Buse Atabay

WEEK 1 - Python Tekrarı

(*)

```
x= 12.2
y= 14
```

} diye tanımladık

x=100 → artık "x" "100", update edildi.

a = x * y (diyebilirsin)

x = 100 * x + (2-x) * x

x yine update edildi

(*) input() → içine string yazılabilir sayı yazıcaksan convert etmen lazım.

ÖR:

yo da "int" diye burda da belirtebilirsin.

```
a = input ('Europe floor?')

b= int (a) + 1
```

→ input'a sayı girild. için belirtmem gerekti (convert)

float → ondalıklı sayı

ÖR:

float old. inputun önünde belirtti direk

```
xh= float (input("Enter Hours"))

xr= input ("Enter rate:")

pay = xh * float (xr)

print ("Pay:", pay)
```

işlem sırasında belirtti "xr" için

(*) ÖR:

```
x=5

if x<10 :

    print ("Smaller")

if x>20 :

    print ("Bigger")

print ("Finish")
```

x=15 olsaydı if line'ları atlar direk "Finish" yazardı.

Output:

```
Smaller
Finish
```

(*) If

if x==5 → 5'e eşitse

if x>=5 → 5'e büyük-eşitse

if x!=6 → 6'ya eşit değildir!

(*) ÖR:

```
x=42

if x>1 :

    print ('More than 1')

    if x<100 :

        print ('Less')

print ('All done')
```

x=0 derse direk {All done} yazar!

Bu if üstteki if'in içinde kalmış o yüzden ilk "if" koşulu sağlarsa o zmn 2. if'e geçebiliyor!

(*) ÖR:

```
x=4

if x>2 :

    print ('Bigger')

else :

    print ('Smaller')

print ('All done')
```

x>2 ise ← if

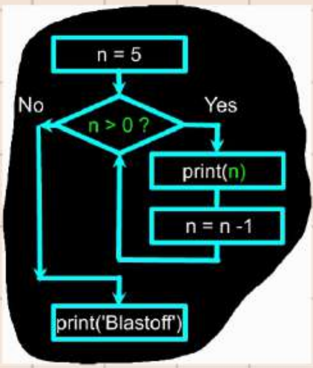
Değilse ← else

(*)

```
if x<2 :
    print ( . _ )
elif x<10 :
    print ( . . . )

else :
    print ( - - - )

print ('All done')
```

üstteki koşul yanlışsa buna geçer x<10'sa onu yazar

⊛ While Loop



```
n=5
while n>0 :
    print n
    n= n-1
print 'Blastoff!'
print n
```

Output

```
5
4
3
2
1 (n>0 olana kadar devam etti)
Blastoff! (Bitti)
0 (En son final value'yu yazdı)
```

n = n-1 satırı çıkarılırsa dönüp dönüp 5 yazacağı için alt alta sonsuz tane "5" yazar.

n = n+1 olursa 1,2,3- - - - ∞ devam eder

n=0 olursa baştaki

⊛ for Loop :

```
for i in [5,4,3,2,1] :
    print (i)
print ('Blastoff')
```

Output

```
5
4
3
2
1
Blastoff
```

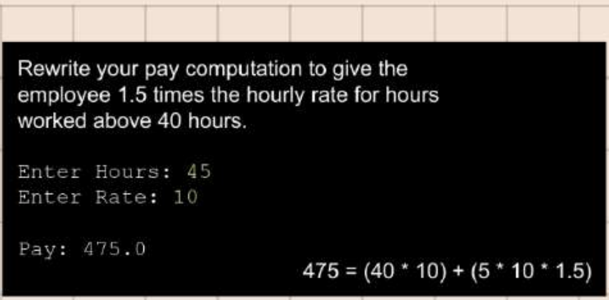
Ör:

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
    print ('Happy New Year :', friend)
print ('Done!')
```

Output

```
Happy new year : Joseph
  "     "    "  : Glenn
  "     "    "  : Sally

Done!
```

EXP 8



Rewrite your pay computation to give the employee 1.5 times the hourly rate for hours worked above 40 hours.

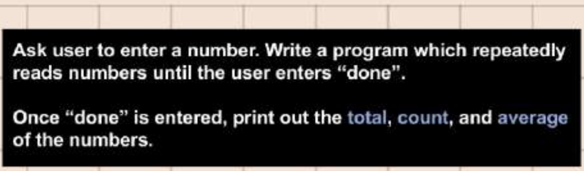
```
Enter Hours: 45
Enter Rate: 10

Pay: 475.0
```

475 = (40 * 10) + (5 * 10 * 1.5)

(40 saatin üzerinde çalışılan saatler için saat ücretinin 1.5 katı verilecek şekilde yazın.)

