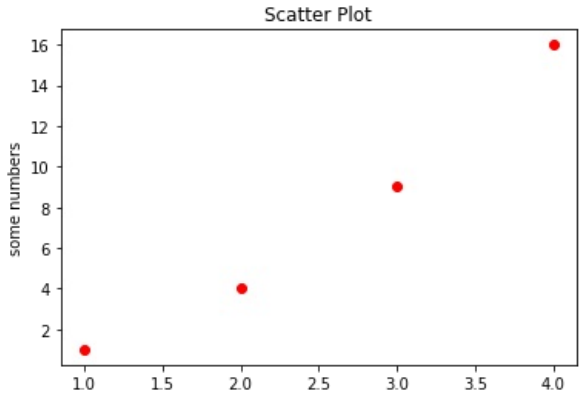
3 LINE PLOT- ÇİZGİ GRAFİĞİ

Tanım:

Çizgi grafiği (Line Plot) bir veya birden fazla değişkenin değişimi gösterir. Bu grafikte herbir veri noktası dağılım grafiğindeki noktaların çizgi ile biribirine bağlanmış hali gibidir.

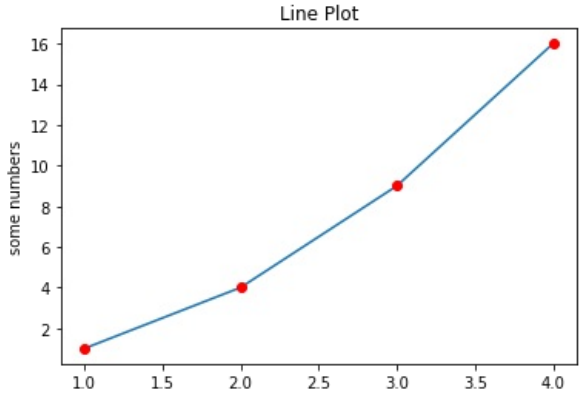
In [9]:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 9, 16],'ro')
plt.ylabel('some numbers')
plt.title("Scatter Plot")
plt.show()
```



In [10]:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.plot([1, 2, 3, 4], [1, 4, 9, 16],'ro')
plt.ylabel('some numbers')
plt.title("Line Plot")
plt.show()
```



In [ ]:

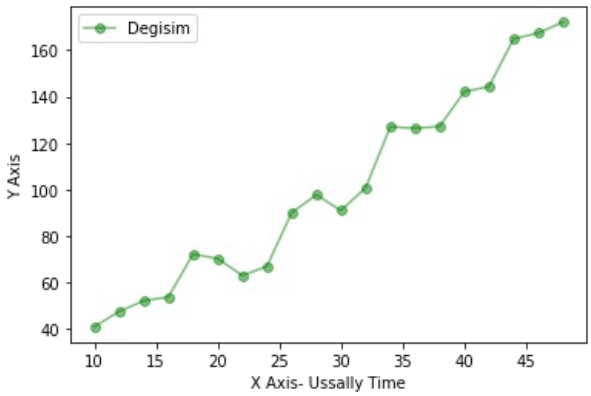
```

```

Çizgi grafiği genellikle bir değişkenin belirli bir zaman aralığında değişiminin eğilimini görselleştirmek amacıyla kullanılır. bu nedenle genellikle X ekseni zaman ekseni olarak kullanılır. X ekseni belirtilen zaman aralığını kullandığı için sıfırdan başlamak zorunda değildir ama her zaman için soldan sağa doğru artan bir zamanı belirtmelidir.

