Bu eğitimde, eksik değerlerle başa çıkmak için üç yaklaşımı öğreneceksiniz. Ardından, bu yaklaşımların etkinliğini gerçek dünya veri setinde karşılaştıracaksınız.

Introduction[¶](https://www.kaggle.com/code/alexisbcook/missing-values" \l "Introduction" \t "_self)

Verilerde eksik değerlerin oluşmasının birçok yolu vardır. Örneğin,

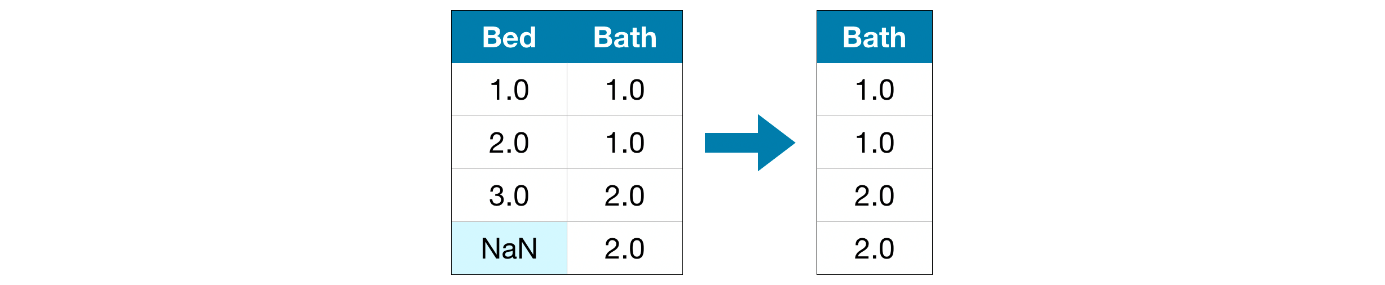
* 2 yatak odalı bir evin, üçüncü bir yatak odasının boyutu için bir değeri olmaz.
* Bir anket katılımcısı gelirini paylaşmamayı seçebilir.

Çoğu makine öğrenmesi kütüphanesi (scikit-learn dahil), eksik değerler içeren veriler kullanarak bir model oluşturmaya çalıştığınızda hata verir. Bu nedenle, aşağıdaki stratejilerden birini seçmeniz gerekecektir.

# Three Approaches[¶](https://www.kaggle.com/code/alexisbcook/missing-values" \l "Three-Approaches" \t "_self)

### **1) A Simple Option: Drop Columns with Missing Values[¶](https://www.kaggle.com/code/alexisbcook/missing-values" \l "1)-A-Simple-Option:-Drop-Columns-with-Missing-Values" \t "_self)**

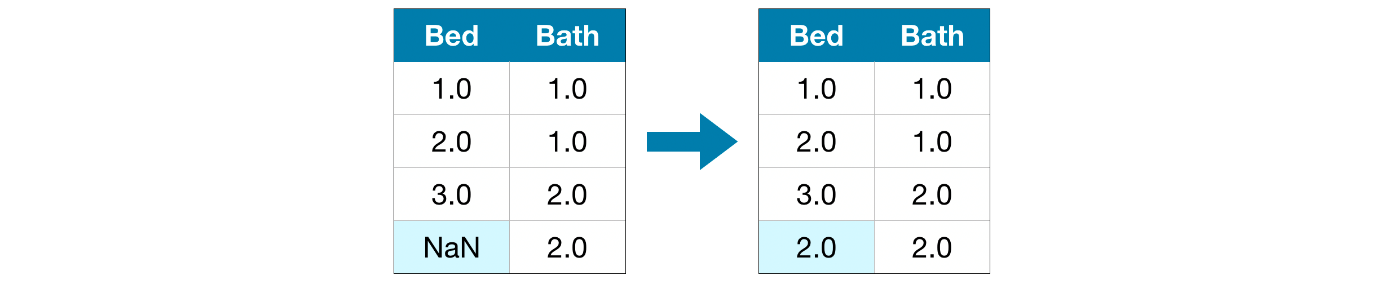
En basit seçenek, eksik değerlere sahip sütunları silmektir.

,

Atılan sütunlardaki değerlerin çoğu eksik değilse, bu yaklaşım ile model, potansiyel olarak faydalı olabilecek birçok bilgiye erişimini kaybeder. Aşırı bir örnek vermek gerekirse, 10.000 satırlık bir veri kümesi düşünün; bu veri kümesinde, önemli bir sütunda yalnızca tek bir veri girişi eksik olsun. Bu yaklaşım, o sütunu tamamen atar!