```
xh= float(input("Enter Hours:"))

xr = float(input("Enter Rate:"))


if  xh > 40

    pay= (xh-40)*xr*1.5+40*xr
else :

    pay= xr*xh
print ("Pay=", pay)
```

EXP 8



Ask user to enter a number. Write a program which repeatedly reads numbers until the user enters "done".

Once "done" is entered, print out the total, count, and average of the numbers.

```
count = 0

tot = 0

while True:

    sval = input("Enter num, if done write done")

    if  sval == "done":

        break

    fval = float(sval)

    print (fval)

    count = count +1

    tot = tot + fval


average = tot/count
```

Ben durdurana kadar çalışmaya devam ediyor.

hem sayı hem string gelebilir.

Bunlar while'ın içinde çünkü her sayı girilişinde her adımda güncellensin istiyorum

→ bu while loop'un içinde değil en son hesaplansın istiyorum.

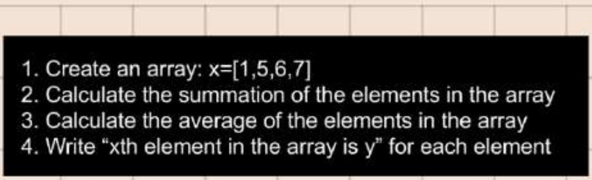
!! ⑪ While döngüsü → Altına yazdığımız koşul doğru old. sürece çalışır.

While True döngüsü → Sürekli çalışır, bir kesme koşulu sağlanana kadar devam eder.

n>0 old. sürece çalıştır diyorum

"done" yazılana kadar sürekli çalıştır

Expe



1. Create an array: x=[1,5,6,7]
2. Calculate the summation of the elements in the array
3. Calculate the average of the elements in the array
4. Write “xth element in the array is y” for each element

* 0. element "1"dir. → List'lerde böyle!

```
x= [1,5,6,7]
print (sum(x))
print (len(x))              → x'te kaç eleman varse → "4"
print(sum(x) / len(x))
for i in range (len(x)):    → 0,1,2,3 yazar
                              i bunları alabilir
        print(i, "th element of the list is:", x[i])
              i+1 dersin
```

4

x[0] = 1
x[1] = 5
⋮

x'in 0. elementi olarak

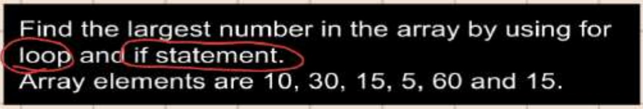
bunu algılıyor normalde zaten sistem

Output

```
19
4
4.75
0 th element of the list is: 1
        ⋮
```

eğer 1'den başlasın istersen

ÖRe



Find the largest number in the array by using for loop and if statement.
Array elements are 10, 30, 15, 5, 60 and 15.

```
x= [10,30,15,5,60,15]
largestnum = 0     → Bunu tanımladım önce
for i in x:
    if i > largestnum:
        largestnum = i

print (largestnum)
```

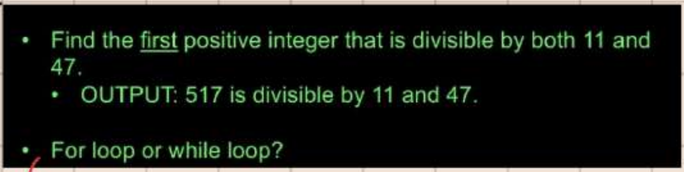
While; koşul doğru old. sürece devam eder.

For; liste, demet gibi veri yapılarının içindeki değerleri kullanarak işlem yapılmak isteniyorsa.

"Döngüdeki iterasyon sayısı biliniyor"

- WEEK 2 -

ÖR:



- Find the first positive integer that is divisible by both 11 and 47.
  - OUTPUT: 517 is divisible by 11 and 47.
- For loop or while loop?

↳ we don't know where to exit the loop, so while loop is more appropriate

1.Option

