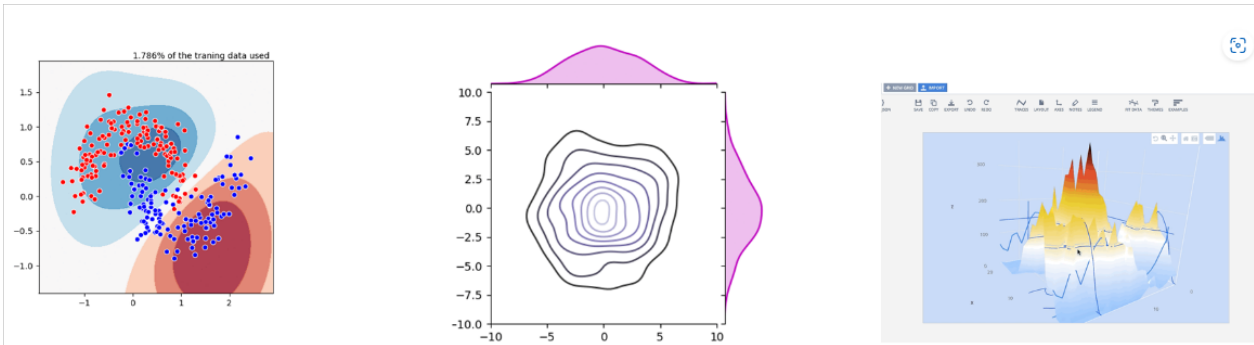
Data Visualization With Python

stackplot

VERİ GÖRSELLEŞTİRME

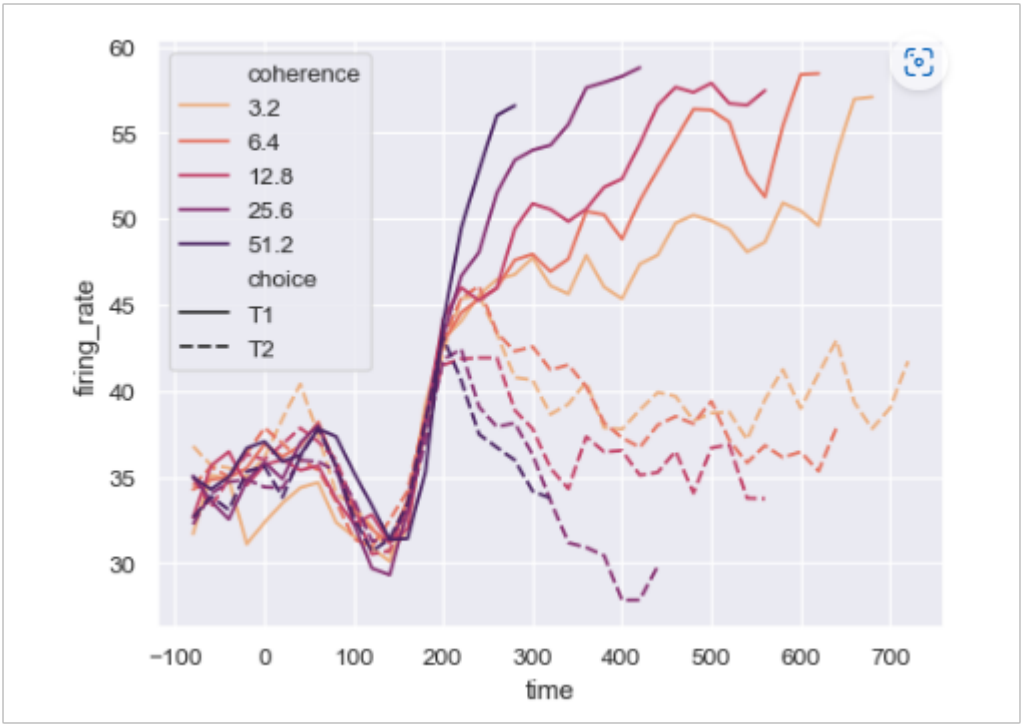


Verilerin görselleştirilmesi uygulamanın anlaşılabilirliği adına en önemli adımlardan biridir. Ayrıca verinin tanımlanmasından analiz sonuçlarının elde edilmesine kadar her adımda kullanılmaktadır. Python'da veri görselleştirmede Matplotlib ve Seaborn kütüphaneleri en yaygın kullanılanlar arasındadır. Özellikle veri bilimciler için çalışmada önemli hedefler, verilerden bilgi çıkarmak ve bunları sunmaktır. Ancak sonuçların sunumunda hedef kitlenin her zaman programlama ve yeterli teknik bilgiye sahip olamayabileceği açıktır. Bu nedenle, görselleştirme araçları sonuçları daha anlaşabilir hale getirmektedir. Bu bölümde, veri setlerinde kullanılabilecek farklı görselleştirme araçları incelenmiştir.

Bunlar aşağıdaki gibidir.

- Çizgi grafiği
- Çubuk grafikler
- Dağılım grafiği
- Alan ve birikimli alan grafikleri
- Pasta grafikler
- Tablo grafiği
- Kutup şeması
- Histogram
- Lolipop grafiği
- Isı haritaları
- Üç boyutlu grafikler

ÇİZGİ GRAFİKLERİ (Line Plot)



İki veya daha fazla sürekli değişken arasındaki ilişkiyi göstermek için çizgi grafiği kullanılır. "matplotlib.pyplot" kütüphanesi kullanılarak çizgi grafiği rahatlıkla oluşturulabilir. Kullanılan modül "plt.plot(veri. 1 , veri. 1 , color='kod: ls='kod: marker=' ')" şeklindedir. Burada "ls" çizgi sitillerini göstermek üzere Tablo 2'den herhangi biri seçilebilir.

Tablo 2. Çizgi Stilleri

Karakter	Kodu	Tanımı
'-'	'solid'	Düz çizgi stili
'--'	'dashed'	Kesikli çizgi stili
'-.'	'dashdot'	Kısa çizgi stili
':'	'dotted'	Noktalı çizgi stili

Çizgi stili seçimi sonrasında Tablo 3'de bulunan renk kodlarından herhangi biri seçilerek "color='kod'" modülü ile çizgilerin rengi değiştirilebilir.

Çizgi stili seçimi sonrasında Tablo 3'de bulunan renk kodlarından herhangi biri seçilerek "color='kod'" modülü ile çizgilerin rengi değiştirilebilir.

Tablo 3. Çizgi Renkleri	
Kodu	**Rengi**
'b'	Mavi
'g'	Yeşil
'r'	Kırmızı
'c'	Turkuaz
'm'	Eflatun
'y'	Sarı
'k'	Siyah
'w'	Beyaz

Grafik üzerinde bulunan verilerin kesişim noktalarının gösterilmesi için Tablo 4'den herhangi biri seçilerek "marker='kod"' modülü ile grafiğe eklenebilir.

Tablo 4. Grafik Çizim Noktaları	
Kodu	**Tanımı**
'.'	Nokta işaretçisi
'o'	Daire işaretçisi
'v'	Üçgen aşağı işaretçisi
'^'	Üçgen yukarı işaretçisi
'<'	Üçgen sol işaretçisi
'>'	Üçgen sağ işaretçisi
's'	Kare işaretçi
'p'	Beşgen işaretçi
'*'	Yıldız işaretçisi
'x'	x işaretçisi

Örnek: Sera gazı emisyonları değerleri bulunan ilgili dosyanın okutulması ve grafiğinin çizimi gösterilmiştir. Öncelikle dosyanın okutulması sağlanmıştır.

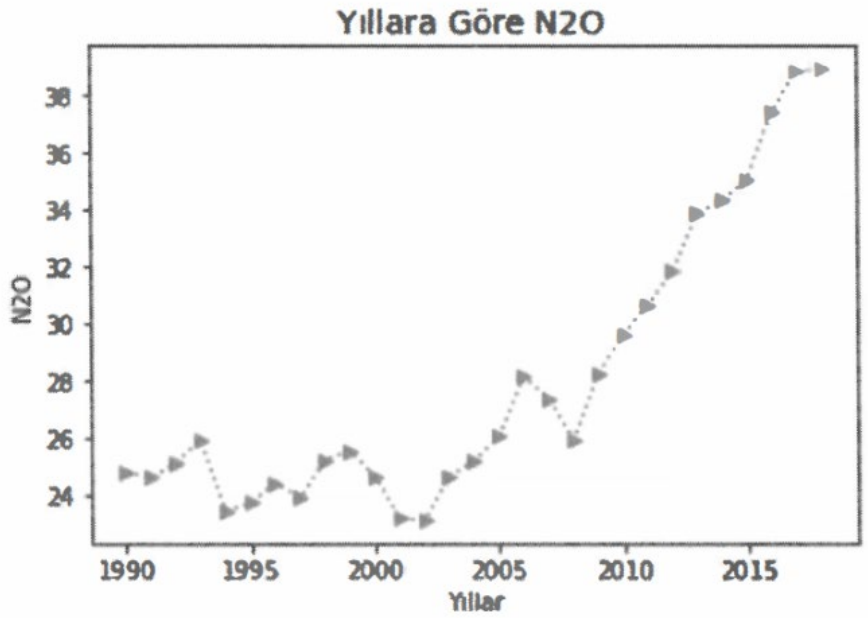
```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt

In [2]: veri= pd.read_excel('C:/Users/ACER/veri.xlsx', "Sayfa3")
        veri.head()
```

Out[2]:

	Yıl	Toplam	CO2	CH4	N2O	F-gazları
0	1990	219.367813	151.508468	42.405060	24.828987	0.625298
1	1991	226.756072	157.982005	43.285364	24.625361	0.863343
2	1992	232.975376	163.922250	43.193125	25.137414	0.722588
3	1993	240.316701	171.011163	42.965635	25.936824	0.403080
4	1994	234.287667	167.433113	42.683702	23.460852	0.709999

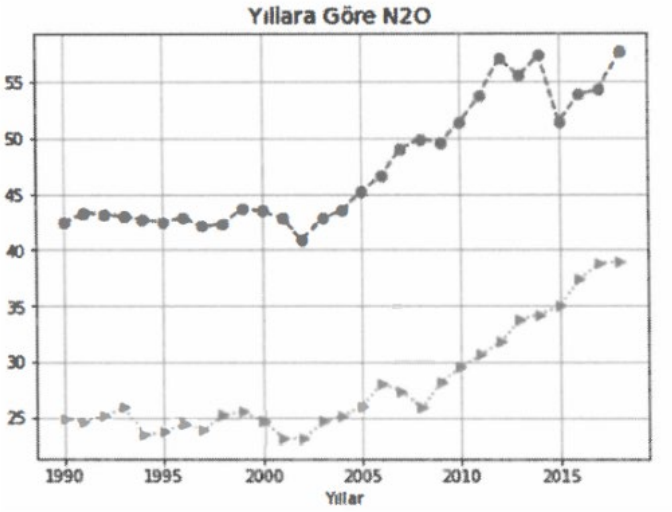
Grafik için koordinatlarının ve başlığının isimlendirilmesi yapılabilir. Ayrıca modül içerisinde "fontsize" kullanılarak yazı büyüklüğü ayarlanabilir ve "plt.rcParams[' figure.figsize'] = (7,5)" modülü grafiğin boyutunun ayarlanmasını sağlar.

```
In [3]: plt.plot(veri.Yıl, veri.N2O,color='c', ls='dotted', marker='>')
        plt.title("Yıllara Göre N2O", fontsize=14)
        plt.xlabel('Yıllar')
        plt.ylabel('N2O')
        plt.rcParams['figure.figsize'] = (7,5) #Grafiğin boyutunu ayarlar
        plt.show()
```

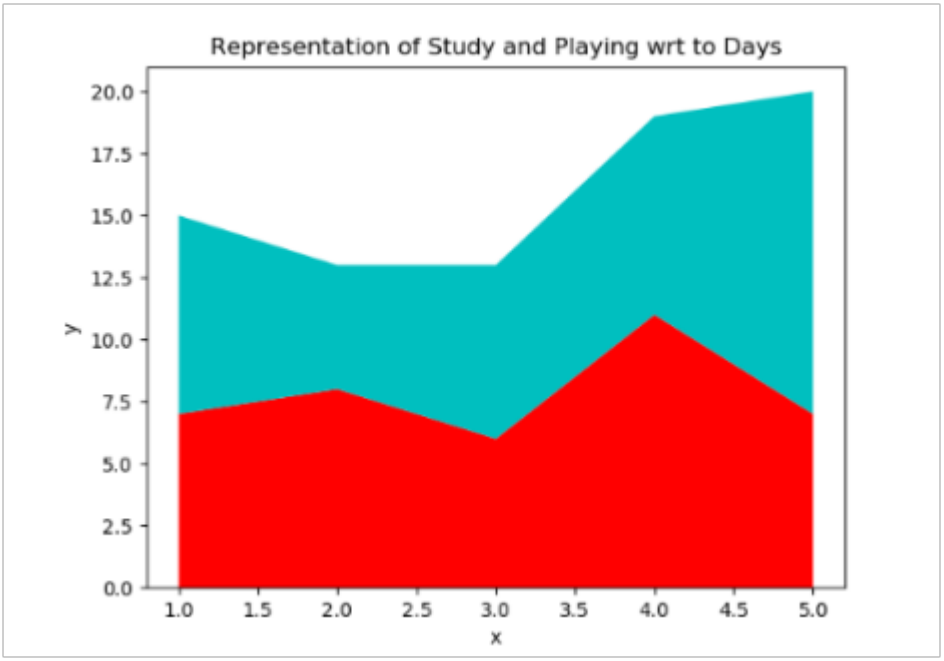


Bir çizgi grafiğinde birden fazla verinin grafiği eklenmek istenirse ilgili veri için "plt.plot(veri. 1 , veri. 1 , color='kod', ls='kod', marker=' ')" şeklinde aynı modülün eklenmesi yeterlidir.

```
In [4]: plt.plot(veri.Yıl, veri.N2O,color='c', ls='dotted', marker='>')
        plt.title("Yıllara Göre N2O", fontsize=14)
        plt.xlabel('Yıllar')
        plt.ylabel('N2O')
        plt.plot(veri.Yıl, veri.CH4,color='r', ls='dashed', marker='o') #2. grafik
        plt.rcParams['figure.figsize'] = (7,5)
        plt.grid(True) #klavuz çizgiler ekler
        plt.show()
```



Birikimli Alan Grafikleri (Stack Plot)



Birikimli alan grafiği, adını bir çizgi grafiğinin altındaki alanı temsil etmesine ve bu tür birkaç grafiğin üst üste istiflenerek bir yığın hissi vermesinden dolayı almaktadır. Y

ekseninde çizilen birden çok değişkenin kümülatif etkisi görselleştirmek istenildiğinde birikimli alan grafikleri kullanılabilir.

