Bu eğitimde, kategorik değişkenin ne olduğunu ve bu tür verileri işlemek için üç yaklaşımı öğreneceksiniz.

Introduction[¶](https://www.kaggle.com/code/alexisbcook/categorical-variables#Introduction)

**Kategorik değişken**, yalnızca sınırlı sayıda değer alan bir değişkendir.

Örneğin, bir anketin size kahvaltıyı ne sıklıkta yediğinizi sorduğunu ve dört seçenek sunduğunu düşünün: "Asla", "Nadiren", "Çoğu gün" veya "Her gün". Bu durumda, veriler kategoriktir çünkü yanıtlar sabit bir kategori kümesine girer.

İnsanlar, hangi marka arabaya sahip oldukları hakkında bir ankete yanıt verseydi, yanıtlar "Honda", "Toyota" ve "Ford" gibi kategorilere ayrılırdı. Bu durumda da veriler kategoriktir.

Bu değişkenleri önceden işlem yapmadan Python'daki çoğu makine öğrenmesi modeline dahil etmeye çalışırsanız bir hata alırsınız. Bu eğitimde, kategorik verilerinizi hazırlamak için kullanabileceğiniz üç yaklaşımı karşılaştıracağız.

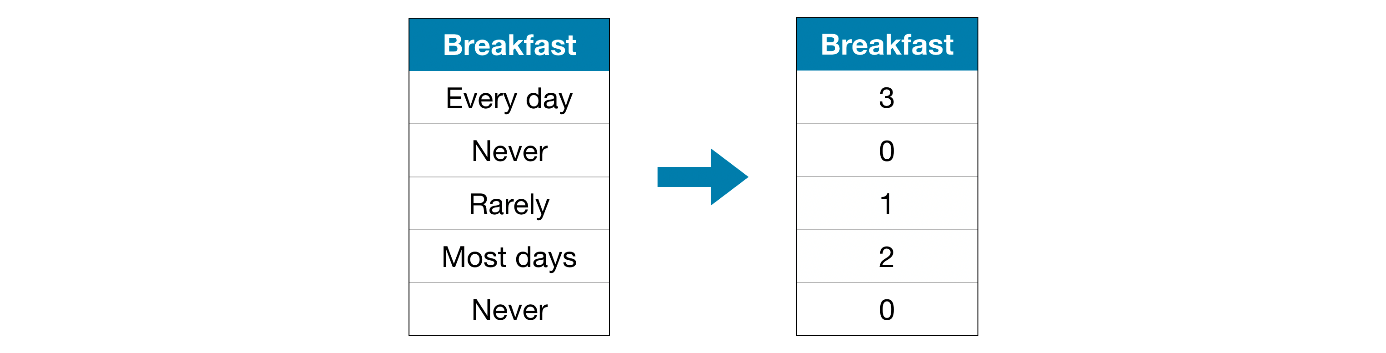
# Three Approaches[¶](https://www.kaggle.com/code/alexisbcook/categorical-variables#Three-Approaches)

### **1) Drop Categorical Variables**[**¶**](https://www.kaggle.com/code/alexisbcook/categorical-variables#1)-Drop-Categorical-Variables)

Kategorik değişkenlerle başa çıkmanın en kolay yolu, onları veri setinden çıkarmaktır. Bu yaklaşım, ancak bu sütunlar faydalı bilgiler içermiyorsa işe yarar.

### **2) Ordinal Encoding**[**¶**](https://www.kaggle.com/code/alexisbcook/categorical-variables#2)-Ordinal-Encoding)

Sıralı kodlama her benzersiz değeri farklı bir tam sayıya atar.