```
x=1
while True:
    if x%11 ==0 and x % 47 ==0 :
        break  → (Eğer üstteki koşulu sağladıysa x break oluyor Loop ve direk
                        print satırına geçiyor.)
    x= x+1  → break değilse "1" arttırıp tekrar check ediyor. → while loop içinde çünkü
print( x, "is the 1st num which is divisible both 11 and 47")
```

2.Option

```
x=1
while x % 11 != 0 or x % 47 != 0 :
    x= x+1

print(x, 'is divisible by 11 and 47')
```

3. Option

```
for x in range(1,600)  → Aralığı sınırladık sayılar çok büyük olm. için
    if x%11 ==0 and x % 47 ==0 :
        print(x, 'is divisible by 11 and 47')
        break
```

⊛ Functions

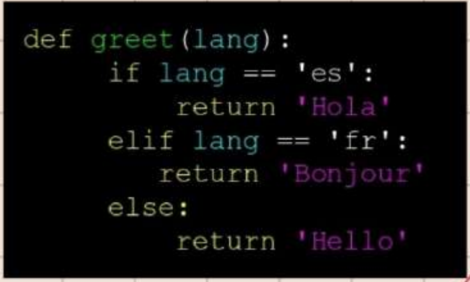
- random. randint ()
- type()
- input
- len()
- max()

→ len('Hello world') → boşluğu da sayar o yüzden cevap=11

→ Max('Hello world') = w

→ Min('Hello world') = d

ÖR:



```python
def greet(lang):
    if lang == 'es':
        return 'Hola'
    elif lang == 'fr':
        return 'Bonjour'
    else:
        return 'Hello'
```

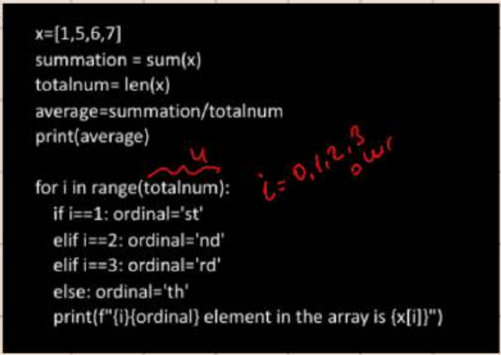
fonk'u geri çağırdık

```
print(greet('en'), 'Glenn')
print(greet('es'), 'Sally')
print(greet('fr'), 'Michael')
```

Output

```
Hello Gleen
Hola Sally
Bonjour Michael
```

ÖR:



```python
x=[1,5,6,7]
summation = sum(x)
totalnum= len(x)
average=summation/totalnum
print(average)

for i in range(totalnum):
  if i==1: ordinal='st'
  elif i==2: ordinal='nd'
  elif i==3: ordinal='rd'
  else: ordinal='th'
  print(f"{i}{ordinal} element in the array is {x[i]}")
```

4

i=0,1,2,3 olur

Output

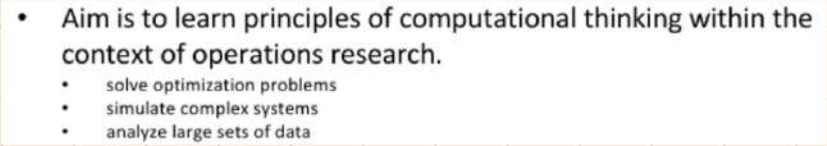
```
6.25
0th elem. in array is 1
1st elem. in array is 5
  ⋮
```

→ Func'la yazıcaz.

