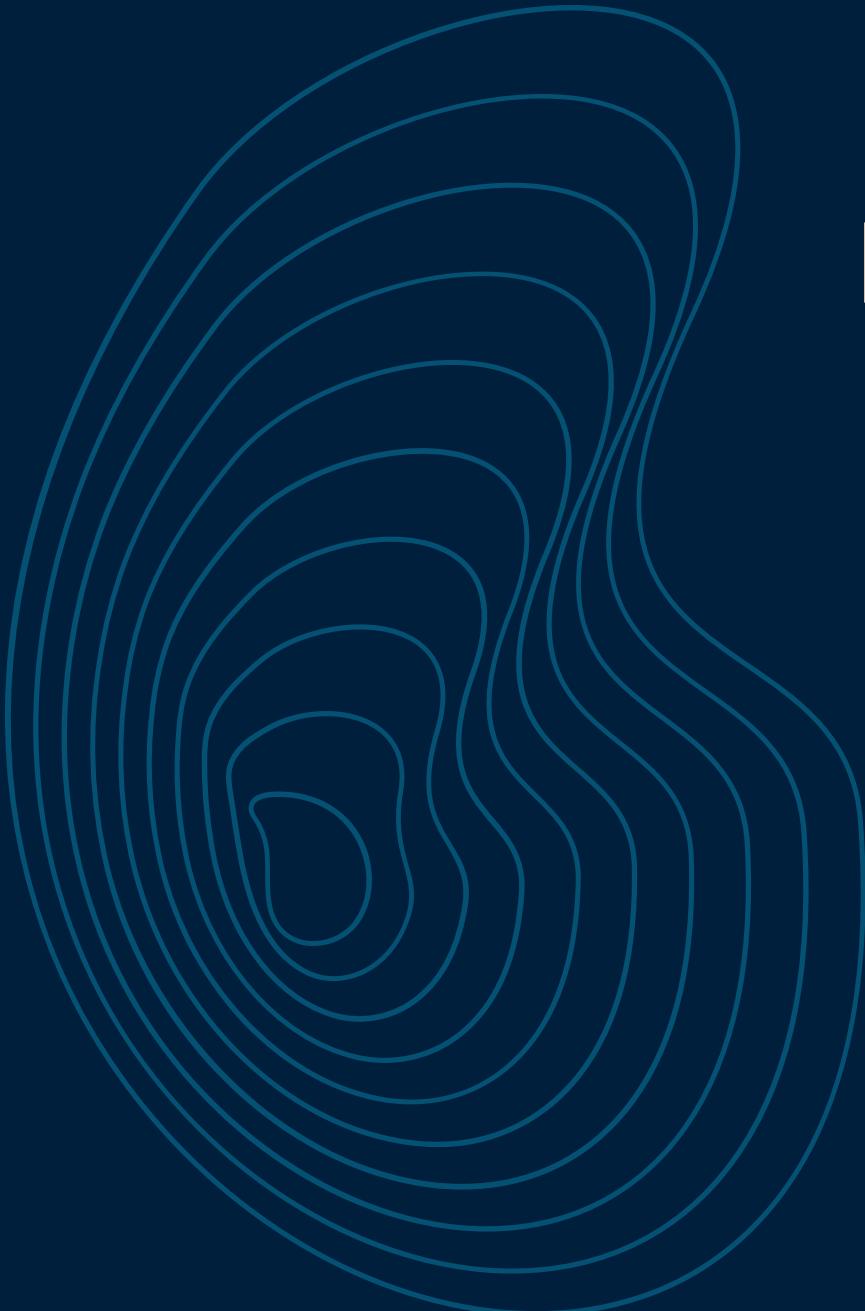


VERİ BİLİMİ KÜTÜPHANELERİ

NUMPY
PANDAS
MATPLOTLIB
SEABORN



MERVAN ÖZEKİNCİ



MERVAN ÖZEKİNCİ

VERİ BİLİMÇİ

[mervanozekinci](#)

ozekincimervan@gmail.com

Mannheim, Almanya

Programlama, Data Science ve Machine Learning gibi alanlara büyük ilgi duymaktayım. Almanya'da Information Technology alanında yüksek lisans yaparken, öğrendiğim bilgileri ücretsiz paylaşma yolculuğunda yaptığım bootcamplerden toparladığımız bilgilerin kitaplaştırılmış halidir.

Umarım bu kitap, ilgi duyduğum alanlara giriş yapmak isteyenlere yararlı olur.

KİTABIN HAZIRLANMASINDA KATKIDA BULUNANLAR



HALİL KOLATAN

Jr. VERİ BİLİMÇİ / FİZİK YL

hkolatan
 halilkolatan@pm.me
 Ankara, Türkiye



ELİF NUR KANAT

FİZİK YL ÖĞRENCİSİ

elifnurkanat
 elifnrkanat@gmail.com
 İstanbul, Türkiye



ÖMER RENÇBERELİ

TIP FAKÜLTESİ ÖĞRENCİSİ

omer-rencbereli
 omerrencbereli@gmail.com
 Ankara, Türkiye



HÜSEYİN ENES TANDOĞAN

BİLGİSAYAR MÜH. ÖĞRENCİSİ

hüseyin-enes-tandoğan
 henestandogan@gmail.com
 İstanbul, Türkiye



SELİN OLGUN

VERİ ANALİSTİ- GENETİK YL- MÜHENDİS

selinolgun
 selinolgun10@gmail.com
 Balıkesir, Türkiye



ESMANUR ÖRÜNKLÜ

ELEKTRONİK HAB. MÜH. ÖĞRENCİSİ

esmanurorunklu
 reyhan_esma_nur@hotmail.com
 İstanbul, Türkiye

İÇİNDEKİLER

| | |
|------------------------|-----------|
| Numpy | 1 |
| Pandas | 13 |
| Matplotlib..... | 79 |
| Seaborn | 84 |

Numpy

NumPy, Python programlama dili için büyük, çok boyutlu dizileri ve matrisleri destekleyen, bu diziler üzerinde çalışacak üst düzey matematiksel işlevler ekleyen bir kütüphanedir. Numpy hakkında detaylı bilgilere [buradan](#) ulaşabilirsiniz.

İlk olarak bu kütüphaneyi kullanmak için kütüphaneyi çağrırmamız gerekmektedir. Import komutu ile kütüphaneyi çağrılmaktayız. Burada `as np` ile Numpy kütüphanesine bir takma ad (alias) ekliyoruz. Bu işlem, komutları ya da komut dizilerine daha sonra kullanmak amacıyla bir kısayol atamamıza imkân sağlamaktadır.

```
import numpy as np
```

np.array(): Numpy'da basit bir şekilde dizin oluşturmamıza olanak sağlayan metottur. array fonksiyonu gönderdiğimiz listeyi bir numpy dizisi olarak döndürmektedir.

```
#Input:  
veri_liste=[1,2,3,4,5,6]  
array = np.array(veri_liste)  
print(array)
```

```
#Output:  
[1 2 3 4 5 6]
```

Eğer çok boyutlu dizi oluşturmak istersek liste içinde liste ekleyerek göndermemiz gerekmektedir.

```
#Input:  
a = np.array([[1, 2], [3, 4]])  
print(a)
```

```
...  
#Output:  
[[1 2]  
 [3 4]]
```

np.zeros(m,n): m x n boyutunda sıfırlardan oluşan bir dizi oluşturmaktadır.

```
...  
#Input:  
sifirlar=np.zeros(10)  
print(sifirlar)
```

```
...  
#Output:  
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

np.ones(m,n): m x n boyutunda birlerden oluşan bir dizi oluşturmaktadır. dtype=int komutu ile varsayılan olarak float olan değişken tipini integer olarak değiştiriyoruz.

```
...  
#Input:  
birler=np.ones(10, dtype=int)  
print(birler)
```

```
...  
#Output:  
[1 1 1 1 1 1 1 1 1 1]
```

np.array().dtype: Dizinin veri tipini göstermektedir. NumPy, Python'a göre çok daha çeşitli sayısal türleri desteklemektedir.

```
#Input:  
birler = np.ones(10,dtype=int)  
birler.dtype
```

```
#Output:  
dtype('int32')
```

np.array().shape: Dizinin satır ve sütun sayısını göstermektedir.

```
#Input:  
a = np.array([[1,2,3],[4,5,6]])  
print(a.shape)
```

```
#Output:  
(2, 3)
```

np.array().reshape: Diziyi girdiğimiz değerlerde yeniden boyutlandırmaktadır. Ama bunu yaparken dizinin yeniden matris oluşturması gerekmektedir. Örnekte görüldüğü üzere 2x3'lük dizi ile 3x2'lik dizi oluşturulmuştur.

```
● ● ●  
#Input:  
a = np.array([[1,2,3],[4,5,6]])  
print(a.reshape(3,2))
```

```
● ● ●  
#Output:  
[[1 2]  
 [3 4]  
 [5 6]]
```

np.array().ndim: Dizinin boyutunu göstermektedir.

```
● ● ●  
#Input:  
a = np.array([[1,2,3],[4,5,6]])  
print(a.ndim)
```

```
● ● ●  
#Output:  
2
```

np.array().size: Dizinin eleman sayısını göstermektedir.

```
...  
#Input:  
a = np.array([[1,2,3],[4,5,6]])  
print(a.size)
```

```
...  
#Output:  
6
```

np.full(): Girdiğimiz ikinci parametre ile mxn boyutunda bir dizi oluşturmaktadır. Gördüğümüz üzere 5 değeri ile 3x2'lik bir dizi döndürülmektedir.

```
...  
#Input:  
a = np.full((3,2), 5)  
print(a)
```

```
...  
#Output:  
[[5 5]  
 [5 5]  
 [5 5]]
```

Verilen aralığa göre array oluşturma:

np.arange(start, stop, step): Başlangıç değeri dahil, bitiş değeri dahil olmayacak şekilde atlama değerine göre array oluşturur. Standart olarak başlangıç değeri 0, atlama değerini 1 olarak kabul eder. "array1" adında 0'dan başlayıp 10'a kadar (10 dahil değil) ardışık sayılarından oluşan bir array oluşturulmuştur.

```
...  
#Input:  
array1 = np.arange(10)  
print(array1)
```

```
...  
#Output:  
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

np.linspace(start, stop, num): Başlangıç ve bitiş değeri dahil olacak şekilde verilen aralığı eşit parçalara bölünmüş bir array oluşturur. Standart olarak 50 elemandan oluşan bir array oluşturur. Çıktının bir seri olması önemlidir. Örneğin aşağıda, 10 ile 90 sayıları dahil olmak üzere 9 elemanlı ve eşit aralıklı bir seri oluşturulmuştur.

```
...  
#Input:  
array2 = np.linspace(10,90,9)  
print(array2)
```

```
...  
#Output:  
[10. 20. 30. 40. 50. 60. 70. 80. 90.]
```

Indexing ve Slicing:

Verilen arrayin belirli bir elemanına ulaşmak için kullanılır. “array[1,2]” şeklinde kullanılırken önce satır indexi, sonra sütun indexi verilen elemani döndürür.

```
# Input:  
array3 = np.array([[5, 7, 1, 0, 5],  
                  [5, 8, 9, 1, 2],  
                  [9, 3, 5, 5, 1]])  
array3[1,2] # satır indexi 1, sütun indexi 2 olan elemani döndürür.
```

```
# Output:  
9
```

Slicing işlemi yaparken verilen indexe göre kırpmaya yapılır. “array3[0:2,1:3]” şeklinde kullanılırken virgülden önceki kısım satır kırpması, virgülünden sonraki kısım sütun kırpması için esas alınır. Önemli nokta, “0:2”, 0’dan 2’ye kadar (2 dahil değil) olan indexleri ifade eder. Diğer önemli nokta “ : ” ifadesinden önce değer girilmemişse baştan itibaren, sonra değer girilmemişse sonuna kadar demektir. “array3[:,3]” 3.index sütununda yer alan tüm satırları sonuçlandırır.

```
# Input:  
array3[0:2,1:3]
```

```
# Output:  
array([[7, 1],  
       [8, 9]])
```

Verilen arrayde indexing yapıp yeni bir değere eşitemek, arrayin o noktasındaki değeri kalıcı olarak değiştirir. Verilen değerin, var olan arraydeki veri tipine dönüşümü mümkün değilse “ValueError” hatası alırız.

```
# Input:  
array3[0,3] = 72  
print(array3)
```

```
# Output:  
[5 7 1 72 5]  
[5 8 9 1 2]  
[9 3 5 5 1]]
```

Fancy index: array4'ün birden çok indexi tek bir liste sayesinde yazdırılır.

```
# Input:  
array4 = np.array([2,3,4,5,6,7,8,9])  
fancy = [0,2,5]  
array4[fancy]
```

```
# Output:  
array([2, 4, 7])
```

Array Manipulation:

numpy.reshape: Reshape fonksiyonu, verileri değiştirmeden diziye yeni bir şekil vermektedir. Aşağıdaki örnekte 6 veriye sahip a dizisi, reshape fonksiyonu kullanılarak 3 sütun, 2 satır olacak şekilde tekrar şekillendirilmiştir. Fonksiyonda yazılan ilk değer sütun sayısını, virgülden sonraki değer satır sayısını temsil etmektedir.

```
#Input:  
  
a = np.arange(6)  
print("Orjinal dizi: ",a)  
b = a.reshape(3,2)  
print("Reshape uygulanan dizi: ",b)
```



```
#Output:  
  
Orjinal dizi: [0 1 2 3 4 5]  
Reshape uygulanan dizi:  
[[0 1]  
 [2 3]  
 [4 5]]
```

numpy.transpose: Bu fonksiyon, verilen dizinin boyutunu transpoz etmektedir. Mümkün olduğu sürece satır verilerini sütun verilerine, sütun verilerini satır verilerine çevirmektedir. Aşağıdaki örnekte 3 sütun 4 satır olarak oluşturulan a dizisi, transpose fonksiyonu ile 4 sütun 3 satır'a çevrilmiştir.

```
#Input:  
  
a = np.arange(12).reshape(3,4)  
print("Orjinal dizi: ", a)  
b = np.transpose(a)  
print("Transpoz edilen dizi: ", b)
```

```
...  
#Output:  
  
Orjinal dizi:  
[[ 0 1 2 3]  
 [ 4 5 6 7]  
 [ 8 9 10 11]]  
  
Transpoz edilen dizi:  
[[ 0 4 8]  
 [ 1 5 9]  
 [ 2 6 10]  
 [ 3 7 11]]
```

numpy.split: Split fonksiyonu, diziyi belirtilen bir eksen boyunca alt dizilere bölmektedir. Split fonksiyonunda ilk terim olan **a** bölünecek diziyi, **3** ise kaç alt diziye bölünmesi gerektiğini ifade etmektedir.

```
...  
#Input:  
  
a = np.arange(9)  
print("İlk dizi:")  
print(a)  
print("Diziyi 3 eşit alt diziye böl:")  
b=np.split(a,3)  
print(b)
```

```
...  
#Output:  
  
İlk dizi:  
[0 1 2 3 4 5 6 7 8]  
Diziyi 3 eşit alt diziye böl:  
[array([0, 1, 2]), array([3, 4, 5]), array([6, 7, 8])]
```

numpy.append: Bu fonksiyon, bir dizinin sonuna elemanlar ekleyebilmektedir. Append fonksiyonunda, ilk terim olan **a** ekleme yapılacak diziyi, ikinci terim olan **[7,8,9]** ise eklenen elemanları temsil etmektedir.

```
#Input:  
  
a = np.array ([[1,2,3],[4,5,6]])  
  
print("İlk dizi:")  
print(a)  
print("Dizinin elemanlar eklenmiş hali:")  
print(np.append(a, [7,8,9]))
```

```
#Output:  
  
İlk dizi:  
[[1 2 3]  
 [4 5 6]]  
Dizinin elemanlar eklenmiş hali::  
[1 2 3 4 5 6 7 8 9]
```

numpy.unique(): Bu fonksiyon, verilen dizideki benzersiz elemanlardan oluşan bir dizi döndürmektedir.

```
#Input:  
  
a = np.array([3,2,6,2,7,5,6,8,2,9])  
  
print("İlk dizi:")  
print(a)  
print("İlk dizinin benzersiz elemanları:")  
  
u = np.unique(a)  
print(u)
```

```
● ● ●  
#Output:  
  
İlk dizi:  
[3 2 6 2 7 5 6 8 2 9]  
İlk dizinin benzersiz elemanları:  
[2 3 5 6 7 8 9]
```

Pandas

Bu kod satırı, Python programlama dilinde "pandas" kütüphanesini kullanabilmemizi sağlayacak şekilde bu kütüphaneyi içe aktarıyor. Pandas, veri analizi ve veri işleme için kullanılan bir kütüphanedir. Pandas hakkında detaylı bilgilere [buradan](#) ulaşabilirsiniz.

```
...  
#input  
  
import pandas as pd
```

