

10

## Veri Okuma

Veri bilimi ile uğraşıyorsanız zamanınızın büyük çoğunu verileri temin etmek ve bu verileri temizleyip düzenleyerek analize hazır hale getirmek olacaktır. Bu kısım, veri analizinin havalı tarafı olmadığı için genelde çok fazla söz konusu edilmez ama aslında iyi veri bilimcisi olmanın en temel şartı veri okuma ve temizleme yani verileri analize hazır hale getirme işini doğru yapmaktadır. Günlük hayatı, gerek profesyonel gerekse akademik çalışmalarında neredeyse hiçbir zaman veriler size analize hazır biçimde gelmez. Çoğu zaman verileri Python dışında bir programdan almanız gereklidir. Yeni bir işte ya da projede çalışmaya başladığınızda sizden önceki çalışmalar farklı bir programda yapılmış olabilir. Ya da sıfırdan bir çalışmaya başlasanız bile ihtiyaç duyduğunuz veriler çoğu zaman Python dışında bir formatta gelecektir. Bu nedenle veri biliminin ilk şartı çalışacağınız verileri analize hazır hale getirmeyi bilmektir.

Veri kaynaklarını aşağıdaki şekilde sınıflandırmamız mümkündür:

- Düz dosyalar: Bunlar düz metin (txt), ya da tablo formatında (csv, excel vb.) olabilir
- Veri tabanı sistemleri: SQL tabanlı sistemlerdir. Bunlar SQL, MySQL, PostgreSQL gibi farklı SQL tipleri olabilir.
- HTML: Web kaynaklı dosyalar
- İstatistiksel programlar: Genelde ticari amaçlı kullanılan ve ücretli olan SAS, STATA veya SPSS gibi dosyalar.

## 10.1. Düz Dosyalardan Veri Okuma

Düz dosyalar genel olarak .txt, .csv gibi tablo formattunda veri içeren dosyalardır. Veri analizi açısından bakacak olursak çalışmamızda gerçekleştirdiğimiz analizlerin çoğu satır ve sütunlardan oluşan tablolar formatındadır. Bu tablolarda bir satır bir veriyi ya da gözleme belirtken sütunlar da bu gözleme ilişkin farklı özellikleri ya da değişkenleri belirtir. Daha önceki bölgelerde gördüğümüz aşağıdaki iris veri setine bakalım. Toplam 150 gözlemden oluşan bu veri setinde her bir satır farklı bir çiçeğe ait ölçümü göstermektedir. Bu ölçülerin neler olduğunu da sütunlarda görüyoruz. Veri analizinde metin, csv ya da excel formatta dosyalarдан okuduğumuz veriler genellikle bu formatta yapılandırılmıştır.

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
1 5.1	3.5	1.4	0.2	setosa
2 4.9	3	1.4	0.2	setosa
3 4.7	3.2	1.3	0.2	setosa
4 4.6	3.1	1.5	0.2	setosa
5 5	3.6	1.4	0.2	setosa
—				
146 6.7	3	5.2	2.3	virginica
147 6.3	2.5	5	1.9	virginica
148 6.5	3	5.2	2	virginica
149 6.2	3.4	5.4	2.3	virginica
150 5.9	3	5.1	1.8	virginica

Şekil 10-1: Düz veri yapısı

### 10.1.1. Metin Dosyalarından Veri Okuma

Metin dosyaları .txt uzantısına sahip düz metin formattında olan dosyalardır. Bir metin dosyasından veri okumak için önce dosyaya bağlantı sağlanmalıdır. Python, `open()` fonksiyonu bu bağlantıyı sağlar:

```
baglanti = open('iris.txt', mode = 'r')
```

Fonksiyondaki ilk argüman dosya ismi, ikinci argüman ise dosyanın hangi modda okunacağıdır. Dosyadan sadece veri okumak için bu argümanı '`r`' (read) olarak belirtiyoruz. Bağlantıya okuma modunda kurarak hám veride herhangi bir değişiklik yapmanın önüne geçmiş oluyoruz. Bağlantı sırasında dosyaya veri yazmak için bu argümanı '`w`' (write) yapmamız gereklidir. Bağlantı dosyاسının bulunduğu dizin yolunu da açıkça yazmak gereklidir: "C:/Users/ilker/Desktop/iris.txt" gibi.

Dosyadan veri okumak için sadece bağlantı kurmak yetmez. Şimdi da dosyadan veri okuyalıız. Bunun için de bağlantı üzerinde `.read()` metodunu kullanıyoruz. `veri = baglanti.read()`. Son olarak kurduğumuz bağlantıyı kapatmamız gerekiyor: `baglanti.close()`

Şimdi bütün işlemlerin bir arada görelim.

```
baglanti = open('C:/Users/.../Desktop/iris.txt', mode = 'r')
veri = baglanti.read()
baglanti.close()
print(veri)
5.1 3.5 1.4 0.2 setosa
4.9 3 1.4 0.2 setosa
4.7 3.2 1.3 0.2 setosa
4.6 3.1 1.5 0.2 setosa
5 3.6 1.4 0.2 setosa
5.4 3.9 1.7 0.4 setosa
4.6 3.4 1.4 0.3 setosa
...
```

Aynı işlemi `with` fonksiyonu ile de yapabiliriz. Bu şekilde `.close()` metodunu kullanma da gereklidir.

```
with open('C:/Users/.../Desktop/iris.txt', mode = 'r') as baglanti:
    print(baglanti.read())
```

Okuyacağınız dosya çok büyükse dosyanın tamamını tek seferde okumak zor olabilir. Bu nedenle bağlantı kurduktan sonra dosya, satır satır okunabilir. Dosyadan verileri satır satır okumak için `.readline()` metodunu kullanabiliriz.

```
with open('C:/Users/... /Desktop/iris.txt', mode = 'r') as baglanti:
    print(baglanti.readline())
    print(baglanti.readline())
    5.1 3.5 1.4 0.2 setosa
    4.9 3 1.4 0.2 setosa
```

Önceki bölgelerde Python'da veri analizi olan iki paketi, NumPy ve Pandas paketlerini görmüştük. Bu paketlerde de verileri kendi formatlarına uygun olarak okumak için gerekli fonksiyonlar yer almaktadır. Böylece okuyacağınız verileri hangi pakette analiz edecekse o paketin ilgili fonksiyonlarını kullanarak aktarabiliriz.

### 10.1.2. NumPy Modülü ile Veri Okuma

İlgili bölümde de gördüğümüz üzere NumPy modülü özellikle sayısal verileri okumak ve analiz etmek için çok kullanışlı bir pakettir. NumPy'da düz dosya okumak için kullanılan iki fonksiyon `loadtxt()` ve `genfromtxt()` fonksiyonlardır.

## PYTHON ile VERİ BİLİMİ

```
import numpy as np
dosya = 'C:/Users/.../veri_dosyasi.txt'
veri = np.loadtxt(dosya, delimiter=",")

print(veri)
[[0.0000e+00 5.0000e+03 2.4000e+04 3.3000e+01]
 [0.0000e+00 2.4000e+03 1.22520e+04 3.1000e+01]
 [0.0000e+00 1.0000e+04 4.9200e+04 2.4000e+01]
 ...
 [0.0000e+00 5.0000e+03 1.0000e+05 2.7000e+01]
 [0.0000e+00 5.0000e+03 2.0000e+05 2.3000e+01]
 [0.0000e+00 7.5000e+03 2.2000e+04 2.5000e+01]]
```

Loadtxt fonksiyonundaki ikinci argüman olan delimiter, bir satırda verilerin birbirle- rinden hangi işaretle ayrıldıklarını belirtir. Bizim dosyamızda veriler virgül ile ayrılmış olduğu için argümanda bunu belirtiyoruz. Veriler, örneğin sekme yani tab ile ayrılmışsa virgül yerine 't' yazmamız gerekiirdi. Çoğu zaman olduğu gibi dosyada ilk satırda sütun başlıklar yer almaktak ise bu durumda üçüncü bir argüman olarak skiprows kullanılır. Bu argüman, dosya başından itibaren kaç satırın atlanacağını belirtir.

```
veri = np.loadtxt(dosya, delimiter=",", skiprows = 1)
```

Kullanışlı bir başka argüman da usecols argümanıdır. Bu da orijinal dosyadan hangi sütunların dikkate alınacağını gösterir. Örneğin, sadece 3,4 ve 5. sütunlardan veri almak istiyorsanz,

```
veri = np.loadtxt(dosya, delimiter=",", usecols=[2,3,4])
```

yazmanız gerekiir. Python'da sıralamanın sıfırdan başladığını tekrar hatırlayalım. Dosya- dan verilerin hangi veri tipinde okunacağını belirtmek için de dtype argümanı kullanılır. Örneğin, okuyacağınız dosyada metin türündeki veriler yer alabilir. Verileri metin olarak okumak için loadtxt fonksiyonunda dtype=str yazmanız gerekiir.

