Introduction

Yeni bir veri kümesiyle ilk kez karşılaştığınızda, bu durum bazen bunaltıcı gelebilir. Yüzlerce veya binlerce özellikle, hatta çoğu zaman bir açıklaması bile olmadan karşılaşabilirsiniz. Nereden başlayacağınızı bilemeyebilirsiniz.

Harika bir ilk adım, bir **özellik fayda ölçütü** kullanarak bir sıralama oluşturmaktır. Bu ölçüt, bir özellik ile hedef arasındaki ilişkileri ölçen bir fonksiyondur. Ardından, başlangıçta geliştirmek için en faydalı özelliklerden oluşan daha küçük bir küme seçebilir ve zamanınızın iyi harcanacağından daha emin olabilirsiniz.

Kullanacağımız ölçüt, "karşılıklı bilgi" (mutual information) olarak adlandırılır. Karşılıklı bilgi, iki nicelik arasındaki ilişkiyi ölçmesi bakımından korelasyona oldukça benzer. Karşılıklı bilginin avantajı, korelasyonun yalnızca **doğrusal** ilişkileri tespit etmesine karşın, karşılıklı bilginin **her türlü** ilişkiyi tespit edebilmesidir.

Karşılıklı bilgi, harika bir genel amaçlı ölçüttür ve özellikle henüz hangi modeli kullanmak istediğinizi bilmediğiniz özellik geliştirme sürecinin başlangıcında çok faydalıdır. Özellikleri şu şekilde sıralayabiliriz:

* Kullanımı ve yorumlaması kolaydır.
* Hesaplama açısından verimlidir.
* Teorik olarak iyi temellendirilmiştir.
* Aşırı öğrenmeye karşı dirençlidir.
* Her türlü ilişkiyi tespit edebilir.

# Mutual Information and What it Measures

Karşılıklı bilgi, ilişkileri **belirsizlik** açısından tanımlar. İki nicelik arasındaki **karşılıklı bilgi** (KB), bir niceliğe dair bilginin diğer nicelik hakkındaki belirsizliği ne ölçüde azalttığının bir ölçüsüdür. Bir özelliğin değerini bilseydiniz, hedef hakkında ne kadar daha emin olurdunuz?

İşte **Ames Konut** verilerinden bir örnek. Şekil, bir evin dış cephe kalitesi ile satıldığı fiyat arasındaki ilişkiyi gösteriyor. Her bir nokta, bir evi temsil ediyor.



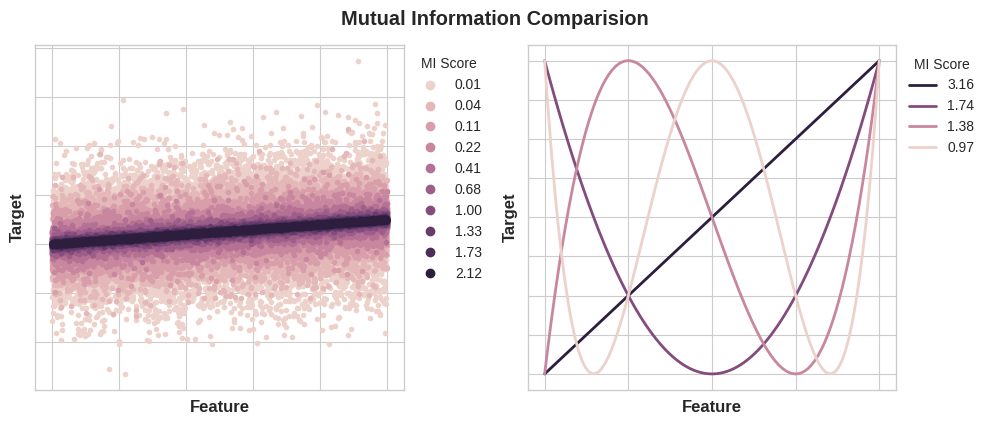
Şekilden de görebileceğimiz gibi, **ExterQual** değerini bilmek, karşılık gelen **Satış Fiyatı** hakkında sizi daha emin kılacaktır; çünkü **ExterQual**'ın her bir kategorisi, Satış Fiyatı'nı belirli bir aralıkta yoğunlaştırma eğilimindedir. **ExterQual**'ın **Satış Fiyatı** ile sahip olduğu karşılıklı bilgi, **ExterQual**'ın dört değeri üzerinden alınan Satış Fiyatı'ndaki ortalama belirsizlik azalmasıdır. Örneğin, "Fair" (Orta) değeri, "Typical" (Tipik) değerine göre daha az sıklıkta görüldüğü için, karşılıklı bilgi puanında daha az ağırlık alır.