```
def myfun(x):
    summation = sum(x)
    totalnum = len(x)
    average = summation/totalnum
    print (average)

    for i in range (totalnum):
        if i==1: ordinal= 'st'
        elif i==2: ordinal= 'nd'
        elif i==3: ordinal = 'rd'
        else: ordinal= 'th'
        print(f{i}{ordinal} elem. in array is {x[i]})

y=[1,5,6,7,8,9,6,8]
myfun(y)
```

→ fonk'mu çağırdım!

→ Optimization Problems (Algorithm)

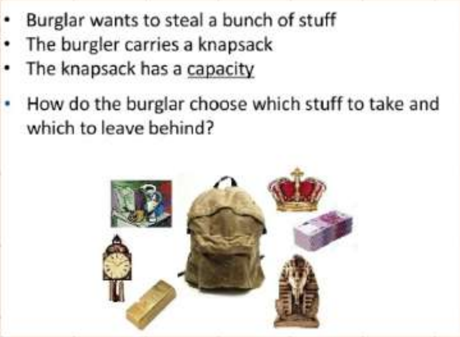


- Aim is to learn principles of computational thinking within the context of operations research.
  - solve optimization problems
  - simulate complex systems
  - analyze large sets of data

Optimization Model

Mathematical Analysis → Obj Func. / → Dec.var / → Constraints

⇒ Knapsack Problem

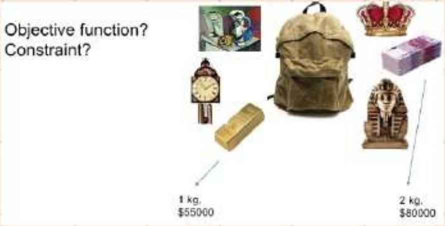


- Burglar wants to steal a bunch of stuff
- The burgler carries a knapsack
- The knapsack has a capacity
- How do the burglar choose which stuff to take and which to leave behind?

Constraints :

Total capacity aşılamaz (can't exced)

Different types



Objective function?
Constraint?

↳ Value = price...

↳ weight = kg

↳ Value = Calorie, level of protein, happiness level...

↳ weight =

has has

* Each item is represented by a pair < value, pair >

* The knapsack can accommodate items with a total weight of no more than "W"

* A vector, L, of length n, represents the set of available items. Each element of the vector is an item.

* A vector, V, of length n, is used to indicate whether or not items are taken

If V[i] = 1 , item is taken

If V[i] = 0 , item is not taken

V that maximizes:

$$\boxed{\sum_{i=0}^{n-1} V[i] * I[i].\text{Value}}$$

Subject to the constraint:

$$\boxed{\sum_{i=0}^{n-1} V[i] * I[i].\text{weight} \leq W}$$

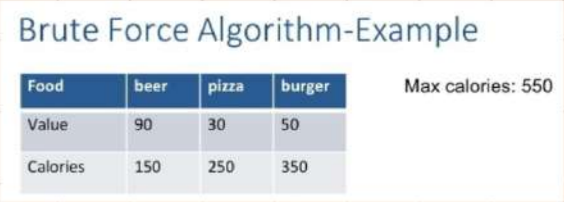
(*) All possible combinations of items ⇒ Power Set

① Bütün kombinasyon alternatiflerini bul.

② Capacity'i aşanları ele!

③ Total value'su MAX↑ olanı seç.

Ör:



Brute Force Algorithm-Example

Food	beer	pizza	burger
Value	90	30	50
Calories	150	250	350

Max calories: 550

① we have 3 items, all combinations:

beer	Pizza	Burger
0	0	0
1	1	1

her biri için 2 alternatif var.

1) take or 2) not

3 items we have

$2^3 = 8$ different alternatives we have

↓

* hiçbirini almayodabilirim

	Calorie	Value
Hiçbiri	0	0
Pizza	250	30
Beer	150	90
Burger	350	50
Pizza + Beer	400	120
Beer + Burger	500	140
~~Burger + Pizza~~	~~600~~	80
~~All ones~~	~~750~~	170

Calorie 550'yi geçenler elenir.

Geri kalan 6 alternatiften Value Max olan seçilir.

"power set = 8"

- Mathematical Model :

Dec. Var (Binary variables)

x_1 → 1, if we select burger
0, otherwise

x_2 → 1, if -- Pizza
0, otherwise

x_3 → 1, if -- Beer
0, otherwise

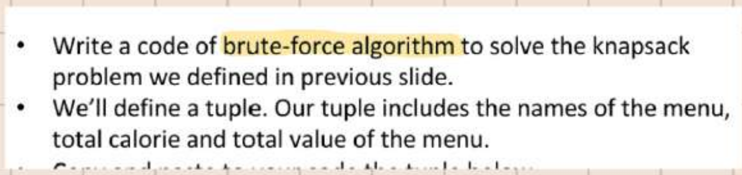
Obj. Func.

$$\text{Max} = x_1 \cdot 30 + x_3 \cdot 90 + x_2 \cdot 50$$

Constraint

$$350x_1 + 250x_2 + 150x_3 \leq 550$$

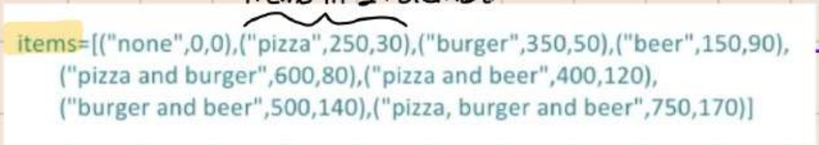
ÖR:



- Write a code of brute-force algorithm to solve the knapsack problem we defined in previous slide.
- We'll define a tuple. Our tuple includes the names of the menu, total calorie and total value of the menu.
- Copy and paste to your code the tuple below.

kod :

items'in 1. elemanı