Çoğu zaman okuyacağımız tabloda farklı türdeki veri sütunları yer alabilir. Bu durumda .loadtxt() metodunu işe yaramayacaktır. Bunun yerine .genfromtxt() metodunu kullanabiliriz. Bu metoda, dtype=None olarak belirtirsek program her sütundaki veri tipini otomatik olarak belirlemeye çalışacaktır. NumPy dizileri aynı türde verilerden oluşmaktadır. Bu nedenle farklı türdeki verileri okuduğumuz zaman dizilerden oluşan bir dizi elde ederiz. Ana diziyi oluşturan diziler verinin her bir satırını belirtir.

```
import numpy as np
```

## VERİ OKUMA

```
dosya = 'C:/Users/.../Desktop/iris.txt'
veri = np.genfromtxt(dosya, delimiter="\t", names=True, dtype=None)
print(veri)

[(5.1, 3.5, 1.4, 0.2, b'setosa') (4.9, 3., 1.4, 0.2, b'setosa')
 (4.7, 3.2, 1.3, 0.2, b'setosa') (4.6, 3.1, 1.5, 0.2, b'setosa')
 (5., 3.6, 1.4, 0.2, b'setosa') (5.4, 3.9, 1.7, 0.4, b'setosa')
 (4.6, 3.4, 1.4, 0.3, b'setosa') (5., 3.4, 1.5, 0.2, b'setosa')
 (4.4, 2.9, 1.4, 0.2, b'setosa') (4.9, 3.1, 1.5, 0.1, b'setosa')
 (5.4, 3.7, 1.5, 0.2, b'setosa') (4.8, 3.4, 1.6, 0.2, b'setosa')
 (4.8, 3., 1.4, 0.1, b'setosa') (4.3, 3., 1.1, 0.1, b'setosa')
 (5.8, 4., 1.2, 0.2, b'setosa') (5.7, 4.4, 1.5, 0.4, b'setosa')
 ...]
```

Fonksiyonda yer alan names=True ifadesi, verinin ilk satırının sütün başlıklarından oluştuğunu belirtir. Csv formatındaki dosyaları okumak için kullanılan benzer bir diğer fonksiyon da `recfromcsv()` metodudur.

```
import numpy as np
dosya = 'C:/Users/.../iris.csv'
veri = np.recfromcsv(dosya, delimiter="\t", names=True, dtype=None)
print(veri)

[(b'5.1,3.5,1.4,0.2, setosa',) (b'4.9,3.1,1.4,0.2, setosa',)
 (b'4.7,3.2,1.3,0.2, setosa',) (b'4.6,3.1,1.5,0.2, setosa',)
 (b'5,3.6,1.4,0.2, setosa',) (b'5.4,3.9,1.7,0.4, setosa',)
 (b'4.6,3.4,1.4,0.3, setosa',) (b'5,3.4,1.5,0.2, setosa',)
 (b'4.4,2.9,1.4,0.2, setosa',) (b'4.9,3.1,1.5,0.1, setosa',)
 ...]
```

### 10.1.3. Pandas Modülü ile Veri Okuma

NumPy dizileri çok faydalı ve hızlı olsa da çalışmalarımızın çoğunda ihtiyaç duyduğumuz esnekliği sağlamaz. Gerçekte kullandığımız verilerin çoğu farklı veri tipindeki sütunlar- dan oluşur (tipik bir excel tablosunu düşünün). Pandas veri çerçevelerinden ilgili bölümde bahsetmiştik. Şimdi farklı türdeki dosyaları Pandas veri çerçevesi olarak okumayı görelim. Pandas modülünde csv dosyası okumak için `.read_csv()` metodunu kullanılır.

```
import pandas as pd
```

## PYTHON İLE VERİ BİLİMİ

```
dosya = 'C:/Users/.../iris.csv'
veri = pd.read_csv(dosya)

print(veri.head())
      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
```

Pandas veri çerçevesini, .values özelliğini kullanarak Numpy dizisine çevirebiliriz.

```
print(veri.values)
[[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']]
```

Read\_csv() metodunu çok sayıda argüman alabilir. Bunlardan birisi dosyadan okunacak satır sayısıdır. Dosyadan, örneğin sadece ilk 3 satırı okumak istiyorsak pd.read\_csv('dosya\_adi', nrows = 3) yazmamız gereklidir.

```
veri = pd.read_csv(dosya, nrows = 3)
print(veri)
      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
```

Benzer şekilde, .read\_csv komutu dosyadaki ilk satırın başlık satırı olduğunu varsayar. Dosyada başlık satırı yoksa ve biz herhangi birşey belirtmezse ilk satırındaki veriler sütun başlıklarını olarak kabul edilir. Dosyada sütun başlıklarının yer almadığını belirtmek için header=None yazılır. Bu bu durumda, sütunlar 0'dan başlamak üzere sayılarla isimlendirilir.

```
veri = pd.read_csv(dosya, header = None)
      0         1         2         3         4
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
```

## VERİ OKUMA

Dosyada başlık satırı yoksa ve biz başlıklarını belirtmek istiyorsak names argümanını kullanarak sütunları isimlendirebiliriz.

```
dosya = 'C:/Users/.../iris.csv'
sutunlar = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width', 'Species']
veri = pd.read_csv(dosya, header = None, names = sutunlar)
```

Varsayılmış ki dosyada sütun başlıklarları var ama biz bunların yerine başka isimler kullanmak istiyoruz. Bu durumda yine header argümanı ile sütun başlıklarının yer aldığı satırın 0'dan başladığını hatırlayın. O halde ilk satır numarası 0 ve kullanacağımız argümanın header = 0 olacaktır. İkinci olarak yine names argümanını kullanarak vereceğimiz yeni isimleri belirtebiliriz.

```
veri = pd.read_csv(dosya, header = 0, names = sutunlar)
```

Dosyada NA (not available: erişilemez) veya NaN (not a number: bir sayı değil) gibi tanimsız veriler belirli karakterlerle gösteriliyorsa bunları belirtmek için na\_values argümanı kullanılır. Örneğin, bir ankete ait sonuçların kaydedildiği dosyada katılımcıların cevap vermediği her soru \*\*\* karakteri ile belirtilmiş olsun. Bu durumda dosyayı doğrudan okumaktansa \*\*\* karakteri görülen yerde NA olması gerektiğini belirtmek için na\_values = "\*\*\*" yazılır. Peki tanimsız veriler farklı sütunlarda farklı karakterlerle belirtilmişse ne yaparız? Bu durumda, na\_values argümanına verileri sözlük şeklinde tanımlarız. Örneğin, doğum tarihi bilgisi girmeyen katılımcılar için bu alan 0 de doğum yeri bilgisi girmeyen katılımcılar içinse \*\*\* ile belirtiliyor olsun. Bu sütunlarda belirtilen karakterlerin görülmeleri durumunda bunların tanimsız olduğunu anlaşılması için na\_values = {'dogum\_tarihi': '0', 'dogum\_yeri': '\*\*\*'} yazarız.

Okunacak csv dosyasında sütun ayıracı virgülden farklı ise bu da sep veya delimiter argümanı ile belirtilir. Örneğin, dosyadaki sütunlar genelde olduğu gibi tab/sekme ile ayrılmışsa pd.read\_csv() metodunda sep='t' (veya delimiter='t') yazılır. Csv dosyalarında sütunlar genellikle noktalı virgül veya virgül ile ayrılır. Virgül, varsayılan ayrıcık olduğundan ayrıca belirtmenize gerek yoktur.

## 10.2. Excel Dosyalarından Veri Okuma

Herhalde gerek iş gerekse akademik çalışmalarında en fazla kullanılan dosya türü excel dosyalarıdır. Bu nedenle veri analizinde sık kullanılan programların hemen hepsinde excel dosyalarından veri okumak olanağıdır. Python'da excel dosyalarından veri okumanın en kolay yolu Pandas modülünü kullanmaktır.

## PYTHON İLE VERİ BİLİMLİ

Pandas modülünde excel dosyasından veri okumak için öncelikle dosya ile bağlantı kurmak gereklidir. Bunun için de .ExcelFile() metodu kullanılır.

```
import pandas as pd  
dosya = 'veri_dosyasi.xlsx'  
veri = pd.ExcelFile(dosya)
```

Bir defa dosya ile bağlantı kurulduktan sonra, dosyadaki sayfaları .sheet\_names() metodu ile görebiliriz. Dosya yolunu tanımlarken dizin de kullanacağınız dizinler arasında \ yerine / işaretini kullanmaya dikkat etmelisiniz. Bildiğiniz gibi Python'da \ işaret, \n, \t gibi kaçış (escape) karakterlerinde kullanıldığı için dosya yolunda bunun yerine / işaretinin kullanılması gerekiyor.

```
print(veri.sheet_names)  
['Sayfa1', 'Sayfa2', 'Sayfa3']
```

Istediğimiz sayfadaki verileri okumak için .parse() metodunu kullanabiliriz. Burada istedigimiz sayfanın ismini metin olarak ya da sayfa numarasını sayı olarak girmemiz mümkün. Bu açıdan, tablo1 = veri.parse('Sayfa1') veya tablo1 = veri.parse(0) aynı sonucu verecektir.