Örneğin bir varlığın zamana bağlı fiyatının değişimi grafiği gibi (dolar, euro, altın, btc, elektrik tüketim-üretim, rüzgar vs vs)

In [17]:

```
import matplotlib.pyplot as plt
import numpy as np

# Fixing random state for reproducibility
np.random.seed(19680801)


x = np.arange(10, 50.0, 2.0)
y = x ** 1.3 + np.random.rand(*x.shape) * 30.0
s = np.random.rand(*x.shape) * 800 + 500

plt.plot(x, y, c="g", alpha=0.5, marker=r'o',
            label="Degisim")


#plt.plot(x, y, s, c="g", alpha=0.5, marker=r'o')

plt.xlabel("X Axis- Ussally Time")
plt.ylabel("Y Axis")
plt.legend(loc='upper left')
plt.show()
```



In [ ]:

Ne için Kullanılır:

Çizgi grafiği bir veya birden fazla değişkenin zamanla değişimini veya eğilimini göstermek amacıyla kullanılır. Ancak çok fazla değişken kullanımı grafiği okunmasını imkansız hale getirebilir ve bu tip grafiğe Spagetti grafiği denilir. Aşağıda kullanım hatalarında örneği verilecektir.

In [ ]:

In [ ]:

In [ ]:

Çeşitleri:

Eğer data noktası sayısı fazla değil ise o noktalar yuvarlak veya benzeri marker ile işaretlenebilir. Böylece grafikte grid izgara sistemi de kullanılıyor ise verinin tam değeri daha anlaşılır hale gelir.

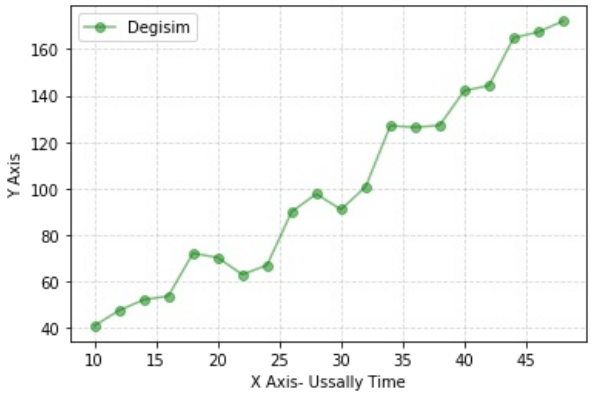
Ayrıca line plot (çizgi grafiği) dağılım grafiklerindeki (scatter plot) trendin gösterimi için de dağılım grafiklerine eklenebilir. Bu tür grafiklere Makine öğrenmesi konusunda sıklıkla rastlayabilirisiniz.

Bu grafik türlerinin alakalı olduğu diğer grafik türleri ise: Alan grafiği - Area chart-, stacket area chart-, streamgraph grafik türleridir ve bu grafikler ilerleyen zamanda anlatılacaktır.

In [ ]:

In [4]:

```
import matplotlib.pyplot as plt
import numpy as np

# Fixing random state for reproducibility
np.random.seed(19680801)


x = np.arange(10, 50.0, 2.0)
y = x ** 1.3 + np.random.rand(*x.shape) * 30.0
s = np.random.rand(*x.shape) * 800 + 500

plt.plot(x, y, c="g", alpha=0.5, marker=r'o',
            label="Degisim")


#plt.plot(x, y, s, c="g", alpha=0.5, marker=r'o')

plt.xlabel("X Axis- Ussally Time")
plt.ylabel("Y Axis")
plt.legend(loc='upper left')
plt.grid(color='gray', linestyle='dashed', alpha=0.3)
plt.show()
```



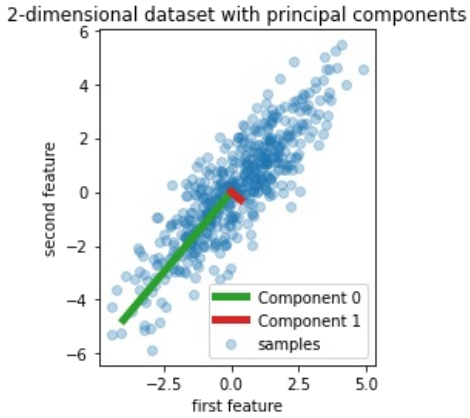
Yukarıdaki grafiğe ızgara eklenmiş ve ızgara tipi, rengi ve şefaflığı ayarlanmıştır.

