

# 9 Pandas Paketi

Önceki bölümde gördüğümüz NumPy paketi, veri saklama açısından oldukça hızlı ve etkin bir paket olmakla birlikte farklı türde verilerin bir arada kullanılması, verilerden seçim yapma gibi konularda yetersiz kalabilmektedir. *Pandas* paketi hem yüksek performanslı hem de esnek veri analizi ihtiyacından doğmuş ve Wes McKinney tarafından geliştirilmiştir. Pandas'ta esas olarak iki temel veri nesnesi bulunmaktadır: seriler ve veri çerçeveleri.

## 9.1. Pandas Serileri

Pandas serilerini endekslenmiş bir boyutlu diziler olarak düşünebiliriz. Pandas serisi aşağıdaki gibi tanımlanır. Nasıl ki NumPy paketini aktarırken geleneksel olarak **np** kısaltması kullanılıyorsa Pandas paketi için de genel olarak **pd** kısaltması kullanılıyor.

```
import pandas as pd
pandas_seri = pd.Series([5, 7, 3, 10, 8, 6])
pandas_seri
0    5
1    7
2    3
3   10
4    8
5    6
dtype: int64
```

## PYTHON İLE VERİ BİLİMİ

```
type(pandas_seri)
pandas.core.series.Series

Varsayılan olarak Pandas serilerinin indeksi 0'dan başlayan ardışık tamsayılardır. Ancak dilersek farklı indeks isimleri de kullanabiliriz.

pandas_seri = pd.Series([5, 7, 3, 10, 8, 6],
                        index = ['a', 'b', 'c', 'd', 'e', 'f'])

pandas_seri
a    5
b    7
c    3
d   10
e    8
f    6
dtype: int64
```

Bu yöntümle seriler bir çeşit sözlük (dictionary) veri yapısı gibi düşünülebiliriz. Ancak sözlüklerde arka arkaya gelen değerleri seçmek mümkün değilken serilerde bu mümkündür.

```
pandas_seri['c':'e']
c    3
d   10
e    8
dtype: int64
```

## 9.2. Pandas Veri Çerçeveleri

Veri analizinde kullanılan verilerin formatı çok farklı olabilmektedir. Buna karşın günlük çalışmalarımızda kullandığımız veriler çoğu zaman satır ve sütunlardan oluşan tablolardan oluşmaktadır. Bu tür verileri iki boyutlu veriler olarak da isimlendirebiliriz. Bu nedenleki basit veri işlemlerinde en çok kullanılan program MS Excel veya benzeri programlardır. Bu tür programlar ile satır ve sütunlarda gösterilebilecek verileri rahatlıkla saklamak ve bunlar üzerinde çeşitli analizler ve hesaplamalar yapmak mümkündür. Bu tür programların kullanım kolaylığı da yaygın olarak kullanılmalarında bir başka etkendir. R programlama dilinin yayılmasının önemli sebeplerinden birisi de farklı veritipindeki sütunlardan oluşan verilerle analiz, modelleme gibi işlemleri kolaylaştırmıştır.

Diğer yandan kullanılan veri kümelerinin büyülüğu arttıkça bu tür programlarda veri işleme ve analiz çok zor ve yavaş bir hale gelmektedir. Bilgisayar donanım teknolojinin önceki yıllara nazaran çok daha gelişmiş olduğu günümüzde dahi birçok veri kümesi birkaç yüz bin satırlık bir dosya hele bir de formüller içeriyorsa bilgisayarı zorlayabilmekte

## PANDAS PAKETİ

ve çalışmaları oldukça yavaşlatabilmektedir. Aşağıda görülen veri seti yapay öğrenme konusunda çok sık kullanılan iris (zambak) veri kümelerinin bir bölümünü göstermektedir. Bu veri kümelerinde 150 zambak çiçeğine ait (tabloda bir kısmı gösterilmiştir) çanak ve taç yapraklarının en ve boy ölçümleri yapılmış ve üç farklı zambak türü için bu ölçümler kaydedilmiştir.

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.4	0.2	setosa
6	4.6	3.4	1.7	0.4	setosa
					setosa

Tablo şeklindeki ya da iki boyutlu veri kümelerinde her satır bir gözlemi ya da bir ölçüyü belirtmektedir. Her sütun ise bu gözlemlere ilişkin farklı özelliklerini yani **değişkenleri** belirmektedir. Örneğin, yukarıdaki tabloda her satır bir çiçek için yapılan ölçümü ifade etmektedir. Sütunlarda ise her bir ölçüme ait farklı özellikler (çanak yaprak en ve boyu ile taç yaprak en ve boyu) ve çiçek türü (setosa, versicolor, virginica) yer almaktadır.

Neyse ki Matlab, R gibi dillerin yanı sıra Python'da da iki (ve daha fazla) boyutlu veri kümeleri için kullanılabilen paketler mevcuttur. Bunlar arasında yer alan NumPy paketini daha önce görmüştük. Ancak NumPy paketinin bir eksikliği, NumPy dizilerinin sadece aynı türde verilerden oluşabileceğidir. Ancak, gerçek hayatı kullandığımız birçok veri kümesi farklı sütunlarda farklı veri türleri içerebilmektedir. Örneğin, yukarıdaki tabloda ilk dört sütundaki veriler sayısal, son sütundaki veri tipi ise metin türündedir.

Pandas paketinde yukarıdaki gibi iki boyutlu veri kümeleri **DataFrame** (veri çerçevesi) olarak isimlendirilir. Veri çerçeveleri, Python'da veri analizini son derece kolaylaştırın çok kullanılmış veri yapılarıdır. R programlama dilini bilenler için Pandas veri çerçevesi, R'daki veri çerçevesi ile aynıdır. Veri çerçeveleri yukarıda anladığımız iki boyutlu veri kümesi özelliklerini taşıır. Her satırda bir gözlem, her sütunda da farklı bir değişken yer alır. Bir sütunda yer alan tüm veriler aynı tiptedir ama farklı sütunlarda farklı veri türleri yer alabilir. Veri çerçeveleri, veri bilimi açısından çok faydalıdır. Veri çerçevelerini kullanarak betimsel ve ilişkisel veri analizi yapmak, istatistiksel modeller kurmak, verileri görselleştirmek daha kolaydır. Bunlardan dolayı veri biliminde Python kullananlar için Pandas modülünden faydalanan standart hale gelmiştir.

Bir pandas DataFrame oluşturmak için farklı yöntemler vardır. Bunlardan bir tanesi sözlük veri yapısını kullanmaktadır. Şimdi yukarıda yer alan tabloyu oluşturmak için önce bu verileri bir sözlük olarak tanıyalım.

## PYTHON İLE VERİ BİLİMİ

```

iris_dict = {
    'Sepal_Length': [5.1, 4.9, 4.7, 4.6, 5.0, 6.7, 6.3, 6.5, 6.2, 5.9],
    'Sepal_Width' : [3.5, 3.0, 3.2, 3.1, 3.6, 3.0, 2.5, 3.0, 3.4, 3.0],
    'Petal_Length': [1.4, 1.4, 1.3, 1.5, 1.4, 5.2, 5.0, 5.2, 5.4, 5.1],
    'Petal_Width' : [0.2, 0.2, 0.2, 0.2, 0.2, 2.3, 1.9, 2.0, 2.3, 1.8],
    'Species'     : ['setosa', 'setosa', 'setosa', 'setosa', 'setosa',
                    'virginica', 'virginica', 'virginica', 'virginica',
                    'virginica']
}

```

Sonrasında da pandas paketini import komutu ile içeri aktarıyoruz.

```
import pandas as pd
```

Son olarak yukarıda tanımladığımız sözlüğü pandas DataFrame veri tipine dönüştürmek için `DataFrame()` metodunu kullanıyoruz. Oluşturulan veri çerçevesinin (DataFrame) satırları sıfırdan başlamak üzere numaralandırılır.

```

iris = pd.DataFrame(iris_dict)
print(iris)

Petal_Length  Petal_Width  Sepal_Length  Sepal_Width   Species
0            1.4         0.2          5.1        3.5      setosa
1            1.4         0.2          4.9        3.0      setosa
2            1.3         0.2          4.7        3.2      setosa
3            1.5         0.2          4.6        3.1      setosa
...