Excel dosyalarından veri okurken .parse() argümanına verebileceğimiz farklı argümanlar da bulunmaktadır. Bunlardan ilk skiprows argümanıdır. Bu argümanı kullanarak sayfada okunmadan atlanması istedigimiz satır numaralarını liste olarak iletебiliriz. Excel dosyalarında zaman zaman ilk satırlarda tablolar ile ilgili açıklayıcı bilgilere yer verilir. Bunlar bizim çalışmalarımızda işimize yaramayacağından, doğrudan atlanabilir.

```
tablo1 = veri.parse(0, skiprows=[0,1,2])
```

Yukarıdaki komut, excel sayfasındaki 1, 2 ve 3. satırları okumadan atlayacaktır. Aynı şekilde dosyadaki sütunların hepsini okumak istemezsiniz. Bu durumda parse\_cols argümanını kullanarak almak istedigimiz sütun numaralarını iletебiliriz.

```
tablo1 = veri.parse(0, skiprows=[0], parse_cols=[1,2,3,4])
```

Bir başka durumda, ilk satırındaki başlıklar okunmayı kendimiz farklı sütun başlıkları vermek isteyebiliriz. Buna neden ihtiyaç duyulur? Aynı formattaki çok sayıda Excel tablosundan veri çekip bunları birlestirecek bir kod yazdığını düşünün. Excel tabloları aynı bilgileri içermesine rağmen, sütun başlıklar, dosyayı kaydedene göre değişmiş olabilir. Örneğin, aynı başlık, bir dosyada "Sipariş No", bir diğerinde "Sip. No.", bir başkasında "Sipariş Numarası" şeklinde kaydedilmiş olabilir. Bu dosyaları olduğu gibi okumak sonrasında verilerin birleştirilmesinde soruna yol açacaktır. Bu durumda en doğrusu bütün dosyalarla ilk satırı atayıp sütunlara kendi seçeceğimiz isimleri vermektedir.

## VERİ OKUMA

```
tablo1 = veri.parse(0, skiprows=[1], parse_cols=[1,2,3,4],  
names = ['sepal_boy', 'sepal_en', 'petal_boy', 'petal_en'])
```

Tüm argümanlara iletебegimiz değerlerin liste formatında olmasına ve sayısal değerlerin de Python indeks yapısına uygun olarak 0 ile başladığını dikkat edin.

Excel dosyasından veri okumak için kullanabilecek bir başka Pandas metodudur. Bu metodun aldığı önemli argümanlar aşağıdaki gibidir<sup>17</sup>.

- sheet\_name: Verinin okunacağı sayfa adı.
- header: Sütun başlıklarının yer aldığı satır. Varsayılan olarak sıfırdır.
- skiprows: Baştan itibaren atlanacak satır sayısı.
- skip\_footer: Sondan itibaren atlanacak satır sayısı.
- index\_col: İndeks olarak kullanılacak sütün.
- names: Dosyada başlık satırı yoksa, başlık isimleri liste olarak iletебilir.

## 10.3. Diğer Dosya Türlerinden Veri Okuma

Sık kullanılan veri analizi programları arasında SAS, STATA, MATLAB ve SPSS gibi programlar da yer almaktadır. Bu programlar kullanılarak oluşturulmuş bir dosya ile karşılaşmanız halinde, bunları da Python'a nasıl aktaracağınızı bilmeniz faydalı olacaktır.

### 10.3.1. SAS Dosyaları

SAS, özellikle iş uygulamalarında sık kullanılan ticari bir yazılımdır. Statistical Analysis Software kelimelarının ilk harflerinden oluşmuştur. SAS yazılıminda kullanılan dosya uzantısı .sas7bdat veya .sas7bdat'dır.

Python'da SAS dosyalarını okumak için öncelikle Pandas modülünde yer alan .read\_sas() metodunu kullanabilirsiniz.

```
dosya = 'C:/Users/ilker/Desktop/veri_dosyasi.sas7bdat'  
veri_tablosu = pd.read_sas(dosya)
```

Python'da SAS dosyalarını okumanın bir başka yolu da sas7bdat kütüphanesi ve bu kütüphanede yer alan SAS7BDAT modülünü kullanmaktadır. Her ne kadar Pandas varken bu modülü önermemsem de yine de okuyucuların bilgiyi olması amacıyla paylaşmak istiyorum. SAS dosyasını okumak için önce kütüphaneden ilgili modülü aktartmak gereklidir.

17 Argümanların tam listesi için: [https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read\\_excel.html](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_excel.html)

### PYTHON İLE VERİ BİLİMİ

```
from sas7bdat import SAS7BDAT
veri = SAS7BDAT('veri_dosyası.sas7bdat')
veri_tablosu = veri.to_data_frame()
Bundan sonra veri_tablosu'nu kullanarak verileri Analiz edebilir ya da grafik çizdirebilir.
```

Sonraki adımda, okuduğumuz dosyayı veri çerçevesine çevirmemiz gereklidir. Bunun için `.to_data_frame()` metodunu kullanıyoruz.

Okumak istediğimiz SAS dosyasına bağlı kurmak için `SAS7BDAT()` fonksiyonunu kullanıyoruz.

```
import h5py
dosya_adi = 'hdf5_dosyası.hdf5'
veri = h5py.File(dosya_adi, 'r')
```

Fonksiyondaki ikinci argümanın ('r') dosyanın sadece okunması anlamına geldiğini artıktır. HDF5 dosyasında yer alan verileri görmek için sözlük veri yapısından tanımlı `.keys()` metodunu kullanabiliriz.

```
for alt_veri in veri.keys():
    print(alt_veri)
```

HDF5 verisinde yer alan alt verilerin kendileri de sözlük yapısında olabilir. Bunların alt verilerini de yine `.keys()` metodunu ile görebilir ya da `.value` metodunu ile verileri inceleyebiliriz.

HDF veri yapısı ve Python'da kullanımı konusunda ayrıntılı bilgi almak isteyen okuyucular Andrew Collette tarafından yazılmış olan *Python and HDF5* kitabına başvurabilir.

### 10.3.2. STATA

STATA, özellikle sosyal bilimler ve ekonomi alanlarında sıkça kullanılan bir istatistiksel analiz programıdır. Statistics and Data sözlerinin bir araya getirilmesinden oluşmuştur. STATA dosyaları .dta uzantısına sahiptir. Artık tahmin edebileceğiniz üzere STATA dosyasını okumak için Pandas modülündeki `.read_stata()` metodunu kullanabiliriz.

```
import pandas as pd
dosya = 'stata_dosyası.dta'
stata_tablosu = pd.read_stata(dosya)
```

### 10.3.3. HDF5 Veri Tipi

HDF5 ya da Hierarchical Data Format 5, çok büyük ölçekteki ve karmaşık yapıda verileri saklamak ve yönetmek üzere geliştirilmiş bir veri modelidir<sup>18</sup>. HDF veri modeli ilk olarak Amerikan, National Center for Supercomputing Applications<sup>19</sup> (NCA) kurumu tarafından geliştirilmiştir. HDF veri yapısı daha sonra kar amacı gütmeyen HDF kurumu tarafından geliştirilmeye devam etmiştir.

Python'un yanı sıra C++, Java, Matlab, Mathematica, R ve Julia gibi çok farklı programlama dilleri için de HDF dosya yapısı ile çalışmaya yönelik kütüphaneler geliştirilmiştir. HDF veri yapısı ileabyte boyutunda dosyalarla bile rahatlıkla çalışılabilir (1 exabyte =  $10^9$  gigabyte).

Python'da HDF5 dosyaları ile çalışabilme için geliştirilmiş olan modül h5py modülüdür<sup>20</sup>. Bir hdf5 dosyasından veri okumak için bu modüldeki `.File()` metodunu kullanılır.

<sup>18</sup> <https://support.hdfgroup.org/HDF5/whatis hdf5.html>

<sup>19</sup> <http://www.ncsa.illinois.edu>

<sup>20</sup> <http://www.h5py.org>

180

### 10.3.4. MATLAB

Matlab özellikle mühendislik ve temel bilimlerde yoğun olarak kullanılan bir programlama dilidir. Matrix Laboratory kelimelarının birleşmesinden oluşmuştur. Matlab'in ticari bir yazılım olması yaygınlığını sınırlasa da ücretsiz bir yazılım olan Octave Matlab'la neredeyse tamamen aynıdır. Vektör ve matrislerle çalışma prensibine sahip olması nedeniyle Matlab kolay ve hızlı bir programlama dilidir. Özellikle lineer cebir uygulamalarında başvurulan bir programdır. Yapay öğrenme, derin öğrenme gibi konularda yakın zamana kadar en fazla kullanılan programlardan birisi Matlab olmakla birlikte son zamanlarda Python daha fazla kullanılmaya başlanmıştır.