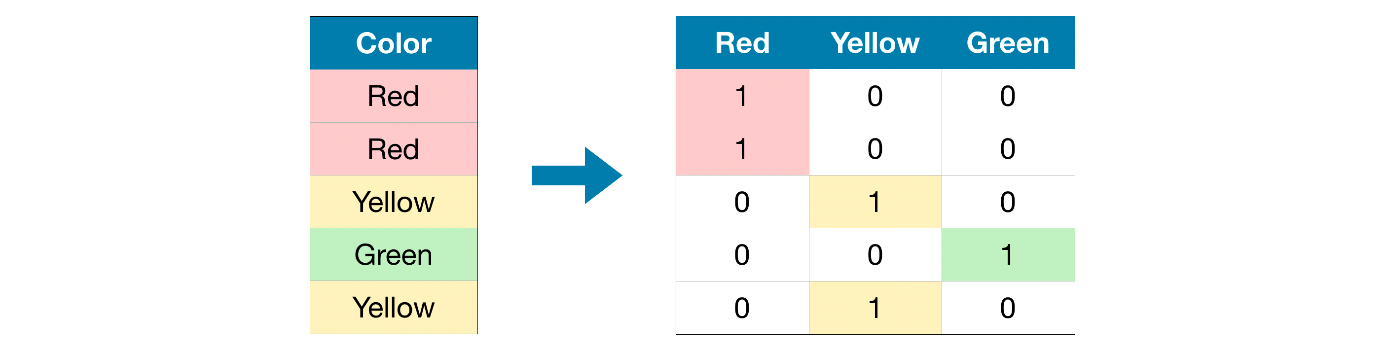


Bu yaklaşım, kategoriler arasında bir sıralama olduğunu varsayar: "Asla" (0) < "Nadiren" (1) < "Çoğu gün" (2) < "Her gün" (3).

Bu varsayım, bu örnekte mantıklıdır çünkü kategoriler arasında tartışmasız bir sıralama vardır. Tüm kategorik değişkenlerin değerlerinde net bir sıralama olmamakla birlikte, bu tür sıralamaya sahip olanlara **sıralı değişkenler (ordinal variables)** denir. Ağaç tabanlı modeller (karar ağaçları ve rastgele ormanlar gibi) için, sıralı kodlamanın (ordinal encoding) sıralı değişkenlerle iyi çalışmasını bekleyebilirsiniz.

### **3) One-Hot Encoding**[**¶**](https://www.kaggle.com/code/alexisbcook/categorical-variables#3)-One-Hot-Encoding)

**Tekil sıcak kodlama (One-hot encoding)**, orijinal verideki her olası değerin varlığını (veya yokluğunu) belirten yeni sütunlar oluşturur. Bunu anlamak için bir örnek üzerinden ilerleyeceğiz.



Orijinal veri setinde, "Renk" üç kategorili bir kategorik değişkendir: "Kırmızı", "Sarı" ve "Yeşil". Bu değişkenin tekil sıcak kodlaması (one-hot encoding), her olası değer için bir sütun ve orijinal veri setindeki her satır için bir satır içerir. Orijinal değer "Kırmızı" olduğunda "Kırmızı" sütununa 1 koyarız; eğer orijinal değer "Sarı" ise "Sarı" sütununa 1 koyarız ve bu şekilde devam ederiz.

Sıralı kodlamanın (ordinal encoding) aksine, tekil sıcak kodlama kategoriler arasında bir sıralama olduğunu **varsaymaz**. Bu nedenle, kategorik verilerde belirgin bir sıralama yoksa (örneğin, "Kırmızı" "Sarı"dan ne daha fazla ne de daha azdır), bu yaklaşımın özellikle iyi sonuç vermesini bekleyebilirsiniz. İçsel bir sıralaması olmayan kategorik değişkenlere **nominal değişkenler** diyoruz.

Tekil sıcak kodlama, kategorik değişken çok sayıda değer alıyorsa genellikle iyi performans göstermez (yani, 15'ten fazla farklı değer alan değişkenler için genellikle kullanmazsınız).

# Example[¶](https://www.kaggle.com/code/alexisbcook/categorical-variables#Example)

Daha önceki eğitimde olduğu gibi, **Melbourne Konut veri seti** ile çalışacağız.

Veri yükleme adımına odaklanmayacağız. Bunun yerine, eğitim ve doğrulama verilerine (training ve validation data) zaten sahip olduğunuzu ve bu verilerin **X\_train**, **X\_valid**, **y\_train** ve **y\_valid** değişkenlerinde bulunduğunu düşünebilirsiniz.

import pandas as pd

from sklearn.model\_selection import train\_test\_split

*# Read the data*

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

*# Separate target from predictors*

y = data.Price

X = data.drop(['Price'], axis=1)

*# Divide data into training and validation subsets*

X\_train\_full, X\_valid\_full, y\_train, y\_valid = train\_test\_split(X, y, train\_size=0.8, test\_size=0.2,

random\_state=0)

*# Drop columns with missing values (simplest approach)*

cols\_with\_missing = [col for col **in** X\_train\_full.columns if X\_train\_full[col].isnull().any()]

X\_train\_full.drop(cols\_with\_missing, axis=1, inplace=True)

X\_valid\_full.drop(cols\_with\_missing, axis=1, inplace=True)

*# "Cardinality" means the number of unique values in a column*

*# Select categorical columns with relatively low cardinality (convenient but arbitrary)*

low\_cardinality\_cols = [cname for cname **in** X\_train\_full.columns if X\_train\_full[cname].nunique() < 10 **and**

X\_train\_full[cname].dtype == "object"]

*# Select numerical columns*

numerical\_cols = [cname for cname **in** X\_train\_full.columns if X\_train\_full[cname].dtype **in** ['int64', 'float64']]

*# Keep selected columns only*

my\_cols = low\_cardinality\_cols + numerical\_cols

X\_train = X\_train\_full[my\_cols].copy()

X\_valid = X\_valid\_full[my\_cols].copy()

Aşağıda head() metoduyla eğitim verilerine bir göz atalım.

