Subject :

Date

DATA VISUALIZATION WITH MATPLOTLIB

Matplotlib ⇒ Python ve Numpy için visualization library
2002 yılında John Hunter tarafından yazıld
Matlab gibi çalışır.

✳ Python üzerinde Matplotlib grafiklerini kullanmak için
`matplotlib.pyplot` modülünü kullanıyoruz.

PRE-CLASS

```
import matplotlib.pyplot as plt
%matplotlib inline

x = [1,2,3,4,5,6,7]

y = [50,51,52,48,47,49,46]
```

✳ `plt.plot(x, y, color='green', linewidth=5, linestyle='dashed')`

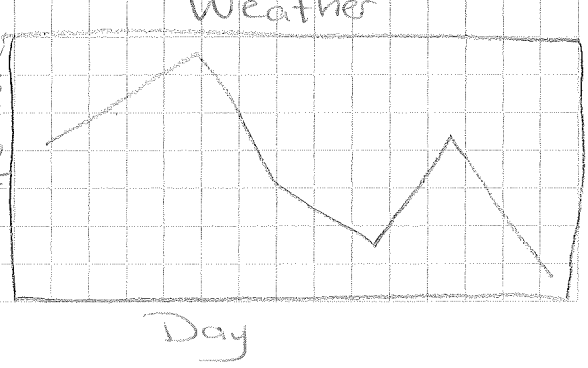
`linewidth=lw` 'de yazabilirsin.

kalınlık

`linestyle` = 'dashed'
'dotted'
Şekil-şükul

✳
```
plt.xlabel('Day')
plt.ylabel('Temperature')
plt.title('Weather')
plt.plot(x,y)
```

⇒



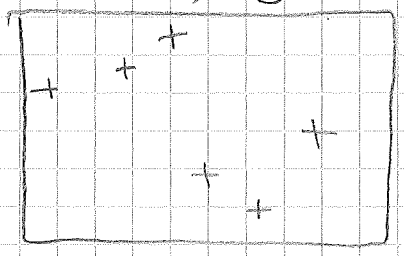
Subject :

Date : ...../...../.....

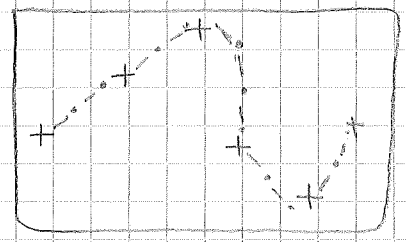
! Matplot web sitesinden kullanmak istediğin kodları seçebilirsin.

Matplot en popüler plotting library'dir
Seaborn gibi diğer plotting library'ler Matplot üzerinden kurulmuştur.

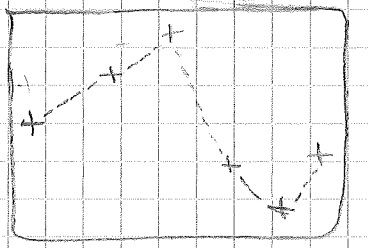
* plt.plot(x, y, 'g+')



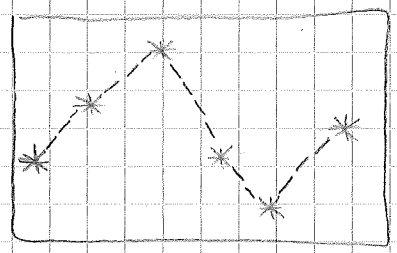
* plt.plot(x, y, 'g+-.')



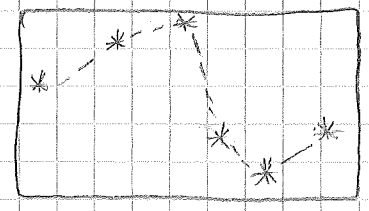
* plt.plot(x, y, 'g+--')



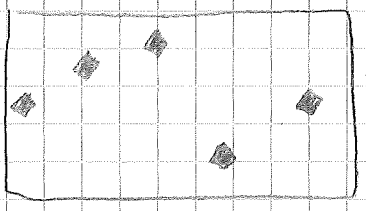
* plt.plot(x, y, 'r*--') / plt.plot(x, y, '--r*') — ikisi de olur



* plt.plot(x, y, 'rD--')



* plt.plot(x, y, 'rD')



* plt.plot(x, y, color='red', marker='D', linestyle='--')

Aynı kod

Subject :

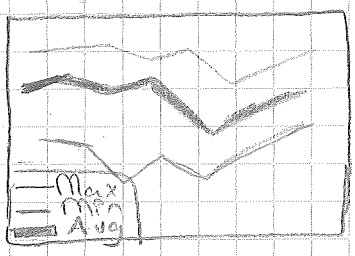
Date : ..../..../.....

- Kodun sonuna `, markersize=20` eklersen noktaları büyütür.

- `plt.plot(x, y, color='green', alpha=0.5)`
  Çizgiyi transparan yaptı.

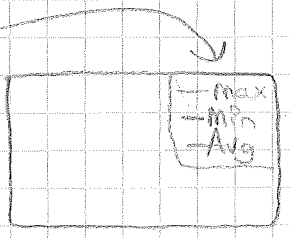
AXES LABELS, LEGEND, GRID

```
days = [1, 2, 3, 4, 5, 6, 7]
max_t = [50, 51, 52, 48, 47, 49, 46]
min_t = [41, 42, 40, 44, 33, 35, 37]
avg_t = [45, 48, 48, 46, 40, 42, 41]
plt.plot(days, max_t, label='Max')
plt.plot(days, min_t, label='Min')
plt.plot(days, avg_t, label='Avg')
plt.legend()
```



default => loc="best"

```
plt.legend(loc="upper right")
```

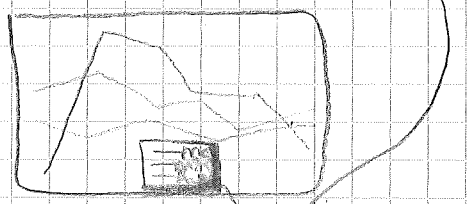


Subject :

Date : ..../..../.....

→ legend hangi renk kime ait default olarak belirler ve bir kutucuğa yazar.

✳ plt.legend(loc='best', shadow=True)



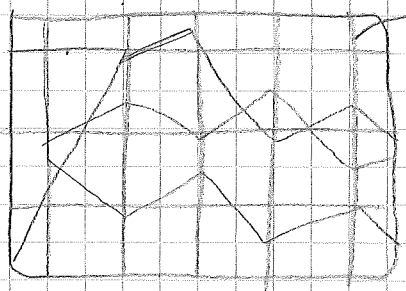
→ Gölgelendirme

✳ plt.legend(loc='best', shadow=True, fontsize="large")

"small"

yazıyı büyütür

✳ plt.grid()



→ grid() tabloya çizgiler çizer

Subject :

Date :...../...../.....

Object-oriented method :

fig=plt.figure()

1. Create a figure that forms an empty canvas
2. Add a set of axes to it
3. Plot whatever you want on these axes

⚠ How to add x-label when using object-oriented method?

```
axes.set_xlabel('____')
```

Subject : Date : ...../...../.....

PROBLEM

Create 4 subplots of $y_1 = x$, $y_2 = x^2$, $y_3 = x^3$, $y_4 = x^{0.5}$

Use points between x=0 to x=10

For $y_1 = x$ use red circles,

$y_2 = x^2$ use green dashes,

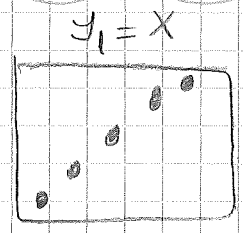
$y_3 = x^3$ use blue triangles,

$y_4 = x^{0.5}$ use black squares.

