Introduction

Bu derste göreceğimiz **hedef kodlama (target encoding)** tekniği, daha önce gördüğümüz sayısal özelliklere yönelik tekniklerden farklı olarak, **kategorik özellikler** için tasarlanmıştır. Bu yöntem, **tek-sıcak (one-hot)** veya **etiket kodlama (label encoding)** gibi, kategorileri sayılara dönüştüren bir yöntemdir. Ancak farkı, bu kodlamayı oluşturmak için **hedefi** de kullanmasıdır. Bu nedenle, hedef kodlamaya **gözetimli (supervised)** bir özellik mühendisliği tekniği denir.

import pandas as pd

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

# Target Encoding

**Hedef kodlama (target encoding)**, bir özelliğin kategorilerini, hedeften türetilen bir sayıyla değiştiren herhangi bir kodlama türüdür.

Bunun basit ve etkili bir versiyonu, 3. Derste gördüğümüz grup toplama (group aggregation) yöntemini, örneğin ortalamayı, uygulamaktır. **Automobiles** veri setini kullanarak bu yöntem, her bir araç markasının ortalama fiyatını hesaplar:

autos["make\_encoded"] = autos.groupby("make")["price"].transform("mean")

autos[["make", "price", "make\_encoded"]].head(10)

| make | price | make\_encoded |
| --- | --- | --- |
| 0 | alfa-romero | 13495 | 15498.333333 |
| 1 | alfa-romero | 16500 | 15498.333333 |
| 2 | alfa-romero | 16500 | 15498.333333 |
| 3 | audi | 13950 | 17859.166667 |
| 4 | audi | 17450 | 17859.166667 |
| 5 | audi | 15250 | 17859.166667 |
| 6 | audi | 17710 | 17859.166667 |
| 7 | audi | 18920 | 17859.166667 |
| 8 | audi | 23875 | 17859.166667 |
| 9 | bmw | 16430 | 26118.750000 |

Bu tür hedef kodlamaya bazen **ortalamalı kodlama (mean encoding)** da denir. İkili bir hedefe uygulandığında ise buna **kutu sayma (bin counting)** adı verilir. (Karşılaşabileceğiniz diğer isimler arasında: olasılık kodlaması, etki kodlaması ve 'bırak-bir-dışarıda' (leave-one-out) kodlaması bulunur.)

# Smoothing

Böyle bir kodlama türü, ne yazık ki birkaç sorun sunar.

### 1. Bilinmeyen Kategoriler

İlk sorun **bilinmeyen kategorilerdir**. Hedef kodlamalar, özellikle aşırı uyum (overfitting) riski taşıdığı için, bağımsız bir "kodlama" bölümünde eğitilmeleri gerekir. Kodlamayı gelecekteki bölümlere birleştirdiğinizde, **Pandas**, kodlama bölümünde bulunmayan herhangi bir kategori için eksik değerleri dolduracaktır. Bu eksik değerleri bir şekilde atamanız (impute) gerekir.

### 2. Nadir Kategoriler

İkinci sorun ise **nadir kategorilerdir**. Bir kategori veri setinde sadece birkaç kez yer aldığında, grubu üzerinde hesaplanan istatistiklerin çok doğru olması pek olası değildir. **Automobiles** veri setinde, **mercury** markası sadece bir kez yer almaktadır. Hesapladığımız "ortalama" fiyat, sadece o tek aracın fiyatıdır ve bu, gelecekte görebileceğimiz Mercury'lerin çok temsili olmayabilir. Nadir kategorileri hedef kodlamak, aşırı uyumu daha olası hale getirebilir.

Bu sorunlara bir çözüm, **yumuşatma (smoothing)** eklemektir. Buradaki fikir, **kategori içi** ortalamayı, **genel** ortalama ile karıştırmaktır. Nadir kategoriler, kendi kategori ortalamalarına daha az ağırlık alırken, eksik kategoriler sadece genel ortalamayı alır.

Sözde kod (pseudocode) olarak bu şekilde ifade edilebilir:

encoding = weight \* in\_category + (1 - weight) \* overall

Burada ağırlık, kategori frekansından hesaplanan 0 ile 1 arasında bir değerdir.

Ağırlık değerini belirlemenin kolay bir yolu, bir m-tahmini hesaplamaktır:

weight = n / (n + m)

Burada n, söz konusu kategorinin verilerde toplam kaç kez geçtiğini gösterir. m parametresi "düzeltme faktörünü" belirler. m'nin daha büyük değerleri, genel tahmine daha fazla ağırlık verir.