NumPy, Python programlama dilinde kullanılan, bilimsel hesaplama ve veri analizi için geliştirilmiş bir kütüphanedir. Bu kütüphane, çok boyutlu dizi (array) yapıları ve matrisler üzerinde hızlı ve etkili işlemler yapmamızı sağlar.

```
...  
#input  
  
import numpy as np
```

pd.Series(): Bu kod bloğunda, öncelikle "pandas" kütüphanesi Python programlama dilinde kullanılmak üzere yükleniyor. Daha sonra, pd.Series() fonksiyonu kullanılarak bir "pandas series" veri yapısı oluşturuluyor. Bu veri yapısı, 1, 2, 3, 4, 5 ve 6 sayılarını içeren bir dizi oluşturur.

```
...  
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(s)
```

```
#Output  
  
0    1  
1    2  
2    3  
3    4  
4    5  
5    6  
dtype: int64
```

type(): "type(s)" kodu, "s" değişkeninin veri tipini (type) kontrol etmek için kullanılır

```
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(type(s))
```

```
#Output  
  
pandas.core.series.Series
```

index(): Index(), bir pandas veri yapısının indeksini (index) döndüren bir yöntemdir. Index, veri yapısının bir veya daha fazla boyutuna atanabilir ve verilerin daha kolay erişilmesine olanak tanır.

```
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(s.index)
```

```
#Output  
  
RangeIndex(start=0, stop=6, step=1)
```

dtype(): dtype, bir pandas DataFrame veya Serisi içindeki her bir sütunun veri türünü belirten bir özelliktir.

```
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(s.dtype)
```

```
#Output  
  
dtype('int64')
```

size(): size, bir pandas Serisi veya DataFrame'in boyutunu (eleman sayısını) döndüren bir yöntemdir.

```
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(s.size)
```

```
#Output  
  
6
```

ndim(): ndim, bir pandas veri yapısının boyutunu belirten bir özelliktir. Bu özellik, veri yapısının kaç boyutlu olduğunu gösterir.

```
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(s.ndim)
```

```
#Output  
  
1
```

values(): Pandas values(), bir pandas veri yapısının değerlerini numpy dizisi olarak döndüren bir yöntemdir.

```
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(s.values)
```



```
#Output  
  
array([1, 2, 3, 4, 5, 6])
```

head(): Seri'nin ilk n öğesini döndürür. Burada n, "3" olarak belirtilmiştir. Dolayısıyla, bu kod, Seri'nin ilk 3 öğesini (yani 1, 2 ve 3) ekrana yazdırır.

```
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(s.head(3))
```



```
#Output  
  
0    1  
1    2  
2    3  
dtype: int64
```

tail(): "tail" yöntemi kullanılarak Seri'nin son n öğesini döndürür. Burada n, "2" olarak belirtilmiştir. Dolayısıyla, bu kod Seri'nin son 2 öğesini (yani 5 ve 6) ekrana yazdırır.

```
#input:  
  
s=pd.Series([1,2,3,4,5,6])  
  
print(s.tail(2))
```

```
#Output  
  
4    5  
5    6  
dtype: int64
```

np.random.rand(): "np.random.rand(5,3)" ifadesi kullanılarak rastgele sayılarından oluşan 5 satır ve 3 sütuna sahip bir numpy dizisi oluşturur. Daha sonra, bu numpy dizisi "pd.DataFrame" fonksiyonu kullanılarak pandas DataFrame'e dönüştürülür.

```
#input:  
  
x=pd.DataFrame(np.random.rand(5,3))  
  
print(x)
```

| | 0 | 1 | 2 |
|---|----------|----------|----------|
| 0 | 0.560770 | 0.102927 | 0.294235 |
| 1 | 0.402128 | 0.056269 | 0.647045 |
| 2 | 0.327853 | 0.669263 | 0.787321 |
| 3 | 0.242096 | 0.356851 | 0.573277 |
| 4 | 0.281929 | 0.757531 | 0.693403 |

pd.DataFrame(columns=): Bu kod, 5 satır ve 8 sütuna sahip rastgele sayılarından oluşan bir pandas DataFrame'i oluşturur ve bu DataFrame'i ekrana yazdırır. Sütun etiketleri, "a" ile "h" arasındaki karakterlerden oluşan bir listeden alınır.

```
#input:
y=pd.DataFrame(np.random.rand(5,8),columns=list("abcdefgh"))
print(y)
```

| | a | b | c | d | e | f | g | h |
|---|----------|----------|----------|----------|----------|----------|----------|----------|
| 0 | 0.077645 | 0.588358 | 0.379389 | 0.572208 | 0.422597 | 0.538009 | 0.100827 | 0.719731 |
| 1 | 0.772909 | 0.221439 | 0.903906 | 0.778645 | 0.377371 | 0.597922 | 0.705775 | 0.188919 |
| 2 | 0.463816 | 0.372222 | 0.453312 | 0.377207 | 0.858552 | 0.053142 | 0.322908 | 0.661116 |
| 3 | 0.336969 | 0.937431 | 0.212808 | 0.914182 | 0.950072 | 0.565915 | 0.033120 | 0.755354 |
| 4 | 0.902122 | 0.364108 | 0.378421 | 0.721968 | 0.327254 | 0.786986 | 0.918249 | 0.166157 |

Seaborn

Seaborn, Python programlama dilinde veri görselleştirme için kullanılan açık kaynaklı bir kütüphanedir.

```
#input:  
  
import seaborn as sns
```

sns.load_dataset(): Bu kod, seaborn kütüphanesinin `load_dataset()` fonksiyonunu kullanarak "titanic" adlı bir veri setini yükler. Daha sonra, yüklenen veri seti `df` adlı bir DataFrame nesnesine atanır. `print(df)` komutu, bu DataFrame nesnesini ekrana yazdırarak verilerin gözlemlenmesini sağlar. Bu veri seti, Titanic gemisinin yolcularının bazı öznitelikleri (örneğin, yaş, cinsiyet, yolcu sınıfı vb.) ve hayatta kalma durumları hakkında bilgiler içermektedir.

```
#input:  
  
df=sns.load_dataset("titanic")  
  
print(df)
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|-----|----------|--------|--------|------|-------|-------|---------|----------|--------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 0 | 2 | male | 27.0 | 0 | 0 | 13.0000 | S | Second | man | True | NaN | Southampton | no | True |
| 887 | 1 | 1 | female | 19.0 | 0 | 0 | 30.0000 | S | First | woman | False | B | Southampton | yes | True |
| 888 | 0 | 3 | female | NaN | 1 | 2 | 23.4500 | S | Third | woman | False | NaN | Southampton | no | False |
| 889 | 1 | 1 | male | 26.0 | 0 | 0 | 30.0000 | C | First | man | True | C | Cherbourg | yes | True |
| 890 | 0 | 3 | male | 32.0 | 0 | 0 | 7.7500 | Q | Third | man | True | NaN | Queenstown | no | True |

891 rows x 15 columns

df.head(): df.head(10) fonksiyonu çağırılır ve df veri setinin ilk 10 gözlemini ekrana yazdırır. head() fonksiyonu, veri setinin belirtilen ilk gözlem sayısını döndürür ve varsayılan olarak ilk 5 gözlemi gösterir.

```
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.head(10))
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|--------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |
| 5 | 0 | 3 | male | NaN | 0 | 0 | 8.4583 | Q | Third | man | True | NaN | Queenstown | no | True |
| 6 | 0 | 1 | male | 54.0 | 0 | 0 | 51.8625 | S | First | man | True | E | Southampton | no | True |
| 7 | 0 | 3 | male | 2.0 | 3 | 1 | 21.0750 | S | Third | child | False | NaN | Southampton | no | False |
| 8 | 1 | 3 | female | 27.0 | 0 | 2 | 11.1333 | S | Third | woman | False | NaN | Southampton | yes | False |
| 9 | 1 | 2 | female | 14.0 | 1 | 0 | 30.0708 | C | Second | child | False | NaN | Cherbourg | yes | False |

df.tail(): df.tail(5) fonksiyonu çağırılır ve df veri setinin son 5 gözlemini ekrana yazdırır. tail() fonksiyonu, veri setinin belirtilen son gözlem sayısını döndürür ve varsayılan olarak son 5 gözlemi gösterir.

```
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.tail(5))
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|-----|----------|--------|--------|------|-------|-------|-------|----------|--------|-------|------------|------|-------------|-------|-------|
| 886 | 0 | 2 | male | 27.0 | 0 | 0 | 13.00 | S | Second | man | True | Nan | Southampton | no | True |
| 887 | 1 | 1 | female | 19.0 | 0 | 0 | 30.00 | S | First | woman | False | B | Southampton | yes | True |
| 888 | 0 | 3 | female | NaN | 1 | 2 | 23.45 | S | Third | woman | False | Nan | Southampton | no | False |
| 889 | 1 | 1 | male | 26.0 | 0 | 0 | 30.00 | C | First | man | True | C | Cherbourg | yes | True |
| 890 | 0 | 3 | male | 32.0 | 0 | 0 | 7.75 | Q | Third | man | True | Nan | Queenstown | no | True |

df.shape: df.shape fonksiyonu çağrırlır ve df veri setinin boyutları hesaplanır. Bu fonksiyon, df veri setinin kaç satır ve kaç sütundan olduğunu veren bir tuple döndürür.

```
...  
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.shape)
```

```
...  
#Output  
(891, 15)
```

df.info(): df.info() fonksiyonu çağırılır ve veri seti hakkında bilgi veren bir özet raporu ekrana yazdırır. Bu özet raporu, veri setindeki sütunların isimlerini, sütunlardaki eksik veri sayısını, veri tiplerini ve veri setindeki toplam veri sayısını içerir.

Bu kodun amacı, df veri seti hakkında genel bir bakış sağlamak ve verileri daha iyi anlamak için kullanılan bir araçtır. info() fonksiyonu veri setindeki sütunların sayısını, türlerini ve eksik verilerin olup olmadığını kolayca kontrol etmeyi sağlar.

```
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.info())
```

#Output:

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 15 columns):  
 #   Column      Non-Null Count  Dtype     
 ---  --          --          --          --  
 0   survived    891 non-null    int64    
 1   pclass      891 non-null    int64    
 2   sex         891 non-null    object    
 3   age         714 non-null    float64   
 4   sibsp       891 non-null    int64    
 5   parch       891 non-null    int64    
 6   fare         891 non-null    float64   
 7   embarked     889 non-null    object    
 8   class        891 non-null    category    
 9   who          891 non-null    object    
 10  adult_male   891 non-null    bool     
 11  deck         203 non-null    category    
 12  embark_town  889 non-null    object    
 13  alive         891 non-null    object    
 14  alone         891 non-null    bool     
 dtypes: bool(2), category(2), float64(2), int64(4), object(5)  
 memory usage: 80.7+ KB
```

df.columns: df.columns fonksiyonu çağırılır ve df veri setinin sütun adlarını döndürür. Sütun adları, veri setindeki her sütunun adını içeren bir liste şeklinde döndürülür.

```
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.columns)
```

```
#Output  
  
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',  
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',  
       'alive', 'alone'],  
      dtype='object')
```

df.index: df.index fonksiyonu çağırılır ve df veri setinin index (dizin) bilgilerini döndürür. Index, veri setindeki her gözlemin bir benzersiz kimlik numarasıdır. Index, veri setindeki her gözlemin bir sıra numarası olabileceği gibi, belirli bir özelliği de temsil edebilir.

```
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.index)
```

```
#Output  
  
RangeIndex(start=0, stop=6, step=1)
```

describe().T: Bu kod parçasının çıktısı, her sütun için aşağıdaki istatistiksel bilgileri içeren bir tablo olacaktır:

Count: sütunda aynı olan değerleri toplayarak adet bilgisi verir

Mean: sütundaki değerlerin aritmetik ortalaması

Std: sütundaki değerlerin standart sapması

Min: sütundaki en küçük değer

25%: sütundaki değerlerin %25'inden küçük olan en büyük değer

50%: sütundaki değerlerin medyanı

75%: sütundaki değerlerin %75'inden küçük olan en büyük değer

Max: sütundaki en büyük değer

Bu istatistiksel bilgiler, veri kümesindeki değişkenler hakkında genel bir fikir verir ve verilerin özelliklerini anlamak için yararlıdır.

```
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.describe().T)
```

#Output:

| | count | mean | std | min | 25% | 50% | 75% | max |
|-----------------|-------|-----------|-----------|------|---------|---------|------|----------|
| survived | 891.0 | 0.383838 | 0.486592 | 0.00 | 0.0000 | 0.0000 | 1.0 | 1.0000 |
| pclass | 891.0 | 2.308642 | 0.836071 | 1.00 | 2.0000 | 3.0000 | 3.0 | 3.0000 |
| age | 714.0 | 29.699118 | 14.526497 | 0.42 | 20.1250 | 28.0000 | 38.0 | 80.0000 |
| sibsp | 891.0 | 0.523008 | 1.102743 | 0.00 | 0.0000 | 0.0000 | 1.0 | 8.0000 |
| parch | 891.0 | 0.381594 | 0.806057 | 0.00 | 0.0000 | 0.0000 | 0.0 | 6.0000 |
| fare | 891.0 | 32.204208 | 49.693429 | 0.00 | 7.9104 | 14.4542 | 31.0 | 512.3292 |

df.isnull: isnull() metodunun çıktısı, True veya False değerleri içeren bir veri kümesi olacaktır. True, ilgili hücrede eksik bir değer olduğunu, False ise eksik bir değer olmadığını gösterir.

```
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.isnull)
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone | |
|-----|----------|--------|-------|-------|-------|-------|-------|----------|-------|-------|------------|-------|-------------|-------|-------|-------|
| 0 | False | False | False | False | False | False | False | False | False | False | False | False | True | False | False | False |
| 1 | False | False | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False | False | False | False | False | True | False | False | False |
| 3 | False | False | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False | False | False | False | False | True | False | False | False |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | False | False | False | False | False | False | False | False | False | False | False | False | True | False | False | False |
| 887 | False | False | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| 888 | False | False | False | True | False | False | False | False | False | False | False | False | True | False | False | False |
| 889 | False | False | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| 890 | False | False | False | False | False | False | False | False | False | False | False | False | True | False | False | False |

891 rows × 15 columns

df.isnull().values: isnull() metodunu kullanarak veri kümesindeki eksik değerleri (NaN) kontrol eder. values özelliği kullanılarak, eksik değerlerin sadece True veya False olduğu bir NumPy dizisi döndürülür.

```
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.isnull().values)
```

#Output:

```
array([[False, False, False, ..., False, False, False],  
       [False, False, False, ..., False, False, False],  
       [False, False, False, ..., False, False, False],  
       ...,  
       [False, False, False, ..., False, False, False],  
       [False, False, False, ..., False, False, False],  
       [False, False, False, ..., False, False, False]])
```

df.isnull().values.any(): print(df.isnull().values.any()) komutunun çıktısı, veri kümesinde en az bir eksik değer varsa True, hiç eksik değer yoksa False değeri olacaktır.

```
...  
  
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.isnull().values.any())
```

```
...  
  
#Output  
  
True
```

df.isnull().sum(): print(df.isnull().sum()) komutunun çıktısı, her bir sütundaki eksik değerlerin sayısını gösteren bir Pandas Serisi olacaktır. Bu Seri, her sütunun ismiyle eşleşen bir indeksle birlikte gelir.

```
...  
#Input:  
  
df=sns.load_dataset("titanic")  
  
print(df.isnull().sum())
```

#Output:

```
survived      0  
pclass        0  
sex           0  
age          177  
sibsp         0  
parch         0  
fare          0  
embarked      2  
class         0  
who           0  
adult_male    0  
deck          688  
embark_town   2  
alive         0  
alone         0  
dtype: int64
```

pd.set_option: pd.set_option("display.max_columns", None) fonksiyonu, ekranda görüntülenen sütun sayısını sınırlamaz. Yani, ekranda tüm sütunlar görüntülenecektir.

pd.set_option("display.width",500) ifadesi ise bir yapılandırma ayarıdır, bu yapılandırma ayarı ekrandaki bir satırda gösterilebilecek en fazla karakter sayısını 500 olarak ayarlar. Bu ayar, ekranda bir satırın fazla sayıda sütun içermesi durumunda, sütunlar arasındaki boşluğu artırarak okunabilirliği artırmak için yapılır.

```
#Input:  
  
pd.set_option("display.max_columns",None)  
  
pd.set_option("display.width",500)
```

check_df(): Bu Python kodu, seaborn kütüphanesinde yer alan "load_dataset" fonksyonunu kullanarak, "tips" adlı veri kümesini yükler ve "df2" adlı bir pandas DataFrame nesnesine atar.

```
def check_df(df):  
    print("*****head*****")  
    print(df.head())  
    print("*****tail*****")  
    print(df.tail())  
    print("*****info*****")  
    print(df.info())  
    print("*****columns*****")  
    print(df.columns)  
    print("*****null_values*****")  
    print(df.isnull().values.any())  
    print("*****shape*****")  
    print(df.shape)  
    print("*****describe*****")  
    print(df.describe().T)  
    print("*****null_sum*****")  
    print(df.isnull().sum())  
    print("*****Corr*****")  
    print(df.corr())  
    print("*****dtypes*****")  
    print(df.dtypes)
```

Daha sonra, yukarıda tanımladığımız "check_df" fonksiyonu çağrılır ve "df2" parametresi ile çağırılır. Bu, "df2" DataFrame'inin farklı özelliklerini kontrol etmek için "check_df" fonksiyonunu kullanarak analiz edileceği anlamına gelir.

"check_df" fonksiyonu, DataFrame'in farklı özelliklerini kontrol ederek sonuçlarını ekrana yazdırır. Bu, DataFrame'in yapısı hakkında genel bir fikir edinmek için kullanışlı bir yöntemdir ve olası veri kalitesi sorunlarını belirlemek için de kullanılabilir.