```

Şimdi biraz daha uzun bir yöntem görelim. Bu yöntemi pek kullanmayacak olsak da farklı veri yapılarını tekrar etmek için iyi bir fırsat. Bu yöntemde veri çerçevesi sütunlarını önce listeler olarak tanımlıyoruz. Daha sonra sütun isimleri ve sütunları zip fonksiyonunu kullanarak tuple verilerden oluşan bir listeye çeviriyoruz. Burada zipledikten sonra tekrar list fonksiyonunu kullandığımıza dikkat edin. Sonraki aşamada zip veri yapısını sözlüğe ve en son olarak da sözlüğü Pandas veri çerçevesine çeviriyoruz.

```
# Önce sütunları liste olarak tanımlayalım.
```

```

Sepal_Length = [5.1, 4.9, 4.7, 4.6, 5.0, 6.7, 6.3, 6.5, 6.2, 5.9]
Sepal_Width = [3.5, 3.0, 3.2, 3.1, 3.6, 3.0, 2.5, 3.0, 3.4, 3.0]
Petal_Length = [1.4, 1.4, 1.3, 1.5, 1.4, 5.2, 5.0, 5.2, 5.4, 5.1]
Petal_Width = [0.2, 0.2, 0.2, 0.2, 0.2, 2.3, 1.9, 2.0, 2.3, 1.8]

```

## PANDAS PAKETİ

```

Species = ['setosa', 'setosa', 'setosa', 'virginica', 'setosa', 'virginica',
           'virginica', 'virginica', 'setosa', 'setosa', 'virginica', 'virginica',
           'virginica', 'virginica', 'virginica', 'virginica']

liste_etiketleri = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_
Width', 'Species']
sutunlar = [Sepal_Length, Sepal_Width, Petal_Length, Petal_Width]
# Liste etiketleri ve sutunları zip fonksiyonu ile bir araya getirelim.
zip_veri = list(zip(liste_etiketleri, sutunlar))
print(zip_veri)

[('Sepal_Length', [5.1, 4.9, 4.7, 4.6, 5.0, 6.7, 6.3, 6.5, 6.2, 5.9]),
 ('Sepal_Width', [3.5, 3.0, 3.2, 3.1, 3.6, 3.0, 2.5, 3.0, 3.4, 3.0]),
 ('Petal_Length', [1.4, 1.4, 1.3, 1.5, 1.4, 5.2, 5.0, 5.2, 5.4, 5.1]),
 ('Petal_Width', [0.2, 0.2, 0.2, 0.2, 0.2, 2.3, 1.9, 2.0, 2.3, 1.8])]

# Ziplenmiş veriyi sözlük veri yapısına çevirelim
veri = dict(zip_veri)
print(veri)

{'Sepal_Length': [5.1, 4.9, 4.7, 4.6, 5.0, 6.7, 6.3, 6.5, 6.2, 5.9],
 'Sepal_Width' : [3.5, 3.0, 3.2, 3.1, 3.6, 3.0, 2.5, 3.0, 3.4, 3.0],
 'Petal_Length': [1.4, 1.4, 1.3, 1.5, 1.4, 5.2, 5.0, 5.2, 5.4, 5.1],
 'Petal_Width' : [0.2, 0.2, 0.2, 0.2, 0.2, 2.3, 1.9, 2.0, 2.3, 1.8]}

# Son olarak sözlüğü Veri Çerçeveşine çevirelim.
iris = pd.DataFrame(veri)
print(iris)

Petal_Length  Petal_Width  Sepal_Length  Sepal_Width   Gözlem_No
0            1.4         0.2          5.1        3.5             1
1            1.4         0.2          4.9        3.0             1
2            1.3         0.2          4.7        3.2             1
3            1.5         0.2          4.6        3.1             1
...

```

Varolan bir veri çerçevesine yeni bir sütun eklemek de mümkündür.

```

iris['Gözlem_No'] = 1
print(iris)

Petal_Length  Petal_Width  Sepal_Length  Sepal_Width   Gözlem_No
0            1.4         0.2          5.1        3.5             1
1            1.4         0.2          4.9        3.0             1
2            1.3         0.2          4.7        3.2             1
3            1.5         0.2          4.6        3.1             1
...

```

## PYTHON ile VERİ BİLİMİ

Yukarıda aktardığımız, sözlükten veri çerçevesi üretme yöntemi günlük hayatda kullanıla bilen bir yöntem değildir. Gerçekte çok büyük verilerle çalıştığımızı düşünürsek tek tek bütün verilerin sözlük olarak girilmesi mümkün değildir.

Günlük hayatda kullandığımız veriler genelce excel, csv gibi formatlarda gelir. Csv formatındaki bir dosyadan pandas formatında veri okumak için pandas paketindeki `.read_csv()` metodunu kullanır. Yukarıdaki iris veri setinin `iris.csv` dosyasında kayıtlı olduğunu düşünelim. Dosyadan veri setini okumak için aşağıdaki kodu yazıyoruz. Metod içerisinde dosya adından önce klasör yolunu tam olarak yazmak gerekiyor<sup>15</sup>.

```
import pandas as pd
iris = pd.read_csv("C:/Users/.../iris.csv")
Şimdi iris veri nesnesinin türüne bakalım:
type(iris)
pandas.core.frame.DataFrame
```

Veri çerçevesi hangi sütunlardan oluşuyor? Bunu görmek için `.columns` özelliğini kullanabiliriz.

```
iris.columns
Index(['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width',
       'Species'],
      dtype='object')
```

Aynı şekilde bir veri çerçevesinin sütun isimlerini değiştirmek de mümkündür.

```
iris.columns = ['Taç Boy', 'Taç En', 'Çanak Boy', 'Çanak En', 'Tür']
print(iris)
Taç Boy Taç En Çanak Boy Çanak En Tür
0 1.4 0.2 5.1 3.5 setosa
1 1.4 0.2 4.9 3.0 setosa
2 1.3 0.2 4.7 3.2 setosa
```

Iris veri çerçevesinin genel yapısını incelemek için `.head()` metodunu kullanabiliriz. Bu metod da argüman olarak görmek istediğimiz satır sayısını iletебiliriz.

```
iris.head(10)
```

<sup>15</sup> Pandas ile veri okuma konusunu sonraki bölümlerde ayrıntılı olarak göreceğiz.  
146

## PANDAS PAKETİ

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.2	setosa
6	4.6	3.4	1.4	0.4	setosa
7	5.0	3.4	1.5	0.3	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

Benzer şekilde, veri çerçevesindeki son satırları görmek için de `.tail()` metodunu kullanılır. Bu metoda da aynı biçimde sondan kaç satırı görmek istediğimiz yazılır.

```
iris.tail(4)
```

	sepal_length	sepal_width	petal_length	petal_width	species
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

Yukarıdaki komutu Jupyter Python defteri kullanarak yazdım ve yukarıda gördüğünüz tablo formatlarını da yine aynı programdan aldım. PyCharm gibi entegre kodlama programlarında sadece yazdırma istediğiniz veri yapısının adını girmeniz (örneğin iris) herhangi bir sonuç vermeyecektir. Bir veri yapısının tamamını ya da bir kısmını yazdırınmak için `print()` fonksiyonunu kullanmanız gereklidir. Diğer yandan Jupyter gibi daha çok veri analizi için geliştirilmiş ortamlarda doğrudan verinin adını yazmanız yetecektir. Jupyter'de print fonksiyonunu kullanırsanız da sonuç alırsınız ancak format biraz daha farklı olur.

```
print(iris.tail(4))
   Sepal_Length Sepal_Width Petal_Length Petal_Width Species
146          6.3        2.5         5.0       1.9       2
147          6.5        3.0         5.2       2.0       2
148          6.2        3.4         5.4       2.3       2
149          5.9        3.0         5.1       1.8       2
```

Veri çerçevesindeki sütunları veri türüne baktığımızda Pandas içinde tanımlanmış olan Index veri türünde olduğunu görüyoruz. Bu tanımlama sayesinde Pandas veri çerçevelerinde arama, sıralama vb. gibi işlemler çok hızlı bir şekilde yapılabilmektedir. Benzer şekilde satırlar da indekslenmektedir. Iris veri çerçevesinin indeksine bakalım:

## PYTHON İLE VERİ BİLİMLİ

```
iris.index  
RangeIndex(start=0, stop=150, step=1)  
Veri çerçevesinin boyutlarına yani kaç satır ve kaç sütundan oluştuğuna  
bakalım.  
iris.shape  
(150, 5)
```

Veri çerçevesine ait bilgileri alabileceğimiz bir diğer metod `.info()` metodudur ancak bu metod istatistiksel özellikler yerine sütunların veri tipi, veri çerçevesindeki satır sayısı, her bir sütundaki veri sayısı gibi değerleri verir.

```
iris.info()  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 150 entries, 0 to 149  
Data columns (total 5 columns):  
Sepal_Length    150 non-null float64  
Sepal_Width     150 non-null float64  
Petal_Length    150 non-null float64  
Petal_Width     150 non-null float64  
Species         150 non-null object  
dtypes: float64(4), object(1)  
memory usage: 5.9+ KB
```

Bir veri çerçevesini olduğu gibi başka bir veri çerçevesine aktarmak için `.copy()` metodunu kullanabiliriz.

```
iris_yeni = iris.copy()
```

Pandas paketinin, NumPy paketi üzerinde geliştirildiğini daha önce belirtmiştık. İstenirse bir pandas veri çerçevesini `.values` özelliğini kullanarak NumPy dizisine çevirebiliriz.

```
np_dizi = iris.values  
print(np_dizi)  
  
[[5.1 3.5 1.4 0.2 'setosa']  
[4.9 3.0 1.4 0.2 'setosa']  
[4.7 3.2 1.3 0.2 'setosa']  
[4.6 3.1 1.5 0.2 'setosa']  
[5.0 3.6 1.4 0.2 'setosa']  
[5.4 3.9 1.7 0.4 'setosa']]
```

## PANDAS PAKETİ

Numpy için geçerli olan metodları pandas veri çerçevelerine de uygulayabiliriz. Örneğin sayısal değerlerden oluşan bir veri çerçevesindeki değerlerin doğal logaritmasını bulmak için NumPy paketindeki `.log()` metodunu uygulayabiliriz.

```
import numpy as np  
# Önce iris veri çerçevesinin sayısal sütunlarını seçelim  
dizi = iris.iloc[:, [0, 1, 2, 3]]  
dizi_log = np.log(dizi)  
print(dizi_log.head())
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
0	1.629241	1.252763	0.336472	-1.609438
1	1.589235	1.098612	0.336472	-1.609438
2	1.547563	1.163151	0.262364	-1.609438
3	1.526056	1.131402	0.405465	-1.609438
4	1.609438	1.280934	0.336472	-1.609438

Şimdiye kadar gördüğümüz pandas serileri ve veri çerçeveleri esasen bir ve iki boyutlu veri yapılarıdır. Ancak, pandas üç ve daha fazla boyutlu veri yapılarına da izin vermektedir. Bunu, pandas verilerinde çoklu indeks ekleyerek yapıyoruz. Önce bir pandas serisine çoklu indeks eklemeyi görelim. Aşağıdaki örnekte farklı hisse senetlerinin 2016 ve 2017 yılı kapanış fiyatlarını görüyoruz.

```
import pandas as pd  
  
indeks = [('ABC', 2016), ('ABC', 2017),  
          ('DEF', 2016), ('DEF', 2017),  
          ('XYZ', 2016), ('XYZ', 2017),  
          ('KLM', 2016), ('KLM', 2017)]  
fiyatlar = [32.5, 12.3, 24.7, 18.6, 20.3, 7.2, 51.9, 56.2]  
  
hisseler = pd.Series(fiyatlar, index=indeks)  
hisseler  
(ABC, 2016)    32.5  
(ABC, 2017)    12.3  
(DEF, 2016)    24.7  
(DEF, 2017)    18.6  
(XYZ, 2016)    20.3  
(XYZ, 2017)    7.2  
(KLM, 2016)    51.9  
(KLM, 2017)    56.2  
dtype: float64
```

Verileri bu şekilde indeksleyerek seride seçim de yapabiliriz.

## PYTHON İLE VERİ BİLİMİ

```

hisseler[('XYZ', 2016):('KLM', 2017)]
    (XYZ, 2016)    20.3
    (XYZ, 2017)    7.2
    (KLM, 2016)   51.9
    (KLM, 2017)   56.2
dtype: float64

```

Yukarıdaki gibi çoklu indeks kullanmanın daha etkin bir yöntemi ise pandas MultiIndex kullanmaktadır.

```

indeks = pd.MultiIndex.from_tuples(indeks)
indeks
MultiIndex(levels=[['ABC', 'DEF', 'KLM', 'XYZ'], [2016, 2017]],
           labels=[[0,0,1,1,3,3,2,2], [0,1,0,1,0,1,0,1]])
hisseler = pd.Series(fiyatlar, index=indeks)
hisseler
ABC 2016    32.5
      2017    12.3
DEF 2016    24.7
      2017    18.6
XYZ 2016    20.3
      2017    7.2
KLM 2016    51.9
      2017    56.2
dtype: float64
indeksleri isimlendirmek de mümkündür.
hisseler.index.names = ['Hisse', 'Yıl']
hisseler
Hisse  Yıl
ABC  2016  32.5
      2017  12.3

```

Yukarıdaki hisseler verisi bir boyutlu pandas verisi ancak ilk iki sütunda belirtilen iki indeksle gösteriliyor. Bu seriyi pandas modülündeki `unstack()` metodunu kullanarak veri çerçevesine çevirebiliriz.

```

hisseler.unstack()
2016 2017
ABC 32.5 12.3
DEF 24.7 18.6
KLM 51.9 56.2
XYZ 20.3 7.2

```

## PANDAS PAKETİ

Veri çerçevesini de `.stack()` metodunu ile eski haline çevirebiliriz. MultiIndex ile çoklu indeks oluşturmanın üç farklı yöntemi vardır.

```

pd.MultiIndex.from_tuples([('x', 1), ('x', 2), ('y', 1), ('y', 2)])
MultiIndex(levels=[['x', 'y'], [1, 2]],
           labels=[[0, 0, 1, 1], [0, 1, 0, 1]])

```

```

pd.MultiIndex.from_arrays([['x', 'x', 'y', 'y'], [1, 2, 1, 2]])
MultiIndex(levels=[['x', 'y'], [1, 2]],
           labels=[[0, 0, 1, 1], [0, 1, 0, 1]])

```

```

pd.MultiIndex.from_product([('y', 'y'), [1, 2]])
MultiIndex(levels=[['x', 'y'], [1, 2]],
           labels=[[0, 0, 1, 1], [0, 1, 0, 1]])

```

Nasıl ki satırlar için çoklu indeks oluşturabiliyorsak sütunlar için de aynı şekilde çoklu indeks oluşturabiliriz. Çoklu indeks kullanım özellikle panel verilerle yapılan çalışmalarında oldukça faydalıdır. Şimdi yukarıdaki hisse senedi tablosuna sütunlarına her çeyrek için kapanış fiyatı ve hacim bilgisini ekleyelim.

```

indeks = pd.MultiIndex.from_product([('ABC', 'DEF', 'KLM'), [2016, 2017]])
sutunlar = pd.MultiIndex.from_product([('Q1', 'Q2', 'Q3', 'Q4'), ['Hacim', 'Kapanış']])

```

```

veri = np.array([10762, 32.5, 12638, 33.5, 13689, 35.2, 18414, 37.4,
                10688, 12.3, 14348, 13.2, 15924, 15.3, 15243, 17.9,
                14841, 24.7, 14966, 22.1, 13098, 19.3, 17439, 15.4,
                15607, 18.6, 16166, 13.4, 10627, 15.1, 13396, 15.3,
                12565, 20.3, 17469, 21.4, 18964, 22.5, 15187, 23.6,
                13203, 7.2, 10629, 7.2, 15239, 8.2, 18307, 9.1])
veri = np.reshape(veri, (6, 8))

```

```

hisseler = pd.DataFrame(veri, index = indeks, columns = sutunlar)
hisseler

```

## PYTHON İLE VERİ BİLİMI

	Q1	Q2		Q3		Q4	
		Hacim	Kapanış	Hacim	Kapanış	Hacim	Kapanış
ABC	2016	10782.0	32.5	12638.0	33.5	13889.0	35.2
	2017	10688.0	12.3	14348.0	13.2	15924.0	15.3
DEF	2016	14841.0	24.7	14966.0	22.1	13098.0	19.3
	2017	15607.0	18.6	16166.0	13.4	10627.0	15.1
KLM	2016	12585.0	20.3	17469.0	21.4	18964.0	22.5
	2017	13203.0	7.2	10629.0	7.2	15239.0	8.2

Şekil 9-1: Pandas veri çerçevesinde çoklu indeks kullanımı

## 9.3. Veri ÇerçeveSinde Seçim Yapma

Bir veri çerçevesindeki herhangi bir sütunu seçmek ya da görmek için `veri_cerçeveSı["sütun_ismi"]` formatı kullanılır. Önce veri çerçevesinin ismi ve yanında tırnak köşeli parantez içinde kullanılır. Yukarıdaki iris veri de, tırnak ya da kesme işaretini içinde görülmek istenen sütun ismi yazılır. Yukarıdaki iris veri setinde `Petal_Length` sütununu görmek için `iris["Petal_Length"]` yazarız.