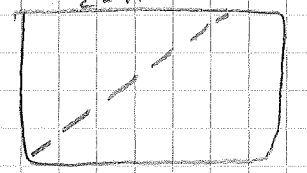
Also give each subplot a title. Include a command to save the figure as a PNG file so that it can be imported into another program such as Microsoft Powerpoint.

```
X = np.linspace(1,10,10)
y1 = x
y2 = x**2
y3 = x**3
y4 = np.sqrt(x)
plt.figure()
```

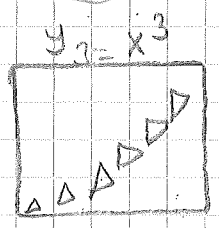
```
plt.subplot(2,2,1)
plt.plot(x, y1, 'ro')
plt.plot('$y_1 = x$')
```



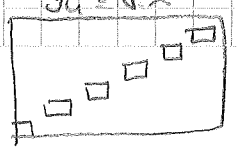

```
plt.subplot(2,2,2)
plt.plot(x, y2, 'g--')
plt.plot('$y_2 = x^2$')
```




```
plt.subplot(2,2,3)
plt.plot(x, y3, 'b^')
plt.plot('$y_3 = x^3$')
```




```
plt.subplot(2,2,4)
plt.plot(x, y4, 'ks')
plt.plot('$y_4 = \sqrt{x}$')
```



Subject :

Date : ...../...../.....

plt.subplot(4,1,4)

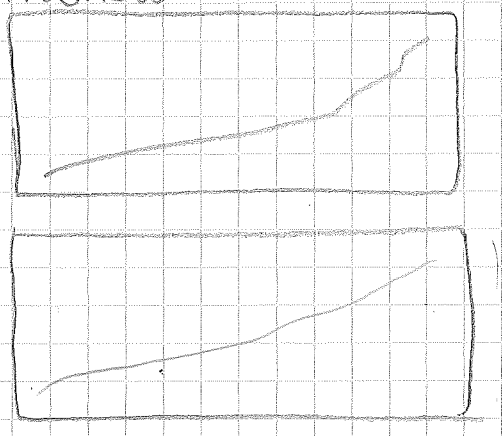
? Tablonun boyutunu değiştiriyor.

FUNCTIONAL METHOD

```
plt.subplot(2,1,1)
plt.plot(x1, y1)
plt.subplot(2,1,2)
plt.plot(x2, y2)
```

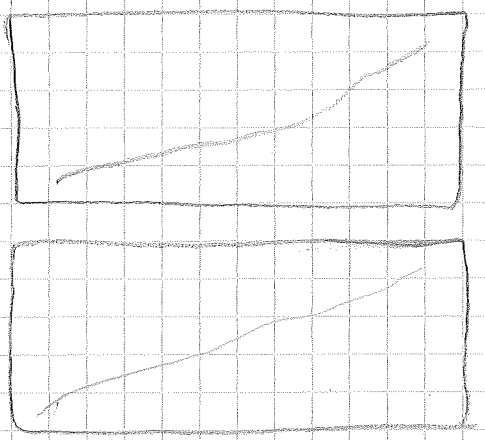
FUNCTIONAL

```
plt.subplot(2,1,1)
plt.plot(age, salary, 'r')
plt.subplot(2,1,2)
plt.plot(age, salary-2, 'b')
plt.show
```



OBJECT ORIENTED METHOD

```
fig, ax=plt.subplots(2,1)
ax[0].plot(age, salary, 'r')
ax[1].plot(age, salary, 'b')
plt.show()
```



```
fig, axes=plt.subplots(1,2)
fig, axes=plt.subplots(nrows=1, ncols=2)
```
Aynısı

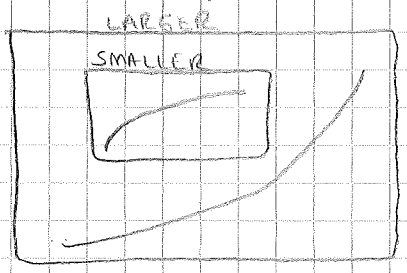
Subject : Date : ...../...../.....

Object-Oriented Method

```
fig = plt.figure()
axes1 = fig.add_axes([0.1,0.1,0.8,0.8])
axes2 = fig.add_axes([0.2,0.5,0.4,0.3])
axes1.plot(x,y)
axes2.plot(y,x)
axes1.set_title('LARGER')
axes2.set_title('SMALLER')
```

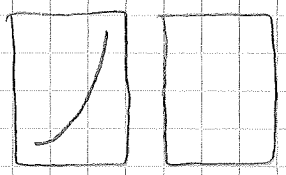
Pozisyon belirler — size 'kaç' belirtiyor.

left, bottom, width, height

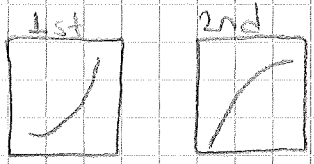



```
fig, axes = plt.subplots(nrows=1, ncols=2)
```

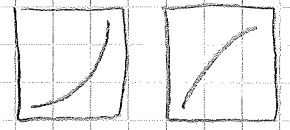
✳
```
axes[0].plot(x,y)
```



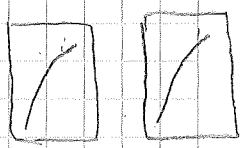
✳
```
axes[0].plot(x,y)
axes[0].set_title('1st')
axes[1].plot(y,x)
axes[1].set_title('2nd')
```



✳
```
axes[0].plot(x,y)
axes[1].plot(y,x)
```



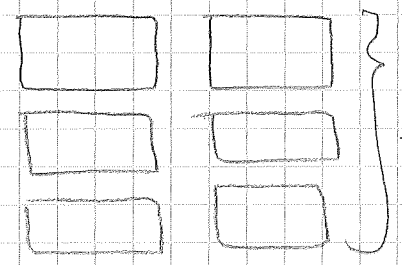
✳
```
fig, axes=plt.subplots(nrows=1,ncols=2)
for i in axes:
    if (5%2==0):
        a.plot(x,y)
    else:
        a.plot(y,x)
```



Subject :

Date ...../...../.....

* `fig, axes = plt.subplots(nrows=3, ncols=2)`



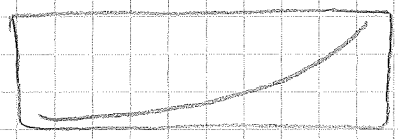
3'e 2

`plt.tight_layout()` → Tabloları birbirinden uzaklaştırdı.

* `fig = plt.figure(figsize=(6,2), dpi=100)` → Tablonun büyüklüğü

`ax = fig.add_axes([0,0,1,1])`

`ax.plot(x,y)`



* `fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(9,2))`

`axes[0].plot(x,y)`

`axes[1].plot(y,x)`

* `fig.savefig('my-chart.png')` → Resim olarak kaydetti?

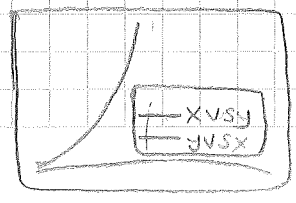
* `fig = plt.figure()`

`axes = fig.add_axes([0,0,1,1])`

`axes.plot(x,y, label='x vs y')`

`axes.plot(y,x, label='y vs x')`

`axes.legend(loc=(0.5,0.1))` → loc=2 (Sadece bunu da yazabilir. Pro)



Subject :

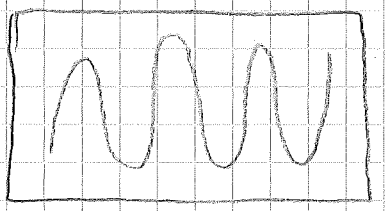
Date : ..../..../.....