In [ ]:

In [ ]:

In [9]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

rng = np.random.RandomState(0)
n_samples = 500
cov = [[3, 3], [3, 4]]
X = rng.multivariate_normal(mean=[0, 0], cov=cov, size=n_samples)
pca = PCA(n_components=2).fit(X)


plt.scatter(X[:, 0], X[:, 1], alpha=0.3, label="samples")
for i, (comp, var) in enumerate(zip(pca.components_, pca.explained_variance_)):
    comp = comp * var  # scale component by its variance explanation power
    plt.plot(
        [0, comp[0]],
        [0, comp[1]],
        label=f"Component {i}",
        linewidth=5,
        color=f"C{i + 2}",
    )
plt.gca().set(
    aspect="equal",
    title="2-dimensional dataset with principal components",
    xlabel="first feature",
    ylabel="second feature",
)
plt.legend()
plt.show()
```



Yukarıdaki örnekte -içeriği şu an için önemli değil sadece gösterim amaçlıdır.- bir dağılım grafiğinde çizgi grafiği eklenmiştir.

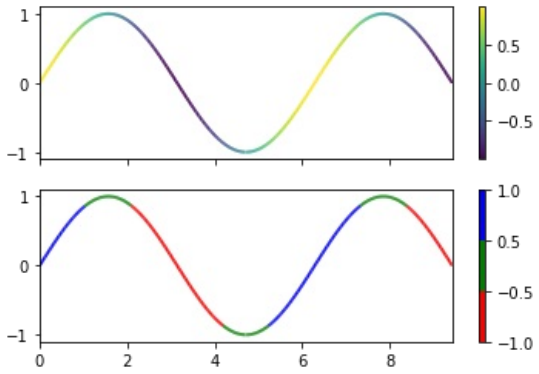
In [ ]:

Çizgi grafikleri kullanıldıkları duruma göre farklı biçimlerde renklendirilebilir. Aşağıda farklı renklendirme kullanılmış iki örnek mevcuttur.

In [ ]:

In [10]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.collections import LineCollection
from matplotlib.colors import ListedColormap, BoundaryNorm

x = np.linspace(0, 3 * np.pi, 500)
y = np.sin(x)
dydx = np.cos(0.5 * (x[:-1] + x[1:]))  # first derivative

# Create a set of line segments so that we can color them individually
# This creates the points as a N x 1 x 2 array so that we can stack points
# together easily to get the segments. The segments array for line collection
# needs to be (numlines) x (points per line) x 2 (for x and y)
points = np.array([x, y]).T.reshape(-1, 1, 2)
segments = np.concatenate([points[:-1], points[1:]], axis=1)

fig, axs = plt.subplots(2, 1, sharex=True, sharey=True)

# Create a continuous norm to map from data points to colors
norm = plt.Normalize(dydx.min(), dydx.max())
lc = LineCollection(segments, cmap='viridis', norm=norm)
# Set the values used for colormapping
lc.set_array(dydx)
lc.set_linewidth(2)
line = axs[0].add_collection(lc)
fig.colorbar(line, ax=axs[0])

# Use a boundary norm instead
cmap = ListedColormap(['r', 'g', 'b'])
norm = BoundaryNorm([-1, -0.5, 0.5, 1], cmap.N)
lc = LineCollection(segments, cmap=cmap, norm=norm)
lc.set_array(dydx)
lc.set_linewidth(2)
line = axs[1].add_collection(lc)
fig.colorbar(line, ax=axs[1])

axs[0].set_xlim(x.min(), x.max())
axs[0].set_ylim(-1.1, 1.1)
plt.show()
```