Otomobiller veri setinde Chevrolet marka üç araç bulunmaktadır. m=2.0 seçerseniz, Chevrolet kategorisi ortalama Chevrolet fiyatının %60'ı artı genel ortalama fiyatın %40'ı ile kodlanacaktır.

chevrolet = 0.6 \* 6000.00 + 0.4 \* 13285.03

m değeri seçilirken, kategorilerin ne kadar değişken olmasını beklediğinizi düşünün. Bir aracın fiyatı, her marka içinde çok fazla değişkenlik gösteriyor mu? İyi tahminler elde etmek için çok fazla veriye ihtiyacınız olur mu? Eğer öyleyse, m için daha büyük bir değer seçmek daha iyi olabilir; eğer her markanın ortalama fiyatı nispeten kararlıysa, daha küçük bir değer de uygun olabilir.

Hedef kodlama, aşağıdaki durumlar için harika bir yöntemdir:

* **Yüksek Kardinaliteli Özellikler:** Çok sayıda kategoriye sahip bir özelliğin kodlanması sorunlu olabilir: tek-sıcak kodlama çok fazla özellik üretebilirken, etiket kodlama gibi alternatifler o özellik için uygun olmayabilir. Hedef kodlama, kategoriler için, özelliğin en önemli özelliği olan hedefle ilişkisini kullanarak sayılar türetir.
* **Alan Bilgisiyle Desteklenen Özellikler:** Daha önceki deneyimlerinizden, kategorik bir özelliğin, bir özellik metriği ile düşük puan alsa bile önemli olabileceğini düşünebilirsiniz. Hedef kodlama, bir özelliğin gerçek bilgilendiriciliğini ortaya çıkarmaya yardımcı olabilir.

# Example - MovieLens1M

MovieLens1M veri kümesi, MovieLens web sitesi kullanıcılarının bir milyon film derecelendirmesini ve her kullanıcıyı ve filmi açıklayan özellikleri içerir. Bu gizli hücre her şeyi ayarlar:

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import seaborn as sns

import warnings

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

plt.rc("figure", autolayout=True)

plt.rc(

"axes",

labelweight="bold",

labelsize="large",

titleweight="bold",

titlesize=14,

titlepad=10,

)

warnings.filterwarnings('ignore')

df = pd.read\_csv("../input/fe-course-data/movielens1m.csv")

df = df.astype(np.uint8, errors='ignore') *# reduce memory footprint*

print("Number of Unique Zipcodes: **{}**".format(df["Zipcode"].nunique()))

Number of Unique Zipcodes: 3439

3000'den fazla kategoriye sahip olan **Zipcode** özelliği, hedef kodlama için iyi bir adaydır ve bu veri setinin büyüklüğü (bir milyonun üzerinde satır), kodlamayı oluşturmak için biraz veri ayırabileceğimiz anlamına gelir.

Hedef kodlayıcıyı eğitmek için %25'lik bir bölüm oluşturarak başlayacağız.

X = df.copy()

y = X.pop('Rating')

X\_encode = X.sample(frac=0.25)

y\_encode = y[X\_encode.index]

X\_pretrain = X.drop(X\_encode.index)

y\_train = y[X\_pretrain.index]

Scikit-learn-contrib'deki category\_encoders paketi, Posta Kodu özelliğimizi kodlamak için kullanacağımız bir m-estimate kodlayıcısı uygular.

from category\_encoders import MEstimateEncoder

# Create the encoder instance. Choose m to control noise.

encoder = MEstimateEncoder(cols=["Zipcode"], m=5.0)

# Fit the encoder on the encoding split.

encoder.fit(X\_encode, y\_encode)

# Encode the Zipcode column to create the final training data

X\_train = encoder.transform(X\_pretrain)

Kodlanmış değerleri hedef değerlerle karşılaştırarak kodlamamızın ne kadar bilgilendirici olabileceğini görelim.

plt.figure(dpi=90)

ax = sns.distplot(y, kde=False, norm\_hist=True)

ax = sns.kdeplot(X\_train.Zipcode, color='r', ax=ax)

ax.set\_xlabel("Rating")

ax.legend(labels=['Zipcode', 'Rating']);

Kodlanmış Posta Kodu özelliğinin dağılımı, gerçek derecelendirmelerin dağılımını kabaca takip eder; bu da, film izleyicilerinin derecelendirmelerinin posta kodundan posta koduna yeterince farklı olduğu ve hedef kodlamamızın yararlı bilgileri yakalayabildiği anlamına gelir.