```
● ● ●  
#Input:  
  
df2=sns.load_dataset("tips")  
  
check_df(df2)
```

#Output:

```
*****head*****  
total_bill  tip    sex smoker  day   time  size  
0      16.99  1.01 Female   No Sun Dinner    2  
1      10.34  1.66  Male    No Sun Dinner    3  
2      21.01  3.50  Male    No Sun Dinner    3  
3      23.68  3.31  Male    No Sun Dinner    2  
4      24.59  3.61 Female   No Sun Dinner    4  
*****tail*****  
total_bill  tip    sex smoker  day   time  size  
239     29.03  5.92  Male    No Sat Dinner    3  
240     27.18  2.00 Female   Yes Sat Dinner    2  
241     22.67  2.00  Male    Yes Sat Dinner    2  
242     17.82  1.75  Male    No Sat Dinner    2  
243     18.78  3.00 Female   No Thur Dinner   2
```

```

*****info*****
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   total_bill  244 non-null    float64 
 1   tip         244 non-null    float64 
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64   
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
None

*****columns*****
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')
*****null_values*****
False

*****shape*****
(244, 7)

*****describe*****
   count      mean       std    min    25%    50%    75%   max
total_bill  244.0  19.785943  8.902412  3.07  13.3475  17.795  24.1275  50.81
tip        244.0   2.998279  1.383638  1.00  2.0000  2.900  3.5625  10.00
size       244.0   2.569672  0.951100  1.00  2.0000  2.000  3.0000  6.00

*****null_sum*****
total_bill  0
tip         0
sex         0
smoker     0
day         0
time        0
size        0
dtype: int64

*****Corr*****
   total_bill      tip      size
total_bill  1.000000  0.675734  0.598315
tip        0.675734  1.000000  0.489299
size       0.598315  0.489299  1.000000

*****dtypes*****
total_bill      float64
tip            float64
sex            category
smoker         category
day            category
time           category
size           int64
dtype: object

```

df.head(): Bu Python kodu, Seaborn kütüphanesindeki "load_dataset" fonksiyonunu kullanarak, "titanic" adlı veri kümesini yükler ve "df" adlı bir pandas DataFrame nesnesine atar.

"df.head()" kodu, "df" DataFrame'inin ilk 5 satırını ekrana yazdırma için kullanılır.

```
#Input  
  
df = sns.load_dataset("titanic")  
df.head()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |

df.corr(): "df.corr()" komutu, "df" DataFrame'inin değişkenleri arasındaki korelasyon matrisini hesaplar ve ekrana yazdırır. Korelasyon matrisi, iki veya daha fazla değişken arasındaki ilişkiyi ölçmek için kullanılır ve her bir değişken çifti için bir korelasyon katsayısı hesaplanır. Bu katsayılar, değişkenler arasındaki ilişkinin gücünü ve yönünü gösterir.

```
#Input  
  
df.corr()
```

#Output:

| | survived | pclass | age | sibsp | parch | fare | adult_male | alone |
|-------------------|-----------|-----------|-----------|-----------|-----------|-----------|------------|-----------|
| survived | 1.000000 | -0.338481 | -0.077221 | -0.035322 | 0.081629 | 0.257307 | -0.557080 | -0.203367 |
| pclass | -0.338481 | 1.000000 | -0.369226 | 0.083081 | 0.018443 | -0.549500 | 0.094035 | 0.135207 |
| age | -0.077221 | -0.369226 | 1.000000 | -0.308247 | -0.189119 | 0.096067 | 0.280328 | 0.198270 |
| sibsp | -0.035322 | 0.083081 | -0.308247 | 1.000000 | 0.414838 | 0.159651 | -0.253586 | -0.584471 |
| parch | 0.081629 | 0.018443 | -0.189119 | 0.414838 | 1.000000 | 0.216225 | -0.349943 | -0.583398 |
| fare | 0.257307 | -0.549500 | 0.096067 | 0.159651 | 0.216225 | 1.000000 | -0.182024 | -0.271832 |
| adult_male | -0.557080 | 0.094035 | 0.280328 | -0.253586 | -0.349943 | -0.182024 | 1.000000 | 0.404744 |
| alone | -0.203367 | 0.135207 | 0.198270 | -0.584471 | -0.583398 | -0.271832 | 0.404744 | 1.000000 |

df.columns: "df.columns" ifadesi, "df" DataFrame'inin sütunlarının adlarını listeler. Bu ifade, DataFrame'in yapısını anlamak ve ilgili sütunlara erişmek için kullanışlı bir yöntemdir.

#Input:

df.columns

#Output:

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

[col.upper() if col.startswith("s") else col.lower() for col in df.columns]:

[col.upper() if col.startswith("s") else col.lower() for col in df.columns] ifadesi bir liste üretecidir ve df.columns ifadesi "df" DataFrame'inin sütun adlarını listeler. Burada her sütun adı üzerinde dönlür ve sütun adının ilk harfi "s" ile başlıyorsa sütun adı büyük harflere dönüştürülür (col.upper()), aksi takdirde küçük harflere dönüştürülür (col.lower()).

```
...  
#Input:  
  
[col.upper() if col.startswith("s") else col.lower() for col in df.columns]
```

#Output:

```
['SURVIVED',  
 'pclass',  
 'SEX',  
 'age',  
 'SIBSP',  
 'parch',  
 'fare',  
 'embarked',  
 'class',  
 'who',  
 'adult_male',  
 'deck',  
 'embark_town',  
 'alive',  
 'alone']
```

df["sex"]: df["sex"] kodu, df DataFrame nesnesindeki sex sütununu (column) seçer. Burada ["sex"] kullanılarak sütuna erişim sağlanır. Bu kodu kullanarak df DataFrame'inin sadece sex sütununu seçebilir ve bu sütundaki verileri inceleyebiliriz.

```
...  
#Input:  
  
df["sex"]
```

#Output:

```
0      male
1    female
2    female
3    female
4      male
...
886    male
887  female
888  female
889    male
890    male
Name: sex, Length: 891, dtype: object
```

df["sex"].value_counts(): df["sex"].value_counts() kodu, df DataFrame nesnesindeki sex sütununda yer alan her bir farklı değerin kaç kez tekrarlandığını hesaplar. Yani, bu kod, sütunda yer alan her bir cinsiyet değerinin toplam sayısını sayar ve her bir değerin sayısını ekrana yazdırır.

```
#Input:
df["sex"].value_counts()
```

```
#Output
male    577
female   314
Name: sex, dtype: int64
```

`df["embarked"].value_counts()`: df["embarked"].value_counts() ifadesi, "titanic" veri kümesindeki "embarked" sütunundaki her bir değerin sayısını hesaplar. Yani, "embarked" sütununda kaç tane "C", "S" ve "Q" olduğunu sayar ve bunları bir Seri nesnesi olarak döndürür. Sonuç, sütundaki farklı değerlerin sayısını verir.

```
...  
#Input:  
  
df["embarked"].value_counts()  
  
...  
#Output  
  
S      644  
C      168  
Q       77  
Name: embarked, dtype: int64
```

[print(col+"\n",df[col].value_counts()):] [print(col+"\n",df[col].value_counts()) for col in df.columns] ifadesi, bir liste üreteci kullanarak, veri kümesindeki her sütunun adını alır ve print(col + "\n", df[col].value_counts()) ifadesini kullanarak sütundaki farklı değerlerin sayısını yazdırır. print() fonksiyonu, sütun adını ve value_counts() sonucunu yazdırmak için kullanılır.

```
...  
#Input:  
  
[print(col+"\n",df[col].value_counts()) for col in df.columns]
```

#Output:

```
survived  
0    549  
1    342  
Name: survived, dtype: int64  
pclass  
3    491  
1    216  
2    184  
Name: pclass, dtype: int64  
sex  
male     577  
female    314  
Name: sex, dtype: int64  
age  
24.00    30  
22.00    27  
18.00    26  
19.00    25  
28.00    25  
..  
36.50     1  
55.50     1  
0.92      1  
23.50     1  
74.00     1
```

df.sex.value_counts(): Bu kod, "titanic" veri kümesindeki sex sütunundaki farklı değerlerin sayısını yazdırma için value_counts() fonksiyonunu kullanır. value_counts() fonksiyonu, bir serinin her bir değerinin kaç kez tekrarlandığını sayar ve sonuçları sıkılık sırasına göre verir.

```
...  
#Input:  
  
df.sex.value_counts()
```

```
...  
#Output  
  
male      577  
female    314  
Name: sex, dtype: int64
```

df.age.sum(): Bu kod, "titanic" veri kümesindeki age sütununun tüm değerlerinin toplamını hesaplamak için sum() fonksiyonunu kullanır. sum() fonksiyonu, bir serinin tüm değerlerinin toplamını hesaplar.

```
...  
#Input:  
  
df.age.sum()
```

```
...  
#Output  
  
21205.17
```

“[:10]” Dilimleme İşareti: Bir listeden veya bir diziden bir aralık seçmek için kullanılır. Kod, df DataFrame'in tüm sütunlarını seçer ve ilk 10 satırı döndürür. Bu, DataFrame üzerinde ".head(10)" yöntemini kullanmakla aynı sonucu verir. Bu durumda, 0'dan 9'a (dahil) olan dizin etiketleriyle satırları seçer.

```
...  
#Input:  
df[:10]
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|--------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |
| 5 | 0 | 3 | male | NaN | 0 | 0 | 8.4583 | Q | Third | man | True | NaN | Queenstown | no | True |
| 6 | 0 | 1 | male | 54.0 | 0 | 0 | 51.8625 | S | First | man | True | E | Southampton | no | True |
| 7 | 0 | 3 | male | 2.0 | 3 | 1 | 21.0750 | S | Third | child | False | NaN | Southampton | no | False |
| 8 | 1 | 3 | female | 27.0 | 0 | 2 | 11.1333 | S | Third | woman | False | NaN | Southampton | yes | False |
| 9 | 1 | 2 | female | 14.0 | 1 | 0 | 30.0708 | C | Second | child | False | NaN | Cherbourg | yes | False |

drop(): "drop()" yöntemi, belirtilen satır veya sütunları DataFrame'den çıkarmak için kullanılır. Burada, "2" indeks numaralı satır (yani 3. satır), "axis=0" parametresiyle birlikte belirtilerek, satır bazında silinir.

"axis=0" parametresi, DataFrame'deki temsil edildiği eksenin belirtir.

".head()" yöntemi, değiştirilmiş DataFrame'in ilk 5 satırını döndürür.

```
...  
#Input:  
df.drop(2, axis=0).head()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |
| 5 | 0 | 3 | male | NaN | 0 | 0 | 8.4583 | Q | Third | man | True | NaN | Queenstown | no | True |

Örnek 2: ".drop()" yönteminin, belirtilen satır veya sütunları DataFrame'den çıkarmak için kullanıldığını öğrendik. Burada ise, "sex" sütunu "axis=1" parametresiyle birlikte belirtilerek, sütun bazında silinir.

"axis=1" parametresi, DataFrame'deki sütunların temsil edildiği ekseni belirtir.

".head(3)" ifadesi, "sex" sütunu çıkarılmış ve değiştirilmiş DataFrame'in ilk 3 satırını döndürür.

```
...  
#Input:  
df.drop("sex",axis=1).head(3)
```

#Output:

| | survived | pclass | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |

3. Örnek: Burada, "sex", "age", "who" ve "fare" sütunları liste olarak "axis=1" parametresiyle birlikte belirtilerek, sütun bazında silinir.

Yani birden fazla sütun ya da satır aynı anda silinebilir.

```
...  
#Input:  
df.drop(["sex","age","who","fare"],axis=1).head(3)
```

#Output:

| | survived | pclass | sibsp | parch | embarked | class | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|-------|-------|----------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | 1 | 0 | S | Third | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | 1 | 0 | C | First | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | 0 | 0 | S | Third | False | NaN | Southampton | yes | True |

4. Örnek: Yukarıdaki örnekte birden fazla sütun silmiştık. Bu örnekte, "indexes" listesinde belirtilen indexler(satırlar) "axis=0" parametresiyle birlikte belirtilerek, satır bazında silinir.

```
#Input:  
indexes=[1,2,3,5,7,9]  
df.drop(indexes, axis=0).head(5)
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|----|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |
| 6 | 0 | 1 | male | 54.0 | 0 | 0 | 51.8625 | S | First | man | True | E | Southampton | no | True |
| 8 | 1 | 3 | female | 27.0 | 0 | 2 | 11.1333 | S | Third | woman | False | NaN | Southampton | yes | False |
| 10 | 1 | 3 | female | 4.0 | 1 | 1 | 16.7000 | S | Third | child | False | G | Southampton | yes | False |

.index 1. Özellik: ".index" özelliği, DataFrame'in satır indekslerine erişmek için kullanılır.

Aşağıdaki örnek ise, "df pandas DataFrame'inin satır indekslerinin "age" sütunundaki değerlere eşitlenmesi" işlemidir.

Burada, "df" DataFrame'inin satır indeksleri "age" sütunundaki değerlere eşitlenir.

"df['age']" ifadesi, "age" sütununun serisini döndürür ve bu seri, ".index" özelliği kullanılarak df'in satır indeksleri olarak ayarlanır.

```
#Input:  
df.index=df["age"]  
df.head()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|------------|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| age | | | | | | | | | | | | | | | |
| 22.0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 38.0 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 26.0 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 35.0 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 35.0 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |

2. yöntem:

Aşağıda kodu çalıştırıldıkten sonra indekslerin değiştiği sonucuna ulaşırız.

Bu yöntem, "age" sütununu DataFrame'in indeksi olarak kalıcı olarak ayarlar. Eğer indekslerin kalıcı olarak değiştirilmesini istemiyorsanız, ikinci bir DataFrame tanımlayıp işlemleri onda yaparak orijinal indexler korunmuş olur.

```
#Input:  
df.head()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|------------|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| age | | | | | | | | | | | | | | | |
| 22.0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 38.0 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 26.0 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 35.0 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 35.0 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |

.index 2. Özellik: Satır indeksleri, DataFrame'de yer alan verileri belirli bir sırada organize etmek ve erişmek için kullanışlıdır.

".index" özelliği , df Pandas DataFrame nesnesinin satır indekslerine erişmek için kullanılır.

```
#Input:  
df.index
```

#Output:

```
Float64Index([22.0, 38.0, 26.0, 35.0, 35.0, nan, 54.0, 2.0, 27.0, 14.0,
               ...
               33.0, 22.0, 28.0, 25.0, 39.0, 27.0, 19.0, nan, 26.0, 32.0],
               dtype='float64', name='age', length=891)
```

inplace=True parametresi: Bu ifadeyi açıklamadan önce aşağıdaki örneğe bakmanızı istiyorum.

df.drop("age", axis=1) komutu, "age" sütununu geçici olarak kaldırıracak ve yeni bir DataFrame döndürecektil. Bu yeni DataFrame, orijinal DataFrame'den farklı bir nesne olarak kalacaktır. Biz bu farklı nesneyi df2 ifadesine eftledik.

Bu nedenle, orijinal DataFrame'deki "age" sütunu kalıcı olarak silinmez ve DataFrame tekrar çağrıldığında veya başka bir değişkene atanmadığı sürece hala mevcut olacaktır.

```
#Input:  
df2=df.drop("age",axis=1)  
df2.head(3)
```

#Output:

| | survived | pclass | sex | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|------|----------|--------|--------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| age | | | | | | | | | | | | | | |
| 22.0 | 0 | 3 | male | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 38.0 | 1 | 1 | female | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 26.0 | 1 | 3 | female | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |

inplace=True parametresi, DataFrame üzerinde gerçekleştirilen değişikliklerin kalıcı olarak uygulanması için kullanılır. Bu, verilerin kopyalanmasına gerek kalmadan DataFrame'in doğrudan değiştirilmesini sağlar ve işlem zamanını ve bellek kullanımını azaltır.

Eğer **inplace=True** kullanılmamasaydı, "age" sütunu yalnızca bu kod satırda kaldırılır ve asıl DataFrame'de değişiklik olmazdı.

```
#Input:  
df.drop("age",axis=1,inplace=True)  
df.head(3)
```

#Output:

| | survived | pclass | sex | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|------|----------|--------|--------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| age | | | | | | | | | | | | | | |
| 22.0 | 0 | 3 | male | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 38.0 | 1 | 1 | female | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 26.0 | 1 | 3 | female | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |

.reset_index(): Bu yöntem, bir DataFrame'deki mevcut indexi sıfırlar ve yeni bir sayısal index oluşturur.

inplace=True parametresi kullanıldığı için, bu değişiklik DataFrame'e kalıcı olarak uygulanır.

Yeni indeksler, sıfırdan başlayarak ardışık sayılarından oluşur. Önceki indekslerdeki değerler, yeni bir sütun olarak DataFrame'e eklenir ve "index" adıyla etiketlenir. Bu örnekte, "age" sütunu önceki indekslere eşit olduğu için, "age" sütunu da "index" sütunu altında eklenir.