MATPLOT STYLING

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style
style.use('ggplot')
```

→ veya 'dark_background' ← Birçok seçenek var

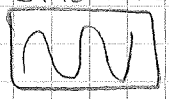
```
x = np.arange(0,30, 0.2)
y = np.sin(x)
plt.plot(x,y)
plt.show
```



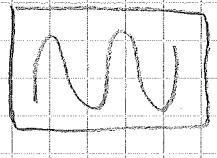
```
plt.grid(True)
```

değince  izgiler çiziyor.

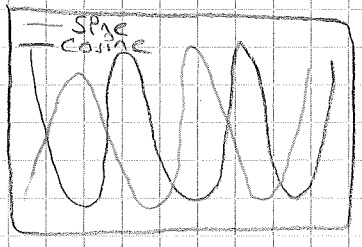
```
plt.title('Sine Function')
```

değince 

```
plt.subtitle('I am bigger')
```

→ 

```
y1 = np.sin(x)
y2 = np.cos(x)
plt.plot(x, y1, label="Sine")
plt.plot(x, y2, label="Cosine")
plt.legend(loc="upper left")
plt.show()
```



'upper right'
'upper left'
'bottom right'
'lower right'
} loc

13.10.2021

Subject :

Date : ...../...../.....

Matplotlib

Bar graphs
Histograms
Pie charts
Scatter plots
Lines

It uses comparatively complex and lengthy syntax

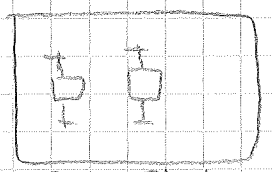
Seaborn

It uses fascinating themes

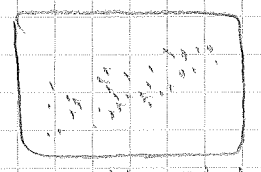
It uses comparatively simple syntax which is easier to to learn understand.

! pip install matplotlib

* import matplotlib
print(matplotlib.__version__) } Hangi versiyonu kullanıyorum?

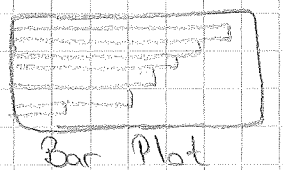


Box Plot



Scatter Plot

Pie chart



Bar Plot
(Kategorik veriler için)

Histogram Plot
(Continuous veriler için)

* import warnings;
warnings.filterwarnings('ignore') → Hata almayın ignore et. (Version farkından dolayı olabilir)

plt.plot(x,y); veya plt.plot(x,y)
plt.show()

Subject : Date : ..../..../....

Two METHODS

Functional

```
plt.plot(age,salary)
plt.xlabel("age")
plt.ylabel("salary")
plt.title("Salary by Age")
plt.show()
```

Object Oriented

```
fig, ax=plt.subplots()
ax.plot(age, salary, "r")
ax.set_xlabel("Age")
ax.set_ylabel("Salary")
ax.set_title("Salary by Age")
```

AYNISI

⇓

Daha pratik bir yöntem. Genelde daha az kod satırıyla yazılır.

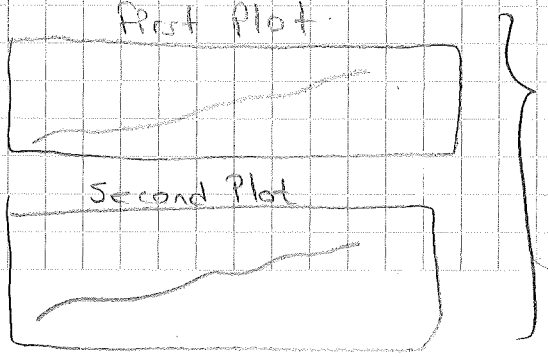
Figure → Genel çerçeve

Axes → Figure'ün altına tanımladığımız subplotlar

1 Figure, 2 Axes

```
fig, ax=plt.subplots(nrows=2, ncols=1)
ax[0].plot(age, salary, 'r')
ax[0].set_title('First Plot')
ax[1].plot(age, salary_2, 'b')
ax[1].set_title('Second plot')
plt.tight_layout()
```

Yanyana çizdirmek için:
nrows=1, ncols=2



1 col, 2 rows

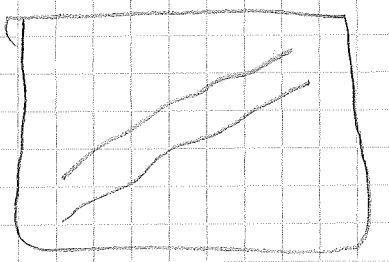
Subject :

Date : ...../...../.....

Multiple-Line in the same graph

Aynı grafikte salary ve salary_2'yi çizdirelim

```
plt.plot(age, salary)
plt.plot(age, salary_2)
plt.xlabel("age")
plt.ylabel("salary")
plt.title("Salary by Age")
plt.show()
```



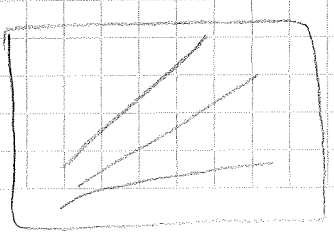
(*)
```
x = np.arange(0, 11)
y = x.copy()
z = x*2
t = np.log(x)
```

OOP ile:

```
fig, ax = plt.subplots()
ax.plot(x,y)
ax.plot(x,z)
ax.plot(x,t)
ax.set_xlabel("x")
ax.set_ylabel("y-z-t")
ax.set_title("Title")
plt.show()
```

Functional ile:

```
plt.plot(x,y)
plt.plot(x,z)
plt.plot(x,t)
plt.xlabel("x")
plt.ylabel("y,z,t")
plt.title("x'e göre y,z,t")
plt.show()
```



(2,2,3)
2'ye 2 alt
3. ye at

Subject :

Date : ...../...../.....

Subplot

Functional file:

2 row 1 column 1.ye red at

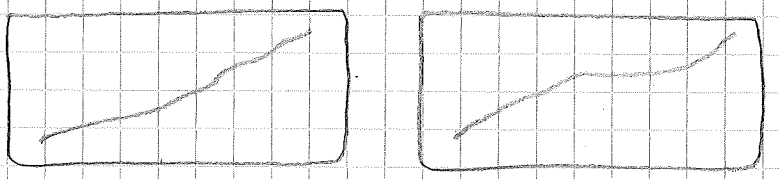
```
plt.subplot(2,1,1)
plt.plot(age, salary, "r")
```

red

2 row 1 column 2.ye blue at

```
plt.subplot(2,1,2)
plt.subplot(age, salary_2, "b")
plt.show()
```

blue



AYNISI

OOPile :

```
fig, ax=plt.subplots(2,1)
ax[0].plot(age, salary, 'r')
ax[1].plot(age, salary, 'b')
plt.show()
```

- `plt.tight_layout()` ⇒ Grafikler arası mesafeyi ayarlar
  `plt.show()`

Grafiklerin altına mutlaka yaz.

```
plt.tight_layout(pad=5)
```

Aralıktaki mesafeyi ayarlayabilirsiniz.

Subject :

Date : ..../..../.....

(*) ax[1][2] → 1. row, 2. column

ax[0][1] → Sıfırıncı row, 1. column

⚠ OOP ile çoğul durumlarda bir defa `plt.subplots()` yazıp içine ne kadar row ve column olduğunu yazabiliriz.

Ancak functional'da her grafik için ayrı ayrı `plt.subplots()` yazmamız gerekiyormuş.

Duştin Hoca'dan :

```
age = [ ---, ---, ---, ---, ---, ---, ---, ---, ---, ---, ---]

salary = [ ---, ---, ---, ---, ---, ---, ---, ---, ---, ---, ---]

fig, axes = plt.subplots(nrows=3, ncols=3)

renk = ['r', 'b', 'c', 'g', 'm', 'k', 'r', 'y', 'g']
a = 0
for x, y, z in axes:
    x.plot(age, salary, renk[a])
    a += 1
    y.plot(age, salary, renk[a])
    a += 1
    z.plot(age, salary, renk[a])
    a += 1
plt.tight_layout()
```

Subject :

Date : ..../..../.....

(*) Dustin hocadan :

```
plt.figure(figsize=(16,5))
a1 = plt.subplot2grid((5,5),(0,0),colspan=3,rowspan=2)
a2 = plt.subplot2grid((5,5),(0,3),rowspan=5)
a3 = plt.subplot2grid((5,5),(2,0),rowspan=3,colspan=3)
a4 = plt.subplot2grid((5,5),(0,4),rowspan=5)

X = np.arange(1,10)

a1.plot(x, np.exp(x))
a1.set_title('exp')

a2.plot(x, x*x)
a2.set_title('square')

a3.plot(x, np.log(x))
a3.set_title('log')

a4.plot(x, np.sqrt(x))
a4.set_title('sqrt')

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

Subject :

Date : ...../...../.....

