Matplotlib Grafik Türleri

Görselleştirmenin başarısı grafik türünün seçimine ve doğru kullanımına bağlıdır. Örnekler:

Çizgi grafiği, belirli bir zaman aralığındaki sayısal değişimleri ve bu zaman içindeki eylemleri göstermek için kullanılır.

Balon grafikleri, kategorilere ayrılmış verilerin ilişkilerini göstermek ve kıyaslamak için kullanılır.

Dağılım grafiğinde iki değişkenin değerlerini görüntülemek için koordinat sistemine yerleştirilen noktalar topluluğu kullanılır.

Histogramlar, değerlerin nerede yoğunlaştığını, maksimum ve minimum noktalarını, veriler arasındaki kopmalar ve olağandışı değerler olup olmadığı hakkında tahminde bulunmamıza yardımcı olur.

Pasta grafikleri, bir daireye orantılı bölümlere bölerek kategoriler arasında oranlar ve yüzdeler göstermeye yardımcı olur.

Kütüphaneleri ekleme ve veri setimizi okuma:

Data info() fonksiyonu verilerin içeriği hakkında bilgi veriyor.

```
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv("iris.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
data
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Id sütununu silme:

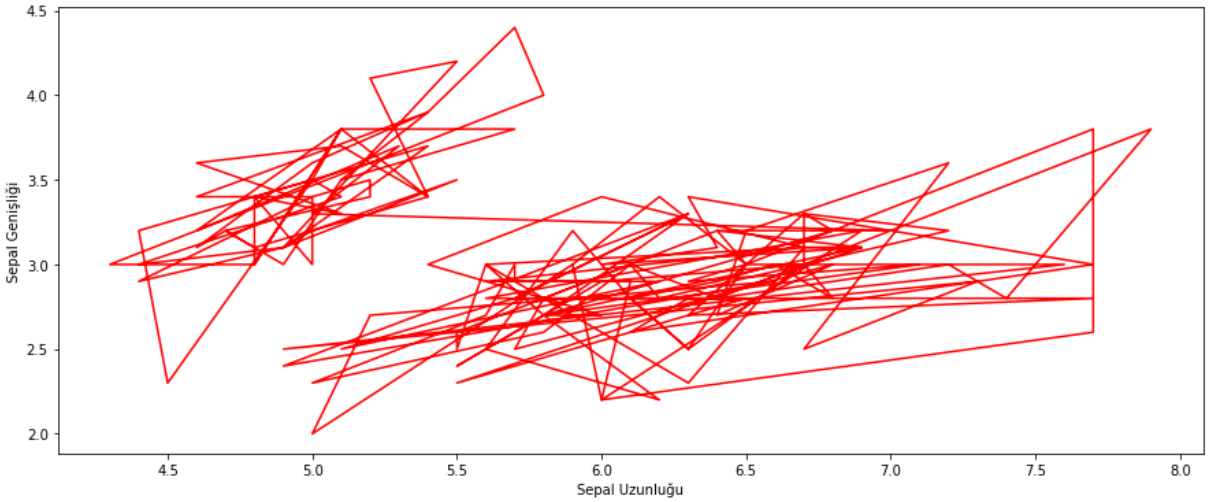
```
data.drop('Id', axis=1)
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Çizgi grafiği ile verinin sepal ölçülerini karılaştırma:

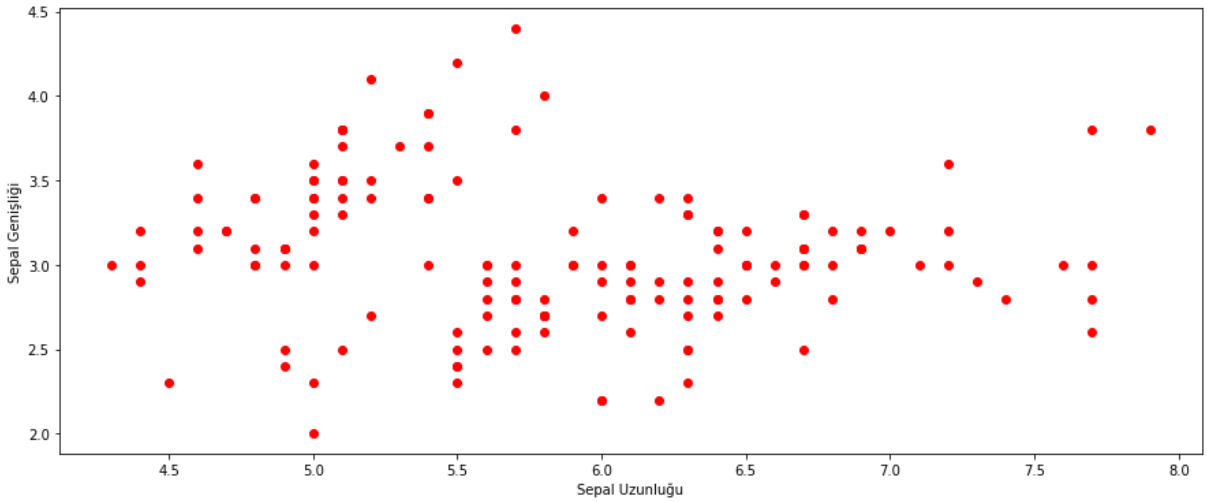
```
plt.figure(figsize=(15,6))

plt.plot(data.SepalLengthCm, data.SepalWidthCm, color="r")

plt.xlabel("Sepal Uzunluğu")
plt.ylabel("Sepal Genişliği")

plt.show()
```



Bu grafikten bir anlam çıkaramayız. Çünkü çizgi grafiğini bir zaman aralığındaki sayısal değişimleri göstermek için kullanılıyor. Veride zaman adında bir özellik yok. Onun yerine setosa, versicolor ve virginica olmak üzere kategoriler var. Kategorileri karşılaştırmak içinde dağılım grafiğini kullanıyoruz.

Plot yerine *Scatter* yazdık. Scatter'ın anlamı dağılımdır. Her bir sample'ın değerini nokta olarak gösterir.

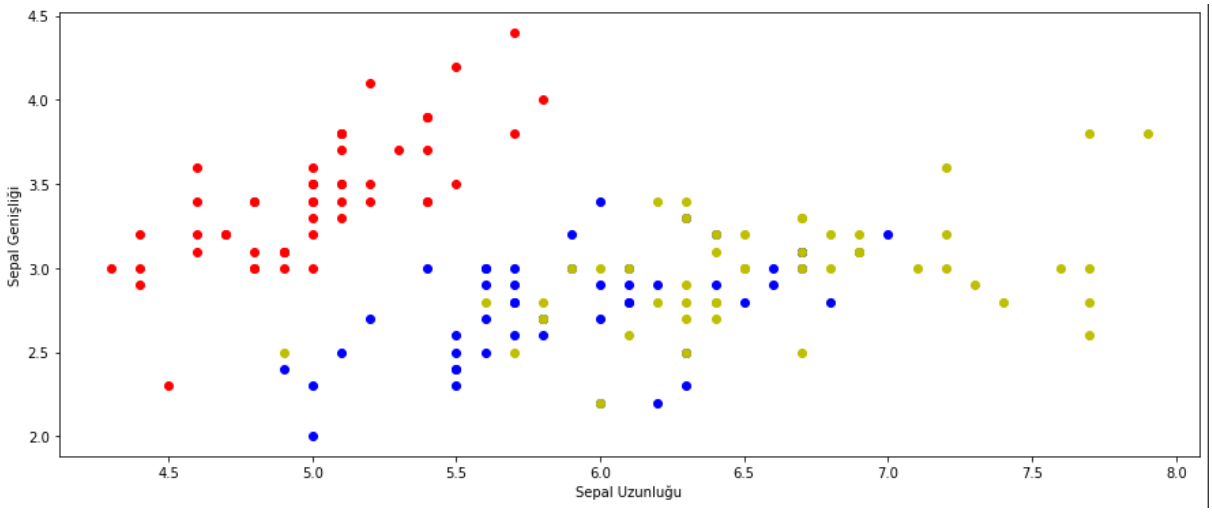
```
plt.figure(figsize=(15,6))

plt.scatter(data.SepalLengthCm, data.SepalWidthCm, color="r")

plt.xlabel("Sepal Uzunluğu")
plt.ylabel("Sepal Genişliği")

plt.show()
```



Kategorilerin renklerini farklı gösterme:

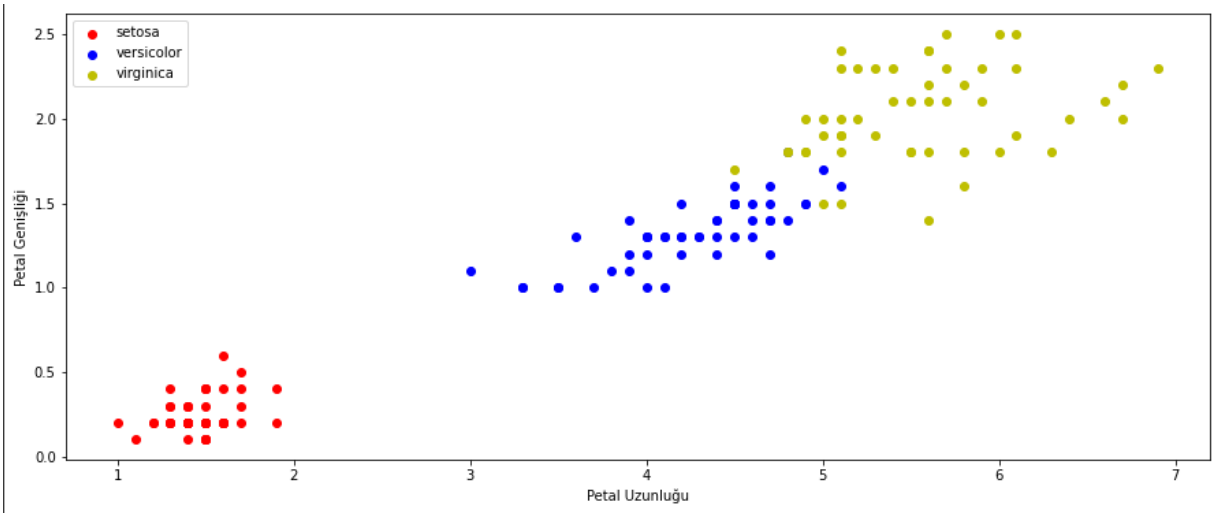
```
[9]  setosa = data[data['Species'] == 'Iris-setosa']
     versicolor = data[data['Species'] == 'Iris-versicolor']
     virginica = data[data['Species'] == 'Iris-virginica']
```

```
plt.figure(figsize=(15,6))

plt.scatter(setosa.SepalLengthCm, setosa.SepalWidthCm, color = "r")
plt.scatter(versicolor.SepalLengthCm, versicolor.SepalWidthCm, color = "b")
plt.scatter(virginica.SepalLengthCm, virginica.SepalWidthCm, color = "y")

plt.xlabel("Sepal Uzunluğu")
plt.ylabel("Sepal Genişliği")

plt.show()
```



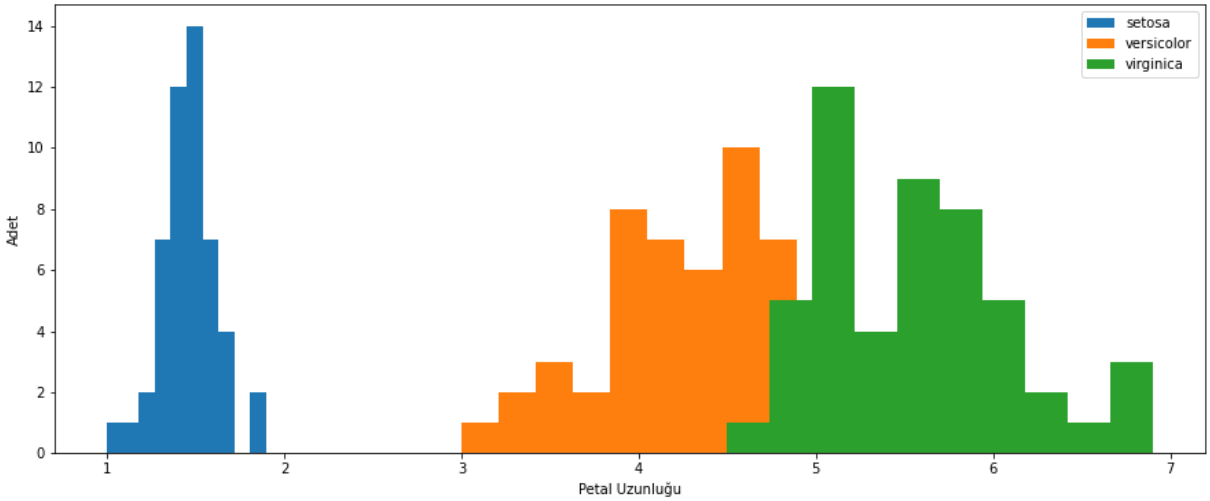
Sepal ölçüleri yerine petal ölçüleri:

```
plt.figure(figsize=(15,6))

plt.scatter(setosa.PetalLengthCm,
            setosa.PetalWidthCm,
            color="r",
            label = "setosa" )
plt.scatter(versicolor.PetalLengthCm,
            versicolor.PetalWidthCm,
            color="b",
            label = "versicolor" )
plt.scatter(virginica.PetalLengthCm,
            virginica.PetalWidthCm,
            color="y",
            label = "virginica" )

plt.xlabel("Petal Uzunluğu")
plt.ylabel("Petal Genişliği")

plt.legend()

plt.show()
```