```
#Input:  
df.reset_index(inplace=True)  
df.head()
```

#Output:

| | age | survived | pclass | sex | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|------|----------|--------|--------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 22.0 | 0 | 3 | male | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 38.0 | 1 | 1 | female | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 26.0 | 1 | 3 | female | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 35.0 | 1 | 1 | female | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 35.0 | 0 | 3 | male | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |

set_option metodu ve display.max_columns parametresi: Tüm sütunları df DataFrame'de görüntülemek bazı işlemler gereklili bunun için ilgili kütüphaneler çağırıldıkten sonra adımlara başlanır.

set_option metodu, pandas kütüphanesindeki **options** modülüne ait bir metottur ve birçok yapılandırma seçeneğini ayarlamamıza olanak tanır. Bu seçenekler, DataFrame'in görüntüleme, ayrıştırma, birleştirme vb. işlemleri için kullanılabilir ve DataFrame'i özelleştirmemizi sağlar. Metodun kullanımı, seçenek adı(**display.max_columns**) ve değerini(**None**) alarak yapılır.

display.max_columns seçeneği, DataFrame'in maksimum sütun sayısını belirler. Varsayılan olarak, DataFrame'in sütunlarından bazıları görüntülenmeyebilir ve ... işaretini kullanılır. Bu seçeneği **None** olarak ayarlamak, DataFrame'in tüm sütunlarının görüntülenmesini sağlar.

Bu örnekte, **pd.set_option("display.max_columns",None)** kullanılarak, DataFrame'in tüm sütunlarının görüntülenmesi sağlandı.

Sonraki satırda, **sns.load_dataset("titanic")** fonksiyonu çağrılarak, "titanic" veri seti **seaborn** kütüphanesi tarafından yüklenir ve bir **DataFrame** olarak **df** değişkenine atılır.

```
#Input:  
import pandas as pd  
import seaborn as sns  
pd.set_option("display.max_columns",None)  
df=sns.load_dataset("titanic")  
df.head()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |

in metodu: **in** kelimesi, bir ögenin bir liste, dizi veya başka bir koleksiyon içinde olup olmadığını sorgulamak için kullanılır.

Bu nedenle "age" in df ifadesi, "age" sütununun df DataFrame'i içinde olup olmadığını sorgular ve sonuç olarak **True** veya **False** döndürür.

```
#Input:  
"age" in df
```

```
...  
#Output:  
True
```

DataFrame'de sütun seçimi: Tek bir köşeli parantez kullanarak seçilen sütun, bir Series olarak döndürülürken, çift köşeli parantez kullanarak seçilen sütun bir DataFrame olarak döndürülür.

1. Örnek: df["age"] ifadesi, Pandas DataFrame'inde sadece "age" sütununu seçmek için kullanılır.

Tek köşeli parantez içinde bir string ifade kullanılması, seçilen sütunun Pandas Series olarak döndürülmesini sağlar.

Tipini sorgulayarak doğruluğunu görelim.

```
...  
#Input:  
type(df["age"])
```

```
...  
#Output:  
pandas.core.series.Series
```

2. Örnek: Tek köşeli parantez içinde çift köşeli parantez kullanılması, seçilen sütunun DataFrame olarak döndürülmesini sağlar. Yani, df[["age"]] ifadesi, "age" sütununu tek sütunlu bir DataFrame olarak döndürür.

```
...  
#Input:  
type(df[["age"]])
```

```
#Output:  
pandas.core.frame.DataFrame
```

3. Örnek: Aşağıdaki kod satırı, "age", "sex" ve "class" sütunlarını içeren bir DataFrame döndürür. Yani, bu ifade verilerin sadece bu üç sütununu içerecek şekilde filtreler. Köşeli parantez içindeki iki adet çift köşeli parantez, birden fazla sütunu seçmek için kullanılır ve sonuç olarak bir DataFrame döndürür.

```
#Input:  
df[["age", "sex", "class"]].head()
```

#Output:

| | age | sex | class |
|---|------|--------|-------|
| 0 | 22.0 | male | Third |
| 1 | 38.0 | female | First |
| 2 | 26.0 | female | Third |
| 3 | 35.0 | female | First |
| 4 | 35.0 | male | Third |

4. Örnek: col_names listesi içindeki "sex", "pclass" ve "alive" sütunlarını içeren DataFrame, df[col_names].head() kullanılarak oluşturulmuştur. df[col_names] ifadesi, belirtilen sütunları içeren DataFrame'i döndürür ve .head() ile sadece ilk beş satır görüntülenir.

```
#Input:  
col_names=["sex", "pclass", "alive"]  
df[col_names].head()
```

#Output:

| | sex | pclass | alive |
|---|--------|--------|-------|
| 0 | male | 3 | no |
| 1 | female | 1 | yes |
| 2 | female | 3 | yes |
| 3 | female | 1 | yes |
| 4 | male | 3 | no |

Yeni bir sütun ekleme: Aşağıdaki kodu çalıştırıldıkten sonra "age" sütunundaki her veri 3 ile çarpılarak "batman" adında yeni bir sütun oluşturulur ve bu sütun DataFrame'e eklenir.

.head() ile kontrol edelim.

```
...  
#Input:  
df["batman"] = df["age"] * 3  
df.head()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone | batman |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|--------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False | 66.0 |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False | 114.0 |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True | 78.0 |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False | 105.0 |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True | 105.0 |

drop() sütun silme: axis = 1 yaparak ve kalıcı silme işlemini inplace=True yaparak batman sütununu silelim. Ve önceki değiştirilmemiş DataFrame'i tekrar görüntüleyelim.

```
...  
#Input:  
df.drop(["batman"], axis=1, inplace=True)
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |

loc: Burada **loc** indeksleme yöntemi kullanılır. **loc** indeksleme yöntemi, satır ve sütunların etiketlerine dayalı olarak bir DataFrame'in belli bir kısmını seçmek için kullanılır. İlk parametre, satırların etiketlerini belirtirken, ikinci parametre sütunların etiketlerini belirtir.

Örneğin, `:4` ifadesi, ilk 5 satırını, "age" ve "class" ifadeleri, "age" ve "class" sütunlarını seçmek için kullanılır.

The screenshot shows a Jupyter Notebook cell. At the top, there are three gray dots indicating previous cells. Below them, the input code is shown in a light gray box:
#Input:
df.loc[:4, ["age", "class"]]

#Output:

| | age | class |
|---|------|-------|
| 0 | 22.0 | Third |
| 1 | 38.0 | First |
| 2 | 26.0 | Third |
| 3 | 35.0 | First |
| 4 | 35.0 | Third |

df.columns: .loc indeksleme yöntemi kullanılarak seçim yapılırken, sütun seçimi için **df.columns.str.contains("s")** ifadesi kullanılır. Bu ifade, DataFrame sütunlarının isimleri üzerinde string işlemleri yapmak için kullanılan bir pandas özelliğidir.

str.contains() yöntemi, bir string içinde belirtilen bir alt stringi (burada "s") içerip içermediğini kontrol eder ve sonucu bir boolean Series olarak döndürür. Sonuçta, sadece "s" karakterini içeren sütunlar **True** olarak işaretlenir ve bu sütunlar **loc** yöntemi kullanılarak seçilir.

```
#Input:  
df.loc[:, df.columns.str.contains("s")]
```

#Output:

| | survived | pclass | sex | sibsp | class |
|-----|----------|--------|--------|-------|--------|
| 0 | 0 | 3 | male | 1 | Third |
| 1 | 1 | 1 | female | 1 | First |
| 2 | 1 | 3 | female | 0 | Third |
| 3 | 1 | 1 | female | 1 | First |
| 4 | 0 | 3 | male | 0 | Third |
| ... | ... | ... | ... | ... | ... |
| 886 | 0 | 2 | male | 0 | Second |
| 887 | 1 | 1 | female | 0 | First |
| 888 | 0 | 3 | female | 1 | Third |
| 889 | 1 | 1 | male | 0 | First |
| 890 | 0 | 3 | male | 0 | Third |

891 rows × 5 columns

~df.columns: Aşağıdaki kod ifadesi, df DataFrame'inde sadece sütun isimlerinde "s" karakterini içermeyen sütunları seçer.

Burada ~ operatörü, df.columns.str.contains("s") ifadesinin sonucunu tersine çevirir. Yani, "s" karakterini içeren sütunlar True olurken, ~ operatörü bunları False olarak işaretler. Bu şekilde, yalnızca "s" karakterini içermeyen sütunlar seçilir.

.loc indeksleme yöntemi kullanarak sütun seçimi yapılırken, sütun etiketlerinin string işlemleriyle seçilmesi için str.contains() yöntemi kullanılır. Sonuçta, yalnızca "s" karakterini içermeyen sütunlar seçilir.

```
#Input:  
df.loc[:, ~df.columns.str.contains("s")]
```

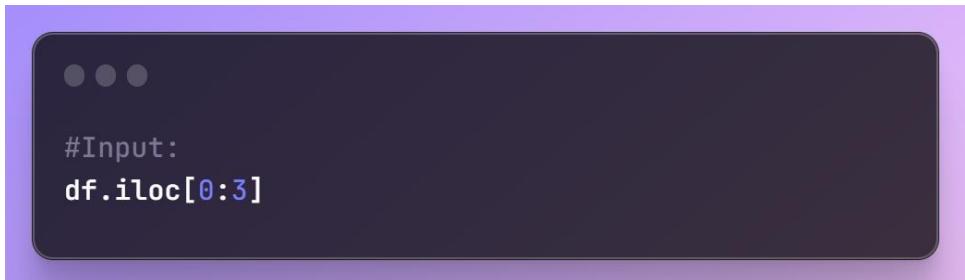
#Output:

| | age | parch | fare | embarked | who | adult_male | deck | embark_town | alive | alone |
|-----|------|-------|---------|----------|-------|------------|------|-------------|-------|-------|
| 0 | 22.0 | 0 | 7.2500 | S | man | True | NaN | Southampton | no | False |
| 1 | 38.0 | 0 | 71.2833 | C | woman | False | C | Cherbourg | yes | False |
| 2 | 26.0 | 0 | 7.9250 | S | woman | False | NaN | Southampton | yes | True |
| 3 | 35.0 | 0 | 53.1000 | S | woman | False | C | Southampton | yes | False |
| 4 | 35.0 | 0 | 8.0500 | S | man | True | NaN | Southampton | no | True |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 27.0 | 0 | 13.0000 | S | man | True | NaN | Southampton | no | True |
| 887 | 19.0 | 0 | 30.0000 | S | woman | False | B | Southampton | yes | True |
| 888 | NaN | 2 | 23.4500 | S | woman | False | NaN | Southampton | no | False |
| 889 | 26.0 | 0 | 30.0000 | C | man | True | C | Cherbourg | yes | True |
| 890 | 32.0 | 0 | 7.7500 | Q | man | True | NaN | Queenstown | no | True |

891 rows × 10 columns

iloc: loc indeksleme yöntemi, sütun veya satırların isimlerini kullanarak seçim yaparken, iloc indeksleme yöntemi, sadece sütun ve satırların pozisyonlarına göre seçim yapar. Yani, iloc kullanırken, sütun veya satır etiketlerini değil, sadece konumlarını belirtmek gereklidir.

Yani kısaca iloc indeksleme yöntemi kullanılarak satır seçimi yapılır.



#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |

2. örneğ: df.loc[0:3] ifadesi, df DataFrame'indeki satırların etiketlerine göre seçim yapmaya çalışır. Eğer df DataFrame'inde satırların etiketleri sayısal bir dizi ise (mesela 0'dan başlayarak artan bir sayısal dizi), bu durumda pandas, etiketleri otomatik olarak algılayabilir ve seçim yapabilir. Ancak eğer satır etiketleri sayısal bir dizi olmayan bir veri tipi ise (mesela string), bu durumda df.loc[0:3] ifadesi hata verecektir.

loc yöntemi, etiketlere göre seçim yapar ve seçim kapsamına dahil edilen son etiket buna dahil edilir.

```
...  
#Input:  
df.loc[0:3]
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |

Sütunlar üzerinde operatör işlemleri:

1.Örnek:

Aşağıdaki kod satırı, "age" sütununda 72'den büyük olan tüm satırları seçer ve bu satırları içeren bir DataFrame oluşturur.

```
...  
#Input:  
df[df["age"]>72]
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|-----|----------|--------|------|------|-------|-------|--------|----------|-------|-----|------------|------|-------------|-------|-------|
| 630 | 1 | 1 | male | 80.0 | 0 | 0 | 30.000 | S | First | man | True | A | Southampton | yes | True |
| 851 | 0 | 3 | male | 74.0 | 0 | 0 | 7.775 | S | Third | man | True | NaN | Southampton | no | True |

2. Örnek: Önceki örnekte olduğu gibi, `df[df["age"]>72]` ifadesi, "age" sütununda 72'den büyük olan tüm satırları içeren bir DataFrame oluşturur. Sonra `count()` yöntemi bu DataFrame'deki satır sayısını hesaplar.

```
#Input:  
df[df["age"]>72].count()
```

#Output:

```
survived      2  
pclass        2  
sex           2  
age           2  
sibsp         2  
parch         2  
fare           2  
embarked      2  
class          2  
who            2  
adult_male     2  
deck           1  
embark_town   2  
alive          2  
alone          2  
dtype: int64
```

3. Örnek: Bu kod satırı, "age" sütununda 72'den büyük olan satırların "survived" sütununda kaç dolu değer olduğunu döndürür. Dolu değer Nan dışındaki değerlerdir.

```
#Input:  
df[df["age"]>72]["survived"].count()
```

```
#Output:  
2
```

4. Örnek: Bu kod satırı, "age" sütunundaki değeri 50'den büyük olan tüm satırları seçer ve bu satırların sadece "age" ve "class" sütunlarını içeren yeni bir DataFrame oluşturur. Yani, bu kod satırı bize "age" sütununda 50'den büyük olan kişilerin "age" ve "class" sütunlarına ait verilerini gösterir.

```
#Input:  
df.loc[df["age"]>50, ["age", "class"]]
```

#Output:

| | age | class |
|----|------|--------|
| 6 | 54.0 | First |
| 11 | 58.0 | First |
| 15 | 55.0 | Second |
| 33 | 66.0 | Second |
| 54 | 65.0 | First |

5. Örnek: Bu kod, "age" sütunundaki değeri 50'den büyük ve "sex" sütunundaki değeri "male" olan tüm satırları seçer ve bu satırların tüm sütunlarını içeren yeni bir DataFrame oluşturur. Yani, bu kod satırı bize "age" sütununda 50'den büyük ve "sex" sütununda erkek olan kişilerin tüm verilerini gösterir.

& işaretti, iki boolean ifadeyi birleştirmek için kullanılır ve "ve" mantıksal işlemini ifade eder. Bu durumda, iki ifade de True olduğunda, sonuç True olur.

Ve bu kod satırında, iki ifadenin de True olduğu satırlar seçilir.

```
#Input:  
df[(df["age"]>50) & (df["sex"]=="male")].head()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|----|----------|--------|------|------|-------|-------|---------|----------|--------|-----|------------|------|-------------|-------|-------|
| 6 | 0 | 1 | male | 54.0 | 0 | 0 | 51.8625 | S | First | man | True | E | Southampton | no | True |
| 33 | 0 | 2 | male | 66.0 | 0 | 0 | 10.5000 | S | Second | man | True | Nan | Southampton | no | True |
| 54 | 0 | 1 | male | 65.0 | 0 | 1 | 61.9792 | C | First | man | True | B | Cherbourg | no | False |
| 94 | 0 | 3 | male | 59.0 | 0 | 0 | 7.2500 | S | Third | man | True | Nan | Southampton | no | True |
| 96 | 0 | 1 | male | 71.0 | 0 | 0 | 34.6542 | C | First | man | True | A | Cherbourg | no | True |

.sum: Bu kod, "age" sütunundaki tüm değerlerin toplamını hesaplar ve sonucu döndürür. Yani, bu kod satırı bize DataFrame'deki tüm kişilerin yaşlarının toplamını verir.

```
#Input:  
df.age.sum()
```

```
#Output:  
21205.17
```

.mean(): Bu aşağıdaki kod, "age" sütunundaki tüm değerlerin ortalamasını hesaplar ve sonucu döndürür.

```
#Input:  
df.age.mean()
```

```
#Output:  
29.69911764705882
```

.min(): Aşağıdaki kod, "age" sütunundaki en küçük değeri (minimum değeri) döndürür. Yani, bu kod satırı bize DataFrame'deki tüm kişilerin yaşları arasında en küçük olanını verir.

```
#Input:  
df.age.min()
```

```
#Output:  
0.42
```

.max(): Bu kod, "age" sütunundaki en büyük değeri (maksimum değeri) döndürür. Yani, bu kod satırı bize DataFrame'deki tüm kişilerin yaşları arasında en büyük olanını verir.

```
#Input:  
df.age.max()
```

```
#Output:  
80.0
```

nlargest(): "age" sütununu sıralar ve en büyük 5 yaş değerine sahip satırları döndürür.