Matlab dosyaları .mat uzantısına sahiptir. Matlab dosyalarını okumak için `scipy.io` modülünde yer alan `scipy.io.loadmat()`, dosyaları Matlab formatında kaydetmek içinse `scipy.io.savemat()` fonksiyonlarını kullanabiliriz. Bir .mat dosyası farklı tipte veri nesnelerinden meydana gelmiş olabilir. Python'da `.loadmat()` metodunu ile .mat dosyasını okuduğunuzda veri bir sözlük veri yapısı şeklinde okunur. Sözlükte anahtarlar, .mat dosyasındaki veri nesnelerinin isimleri, değerler ise veri nesnelerini belirtir. Aşağıdaki örnekte, Matlab'da hazır olan örnek dosyalardan gprdata.mat dosyasının Python'dan okunması gösteriliyor. Başında ve sonunda '\_\_' olmayan değişkenler okuduğumuz veri dosyasında yer alan veri nesnelerini ifade ediyor.

```
import scipy.io
dosya = 'C:/Users/ilker/Desktop/gprdata.mat'
mat_veri = scipy.io.loadmat(dosya)
```

## PYTHON ile VERİ BİLİMİ

```

print(mat_veri.keys())
print(mat_veri['__header__', '__version__', '__globals__', 'Xtest', 'Xtrain',
      'ytest', 'ytrain'])

print(type(mat_veri['Xtrain']))
<class 'numpy.ndarray'>

import numpy
print(numpy.shape(mat_veri['Xtrain']))
(500, 6)

```

## 10.4. İlişkisel Veri Tabanlarından Veri Okuma

İlişkisel veri tabanlarını basitçe, farklı tablolardan oluşan, her tablodaki kayıtların (satır) belirli bir anahtar değişken (sütun) ile belirlendiği ve tablolar arasındaki ilişkiler için de bu anahtar değişkenlerin kullanıldığı veri tabanlarıdır. İlişkisel veri tabanı ilk olarak 1970 yılında E.F. Codd tarafından önerilmiştir<sup>21</sup>. Bütün ilişkisel veri tabanları veri arama ve düzenleme için SQL (Structured Query Language) dilini kullanır. Kullanılmakta olan birkaç farklı SQL türü vardır: MySQL, PostgreSQL, SQLite vb.

İlişkisel veri tabanının çok sayıda tablodan oluştuğunu söylemişik. Tabloları Pandas veri çerçevesi gibi düşünebilirsiniz. Bir tablodaki her bir satır bir veri, gözlem veya kaydı temsil eder. Her sütun ise bu tekil gözlemlere ilişkin bir özelliği belirtir.

Örnek olmasından bir kütüphanenin veri tabanını oluşturma görevini aldığınızı varsayılmı. Hangi verileri sakladınız? Öncelikle kütüphanedeki bütün kitaplar için bir tablo oluşturmak gereklidir. Kitap adı, yayınevî, yazarı, kitabına hangi alanla ilgili olduğu vb bilgileri bu tabloda saklayabilirsiniz. İkinci olarak kütüphane üyelerinin yer aldığı bir tablo oluşturmak gerekecektir. Bu tabloda da üye numarası, adı, soyadı, telefonu, eposta ve adres bilgileri yer alabilir. Üçüncü olarak ödünç verilen kitapları takip edebilmek için bir tablo oluşturmak gerekecektir. Hangi üyenin, hangi kitabı ne zaman aldı, ne zaman teslim ettiği vb. bilgiler bu tabloda takip edilebilir. Bu tablolar artırılabilir. Örneğin, kitap siperişlerinde kolaylık sağlamak için yayınevlerine ilişkin bilgiler başka bir tabloda yer alabilir. Bütün bu tabloları da anahtar değişkenlerle birbirine bağlamak gereklidir. Örneğin, kitap tablosu ile ödünç kitap tablosunu kitap no veya kitap adı gibi bir alanla ilişkilendirmek gereklidir ki bir kitap hangi tarihte ödünç verilmiş, o sırada kütüphanede mi ödünç verilmiş anlaşılır. Yine üye tablosu ile ödünç kitap tablosu da üye no gibi bir alanla

<sup>21</sup> [https://en.wikipedia.org/wiki/Relational\\_database](https://en.wikipedia.org/wiki/Relational_database)

## VERİ OKUMA

birleştirilebilir. Böylece hangi üyenin hangi kitapları aldığı, o sırada üzerinde kitap olup olmadığı varsa kaç tane kitap almış olduğu, aldığı kitapları gecikmeli olarak tade edip etmediği takip edilebilir. Benzer şekilde kitap bilgileri tablosu ile yayınevî tablosu da kitap no gibi bir alanla birleştirilebilir. Bu örnekleri artırmak da mümkündür. Gördüğünüz gibi basit bir kütüphane uygulaması için bile dört tane tablodan oluşan bir ilişkisel veri tabanı kurduk. Çok daha karmaşık verilerin kaydedilip kullanılan fabrika, banka, üniversite gibi kuruluşlarda çok daha fazla sayıda tabloya ve değişkene ihtiyaç vardır.

Sündür tarif ettigimiz kütüphane veri tabanına ilişkin tabloları görelim. Kitapta gereksiz yer kaplamaması için tablolardaki kayıt sayısını az tuttum.

Kitap Listesi						
Kitap No	Kitap Adı	Yayinevi	Yazar	Yayın Yılı	ISBN	Konusu
1	R ile İstatistiksel Programlama	Pusula	İlker Arslan	2015	9786056460890	Bilgisayar
2	Ortadoğu	Arkadaş	Bernard Lewis	2015	9789755094427	Tarih
3	Felsefeye Giriş - 1	İş Bankası	H. Ziya Ülken	2008	2789785876106	Felsefe
4	Zamanın Kısa Tarihi	Alfa	S. Hawking	1988	9786051067582	Bilim
5	Fizik Yasaları Üzerine	Alfa	R. Feynman	2015	9786051066509	Bilim

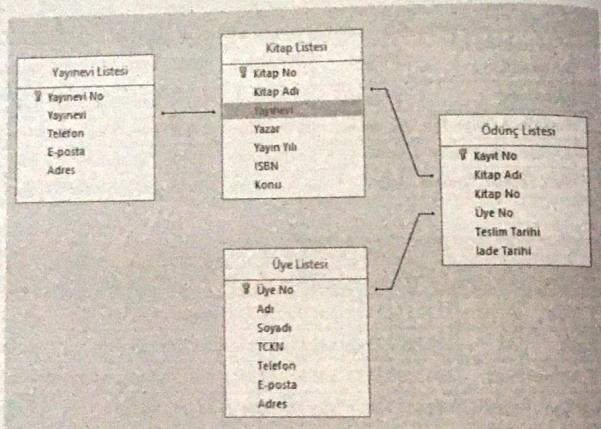
Üye Listesi						
Üye No	Adı	Soyadı	TCKN	Telefon	E-posta	Adres
1	İlker	Arslan	11111111111	5300000000	ilkarslan@gmail.com	İstanbul
2	Ali	Ak	22222222222	5321111111	ali.ak@email.com	İstanbul
3	Ahmet	Sarı	33333333333	5453333333	ahmet.sari@eposta.com	İstanbul
4	Ayşe	Kitapçı	44444444444	5554444444	ayse@kitap.com	İstanbul

Ödünç Listesi				
Kayıt No	Kitap Adı	Kitap No	Üye No	Teslim Tarihi / İade Tarihi
1	R ile İstatistiksel Programlama	12	01.01.2015	25.03.2015
2	Zamanın Kısa Tarihi	43	05.01.2015	15.04.2015
3	Felsefeye Giriş - 1	31	03.03.2015	05.05.2015

## PYTHON ile VERİ BİLİMİ

Yayinevi Listesi				
Yayinevi No	Yayinevi	Telefon	E-posta	Adres
2	Alfa	0212 0000000	alfa@kitap.com	İstanbul
3	İş Bankası Kültür	0212 1111111	isbank@kitap.com	İstanbul
4	Pusula	0212 2222222	pusula@kitap.com	İstanbul
5	Arkadaş	0212 3333333	arkadas@kitap.com	İstanbul

Bu tabloları birbirleriyle ilişkilendirmek için de ilgili alan isimleri kullanabilir. Yukarıda, ki tablolarnı birbirleriyle ilişkilerini aşağıdaki şekilde daha rahat görebiliriz.



