayabileceğini görüyorsunuz. Modelinizin öğrenemediği ilişkileri, dönüşümler aracılığıyla kendiniz sağlayabilirsiniz. Özellik kümenizi geliştirirken, modelinizin en iyi performansı elde etmek için hangi bilgileri kullanabileceğini düşünün.

# Example - Concrete Formulations

Bu fikirleri göstermek için, bir veri kümesine birkaç yapay özellik eklemenin rastgele orman modelinin tahmin performansını nasıl artırabileceğini göreceğiz.

"Concrete" (Beton) veri kümesi, çeşitli beton formülasyonlarını ve ortaya çıkan ürünün basınç dayanımını içerir. Basınç dayanımı, o tür betonun ne kadar yüke dayanabileceğini ölçen bir değerdir. Bu veri kümesinin görevi, bir betonun formülasyonuna bakarak basınç dayanımını tahmin etmektir.

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

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

df = pd.read\_csv("../input/fe-course-data/concrete.csv")

df.head()

| Cement | BlastFurnaceSlag | FlyAsh | Water | Superplasticizer | CoarseAggregate | FineAggregate | Age | CompressiveStrength |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 540.0 | 0.0 | 0.0 | 162.0 | 2.5 | 1040.0 | 676.0 | 28 | 79.99 |
| 1 | 540.0 | 0.0 | 0.0 | 162.0 | 2.5 | 1055.0 | 676.0 | 28 | 61.89 |
| 2 | 332.5 | 142.5 | 0.0 | 228.0 | 0.0 | 932.0 | 594.0 | 270 | 40.27 |
| 3 | 332.5 | 142.5 | 0.0 | 228.0 | 0.0 | 932.0 | 594.0 | 365 | 41.05 |
| 4 | 198.6 | 132.4 | 0.0 | 192.0 | 0.0 | 978.4 | 825.5 | 360 | 44.30 |

Burada, her bir beton çeşidinde kullanılan çeşitli bileşenleri görebilirsiniz. Birazdan, bunlardan türetilen bazı ek yapay özellikler eklemenin, bir modelin aralarındaki önemli ilişkileri öğrenmesine nasıl yardımcı olabileceğini göreceğiz.

Öncelikle, modeli artırılmamış veri kümesi üzerinde eğiterek bir **başlangıç seviyesi (baseline)** oluşturacağız. Bu, yeni özelliklerimizin gerçekten faydalı olup olmadığını belirlememize yardımcı olacaktır.

Bunun gibi başlangıç seviyeleri oluşturmak, özellik mühendisliği sürecinin başlangıcında iyi bir uygulamadır. Bir başlangıç seviyesi puanı, yeni özelliklerinizi saklamaya değer olup olmadığına veya onları atıp muhtemelen başka bir şey denemeniz gerekip gerekmediğine karar vermenize yardımcı olabilir.

X = df.copy()

y = X.pop("CompressiveStrength")

*# Train and score baseline model*

baseline = RandomForestRegressor(criterion="absolute\_error", random\_state=0)

baseline\_score = cross\_val\_score(

baseline, X, y, cv=5, scoring="neg\_mean\_absolute\_error"

)

baseline\_score = -1 \* baseline\_score.mean()

print(f"MAE Baseline Score: **{**baseline\_score**:**.4**}**")

MAE Baseline Score: 8.232

Eğer evde yemek yapıyorsanız, bir tarifteki malzemelerin **mutlak miktarlarından** ziyade **oranının**, tarifin nasıl sonuçlanacağı hakkında genellikle daha iyi bir tahminci olduğunu bilebilirsiniz. O halde, yukarıdaki özelliklerin oranlarının, **Basınç Dayanımı** için iyi bir tahminci olacağını düşünebiliriz.

Aşağıdaki hücre, veri kümesine üç yeni oran özelliği ekliyor.

X = df.copy()

y = X.pop("CompressiveStrength")

*# Create synthetic features*

X["FCRatio"] = X["FineAggregate"] / X["CoarseAggregate"]

X["AggCmtRatio"] = (X["CoarseAggregate"] + X["FineAggregate"]) / X["Cement"]

X["WtrCmtRatio"] = X["Water"] / X["Cement"]

*# Train and score model on dataset with additional ratio features*

model = RandomForestRegressor(criterion="absolute\_error", random\_state=0)

score = cross\_val\_score(

model, X, y, cv=5, scoring="neg\_mean\_absolute\_error"

)

score = -1 \* score.mean()

print(f"MAE Score with Ratio Features: **{**score**:**.4**}**")

MAE Score with Ratio Features: 7.948

Ve gerçekten de performans arttı! Bu, yeni oran özelliklerinin modele daha önce fark etmediği önemli bilgileri açığa çıkardığının kanıtıdır.

# Continue

Yeni özellikler geliştirmenin model performansını artırabileceğini gördük. Peki, veri setinde birleştirilmeye değer olabilecek özellikleri nasıl belirleyebilirsiniz? Ortak bilgilerle faydalı özellikleri keşfedin.