X\_train.head()

| Type | Method | Regionname | Rooms | Distance | Postcode | Bedroom2 | Bathroom | Landsize | Lattitude | Longtitude | Propertycount |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12167 | u | S | Southern Metropolitan | 1 | 5.0 | 3182.0 | 1.0 | 1.0 | 0.0 | -37.85984 | 144.9867 | 13240.0 |
| 6524 | h | SA | Western Metropolitan | 2 | 8.0 | 3016.0 | 2.0 | 2.0 | 193.0 | -37.85800 | 144.9005 | 6380.0 |
| 8413 | h | S | Western Metropolitan | 3 | 12.6 | 3020.0 | 3.0 | 1.0 | 555.0 | -37.79880 | 144.8220 | 3755.0 |
| 2919 | u | SP | Northern Metropolitan | 3 | 13.0 | 3046.0 | 3.0 | 1.0 | 265.0 | -37.70830 | 144.9158 | 8870.0 |
| 6043 | h | S | Western Metropolitan | 3 | 13.3 | 3020.0 | 3.0 | 1.0 | 673.0 | -37.76230 | 144.8272 | 4217.0 |

Şimdi, eğitim verilerindeki tüm kategorik değişkenlerin bir listesini elde edeceğiz.

Bunu, her sütunun **veri tipini (veya dtype)** kontrol ederek yapıyoruz. **'object'** veri tipi, bir sütunun metin içerdiğini gösterir (teorik olarak başka şeyler de olabilir, ancak bu bizim amacımız için önemli değil). Bu veri setinde, metin içeren sütunlar kategorik değişkenleri belirtir.

*# Get list of categorical variables*

s = (X\_train.dtypes == 'object')

object\_cols = list(s[s].index)

print("Categorical variables:")

print(object\_cols)

Categorical variables:

['Type', 'Method', 'Regionname']

### **Define Function to Measure Quality of Each Approach**[**¶**](https://www.kaggle.com/code/alexisbcook/categorical-variables#Define-Function-to-Measure-Quality-of-Each-Approach)

Kategorik değişkenlerle başa çıkmak için kullanılan üç farklı yaklaşımı karşılaştırmak amacıyla **score\_dataset()** adında bir fonksiyon tanımlıyoruz. Bu fonksiyon, bir **rastgele orman (random forest)** modelinden elde edilen **ortalama mutlak hatayı (mean absolute error - MAE)** rapor eder. Genel olarak, MAE'nin mümkün olduğunca düşük olmasını isteriz!

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error

*# Function for comparing different approaches*

def score\_dataset(X\_train, X\_valid, y\_train, y\_valid):

model = RandomForestRegressor(n\_estimators=100, random\_state=0)

model.fit(X\_train, y\_train)

preds = model.predict(X\_valid)

return mean\_absolute\_error(y\_valid, preds)

### **Score from Approach 1 (Drop Categorical Variables)**[**¶**](https://www.kaggle.com/code/alexisbcook/categorical-variables#Score-from-Approach-1-(Drop-Categorical-Variables))

select\_dtypes() metoduyla nesne sütunlarını bırakıyoruz.

drop\_X\_train = X\_train.select\_dtypes(exclude=['object'])

drop\_X\_valid = X\_valid.select\_dtypes(exclude=['object'])

print("MAE from Approach 1 (Drop categorical variables):")

print(score\_dataset(drop\_X\_train, drop\_X\_valid, y\_train, y\_valid))

MAE from Approach 1 (Drop categorical variables):

175703.48185157913

### **Score from Approach 2 (Ordinal Encoding)**[**¶**](https://www.kaggle.com/code/alexisbcook/categorical-variables#Score-from-Approach-2-(Ordinal-Encoding))

Scikit-learn kütüphanesinde, sıralı kodlamaları elde etmek için kullanılabilecek **OrdinalEncoder** adında bir sınıf bulunmaktadır. Bu sınıfı kullanarak, kategorik değişkenler üzerinde döngüye girip her sütuna ayrı ayrı sıralı kodlayıcıyı uyguluyoruz.

from sklearn.preprocessing import OrdinalEncoder

*# Make copy to avoid changing original data*

label\_X\_train = X\_train.copy()

label\_X\_valid = X\_valid.copy()