```
#Input:  
df.nlargest(5, "age")
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|-----|----------|--------|------|------|-------|-------|---------|----------|-------|-----|------------|------|-------------|-------|-------|
| 630 | 1 | 1 | male | 80.0 | 0 | 0 | 30.0000 | S | First | man | True | A | Southampton | yes | True |
| 851 | 0 | 3 | male | 74.0 | 0 | 0 | 7.7750 | S | Third | man | True | NaN | Southampton | no | True |
| 96 | 0 | 1 | male | 71.0 | 0 | 0 | 34.6542 | C | First | man | True | A | Cherbourg | no | True |
| 493 | 0 | 1 | male | 71.0 | 0 | 0 | 49.5042 | C | First | man | True | NaN | Cherbourg | no | True |
| 116 | 0 | 3 | male | 70.5 | 0 | 0 | 7.7500 | Q | Third | man | True | NaN | Queenstown | no | True |

nsmallest(): "age" sütununu sıralar ve en küçük 5 yaş değerine sahip satırları döndürür.

```
#Input:  
df.nsmallest(5, "age")
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|-----|----------|--------|--------|------|-------|-------|---------|----------|--------|-------|------------|------|-------------|-------|-------|
| 803 | 1 | 3 | male | 0.42 | 0 | 1 | 8.5167 | C | Third | child | False | NaN | Cherbourg | yes | False |
| 755 | 1 | 2 | male | 0.67 | 1 | 1 | 14.5000 | S | Second | child | False | NaN | Southampton | yes | False |
| 469 | 1 | 3 | female | 0.75 | 2 | 1 | 19.2583 | C | Third | child | False | NaN | Cherbourg | yes | False |
| 644 | 1 | 3 | female | 0.75 | 2 | 1 | 19.2583 | C | Third | child | False | NaN | Cherbourg | yes | False |
| 78 | 1 | 2 | male | 0.83 | 0 | 2 | 29.0000 | S | Second | child | False | NaN | Southampton | yes | False |

style.bar(color=" yellow"): "style.bar" yöntemi kullanılarak, her hücrenin değerine bağlı olarak sarı renkte bir çubuk grafiği eklenir.

```
#Input:  
df.head(10).style.bar(color="yellow")
```

#Output:

| survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|----------|--------|----------|-----------|-------|-------|-----------|----------|--------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 male | 22.000000 | 1 | 0 | 7.250000 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 female | 38.000000 | 1 | 0 | 71.283300 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 female | 26.000000 | 0 | 0 | 7.925000 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 female | 35.000000 | 1 | 0 | 53.100000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 male | 35.000000 | 0 | 0 | 8.050000 | S | Third | man | True | NaN | Southampton | no | True |
| 5 | 0 | 3 male | nan | 0 | 0 | 8.458300 | Q | Third | man | True | NaN | Queenstown | no | True |
| 6 | 0 | 1 male | 54.000000 | 0 | 0 | 51.862500 | S | First | man | True | E | Southampton | no | True |
| 7 | 0 | 3 male | 2.000000 | 3 | 1 | 21.075000 | S | Third | child | False | NaN | Southampton | no | False |
| 8 | 1 | 3 female | 27.000000 | 0 | 2 | 11.133300 | S | Third | woman | False | NaN | Southampton | yes | False |
| 9 | 1 | 2 female | 14.000000 | 1 | 0 | 30.070800 | C | Second | child | False | NaN | Cherbourg | yes | False |

head(10): ilk 10 satırı döndürür.

```
#Input:  
df.head(10)
```

#Output:

| survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|----------|--------|----------|------|-------|-------|---------|----------|--------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |
| 5 | 0 | 3 male | NaN | 0 | 0 | 8.4583 | Q | Third | man | True | NaN | Queenstown | no | True |
| 6 | 0 | 1 male | 54.0 | 0 | 0 | 51.8625 | S | First | man | True | E | Southampton | no | True |
| 7 | 0 | 3 male | 2.0 | 3 | 1 | 21.0750 | S | Third | child | False | NaN | Southampton | no | False |
| 8 | 1 | 3 female | 27.0 | 0 | 2 | 11.1333 | S | Third | woman | False | NaN | Southampton | yes | False |
| 9 | 1 | 2 female | 14.0 | 1 | 0 | 30.0708 | C | Second | child | False | NaN | Cherbourg | yes | False |

groupby().mean(): "groupby" yöntemi, veri çerçevesinde belirtilen sütuna göre gruplandırma yapar.

"pclass" sütunu, yolcuların seyahat ettiği sınıfı ifade eder. ["survived"] ifadesi, sadece hayatı kalan yolcuların verilerini seçmek için kullanılır. "mean" yöntemi, her bir sınıf için hayatı kalan yolcuların oranının ortalamasını hesaplar. Sonuç olarak, her bir sınıf için hayatı kalma oranlarının ortalamasını içeren bir veri çerçevesi döndürülür.

```
#Input:  
df.groupby("pclass")["survived"].mean()
```

#Output:

```
pclass  
1    0.629630  
2    0.472826  
3    0.242363  
Name: survived, dtype: float64
```

Örnek 2: "groupby" yöntemi, veri çerçevesinde belirtilen sütuna göre gruplandırma yapar.

"embark_town" sütunu, yolcuların gemiye hangi limandan bindiklerini ifade eder.

["age"] ifadesi, yașlarını ifade eder. "mean" yöntemi, her bir sınıf için hayatı kalan yolcuların oranının ortalamasını hesaplar. Sonuç olarak, her bir liman için yolcuların yaş ortalamasını içeren bir veri çerçevesi döndürülür.

```
#Input:  
df.groupby("embark_town")["age"].mean()
```

#Output:

```
embark_town  
Cherbourg      30.814769  
Queenstown    28.089286  
Southampton   29.445397  
Name: age, dtype: float64
```

groupby().count(): "groupby" yöntemi, veri çerçevesinde belirtilen sütuna göre gruplandırma yapar. "embark_town" sütunu, yolcuların gemiye hangi limandan bindiklerini ifade eder. ["survived"] ifadesi, sadece hayatta kalan yolcuların verilerini seçmek için kullanılır. "count" yöntemi, her bir liman için hayatta kalan yolcuların sayısını hesaplar. Sonuç olarak, her bir liman için kaç yolcunun hayatta kaldığını içeren bir veri çerçevesi döndürülür.

```
...  
#Input:  
df.groupby("embark_town")["survived"].count()
```

#Output:

```
embark_town  
Cherbourg      168  
Queenstown     77  
Southampton    644  
Name: survived, dtype: int64
```

Örnek 2: "groupby" yöntemi, veri çerçevesinde belirtilen sütunlara göre gruplandırma yapar.

["survived"] ifadesi, sadece hayatta kalan yolcuların verilerini seçmek için kullanılır. "count" yöntemi, her bir grup için hayatta kalan yolcuların sayısını hesaplar.

Sonuç olarak, "embark_town", "age" ve "sex" sütunlarına göre gruplandırılmış her bir kombinasyon için hayatta kalan yolcuların sayısını içeren bir veri çerçevesi döndürülür.

```
...  
#Input:  
df.groupby(["embark_town", "age", "sex"])["survived"].count()
```

#Output:

```
embark_town  age      sex
Cherbourg    0.42    male     1
              0.75    female   2
              1.00    female   1
                           male     1
              3.00    female   1
                           ..
Southampton  65.00   male     1
              66.00   male     1
              70.00   male     2
              74.00   male     1
              80.00   male     1
Name: survived, Length: 236, dtype: int64
```

groupby().agg(): Her bir cinsiyet için yaşın en küçük, ortalama ve en büyük değerlerini hesaplayarak sonuçları bir veri çerçevesinde döndürür.

```
#Input:
df.groupby("sex").agg({"age":"mean"})
```

#Output:

```
age
sex
female  27.915709
male    30.726645
```

groupby().agg(): "groupby" yöntemi, veri çerçevesinde belirtilen sütuna göre gruplandırma yapar.

"agg" yöntemi, gruplanmış veriler üzerinde belirtilen işlemleri uygular.

{"age": "mean"} ifadesi, yaş sütununun ortalamasını hesaplamak için kullanılır.

Sonuç olarak, "sex" sütununa göre gruplanmış her bir cinsiyet için yaş ortalamalarını içeren bir veri çerçevesi döndürülür.

```
...  
#Input:  
df.groupby("sex").agg({"age":["min","mean","max"]})
```

#Output:

| age |
|------------------|
| sex |
| female 27.915709 |
| male 30.726645 |

Örnek 2: "groupby" yöntemi, veri çerçevesinde belirtilen sütuna göre gruplandırma yapar.

"agg" yöntemi, gruplanmış veriler üzerinde belirtilen işlemleri uygular.

{"age": "mean", "survived": "mean"} ifadesi, yaş ve hayatı kalma sütunlarının ortalamalarını hesaplamak için kullanılır.

Sonuç olarak, "sex" sütununa göre gruplanmış her bir cinsiyet için yaş ve hayatı kalma oranını ortalamalarını içeren bir veri çerçevesi döndürülür.

```
...  
#Input:  
df.groupby("sex").agg({"age":"mean",  
                      "survived":"mean"})
```

#Output:

| | age | survived |
|--------|-----------|----------|
| sex | | |
| female | 27.915709 | 0.742038 |
| male | 30.726645 | 0.188908 |

Örnek 3: "groupby" yöntemi, veri çerçevesinde belirtilen sütunlara göre gruplandırma yapar.

"agg" yöntemi, gruplanmış veriler üzerinde belirtilen işlemleri uygular.

{"age": "mean", "survived": "mean"} ifadesi, yaş ve hayatta kalma sütunlarının ortalamalarını hesaplamak için kullanılır.

Sonuç olarak, "sex" ve "class" sütunlarına göre gruplanmış her bir kombinasyon için yaş ve hayatta kalma oranı ortalamalarını içeren bir veri çerçevesi döndürülür.

```
...  
#Input:  
df.groupby(["sex","class"]).agg({"age":"mean",  
                                 "survived":"mean"})
```

#Output:

| | | age | survived |
|--------|--------|-----------|----------|
| | sex | class | |
| female | First | 34.611765 | 0.968085 |
| | Second | 28.722973 | 0.921053 |
| | Third | 21.750000 | 0.500000 |
| male | First | 41.281386 | 0.368852 |
| | Second | 30.740707 | 0.157407 |
| | Third | 26.507589 | 0.135447 |

Örnek 4: "groupby" yöntemi, veri çerçevesinde belirtilen sütunlara göre gruplandırma yapar.

"agg" yöntemi, gruplanmış veriler üzerinde belirtilen işlemleri uygular.

{"age": "mean", "survived": ["mean", "count"]} ifadesi, yaş sütununun ortalamasını ve hayatı kalma sütununun ortalama ve sayısını hesaplamak için kullanılır.

Sonuç olarak, "sex" ve "class" sütunlarına göre gruplanmış her bir kombinasyon için yaş ortalaması, hayatı kalma oranı ortalaması ve hayatı kalma sayısını içeren bir veri çerçevesi döndürülür.

```
#Input:  
df.groupby(["sex", "class"]).agg({"age": "mean",  
                                  "survived": ["mean", "count"]})
```

#Output:

| | | age | survived | |
|--------|--------|-----------|----------|-------|
| | | mean | mean | count |
| | sex | class | | |
| female | First | 34.611765 | 0.968085 | 94 |
| | Second | 28.722973 | 0.921053 | 76 |
| | Third | 21.750000 | 0.500000 | 144 |
| male | First | 41.281386 | 0.368852 | 122 |
| | Second | 30.740707 | 0.157407 | 108 |
| | Third | 26.507589 | 0.135447 | 347 |

pivot_table(): "pivot_table" yöntemi, veri çerçevesindeki belirtilen sütunları çapraz tabloya dönüştürür.

"survived" ve "sex" sütunları, tabloda satır ve sütunları oluşturmak için kullanılır.

"embarked" sütunu, tablonun sütunlarını oluşturmak için kullanılır.

"survived" sütunu, hesaplanacak değerleri belirtmek için kullanılır.

Sonuç olarak, "sex" ve "embarked" sütunlarına göre bir çapraz tablo oluşturulur ve her bir kombinasyon için hayatta kalma oranları hesaplanarak doldurulur.

```
#Input:  
df.pivot_table("survived", "sex", "embarked")
```

#Output:

| embarked | C | Q | S |
|----------|----------|----------|----------|
| sex | | | |
| female | 0.876712 | 0.750000 | 0.689655 |
| male | 0.305263 | 0.073171 | 0.174603 |

isnull().sum(): "isnull()" yöntemi, veri çerçevesindeki her bir hücrede NaN olup olmadığını kontrol eder ve bir Boolean (True/False) değer döndürür.

"sum()" yöntemi, her sütundaki True değerlerinin sayısını toplar.

```
#Input:  
df.isnull().sum()
```

#Output:

```
survived      0
pclass        0
sex           0
age          177
sibsp         0
parch         0
fare           0
embarked      2
class          0
who            0
adult_male     0
deck          688
embark_town    2
alive          0
alone          0
dtype: int64
```

dropna(): Eksik veri içeren tüm satırlar kaldırılarak güncellenmiş bir veri çerçevesi döndürülür.

```
#Input:  
df.dropna()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|-----|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 6 | 0 | 1 | male | 54.0 | 0 | 0 | 51.8625 | S | First | man | True | E | Southampton | no | True |
| 10 | 1 | 3 | female | 4.0 | 1 | 1 | 16.7000 | S | Third | child | False | G | Southampton | yes | False |
| 11 | 1 | 1 | female | 58.0 | 0 | 0 | 26.5500 | S | First | woman | False | C | Southampton | yes | True |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 871 | 1 | 1 | female | 47.0 | 1 | 1 | 52.5542 | S | First | woman | False | D | Southampton | yes | False |
| 872 | 0 | 1 | male | 33.0 | 0 | 0 | 5.0000 | S | First | man | True | B | Southampton | no | True |
| 879 | 1 | 1 | female | 56.0 | 0 | 1 | 83.1583 | C | First | woman | False | C | Cherbourg | yes | False |
| 887 | 1 | 1 | female | 19.0 | 0 | 0 | 30.0000 | S | First | woman | False | B | Southampton | yes | True |
| 889 | 1 | 1 | male | 26.0 | 0 | 0 | 30.0000 | C | First | man | True | C | Cherbourg | yes | True |

182 rows x 15 columns

isnull().sum(): "isnull()" yöntemi, veri çerçevesindeki her bir hücrede NaN olup olmadığını kontrol eder ve bir Boolean (True/False) değer döndürür.

"sum()" yöntemi, her sütundaki True değerlerinin sayısını toplar.

Sonuç olarak, her sütundaki eksik veri (NaN) sayısı, bir dizi olarak döndürülür.

```
● ● ●  
#Input:  
df.isnull().sum()
```

#Output:

```
survived          0  
pclass            0  
sex               0  
age              177  
sibsp             0  
parch             0  
fare              0  
embarked          2  
class             0  
who               0  
adult_male        0  
deck              688  
embark_town       2  
alive              0  
alone              0  
dtype: int64
```

fillna(): Veri setindeki eksik değerleri doldurma fonksiyonudur. "age" sütunundaki eksik değerleri 35 ile doldurur.

```
● ● ●  
#Input:  
df["age"] = df["age"].fillna(35)
```

```
...  
#Input:  
df.isnull().sum()
```

#Output:

```
survived      0  
pclass        0  
sex           0  
age           0  
sibsp         0  
parch         0  
fare           0  
embarked      2  
class          0  
who            0  
adult_male     0  
deck          688  
embark_town    2  
alive          0  
alone          0  
dtype: int64
```

Apply ve Lambda

`head()`: ilk 5 satırı getirir.

```
#Input:  
df.head()
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True |

`new_age` adında yeni bir sütun oluşturur ve yaş değişkenini 2 ile çarpar.

`age2` adında yeni bir sütun oluşturur ve yaş değişkenini 3 ile çarpar.

```
#Input:  
df["new_age"] = df["age"] * 2  
df["age2"] = df["age"] * 3
```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone | new_age | age2 |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|---------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False | 44.0 | 66.0 |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False | 76.0 | 114.0 |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True | 52.0 | 78.0 |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False | 70.0 | 105.0 |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True | 70.0 | 105.0 |

apply(lambda x:x/5): "age" ve "age2" sütunlarındaki her bir değeri 5'e bölen bir lambda fonksiyonunu bu sütunlara uygular.

```
#Input:  
df[["age", "age2"]].apply(lambda x:x/5)
```

#Output:

| | age | age2 |
|-----|-----|------|
| 0 | 4.4 | 13.2 |
| 1 | 7.6 | 22.8 |
| 2 | 5.2 | 15.6 |
| 3 | 7.0 | 21.0 |
| 4 | 7.0 | 21.0 |
| ... | ... | ... |
| 886 | 5.4 | 16.2 |
| 887 | 3.8 | 11.4 |
| 888 | 7.0 | 21.0 |
| 889 | 5.2 | 15.6 |
| 890 | 6.4 | 19.2 |

891 rows × 2 columns

lambda: Lambda fonksiyonu tek satırlık basit bir fonksiyon tanımlama yöntemidir. Lambda fonksiyonları genellikle başka bir fonksiyon içinde geçici olarak kullanılırlar.

```
...  
#Input:  
a=lambda x,y : x+y  
a(3,5)
```

```
...  
#Output:  
8
```

Aynı işlemi fonksiyon tanımlayarak yapmak istersek:

```
...  
#Input:  
def topla(x,y):  
    return x+y
```

"age" sütunundaki her satırın 30 yaşından büyük olup olmadığını kontrol eder ve sonucu "age_aralik" sütununa yazar. Bu yeni sütun, her satır için ya 1 (eğer yaş 30'dan büyükse) veya 0 (eğer yaş 30'dan küçük veya eşitse) içerir.