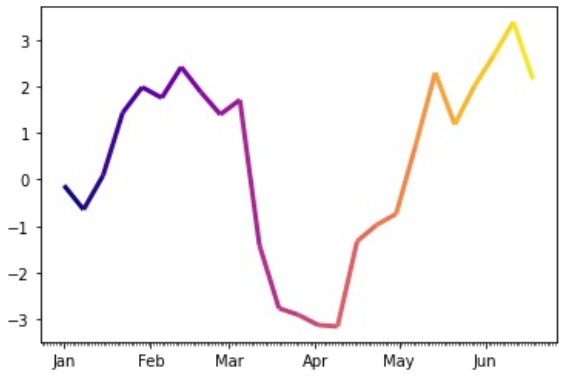


In [ ]:

```
```

In [11]:

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import numpy as np
from matplotlib.collections import LineCollection

dates = pd.date_range("2017-01-01", "2017-06-20", freq="7D" )
y = np.cumsum(np.random.normal(size=len(dates)))

s = pd.Series(y, index=dates)

fig, ax = plt.subplots()

#convert dates to numbers first
inxval = mdates.date2num(s.index.to_pydatetime())
points = np.array([inxval, s.values]).T.reshape(-1,1,2)
segments = np.concatenate([points[:-1],points[1:]], axis=1)

lc = LineCollection(segments, cmap="plasma", linewidth=3)
# set color to date values
lc.set_array(inxval)
# note that you could also set the colors according to y values
# lc.set_array(s.values)
# add collection to axes
ax.add_collection(lc)


ax.xaxis.set_major_locator(mdates.MonthLocator())
ax.xaxis.set_minor_locator(mdates.DayLocator())
monthFmt = mdates.DateFormatter("%b")
ax.xaxis.set_major_formatter(monthFmt)
ax.autoscale_view()
plt.show()
```



In [ ]:

Bu grafik türünden türetilmiş diğer grafik türleri ise: Alan grafiği - Area chart-, stacket area chart-, streamgraph grafik türleridir ve bu grafikler detaylı biçimde ilerleyen zamanda anlatılacaktır.

In [ ]:

In [ ]:

In [ ]:

Kullanım Hataları:

Line - Çizgi grafiklerinde Y ekseninin sıfırdan başlayıp başlamaması genellikle sıklıkla tartışılan bir konudur ancak burada temel amacınızın ne olduğu oldukça önemlidir. Bir birine çok yakın olması nedeniyle ifade edemediğiniz bir grafikte minimum değer sıfır yerine kullanabilirisniz ancak bu durumda grafiği aşırı abartıp karşı tarafı yanıltma olasılığınız orataya çıkar.

In [ ]:

Bazen birbirinden farklı iki değişkenin arasındaki ilişkiyi göstermek amacıyla kullanılan çizgi grafiklerde ilişkili değişkenler biribirinden oldukça farklı değerlerde olabilir. (örneğin satılan araba sayısı ile araba fiyatları gibi) Eğer biribirinden farklı iki değişkeni tek bir X ekseninde değrlendirecekseniz ikincil bir Y ekseni ekleyip (Dual axis) grafiği oluşturabilirsiniz. Ancak budurumda grafiğin okunurluluğunun iyi olabilmesi için renklendirme veya çizginin tipi gibi oldukça ayırıcı bir yol izlemeniz gerekir.

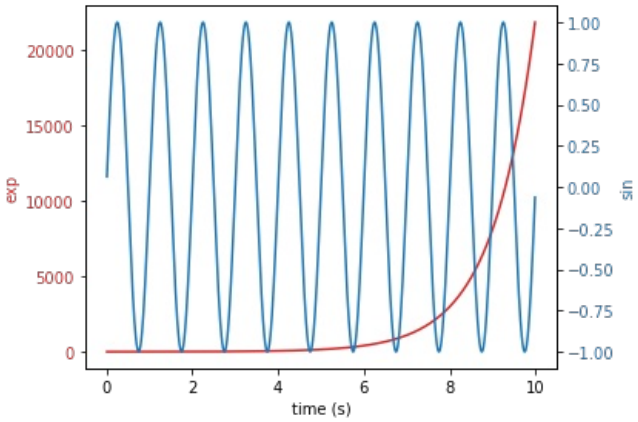
Aşağıda bir Dual Axis grafik örneği verilmiştir.

In [10]:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
```