Histogram grafiği: Sampleların maksimum ve minimum değerini görmemize yardımcı oluyor. Hist yazıyoruz.

```
plt.figure(figsize=(15,6))

plt.hist(setosa.PetalLengthCm, label="setosa")
plt.hist(versicolor.PetalLengthCm, label="versicolor")
plt.hist(virginica.PetalLengthCm, label="virginica")

plt.xlabel("Petal Uzunluğu")
plt.ylabel("Adet")

plt.legend()

plt.show()
```



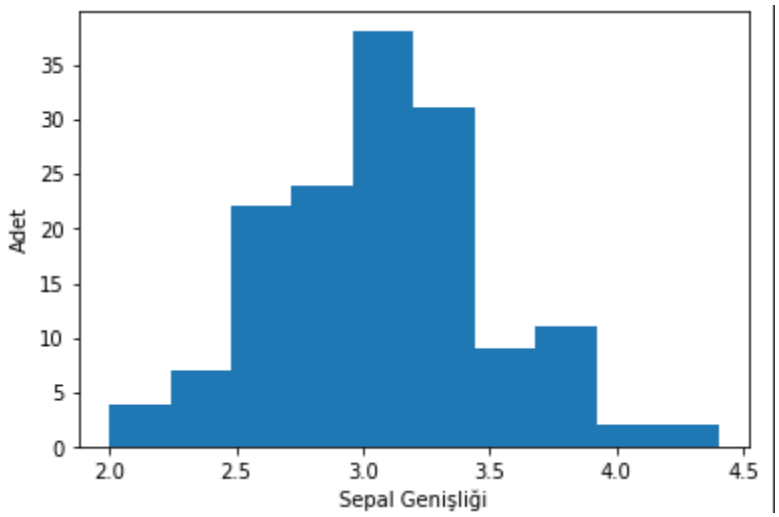
Grafiği incelediğimizde setosa türü ve versicolor arasında bir kopma olmuş. Türler arası geçiş histogram grafiğinde de belli oluyor. Bu grafikte bir uzunlukta kaç tane sample olduğu da gözüküyor.

Yukarıdaki kodun farklı versiyonu:

```
plt.figure(figsize=(15,6))

setosa.PetalLengthCm.plot.hist(label="setosa")
versicolor.PetalLengthCm.plot.hist(label="versicolor")
virginica.PetalLengthCm.plot.hist(label="virginica")

plt.xlabel("Petal Uzunluğu")
plt.ylabel("Adet")

plt.legend()

plt.show()
```