Örnek: Excel dosyasında bulunan ülkelere ait sağlık verilerinin içe aktarılması ile birikimli alan grafiği uygulaması yapılmıştır.

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd

In [2]: df= pd.read_excel('C:/Users/ACER/data/saglik.xlsx', "Sayfa2")
        df.head()
```

Out[2]:

	Ülke	G1	G2	G3	G4	G5	G6	G7	G8	G9	...	G21	G22
0	DEU	29.96	11.33	8.64	100.00	99.22	87.0	0.1	3.7	2.2	...	2.5	11.500
1	USA	20.35	17.78	9.02	99.27	99.97	91.0	0.4	6.5	3.5	...	2.5	8.600
2	ARG	18.42	8.58	6.70	99.83	96.24	86.0	0.4	9.9	6.4	...	4.6	7.609
3	AUS	13.19	9.34	6.38	100.00	99.99	95.0	0.1	3.7	2.3	...	2.5	6.300
4	BRA	12.00	8.62	3.78	98.38	89.31	83.0	0.5	14.4	8.1	...	2.5	6.452

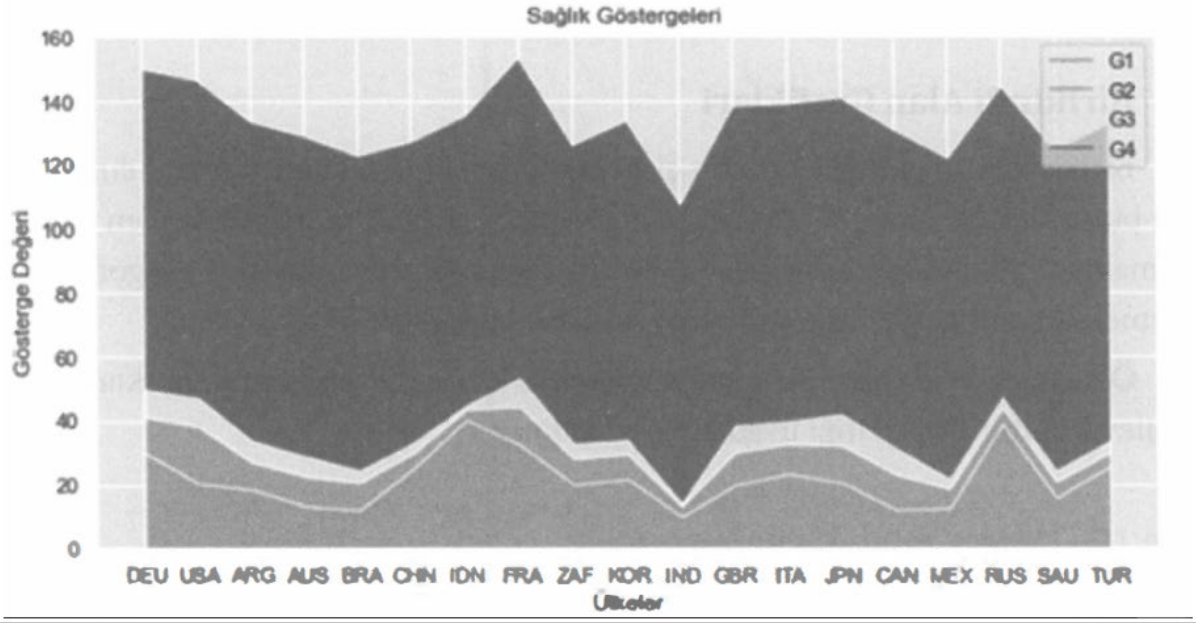
5 rows × 31 columns

Elde edilen veriler için birikimli alan grafiği aşağıdaki gibi çizilebilir.

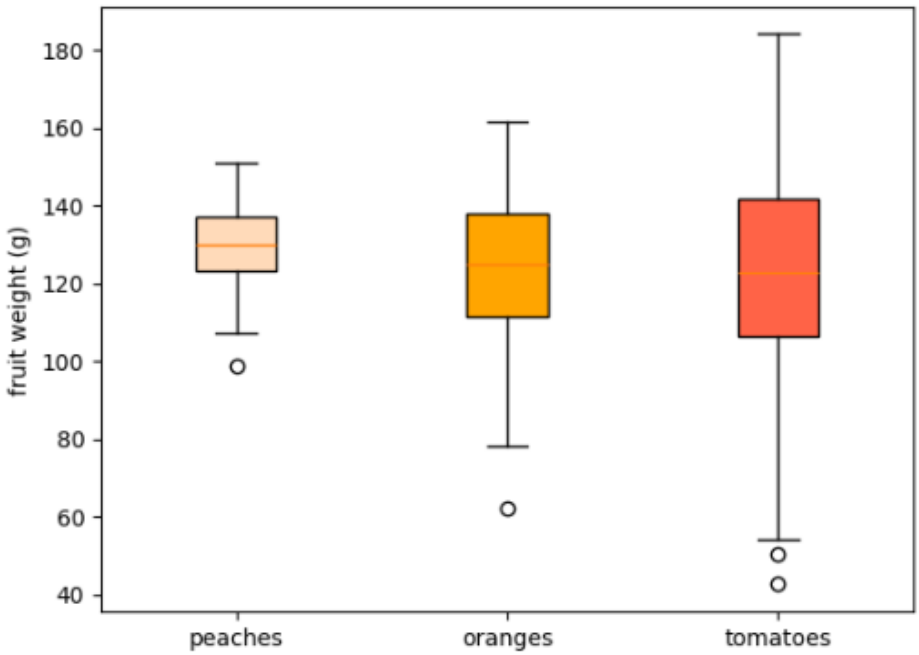
In [4]:

```
plt.plot([],[], color='sandybrown', label='G1')
plt.plot([],[], color='tan', label="G2")
plt.plot([],[], color='bisque', label='G3')
plt.plot([],[], color='darkcyan', label='G4')
plt.stackplot(df.Ülke, df.G1, df.G2, df.G3,
df.G4, colors=['sandybrown', 'tan', 'bisque', 'darkcyan'])
plt.rcParams['figure.figsize'] = (10,5) #Grafiğin boyutunu ayarlar
plt.legend()
plt.title('Sağlık Göstergeleri')
plt.xlabel('Ülkeler')
plt.ylabel('Gösterge Değeri')
```

Out[4]: Text(0, 0.5, 'Gösterge Değeri')



Kutu Grafikleri (Box Plot)



Kutu grafikleri tanımlayıcı istatistiklerin yanında verinin açıklanmasında uygun bir araçtır. Grafik üzerinde; aralık, çeyrekler arası aralık, medyan, mod, aykırı değerler ve tüm çeyrekler gösterilebilir.

Örnek: Endeks verilerinin bulunduğu dosya için kutu grafiği çizimleri yapılmıştır.

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        plt.style.use('ggplot')

In [2]: df=pd.read_csv("C:/Users/ACER/indeks.csv")
        df.head()
```

Out[2]:

	Tarih	Bist-100	Bovespa	İtaly40	KOSPI	Nikkei 225	BMVIIPC	Shanghai	Tadawul
0	2010-01-04	4.727282	4.845378	3.362313	3.229462	4.027545	4.515324	3.511049	3.792515
1	2010-01-05	4.733318	4.846583	3.362520	3.228046	4.028646	4.514983	3.516162	3.795122
2	2010-01-06	4.736774	4.849600	3.363744	3.231806	4.030658	4.516273	3.512447	3.796637
3	2010-01-07	4.740149	4.847888	3.365338	3.226200	4.028639	4.519363	3.504169	3.797341
4	2010-01-08	4.738764	4.846725	3.367207	3.229236	4.033356	4.517091	3.504607	3.798044

İlgili veri incelendiğinde birden fazla sütunun tanımlı olduğu endeksler görülmektedir. Öncelikle seçilecek olan endeks yani sütunda yer alan verinin seçilmesi gerekmektedir.

```
In [3]: x=df[["Tadawul"]].values
        x=x.flatten()
        x

Out[3]: array([3.79251496, 3.79512195, 3.79663677, ..., 3.92163836, 3.92142686,
               3.9237221 ])

In [4]: y=df[["Bist-100"]].values
        y=y.flatten()
        y

Out[4]: array([4.72728223, 4.73331806, 4.73677374, ..., 5.05569786, 5.05976772,
               5.05852077])

In [5]: z=df[["İtaly40"]].values
        z=z.flatten()
        z

Out[5]: array([3.3623128 , 3.36252017, 3.36374356, ..., 3.36513217, 3.35972168,
               3.35972168])
```

Sorgular incelendiğinde örneğin x değerleri Tadawul endeksinde bulunan değerlerdir. Değerlerin sütundan alınabilmesi için ".values" ifadesi sona eklenmiştir. Bu durumda x ifadesi her bir satır değerini bir dizi olarak alacaktır. Ancak uygulamada hepsinin tek bir dizi içinde yer alması gerektiğinden dolayı "x.flatten()" ifadesi eklenerek bu durum sağlanmıştır. Elde edilen sorgulama sonucunda çizilen kutu grafiği aşağıdaki gibidir. Burada üç ayrı endeks değeri için inceleme mevcuttur. Kutu grafiği "boxplot ()" modülü parametreleri aşağıdaki gibi tanımlanabilir.

• x: İncelenen verilerdir.

• vert: "False" olduğunda çizim yönünü yatay olarak ayarlar. Varsayılan yön dikeydir.

• showmeans: "True" olduğunda verilerin ortalamasını gösterir.

• meanline: "True" olduğunda bir çizgi olarak ortalamayı temsil eder. Varsayılan temsil bir noktadır.

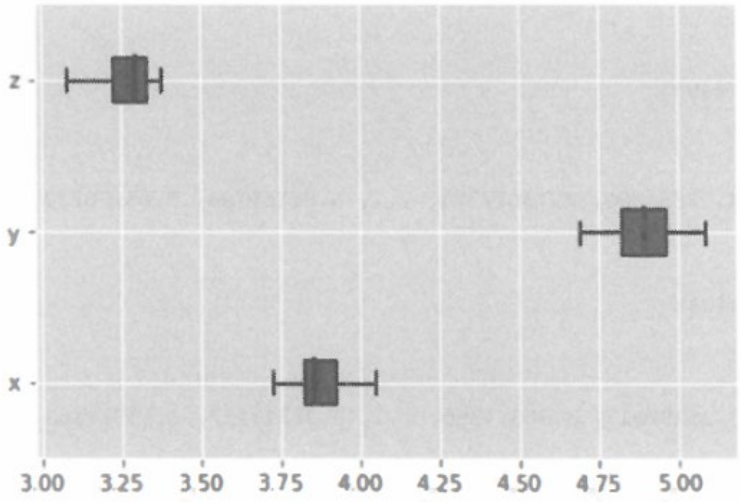
• labels: Verilerin etiketleridir.

• patch_artist: Grafiğin nasıl çizileceğini belirler.

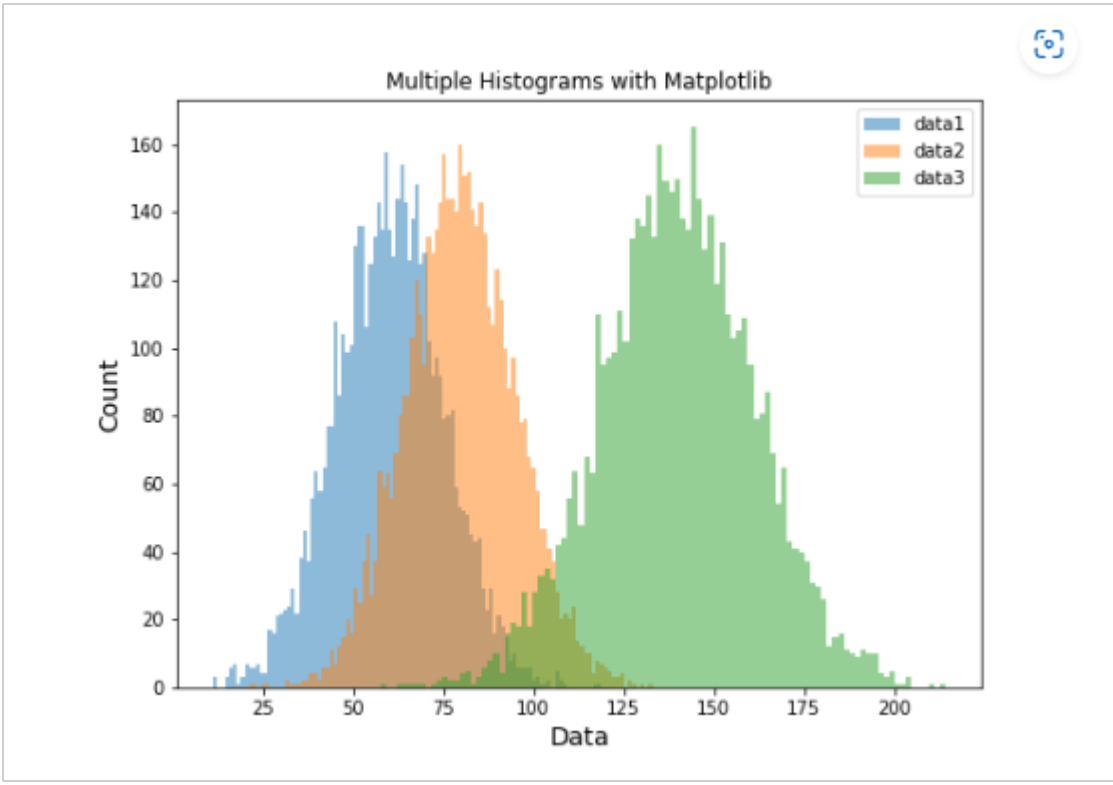
• medianprops: Medyanı temsil eden çizginin özelliklerini belirtir. Renk, kalınlık vb.

• meanprops: Ortalamayı temsil eden çizgi veya noktanın özelliklerini gösterir. Renk, kalınlık vb.