In [13]:

```
import numpy as np
import matplotlib.pyplot as plt

# Create some mock data
t = np.arange(0.01, 10.0, 0.01)
data1 = np.exp(t)
data2 = np.sin(2 * np.pi * t)

fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('time (s)')
ax1.set_ylabel('exp', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('sin', color=color)  # we already handled the x-label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```



In [ ]:

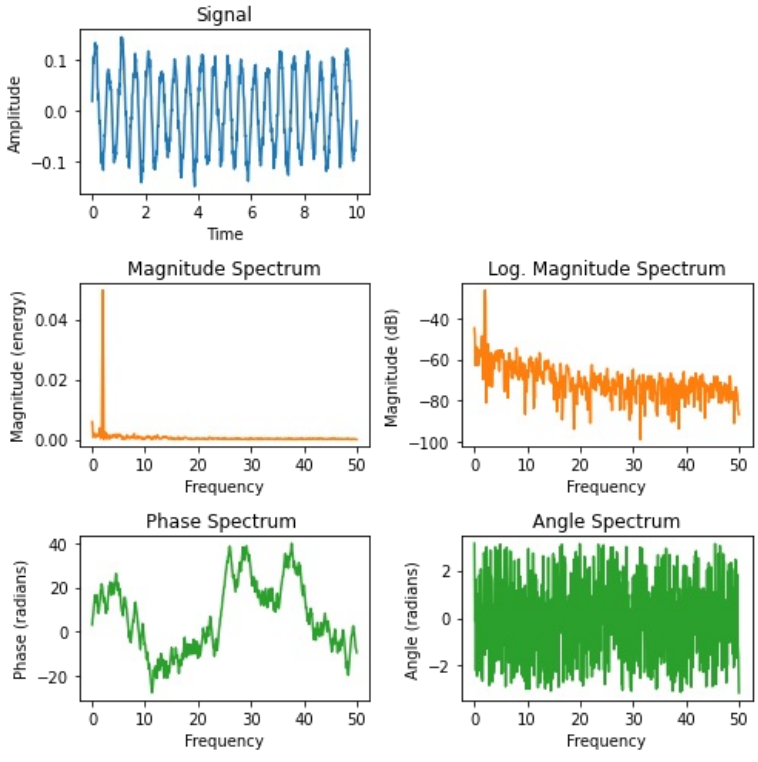
In [ ]:

Aşağıda farklı çizgi grafikleri mevcuttur. Bu grafiklerde maksimum değerin aşırı yüksek olması ve yüksek frekans nedeniyle okunaklı olmayan örnekler bulunmaktadır. Bu tip grafikleri düzeltmenin çeşitli yöntemleri bulunmaktadır tabiki amacınız maksimum değere veya yüksek veri frekansına vurgu yapmak değilse...

Not: Örnekler sadece gösterim amacıyla seçilmiş farklı amaçlar için üretilmiş örneklerdir.

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np


np.random.seed(0)

dt = 0.01  # sampling interval
Fs = 1 / dt  # sampling frequency
t = np.arange(0, 10, dt)

# generate noise:
nse = np.random.randn(len(t))
r = np.exp(-t / 0.05)
cnse = np.convolve(nse, r) * dt
cnse = cnse[:len(t)]

s = 0.1 * np.sin(4 * np.pi * t) + cnse  # the signal

fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(7, 7))

# plot time signal:
axs[0, 0].set_title("Signal")
axs[0, 0].plot(t, s, color='C0')
axs[0, 0].set_xlabel("Time")
axs[0, 0].set_ylabel("Amplitude")

# plot different spectrum types:
axs[1, 0].set_title("Magnitude Spectrum")
axs[1, 0].magnitude_spectrum(s, Fs=Fs, color='C1')

axs[1, 1].set_title("Log. Magnitude Spectrum")
axs[1, 1].magnitude_spectrum(s, Fs=Fs, scale='dB', color='C1')

axs[2, 0].set_title("Phase Spectrum ")
axs[2, 0].phase_spectrum(s, Fs=Fs, color='C2')

axs[2, 1].set_title("Angle Spectrum")
axs[2, 1].angle_spectrum(s, Fs=Fs, color='C2')

axs[0, 1].remove()  # don't display empty ax

fig.tight_layout()
plt.show()
```