```
iris["Sepal_Length"]
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
5      5.4
6      4.6
...
144     6.7
145     6.7
146     6.3
147     6.5
148     6.2
149     5.9
Name: Sepal_Length, Length: 150, dtype: float64
```

Veri setinin yanında tek köşeli parantez kullanırsak sütundaki veriyi bir boyutlu olarak indirir. Bunu bir liste olarak da düşünübilirsiniz. Nitekim, bu şekilde çektiğimiz verinin türünü incelediğimizde bir boyutlu Series veri tipi olduğunu görürüz.

```
type(iris_veri["Sepal.Length"])
pandas.core.series.Series
```

## PANDAS PAKETİ

Yukarıda da görüldüğü gibi veri çerçevesinden çektiğimiz sütunun türü `Series` yani bir boyutlu bir dizidir. Eğer istedigimiz sütunun da veri çerçevesi gibi davranışmasını istiyorsak sütun ismini iki köşeli parantez içine yazmamız gereklidir. Sorgulamayı `iris[["Sepal_Length"]]` şeklinde iki köşeli parantez içinde yaparsak aşağıdaki şekilde bir pandas veri çerçevesi ile karşılaşırız.

```
Sepal_Length
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
type(iris[["Sepal_Length"]])
pandas.core.frame.DataFrame
```

Gördüğü gibi veri çerçevesindeki bir sütunu bir adet köşeli parantezle çekince bir boyutlu seri, iki adet köşeli parantezle çekince DataFrame veri yapısı ile karşılaşırız.

Peki birden fazla sütunu çekmek istersek ne yapmamız gereklidir? Sorgulamayı, bir adet köşeli parantezle yaptığımızda bir boyutlu seri üretildiğini söyleyelim. Bu nedenle örneğin `iris[["Sepal_Length", "Sepal_Width"]]` şeklinde yazarsak hata mesajı ile karşılaşırız. Bu nedenle birden fazla değişkeni seçmek için iki adet köşeli parantez kullanmalıyız.

```
iris[["Sepal_Length", "Sepal_Width"]]
   Sepal_Length  Sepal_Width
0            5.1         3.5
1            4.9         3.0
2            4.7         3.2
3            4.6         3.1
4            5.0         3.6
5            5.4         3.9
...
```

Sütunları nasıl seçeceğimizi gördük. Şimdi de satırları nasıl seçeceğimizi görelim. Yine iris veri çerçevesi ile devam edelim. Bir veri çerçevesinde belirli satırları seçmek için istedigimiz satır numaralarını arada iki nokta olacak şekilde yazıyoruz. Burada iki noktayı tekrar hatırlatmakta faydalı var. Python'da indeks numaraları sıfırdan başlıyor ve indeks sonadaki sayıyı kapsamıyor. Yani [2:5] indeks numaralarını yazdiğimda, 2 indeks numarası ile başlayan ve 4 indeks numarası ile biten satırlar seçilecek, 5 indeks numarasına sahip satır seçilmeyecektir. Indeks numaralarının sıfırdan başladığını düşünürsek yaptığımız seçim 3. sıradaki satırla başlayıp 5. sıradaki satırla sona erecektir. Sonuç olarak `iris[2:5]` yazdığımızda aşağıdaki şekilde çerçeve içinde belirtilmiş olan satırlar seçilecektir.

PyTorch for Deep Learning

Item	Length	Width	Height	Weight	Volume	Cost
1	10	10	10	10	1000	100
2	20	20	20	20	2000	200
3	30	30	30	30	3000	300
4	40	40	40	40	4000	400
5	50	50	50	50	5000	500

newly-arrived immigrants

Naar de voorstelling gaf de burgemeester Stenlyck verschillende beroepen aan de verschillende regio's en gemeenten. Deel I behelst verschillende beroepen van de landelijke regio's en deel II behelst verschillende beroepen van de stedelijke regio's.

	without	with_damal	with_juliette	background
ABP	3.29	3.29	3.30	3.33
BEP	5.46	5.40	4.69	5.34
BBP	25.79	15.25	18.75	16.39
BPB	45.55	45.55	45.25	44.33
DPB	27.00	34.00	29.00	28.00
DPBP	27.74	34.28	31.50	28.48
DPBPB	45.32	38.25	49.25	41.25
DPBPP	45.32	35.00	57.45	46.25

Változások először a hosszú ideig tartó visszavonulásban, majd rövid időn belül következnek. Ezért minden fejezet nyitásakor (pl. I. fejezet) fontos megemlíteni ezeket.

卷之三

Strecker (1997) 9997-3  
aridic 99.3  
ar. aridic 99.3  
ar. pedunc 99.3  
aridic 99.3  
aridic 99.3

Wanneer de ziel hielp prezenen katholieken rapporteerden dat zij waren tegelijk toe aan een aantal verschillende geloofsovertuigingen, van de katholieke tot de protestantse en zelfs atheïstische.

transferred 2000/1988/1  
MELVILLE 40.00000 30.00000 0.00000  
0.00 40.0 30.0 10.0 10.0

se asemăna cu unii membri ai organizației.

1990-1991  
1991-1992  
1992-1993

De asemenea se pot obține rezultate foarte bune și la o temperatură de 100°C, dacă se folosesc soluții de sulfat de zinc și sulfat de amoniu.

www.millennium.com

http://www.sciencedirect.com/science/journal/03781909

www.merriam-webster.com

Inde de borden facile voter sur n'importe quelle autre question générale. De tel un  
géniale une évidente suffisance. Toute une de utile pour voter et déclencher n'importe  
quelque chose que l'on peut voter.

```
model.fit([x_train, y_train], [y_train, x_train])
score = model.evaluate([x_test, y_test], [y_test, x_test])
print(score)
```

Digitized by srujanika@gmail.com

## PYTHON İLE VERİ BİLİMİ

```
hisseler.loc['ERN', 'acilis']  
95.0
```

Belirli değişkenlere ait tüm satırları seçirmek için satır ismi yazmadı bunun yerine önceli böümlerde gördüğümüz gibi : yazmamız yeterlidir.

```
hisseler.loc[:, ["en_dusuk", "en_yuksek"]]  
          en_dusuk  en_yuksek  
ABC        3.15    3.50  
DEF        5.00    6.00  
KLM      15.25   16.75  
XYZ      75.00   85.25  
IJK      16.00   19.00  
VYZ      24.10   27.50  
ERN      90.20   99.50  
HEA      53.00   57.40
```

Yukarıda tüm satırlar için mutlaka : kullanmamız gerektigine dikkat edin. Bunun yerine sadece hisseler.loc[["en\_dusuk", "en\_yuksek"]]] yazarsanız hata mesajı ile karşılaşırınız.

Satır ve sütün isimleri yerine indeks numaraları ile seçim yapmak için iloc fonksiyonu kullanılır. Bu kullanım şeklinde tek fark satır ve sütunların isimleri yerine indeks numaralarını kullanmadıdır.