In [6]:
```
fig, ax = plt.subplots()
ax.boxplot((x, y, z), vert=False, showmeans=True, meanline=True,
           labels=('x', 'y', 'z'), patch_artist=True,
           medianprops={'linewidth': 2, 'color': 'purple'},
           meanprops={'linewidth': 2, 'color': 'red'})
plt.show()
```



Histogramlar



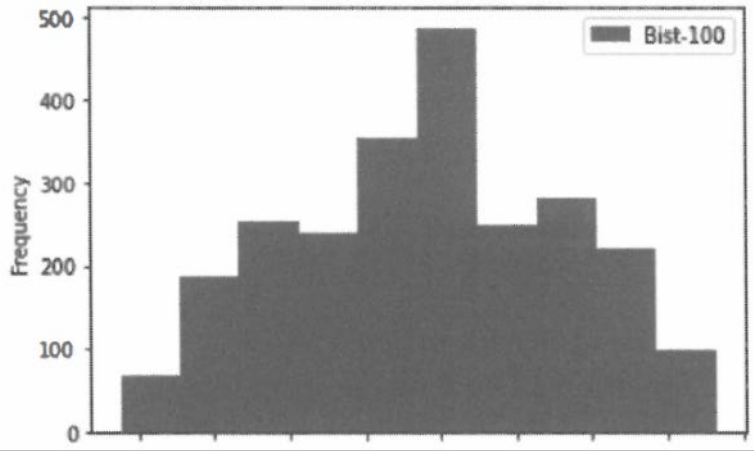
Histogram, sıralı bir veri kümesindeki değerleri bölmeler olarak da adlandırılan aralıklara böler. Bu grafik özellikle çok sayıda farklı değer olduğunda yararlıdır. Veri noktaları ayrı, eşit aralıklı bölmelere bölünür ve her bölmedeki veri noktalarının sayısı çizilir. Bir bölmenin alt ve üst sınır değerlerine bölme kenarları denir. Histogram, değerlerin frekansının ayrık bir görüntüsünü veren bir tür çubuk grafiğidir. Frekans, her bölmeye karşılık gelen tek bir değerdir ve bölmenin kenarları arasındaki değere sahip öğe sayısıdır. Örneğin; veri kümesi O, 5, 1 0 ve 1 5 bölme kenarlarıyla bölünürse, üç bölme elde edilir. İlk ve en soldaki bölme, Ü>a eşit veya O>dan büyük ve 5>ten küçük değerleri içerir. İkinci bölme, 5>e eşit veya 5>ten büyük ve l O>dan küçük değerleri içerir. Üçüncü ve en sağdaki bölme, lO>a eşit veya l O>dan büyük ve 1 5>e eşit veya daha küçük değerleri içerir. Histogram grafiğini elde etmenin birden fazla yolu mevcuttur. İlk olarak Pandas kütüphanesi ile verinin okutulması sonrası " df[["Sütunadı"]] .plot.hist(bins= l O)" komutu ile elde edilebilir. "bin= l ü" modülü ise bölmenin kenarlarını veya sınırlarını içerir. Yani 1 O adet bölme oluşturulmuştur.

```
In [1]: import pandas as pd

In [2]: df=pd.read_csv("C:/Users/ACER/indeks.csv")
        df.head()
```

Out[2]:

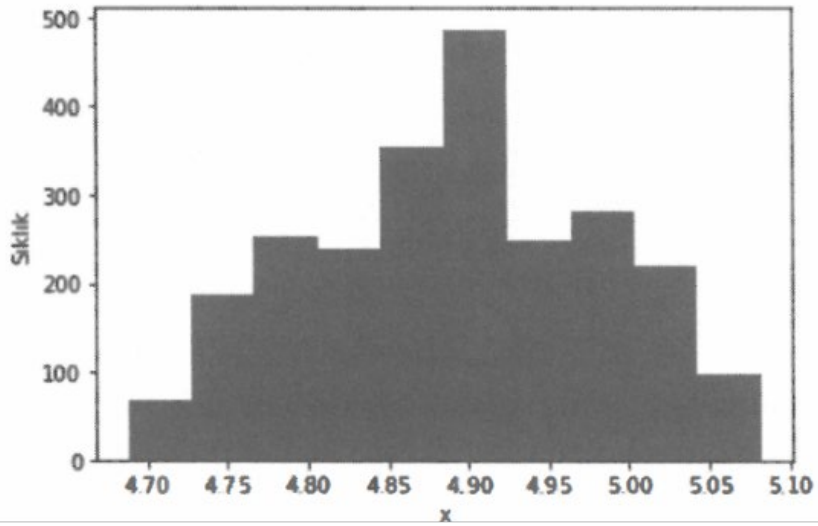
	Tarih	Bist-100	Bovespa	İtaly40	KOSPI	Nikkei 225	BMVIIPC	Shanghai	Tadawul
0	2010-01-04	4.727282	4.845378	3.362313	3.229462	4.027545	4.515324	3.511049	3.792515
1	2010-01-05	4.733318	4.846583	3.362520	3.228046	4.028646	4.514983	3.516162	3.795122
2	2010-01-06	4.736774	4.849600	3.363744	3.231806	4.030658	4.516273	3.512447	3.796637
3	2010-01-07	4.740149	4.847888	3.365338	3.226200	4.028639	4.519363	3.504169	3.797341
4	2010-01-08	4.738764	4.846725	3.367207	3.229236	4.033356	4.517091	3.504607	3.798044

```
In [3]: df[["Bist-100"]].plot.hist(bins=10)
Out[3]: <AxesSubplot:ylabel='Frequency'>
```



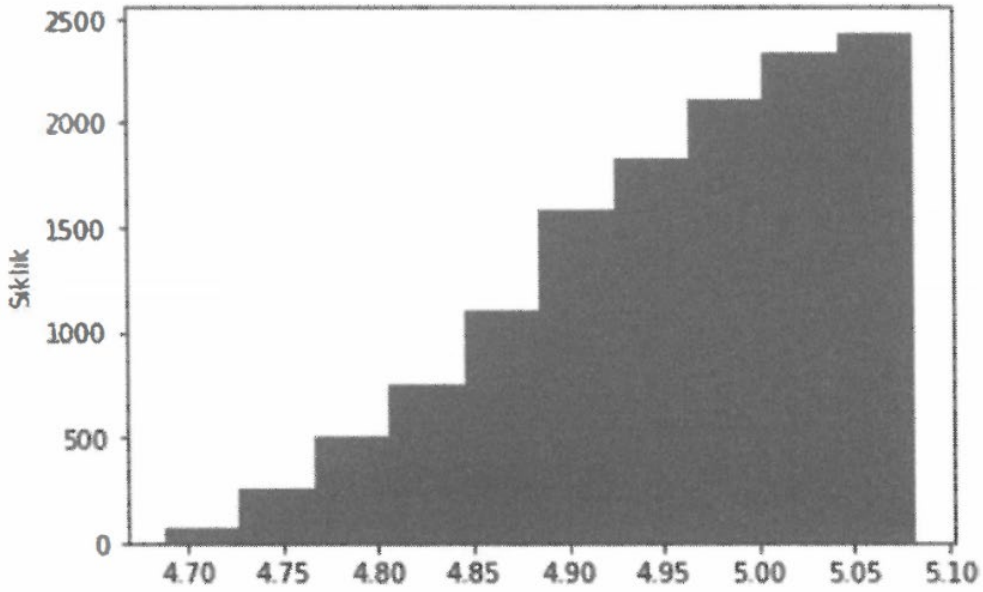
Aynı dosya üzerinde ayrıca "np.histogram()" modülü ile Numpy kütüphanesi kullanılarak elde edilebilir. Burada Pandas kütüphanesine ek olarak "import numpy as np'' ve "import matplotlib.pyplot as plt" kütüphanelerinin eklenmesi gerekmektedir.

```
In [3]: y=df[["Bist-100"]].values
        x=y.flatten()
        x

Out[3]: array([4.72728223, 4.73331806, 4.73677374, ..., 5.05569786, 5.05976772,
               5.05852077])

In [4]: hist, bin_edges = np.histogram(x, bins=10)
        bin_edges

Out[4]: array([4.68788045, 4.72731537, 4.7667503 , 4.80618523, 4.84562016,
               4.88505509, 4.92449002, 4.96392494, 5.00335987, 5.0427948 ,
               5.08222973])

In [5]: fig, ax = plt.subplots()
        ax.hist(x, bin_edges, cumulative=False)
        ax.set_xlabel('x')
        ax.set_ylabel('Sıklık')
        plt.show()
```



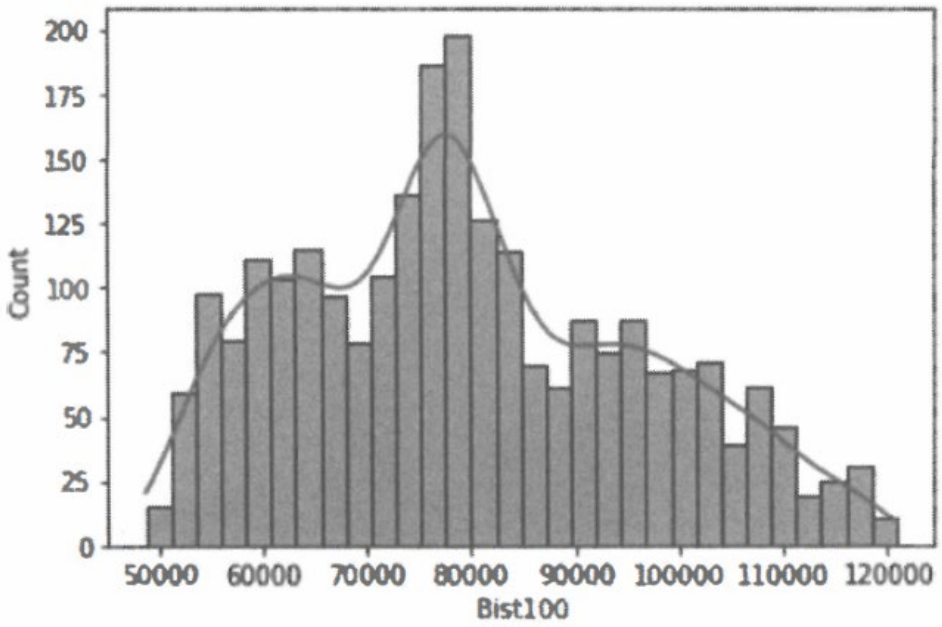
Burada "hist" her bölmeye karşılık gelen öğelerin sıklığını veya sayısını içerir. Eğer modüle "cumulative=True" eklenirse birikimli dağılım elde edilir.

```
In [7]: fig, ax = plt.subplots()
        ax.hist(x, bin_edges, cumulative=True)
        ax.set_xlabel('x')
        ax.set_ylabel('Sıklık')
        plt.show()
```

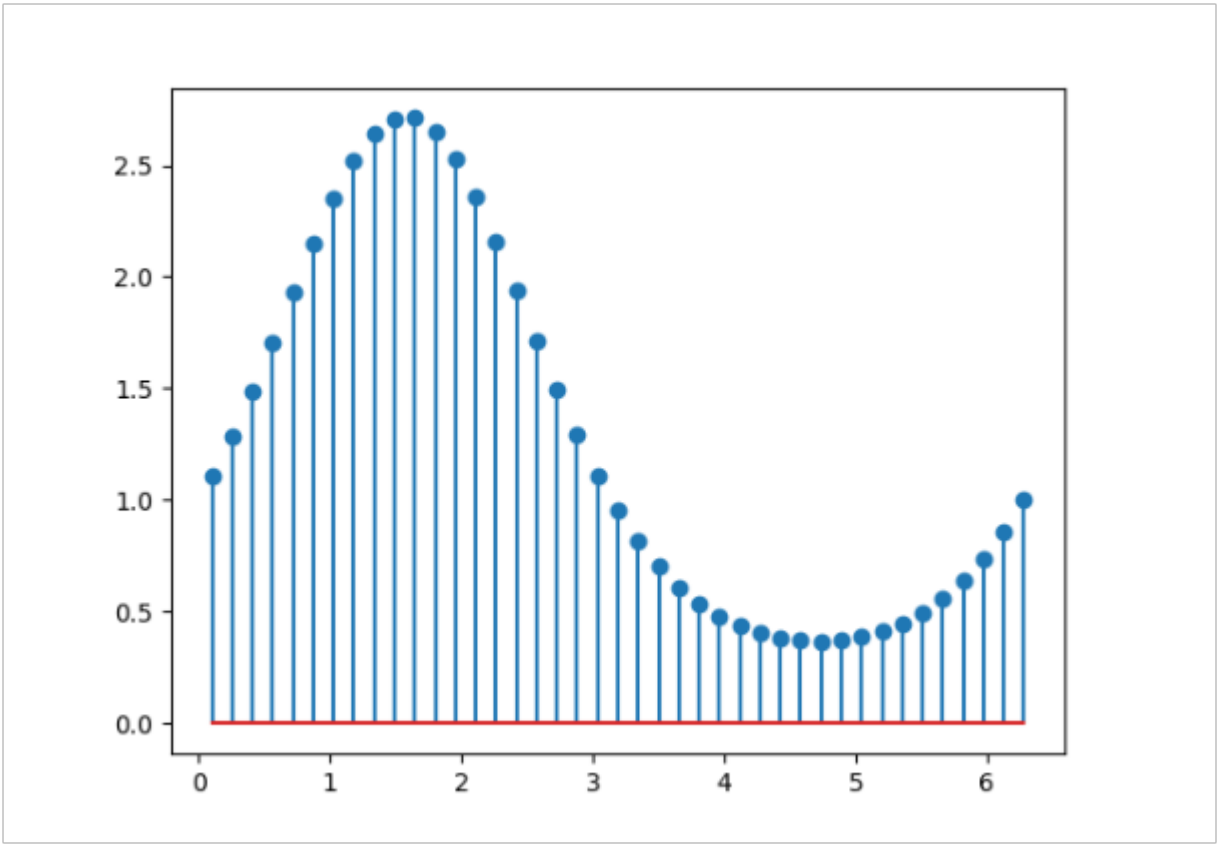


Bu grafikte ilk ve en soldaki bölmenin sıklığı, bu bölmedeki öğelerin sayısıdır. İkinci bölmenin sıklığı, birinci ve ikinci bölmelerdeki öğe sayılarının toplamıdır. Diğer bölmeler de aynı düzeni izler. Son olarak, son ve en sağdaki bölmenin sıklığı bu veri kümesinde 2432 olan toplam öğe sayısıdır Seaborn kütüphanesinde yapılan histogram uygulamaları aşağıdaki gibidir. Verinin bu bölümde "df" yerine "veri" terimi ile gösterilmiştir. "kde=True" normalleştirilmiş verinin çizgi grafiğini verir.

```
In [9]: import seaborn as sns #seaborn kütüphanesi

In [10]: sns.histplot(x=veri.Bist100,bins=30, kde=True)

Out[10]: <AxesSubplot:xlabel='Bist100', ylabel='Count'>
```



Lolipop Grafiği (Stem)



Verilerdeki sıralamanın görsel olarak verilmesi için lolipop grafiği kullanılabilir. Örnek: Excel dosyasında bulunan ülkelere ait sağlık verileri içe aktarılmış ve lolipop grafiği çizilmiştir.