Manuel figure :

```
fig = plt.figure()
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
```

Automatic figure

```
fig, ax = plt.subplots()
```

Adding Axes

```
ax1 = fig.add_axes([0.0, 0.0, 1, 1])
ax2 = fig.add_axes([1.1, 0.3, 0.4, 0.4])
```

left, bottom, width, height

[1.1, 0.3] → Pozisyon belirler.

[0.4, 0.4] → Size belirler.

Kodların başına yazınca grafiklerin birbirlerine göre boyut ve pozisyonlarını belirliyor.

Subplot

```
plt.subplot(2, 1, 1)
```

2 → row, 1 → column, 1 → 1. yere yaz

```
plt.subplot(2, 1, 2)
```

2 → 2. yere yaz

```
fig, ax = plt.subplots(2, 1)
ax[0].plot(..., ..., ...)
```

ax[0] → Soldaki kodla aynı 1. grafiğe yaz demek

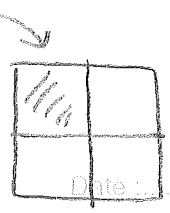
```
ax[1].plot(..., ..., ...)
```

ax[1] → Soldaki kodla aynı 2. grafiğe yaz demek

Subject :

Date : ../...../.....

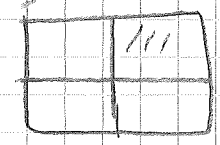
2'ye 2 bir grafik çiz 1.'ye yaz



```
plt.subplot(2,2,1)
plt.plot(age,salary,"r")
```

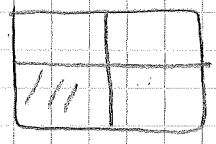
2.'ye yaz

```
plt.subplot(2,2,2)
plt.plot(age,salary-2,"b")
```



3.'ye yaz

```
plt.subplot(2,2,3)
plt.plot(age,salary-3,"yellow")
```

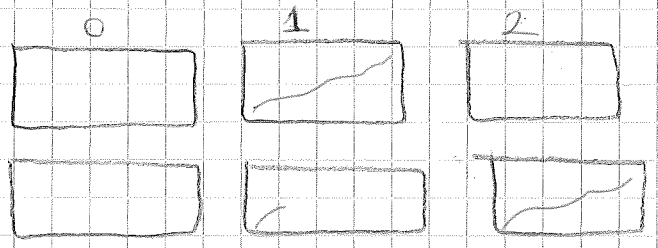


```
plt.subplot(2,2,4)
plt.plot(age,salary-4,"g")
```

4.'ye yaz

```
plt.tight_layout()   → Hizala
plt.show()
```

```
fig, ax=plt.subplots(2,3)  → 2x3=6 grafik olacak
ax[1][2].plot(age,salary)  → 1. satır, 2. sütuna yaz
ax[0][1].plot(age,salary)  → 0. satır, 1. sütuna yaz
plt.tight_layout()
plt.show()
```



ax[1,2] de yazabilirsin

ax[0,1] de yazabilirsin

Subject :

Date ...../...../.....

```
fig, ax = plt.subplots(nrows=2, ncols=3)
```

İlk etapta böyle yaz öğrenmek için ama sonra;

```
fig, ax = plt.subplots(2,3)
```

yazabilirsin aynısı.

Figure Size

```
fig, ax = plt.subplots(figsize=(16,4))
```

↓ grafiğimizin boyutu

FUNCTIONAL (subplot)

```
# plt.subplot(nrows, ncols, plot_number)
plt.subplot(2,1,1)
plt.plot(age, salary, 'r')
plt.title("First plot")
plt.subplot(2,1,2)
plt.plot(age, salary-2, 'b')
plt.title("Second plot")
plt.tight_layout()
```

OBJECT ORIENTED (subplots)

```
fig, ax = plt.subplots(2,1)
ax[0].plot(age, salary, 'r')
ax[0].set_title('First plot')
ax[1].plot(age, salary-2, 'b')
ax[1].set_title('Second plot')
plt.tight_layout()
```

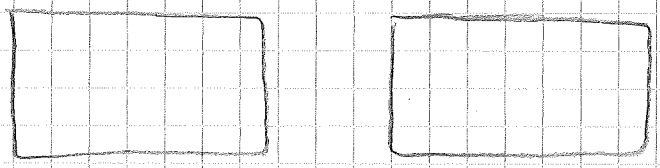
Subject :

Date : ..../..../.....

fig, suptitle ("Horizontally stacked subplots")

2 grafiğin ortak title'ı demek

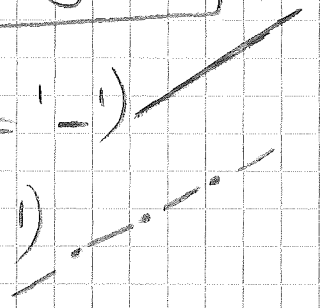
Horizontally stacked subplots



Linestyles → Grafikteki çizgileri ayırmak için

linestyle = '-')

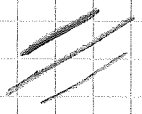
ls = '-.')



Linewidth → Çizginin kalınlığı, inceliği

lw = 0.25)

linewidth = 1.00)

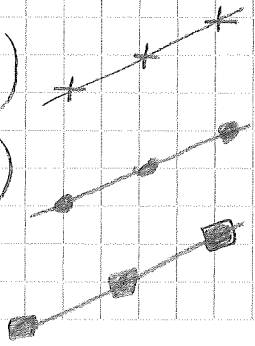


Marker

marker = '+')

marker = 'o')

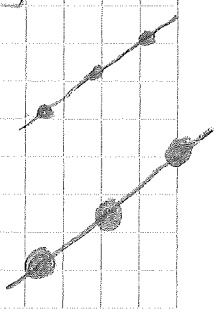
marker = 's')



Marker size

markersize = 2)

markersize = 4)



Subject :

Date : ...../...../.....

```
plt.figure(figsize=(15,5))
plt.suptitle("Linestyles", fontsize=10, y=1.2)
```

"Linestyles" → Ortak title

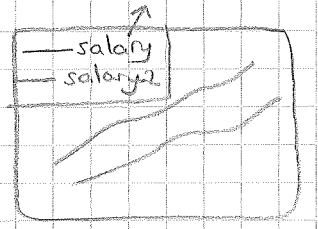
fontsize=10 → Ortak title'ın boyutu

y=1.2 → Ortak title'ın konumu

Legend → Etiket

```
plt.plot(age, salary, label="salary")
plt.plot(age, salary-2, label="salary-2")
```

plt.legend() → yukarda label belirtmezsem yazmaz



plt.legend(loc=4) ⇝ Sağ alta yazar.

plt.legend(loc="best") → En iyi yere yaz.

Subject :

Date : ...../...../.....

Plot Range and Adding Extra Lines

Plot Range with set_ylim & set_xlim

```
x=np.arange(1,10)
fig, ax=plt.subplots(nrows=1, ncols=2, figsize=(10,4))
```

□□ demek

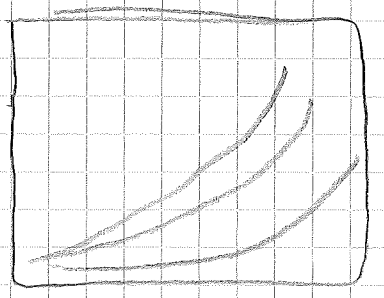
```
ax[0].plot(x, x**2, x**3, x**4, 'r')
ax[1].plot(x, x**2, x**3, x**4, 'r')
```

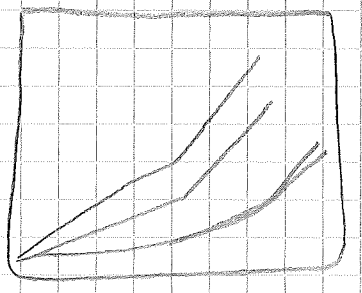
Functional methodla tek tek yazardik. Bu yöntemle kisa.

```
ax[1].set_xlim([1,3])
ax[1].set_ylim([1,50])
plt.show()
```

1 ile 3 arasini focusla, çünkü onlari incelemek istiyorum

y'lerini 1 ile 50 arasinda focusla





^

Subject :

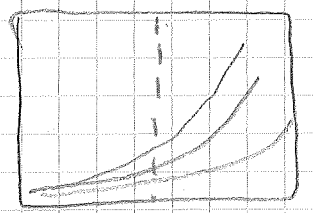
Date :...../...../.....