```
items=[("none",0,0),("pizza",250,30),("burger",350,50),("beer",150,90),
    ("pizza and burger",600,80),("pizza and beer",400,120),
    ("burger and beer",500,140),("pizza, burger and beer",750,170)]
```

Sadece bu yazılıyken :

*print(items[0])

↳ ('none', 0, 0)

*print(items[0][0]) — items'in 0. elemanının 0. elemanı

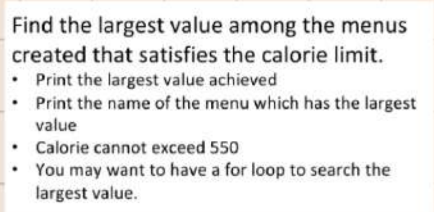
↳ none

*print(items[1][0])

↳ pizza

O_o

ÖR:



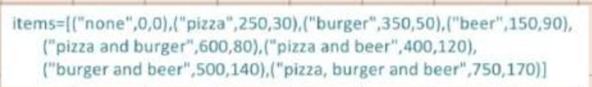
Find the largest value among the menus created that satisfies the calorie limit.

- Print the largest value achieved
- Print the name of the menu which has the largest value
- Calorie cannot exceed 550
- You may want to have a for loop to search the largest value.

output :

largest value is 140 by picking burger and beer

kod :



```
items=[("none",0,0),("pizza",250,30),("burger",350,50),("beer",150,90),
    ("pizza and burger",600,80),("pizza and beer",400,120),
    ("burger and beer",500,140),("pizza, burger and beer",750,170)]
```