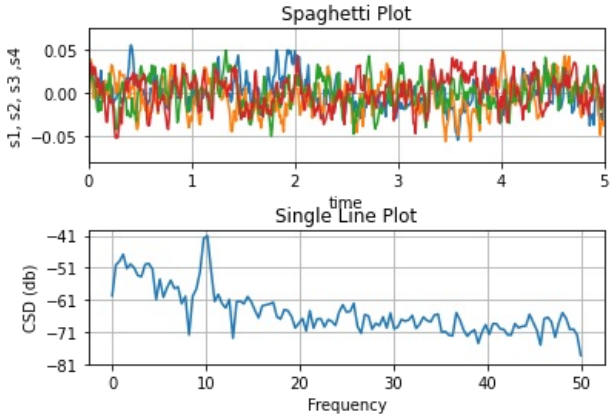
In [ ]:

```

```

Aşağıdaki örnek ise Spaghetti grafiği örneğidir. Çok sayıda karışık verinin tek grafikte çizilmesi okunaklı olmayan bir grafik sağlar.

In [7]:

```
import numpy as np
import matplotlib.pyplot as plt


fig, (ax1, ax2) = plt.subplots(2, 1)
# make a little extra space between the subplots
fig.subplots_adjust(hspace=0.5)

dt = 0.01
t = np.arange(0, 30, dt)

# Fixing random state for reproducibility
np.random.seed(19680801)


nse1 = np.random.randn(len(t))                 # white noise 1
nse2 = np.random.randn(len(t))                 # white noise 2
nse3 = np.random.randn(len(t))                 # white noise 3
nse4 = np.random.randn(len(t))                 # white noise 4
r = np.exp(-t / 0.05)

cnse1 = np.convolve(nse1, r, mode='same') * dt   # colored noise 1
cnse2 = np.convolve(nse2, r, mode='same') * dt   # colored noise 2
cnse3 = np.convolve(nse3, r, mode='same') * dt   # colored noise 3
cnse4 = np.convolve(nse4, r, mode='same') * dt   # colored noise 4

# two signals with a coherent part and a random part
s1 = 0.01 * np.sin(2 * np.pi * 10 * t) + cnse1
s2 = 0.01 * np.sin(2 * np.pi * 10 * t) + cnse2
s3 = 0.01 * np.sin(2 * np.pi * 10 * t) + cnse3
s4 = 0.01 * np.sin(2 * np.pi * 10 * t) + cnse4

ax1.plot(t, s1, t, s2, t, s3, t, s4)
ax1.set_xlim(0, 5)
ax1.set_xlabel('time')
ax1.set_ylabel('s1, s2, s3 ,s4')
ax1.set_title('Spaghetti Plot')
ax1.grid(True)

cxy, f = ax2.csd(s1, s2, 256, 1. / dt)  #cxy, f = ax2.csd(s1, s2, 256, 1. / dt, marker='o') marker burada bozuyor
.
ax2.set_ylabel('CSD (db)')
ax2.set_title('Single Line Plot')
plt.show()
```



In [ ]:

```

```