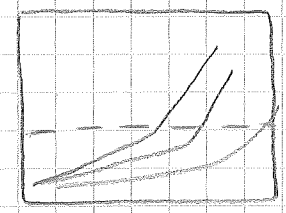
Adding Line

Önceki kodun üzerine :

ax[0].axvline (x=5, ls="--") → Dikey

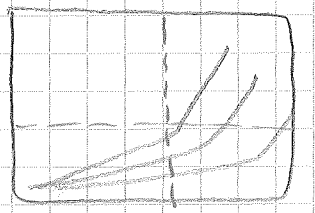
ax[1].axhline (y=5, ls="--") → Yatay





ax[1].axvline (x=2, ls="--")
ax[1].axhline (y=20, ls="--")

} 1. indexe hem dikey hem yatay line



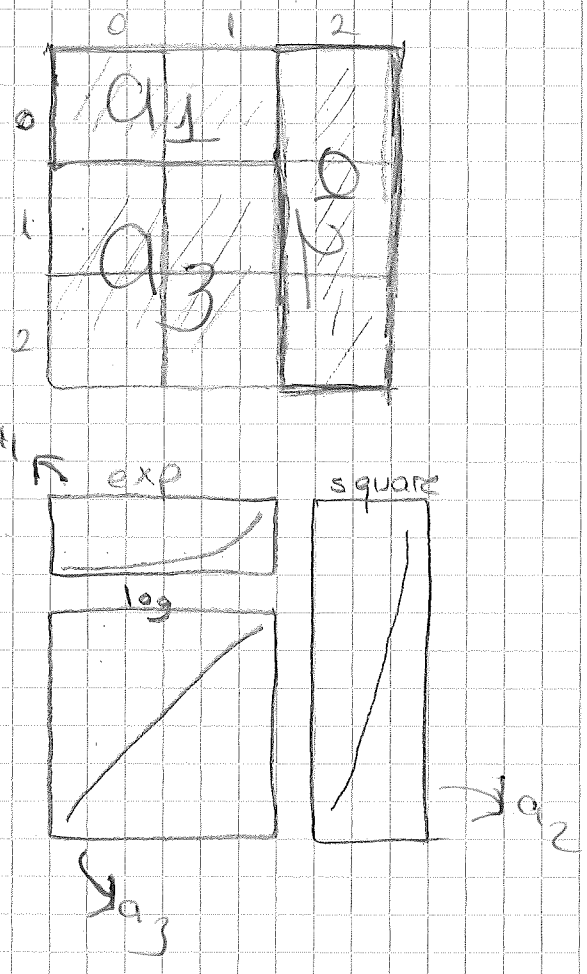
3'e 3'lük tablo yap (9'lu)

0. satır, 0. sütundan 2 sütun birleştir.

```
a1 = plt.subplot2grid((3,3), (0,0), colspan=2)
a2 = plt.subplot2grid((3,3), (0,2), rowspan=3)
a3 = plt.subplot2grid((3,3), (1,0), rowspan=2, colspan=2)
x = np.arange(1,10)
a1.plot(x, np.exp(x))
a1.set_title('exp')
a2.plot(x, x*x)
a2.set_title('square')
a3.plot(x, np.log(x))
a3.set_title('log')
plt.tight_layout()
plt.show()
```

0. satır 2. sütundan 3 satır birleştir

1. satır, 0. sütundan 2 satır 2 sütun birleştir



Subject :

Date : ...../...../.....

Scatter Plot

```
plt.scatter(x,y)
plt.show()
```

```
tips = sns.load_dataset("tips")
tips.head()
```

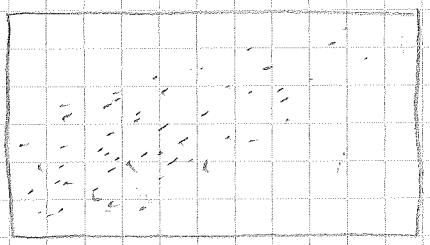
! sns.get_dataset_names()
↓
Datasetlerin isimlerini listeliyor.

Bu datasetle ilgili araştırmalar yapacaksınız.

→

total_bill	tip	sex	smoker	day	time	size

```
plt.scatter(tips["tip"], tips["total_bill"]);
```



veya

```
plt.scatter(tips.tip, tips.total_bill)
```

↓
Yerler değişebiliyor.

Subject :

Date : ..../..../.....

Bar Chart

```
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23, 17, 35, 29, 12]
fig, ax = plt.subplots()
ax.bar(langs, students)
```

Tips dataframe'de günlere göre total bill'e bakalım:

* `tips.groupby("day").sum()`

* `day = tips.groupby("day").sum().index`
  `day`

* `tips.day.unique()` → day'deki unique değerleri aldık

* `day_of_total_bill = list(tips.groupby("day")["total_bill"].sum())`
  → liste halinde istedik

* `fig, ax = plt.subplots(figsize=(12,8))`
  `ax.bar(day, day_of_total_bill)`
  `plt.show()`

  } Barplotu çizdirdik
  (bar yerine barh yazarsan horizontal görür.

Subject :

Date : ...../...../.....

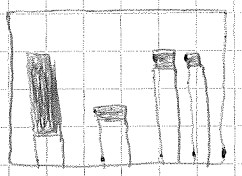
(*) Günlere göre tips'i tanımlayalım:

```
tipp = list(tips.groupby('day')['tip'].sum())
tipp

fig, ax = plt.subplots()
ax.bar(day, tipp)
plt.show
```

(*) 2 barplotu üst üste çizdirdiği için aynı renkte iki grafiği inceleyebiliriz.

"Üst üste çizdi"



```
fig, ax = plt.subplots()
ax.bar(day, day_of_total_bill, label="total_bill")
ax.bar(day, tipp, label="tipp")
plt.legend()
```

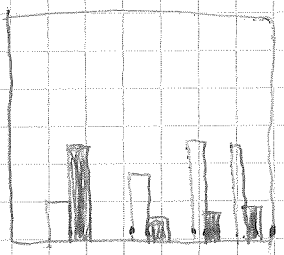
"Üst üstede yan yana geldi."

Optional hard to reading

(*)

```
fig, ax = plt.subplots(figsize=(10,5))
```

yan yana çizdi



```
p = np.arange(len(day))
width = 0.15
ax.bar(p - width/2, day_of_total_bill, width, label="total_bill")
ax.bar(p + width/2, tipp, width, label="tipp")
ax.set_xticks(p)
ax.set_xticklabels(day)
plt.legend()
plt.show()
```

su kadar sağa kaydır

su kadar sağa kaydır

Subject :

Date : ..../..../.....

Histogram

```
plt.hist(x)
plt.show()
```

(*) `x = np.random.randn(10)`

```
plt.hist(x, bins=30)
plt.show()
```

bins → Çubuk sayısını artırır ve okunabilirlik artar.

(*) `plt.hist(tips["total_bill"], bins=40)`
`plt.show()`

→ Köşeli parantez yerine nokta da kullanabiliriz.

(*) `plt.hist(tips.tip, bins=40)`
`plt.show()`

---

2 kadın arasında correlation var mı?

`np.corrcoef(tips.tip, tips.total_bill)`

→ 0.67

Subject :

Date : ...../...../.....