```
hisseler.iloc[1]  
acilis      5.48  
en_dusuk    5.00  
en_yuksek   6.00  
kapanis     6.00  
Name: DEF, dtype: float64
```

```
hisseler.iloc[[1]]  
      acilis  en_dusuk  en_yuksek  kapanis  
DEF      5.48       5.0       6.0       6.0
```

```
hisseler.iloc[6,0]  
95.0
```

```
hisseler.iloc[[1,2,3]]  
      acilis  en_dusuk  en_yuksek  kapanis  
DEF      5.48       5.0       6.0       6.0  
KLM      15.75     15.25     16.75     16.5  
XYZ      80.20     75.00     85.25     84.3
```

Yine iki adet köşeli parantez kullandığımıza dikkat edin. Bir tane köşeli parantez kullanmadı halinde sonuç aşağıdaki gibi olacaktır.

	acilis	5.48
	en_dusuk	5.00
	en_yuksek	6.00
	kapanis	6.00
Name:	DEF	dtype: float64

Artık sunu biliyoruz ki sonucu yine veri çerçevesi olarak almak için çift köşeli parantez kullanmak gerekir.

Birden fazla satır ve sütunda seçim yapmak için yine loc fonksiyonunda olduğu şekilde ama bu defa satır ve sütun indeks numaralarını belirterek yazıyoruz.

```
hisseler.iloc[[1,2,3], [2,3]]  
          en_yuksek  kapanis  
DEF            6.00      6.0  
KLM          16.75     16.5  
XYZ          85.25     84.3
```

Sütun ismi ve index numaralarını bir arada kullanmak da mümkündür. Örneğin, ilk 4 sıradaki hissenin kapanış fiyatlarını görelim:

```
hisseler['kapanis'][0:4]  
ABC      3.15  
DEF      6.00  
KLM     16.50  
XYZ     84.30  
Name: kapanis, dtype: float64
```

Satır seçimine 0:4 yazdığımızda 0, 1, 2, ve 3. sıradaki yanı ilk dört sıradaki verileri çektiğimize dikkat edin. ":" işaretini satır veya sütun isimleriyle de kullanabiliriz.

```
hisseler.loc['ABC':'KLM', 'en_dusuk':'kapanis']  
          en_dusuk  en_yuksek  kapanis  
ABC      3.15      3.50      3.15  
DEF      5.00      6.00      6.00  
KLM     15.25     16.75     16.50
```

Satır ya da sütunları ters sırada yazdırma da mümkündür.

```
hisseler.loc['KLM':'ABC':-1]
```

## PYTHON İLE VERİ BİLİMLİ

	acilis	en_dusuk	en_yuksek	kapanis
KLM	15.75	15.25	16.75	16.50
DEF	5.48	5.00	6.00	6.00
ABC	3.25	3.15	3.50	3.15

Buraya kadar satır ve sütunlardan seçim yapmak için çok sayıda yöntem aktardık. Bu nalar kullanımların özellikle ilk kullanımlarda karıştırılması normaldir. Kolay başvuru için yukarıda aktardığımız satır ve sütün seçim yöntemlerini aşağıdaki tabloda özetliyoruz.

Kullanım	Sonuç
hisseler["kapanis"]	Sütunu bir boyutlu veri dizisi olarak alır
hisseler[["kapanis"]]	Sütunu pandas veri çerçevesi olarak alır
hisseler[["en_yuksek", "kapanis"]]	Sütunları pandas veri çerçevesi olarak alır
hisseler[2:5]	3. saturday (indeks no:2) başlayarak 5. saturday (indeks no:4) dahil olacak şekilde saturdayları alır
hisseler.loc["ERN"]	Satırı bir boyutlu veri dizisi olarak alır
hisseler.loc[["ERN"]]	Satırı pandas veri çerçevesi olarak alır
hisseler.loc[["ERN", "HEA"]]	Satırı pandas veri çerçevesi olarak alır
hisseler.loc[["ERN", "HEA"], ["en_yuksek"]]	Satır ve sütunları pandas veri çerçevesi olarak alır
hisseler.loc[:, ["en_dusuk", "en_yuksek"]]	İstenen sütunların tüm saturdayları alır
hisseler.iloc[[1]]	İndeks numarası 1 olan (2. sıradaki) satırı alır
hisseler.iloc[[1,2,3], [2,3]]	İndeks numarası 1,2 ve 3 olan saturdaylar ile 2,3 olan sütunlar

Bir pandas veri çerçevesine for döngüsü uygularsak ne olur? Yukarıda kullandığımız hisseler veri çerçevesine for döngüsü uygulayalım.

```
for i in hisseler:
    print(i)

acilis
en_dusuk
en_yuksek
kapanis
```

Göründüğü gibi sadece değişken isimleri yazdırılıyor. Bir pandas veri çerçevesindeki bilgilere erişmek için pandas paketinde yer alan `.iterrows()` metodu kullanılır. Şimdi bu metodunu kullanarak bir for döngüsü yazalım.

```
for hisse, satır in hisseler.iterrows():
    print(hisse)
    print(satır)
```

## PANDAS PAKETİ

```
ABC
acilis      3.25
en_dusuk   3.15
en_yuksek  3.50
kapanis    3.15
Name: ABC, dtype: float64

DEF
acilis      5.48
en_dusuk   5.00
en_yuksek  6.00
kapanis    6.00
Name: DEF, dtype: float64
```

Diyelim ki her defasında sadece belirli bir sütundaki verilere erişmek istiyoruz.

```
for hisse, satır in hisseler.iterrows():
    print(hisse + " : " + str(satır["kapanis"]))
ABC : 3.15
DEF : 6.0
KLM : 16.5
XYZ : 84.3
IJK : 18.2
VYZ : 26.8
ERN : 97.5
HEA : 59.3
```

For döngüsü kullanarak pandas veri çerçevesine yeni bir sütun da ekleyebiliriz. Şimdi her bir hissenin günlük açılış fiyatı ile kapanış fiyatı arasındaki farkı alarak `gündük_fark` isminde bir sütun oluşturmak istediğimizi düşünelim. Bunu aşağıdaki şekilde kolayca gerçekleştirebiliriz.

```
for hisse, satır in hisseler.iterrows():
    hisseler.loc[hisse, "gündük_fark"] = satır["kapanis"] - satır["acilis"]
print(hisseler)
```

	acilis	en_dusuk	en_yuksek	kapanis	gündük_fark
ABC	3.25	3.15	3.50	3.15	-0.10
DEF	5.48	5.00	6.00	6.00	0.52
KLM	15.75	15.25	16.75	16.50	0.75
XYZ	80.20	75.00	85.25	84.30	4.10
IJK	17.50	16.00	19.00	18.20	0.70

## PYTHON ile VERİ BİLİMI

VYE	25.70	24.10	27.50	26.80	1.10
ERN	95.00	90.20	99.50	97.50	2.50
SEA	55.00	53.00	57.40	59.30	4.30

Aslında, yukarıdaki tabloda döngüsünü, veri çerçevelerinde döngülerin nasıl çalıştığını görmek için yazdım. Ama yeni bir sütun eklemenin çok daha kolay ve hızlı bir yolu vardır. Aşağıdaki kod da yukarıdaki sonucun aynısını çok daha kısa bir sürede verecektir.

```
hisseler["gunluk_fark"] = hisseler["kapanis"] - hisseler["acilis"]
```

## 9.4. Pandas ile Veri Analizi

Bir veri kümesi ile çalışmaya başladan önce ilk olarak verilerin özelliklerine bakılır. Bu özellikle de genellikle minimum, maksimum, ortalama, standart sapma, medyan, %25'lik dilim, %75'lik dilim gibi değerlerdir. Pandas modülünde bu özelliklerin hepsini bir arada gösteren `.describe()` metodu mevcuttur. Şimdi yine iris veri setini bu metodu kullanarak inceleyelim<sup>16</sup>.

```
import pandas as pd
iris.describe()
   Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count    150.000000    150.000000    150.000000    150.000000
mean     5.843333    3.057333    3.758000    1.199333
std      0.828066    0.435866    1.765298    0.762238
min      4.300000    2.000000    1.000000    0.100000
25%     5.100000    2.800000    1.600000    0.300000
50%     5.800000    3.000000    4.350000    1.300000
75%     6.400000    3.300000    5.100000    1.800000
max     7.900000    4.400000    6.900000    2.500000
```

Her bir sütun için yukarıdan aşağıya sırasıyla count (veri sayısı), mean (ortalama), std (standart sapma), min (en düşük değer), 25% (%25 yüzdelik), 50% (medyan), 75% (%75 yüzdelik) ve max (en yüksek değer) görülmektedir.

Yukarıdaki tabloda sadece sayısal verilere ilişkin özel bilgileri görüyoruz. Ancak iris veri çerçevesinde bir de Species yani tür bilgisini içeren bir sütun bulunuyor. Bu sütuna ait özel bilgileri de `.describe()` metodu ile görebiliyoruz. Aşağıda yer alan sonucu incelediğimizde

16 Iris veri setini indirmeyi ileride göreceğiz ama veri setini kullanmaya başlamak için aşağıdaki kodları kullanabilirsiniz.