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd

In [2]: df= pd.read_excel('C:/Users/ACER/data/saglik.xlsx', "Sayfa2")
        df.head()
```

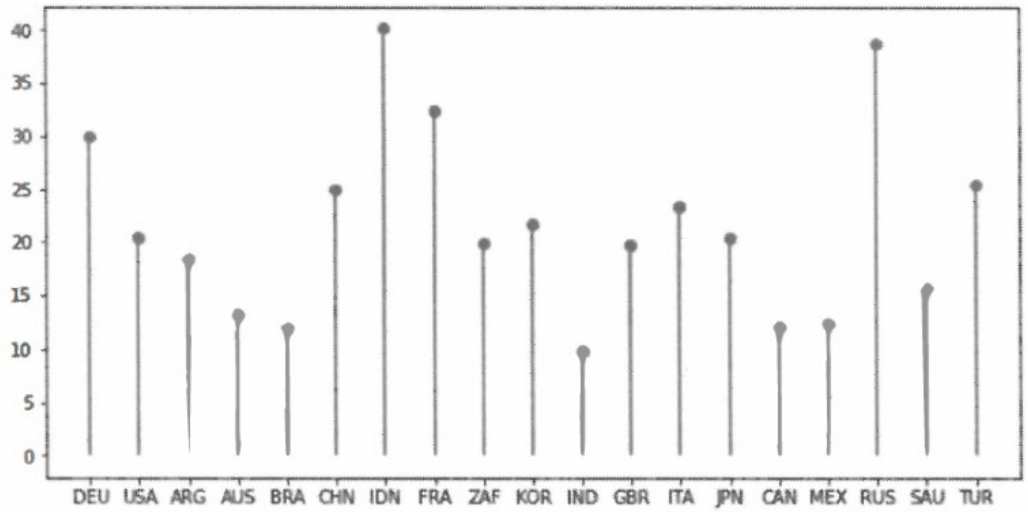
Out[2]:

	Ülke	G1	G2	G3	G4	G5	G6	G7	G8	G9	...	G21	G22
0	DEU	29.96	11.33	8.64	100.00	99.22	87.0	0.1	3.7	2.2	...	2.5	11.500
1	USA	20.35	17.78	9.02	99.27	99.97	91.0	0.4	6.5	3.5	...	2.5	8.600
2	ARG	18.42	8.58	6.70	99.83	96.24	86.0	0.4	9.9	6.4	...	4.6	7.609
3	AUS	13.19	9.34	6.38	100.00	99.99	95.0	0.1	3.7	2.3	...	2.5	6.300
4	BRA	12.00	8.62	3.78	98.38	89.31	83.0	0.5	14.4	8.1	...	2.5	6.452

5 rows × 31 columns

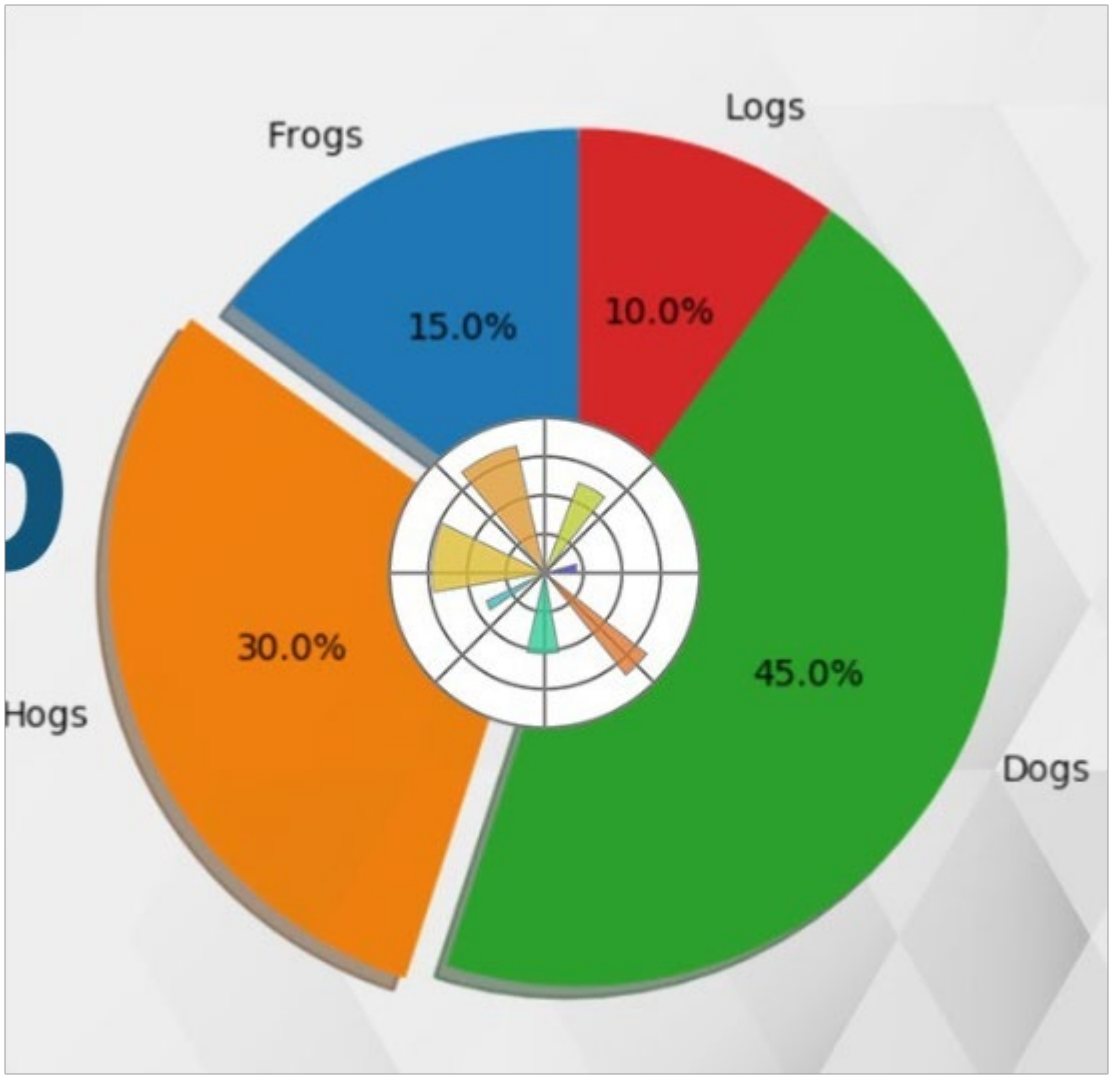
Elde edilen veriler için lolipop grafiğinin çizimi aşağıdaki gibi yapılabilir.

```
In [5]: fig, ax = plt.subplots()
        ax.stem(df.Ülke, df.G1, basefmt=' ')
        plt.rcParams['figure.figsize'] = (10,5) #Grafiğin boyutunu ayarlar
```



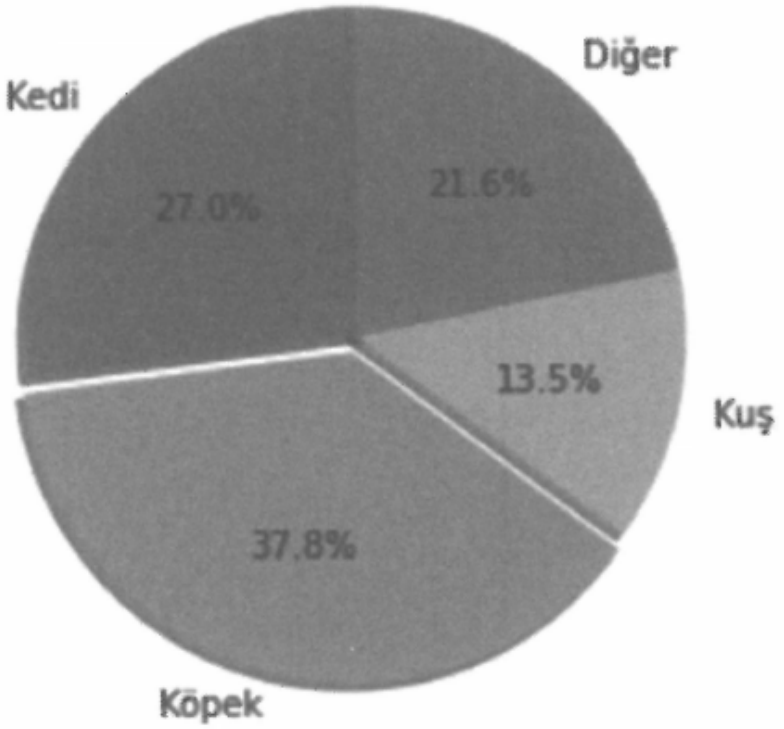
Pasta Grafikleri (Pie Charts)

Pasta grafiklerde her dilim veri kümesinden tek bir etikete karşılık gelir ve frekanslarının oranına sahip parçayı temsil eder. Nominal veriler gibi sıralanamayan etiketlerle de işlem yapılabilir. Pasta grafiği, birden çok dilime bölünmüş bir dairedir.


Frogs
Logs
15.0%
10.0%
30.0%
45.0%
Hogs
Dogs

```
In [1]: import seaborn as sns
        import matplotlib.pyplot as plt

In [2]: txtLabels = 'Kedi', 'Köpek', 'Kuş', 'Diğer'
        fractions = [50, 70, 25, 40]
        offsets =(0, 0.05, 0, 0)

In [3]: plt.pie(fractions, explode=offsets, labels=txtLabels,
        autopct='%1.1f%%', shadow=True, startangle=90,
        colors=sns.color_palette('muted') )
        plt.axis('equal')

Out[3]: (-1.1108134987308598,
         1.1083517787460202,
         -1.1697258062766704,
         1.1033202764893653)
```

\



Bu grafiğin oluşmasında yer alan modülde kullanılan ifadelerin açıklamaları aşağıdaki gibidir.

- txtLabels: Her parça için tanımlı etiketi.
- fractions: Etiketler altında yer alan verilerin frekansları.
- offsets: İlgili parçanın şekilden uzaklığı.
- autopct: Şekilde gösterilen frekansların formatı.

Yüzdeler, toplamlarına kıyasla her bir değerin göreceli boyutunu belirtir. Örnek: Bir çalışmada yer alan katılımcılar için eğitim seviyesi pasta grafiği şeklinde gösterilmiştir. Uygulamaya öncelikle verilerin içe aktarılması ile başlanmıştır. ".values" ifadesi ile "okulturu" sütununda yer alan verilerin değerleri alınmıştır.

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt

In [2]: veri= pd.read_excel('C:/Users/ACER/data/test.xlsx')
        veri.head()
```

Out[2]:

	cinsiyet	medenidurum	egitimduzey	okulturu	cocuk	kaygiontest	kaygisontest	tutumontest
0	1	2	1	2	2	17	21	19
1	1	2	1	2	2	18	27	15
2	1	2	1	2	2	20	21	15
3	1	1	2	2	1	23	27	14
4	1	1	1	2	1	20	12	19

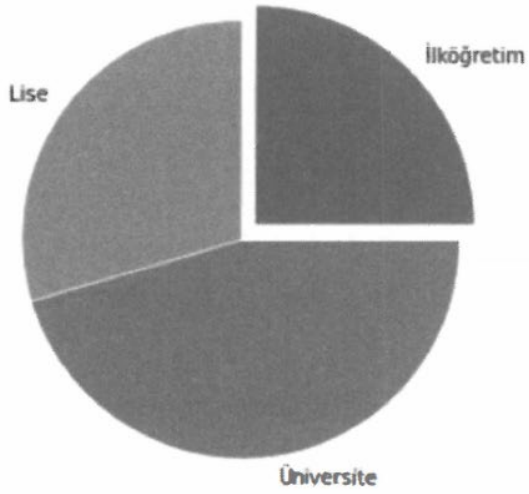
```
In [3]: okulturu=veri["okulturu"].values
        okulturu

Out[3]: array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3,
               3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
               2, 3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
```

Elde edilen dizi için okul türlerindeki katılımcı sayısı "len()" modülü ile belirlenmiştir.

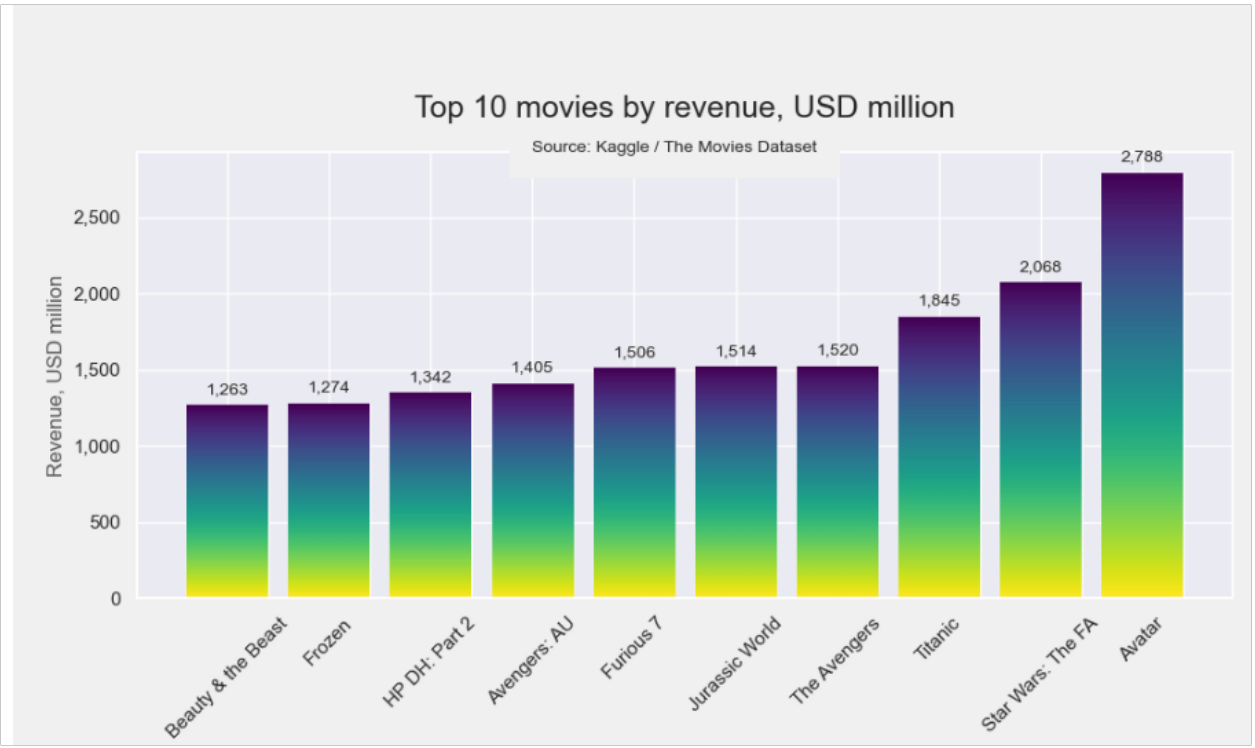
```
In [4]: İlk=len(okulturu[okulturu==1])
        Lise=len(okulturu[okulturu==2])
        Üniv=len(okulturu[okulturu==3])

In [5]: y=[İlk,Lise,Üniv]
        y

Out[5]: [66, 79, 121]
```