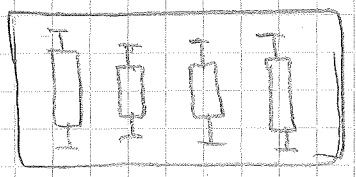
Box Plot

Outliers'ları tespit ediyoruz.

```
plt.boxplot(x,)
plt.show()
```
→ Tek boyutlu yani y değeri yok

(*)
```
x= np.random.normal(170,10,250)
plt.boxplot(x)
plt.show()
```

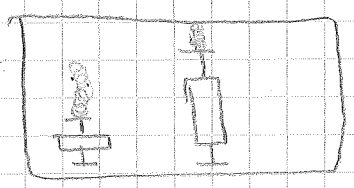
(*)
```
box= np.array(np.random.rand(10,4))
plt.boxplot(box, labels=['A','B','C','D'])
plt.show()
```



4'ü aynı grafikte gördük

(*)
```
plt.boxplot(tips["tip"])
plt.show()
```

(*)
```
plt.boxplot([tips.tip, tips.total_bill])
plt.show()
```
→ İkisi aynı grafikte (tip, total_bill)



Subject :

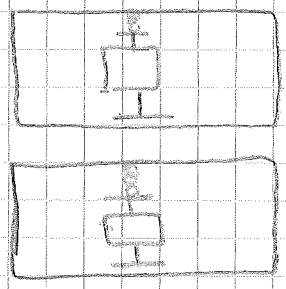
Date : ..../..../.....

(*) fig, ax = plt.subplots(nrows=2, ncols=1)

ax[0].boxplot(tips.tip)

ax[1].boxplot(tips.total_bill)

plt.show()



(*) tips.iloc[:, 0:2]

↳ Satırların hepsini sütunlardan da 0. ve 1. index'leri getir.

(*) plt.boxplot(tips.iloc[:,0:2], labels=['total_bill', 'tip'])

plt.show()

Subject : Date ....../....../......

Pie Chart

```
plt.pie(y, labels = mylabels)
plt.show()
```

(*) y = np.array([35, 25, 25, 15]) → Verideki değerlerin toplamı 100 olmalı

mylabels = ["apples", "bananas", "cherries", "dates"]

```
plt.figure(figsize = (10, 8))
plt.pie(y, labels=mylabels, labeldistance =0.7,
plt.show()                  autopct="%.2f")
```

labeldistance → Merkeze olan uzaklık

autopct → Virgülden sonra bir basamak al

Hangi dilimi öne çıkaracağımızı söylüyoruz.
explode = [0.1, 0, 0, 0] → 0. indexi öne çıkar

Kodun içine yazdık.

(*) plt.pie(y, labels=mylabels, explode = myexplode,
labeldistance=0.7, autopct = "%.2f")

(*) plt.pie(y, labels=mylabels, labeldistance = 0.7 , autopct="%.2f",
startangle= 90 , → İlk indexi 90°'ye koy, diğerlerini ona göre döndür. (Saat 3 yönüne koy)
shadow=True , → Gölgelene yapıyor
pctdistance = 1.1 , → Dilimi uzaklaştırıyor
colors = color_list , → Yukarda bir color_list tanımlamıştı.

Subject :

Date : ...../...../.....

autopct ⇒ Virgülden sonra kaç basamak alınacak

startangle ⇒ Kaç derece döndürülecek

pctdistance ⇒ Sayıların yazılacağı mesafe

explode ⇒ Merkezden uzaklaşan parça dilimleri

labeldistance ⇒ Birimlerin merkezden uzaklığı
(0 merkez, 1 çember)

Subject :

18.10.2021

Date :...../...../.....

SEABORN

Seaborn daha esnek ve kullanışlymış.

import seaborn as sns

print(sns.get_dataset_names() → Datasetlerin isimlerine baktık

Scatter Plots

```
sns.scatterplot( x="total_bill",
                 y="tip",
                 data=tips,
                 hue="sex",
                 s=200,        ← size
                 style="sex"
                 alpha=0.2)    → Transparanlığı ayarlar
```

Seaborn'da görsel olarak daha güzel grafikler çizeriz.
Karmaşık grafikleri daha kolay çizebiliriz.

Gruplamada { hue, size, style } kullanabiliriz

! plt.savefig('example_scatter.jpg')

Notebook nerdeyse grafiği jpg olarak araya kaydeder

Subject :

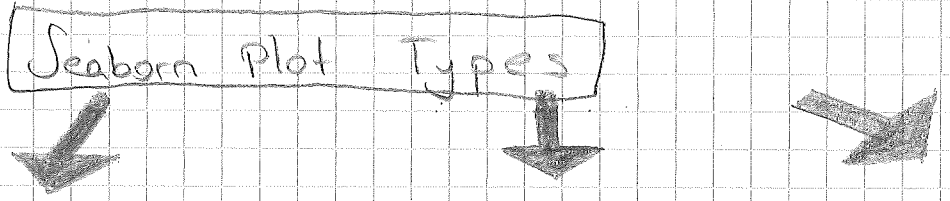
Date : ...../...../.....

Matplotlib'i seaborn'a entegre edip birlikte kullanabiliyoruz.

```
plt.figure(figsize=(12,8))
ax=sns.scatterplot(x='total_bill', y='tip', data=tips)
ax.set(xlabel="Total Bill", ylabel="Tip", title="Some title")
#plt.legend(labels=['Legend1'])
plt.show()
```

Hue → groupby yapıyor

2 kategorik değeri sınıflandırmada kullanılır.



Distributions Plots

- ✱ kdeplot
- ✱ rugplot
- ✱ displot
- ✱ hisplot

↓

En çok kullanacaklarımız.

Categorical Plots

- ✱ barplot
- countplot
- ✱ boxplot
- swarmplot
- violinplot

Comparison Plots

- jointplot
- ✱ pairplot
- catplot
- ✱ matrix plot
- grid plot

Subject :

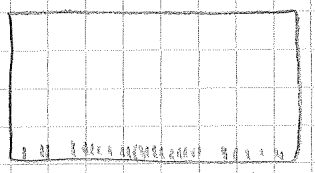
Date : ...../...../.....

rugplot → Tek değişkenli

Her bir değer için bir çizgi koyar. Sadece x'i tanımlıyoruz

```
import seaborn as sns
tips=sns.load_dataset("tips")
sns.rugplot(x="total_bill", data=tips, height=0.5)
sns.rugplot(tips["total_bill"])
```

height → çizgileri uzatır

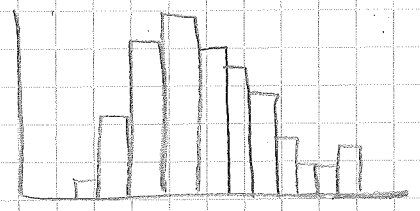


→ sadece x değeri var.

displot → X'deki her unique değeri sayıp y'ye atar. Skewness hakkında bilgi elde edebilirsiniz.

```
sns.displot(x='total_bill', data=tips, kde=True)
sns.displot(tips['total_bill'])
```

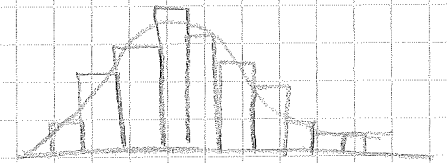
kde=True → bu çizgi



hisplot

```
sns.hisplot(x="total_bill", data=tips, kde=True, bins=10)
sns.hisplot(tips["total_bill"])
```

bins → Her aralığa kaç değer düşüyor onu böler



Adding in a Grid

"whitegrid"

```
sns.set(style="darkgrid")
```
→ Arkaplana çizgi çekiyor

```
sns.hisplot(data=tips, x="total_bill", bins=20)
```

Daha kapsamlı kodlar

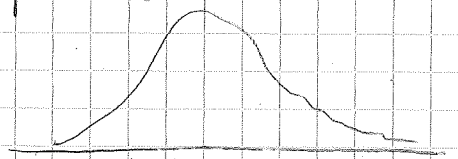
```
sns.set(style="darkgrid", rc={"grid.color": ".6",
                              "grid.linestyle": ":"})
sns.hisplot(data=tips, x="total_bill", bins=20);
```

↓ Grid'ler arasındaki stili değiştiriyor.

color = renk
edgecolor = kenarlık
lw = kenarlık kalınlığı
ls = kenar şekli

The Kernel Density Estimation Plot

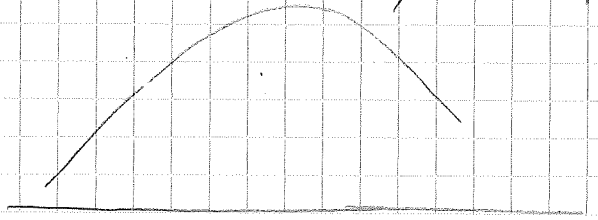
```
sns.kdeplot(x="total_bill", data=tips);
```



↓ Sadece kde çizer. Bar plotlar olmaz.