```

...
#Input:
df["age_aralik"]=df["age"].apply(lambda x :1 if x >30 else 0)

df.head()

```

#Output:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive | alone | new_age | age2 | age_aralik |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|---------|-------|------------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no | False | 44.0 | 66.0 | 0 |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes | False | 76.0 | 114.0 | 1 |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes | True | 52.0 | 78.0 | 0 |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes | False | 70.0 | 105.0 | 1 |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no | True | 70.0 | 105.0 | 1 |

Join İşlemleri

Örnek bir veri seti hazırlıyoruz (1-30 arası ve 5'e 3'lük). Sütunlarınıza a, b, c olarak adlandırıyoruz.

```

...
#Input:
import pandas as pd
import numpy as np

m=np.random.randint(1,30,size=(5,3))
df1=pd.DataFrame(m,columns=["a","b","c"])
df1

```

#Output:

| | a | b | c |
|---|----|----|----|
| 0 | 14 | 21 | 9 |
| 1 | 14 | 12 | 13 |
| 2 | 24 | 7 | 8 |
| 3 | 5 | 14 | 29 |
| 4 | 12 | 23 | 26 |

df2 adında yeni bir tablo oluşturuyoruz ve df1'deki bütün değerlerimize 72 ekliyoruz.

```
● ● ●  
#Input:  
df2=df1+72  
df2
```

#Output:

| | a | b | c |
|---|----|----|-----|
| 0 | 86 | 93 | 81 |
| 1 | 86 | 84 | 85 |
| 2 | 96 | 79 | 80 |
| 3 | 77 | 86 | 101 |
| 4 | 84 | 95 | 98 |

concat(): İki veya daha fazla veri çerçevesini birleştirerek yeni bir veri çerçevesi oluşturur.

```
...  
#Input:  
pd.concat([df1, df2])
```

#Output:

| | a | b | c |
|---|----|----|-----|
| 0 | 14 | 21 | 9 |
| 1 | 14 | 12 | 13 |
| 2 | 24 | 7 | 8 |
| 3 | 5 | 14 | 29 |
| 4 | 12 | 23 | 26 |
| 0 | 86 | 93 | 81 |
| 1 | 86 | 84 | 85 |
| 2 | 96 | 79 | 80 |
| 3 | 77 | 86 | 101 |
| 4 | 84 | 95 | 98 |

ignore_index(): Birleştirilmiş veri çerçevesindeki satırların yeni bir indeks numarası olarak atanmasını sağlar.

```
#Input:  
pd.concat([df1,df2],ignore_index=True)
```

#Output:

| | a | b | c |
|---|----|----|-----|
| 0 | 14 | 21 | 9 |
| 1 | 14 | 12 | 13 |
| 2 | 24 | 7 | 8 |
| 3 | 5 | 14 | 29 |
| 4 | 12 | 23 | 26 |
| 5 | 86 | 93 | 81 |
| 6 | 86 | 84 | 85 |
| 7 | 96 | 79 | 80 |
| 8 | 77 | 86 | 101 |
| 9 | 84 | 95 | 98 |

Merge

Örnek bir veri seti hazırlıyoruz.

```
#Input:  
#Merge ile birlestirme  
df1=pd.DataFrame({ "isciler": ["ar", "ac", "hu", "ce", "me"],  
                  "group": ["x", "y", "z", "t", "h"] })  
df2=pd.DataFrame({ "isciler": ["ac", "ce", "ar", "hu", "me"],  
                  "st_data": [2010, 2009, 2000, 1999, 1980] })
```

Verimizi yazdırıyoruz.

```
#Input:  
df1
```

#Output:

| | isciler | group |
|---|---------|-------|
| 0 | ar | x |
| 1 | ac | y |
| 2 | hu | z |
| 3 | ce | t |
| 4 | me | h |

Verimizi yazdırıyoruz.

```
#Input:  
df2
```

#Output:

| | isciler | st_data |
|---|---------|---------|
| 0 | ac | 2010 |
| 1 | ce | 2009 |
| 2 | ar | 2000 |
| 3 | hu | 1999 |
| 4 | me | 1980 |

merge(): Bir veri çerçevesindeki sütunları veya satırları birleştirmek için kullanılan bir Pandas fonksiyonudur.

```
#Input:  
pd.merge(df1,df2)
```

#Output:

| | isciler | group | st_data |
|---|---------|-------|---------|
| 0 | ar | x | 2000 |
| 1 | ac | y | 2010 |
| 2 | hu | z | 1999 |
| 3 | ce | t | 2009 |
| 4 | me | h | 1980 |

Örnek 2: Her iki veri tablosundaki “işçiler” sütunu üzerindeki eşleşen değerleri içeren birleştirilmiş bir veri tablosu oluşturur.

```
...  
#Input:  
pd.merge(df1,df2,on="isciler")
```

#Output:

| | isciler | group | st_data |
|---|---------|-------|---------|
| 0 | ar | x | 2000 |
| 1 | ac | y | 2010 |
| 2 | hu | z | 1999 |
| 3 | ce | t | 2009 |
| 4 | me | h | 1980 |

Matplotlib

Matplotlib, Python'da statik, animasyonlu ve etkileşimli görselleştirmeler oluşturmak için kullanılan kapsamlı bir kütüphanedir. Veri analizinde, veri görselleştirme amacıyla yaygın olarak kullanılmaktadır. Matplotlib hakkında detaylı bilgilere [buradan](#) ulaşabilirsiniz.

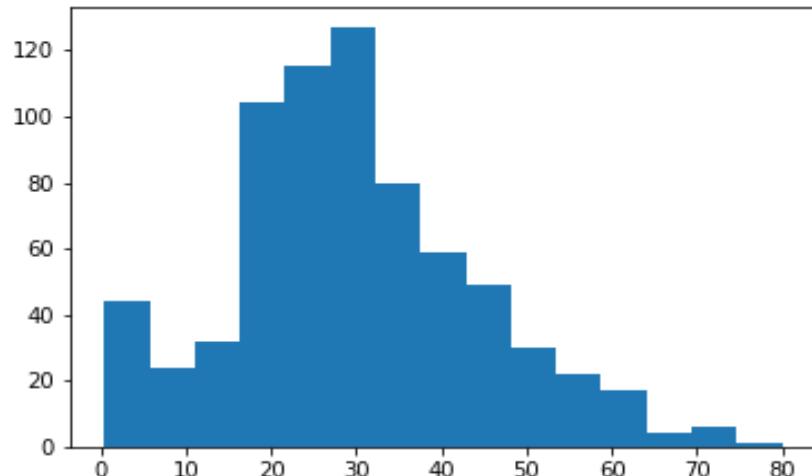
import matplotlib.pyplot as plt: Matplotlib kütüphanesini yüklemeyi sağlar.

import seaborn as sns: Seaborn kütüphanesini yüklemeyi sağlar.

plt.histogram(x): Matplotlib kütüphanesindeki histogram grafiğini çizdirir. Birçok parametresi olsa da temelde verilen dataframe'deki bir özelliğin dağılımını histogram şeklinde gösterir. "bins=None" default parametresi istenilen şekilde aralıklar oluşturabilir.

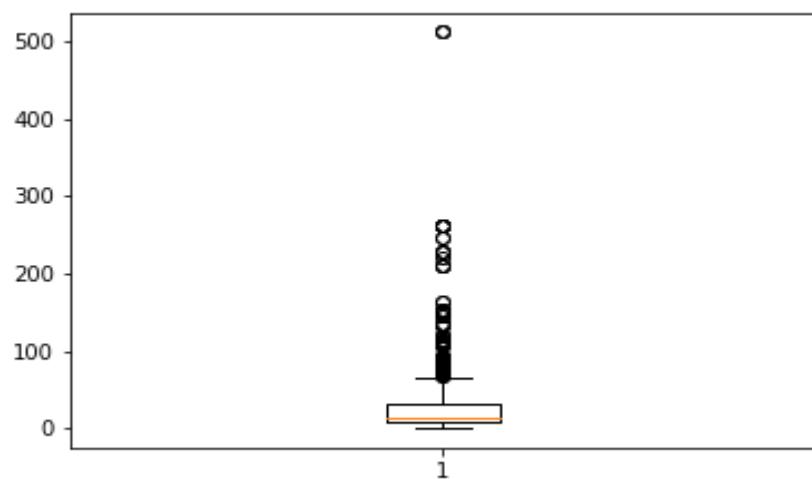
- Aşağıda yer alan kod bloğu seaborn kütüphanesinde hazır olarak yer alan "titanic" veri setindeki "age" sütununu histogram grafiğine dönüştürür.
-

```
# Input:  
import seaborn as sns  
import matplotlib.pyplot as plt  
df = sns.load_dataset("titanic")  
plt.hist(df["age"])  
plt.show()
```



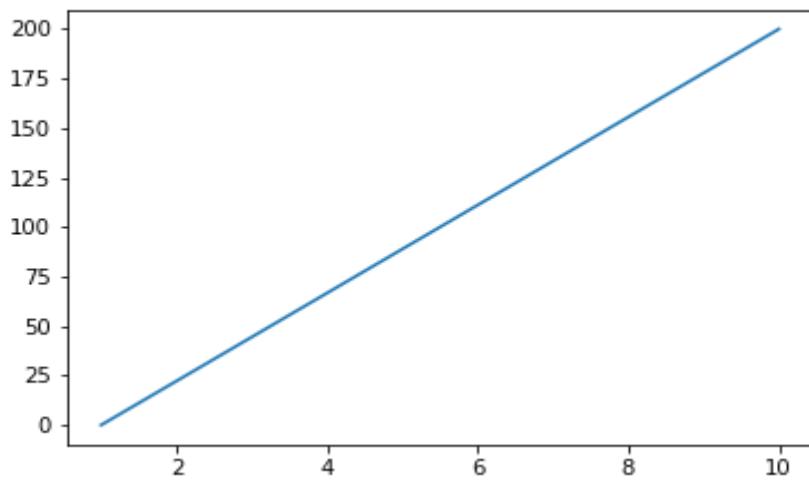
plt.boxplot(x): Verilen bir array veya sıralı vektörlerin boxplot grafiği sayesinde görselleştirilmesini sağlar. Örneğin “titanic” veri setinde “fare” sütununun dağılımını, çeyrekler aralığını, aykırı değerleri gösterir.

```
...  
# Input:  
plt.boxplot(df["fare"])  
plt.show()
```



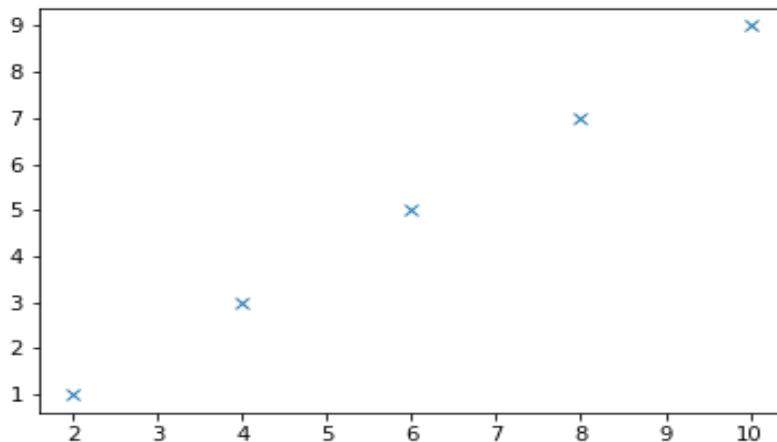
plt.plot(x,y): x'lere karşılık gelen y değerlerini çizdiren grafiktir. Matplotlib kütüphanesindeki en temel kütüphanelerden biri olsa da çok çeşitli parametreler alır.

```
...  
# Input:  
x = np.array([1,10])  
y = np.array([0,200])  
plt.plot(x,y)  
plt.show()
```



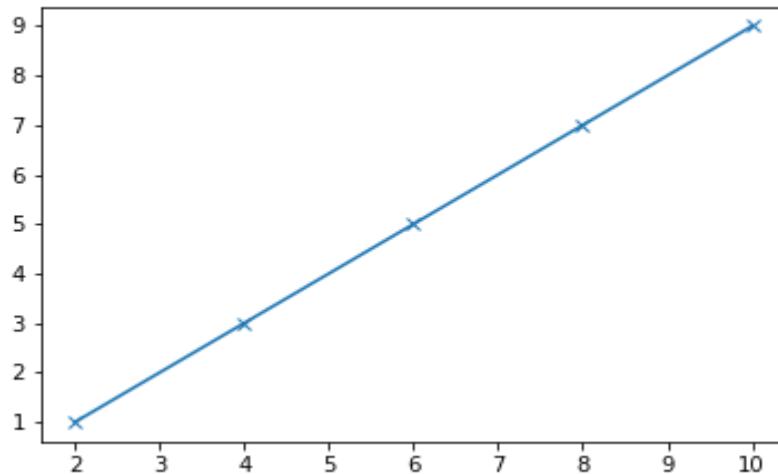
- Eğer sadece x ve y değerlerini işaretlemek istersek, string şeklinde “x” veya “*” gibi semboller kullanılır.

```
# Input:  
x = np.array([2,4,6,8,10])  
y = np.array([1,3,5,7,9])  
plt.plot(x,y,"x")  
plt.show()
```



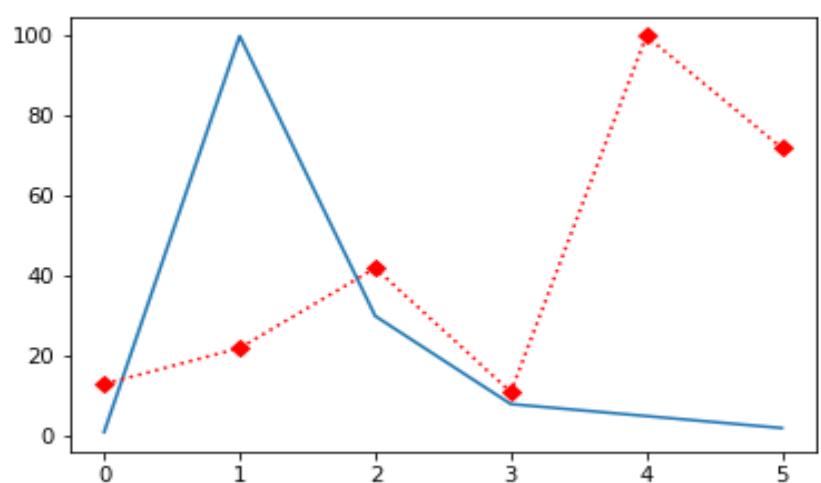
- “marker” parametresinin “x” olması ile sadece string şeklinde “x” olması arasındaki fark aşağıdaki kod bloğunda yer alıyor.

```
# Input:  
plt.plot(x,y, marker="x")  
plt.show()
```



- “linestyle” ve “color” parametrelerini değiştirmek daha farklı bir lineplot çizilebilir. Bunun yanında farklı lineplotlar üst üste çizilebilir. Bu ve bunun gibi daha gelişmiş ayarlamalar için matplotlib kütüphanesinin dokümantasyonu okumanızı tavsiye ederim.

```
# Input:  
x = np.array([1,100,30,8,5,2])  
y = np.array([13,22,42,11,100,72])  
plt.plot(x)  
plt.plot(y,marker="D",linestyle="dotted",color="r")  
plt.savefig("combined.png")  
plt.show()
```



Seaborn

Seaborn, Python programlama dilini kullanarak verileri görselleştirmek için yüksek seviye API sağlayan açık kaynaklı bir Python kütüphanesidir. Seaborn hakkında detaylı bilgiye [buradan](#) ulaşabilirsiniz.

Kütüphanelerin yüklenmesi:

İlk olarak gerekli kütüphaneleri yükleyelim:

```
#Input:  
  
import seaborn as sns  
from matplotlib import pyplot as plt  
import pandas as pd
```

Seaborn dışında yüklediğimiz diğer kütüphanelerden Pandas, veri analizi ve veri işleme için kullanılan bir kütüphanedir. Matplotlib ise veri görselleştirmesinde kullandığımız temel bir Python kütüphanesidir. 2 ve 3 boyutlu çizimler yapmamızı sağlar.

Gerekli Veri setinin Yüklenmesi: Seaborn içerisinde bazı veri setleri ile gelmektedir. Veri setinin yüklenmesi için `load_dataset()` fonksiyonu kullanılmaktadır. Örneğin ‘tips’ veri setini yükleyelim:

```
#Input:  
  
df = sns.load_dataset('tips')  
df.head()
```

#Output:

| | total_bill | tip | sex | smoker | day | time | size |
|---|------------|------|--------|--------|-----|--------|------|
| 0 | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| 1 | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| 2 | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| 3 | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| 4 | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

Renk Paleti:

Seaborn, grafiklere renk vermek ve daha estetik göstermek için kullanılabilecek color_palette() adlı bir fonksiyon sağlamaktadır. Renk paletleri, qualitative (niteliksel), sequential (ardışık) ve diverging (çeşitlilik) paletleri olmak üzere üçe ayrılmaktadır. Aşağıda paletlerle ilgili örnekler verilmiştir:

- 1- **Qualitative Paleti:** Niteliksel yani Qualitative paletler, kategorik verileri çizmek için en uygun paletlerdir.