```
import seaborn as sns
iris = sns.load_dataset('iris')
```

## PANDAS PAKETİ

bu sütunda 3 farklı değer olduğunu anlıyoruz. Bunlardan en sık görülenin setosa türü olduğu belirtilmiş. Bu kısma dikkat etmek gerekiyor. Asında iris veri setinde her bir tür ait (setosa, versicolor, virginica) gözlem var. Ancak ilk sırada setosa yer aldığı için top (en çok) görülenin setosa olduğu belirtiliyor. Freq bilgisi ise en sık görülen tür ait kaç gözlem olduğunu belirtiyor.

```
iris['species'].describe()
   species        150
  count           150
 unique          3
       setosa
  top            50
 freq: Species, dtype: object
```

Yukarıdaki gibi sınıf, tür, kategori belirten değişkenler kategorik değişkenler olarak adlandırılır. Veri biliminde sıklıkla kategorik değişkenlerle çalışmamız gereklidir. Bir sütunda hangi kategorik değişkenlerin yer aldığıni görmek için `.unique()` metodunu kullanabiliriz. Bu metod, özellikle çok büyük veri setlerinde oldukça kullanışlı olabilir.

```
iris['species'].unique()
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

Sayısal bilgilere ilişkin ortalama, standart sapma vb özel değerleri ayrı ayrı hesaplamak için de Pandas metodları bulunmaktadır.

```
iris.count()
   Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count    150          150          150          150
Species          150
dtype: int64
```

Sadece istediğimiz sütunları seçerek bunlara ilişkin değerleri de görebiliriz. Örneğin, Petal\_Length ve Petal\_Width sütunlarının veri sayısını görmek istiyorsak `iris[['Petal_Length', 'Petal_Width']].count()` yazmamız gereklidir. Buradaki çift köşeli parantez yapısına dikkat edin. Bu şekilde yazıyoruz çünkü istediğimiz sütunları liste olarak iletiyoruz. Bunu daha iyi anlamak için aşağıdaki şekilde de yazabiliriz.

```
veriler = ['petal_length', 'petal_width']
iris[veriler].count()
```

Aynı şekilde diğer değerleri hesaplamak için de aşağıda belirtilen metodlar kullanılır.

Hesaplama	Metod
Ortalama	.mean()
Standart sapma	.std()
Medyan	.median()
Yüzdelik dilim	.quantile(y)

```
iris.std()
Sepal_Length    0.828066
Sepal_Width     0.435866
Petal_Length    1.765298
Petal_Width     0.762238
dtype: float64
```

Gösterilen metodalar arasında `.quantile()` metodunu açıklamamız gerekiyor. Bu metodu kullanırken hangi yüzdelik dilimi istediğimizi, ondalık sayı şeklinde belirtmemiz gerekiyor. Örneğin, %25'lik dilimi hesaplamak için `.quantile(0.25)` yazmamız gereklidir.

```
iris.quantile(0.25)
Sepal_Length    5.1
Sepal_Width     2.8
Petal_Length    1.6
Petal_Width     0.3
Name: 0.25, dtype: float64

iris.quantile([0.25, 0.75])
   sepal_length  sepal_width  petal_length  petal_width
0.25          5.1         2.8           1.6          0.3
0.75          6.4         3.3           5.1          1.8
```

Herhangi bir sütuna ait bir değeri görmek için bu sütunu seçip istediğimiz metodu uygulayabiliriz.

```
iris['petal_length'].min()
1.0
iris['petal_width'].max()
2.5
```

Şimdiye kadar gördüğümüz metodları, sütunların yanı sıra satırlara da uygulayabiliriz. Yani satır bazında minimum, maximum, ortalama vb. gibi değerleri hesaplayabiliyoruz. Bu nedenle uyguladığımız metod içinde `axis='columns'` argümanını belirtmemiz gereklidir.

```
iris.mean(axis='columns')
0    2.550
```

```
2.375
1    2.350
2    2.350
3    2.550
4    2.850
5    2.425
6    2.525
7    ...
iris.std(axis='columns')
0    2.179449
1    2.036950
2    1.997498
3    1.912241
4    2.156386
5    2.230844
6    1.936276
7    2.109305
...
```

Iris veri setinin üç farklı türde ait gözlemlerden meydana geldiğini daha önce belirtmiştık. Şimdiye kadar verdigimiz örneklerde her üç türde ait gözlemleri bir arada değerlendirdik ve tür farklılıklarını dikkate almadık. Peki ya her bir türde ait istatistiksel değerleri ayrı ayrı hesaplamak istersek ne yapacağız? Bunun için mantıksal seçim kurallarını uygulayabiliriz. Varsayılmak üzere versicolor türüne ait özel bilgileri görmek istiyoruz. Öncelikle, versicolor türüne ait gözlemleri seçip sonra da yaptığım seçime istediğim metodu uygulayabiliriz.

Önce versicolor türüne ait gözlemlerin hangi satırlarda yer aldığıni görelim. Aşağıda, `kosul` olarak belirttiğimiz değişken bize False veya True değerlerden oluşan bir seri verecek.

```
kosul = iris['species'] == 'versicolor'
type(kosul)
pandas.core.series.Series
kosul
0    False
1    False
2    False
3    False
4    False
5    False
...

```

Şimdi, bu kosul değerini kullanarak iris veri çerçevesinde sadece versicolor türlerinin olduğu satırları seçebiliriz.