```
def knapsack (items, cal):
    largestVal = 0
    for i in range (len(items)):
        if items [i][1] < cal:
            if items [i][2] > largestVal:
                largestVal = items [i][2]
                pickeditems = items [i][0]
    print("largest value is", largestVal, "by picking", pickeditems)
knapsack(items, 550)
```

iteration sayımın "8" olucağını biliyorum o yüzden "for"

her grubun 1. elemanı calorie değeri çünkü

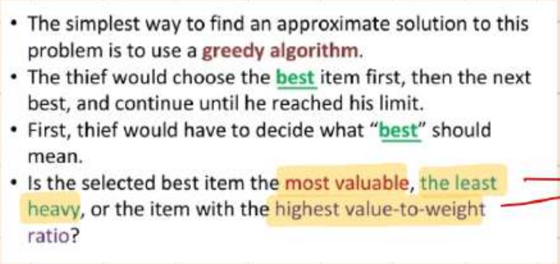
→ Bu koşul sağlanıyorsa largest Val. check edilebil

bunu 1000 yapsaydık

↳ largest value is 170 by picking pizza, burger and beer

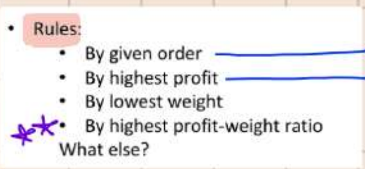
- WEEK 3 -

→ Greedy Algorithm



- The simplest way to find an approximate solution to this problem is to use a **greedy algorithm**.
- The thief would choose the **best** item first, then the next best, and continue until he reached his limit.
- First, thief would have to decide what "**best**" should mean.
- Is the selected best item the most valuable, the least heavy, or the item with the highest value-to-weight ratio?

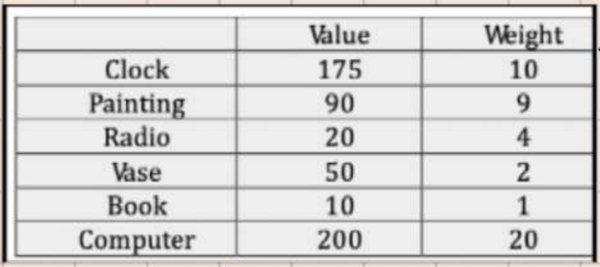
Bu 3 "best way" seçeneğinden bizim case'imiz için uygun olanı kullanacağız.



- Rules:
  - By given order → direk sorunun verdiği order'la
  - By highest profit → Highest value'dan başlayarak sıralarız o sırayla alabiliyorsak almaya başlarız
  - By lowest weight
  - ** By highest profit-weight ratio

  What else?

• By Giving Order (Random order)

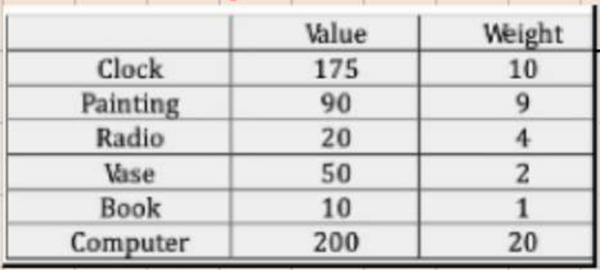


	Value	Weight	V/w	Taken or not	Remaining capacity	Total value
Clock	175	10		1	10	
Painting	90	9		1	1	275
Radio	20	4		0	1	
Vase	50	2		0	1	
Book	10	1		1	0	
Computer	200	20		0	0	

(Capacity = 20 kg)

yukarıdan aşağıya bu sıralamayla gidip alıp alamayacağına bakıyorsun

• Value / weight Ratio



	Value	Weight	V/w
Clock	175	10	10
Painting	90	9	17,5 → next valuable (2)
Radio	20	4	10
Vase	50	2	25 → most valuable (1)
Book	10	1	5
Computer	200	20	10

kapasitem kalıyorsa valuableları en büyük olodan başlayarak alırım

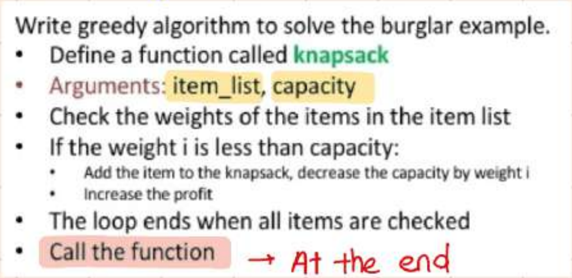
	Value	Weight	V/w	Taken or not	Remaining Capacity	Total value
Vase	50	2	25	1	18	
Clock	175	10	17,5	1	8	255
Computer	200	20	10	0	8	
Painting	90	9	10	0	8	
Book	10	1	10	1	7	
Radio	20	4	5	1	3	

most valuable to less

** Ratio'ya göre büyükten küçüğe olacak şekilde yeni order yapıldı

- By Lowest Weight → Lowest to highest şeklinde weight'leri sıralarsın. Capacity'in bitene kadar alırsın.

- By giving Order kod Çözümü



Write greedy algorithm to solve the burglar example.
- Define a function called **knapsack**
- Arguments: item_list, capacity
- Check the weights of the items in the item list
- If the weight i is less than capacity:
  - Add the item to the knapsack, decrease the capacity by weight i
  - Increase the profit
- The loop ends when all items are checked
- Call the function → At the end

capacity