```
● ● ●  
#Input:  
  
current_palette = sns.color_palette()  
sns.palplot(current_palette)  
plt.show()
```

#Output:



- 2- **Sequential Paleti:** Ardışık veya sequential grafikler, bir aralık içinde göreceli olarak düşük değerlerden yüksek değerlere doğru değişen verilerin dağılımını ifade etmek için uygundur.

```
● ● ●  
#Input:  
  
current_palette = sns.color_palette()  
sns.palplot(sns.color_palette('Greens'))  
plt.show()
```

#Output:



3- **Diverging Paleti:** Farklılaşan ya da Diverging paletler iki farklı renk kullanmaktadır. Her renk, ortak bir noktadan her iki yöne doğru değişen değerdeki varyasyonu temsil etmektedir.

```
...  
#Input:  
  
current_palette = sns.color_palette()  
sns.palplot(sns.color_palette('BrBG', 7))  
plt.show()
```

#Output:



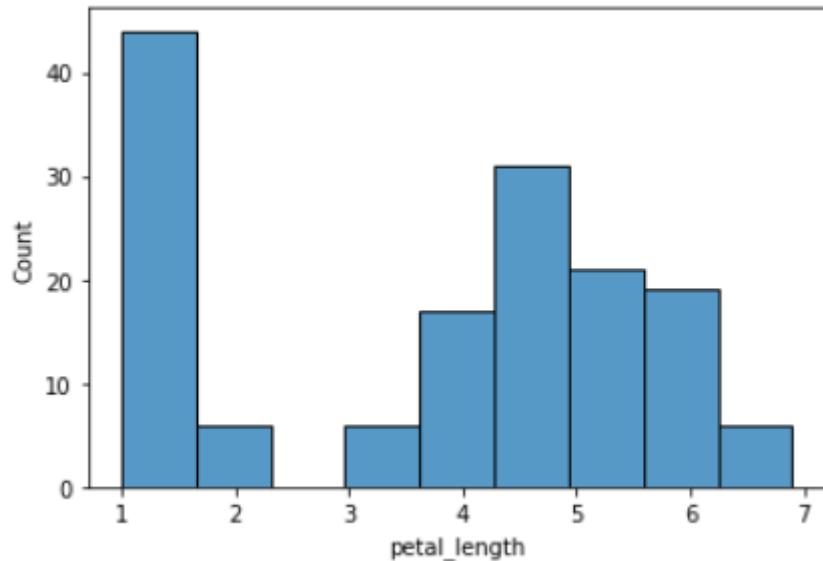
Histogram: Histogramlar, veri aralığı boyunca kutular oluşturarak ve ardından her kutuya düşen gözlem sayısını göstermek için çubuklar çizerek veri dağılımını temsil etmektedir. Histogramı göstermek için yine Seaborn içerisinde bulunan verisetlerinden ‘iris’ isimli veri setini kullanacağımız:

```
● ● ●

#Input:

df = sns.load_dataset('iris')
sns.displot(df['petal_length'], kde = False)
plt.show()
```

#Output:

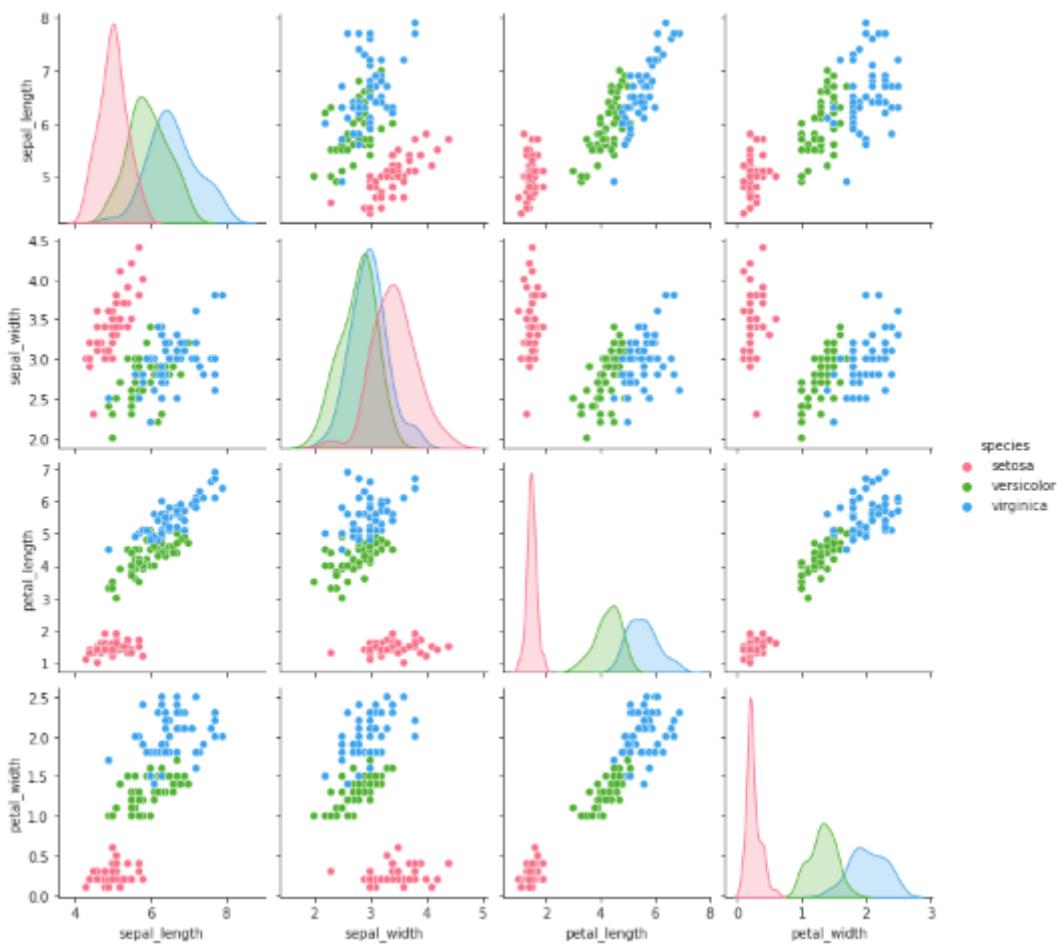


Veri Setinde Değişkenler Arasındaki İlişkileri Görselleştirme:

Veri setleri, çok sayıda değişken içermektedir. Bu gibi durumlarda, her bir değişken arasındaki ilişkinin analiz edilmesi gerekebilir. Normal şartlarda $(n,2)$ kombinasyonları için iki Değişkenli Dağılımı çizmek çok karmaşık ve zaman alan bir işlem olacaktır ancak bir veri kümesindeki çoklu **çift değişkenli** dağılımları çizmek için `pairplot()` fonksiyonunu kullanılabilmektedir. Bu, bir `DataFrame`'deki $(n,2)$ değişken kombinasyonu için ilişkiyi bir grafik matrisi olarak göstermektedir.

```
#Input:
df = sns.load_dataset('iris')
sns.set_style('ticks')
sns.pairplot(df, hue = 'species', diag_kind = 'kde', kind = 'scatter', palette = 'husl')
plt.show()
```

#Output:



Her bir grafikteki değişkenleri gözlemlileyebiliriz. Grafikler, satır adının x eksenini ve sütun adının y eksenini temsil ettiği matris biçimindedir. Diyagonal grafikler çekirdek yoğunluğu grafikleri, diğer grafikler ise belirtildiği gibi dağılım grafikleridir.

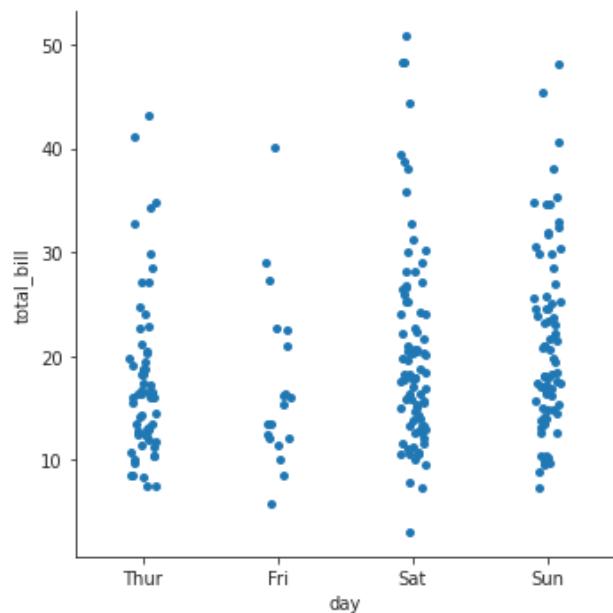
Kategorik Verilerin Görselleştirilmesi:

Ana değişkenlerden biri "kategorik" (ayrı gruplara bölünmüş) ise, görselleştirme için daha özel bir yaklaşım kullanmak gerekmektedir. Seaborn'da kategorik verileri içeren bir ilişkiyi görselleştirmenin birkaç farklı yolu vardır:

Aşağıda kullanılan catplot() fonksiyonu, verilerin dağılım (scatter) grafikleri ile gösterimini sağlamaktadır. Bu örnek için kullanılan tips veri setinde, haftanın günlerine (day) göre toplam bahşış (total_bill) sayıları görselleştirilmiştir:

```
...  
#Input:  
  
df = sns.load_dataset('tips')  
sns.catplot(data= df, x= 'day', y= 'total_bill')  
plt.show()
```

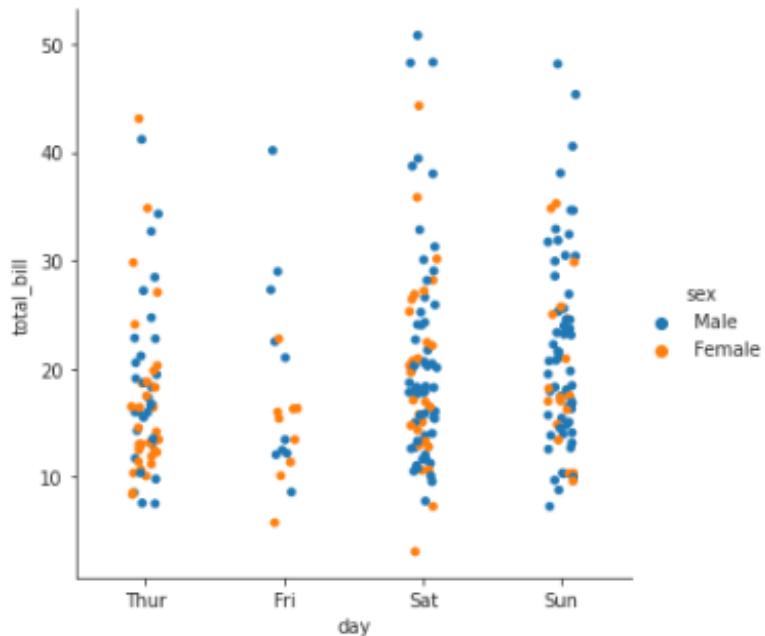
#Output:



Değişken arası ilişkileri gösteren çizimlere benzer şekilde, bir renk tonu (hue) özelliği kullanarak kategorik bir çizime başka bir boyut eklemek mümkündür. Dağılım grafiklerinde yalnızca noktaların rengini değiştirerek boyut eklenebilmektedir:

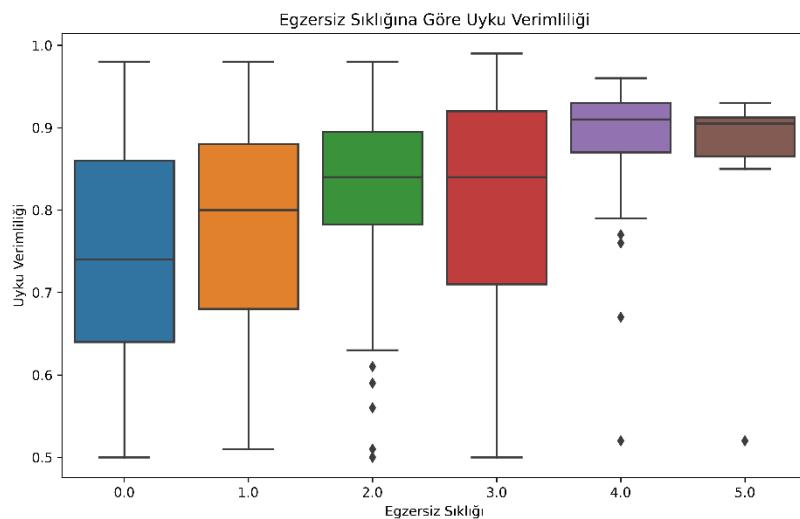
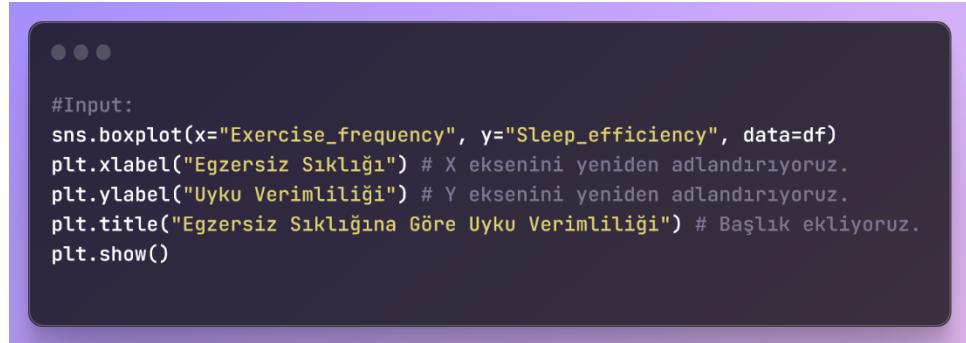
```
#Input:  
  