## PYTHON DE VERS BILDE

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	5.4	3.0	4.5	0.3	Iris-setosa
2	5.9	3.0	4.2	0.3	Iris-setosa
3	6.4	2.8	4.5	0.2	Iris-setosa
4	6.9	3.1	4.9	0.2	Iris-setosa
5	7.3	3.3	5.0	0.1	Iris-setosa
6	7.9	3.7	5.4	0.4	Iris-setosa
7	8.7	3.0	6.4	0.3	Iris-setosa
8	9.2	2.2	7.0	0.2	Iris-setosa
9	9.5	2.4	7.9	0.2	Iris-setosa
10	10.0	2.9	8.0	0.1	Iris-setosa
11	10.4	3.0	8.1	0.1	Iris-setosa
12	10.9	3.4	8.6	0.2	Iris-setosa
13	11.3	3.0	9.0	0.2	Iris-setosa
14	11.8	3.3	9.6	0.1	Iris-setosa
15	12.0	3.0	9.8	0.2	Iris-setosa
16	12.2	3.4	10.0	0.2	Iris-setosa
17	12.5	3.0	10.2	0.2	Iris-setosa
18	12.9	3.4	10.4	0.2	Iris-setosa
19	13.4	3.0	10.6	0.2	Iris-setosa
20	13.9	3.4	10.8	0.2	Iris-setosa
21	14.3	3.0	11.0	0.2	Iris-setosa
22	14.7	3.4	11.2	0.2	Iris-setosa
23	15.1	3.0	11.4	0.2	Iris-setosa
24	15.5	3.4	11.6	0.2	Iris-setosa
25	15.9	3.0	11.8	0.2	Iris-setosa
26	16.3	3.4	12.0	0.2	Iris-setosa
27	16.7	3.0	12.2	0.2	Iris-setosa
28	17.1	3.4	12.4	0.2	Iris-setosa
29	17.5	3.0	12.6	0.2	Iris-setosa
30	17.9	3.4	12.8	0.2	Iris-setosa
31	18.3	3.0	13.0	0.2	Iris-setosa
32	18.7	3.4	13.2	0.2	Iris-setosa
33	19.1	3.0	13.4	0.2	Iris-setosa
34	19.5	3.4	13.6	0.2	Iris-setosa
35	19.9	3.0	13.8	0.2	Iris-setosa
36	20.3	3.4	14.0	0.2	Iris-setosa
37	20.7	3.0	14.2	0.2	Iris-setosa
38	21.1	3.4	14.4	0.2	Iris-setosa
39	21.5	3.0	14.6	0.2	Iris-setosa
40	21.9	3.4	14.8	0.2	Iris-setosa
41	22.3	3.0	15.0	0.2	Iris-setosa
42	22.7	3.4	15.2	0.2	Iris-setosa
43	23.1	3.0	15.4	0.2	Iris-setosa
44	23.5	3.4	15.6	0.2	Iris-setosa
45	23.9	3.0	15.8	0.2	Iris-setosa
46	24.3	3.4	16.0	0.2	Iris-setosa
47	24.7	3.0	16.2	0.2	Iris-setosa
48	25.1	3.4	16.4	0.2	Iris-setosa
49	25.5	3.0	16.6	0.2	Iris-setosa
50	25.9	3.4	16.8	0.2	Iris-setosa
51	26.3	3.0	17.0	0.2	Iris-setosa
52	26.7	3.4	17.2	0.2	Iris-setosa
53	27.1	3.0	17.4	0.2	Iris-setosa
54	27.5	3.4	17.6	0.2	Iris-setosa
55	27.9	3.0	17.8	0.2	Iris-setosa
56	28.3	3.4	18.0	0.2	Iris-setosa
57	28.7	3.0	18.2	0.2	Iris-setosa
58	29.1	3.4	18.4	0.2	Iris-setosa
59	29.5	3.0	18.6	0.2	Iris-setosa
60	29.9	3.4	18.8	0.2	Iris-setosa
61	30.3	3.0	19.0	0.2	Iris-setosa
62	30.7	3.4	19.2	0.2	Iris-setosa
63	31.1	3.0	19.4	0.2	Iris-setosa
64	31.5	3.4	19.6	0.2	Iris-setosa
65	31.9	3.0	19.8	0.2	Iris-setosa
66	32.3	3.4	20.0	0.2	Iris-setosa
67	32.7	3.0	20.2	0.2	Iris-setosa
68	33.1	3.4	20.4	0.2	Iris-setosa
69	33.5	3.0	20.6	0.2	Iris-setosa
70	33.9	3.4	20.8	0.2	Iris-setosa
71	34.3	3.0	21.0	0.2	Iris-setosa
72	34.7	3.4	21.2	0.2	Iris-setosa
73	35.1	3.0	21.4	0.2	Iris-setosa
74	35.5	3.4	21.6	0.2	Iris-setosa
75	35.9	3.0	21.8	0.2	Iris-setosa
76	36.3	3.4	22.0	0.2	Iris-setosa
77	36.7	3.0	22.2	0.2	Iris-setosa
78	37.1	3.4	22.4	0.2	Iris-setosa
79	37.5	3.0	22.6	0.2	Iris-setosa
80	37.9	3.4	22.8	0.2	Iris-setosa
81	38.3	3.0	23.0	0.2	Iris-setosa
82	38.7	3.4	23.2	0.2	Iris-setosa
83	39.1	3.0	23.4	0.2	Iris-setosa
84	39.5	3.4	23.6	0.2	Iris-setosa
85	39.9	3.0	23.8	0.2	Iris-setosa
86	40.3	3.4	24.0	0.2	Iris-setosa
87	40.7	3.0	24.2	0.2	Iris-setosa
88	41.1	3.4	24.4	0.2	Iris-setosa
89	41.5	3.0	24.6	0.2	Iris-setosa
90	41.9	3.4	24.8	0.2	Iris-setosa
91	42.3	3.0	25.0	0.2	Iris-setosa
92	42.7	3.4	25.2	0.2	Iris-setosa
93	43.1	3.0	25.4	0.2	Iris-setosa
94	43.5	3.4	25.6	0.2	Iris-setosa
95	43.9	3.0	25.8	0.2	Iris-setosa
96	44.3	3.4	26.0	0.2	Iris-setosa
97	44.7	3.0	26.2	0.2	Iris-setosa
98	45.1	3.4	26.4	0.2	Iris-setosa
99	45.5	3.0	26.6	0.2	Iris-setosa
100	45.9	3.4	26.8	0.2	Iris-setosa
101	46.3	3.0	27.0	0.2	Iris-setosa
102	46.7	3.4	27.2	0.2	Iris-setosa
103	47.1	3.0	27.4	0.2	Iris-setosa
104	47.5	3.4	27.6	0.2	Iris-setosa
105	47.9	3.0	27.8	0.2	Iris-setosa
106	48.3	3.4	28.0	0.2	Iris-setosa
107	48.7	3.0	28.2	0.2	Iris-setosa
108	49.1	3.4	28.4	0.2	Iris-setosa
109	49.5	3.0	28.6	0.2	Iris-setosa
110	49.9	3.4	28.8	0.2	Iris-setosa
111	50.3	3.0	29.0	0.2	Iris-setosa
112	50.7	3.4	29.2	0.2	Iris-setosa
113	51.1	3.0	29.4	0.2	Iris-setosa
114	51.5	3.4	29.6	0.2	Iris-setosa
115	51.9	3.0	29.8	0.2	Iris-setosa
116	52.3	3.4	30.0	0.2	Iris-setosa
117	52.7	3.0	30.2	0.2	Iris-setosa
118	53.1	3.4	30.4	0.2	Iris-setosa
119	53.5	3.0	30.6	0.2	Iris-setosa
120	53.9	3.4	30.8	0.2	Iris-setosa
121	54.3	3.0	31.0	0.2	Iris-setosa
122	54.7	3.4	31.2	0.2	Iris-setosa
123	55.1	3.0	31.4	0.2	Iris-setosa
124	55.5	3.4	31.6	0.2	Iris-setosa
125	55.9	3.0	31.8	0.2	Iris-setosa
126	56.3	3.4	32.0	0.2	Iris-setosa
127	56.7	3.0	32.2	0.2	Iris-setosa
128	57.1	3.4	32.4	0.2	Iris-setosa
129	57.5	3.0	32.6	0.2	Iris-setosa
130	57.9	3.4	32.8	0.2	Iris-setosa
131	58.3	3.0	33.0	0.2	Iris-setosa
132	58.7	3.4	33.2	0.2	Iris-setosa
133	59.1	3.0	33.4	0.2	Iris-setosa
134	59.5	3.4	33.6	0.2	Iris-setosa
135	59.9	3.0	33.8	0.2	Iris-setosa
136	60.3	3.4	34.0	0.2	Iris-setosa
137	60.7	3.0	34.2	0.2	Iris-setosa
138	61.1	3.4	34.4	0.2	Iris-setosa
139	61.5	3.0	34.6	0.2	Iris-setosa
140	61.9	3.4	34.8	0.2	Iris-setosa
141	62.3	3.0	35.0	0.2	Iris-setosa
142	62.7	3.4	35.2	0.2	Iris-setosa
143	63.1	3.0	35.4	0.2	Iris-setosa
144	63.5	3.4	35.6	0.2	Iris-setosa
145	63.9	3.0	35.8	0.2	Iris-setosa
146	64.3	3.4	36.0	0.2	Iris-setosa
147	64.7	3.0	36.2	0.2	Iris-setosa
148	65.1	3.4	36.4	0.2	Iris-setosa
149	65.5	3.0	36.6	0.2	Iris-setosa
150	65.9	3.4	36.8	0.2	Iris-setosa
151	66.3	3.0	37.0	0.2	Iris-setosa
152	66.7	3.4	37.2	0.2	Iris-setosa
153	67.1	3.0	37.4	0.2	Iris-setosa
154	67.5	3.4	37.6	0.2	Iris-setosa
155	67.9	3.0	37.8	0.2	Iris-setosa
156	68.3	3.4	38.0	0.2	Iris-setosa
157	68.7	3.0	38.2	0.2	Iris-setosa
158	69.1	3.4	38.4	0.2	Iris-setosa
159	69.5	3.0	38.6	0.2	Iris-setosa
160	69.9	3.4	38.8	0.2	Iris-setosa
161	70.3	3.0	39.0	0.2	Iris-setosa
162	70.7	3.4	39.2	0.2	Iris-setosa
163	71.1	3.0	39.4	0.2	Iris-setosa
164	71.5	3.4	39.6	0.2	Iris-setosa
165	71.9	3.0	39.8	0.2	Iris-setosa
166	72.3	3.4	40.0	0.2	Iris-setosa
167	72.7	3.0	40.2	0.2	Iris-setosa
168	73.1	3.4	40.4	0.2	Iris-setosa
169	73.5	3.0	40.6	0.2	Iris-setosa
170	73.9	3.4	40.8	0.2	Iris-setosa
171	74.3	3.0	41.0	0.2	Iris-setosa
172	74.7	3.4	41.2	0.2	Iris-setosa
173	75.1	3.0	41.4	0.2	Iris-setosa
174	75.5	3.4	41.6	0.2	Iris-setosa
175	75.9	3.0	41.8	0.2	Iris-setosa
176	76.3	3.4	42.0	0.2	Iris-setosa
177	76.7	3.0	42.2	0.2	Iris-setosa
178	77.1	3.4	42.4	0.2	Iris-setosa
179	77.5	3.0	42.6	0.2	Iris-setosa
180	77.9	3.4	42.8	0.2	Iris-setosa
181	78.3	3.0	43.0	0.2	Iris-setosa
182	78.7	3.4	43.2	0.2	Iris-setosa
183	79.1	3.0	43.4	0.2	Iris-setosa
184	79.5	3.4	43.6	0.2	Iris-setosa
185	79.9	3.0	43.8	0.2	Iris-setosa
186	80.3	3.4	44.0	0.2	Iris-setosa
187	80.7	3.0	44.2	0.2	Iris-setosa
188	81.1	3.4	44.4	0.2	Iris-setosa
189	81.5	3.0	44.6	0.2	Iris-setosa
190	81.9	3.4	44.8	0.2	Iris-setosa
191	82.3	3.0	45.0	0.2	Iris-setosa
192	82.7	3.4	45.2	0.2	Iris-setosa
193	83.1	3.0	45.4	0.2	Iris-setosa
194	83.5	3.4	45.6	0.2	Iris-setosa
195	83.9	3.0	45.8	0.2	Iris-setosa
196	84.3	3.4	46.0	0.2	Iris-setosa
197	84.7	3.0	46.2	0.2	Iris-setosa
198	85.1	3.4	46.4	0.2	Iris-setosa
199	85.5	3.0	46.6	0.2	Iris-setosa
200	85.9	3.4	46.8	0.2	Iris-setosa
201	86.3	3.0	47.0	0.2	Iris-setosa
202	86.7	3.4	47.2	0.2	Iris-setosa