Şekil 10-2: İlişkisel veri yapısı türü

Python kullanarak bir ilişkisel veri tabanına bağlanmak ve bu veri tabanında SQL yardımı ile sorgu yapmak ve veri çekmek mümkündür. Bunun için tabii ki öncelikle veri tabanı ile bağlantı kurmanız gereklidir. Bunun için kullanılabilir çok sayıda modül vardır: sqlalchemy, sqlite3, pypyodbc gibi... Örnek olarak sqlalchemy modülünü inceleyelim. Öncelikle sqlalchemy modülünden `create_engine` fonksiyonunu aktarıyoruz ve istediğimiz SQL sunucusuna bağlanmak için bu fonksiyonu kullanıyoruz. Çoğunuzun bildiği gibi ilişkisel veri tabanları kurumsal olarak kullanılan veri tabanlarıdır. Bu nedenle, buradaki örnekler sql sunucusuna erişiminiz olan bir ortamda yani iş yerinizde deneyebilirsiniz. İşinizde ihtiyaç duyduğunuz veriler sql sunucusunda ise ancak siz çalışmalarınızı Python'da yürütmek istiyorsanız burada verilen örnekleri uyarlayarak kullanabilirsiniz. Çoğu zaman SQL

## VERİ OKUMA

sunucusuna bağlantı yetkisi ve kullanmanız gereken sunucu bağlantısı için kurumunuzun bilgi teknolojileri bölümü ile temas kurmanız gereklidir.

```
from sqlalchemy import create_engine
```

SQL sunucusu ile bağlantı kurmak için kullandığınız `create_engine()` fonksiyonunda argümanları aşağıdaki format ve sıradır kullanmalısınız. Yukarıda da belirttiğim üzere kullanıcı adı, şifre, host, veri tabanı gibi bilgileri (tabii ki erişim yetkiniz varsa) çalışığınız kurumun bilgi teknolojileri bölümünden almanız gerekiyor.

```
baglan = create_engine('sql_türü://kullanıcı:sifre@host:port/veri_tabanı')
```

SQL türü, sqlite, mysql, postgresql, oracle ya da mssql olabilir. Bağlantı kurulduğundan sonra, bağlandığımız veri tabanında hangi tabolların yer aldığı görmek için `.table_names()` metodunu kullanabilirsiniz.

```
tablolar = baglan.table_names()
print(tablolar)
```

Şimdi oluşturduğumuz bağlantıyı `.connect()` fonksiyonu ile kullanabiliriz.

```
baglanti = baglan.connect()
```

Bu bağlantıyla kullanarak SQL sorgusu yapmak için `.execute()` metodunu kullanmak gereklidir.

```
sorgul = baglanti.execute('SELECT * FROM Tablo_1')
```

Bu kodun çalışması sonucunda oluşturulan sorgul bir tablo ya da veri çerçevesi değil sqlalchemy modülünde tanımlanmış olan bir sorgu nesnesidir. Şimdi bu sorguyu bir veri çerçevesine çevirmek gerekiyor. Bunun için de Pandas modülünden faydalanaçagız. Sorgu nesnesini veri çerçevesine çevirmek için `.fetchall()` metodu kullanılır.

```
sonuc_tablo = pd.DataFrame(sorgul.fetchall())
```

Bütün sorgu sonucu yerine belirli sayıda satırı çekmek isteyebiliriz. Bunun farklı sebepleri olabilir. Örneğin, sorgu çok büyük olduğu için tamamını aynı anda çekmek mümkün olmayabilir. Bu durumda, `.fetchall()` yerine `.fetchmany()` metodunu size argümanı ile birlikte kullanırız. Size argümanı kaç satır okumak istediğimizi belirtir.

```
sonuc_tablo = pd.DataFrame(sorgul.fetchmany(size=100))
```

Bu şekilde elde ettiğimiz veri çerçevesinin bir eksigi var: sütun isimleri eksik. Önceki kodda oluşturulan sorgu nesnesi sözlük yapısında olduğu için sütun isimleri anahtar kelime-

## PYTHON ile VERİ BİLİMI

ler olarak saklanıyor. Veri çerçevesine sütun isimlerini de eklemek için sözlük anahtar kelimelerini veri çerçevesinin sütun isimlerine atamak gereklidir.

```
sonuc_table.columns = sorgui.keys()
```

Bağlandığınız tablolarda sorgulama yapmak için SQL kodları kullanmanız gerekmektedir. SQL için çok sayıda kaynak ve kitap bulunmaktadır. Başlangıç seviyedeki bilgiler için dileğinden okurlar ekler bölümünde yer alan SQL Sorguları bölümünde göz atılabilir. Burada giriş seviyesindeki SQL bilgileri verilmiştir.

Tablolarda yaptığımız sorguları veri çerçevesi olarak kaydettikten sonra ilgili bağlantıyı kapatabiliyoruz. Bağlantıyı kapatmak için `.close()` metodunu kullanmak gereklidir: `sorgui.close()`.

Bir defa `.create_engine()` komutu ile SQL sunucusuna bağlandıktan sonra `.connect()` metodunu ile bağlantıyı canlı hale getirmemiz, `.execute()` ile SQL sorgusunu gerçekleştirmemiz, `.fetchall` (veya `.fetchmany`) komutu ile sorgu nesnesini ver çerçevesine çevirmemiz, veri çerçevesinin sütun isimlerini atamamız ve son olarak bağlantıyı kapatmamız gereklidir. Öyleyse pandas modülünden bunların hepsi tek satırda yapmak mümkün olacaktır. Bunun için pandas modülünden `.read_sql_query()` metodunu kullanmamız gereklidir.

```
baglan = create_engine('sql_türü://kullanıcı_adi:sifre@host:port/veri_tabani')
sonuc_table = pd.read_sql_query('SELECT * FROM Tablo1', baglan)
```

Göründüğü gibi bu metod sayesinde sqlalchemy modülünde 3-4 satırda yapılan işi tek satırda yapmak mümkün oluyor. Ancak hala `.create_engine()` metodu için sqlalchemy modülünde ihtiyaç duyduğumuzu dikkat edin.

## 10.5. Internetten Veri Okuma

Şimdide kadar gördüğümüz örneklerde bilgisayaramızda (ya da yerel bir bellekte) yer alan veri dosyalarını okuduk. Ancak zaman zaman bir internet sayfasında yer alan verileri indirip kullanmamız gerekebilir. Internetten veri okurken çoğu zaman ilgili siteye girip veri dosyasının linkini kullanarak dosyayı bilgisayaramıza indiririz. Daha sonra da bilgisayaramızdaki bu dosyanın türne göre önceki bölmüllerde anlattığımız metodları kullanarak dosyadaki verileri okuruz. Ancak, bu her zaman etkin bir yöntem olmayabilir. Örneğin, bazı finansal verileri günlük olarak indirip analiz ettiğimizi düşünün. Her gün, ilgili internet sitesine gidip dosyaları manuel olarak indirmek sonra Python kodunu çalıştırıp verileri okumak gereksiz zaman kaybına yol açacaktır. Bunun yerine verileri doğrudan Python'dan okuyabiliriz. Bu sayede, sizin yaptığınız çalışmaya daha sonra bir başkasının kontrol etmesi ve aynı çalışmayı tekrarlaması da olasıdır.

### VERİ OKUMA

Akademik veya profesyonel iş hayatında, yaptığımız çalışmaların tekrar edilebilir olması son derece önemlidir. Bu konuya ilkiemizde henüz yeterince önem verilmeyen maaşef. Ancak, çalışmalarınızı, bir başkasının, sizin yazdığınız kodlarla, çalışmanızı, siz olmasanız bile baştan sona tekrar edebilecek ve aynı ya da benzer sonuçlara ulaşacak şekilde tasarlamamanız öneririm.

İnternette veri biliimi, finans, yapay öğrenme gibi farklı konularda kullanabileceğiniz çok fazla veri setleri mevcuttur. Örneğin, <https://archive.ics.uci.edu/ml/index.php><sup>22</sup> adresinde yapay öğrenme çalışmalarında kullanabileceğiniz çok sayıda veri seti bulunmaktadır. Ya da quandl.com veya finance.yahoo.com gibi sayfalarдан finansal verileri temin etmek mümkündür.

### 10.5.1. Internetten Veri Dosyası Okuma

Python'da internette veri okumak için kullanılabilen farklı modüller ve fonksiyonlar vardır. Önce, `urllib.request` modülüne ve bu modüldeki `urlretrieve` fonksiyonlarına göz atalım. Bu fonksiyon bir internet linkindeki dosyayı bilgisayaramıza aktarmak için kullanılır. Örneğin yukarıda verdigimiz çok sayıda yapay öğrenme veri setinin yer aldığı UCI sitesindeki iris veri setini okuyalım. UCI sayfasından, yapay öğrenme konusunda çok ünlü olan ve kitabin farklı bölmüllerinde kullandığımız iris veri setini çekelim.

```
from urllib.request import urlretrieve
import pandas as pd

link = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/
iris.data'

urlretrieve(link, 'iris.csv') # linkte yer alan dosyayı iris.csv olarak kaydet
iris = pd.read_csv('iris.csv', sep=',')
# sep argümanı, csv dosyasında sütunların hangi işaretle ayrıldıklarını belirtir.

print(iris.head())
    5.1  3.5  1.4  0.2  Iris-setosa
0  4.9  3.0  1.4  0.2  Iris-setosa
1  4.7  3.2  1.3  0.2  Iris-setosa
2  4.6  3.1  1.5  0.2  Iris-setosa
3  5.0  3.6  1.4  0.2  Iris-setosa
4  5.4  3.9  1.7  0.4  Iris-setosa
```

Yukarıdaki yöntemde, ilgili dosyayı önce bilgisayaramıza indirmemiz gereklidir. Pandas modülünü kullanarak veri dosyasını indirmeden de veri okuyabiliriz.

```
import requests
from bs4 import BeautifulSoup
link = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/
iris.data'
iris = pd.read_csv(link, sep=',')
```

<sup>22</sup> University of California Irvine Machine Learning Repository

```
print(iris.head())
5.1  3.5  1.4  0.2  Iris-setosa
0   4.9  3.0  1.4  0.2  Iris-setosa
1   4.7  3.2  1.3  0.2  Iris-setosa
2   4.6  3.1  1.5  0.2  Iris-setosa
3   5.0  3.6  1.4  0.2  Iris-setosa
4   5.4  3.9  1.7  0.4  Iris-setosa
```

### 10.5.2. HTML Verilerini Okuma

Gördüğünüz gibi ikinci yöntemde iki yerine bir satırda istediğimiz işlemi gerçekleştiriyor. Aynı şekilde, Pandas `.read_excel()` fonksiyonunu kullanarak internette yer alan bir dosyayı, sanka bilgisayarımızdaymış gibi linkini belirterek okuyabiliriz. Bunun için dosya linkini (`url`) yazmanız yeterlidir.