Okul türlerine ait sayılar ayrı ayrı bulunduktan sonra bunlar bir liste haline getirilerek pasta grafiği elde edilmiştir.

```
In [6]: etiketler = ["İlköğretim", "Lise", "Üniversite"]

In [7]: plt.rcParams['figure.figsize'] = (7,5)
        plt.pie(y, labels = etiketler, explode=[0.1,0,0.01]) # explode merkezden uzaklık
        plt.show()
```



Çubuk Grafikler (Bar Chart)



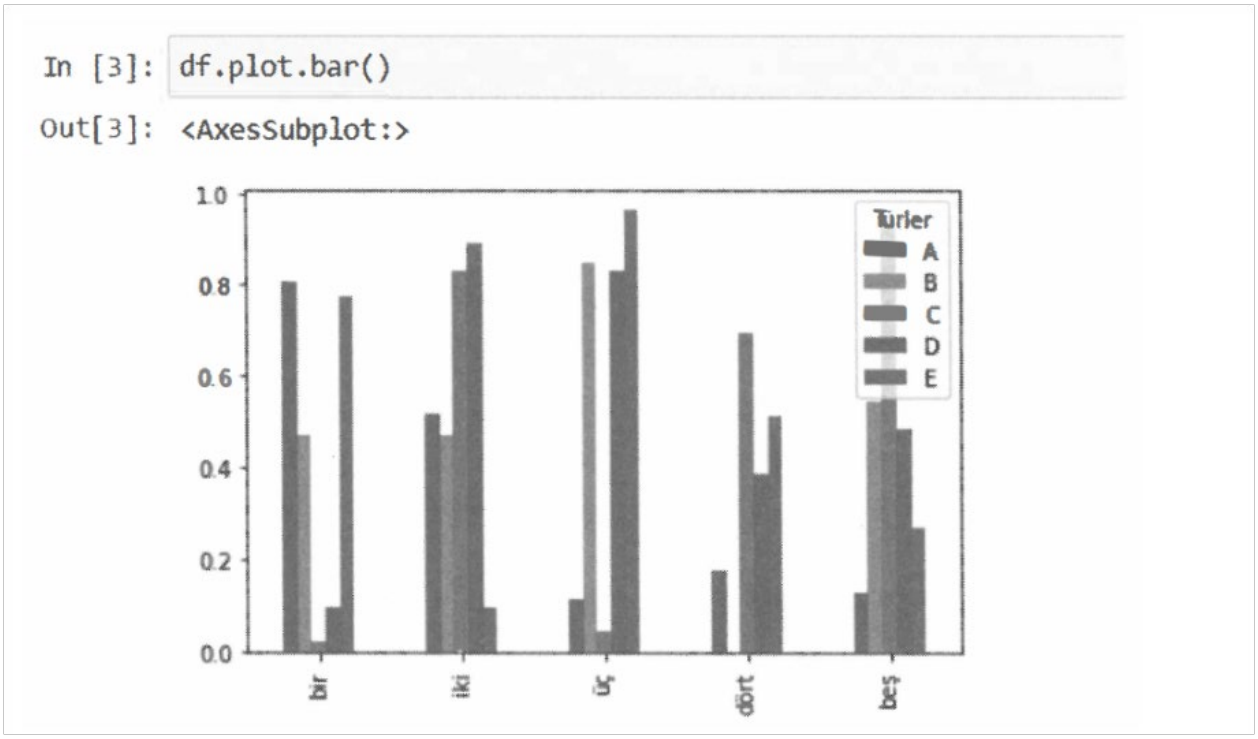
Çubuk grafikte şekiller dikdörtgen ve paralel çubuklardan meydana gelmektedir. Çubuk grafikler; belirli etiketlere veya farklı sayısal değerlere karşılık gelen verileri gösterir. Her bir çubuk görselde etiketleri ve buna karşılık gelen frekansları gösterir. "plot.bar ()" dikey ve "plot.barh ()" yatay çubuk grafikleri oluşturur.

```
In [1]: import pandas as pd
        import numpy as np

In [2]: df = pd.DataFrame(np.random.rand(5,5),
             index=['bir', 'iki', 'üç', 'dört', 'beş'],
             columns=pd.Index(['A', 'B', 'C', 'D','E'], name='Türler'))
        df
```

Out[2]:

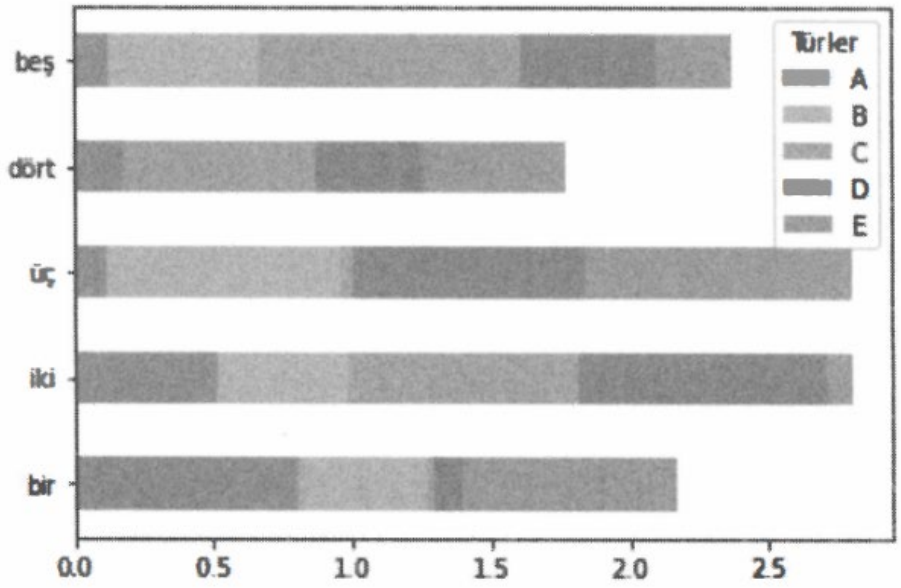
Türler	A	B	C	D	E
bir	0.804360	0.470039	0.024269	0.098217	0.772049
iki	0.517017	0.473383	0.829049	0.888529	0.100093
üç	0.117816	0.846081	0.047121	0.829200	0.962723
dört	0.176592	0.001007	0.695982	0.385601	0.512113
beş	0.130789	0.544178	0.938857	0.483692	0.273203

Pandas ve Numpy gerekli kütüphaneler eklendikten sonra veriler rasgele olarak üretilmiştir. Burada "np.random.rand(S,5)" satır ve sütun sayısı, "index" satır isimleri ve "columns" sütun isimlerini vermektedir. Dikey sütunlar ile oluşturacak ayrık sütunlu grafik aşağıdaki gibi oluşturulur.



Yatay olarak çubuk grafikleri de dördüncü sorgudaki gibi oluşturulabilir. Ayrıca veriler sütunlar ayrı olmak üzere tek bir sütun üzerinde değerleriyle orantılı olarak "stacked=True" modülüyle oluşturabilir. "alpha=0.5" değeri ise görselin konstrat ayarıdır.

```
In [4]: df.plot.barh(stacked=True, alpha=0.5)
Out[4]: <AxesSubplot:>
```



Örnek: Avrupa ülkeleri sağlık göstergeleri kullanılarak çubuk grafikleri "matplotlib.pyplot" kütüphanesi ile çizilmiştir.

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt

In [2]: veri= pd.read_excel('C:/Users/ACER/saglik.xlsx', "Sayfa2")
        veri.head()
```

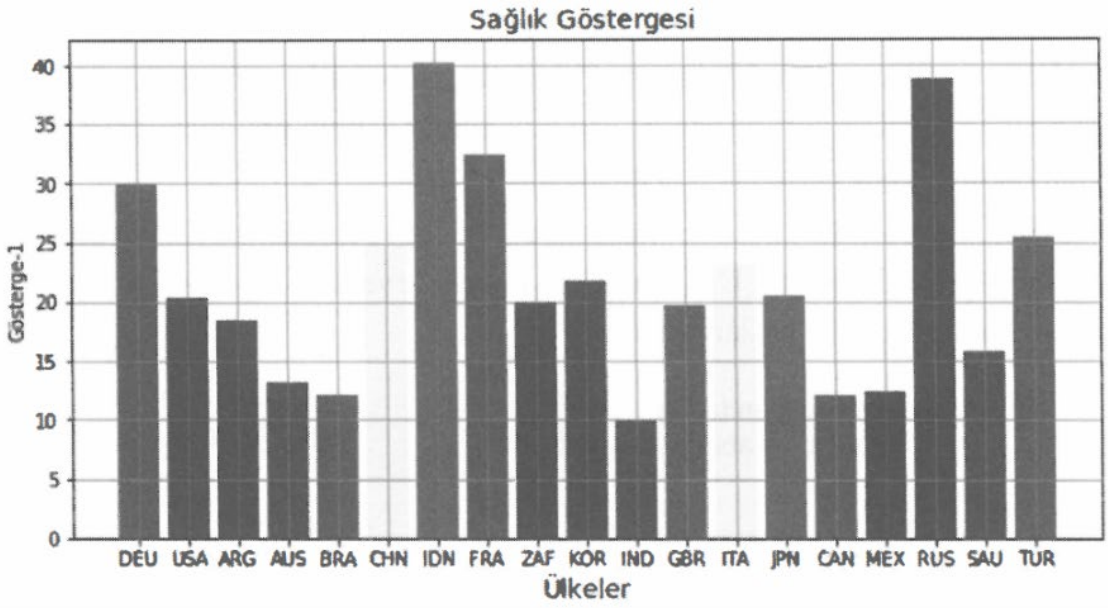
Out[2]:

	Ülke	G1	G2	G3	G4	G5	G6	G7	G8	G9	...	G21
0	DEU	29.96	11.33	8.64	100.00	99.22	87.0	0.1	3.7	2.2	...	2.5
1	USA	20.35	17.78	9.02	99.27	99.97	91.0	0.4	6.5	3.5	...	2.5
2	ARG	18.42	8.58	6.70	99.83	96.24	86.0	0.4	9.9	6.4	...	4.6
3	AUS	13.19	9.34	6.38	100.00	99.99	95.0	0.1	3.7	2.3	...	2.5
4	BRA	12.00	8.62	3.78	98.38	89.31	83.0	0.5	14.4	8.1	...	2.5

5 rows × 31 columns

Elde edilen veriler için birinci gösterge seçilerek ülkelerin dağılımı incelenmiştir.

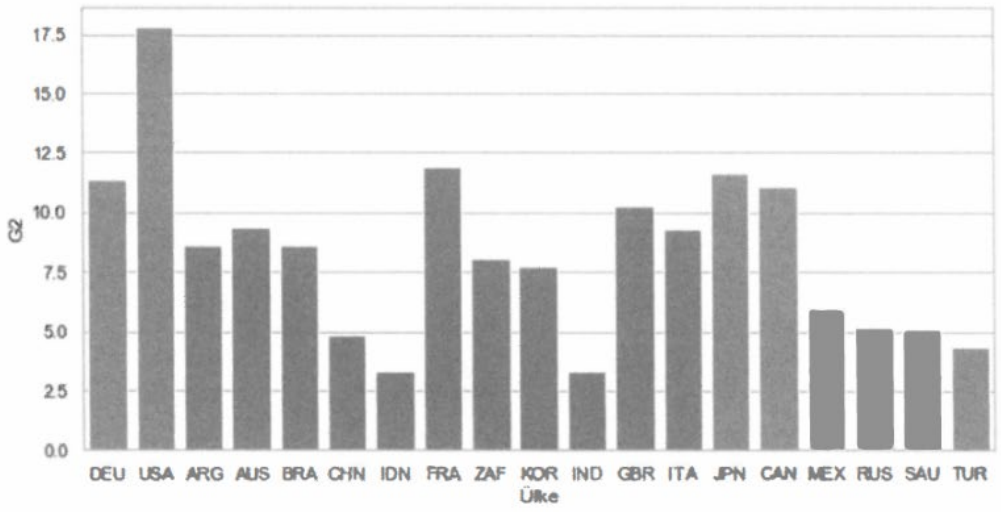
```
In [4]: renkler = ['green','blue','purple','brown','teal',"yellow","red"]
        plt.bar(veri.Ülke, veri.G1, color=renkler)
        plt.title('Sağlık Göstergesi',fontsize=14)
        plt.xlabel('Ülkeler',fontsize=14)
        plt.ylabel('Gösterge-1')
        plt.rcParams['figure.figsize'] = (10,5) #Grafiğin boyutunu ayarlar
        plt.grid(True)
        plt.show()
```



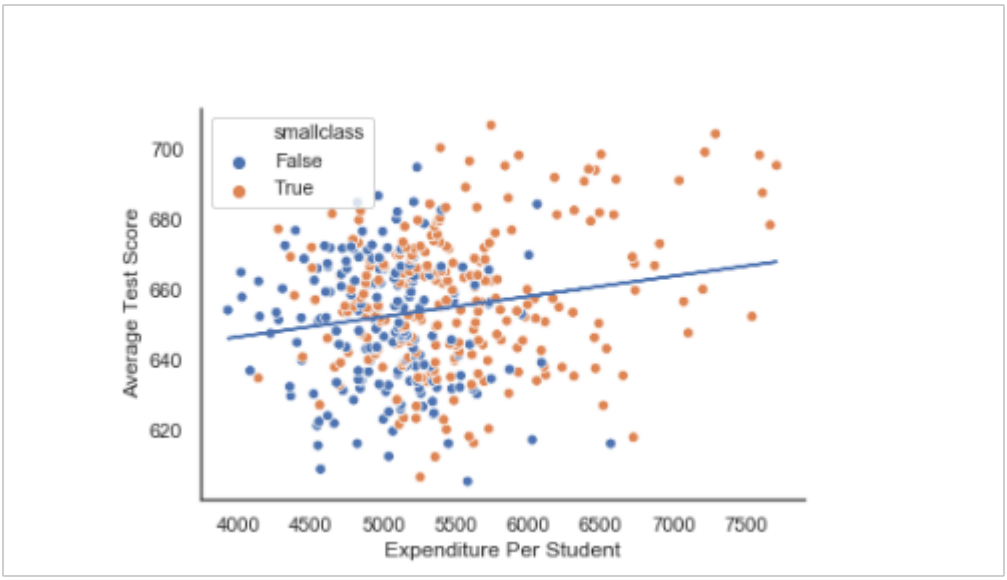
"plt.bar(veri.Ülke, veri.Gl , color=renkler)" modülü grafiğin çizilmesi için yeterlidir. "color" modülü için bir renk dizisi tanımlanarak bu aralıkta grafiğin renklendirilmesi sağlanmıştır. "fontsize" başlık ve etiketler için yazıların büyüklüklerini ayarlamaktadır. "plt.grid" modülü kılavuz çizgilerin eklenmesini sağlamaktadır.

Seaborn kütüphanesi ile gösterge 2 için grafik aşağıdaki şekilde bulunabilir.