## PYTHON İLE VERİ BİLİMLİ

```
131      7.9      3.8      5.4      2.0      2
135      7.7      3.0      6.1      2.3      2
```

Bir sütun verilerine koşul uygulayarak bir başka sütundan veri çekmek de mümkündür. Örneğin, `sepal_length` değeri 7.5'tan büyük olan satırların `petal_length` değerlerini görelim.

```
iris.petal_length[iris.sepal_length > 7.5]
105   6.6
117   6.7
118   6.9
122   6.7
131   6.4
135   6.1
Name: petal_length, dtype: float64
```

Buraya kadar anlattıklarımız dışında Pandas modülünde kullanılan farklı metodlar da vardır. Örneğin bir veri çerçevesinde hiç sıfır içermeyen sütunları görmek için `.all()`, sütündan farklı değerler içeren sütunları görmek için `.any()`, NaN değeri içeren sütunları görmek için `.isnull()`, içermeyenleri görmek için `.notnull()` metodları mevcuttur.

Son olarak Pandas veri çerçevelerinde gerçekleştirilebilecek yeniden şekillendirme ve manipülasyon işlemlerine göz atalım. Örneklerimizde kullanmak üzere aşağıdaki df veri çerçevesini oluşturalım.

```
import pandas as pd
import numpy as np

degisken = np.repeat(['A', 'B', 'C', 'D'], [3, 3, 3, 3], axis=0)
deger = np.random.random(12)
df_dict = {'degisken': degisken, 'deger': deger}
df = pd.DataFrame(df_dict)
df = df[['degisken', 'deger']]
print(df)

degisken      deger
0           A  0.098552
1           A  0.116679
2           A  0.459386
3           B  0.161051
4           B  0.166573
5           B  0.337498
6           C  0.611188
```

## PANDAS PAKETİ

	C	0.339547
7	C	0.913746
8	D	0.626261
9	D	0.191192
10	D	0.148170
11		

Şimdi bu veri çerçevesini öyle değiştirelim ki tarih sütunu yine kalsın ama sütun isimleri A, B, C, D değişken isimleri olsun. Bunun için kullanabileceğimiz Pandas metodlarından birisi `pivot()` metodudur. Excel'de pivot işlemi yapanlar bu metodun nasıl çalıştığını tahlimin edebilirler.

```
df2 = df.pivot(columns='degisken', values='deger')
print(df2)
degisken      A      B      C      D
0    0.098552  NaN  NaN  NaN
1    0.116679  NaN  NaN  NaN
2    0.459386  NaN  NaN  NaN
3    NaN  0.161051  NaN  NaN
4    NaN  0.166573  NaN  NaN
5    NaN  0.337498  NaN  NaN
6    NaN  NaN  0.611188  NaN
7    NaN  NaN  0.339547  NaN
8    NaN  NaN  0.913746  NaN
9    NaN  NaN  NaN  0.626261
10   NaN  NaN  NaN  0.191192
11   NaN  NaN  NaN  0.148170
```

Sütunları tekrar satır şeklinde yazmak için `.melt()` metodunu kullanabiliriz.

```
df3 = df2.melt(value_vars=['A', 'B', 'C', 'D'], value_name='deger').
dropna()
print(df3)

degisken      deger
0           A  0.789559
1           A  0.820599
2           A  0.646496
15          B  0.747331
16          B  0.543986
17          B  0.329257
30          C  0.211563
31          C  0.316435
32          C  0.629631
```

## PYTHON ile VERİ BİLİMİ

```
45      D  0.153545
46      D  0.268958
47      D  0.431187
```

İki veri çerçevesini anahtar bir sütun kullanarak birleştirmek için `.merge()` fonksiyonu kullanılır. Örnek olarak aşağıdaki iki Pandas veri çerçevesini ele alalım.

	X	Y1
0	Ali	97
1	Baran	85
2	Mehmet	76

  

	X	Y2
0	Ali	75
1	Baran	94
2	Umut	96

Veri çerçevelerinin isimleri sırasıyla df1 ve df2 olsun. İki veri çerçevesini birleştirirken kullanabileceğimiz yöntemler, 'left', 'right', 'inner' ve 'outer' yöntemleridir. Aşağıdaki örneklere bu yöntemleri kullanırken önce df1 ve sonra df2 yazdığımızı dikkat edin. Bu, özellikle 'left' ve 'right' yöntemleri için önemlidir.

İlk yani 'left' yönteminde, önce yazılan veri çerçevesi esas alınır. Argüman olarak birleştirilecek iki veri çerçevenin isimleri, birleştirme yöntemi (how) ve veri çerçeveleri birleştirilirken sütunun dikkate alınacağı (on) belirtilir.

```
df3 = pd.merge(df1, df2, how='left', on='X')
print(df3)
      X   Y1   Y2
0  Ali  97.0  75.0
1 Baran  85.0  94.0
2 Mehmet  76.0    NaN
```

  

```
df4 = pd.merge(df1, df2, how = 'right', on='X')
print(df4)
      X   Y1   Y2
0  Ali  97.0  75.0
1 Baran  85.0  94.0
2 Umut    NaN  96.0
```

Left ve right yöntemlerinin çalışma mantığı benzerdir. Diğer yandan, how = 'inner' seçildiğinde birleştirilecek sütunun sadece her iki veri çerçevesinde de yer alan satırları yani kesişim kümesi dikkate alınır.

## PANDAS PAKETİ

```
df5 = pd.merge(df1, df2, how='inner', on='X')
print(df5)
      X   Y1   Y2
0  Ali  97.0  75.0
1 Baran  85.0  94.0
```

Son olarak how='outer' seçtiğimizde de birleştirilecek sütundaki tüm satırlar yani bileşim kümesi dikkate alınır. Veri çerçevelerinden birinde olmayan değerler için NaN atanır.

```
df6 = pd.merge(df1, df2, how='outer', on='X')
print(df6)
      X   Y1   Y2
0  Ali  97.0  75.0
1 Baran  85.0  94.0
2 Mehmet  76.0    NaN
3  Umut    NaN  96.0
```