Şimdiye kadar hep url'den bahsettim. Hepimiz url'nin internet adresi için kullandığını biliyoruz. Peki ne anlama geliyor url? URL kelimesi **Uniform Resource Locator**<sup>23</sup> kelimenin ilk harflerinden oluşuyor. Internette yer alan her bir dosyanın, fotoğrafın, sayfanın yer standart formattaki url adresi sayesinde belirlenir. URL adresi dendiginde akımla en çok internet sayfaları gelir. Ancak, FTP (File Transfer Protocol, Dosya Transferi Protokolü) veya Veri Tabanı Erişimleri için de url adresleri kullanılabilir. Bir web sayfasının düşünelim: <http://www.ilker.arslan.com/index.html> (Böyle bir adres yok, ben uydurdum). Bu url adresi üç ana parçadan oluşur: protocol ([http<sup>24</sup>](http://)), hostname ya da host adı<sup>25</sup> ([www.ilker.arslan.com](http://www.ilker.arslan.com)), ve son olarak sayıya da dosya adı (`index.html`). HTTP, Hypertext Transfer Protocol kelimelerinden türetilmiştir<sup>26</sup>. HTTP, "bir kaynaktan dağıtılan ve ortak kullanıma açık olan bilgi sistemleri protokolüdür".

Internette her türlü veri paylaşımı http yapısı ile gerçekleştiriliyor. Örneğin, Google Chrome, Safari, Internet Explorer gibi bir program kullanarak herhangi bir internet sayfasına girdiğimizde, bu sayfadan veri çekmek istiyoruz. Diğer bir deyişle bu siteye bir veri talebi veya http talebinde bulunuyoruz. Python kullanarak bir siteden veri indirdiğimizde de aynı şekilde http talebinde bulunuyoruz. Bu talebi, yukarıda da gördüğümüz `urlretrieve()` fonksiyonu ile gerçekleştirebiliriz.

23 <https://en.wikipedia.org/wiki/URL>

24 Protokol, http veya https olabilir. Https, http'ye göre daha güvenli bir protokol şeklidir.

25 Host kelimesine karşılık gelen bilgisayar adı, sistem adı, anamakine adı gibi kullanımlar var ama tam anlamıyla içime sinen bir Türkçe karşılık bulamadım. Okuyucularımın hoşgörüsününe saygınlıyorum. Uygun bir öneri olursa sonraki baskılar için mutlaka dikkate alırım

26 [https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

Peki bir sayfadan, veri talebinde bulunduğuuzda çektiğimiz veri tam olarak ne verisidir? Yukarıda gördüğümüz gibi bir csv veya excel dosyası olabilir. Ancak birçok internet sayfası HTML adı verilen bir dille yazılmış olan dokümanlardır. HTML<sup>27</sup> (Hypertext Markup Language) web sayfaları ve uygulamaları oluşturmak için kullanılan standart metin işaretlemesi dildir. CSS (Cascading Style Sheets) ve JavaScript dilleri ile birlikte web sayfaları oluşturmak için kullanılır. HTML parçaları, bir internet sayfasının yapıtaşlarını (başlıklar, paragraflar, cümleler, bölümler vb.) oluşturur. İşte, bir sayfadan veri talebinde bulunduğuuzda bu html parçalarına ilişkin verileri çekmiş oluruz.

Burada, HTML verilerinin yapısından bahsetmemiz gerekiyor. Veri türlerini sınıflandırmamın bir yolu yapılandırılmış veri ve yapılandırılmış veri şeklinde ayırmaktır. Yapılandırılmış veriler, tanımlanmış bir modelde sahip olan ya da belirli bir biçimde organizelidilmiş olan verilerdir. Örneğin, SQL verileri ya da excel dosyaları yapılandırılmış türde verilerdir. Belirli bir veri modeli, biçimini vb. olmayan veri türleri ise yapılandırılmış verilerdir. Örneğin, Twitter veya Facebook gibi sosyal medya sayfalarında kullanıcıların yazdıkları yapılandırılmış verilerdir.

HTML verileri hem yapılandırılmış hem de yapılandırılmış veriler olarak düşünülebilir. Bir html dosyası çeşitli metinlerde oluşur. Ancak, bu metinler çeşitli etiketlerle (tag) birbirinden ayrılır. Örneğin, bir yazının başlık olduğunu belirtmek için `<head>Başlık</head>` şeklinde iki başlık etiketi içinde yazmak gereklidir. Benzer şekilde, paragraph için `<p></p>`, başka bir web sayfasına link vermek için `<a></a>`, fotoğraf için `<img>` gibi çeşitli etiketler kullanılır. Çok sayıda html etiketine ilişkin bilgileri <https://www.w3schools.com/tags/> sayfasından görebilirsiniz.

Şimdi, bir internet sayfasından html verisi çekmek için gerekli kodları adım adım görelim. Önce, `urllib` modülünden, gerekli fonksiyonları aktarıyoruz ve veri çekmek istediğimiz sayfa tanımlıyoruz.

```
from urllib.request import urlopen, Request
sayfa = 'www.datakod.net'
```

Sonraki adımda web sayfasına `Request()` ile bir veri talebi gönderiyoruz ve gelen yanıt `urlopen()` fonksiyonu ile kaydediyoruz. Bu kaydettiğimiz `urllib` modülünde tanımlanmış olan bir cevap (`response`) nesnesi. Bu nesneyi okumak için de `.read()` metodunu kullanıyoruz. Böylece veri çektiğimiz sayfanın html kodunu, metin olarak çekmiş oluyoruz.

```
veri_talebi = Request(sayfa)
gelen_veri = urlopen(veri_talebi)

veri = gelen_veri.read()
```

27 <https://en.wikipedia.org/wiki/HTML>

## PYTHON İLE VERİ BİLİMİ

Son olarak gerçekleştirdiğimiz bağlantıyı kapatmamız gerekiyor.

```
gelen_veri.close()
```

Veri değişkenini yazdırduğumuzda aşağıdaki şekilde sayfanın html kodunu görebiliriz. Fazla yer kaplamaması için buraya çıktıının sadece başlangıç kısmı alınmıştır.

```
b'<!DOCTYPE html><html itemscope itemtype="http://schema.org/WebPage" lang="tr-TR"><head><meta charset="UTF-8"><meta name="viewport" content="width=device-width, initial-scale=1"><link rel="profile" href="http://gmpg.org/xfn/11"/><link rel="pingback" href="http://www.datakod.net/xmlrpc.php">
```

Yukarıda yaptığımız işin aynısını daha basit bir biçimde, çok yaygın olarak kullanılan Requests<sup>28</sup> paketi ile de gerçekleştirebiliriz. Requests paketini kullanarak önce `.get()` metodunu ile ilgili sayfaya bağlanıp verileri çekiyoruz. Sonrasında, `text` metodunu kullanarak çekilen veriyi metne çeviriyoruz.

```
import requests

sayfa = 'http://www.datakod.net'

veri = requests.get(sayfa)

sonuc = veri.text

print(sonuc)

<!DOCTYPE html>
<html itemscope itemtype="http://schema.org/WebPage" lang="tr-TR">
<head>
```

Yine sonucu yazdırduğumuzda bu defa html kodunun daha düzgün bir şekilde yazdırıldı-

28 <http://docs.python-requests.org/en/master/>

## VERİ OKUMA

gini görüyoruz. Buraya kadar bir web sayfasından html verilerini çekmeyi gördük. Ancak yukarıda da gördüğümüz gibi şimdije kadar yaptıklarımızla verileri sadece düz metin olarak, html etiketleri ile birlikte çekemeliyiz. Bu format da veri analizi için çok uygun değil. Python kullanarak html verilerini yapılandırılmış bir şekilde çekemebilmek için BeautifulSoup isimli bir modül geliştirilmiştir<sup>29</sup>.

```
from bs4 import BeautifulSoup
import requests

sayfa = 'https://finance.yahoo.com'
veri = requests.get(sayfa)

sonuc = veri.text

sonuc_duzgun = BeautifulSoup(sonuc, 'html.parser')

print(sonuc_duzgun.prettify())
```

Burada yaptıklarımızın öncekinden farkı, elde ettigimiz metni, BeautifulSoup() fonksiyonunu kullanarak bir BeautifulSoup nesnesine çeviriyoruz. Sonrasında kullandığımız `prettify()` metodunu, bu nesneyi düzgün, sorgulanabilir bir formata getiriyor. Bu, sadece çektiğimiz html verisini düzgün bir şekilde yazdırma yaramıyor. Aynı zamanda, çektiğimiz veriler arasında çeşitli sorgular yapmamıza da olanak sağlıyor.