```
In [9]: import seaborn as sns  #seaborn kütüphanesi

In [10]: sns.set_theme(style="whitegrid")
         ax = sns.barplot(x="Ülke", y="G2", data=veri)
```



Dağılım Grafikleri



Nokta grafikleri veya dağılım grafikleri, iki tek boyutlu veri serisi arasındaki ilişkiyi incelemek ve görüntülemek için kullanılır. Dağılım grafikleri, basit olmalarına rağmen görselleştirme için güçlü araçlardır.

Örnek: Seaborn kütüphanesi kullanarak ithalat ve ihracat verilerine göre dağılım grafiği incelenmiştir. Uygulama için öncelikle veriler ve gerekli kütüphanelerin içe aktarılması ile başlanmıştır.

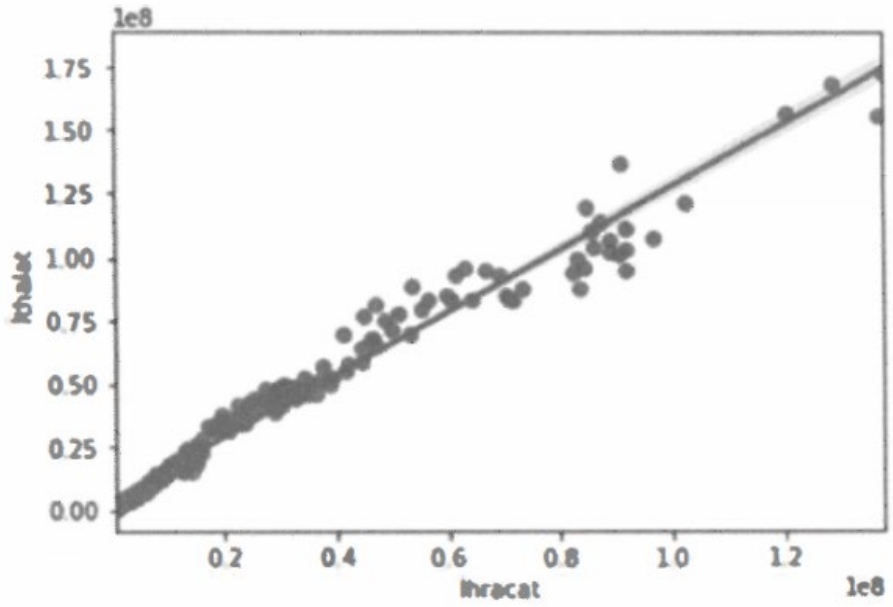
```
In [1]: import pandas as pd
        import seaborn as sns

In [2]: df = pd.read_excel("veri.xlsx", "Sayfa1")
        df.head()
```

Out[2]:

	Aylar	İhracat	İthalat
0	1998-1	463496.833467	6.508007e+05
1	1998-2	460775.662337	8.587170e+05
2	1998-3	582218.796419	1.016087e+06
3	1998-4	468849.576997	8.849401e+05
4	1998-5	608241.467409	1.043346e+06

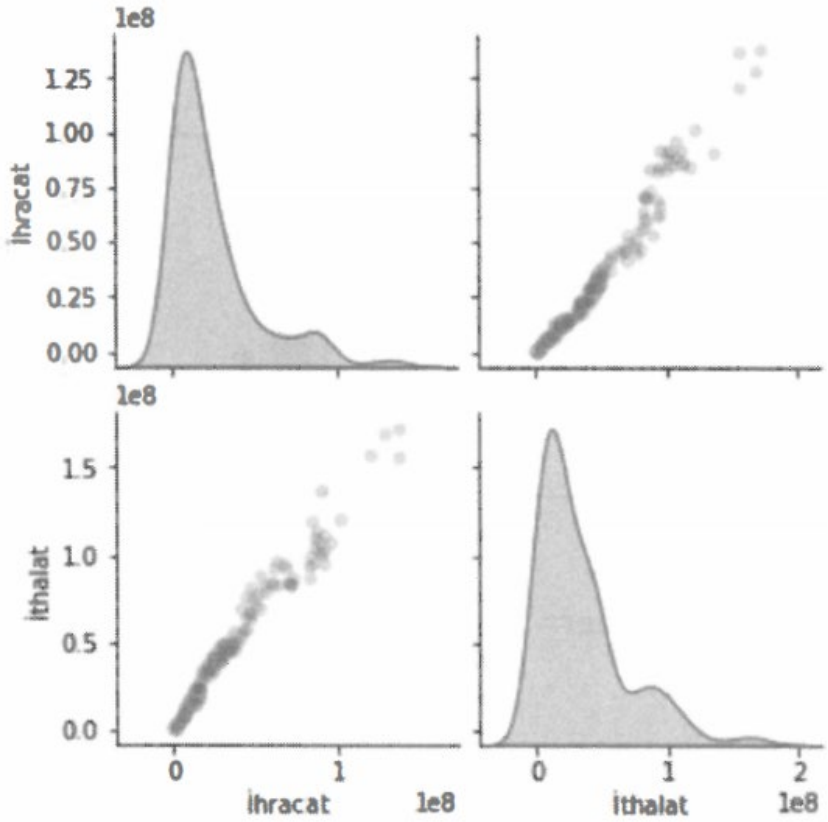
Verilerin okutulması sonrası Seaborn kütüphanesinden ".regplot" modülü kullanılarak regresyon grafiği ve nokta dağılımları elde edilmiştir.

```
In [3]: sns.regplot('İhracat', 'İthalat', data=df)

Out[3]: <AxesSubplot:xlabel='İhracat', ylabel='İthalat'>
```



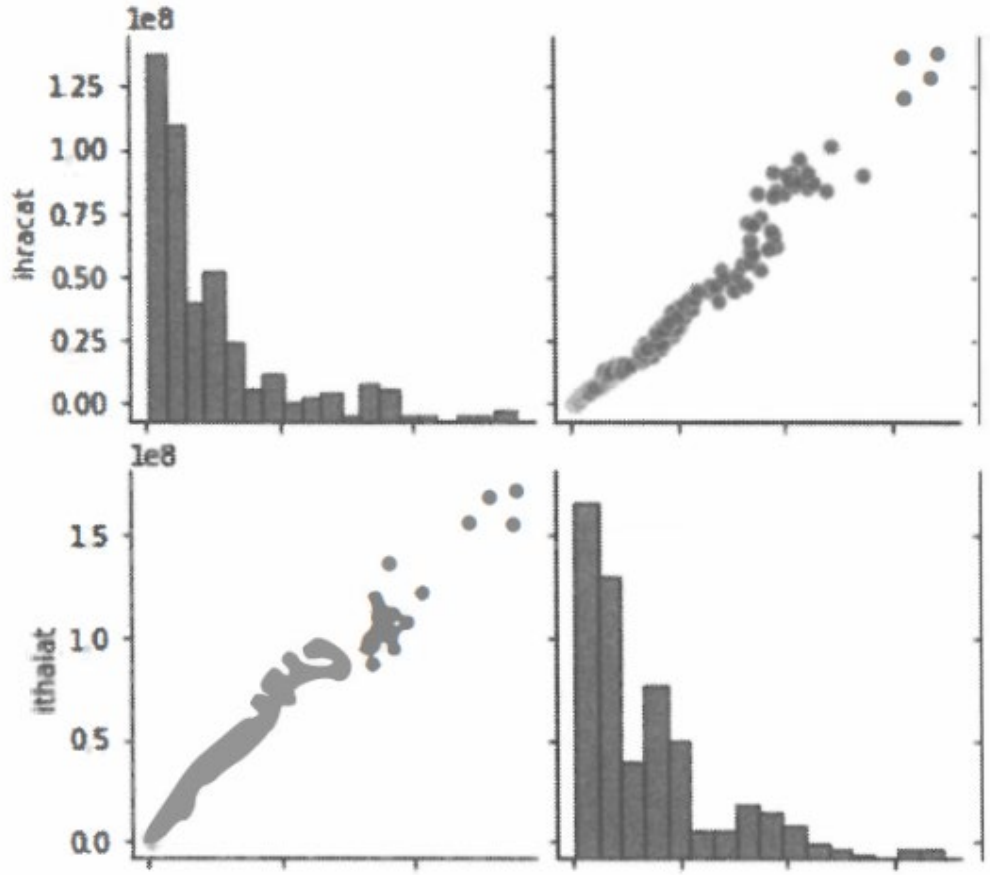
Keşifse! veri analizinde, bir grup değişken arasındaki tüm dağılım grafiklerine bakmak faydalı olabilir. İkili ya da daha fazla veri için çiftler grafiği veya dağılım grafiği matrisi olarak bilinen grafikler çizilebilir. Seaborn kütüphanesi sayesinde köşegenlerde histogram ve veriler arasındaki korelasyonu gösteren grafikler elde edilebilir.

```
In [4]: sns.pairplot(df, diag_kind='kde', plot_kws={'alpha': 0.2})
Out[4]: <seaborn.axisgrid.PairGrid at 0x1ab260ddf10>
```



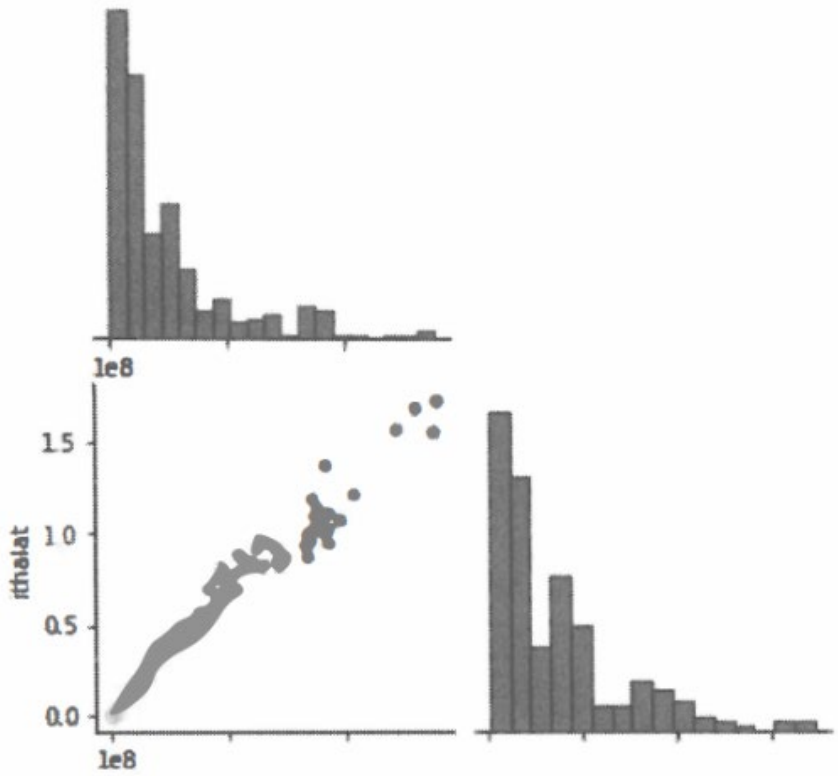
Köşegen elemanları "diag_kind= "hist" şeklinde girilerek histogram olarak çizilebilir.

```
In [6]: sns.pairplot(df, diag_kind="hist", plot_kws={"alp
Out[6]: <seaborn.axisgrid.PairGrid at 0x23af3f52370>
```



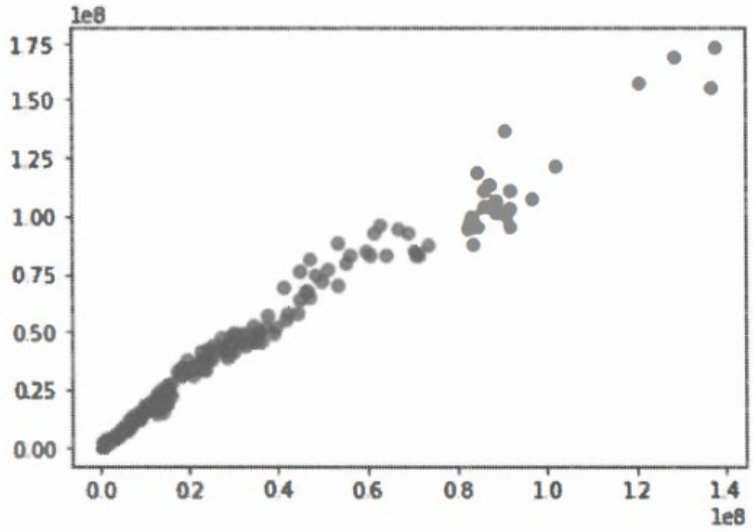
Ayrıca modül içerisine "corner=True'' eklenirse simetrik tablo için aynı olan üst kısmını atar.

```
In [8]: sns.pairplot(df, corner=True )
Out[8]: <seaborn.axisgrid.PairGrid at 0x23af6b71790>
```



Matplotlib.pyplot kütüphanesi kullanarak dağılım grafiği ithalat ve ihracat için çizilebilir.