*# Apply ordinal encoder to each column with categorical data*

ordinal\_encoder = OrdinalEncoder()

label\_X\_train[object\_cols] = ordinal\_encoder.fit\_transform(X\_train[object\_cols])

label\_X\_valid[object\_cols] = ordinal\_encoder.transform(X\_valid[object\_cols])

print("MAE from Approach 2 (Ordinal Encoding):")

print(score\_dataset(label\_X\_train, label\_X\_valid, y\_train, y\_valid))

MAE from Approach 2 (Ordinal Encoding):

165936.40548390493

Yukarıdaki kod hücresinde, her sütun için her benzersiz değere rastgele farklı bir tam sayı atadık. Bu, özel etiketler sağlamaktan daha basit olan yaygın bir yaklaşımdır. Ancak, tüm sıralı değişkenler için daha iyi bilgilendirilmiş etiketler sağlarsak performansta ek bir artış bekleyebiliriz.

### **Score from Approach 3 (One-Hot Encoding)**[**¶**](https://www.kaggle.com/code/alexisbcook/categorical-variables#Score-from-Approach-3-(One-Hot-Encoding))

One-hot kodlama elde etmek için scikit-learn'deki **OneHotEncoder** sınıfını kullanıyoruz. Bu sınıfın davranışını özelleştirmek için kullanılabilecek birçok parametre vardır.

* **handle\_unknown='ignore'** parametresini, doğrulama verileri, eğitim verilerinde temsil edilmeyen sınıflar içerdiğinde hatalardan kaçınmak için ayarladık.
* **sparse=False** ayarı ise, kodlanmış sütunların bir NumPy dizisi olarak döndürülmesini sağlar (seyrek bir matris yerine).

Kodlayıcıyı kullanmak için, yalnızca tekil sıcak kodlama yapmak istediğimiz kategorik sütunları sağlıyoruz. Örneğin, eğitim verilerini kodlamak için X\_train[object\_cols]'ı veriyoruz. (Aşağıdaki kod hücresinde object\_cols, kategorik veriler içeren sütun adlarının bir listesidir ve bu nedenle X\_train[object\_cols], eğitim setindeki tüm kategorik verileri içerir.)

from sklearn.preprocessing import OneHotEncoder

*# Apply one-hot encoder to each column with categorical data*

OH\_encoder = OneHotEncoder(handle\_unknown='ignore', sparse=False)

OH\_cols\_train = pd.DataFrame(OH\_encoder.fit\_transform(X\_train[object\_cols]))

OH\_cols\_valid = pd.DataFrame(OH\_encoder.transform(X\_valid[object\_cols]))

*# One-hot encoding removed index; put it back*

OH\_cols\_train.index = X\_train.index

OH\_cols\_valid.index = X\_valid.index

*# Remove categorical columns (will replace with one-hot encoding)*

num\_X\_train = X\_train.drop(object\_cols, axis=1)

num\_X\_valid = X\_valid.drop(object\_cols, axis=1)

*# Add one-hot encoded columns to numerical features*

OH\_X\_train = pd.concat([num\_X\_train, OH\_cols\_train], axis=1)

OH\_X\_valid = pd.concat([num\_X\_valid, OH\_cols\_valid], axis=1)

*# Ensure all columns have string type*

OH\_X\_train.columns = OH\_X\_train.columns.astype(str)

OH\_X\_valid.columns = OH\_X\_valid.columns.astype(str)

print("MAE from Approach 3 (One-Hot Encoding):")

print(score\_dataset(OH\_X\_train, OH\_X\_valid, y\_train, y\_valid))

MAE from Approach 3 (One-Hot Encoding):

166089.4893009678

# Which approach is best?[¶](https://www.kaggle.com/code/alexisbcook/categorical-variables#Which-approach-is-best?)

Bu durumda, en yüksek MAE (Ortalama Mutlak Hata) skoruna sahip olduğu için kategorik sütunları atmak (Yaklaşım 1) en kötü performansı göstermiştir. Diğer iki yaklaşıma gelince, dönen MAE skorları birbirine çok yakın olduğu için, birinin diğerine göre anlamlı bir faydası yok gibi görünüyor.

Genel olarak, tekil sıcak kodlama (Yaklaşım 3) tipik olarak en iyi performansı gösterirken, kategorik sütunları atmak (Yaklaşım 1) en kötü performansı gösterir. Ancak bu durum, veri setine göre değişir.

# Conclusion[¶](https://www.kaggle.com/code/alexisbcook/categorical-variables#Conclusion)

Dünya kategorik verilerle dolu. Bu yaygın veri türünü nasıl kullanacağınızı bilirseniz çok daha etkili bir veri bilimcisi olursunuz!