### **2) A Better Option: Imputation[¶](https://www.kaggle.com/code/alexisbcook/missing-values" \l "2)-A-Better-Option:-Imputation" \t "_self)**

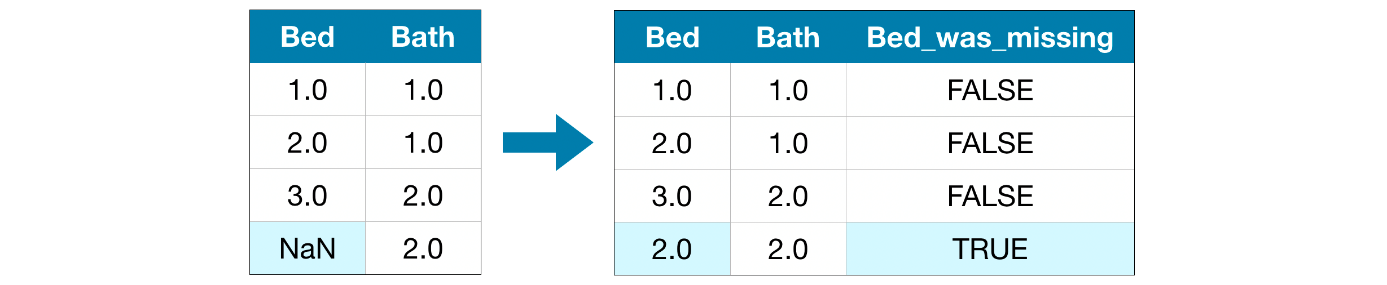
**Atama (Imputation)**, eksik değerleri bazı sayılarla doldurur. Örneğin, her sütundaki eksik değerleri ortalama değerle doldurabiliriz.



Atanan değer çoğu durumda tam olarak doğru olmayacaktır, ancak genellikle sütunu tamamen atmaktan elde edeceğinizden daha doğru modellere yol açar.

### **3) An Extension To Imputation[¶](https://www.kaggle.com/code/alexisbcook/missing-values" \l "3)-An-Extension-To-Imputation" \t "_self)**

Imputation (değer atama) standart bir yaklaşımdır ve genellikle iyi sonuç verir. Ancak, atanan değerler, asıl değerlerinden (veri setinde toplanmayan) sistematik olarak yüksek veya düşük olabilir. Ya da eksik değerli satırlar, başka bir şekilde benzersiz olabilir. Böyle bir durumda, modeliniz hangi değerlerin başlangıçta eksik olduğunu göz önünde bulundurarak daha iyi tahminler yapabilir.



Bu yaklaşımda, daha önce olduğu gibi eksik değerleri atarız. Ek olarak, orijinal veri setinde eksik girişi olan her sütun için, atanan girişlerin konumunu gösteren yeni bir sütun ekleriz.

Bazı durumlarda bu, sonuçları önemli ölçüde iyileştirirken, bazen ise hiç yardımcı olmaz.

# Example

Örnekte, **Melbourne Konut veri seti** ile çalışacağız. Modelimiz, oda sayısı ve arsa büyüklüğü gibi bilgileri kullanarak ev fiyatını tahmin edecek.

Veri yükleme adımına odaklanmayacağız. Bunun yerine, eğitim ve doğrulama verilerini (training and validation data) zaten elinizde, yani **X\_train**, **X\_valid**, **y\_train** ve **y\_valid** içinde bulundurduğunuzu düşünebilirsiniz.

import pandas as pd

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

*# Load the data*

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

*# Select target*

y = data.Price

*# To keep things simple, we'll use only numerical predictors*

melb\_predictors = data.drop(['Price'], axis=1)

X = melb\_predictors.select\_dtypes(exclude=['object'])

*# Divide data into training and validation subsets*

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

random\_state=0)

### **Define Function to Measure Quality of Each Approach[¶](https://www.kaggle.com/code/alexisbcook/missing-values" \l "Define-Function-to-Measure-Quality-of-Each-Approach" \t "_self)**

Missing value'larla (eksik değerlerle) başa çıkma konusunda farklı yaklaşımları karşılaştırmak için score\_dataset() adında bir fonksiyon tanımlıyoruz. Bu fonksiyon, bir rastgele orman (random forest) modelinden elde edilen **ortalama mutlak hatayı (mean absolute error - MAE)** raporlar.

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=10, 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 Columns with Missing Values)**[**¶**](https://www.kaggle.com/code/alexisbcook/missing-values#Score-from-Approach-1-(Drop-Columns-with-Missing-Values))

Hem eğitim (training) hem de doğrulama (validation) setleriyle çalıştığımız için, her iki DataFrame'de de aynı sütunları atmaya dikkat ediyoruz.