```
In [11]: import matplotlib.pyplot as plt #matplotlib kütüphanesi

In [12]: plt.scatter(x=df.ihracat, y=df.ithalat, color="brown", alpha=0.5)
         plt.show()
```



Örnek: Matplotlib. pyplot kütüphanesi kullanarak ülkelerin sağlık gösterge verilerine göre dağılım grafiği için öncelikle veriler ve kütüphaneler içe aktarılarak başlanmıştır.

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt

In [2]: veri= pd.read_excel('C:/Users/ACER/saglik.xlsx', "Sayfa2")
        veri.head()
```

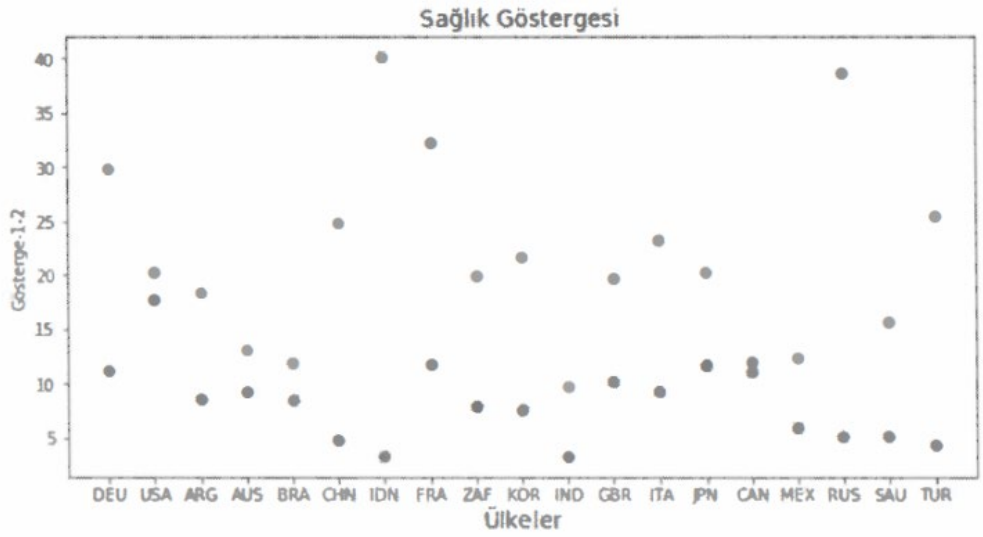
Out[2]:

	Ülke	G1	G2	G3	G4	G5	G6	G7	G8	G9	...	G21	G22
0	DEU	29.96	11.33	8.64	100.00	99.22	87.0	0.1	3.7	2.2	...	2.5	11.500
1	USA	20.35	17.78	9.02	99.27	99.97	91.0	0.4	6.5	3.5	...	2.5	8.600
2	ARG	18.42	8.58	6.70	99.83	96.24	86.0	0.4	9.9	6.4	...	4.6	7.609
3	AUS	13.19	9.34	6.38	100.00	99.99	95.0	0.1	3.7	2.3	...	2.5	6.300
4	BRA	12.00	8.62	3.78	98.38	89.31	83.0	0.5	14.4	8.1	...	2.5	6.452

5 rows × 31 columns

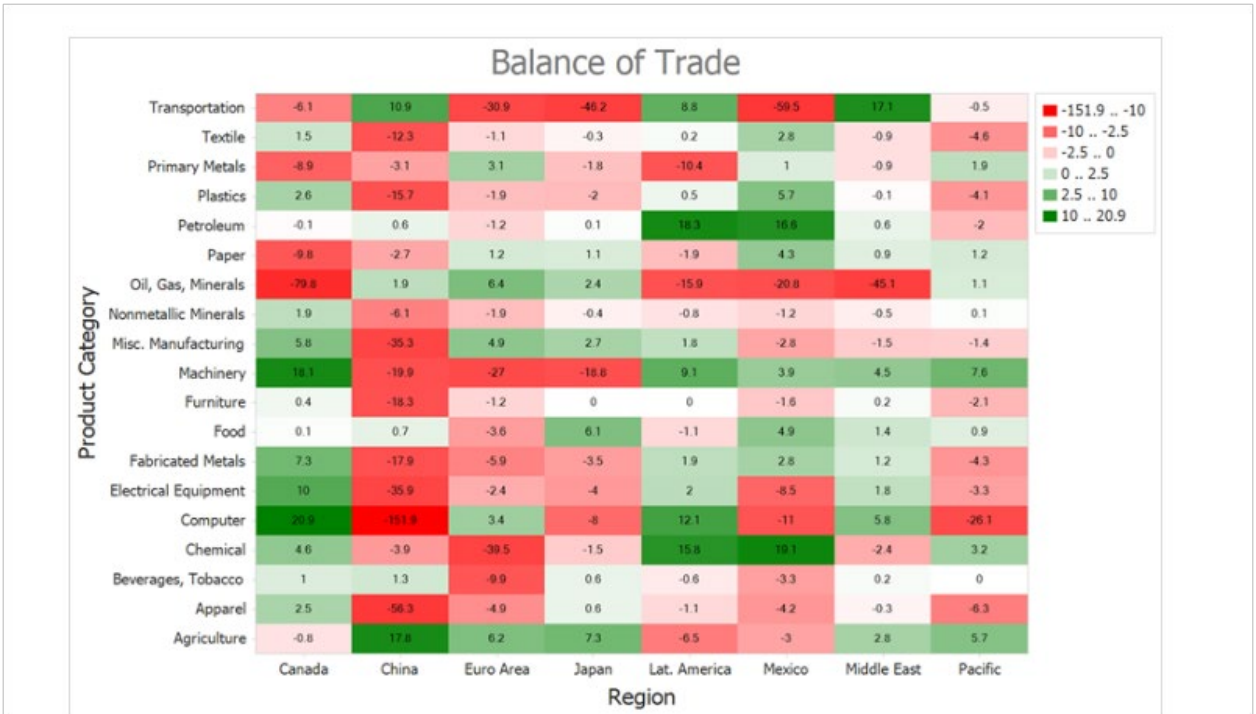
Elde edilen veriler için birinci ve ikinci gösterge kullanılarak ülkelerin nokta dağılım grafiği çizilmiştir.

```
In [5]: plt.scatter(x=veri.Ülke, y=veri.G1,color="red")
        plt.scatter(x=veri.Ülke, y=veri.G2, color="blue")
        plt.title('Sağlık Göstergesi',fontsize=14)
        plt.xlabel('Ülkeler',fontsize=14)
        plt.ylabel('Gösterge-1-2')
        plt.rcParams['figure.figsize'] = (10,5) #Grafiğin boyutunu ayarlar
        plt.show()
```



"plt.scatter()" modülü iki kez kullanılarak grafiklerin tek bir görsel üzerinde gösterimi sağlanmıştır. Başlık ve eksenlerin isimleri "label" modülleri ile değiştirilip "fontsize" modülü ile boyutları ayarlanabilmektedir.

Isı Haritaları()



Isı haritaları, kovaryans ve korelasyon matrislerini göstermek için kullanılır. Bir ısı haritası, renk kodlu bir matris kullanarak birden çok değişken arasındaki korelasyonu gösterir. Burada renk doygunluğunun değişimi değişkenler arasındaki korelasyonun gücünü temsil etmektedir. Aynı zamanda ısı haritası, birden çok değişkenin aynı anda görselleştirilmesine yardımcı olabilir.

.**Örnek**: Endeks değerlerinin bulunduğu dosyanın okutulması ve endeksler arası korelasyon değeri "correlation= df.corr(method='pearson')" modülü ile hesaplanmıştır.

```
In [1]: import pandas as pd
        import seaborn as sns

In [2]: df = pd.read_csv("tam.csv")
        df.head()
```

Out[2]:

	Tarih	Bist-100	Bovespa	İtaly40	KOSPI	Nikkei 225	BMVIIPC	Shanghai	Tadawul
0	2010-01-04	4.727282	4.845378	3.362313	3.229462	4.027545	4.515324	3.511049	3.792515
1	2010-01-05	4.733318	4.846583	3.362520	3.228046	4.028646	4.514983	3.516162	3.795122
2	2010-01-06	4.736774	4.849600	3.363744	3.231806	4.030658	4.516273	3.512447	3.796637
3	2010-01-07	4.740149	4.847888	3.365338	3.226200	4.028639	4.519363	3.504169	3.797341
4	2010-01-08	4.738764	4.846725	3.367207	3.229236	4.033356	4.517091	3.504607	3.798044

```
In [3]: correlation = df.corr(method='pearson')
        correlation
```

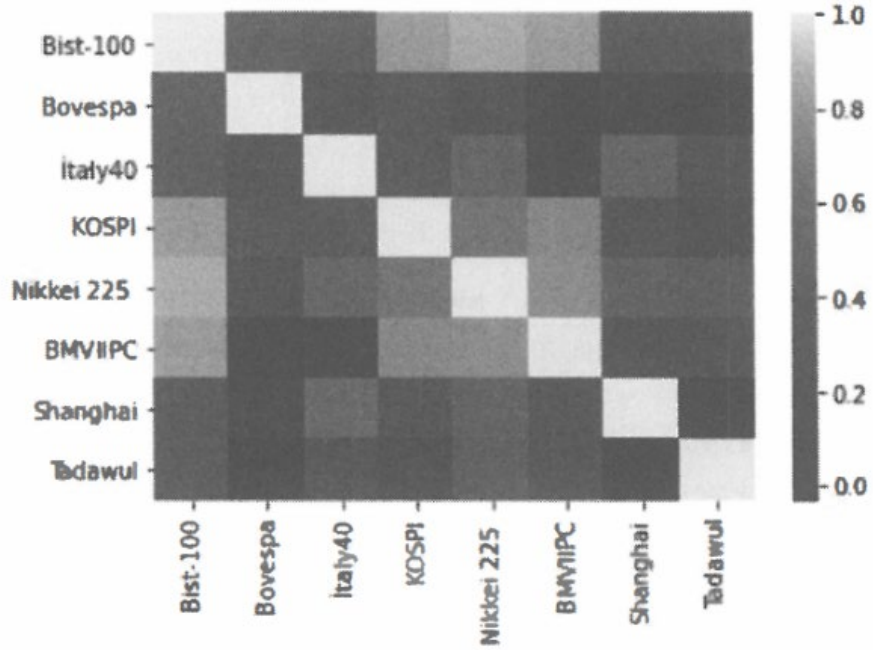
Out[3]:

	Bist-100	Bovespa	İtaly40	KOSPI	Nikkei 225	BMVIIPC	Shanghai	Tadawul
Bist-100	1.000000	0.489413	0.459935	0.809448	0.865206	0.817538	0.404127	0.413048
Bovespa	0.489413	1.000000	0.359136	0.423164	0.339150	0.113983	0.214098	0.088523
İtaly40	0.459935	0.359136	1.000000	0.420827	0.565419	0.194820	0.586554	0.386370
KOSPI	0.809448	0.423164	0.420827	1.000000	0.664751	0.759247	0.388197	0.277318
Nikkei 225	0.865206	0.339150	0.565419	0.664751	1.000000	0.778770	0.530922	0.498204
BMVIIPC	0.817538	0.113983	0.194820	0.759247	0.778770	1.000000	0.328778	0.356698
Shanghai	0.404127	0.214098	0.586554	0.388197	0.530922	0.328778	1.000000	-0.032318
Tadawul	0.413048	0.088523	0.386370	0.277318	0.498204	0.356698	-0.032318	1.000000

Korelasyon değerleri sonrası ısı haritaları yardımıyla endeksler arası ilişkilerin görsel çıktısı elde edilmiştir.

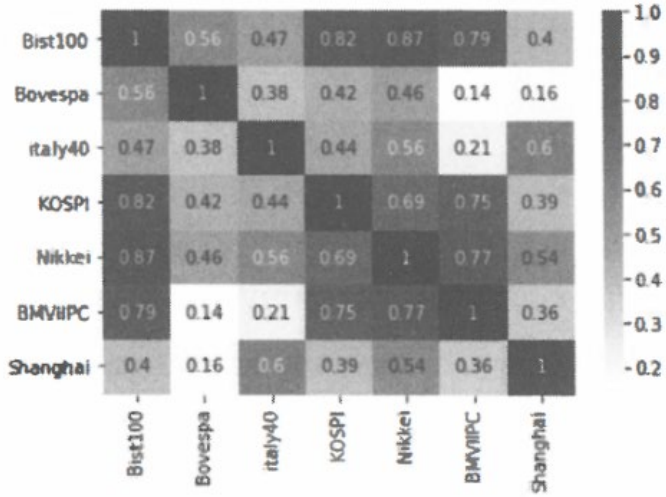
```
In [4]: sns.heatmap(correlation,xticklabels=correlation.columns,
        yticklabels=correlation.columns)
```

```
Out[4]: <AxesSubplot:>
```



Modül içerisinde renk ve değerlerin yazılması ayarları değiştirilebilir.

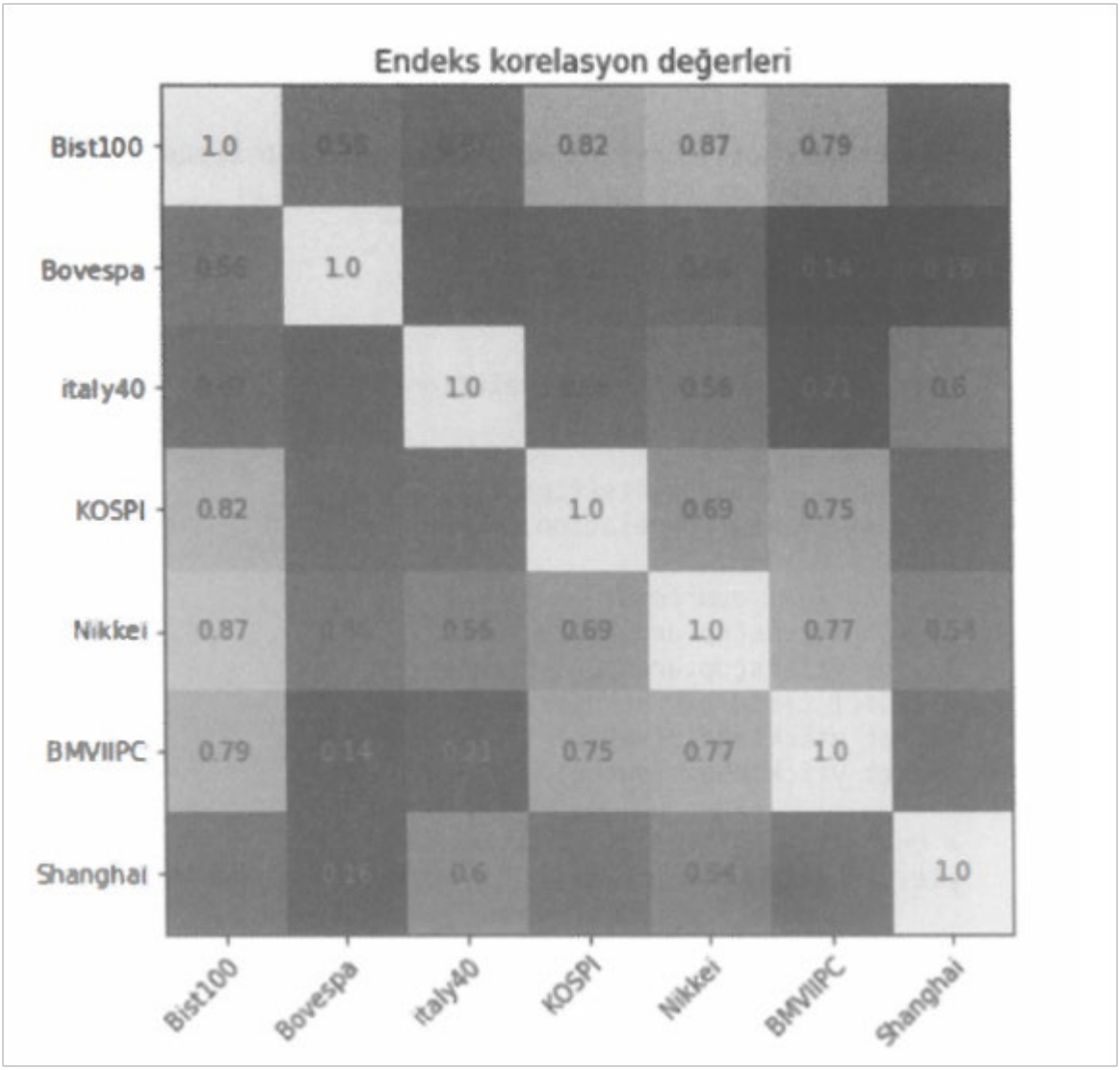
```
In [7]: ax = sns.heatmap(correlation,xticklabels=correlation.columns,
                        yticklabels=correlation.columns, annot=True, cmap="YlGnBu")
```