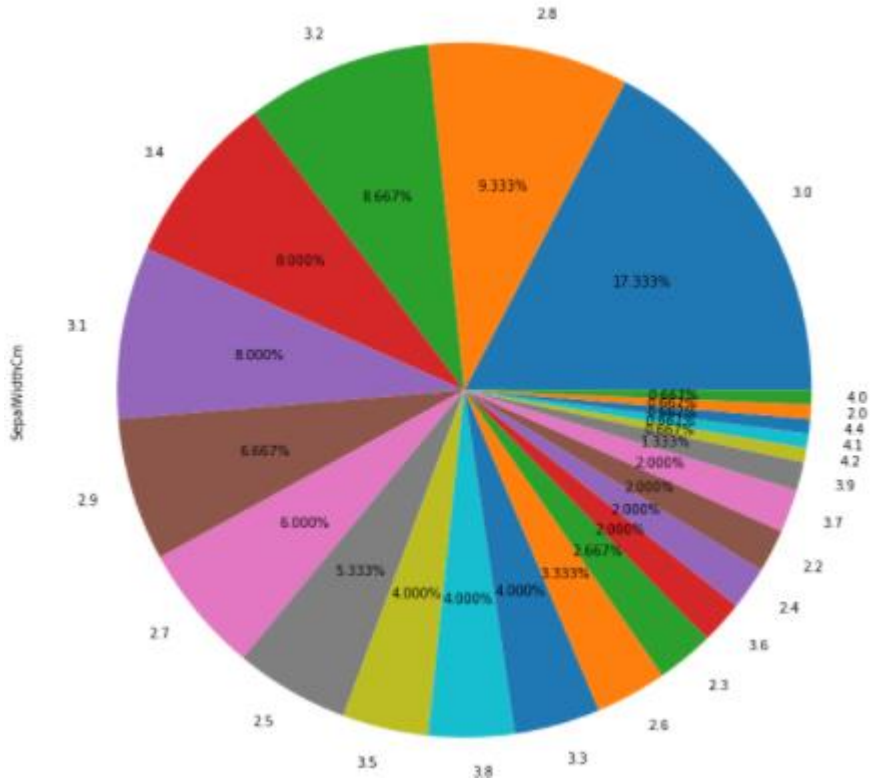
Sadece bütün sampleların sepal genişliğini histogram olarak gösterme:

```
data.SepalWidthCm.plot.hist()

plt.xlabel("Sepal Genişliği")
plt.ylabel("Adet")

plt.show()
```



Değerleri yüzdelik olarak görmek için pasta grafiği:

```
data.SepalWidthCm.value_counts().plot.pie(autopct= '%1.3f%%', figsize=(12,12))
plt.show()
```



value_counts() : Aynı genişliğe sahip sampleları bir dilimde gösterir.

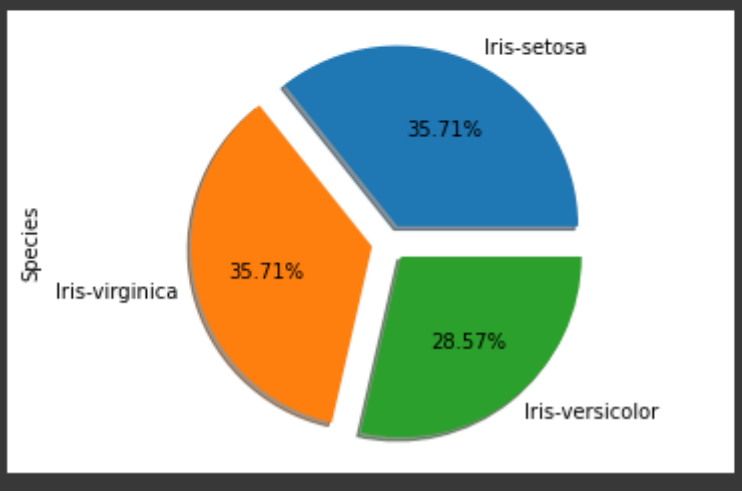
Genişliklerin yüzde olarak ne kadar bir payda sahip olduklarını görmek için de autopct yazıp % işaretlerinin arasına 1.1f ve yeniden % işaretini yazıyoruz. 1.1f'nin anlamı noktadan sonra bir rakam göstermektir.

Explode daireden dilim ayırmamızı sağlıyor. 3 tane kategori olduğu için 3 tane değer girdik. Dairenin merkezinden 0.1 uzunluğundan uzaklaş dedik.

Shadow ile gölge eklendi.

Axis() fonksiyonu ile düzgün bir daire grafiği oluşturuluyor.

```
random_data.Species.value_counts().plot.pie(autopct = '%1.2f%%',
                                            explode = [0.1, 0.1, 0.1],
                                            shadow = True)

plt.axis("equal")

plt.show()
```



KAYNAKÇA

Bilgeiş "Herkes için Yapay Zekâ I" eğitimi.

KODLUYORUZ
geleceği kodluyoruz >_

EMpower
Enriching young lives in emerging markets