BeautifulSoup modülünde tanımlanmış, html verilerini sorgulamaya yarayan çok sayıda metod vardır. Aşağıda bu metodlara örnekler verilmiştir. Tüm örnekleri incelemek isterseniz modülün <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> sayfasında yer alan dokümanlara göz atabilirsiniz.

```
print(sonuc_duzgun.title)
<title>Yahoo Finance - Business Finance, Stock Market, Quotes, News</title>

print(sonuc_duzgun.get_text())
window.performance && window.performance.mark && window.performance.
mark('PageStart');Yahoo Finance - Business Finance, Stock Market,
Quotes, News(function(html){var c = html.className;c += " JsEnabled";c =
c.replace("NoJs","");
/* CDATA[ */
var ajaxurl = "http://www.datakod.net/wp-admin/admin-ajax.php";
/* ]]> */

```

```
</script>
print(sonuc_duzgun.p)
<p class="CSc-fuji-grey-a Tov(e) firefox_D(b)! Tsh(streamHeroTextShadow)
Fw(400) Pb(4px) Fz(14px) LineClamp(2,34px) Pt(6px)" data-
```

29 <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

PYTHON ile VERİ BİLİMİ

```
reactid="15">><span class="Mend5px" data-reactid="20">The morning's top
headlines</span>

print(soumc_duzgun.find_all('p'))
(<p class="O{fc-fuji-grey-8; Tov(e) firefox_D(b); Tsh(streamHeroTextShadow,
fw:20px; Pb:4px; Fz:14px; LineClamp(2,34px); Pt:6px)" data-
reactid="19">><span class="Mend5px" data-reactid="20">The morning's top
headlines</span>

linkler = soumc_duzgun.find_all('a')

for link in linkler:
    print(link.get('href'))
https://finance.yahoo.com/
https://mail.yahoo.com/?intl=us&lang=en-US&.partner=none&.src=finance
/quote/ES=F?p=ES=F
/quote/YM=F?p=YM=F
/quote/NQ=F?p=NQ=F
/quote/RUT?p=RUT
/quote/CLF1pCCLP
```

### 10.5.3. API Verileri Okuma<sup>30</sup>

API, Application Programming Interface kelimelerinden türetilmiş olup "Uygulama Programlama Arayüzü" anlamına gelir<sup>31</sup>. Bir uygulamaya ait özelliklerin başka bir uygulamada kullanılabilmesini sağlar. API basitçe, yazılım uygulamaları ile bağlantı kurmayara yanarak protokol ve program parçalarıdır. Bir diğer deyişle API'ları, iki farklı program arasında iletişim kurmayara yanarak kodlar olarak da düşünebiliriz. Nasıl ki bir programın kullanıcı arayüzü insanların o program kullanmasını kolaylaştıryorsa, API'lar da program geliştiricilerin kendi programlarında çeşitli teknolojileri kullanmalara olanak verir. Örneğin Twitter API'yi kullanarak yazdığınız bir programda tweet yazacak, tweetlere yorum yapacak kodlar geliştirebilirsiniz. Konumuzun dışında olduğu için API konusunda daha ileriye gitmeyeceğiz. Konunun bizi ilgilendiren tarafı, API'ları kullanarak farklı uygulamardan veri okumanın mümkün olmasıdır. Örneğin Twitter API kullanarak Python'da twitter verilerini çekip analiz edebiliriz. IMDB tarafından geliştirilen OMDB API kullanarak Python'da IMDB film veri tabanındaki verileri okuyabiliriz. Ya da Wikipedia'dan veri çekip analiz etmek istiyorsak Wikipedia API'sını kullanabiliriz. Nitekim, Twitter, Instagram veya Facebook gibi sosyal sitelerin API'ları pazarlama firmaları ve araştırmacılar tarafından yaygın olarak kullanılmaktadır.

30 Bu bölümde aktarılanlar bazı okuyucular için ileri seviye olabilir. İsteyen okuyucular bu bölüm atlayabilirler.

<sup>31</sup> [https://en.wikipedia.org/wiki/Application\\_programming\\_interface](https://en.wikipedia.org/wiki/Application_programming_interface)

192

## VERİ OKUMA

JSON veri formatı, Python'daki sözlük veri yapısına çok benzer.

```
{'ogrenci_no': 54321,  
 'ogrenci_adi': 'Eren',  
 'ogrenci_soyad': 'Arslan',  
 'ogrenci_sinif': 6}
```

Şimdi bir internet adresinde yer alan json formatındaki dosyayı okuyalım. JSON dosyanın okumak için çok farklı yöntemler vardır. Pandas modülünde yer alan `read_json()` metodu en kullanışlılarından birisi.

```
import pandas as pd

link = 'https://www.quandl.com/api/v3/datasets/WIKI/FB/data.json'
veri = pd.read_json(link)

print(veri)

collapse dataset_data
column_index None
column_names None
data [[2018-01-08, 187.2, 188.9, 186.33, 188.28, 14...]
end_date 2018-01-08
frequency daily
limit None
order None
start_date 2012-05-18
transform None
```

<sup>32</sup> <https://en.wikipedia.org/wiki/JSON>

## PYTHON İLE VERİ BİLİMİ

```
Requests modülündü kullanarak da aynı işlemi gerçekleştirebiliriz.

import requests

link = 'https://www.quandl.com/api/v3/datasets/WIKI/FB/data.json'

dosya = requests.get(link)
dosya_json = dosya.json()

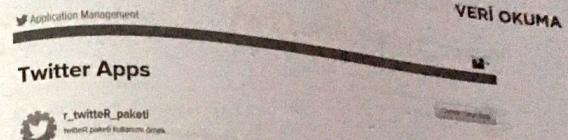
print(dosya_json)

{'dataset_data': {'limit': None, 'transform': None, 'column_index': None,
'column_names': ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Ex-
Dividend', 'Split Ratio', 'Adj. Open', 'Adj. High', 'Adj. Low', 'Adj.
Close', 'Adj. Volume'], 'start_date': '2012-05-18',
'end_date': '2018-01-08',
'frequency': 'daily',
'data': [[['2018-01-08', 187.2, 188.9, 186.33, 188.28, 14719216.0,
0.0, 1.0, 187.2, 188.9, 186.33, 188.28, 14719216.0], ['2018-01-05',
185.59, 186.9, 184.93, 186.85, 13042388.0, 0.0, 1.0, 185.59, 186.9,
184.93, 186.85, 13042388.0]]...]
```

API kullanımına örnek olarak cogumuzun ilgisini çeken bir konuya Twitter API kullanarak veri çekmeyi ve analiz etmeyi görelim. Twitter verisi araştırmacılar ve pazarlama bölgümleri tarafından ciddi şekilde incelenen ve kullanılan bir veri. Örneğin tüketicilerin yeni çıkan ürün veya seyircilerin bir sinema filmi hakkındaki düşünceleri ya da vatandaşların politikalar hakkındaki görüşleri hakkında öngörüde bulunmak için twitter verisi son derece zengin imkanlar sunuyor. Artık, çeşitli ekonomik kurumlar örneğin İngiltere Merkez Bankası halkın ekonomik gidişat hakkındaki düşüncelerini ölçmek için sosyal medya analizini kullanıyor. Ancak, saniyede binlerce tweetin yazıldığı<sup>33</sup> bir dönemde bu kadar çok yapılandırılmış verinin analizi çok da kolay bir iş değil. Burada da devreye çeşitli yapay öğrenme algoritmaları giriyor. Simdilik analiz kismini bir tarafa bırakıp veri okuma tarafıyla ilgilenebilir.

Python kullanarak Twitter verilerini okuyabilmek için öncelikle bir Twitter hesabınızın olması gereklidir. Daha sonra Twitter kullanıcı adı ve şifrenizle <https://apps.twitter.com> adresine giriş yapın. Kullanıcı girişi sonrasında aşağıdaki gibi bir ekranla karşılaşacaksınız.

33. <http://www.internetlivestats.com/twitter-statistics/>



Şekil 10-3: Twitter API kullanıcı girişi

Ekranda gördüğünüz Create New App yazılı butona tıklayın. Şimdi de aşağıdaki ekran belirecek. Gördüğünüz boşlukları doldurduktan sonra Create Your Twitter Application butonuna tıklayın.

### Create an application

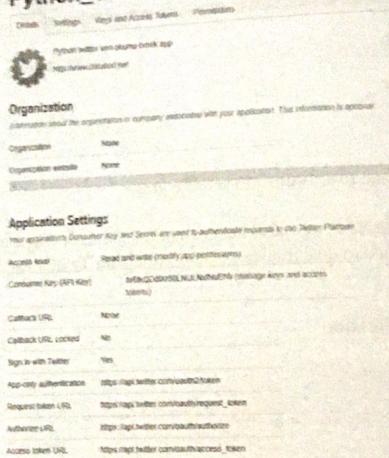
The form contains fields for Application Details, Website, Callback URL, and Developer Agreement. The 'Developer Agreement' checkbox is checked.