Cut of KDE → Hangi aralığı istiyorsak onu kırptık.

```
sns.kdeplot(x="total_bill", data=tips, clip=[10,20]);
```

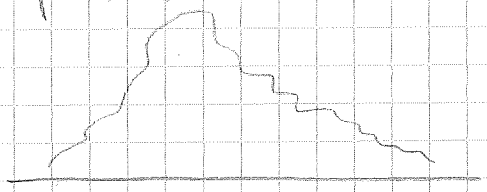


Subject :

Date : ...../...../.....

Bandwidth

```
sns.kdeplot(x="total_bill", data=tips, bw_adjust=0.2);
```



, shade=True

dersem içini doldurur.

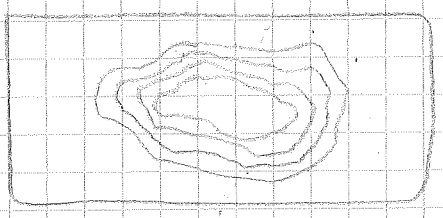
, color="red"

dersem renkli olarak doldurur.

Dimensional KDE Plots

```
data = pd.DataFrame(np.random.normal(0, 1, size=(100, 2)),
                    columns=['x', 'y'])
data

sns.kdeplot(data=data, x="x", y="y")
```



, shade=True

içini doldurur.

, cbar=True

Sağ taraftaki sayıları ekledi ama anlamadım.

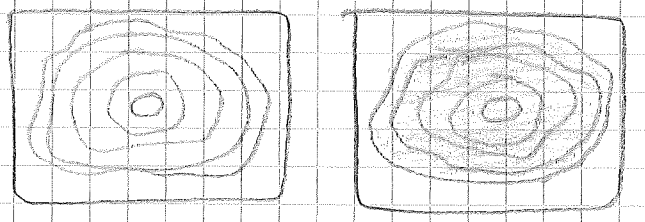
Subject :

Date : ..../..../.....

Muhammet kocadağ

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.kdeplot(data=data, x="x", y="y")

plt.subplot(1,2,2)
sns.kdeplot(data=data, x="x", y="y", shade=True);
```



Subject :

21.10.2021

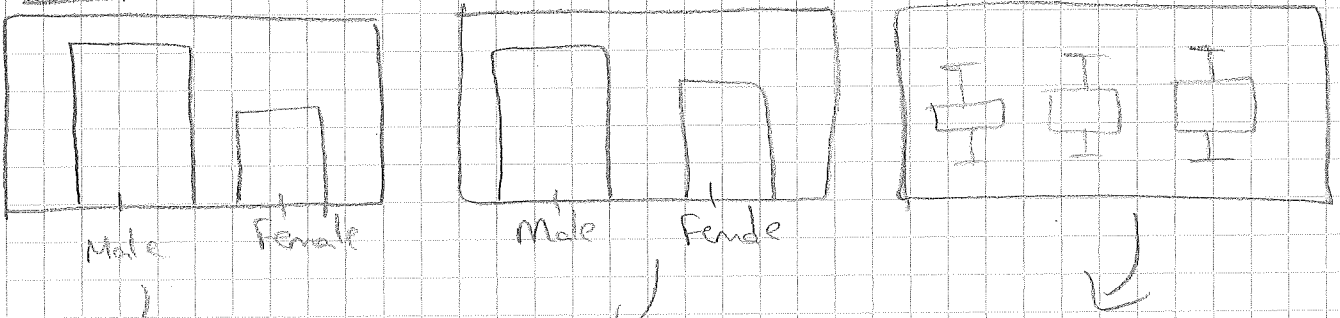
Date : ..../..../.....

Categorical Plots

barplot

countplot

boxplot

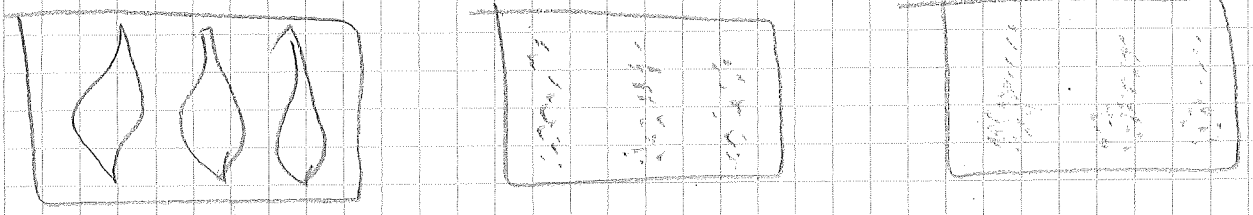


Default'u mean

violinplot

stripplot

swarmplot



① countplot → Sadece x değeri veriyoruz
Barplot'ta x-y değerleri veriyoruz.

```
import seaborn as sns
tips = sns.load_dataset("tips")
sns.countplot(x='day', data=tips)
sns.countplot(tips['day'])
```

Subject :

Date : ...../...../.....

```
fig, ax=plt.subplots()
ax=sns.countplots(x='day' ,data=tips)
for p in ax.patches:
    ax.annotate((p.get_height()), (p.get_x()+3,
                p.get_height()+0.5));
```

```
sns.countplot(mpg.model_year);
```

veya

```
sns.countplot(x='model_year', data=mpg);
```

Duruşa göre

```
fig, ax=plt.subplots()
ax=sns.countplot(x='day',data=tips, hue='sex')
for p in ax.patches:
    ax.annotate((p.get_height()),(p.get_x()+0.1,
                p.get_height()+0.7))
    ax.annotate((round(p.get_height()/tips.day.count(),
                2),
                (p.get_x()+0.1,
                p.get_height()-4))
```

Subject: Model yıllara göre orijinleri karşılaştıralım: Date: ..../..../.....

```
sns.countplot(x="model_year", hue="origin", data=mpg);

sns.countplot(mpg.model_year[mpg.model_year<75]);
```

Subject :

Date : ..../..../.....

Barplot → x'e kategorik değerleri tanımlar. x'e karşılık gelen değerleri y'ye atar.

(*) sns.barplot(x="sex", y="total_bill", data=tips)

↓

Sex'e göre total_billeri aldık

(*) sns.barplot(x="day", y="total_bill", data=tips, hue="sex")

↓ ↓ ↓

Günlere göre total_billerini aldık tips datasında çalıştık.

İçine başka neler verebiliriz?

estimator=np.sum

np.mean

np.median

np.max

np.count_nonzero

Subject :

Date : ..../...../.....

Barplot ve Countplot Birlikte

```
fig, ax=plt.subplots(1, 2, figsize=(12,4))

sns.barplot ( x="day", y="total_bill", data=tips, ax= ax[0], estimator=np.count_nonzero)
sns.countplot(x="day", data=tips, ax=ax[1]
plt.tight_layout()
```

Barplotta y verdik countplotta vermedik. Sayıp kendisi yazdı.

Boxplot → x ve y'yi alıyor

```
sns.boxplot(x='day', y='total_bill', data=tips)
```

hue='sex' ekledik. Gender'a göre değerlere baktık.

```
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Seaborn'un yetmediği yerde matplotlib birlikte seaborn'u kullanabiliyoruz.

```
plt.figure(figsize=(10,5))
```

Subject :

Date : ...../...../.....

Orientation → Görsel olarak x ve y'i ters çeviriyor

```
sns.boxplot(y="total_bill", x='day', data=tips,
            hue='sex',
            orient='h',
            width=0.3)
```

↳ Kutuların boyutlarıyla oynuyoruz.

Violinplot | boxplot + kde

1 veya daha fazla kategorik değişkenin dağılımlarını gösteren grafik elde edebiliyoruz.

```
sns.violinplot(x='day', y='total_bill', data=tips,
               hue='sex',
               inner=None
               bw=0.2);
```

x ve y'nin yerini değiştirirsen grafik yan döner.

inner: İçinden boxplotu çıkarır sadece kde'ye göre gösterir

bw: Dalgalı bir yapı oldu. Hassasiyetleri daha iyi göstermek için.