(Teknik not: Burada belirsizlik olarak adlandırdığımız şey, bilgi teorisinden "entropi" olarak bilinen bir nicelik kullanılarak ölçülür. Bir değişkenin entropisi kabaca şunu ifade eder: "O değişkenin bir kez gerçekleşmesini tanımlamak için ortalama kaç tane evet-hayır sorusu sormanız gerekir?" Ne kadar çok soru sormanız gerekirse, değişken hakkında o kadar belirsiz olmalısınız. Karşılıklı bilgi, özelliğin hedef hakkında kaç soruyu yanıtlamasını beklediğinizdir.)

# Interpreting Mutual Information Scores

İki nicelik arasındaki olası en düşük karşılıklı bilgi değeri 0.0'dır. Karşılıklı bilgi sıfır olduğunda, nicelikler birbirinden bağımsızdır: Biri diğeri hakkında size hiçbir şey söyleyemez. Tersine, teorik olarak karşılıklı bilginin olabileceği bir üst sınır yoktur. Ancak pratikte, 2.0 civarı veya daha yüksek değerler nadir görülür. (Karşılıklı bilgi logaritmik bir niceliktir, bu yüzden çok yavaş artar.)

Aşağıdaki şekil, karşılıklı bilgi değerlerinin bir özelliğin hedefle olan ilişki türü ve derecesiyle nasıl bir ilişki kurduğuna dair bir fikir verecektir.



Sol: Özellik ve hedef arasındaki bağımlılık sıkılaştıkça karşılıklı bilgi artar. Sağ: Karşılıklı bilgi, her türlü ilişkiyi (sadece korelasyon gibi doğrusal değil) yakalayabilir.

Karşılıklı bilgiyi uygularken hatırlamanız gerekenler şunlardır:

* Karşılıklı bilgi, tek başına ele alındığında, bir özelliğin hedef için bir tahminci olarak sahip olduğu **görece potansiyeli** anlamanıza yardımcı olabilir.
* Bir özelliğin, diğer özelliklerle etkileşime girdiğinde çok bilgilendirici olması, ancak tek başına o kadar da bilgilendirici olmaması mümkündür. Karşılıklı bilgi, özellikler arasındaki **etkileşimleri tespit edemez.** Bu, **tek değişkenli** bir ölçüttür.
* Bir özelliğin **gerçek** faydası, onu kullandığınız **modele bağlıdır.** Bir özellik, hedefle olan ilişkisinin modelinizin öğrenebileceği bir ilişki olduğu ölçüde faydalıdır. Yüksek bir karşılıklı bilgi puanına sahip olması, modelinizin bu bilgiyi kullanabileceği anlamına gelmez. Bu ilişkiyi ortaya çıkarmak için önce özelliği dönüştürmeniz gerekebilir.

# Example - 1985 Automobiles

1985 model yılından 193 arabadan oluşan "Automobile" (Otomobil) veri kümesi. Bu veri kümesinin amacı, arabanın markası, gövde stili (body\_style) ve beygir gücü (horsepower) gibi 23 özelliğinden yola çıkarak arabanın **fiyatını** (hedef) tahmin etmektir. Bu örnekte, özellikleri karşılıklı bilgi ile sıralayacak ve sonuçları veri görselleştirmesi ile inceleyeceğiz.

Bu gizli hücre, bazı kütüphaneleri içe aktarıyor ve veri kümesini yüklüyor.

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import seaborn as sns

plt.style.use("seaborn-whitegrid")

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

df.head()

| symboling | make | fuel\_type | aspiration | num\_of\_doors | body\_style | drive\_wheels | engine\_location | wheel\_base | length | ... | engine\_size | fuel\_system | bore | stroke | compression\_ratio | horsepower | peak\_rpm | city\_mpg | highway\_mpg | price |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 3 | alfa-romero | gas | std | 2 | convertible | rwd | front | 88.6 | 168.8 | ... | 130 | mpfi | 3.47 | 2.68 | 9 | 111 | 5000 | 21 | 27 | 13495 |
| 1 | 3 | alfa-romero | gas | std | 2 | convertible | rwd | front | 88.6 | 168.8 | ... | 130 | mpfi | 3.47 | 2.68 | 9 | 111 | 5000 | 21 | 27 | 16500 |
| 2 | 1 | alfa-romero | gas | std | 2 | hatchback | rwd | front | 94.5 | 171.2 | ... | 152 | mpfi | 2.68 | 3.47 | 9 | 154 | 5000 | 19 | 26 | 16500 |
| 3 | 2 | audi | gas | std | 4 | sedan | fwd | front | 99.8 | 176.6 | ... | 109 | mpfi | 3.19 | 3.40 | 10 | 102 | 5500 | 24 | 30 | 13950 |
| 4 | 2 | audi | gas | std | 4 | sedan | 4wd | front | 99.4 | 176.6 | ... | 136 | mpfi | 3.19 | 3.40 | 8 | 115 | 5500 | 18 | 22 | 17450 |