Şekil 10-4: Twitter APP'de yeni bir uygulama oluşturma

Sonrasında aşağıdaki ekranla karşılaşacaksınız. Ekranda yukarıda gördüğünüz "Keys and Access Tokens" sekmesine tıklayın. Bu sekmede gördüğünüz Consumer Key (API Key), Consumer Secret (API Secret), Access Token ve Access Token Secret değerlerini kaydedin. Access Token ve Access Token değerlerini göremiyorsanız aşağıda göreceğiniz "Create Access Token" butonuna tıklayın.

## PYTHON İLE VERİ BİLİMİ

### Python\_Twitter\_App\_Ornek



Şekil 10-5: Python Twitter APP

Şimdi, Python'da Twitter verisi okumak için Tweepy paketini kullanacağız. Önce access token, access token key, consumer key ve consumer secret değerlerini aşağıdaki şekilde kopyalayın. Bu değerler Twitter API'ya bağlanmak için gerekli şifrelerdir.

```
import tweepy

access_token = '1573585004-Dvh9nr02G7xVjotih20VLe4HyxjhdbcONGZNODg'
access_token_secret = 'JmF5bJlQ2YFCQTie6iHlwIu5wlk4jZQXwi4iuDx3FIKWN'
consumer_key = 'br6tkQDdlXr50LNULNxfnNuEtW'
consumer_secret = 'PTo51V8T5TPtFftfx7PqggmXqVi0OR9PMmLpSypSQDPCKM188yk'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

# Kendi twitter kullanıcıyı yazıyorum
user = api.get_user('nalxrarekli')

# Kullanıcı adımı yazdır
print(user.screen_name)
```

196

```
# Kaç tane takipçim var?
print(user.followers_count)
```

```
# Takipçilerimi yazdır
for friend in user.followers():
    print(friend.screen_name)
```

```
giulioravizza
TUTORIALCOMPUTI
python_4coders
kodla_Turkiye
copyABrain
padicCharles
ButunErol
muthisbilimnet
SimPolProject
...
```

Şimdi Twitter'dan birkaç tweet çekelim.

```
api = tweepy.API(auth)
```

```
public_tweets = api.home_timeline()
for tweet in public_tweets:
    print(tweet.text)
RT @gnayyar: @wareFLO @Colin_Hung @JoeBabaian @CoherenceMed @EMRAAnswers @RasusShrestha @techguy @drstclaire @dmnic1 Thanks so much, Chuck! L. Top story: UK's Carphone Warehouse fined nearly $540k for 2015 hack | TechCru.. https://t.co/42WPuAqh2l, see more https://t.co/16gq31CqEh
RT @astropy: On the #arxiv today: a new #astropy paper describing the status of the project, and key updates since our original v0.2 paper!... 5 facts about Iran - The public unrest that swept across Iran starting in late
```

Gelen liste çok uzun ancak ben fazla yer tutmaması için sadece ilk satırları aldım. Tweepy modülünün kullanım amaçlarından birisi de bir oturum sırasında çok sayıda twitter versiyonunu indirebilmektir. Bu da Tweepy modülünde geliştirilmiş olan StreamListener<sup>34</sup> sınıfının işlevi耳. Önce, StreamListener sınıfından devralan yeni bir sınıf oluşturulur. Daha sonra api nesnesi ve StreamListener nesnesi kullanılarak akış nesnesi oluşturulur. Son olarak istenilen akış yakalamak için filter metodunu kullanılır<sup>35</sup>. Örneğimizde, içinde

<sup>34</sup> Akış Dinleyicisi olarak çevirebiliriz. Açılmış olan Twitter oturumundaki akış yakalamak için kullanılır.

<sup>35</sup> Buradaki örneği tweepy modülünün <http://docs.tweepy.org/en/v3.5.0/streaming.html#step-1-creating-a-streamlistener> sayfasından aldım.

## VERİ OKUMA

## PYTHON ile VERİ BİLİMİ

python kelimesi geçen tweetleri yakalamasını istedik. Aşağıdaki kodu çalıştırıldığınızda, siz durdurana kadar, içinde python kelimesi geçen bütün twitlerin canlı olarak filtrelenip yazdırıldığını göreceksiniz.

```
import tweepy, json

access_token = '1573585004-Dvh9nr0ZG7xVjotih2OVLe4HyxjhdbcONGZNODg'
access_token_secret = 'JmF5bj1Q2YFCQTie6iHlwIu5wlk4jZQXwi4iuDx3FIKWN'
consumer_key = 'hr6tkQdd1xr50LNULNxIJuEtW'
consumer_secret = 'PTo51V8T5TPtFftfx7PqgmXqVi0OR9PMmLpSypSQDPCKM188yk'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

class MyStreamListener(tweepy.StreamListener):

    def on_status(self, status):
        print(status.text)

myStreamListener = MyStreamListener()
myStream = tweepy.Stream(auth = api.auth, listener=myStreamListener)

myStream.filter(track=['python'])

RT @tkb: Predicting poverty is hard - can you do it better? A new data science competition from @worldbankdata - https://t.co/zI5vEklnRs #b...

In part three of his #SQLServer #MachineLearningServices series, Rob demonstrates matplotlib, a 2D plotting library.. https://t.co/eThDetjtrs

RT @Electronhacks: @rondagdag @haroldpulcher we need to buy some of these micro python dev boards! #CES2018 https://t.co/5NfuoCYtuU

termnote 1.1.1: A simple, yet powerful and handy terminal based note taking python app https://t.co/lcowCVZSw3

RT @KirkDBorne: Step-by-Step Guide to Learn #Python for #DataScience in 3 days: https://t.co/pGQBtWgTcQ #abdsc #BigData #MachineLearning #P...
```

...

198

## 10.6. Büyük Dosyalardan Veri Okuma

### VERİ OKUMA

Zaman zaman çok büyük dosyalardan veri okumamız gerekebilir. Ancak özellikle günümüzde verilerin çok büyük boyutlara ulaşabileceğini düşünürsek bilgisayar belgesi dosyanın tamamını okumaya yetmeyecek. Bu durumda, önceki bölümlerde gördüğümüz yineleyiciler yardım ile dosayı parça parça okuyup istediğimiz işlemleri bitirdikten sonra dosyadan bir miktar daha veri okuyarak dosyanın tamamını okuyabiliriz. Bu işlemde yineleme nesne okunacak dosya oluyor. Bir yineleyici yardımı ile dosyayı tarayıp verileri parçalar halinde alabiliriz.

Günlük kullanımda en yaygın veri formatlarından birisi csv formatıdır. Çok büyük ölçülü veriler daha çok bu formatta saklanır. Csv formatındaki verileri okumak için pandas paketinde yer alan `read_csv()` fonksiyonundan faydalanağız. Bu fonksiyona, argüman olarak okunacak dosyanın adı ve her yinelemede okunacak verinin büyütüğü (chunksize) atanır. Bu fonksiyon ile okunan veriler, veri çerçevesi formatındadır.

Varsayılm ki bir csv dosyasındaki bir sütunda yer alan verileri okumak istiyoruz. Bu veriler de A sütununda olsun.

```
1 import pandas as pd
2 sonuc = []
3
4 for veri in pd.read_csv('veri_dosyası.csv', chunksize = 100):
5     sonuc.append(veri['A'])
```

Şimdi de tamamı tek seferde okunamayacak kadar büyük bir dosyanın bir sütununda yer alan kategorik değişkenlerin her birinden kaçar adet olduğunu bulmak isteyelim. Okuyacağımız veri `kredi.csv` dosyasında yer alan kredi türü sütunu olsun. Bu sütunda yer alan her bir kredi türünden kaç adet olduğunu bulacağız.

```
1 import pandas as pd
2
3 krediler = {}
4
5 for veri in pd.read_csv("../krd.csv", chunksize = 100):
6     for kredi in veri['Kredi_Turu']:
7         if kredi in krediler.keys():
8             krediler[kredi] += 1
9         else:
10             krediler[kredi] = 1
11
12 print(krediler)
```

{'Kredi Kartı': 7793,  
'Konut Kredisи': 533,

199

'İhtiyaç Kredisi': 7298,  
'Ek Hesap': 3117,  
'Ticari Kredi': 1737}

## 10.7. Veri Dosyası Yazdırma

Şimdiye kadar hep verileri nasıl yazacağımızı gördük. Şimdi de çalıştığımız bir veriyi dosyaya kaydetmeyi görelim. Pandas modülü dosya kaydetmek için farklı yöntemler sunmaktadır. Çalıştığımız bir veri çerçevesini csv veya excel dosyası olarak kaydetmek istiyorsak Pandas modülündeki `.to_csv()` ve `.to_excel()` metodlarını kullanabiliriz. Çalıştığımız veri çerçevenin adı veri olsun. Önce veri çerçevesini csv dosyası olarak kaydedelim. Kaydedeceğimiz veriye `.to_csv()` metodunu uyguluyoruz. Argüman olarak kaydedeceğimiz dosyanın ismi ve sütun ayracının ne olması gerektiğini (virgül noktalı virgül, sekme vb.) yazarız.

```
import pandas as pd  
veri.to_csv('veri_dosyasi.csv', sep = ', ')  
Aynı veriyi excel olarak kaydetmek artık tahmin edebileceğiniz gibi aşağıdaki şekilde ya-  
pılır.  
  
veri.to_excel ('veri_dosyasi.xlsx', sep = ', ')
```