Subject :

Date : ...../...../.....

Hoca bir data set gönderdi:

```
df=pd.read_csv("Students Performance.csv")

sns.boxplot(x="parental level of education",
            y="math score", data=df)
```

X eksenindeki yazılar değil

Görseller kötü ya `rotation=45` gibi rotation veriniz.

ya da

`plt.figure(figsize=(16,6))` yazarız.

Subject :

Date : ..../..../.....

Swarmplot

Noktalı ağaç gibi bişiler oldu.

```
sns.swarmplot(x='day', y='total_bill', data=tips,
              size=7,
              hue="gender"
              dodge=True)
```

→ Sadece x'i de verebiliyorum

→ İç içe olan görseli ayırdı

Boxenplot → Daha büyük veri setlerinde kullanılırmış.

```
sns.boxenplot(y='math score', x="race/ethnicity", data=df
```

Lineplot

```
sns.lineplot(data=flights, x="year", y="passengers",
             hue="month")
```

Subject :

Catplot → Bir veya daha fazla grafik aynı anda. Kategorik verilerin hepsini aynı anda inceleyebiliriz

Catplot kind = barplot
violinplot
stripplot
swarmplot
boxplot

Aynı anda subpotlar şeklinde grafik çizdirecek bize

```
import seaborn as sns
tips = sns.load_dataset("tips")
sns.catplot(data=tips, x="day", y="total_bill",
            kind="bar", col="smoker", row="sex")
```

Comparison Plots → İki değişkenli bir dağılım elde edilir

Pairplot

```
tips = sns.load_dataset("tips")
sns.pairplot(tips, hue="smoker")
```

Subject :

Date : ..../..../.....

Grids

1) Pairgrid → Sayısal değişkenler için grid oluşturur.

```
g = sns.PairGrid(tips)
g = g.map_upper(sns.scatterplot)
g = g.map_diag(sns.histplot)
g = g.map_lower(sns.kdeplot)
```

2) Facegrid

```
g = sns.FacetGrid(data = tips, col = "time", row = "smoker")
g = g.map(plt.hist, "total_bill")
```

Matrix Plot

```
sns.heatmap(df.corr()) → Corelasyona göre renk skalası veriyor.
```

Subject :

Date : ...../...../.....

Catplot

estimator → default'u mean

```
sns.catplot(x="pclass", y="survived", data="titanic",
    kind="box"
    kind="violin"
    kind="swarm"
    kind="strip"

    kind="bar", estimator=np.count_nonzero,
                np.sum
    ci=5d,
    hue="sex",
    col="alone",
    row="embarked");
```

Alone:
→ True-false olur

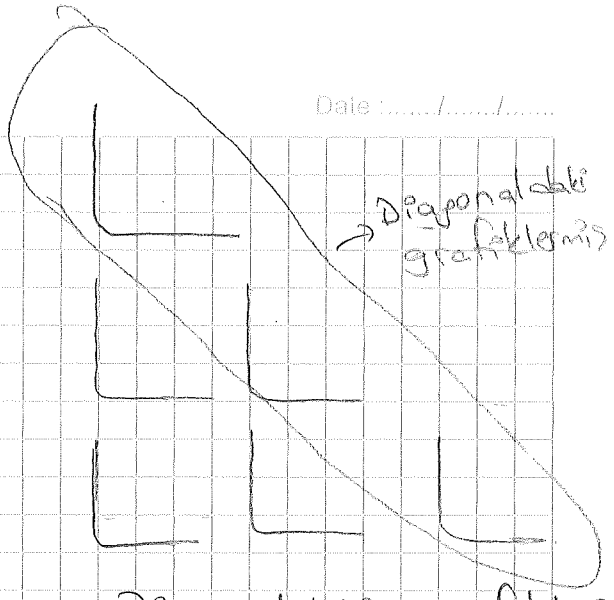
Joinplot

```
sns.joinplot(x="total_bill", y="tip", data=tips,
    kind="hex"
    kind="reg"
    kind="kde"
    kind=
```

Subject :

Date : ...../...../.....

Pairplot

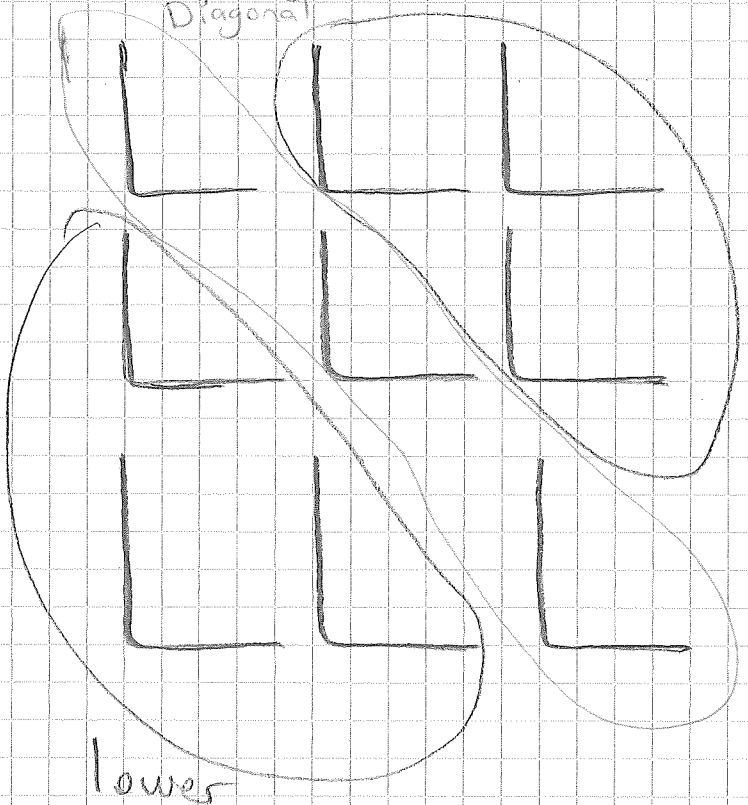



```
sns.pairplot(tips,
             hue="sex",
             palette="viridis",
             corner="True",
             diag_kind='hist'
```

→ Diagonaldaki grafikleri histogram grafiklerine çevirdi.

Pairgrid

Birden çok grafiği birleştirmek yine aynı şekilde.



Subject :

Date : ...../...../.....

```
g=sns.PairGrid(tips) → Bu tek fonksiyon
g=g.map_upper(sns.scatterplot)
g=g.map_diag(sns.histplot)
```

→ Lower tanımlamadığımız için grafikler boş geldi.

```
g=g.map_lower(sns.kdeplot, color="red")
```

```
g=sns.PairGrid(tips, hue="sex", palette="viridis")
g=g.map_upper(sns.scatterplot, linewidths=1, edgecolor="w", s=40)
g=g.map_diag(sns.distplot)
g=g.map_lower(sns.kdeplot)
g=g.add_legend();
```

```
a=sns.pairplot(tips)
a.map_lower(sns.kdeplot, color='purple')
a.map_upper(sns.kdeplot, color='g');
```

Subject :

Date : ...../...../.....

FacetGrid

```
g=sns.FacetGrid(data=tips, col="time", row="smoker")
g=g.map(plt.hist, "total_bill")
```

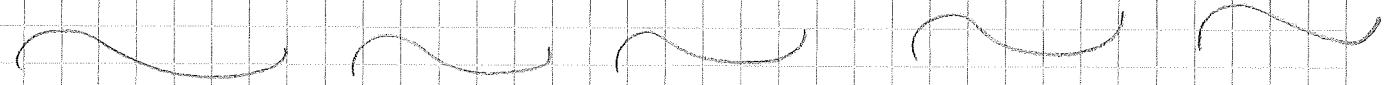
↳ sns'le de yapılabilir.

Time ve smoker'a göre gruplayıp total_bill'in histogramını çizdirecek

```
g=g.map(plt.scatter, "total_bill", "tip")
```

Heatmap

```
sns.heatmap(data=tips.corr(), annot=True
            center=1)
```



```
iris=sns.load_dataset("iris")
species=iris.pop("species")
sns.clustermap(iris)
```