MI için kullanılan scikit-learn algoritması, **ayrık** özellikleri **sürekli** özelliklerden farklı şekilde ele alır. Sonuç olarak, hangisinin hangisi olduğunu ona söylemeniz gerekir. Genel bir kural olarak, float veri tipine sahip olması gereken her şey ayrık değildir. Kategorik (object veya categorical veri tipine sahip) özellikler, onlara bir **etiket kodlaması (label encoding)** uygulayarak ayrık olarak ele alınabilir. (Etiket kodlamalarını, **Kategorik Değişkenler** dersimizde tekrar gözden geçirebilirsiniz.)

X = df.copy()

y = X.pop("price")

*# Label encoding for categoricals*

for colname **in** X.select\_dtypes("object"):

X[colname], \_ = X[colname].factorize()

*# All discrete features should now have integer dtypes (double-check this before using MI!)*

discrete\_features = X.dtypes == int

Karşılıklı bilgi ölçütleri, scikit-learn'ün feature\_selection modülünde yer alır: Biri gerçek değerli hedefler için (mutual\_info\_regression) ve diğeri kategorik hedefler için (mutual\_info\_classif). Bizim hedefimiz olan **fiyat**, gerçek değerli bir hedeftir. Bir sonraki hücre, özelliklerimiz için MI puanlarını hesaplar ve bunları düzenli bir veri çerçevesinde (dataframe) toplar.

from sklearn.feature\_selection import mutual\_info\_regression

def make\_mi\_scores(X, y, discrete\_features):

mi\_scores = mutual\_info\_regression(X, y, discrete\_features=discrete\_features)

mi\_scores = pd.Series(mi\_scores, name="MI Scores", index=X.columns)

mi\_scores = mi\_scores.sort\_values(ascending=False)

return mi\_scores

mi\_scores = make\_mi\_scores(X, y, discrete\_features)

mi\_scores[::3] *# show a few features with their MI scores*

curb\_weight 1.540126

highway\_mpg 0.951700

length 0.621566

fuel\_system 0.485085

stroke 0.389321

num\_of\_cylinders 0.330988

compression\_ratio 0.133927

fuel\_type 0.048139

Name: MI Scores, dtype: float64

Ve şimdi karşılaştırmaları kolaylaştırmak için bir çubuk grafiği:

def plot\_mi\_scores(scores):

scores = scores.sort\_values(ascending=True)

width = np.arange(len(scores))

ticks = list(scores.index)

plt.barh(width, scores)

plt.yticks(width, ticks)

plt.title("Mutual Information Scores")

plt.figure(dpi=100, figsize=(8, 5))

plot\_mi\_scores(mi\_scores)

Veri görselleştirme, bir fayda sıralamasının harika bir devamıdır. Bunlardan birkaçına daha yakından bakalım.

Beklenebileceği gibi, yüksek puan alan curb\_weight özelliği, hedef olan fiyatla güçlü bir ilişki sergiliyor.

sns.relplot(x="curb\_weight", y="price", data=df);

fuel\_type özelliğinin karşılıklı bilgi puanı oldukça düşüktür, ancak şekilden de görebileceğimiz gibi, **beygir gücü** özelliğinde farklı eğilimlere sahip iki **fiyat** popülasyonunu açıkça ayırıyor. Bu durum, fuel\_type'ın bir **etkileşim etkisi**ne katkıda bulunduğunu ve sonuçta önemsiz olmayabileceğini gösteriyor. Bir özelliğin karşılıklı bilgi puanına bakarak önemsiz olduğuna karar vermeden önce, olası etkileşim etkilerini araştırmak faydalıdır. Bu noktada alan bilgisi (domain knowledge) size çok rehberlik edebilir.

sns.lmplot(x="horsepower", y="price", hue="fuel\_type", data=df);

Veri görselleştirme, özellik mühendisliği araç setinize harika bir katkı sağlar. Karşılıklı bilgi gibi faydalı metriklerin yanı sıra, bu tür görselleştirmeler verilerinizdeki önemli ilişkileri keşfetmenize de yardımcı olabilir. Daha fazla bilgi edinmek için Veri Görselleştirme kursumuza göz atın!