df = sns.load_dataset('tips')  
sns.catplot(data= df, x= 'day', y= 'total_bill', hue= 'sex')  
plt.show()
```

#Output:

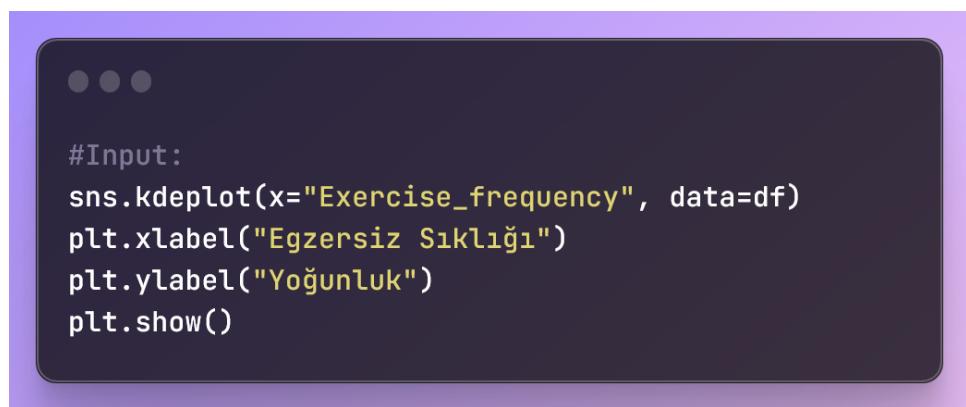


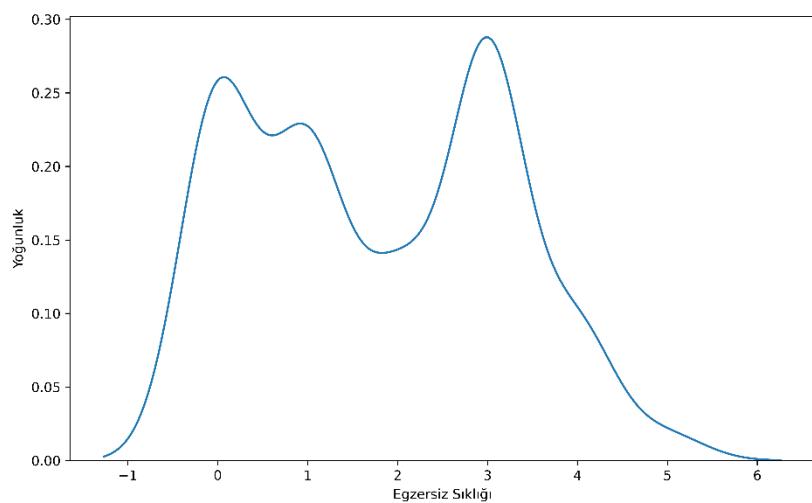
Yukarıdaki grafikte haftanın günlerine (day) göre toplam bahış (total_bill) sayılarının görselleştirildiği grafiğe cinsiyet de (sex) eklenerek bir boyut daha kazandırılmıştır.

Box Plots: Kutu grafikleri dağılımın çeyrek değerlerini aşırı değerlerle birlikte göstermektedir. Kutu grafikleri minimum değer, ilk çeyrek, medyan, üçüncü çeyrek ve maksimum değer olmak üzere verinin birçok değerlerini göstermektedir. Örnekte bir veri setinin parametrelerinin kutu grafiği örneğini görmekteyiz.



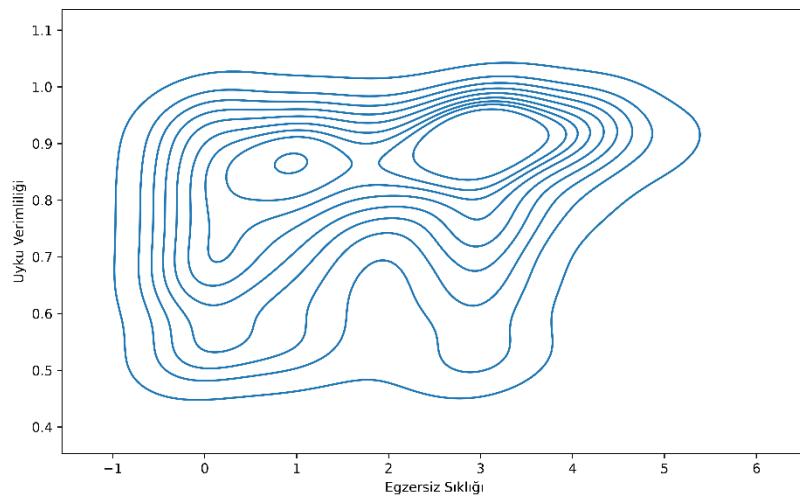
KDE (Kernel Density Estimate) Plot: KDE Plot yani *Çekirdek Yoğunluk Tahmini Grafiği*, histograma benzer şekilde bir veri kümesindeki gözlemlerin dağılımını görselleştirmek için kullanılan bir yöntemdir. KDE, verileri bir veya daha fazla boyutta sürekli bir olasılık yoğunluğu eğrisi kullanarak temsil eder. İlk olarak tek bir değişkenin yoğunluğunu inceliyoruz.



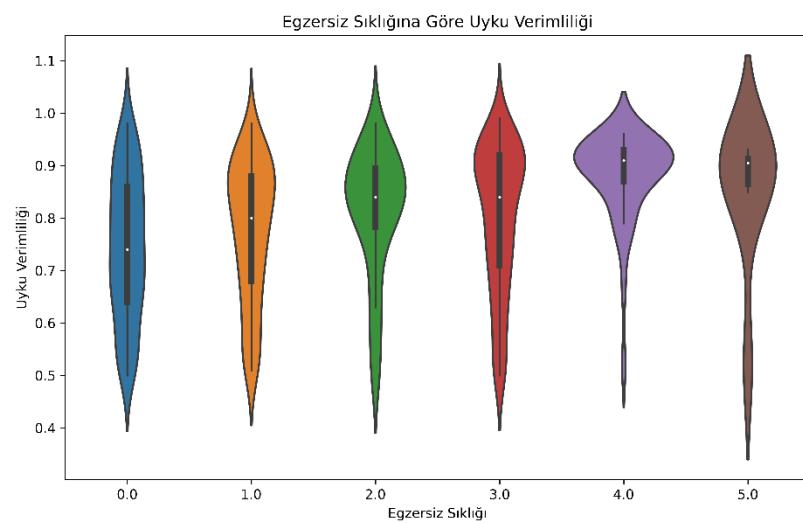
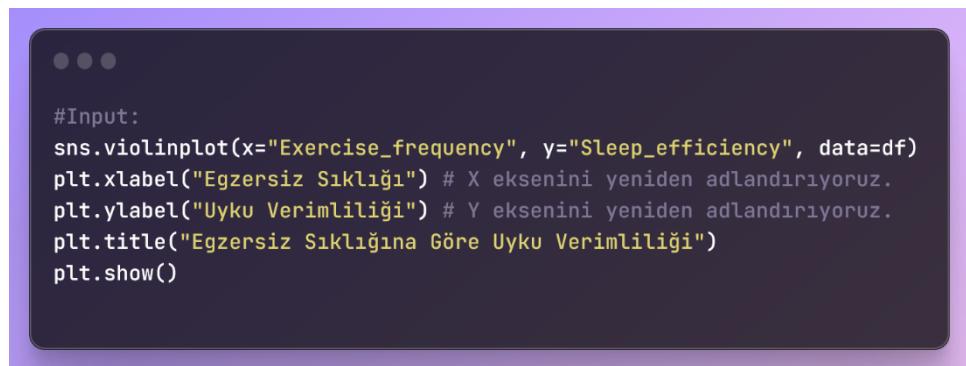


KDE Plot bize birden fazla değişkenin olasılık yoğunluğunu da çizdirmemize olanak sağlamaktadır. Aşağıda egzersiz sıklığının uyku verimliliğine göre yoğunluk dağılımını görebiliyoruz.

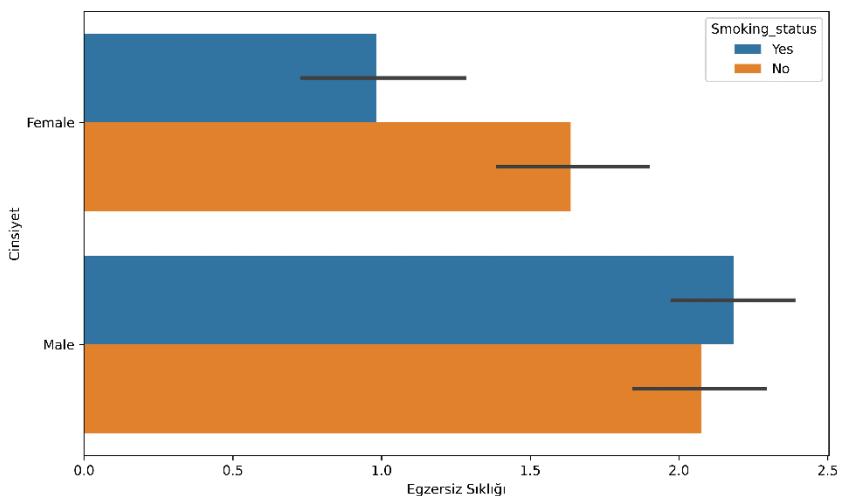
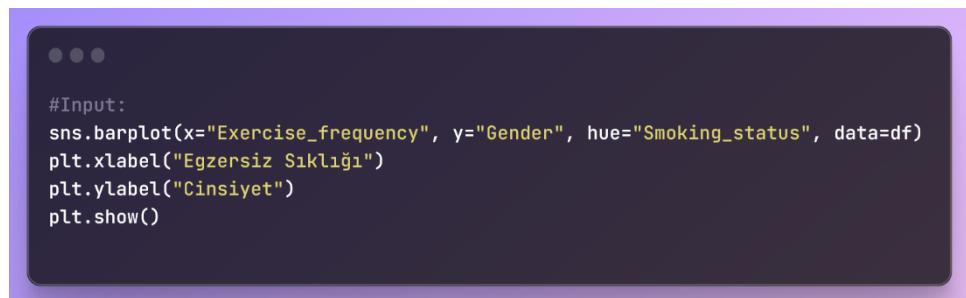
```
#Input:  
sns.kdeplot(x="Exercise_frequency", y="Sleep_efficiency", data=df)  
plt.xlabel("Egzersiz Sıklığı")  
plt.ylabel("Uyku Verimliliği")  
plt.title("Egzersiz Sıklığına Göre Uyku Verimliliği")  
plt.show()
```



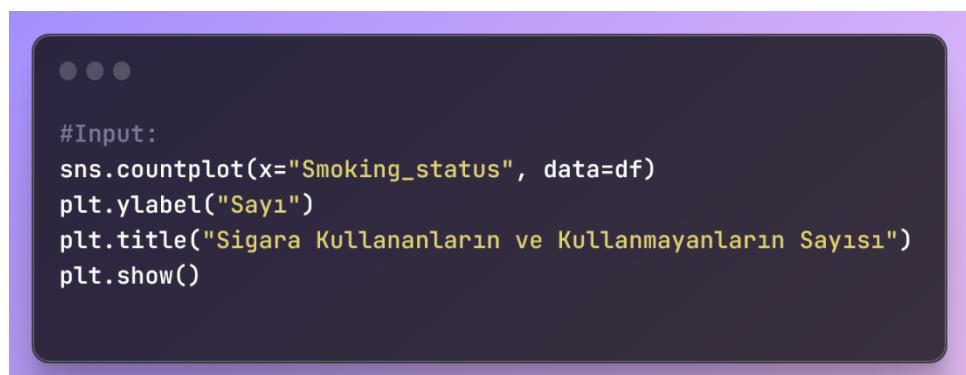
Violin Plot: Violin plot yani *keman grafiği* kutu grafiği ile yoğunluk grafiğinin bir kombinasyonudur. Bu grafik ile verilerin dağılımını ve yoğunluklarını açık bir şekilde görebilmekteyiz. Bundan dolayı veri analizi için oldukça kullanışlı bir grafik türüdür. Kutu grafiğinin tüm özelliklerini taşımaktadır. Ek olarak yoğunluk grafiğinin yoğunluk ve dağılım özelliklerini de kullanmaktadır. Aşağıdaki grafiği incelediğimizde değerlerin kutu grafiğinin çıktıları ile örtüştüğünü görebiliriz.

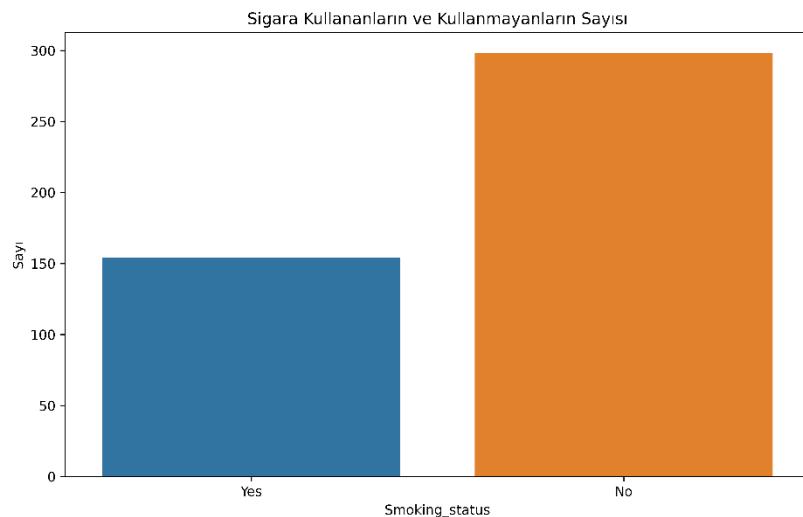


Bar Plot: Bar plot, yani sütun grafiği kategorik bir değişken ile sürekli bir değişken arasındaki ilişkiyi göstermektedir. Veriler, çubuğun uzunluğunun o kategorideki verilerin oranını temsil ettiği dikdörtgen çubuklar halinde temsil edilir. Aşağıdaki örnekte sigara kullanma durumuna göre farklı cinsiyetlerin egzersiz sikliği dağılımını görmekteyiz.



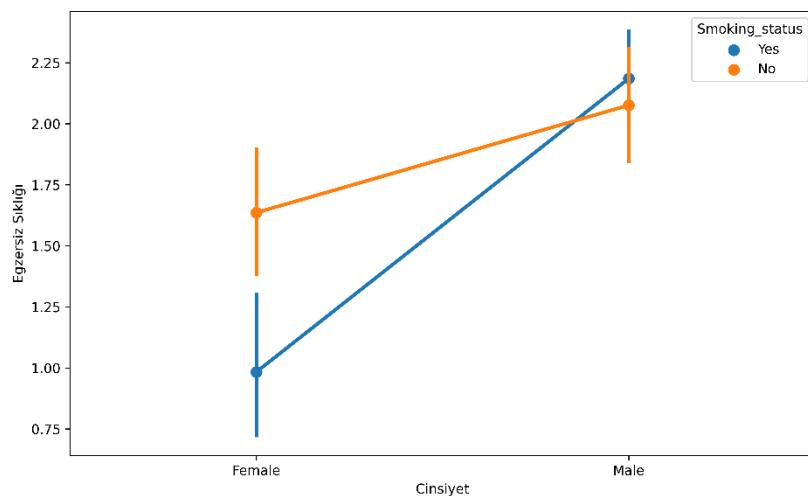
Count Plot: Her bir kategorik bölmedeki gözlemlerin sayılarını dikdörtgen çubuklar kullanarak göstermektedir. Aşağıdaki örnekte, yukarıdaki grafikte gözlemlediğimiz sigara kullanım durumlarının sayısını görmekteyiz.



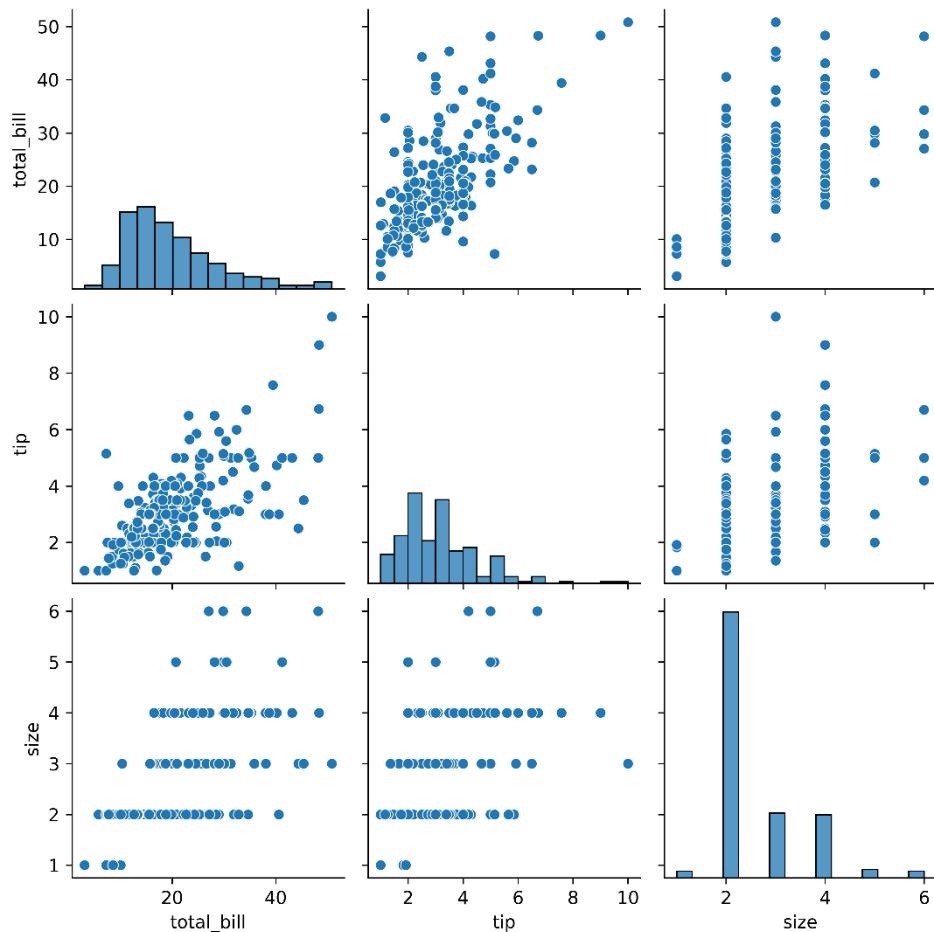
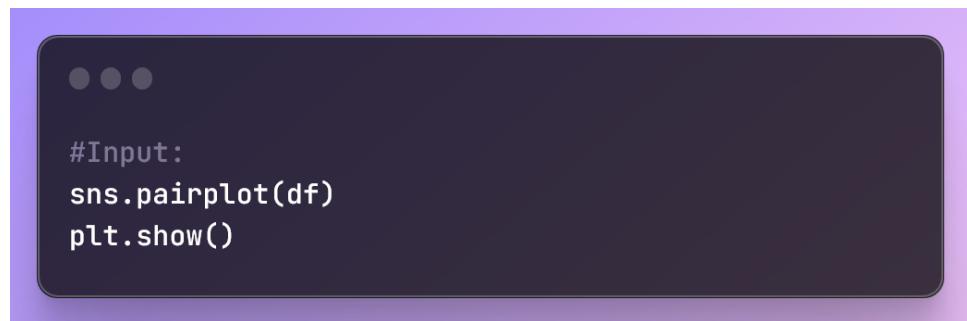


Point Plots: Point plots, yani *nokta grafikleri* sütun grafikleriyle aynı işlevi görmektedir ancak farklı bir tarzdadır. Bir nokta grafiği, dağılım grafiği noktalarının konumuna göre sayısal bir değişken için merkezi eğilim tahminini temsil eder ve hata çubuklarını kullanarak bu tahminin etrafındaki belirsizliğin bir göstergesini sağlar.

```
#Input:
sns.pointplot(x="Gender", y="Exercise_frequency", hue="Smoking_status", data=df)
plt.xlabel("Cinsiyet")
plt.ylabel("Egzersiz Sıklığı")
plt.show()
```



Pair Grid: Tüm veri çerçevesi boyunca, sayısal değişkenler için ikili ilişkileri çizmektedir. Ayrıca değişkenlerin bir alt kümesini göstermek veya satır ve sütunlarda farklı değişkenleri çizmek de mümkündür.



Heatmap: Heatmap, yani *ısı haritası* matrisin değerini görselleştirmek için renkleri kullanan verilerin grafiksel bir gösterimdir. Değişkenler arasındaki korelasyon ısı haritası ile çok rahat bir şekilde okunabilmektedir.