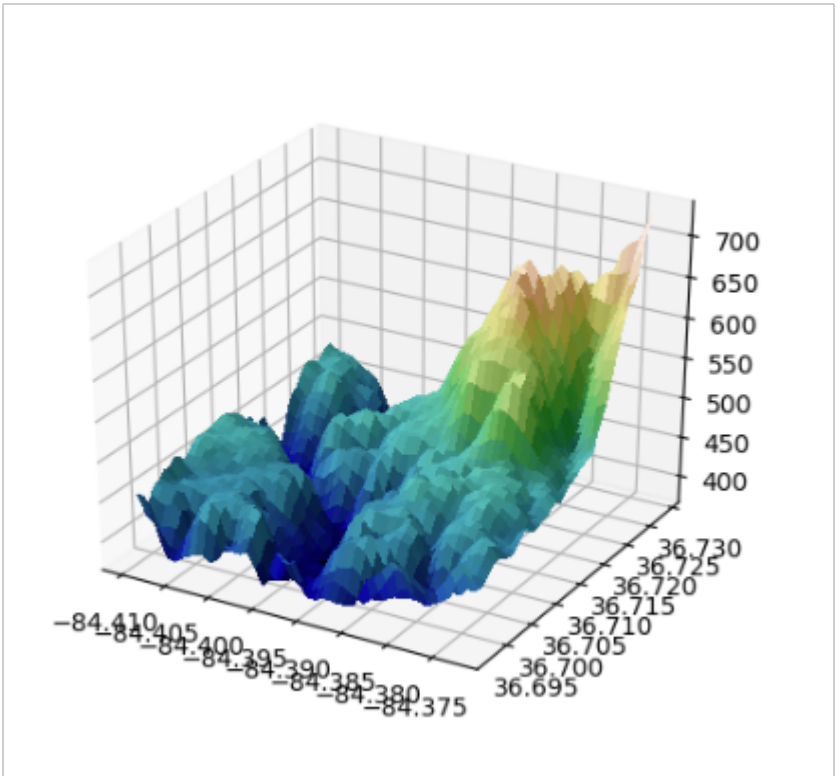
Görsel çıktının yanında yer alan renk konstrat değişimine dikkat edilirse açık renkler güçlü ilişkiyi gösterirken koyu renkler zayıf ilişkiyi temsil etmektedir.

In [7]:
```
import matplotlib.pyplot as plt #matplotlib kütüphanesi
import numpy as np
```

In [8]:
```
satır = df.columns
sutun = df.columns

correlation = np.array(correlation)


fig, ax = plt.subplots(figsize=(8, 6))
im = ax.imshow(correlation,cmap = 'Greens')

# Etiketleri ayarlama
ax.set_xticks(np.arange(len(satır)))
ax.set_yticks(np.arange(len(sutun)))
# ilgili liste girişlerini etiketleme
ax.set_xticklabels(satır)
ax.set_yticklabels(sutun)

# Onay etiketlerini döndürme ve hizalamalar.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
         rotation_mode="anchor")

# For döngüsünü kullanarak metin ek açıklamaları oluşturma
for i in range(len(sutun)):
    for j in range(len(satır)):
        text = ax.text(j, i, correlation[i, j],
                       ha="center", va="center", color="r")

ax.set_title("Endeks korelasyon değerleri")
fig.tight_layout()
plt.show()
```


Endeks korelasyon değerleri
Bist100
Bovespa
italy40
KOSPI
Nikkei
BMVIIPC
Shanghai
1.0
0.82
0.87
0.79
1.0
1.0
0.56
0.69
0.75
1.0
0.77
1.0
1.0

Üç Boyutlu Grafikler ()



İki boyutlu grafikler, veri görselleştirmenin temelini oluşturmasına rağmen üç girdi verisi için yeterli değildir. Bundan dolayı üç boyutlu grafikler bu durumu karşılar ve görsellik açısından katkıda bulunur. Üç boyutlu grafiklerin çizimi için "matplotlib.pyplot" kütüphanesi ve "mpl_toolkits" kütüphanesinden "mplot3d" modülü içe aktarılmıştır.

Örnek: Uygulamada öncelikle sera gazı değerlerinin yıllara göre salımını gösteren verinin aktarılması ve grafiğin boyutlarında kullanılacak değerlerin tanımlaması yapılmıştır.

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from mpl_toolkits import mplot3d
        import pandas as pd

In [2]: veri= pd.read_excel('C:/Users/ACER/veri.xlsx', "Sayfa3")
        veri.head()
```

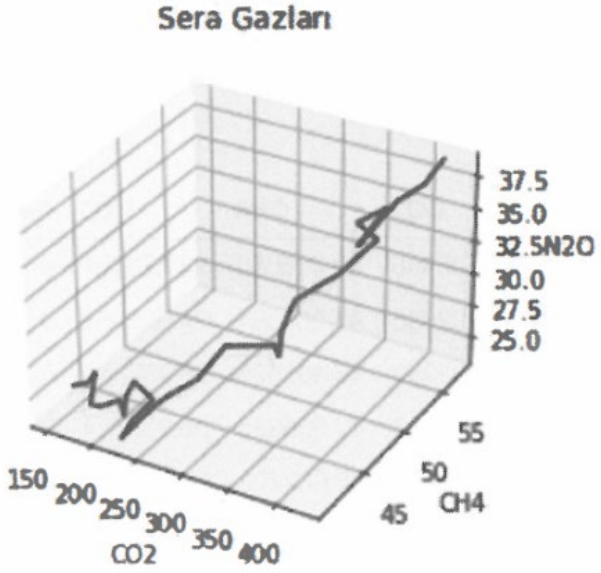
Out[2]:

	Yıl	Toplam	CO2	CH4	N2O	F-gazları
0	1990	219.367813	151.508468	42.405060	24.828987	0.625298
1	1991	226.756072	157.982005	43.285364	24.625361	0.863343
2	1992	232.975376	163.922250	43.193125	25.137414	0.722588
3	1993	240.316701	171.011163	42.965635	25.936824	0.403080
4	1994	234.287667	167.433113	42.683702	23.460852	0.709999

```
In [3]: X=veri.CO2.values
        Y=veri.CH4.values
        Z=veri.N2O.values
```

3 boyutlu çizgi grafiği için "ax.plot3D(X, Y, Z, <renk')" modülü ile veriler boyutlar için tanımlanır ve grafiğin rengi burada belirlenir. Boyutların isimlendirilmesinde "ax.set_xlabel(' ')" modülü her bir boyut için ve "ax.set_title(" ")" modülü ise de grafik başlığı için kullanılır. Uygulamada "ax = plt.axes(projection='3d')" modülü grafiğin çizilmesini sağlar.

```
In [4]: ax = plt.axes(projection='3d')
        ax.set_xlabel('CO2')
        ax.set_ylabel('CH4')
        ax.set_zlabel('N2O')
        ax.set_title("Sera Gazları")
        ax.plot3D(X, Y, Z, 'blue')

Out[4]: [<mpl_toolkits.mplot3d.art3d.Line3D at 0x245f9d41820>]
```



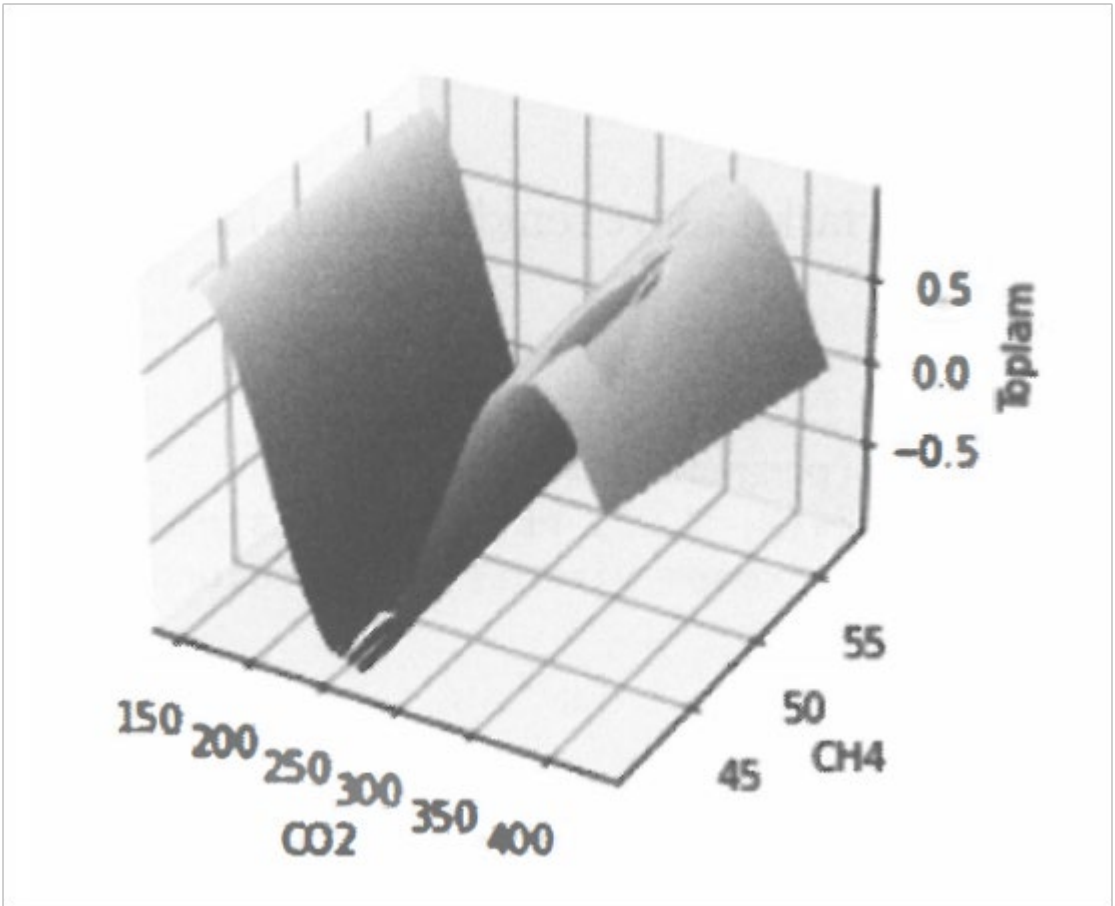
Üç boyutlu yüzey grafiği için "Z=np.sin(np.sqrt(x + y))" olacak şekilde x ve y değerlerine bağlı bir fonksiyon tanımlanmıştır. Burada ayrıca grafiğin elde edilebilmesi için "np.meshgrid(x, y)" modülü ile koordinat vektörleri koordinat matrislerine çevrilir. Grafiğin çizilmesinde renklendirme aşamasında "cmap" modülü için aşağıdaki girdilerden biri kullanılabilir.

cmaps['Perceptually Uniform Sequential'] = ['viridis', 'plasmi, 'inferno', 'magma, 'cividis']

cmaps['Sequential'] = ['Greys', 'Purples', 'Blues', 'Greens', 'Oranges', 'Reds', 'YlOrBr', 'YlOrRd', 'OrRd', 'PuRd', 'RdPu', 'BuPu', 'GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn', 'YlGn']

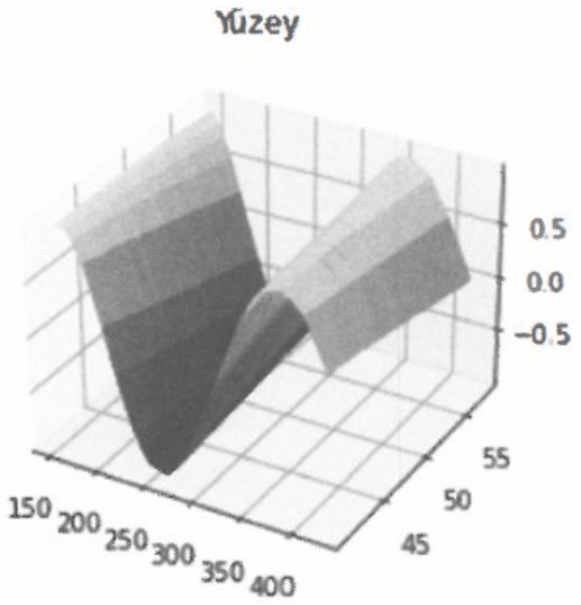
In [5]:
```
def f(x, y):
    return np.sin(np.sqrt(x + y))

x = veri.CO2.values
y = veri.CH4.values

X, Y = np.meshgrid(x, y)
Z = f(X, Y)
```

In [6]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 100,cmap='plasma')
ax.set_xlabel('CO2')
ax.set_ylabel('CH4')
ax.set_zlabel('Toplam');
```



Uygulamanın devamında farklı açı ve renkler kullanılarak çizilen yüzey grafiği çizimleri gösterilmiştir.

```
In [9]: ax = plt.axes(projection='3d')
        ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
                        cmap='viridis', edgecolor='none')
        ax.set_title('Yüzey');
```



Üç boyutlu grafik çiziminde kullanılan diğer bir yöntemde tel kafes şeklinde çizimdir. "ax.plot_wireframe(X, Y, Z, color='renk')" modülü yardımıyla girdi değişkenleri ve rengi belirlenebilir.

In [8]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_wireframe(X, Y, Z, color='purple')
ax.set_title('tel kafes');
```