*# Get names of columns with missing values*

cols\_with\_missing = [col for col **in** X\_train.columns

if X\_train[col].isnull().any()]

*# Drop columns in training and validation data*

reduced\_X\_train = X\_train.drop(cols\_with\_missing, axis=1)

reduced\_X\_valid = X\_valid.drop(cols\_with\_missing, axis=1)

print("MAE from Approach 1 (Drop columns with missing values):")

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

MAE from Approach 1 (Drop columns with missing values):

183550.22137772635

### **Score from Approach 2 (Imputation)**[**¶**](https://www.kaggle.com/code/alexisbcook/missing-values#Score-from-Approach-2-(Imputation))

Şimdi, eksik değerleri her sütundaki ortalama değerle değiştirmek için **SimpleImputer**'ı kullanacağız.

Bu yöntem basit olmasına rağmen, ortalama değerle doldurma genellikle oldukça iyi sonuç verir (ancak bu, veri setine göre değişir). İstatistikçiler, eksik değerleri belirlemek için regresyon ataması (regression imputation) gibi daha karmaşık yollar deneseler de, bu karmaşık stratejiler, sonuçları gelişmiş makine öğrenmesi modellerine uyguladığınızda genellikle ek bir fayda sağlamaz.

from sklearn.impute import SimpleImputer

*# Imputation*

my\_imputer = SimpleImputer()

imputed\_X\_train = pd.DataFrame(my\_imputer.fit\_transform(X\_train))

imputed\_X\_valid = pd.DataFrame(my\_imputer.transform(X\_valid))

*# Imputation removed column names; put them back*

imputed\_X\_train.columns = X\_train.columns

imputed\_X\_valid.columns = X\_valid.columns

print("MAE from Approach 2 (Imputation):")

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

MAE from Approach 2 (Imputation):

178166.46269899711

Approach 2'nin MAE'sinin (Ortalama Mutlak Hata) Approach 1'den daha düşük olduğunu görüyoruz, bu nedenle bu veri setinde Approach 2 daha iyi performans gösterdi.

### Yaklaşım 3'ten Elde Edilen Skor (Atama Yönteminin Bir Uzantısı)

Şimdi, eksik değerleri atıyoruz, aynı zamanda hangi değerlerin atandığının kaydını da tutuyoruz.

*# Make copy to avoid changing original data (when imputing)*

X\_train\_plus = X\_train.copy()

X\_valid\_plus = X\_valid.copy()

*# Make new columns indicating what will be imputed*

for col **in** cols\_with\_missing:

X\_train\_plus[col + '\_was\_missing'] = X\_train\_plus[col].isnull()

X\_valid\_plus[col + '\_was\_missing'] = X\_valid\_plus[col].isnull()

*# Imputation*

my\_imputer = SimpleImputer()

imputed\_X\_train\_plus = pd.DataFrame(my\_imputer.fit\_transform(X\_train\_plus))

imputed\_X\_valid\_plus = pd.DataFrame(my\_imputer.transform(X\_valid\_plus))

*# Imputation removed column names; put them back*

imputed\_X\_train\_plus.columns = X\_train\_plus.columns

imputed\_X\_valid\_plus.columns = X\_valid\_plus.columns

print("MAE from Approach 3 (An Extension to Imputation):")

print(score\_dataset(imputed\_X\_train\_plus, imputed\_X\_valid\_plus, y\_train, y\_valid))

MAE from Approach 3 (An Extension to Imputation):

178927.503183954

Gördüğümüz gibi Yaklaşım 3, Yaklaşım 2'den biraz daha kötü performans gösterdi.

### **So, why did imputation perform better than dropping the columns?**[**¶**](https://www.kaggle.com/code/alexisbcook/missing-values#So,-why-did-imputation-perform-better-than-dropping-the-columns?)

Eğitim verileri 10864 satır ve 12 sütundan oluşuyor; üç sütunda eksik veriler bulunuyor. Her sütunda, girdilerin yarısından azı eksik. Dolayısıyla, sütunları silmek birçok yararlı bilgiyi ortadan kaldırıyor ve bu nedenle veri yüklemenin daha iyi performans göstermesi mantıklı.

*# Shape of training data (num\_rows, num\_columns)*

print(X\_train.shape)

*# Number of missing values in each column of training data*

missing\_val\_count\_by\_column = (X\_train.isnull().sum())

print(missing\_val\_count\_by\_column[missing\_val\_count\_by\_column > 0])

(10864, 12)

Car 49

BuildingArea 5156

YearBuilt 4307

dtype: int64

# Conclusion

Yaygın olduğu üzere, eksik değerlerin girilmesi (Yaklaşım 2 ve Yaklaşım 3'te), eksik değerlere sahip sütunları bıraktığımız zamana (Yaklaşım 1'de) kıyasla daha iyi sonuçlar verdi