```
def knapsack_greedy (item_list, cap)
    remaining_cap = cap
    total_profit = 0

    for item in item_list :
**      name, profit, weight = item
        if weight <= remaining_cap :
            print(name, "is taken")
            remaining_cap -= weight
            total_profit += profit
    print("Total profit is", total_profit)
```

→ item'in içindeki 3'lüye isimlerini verdik, kod algılıyo

→ print(name), dersem:
→ clock
painting
radio
⋮

Output:

```
clock is taken
painting is taken
book is taken
Total profit is 275
```

```
C=20
item_list = [
    ("clock", 175, 90),
    ("painting, 90, 9),
   ("radio", 20, 4),
    ("vase", 50, 2),
    ("book", 10, 1),
    ("computer", 200, 20),
]
knapsack greedy (item_list, C)
```

→ capacity'e bir değer atadım

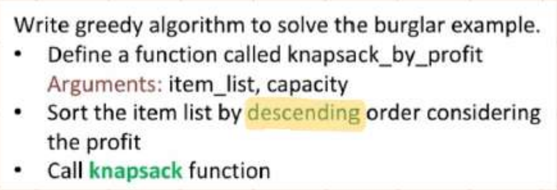
→ Fonk'u çağırmak sadece for döngüsünü çalıştırdı outputta hiçbir şeyi print etmiyor!

• By Profit kod Çözümü

true → descending order

key lambda x : x[...]

old gibi yazabilirsin değiştirmeden



Write greedy algorithm to solve the burglar example.

- Define a function called knapsack_by_profit
  Arguments: item_list, capacity
- Sort the item list by descending order considering the profit
- Call **knapsack** function

```
def knapsack (item_list, cap):
        remaming-cap = cap
        total_profit = 0

        for item in item_list:
            item = name, profit, weight
                if weight < remaming_cap
                    print(name, "is taken")
                    remaming_cap -= weight
                    total_profit += profit
                print("total profit is", total_profit)

def knapsack_by-profit (item_list, cal):
        for item in item_list:
            sorted_items = (item_list, key=lambda x: x[0], reverse=True)
```

```
def knapsack_greedy (item_list, cap):
        remaining_cap = cap
        total_profit = 0

        for item in item_list:
            name, profit, weight = item
            if weight <= remaining_cap:
                print(name, "is taken")
                remaining_cap -= weight
                total_profit += profit
        print("Total profit is", total_profit)

def knapsack_by-profit (item_list, capacity):
        sorted_items = sorted (item_list, key=lambda x: x[1], reverse=True)
        knapsack_greedy (sorted_items, capacity)
```

yeni ekledi

profit'e göre koşulu için yeni fonk. tanımladım. (soruda da söylüyer)

neyin değerinin büyüklüğüne göre sıralanacak istiyorsan onun indexi "profit"

True dersen "Azalan sırayla"
False " "Artan sırayla" yazar.

→ Farklı bir fonk. içinde önceki fonk'n knapsack_greedy'i çağırdık

↳ benim yeni item_list'im sorted_item old. için

yukarıdaki uygulanalar sorted_list'le yapılsın istediğim için üstteki fonk'u çağırıp içine "sorted_items" yazdım.

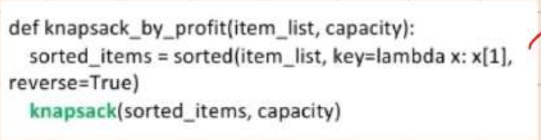
```
C=20
item_list = [
    ("clock", 175, 90),
    ("painting", 90, 9),
    ("radio", 20, 4),
    ("vase", 50, 2),
    ("book", 10, 1),
    ("computer", 200, 20),
]

knapsack_by_profit (item_list, C)
```

→ en sonda da 2. fonk'mu çağırdım

Output:

```
computer is taken
Total profit is 200
```

- By Weight'e göre kod Çözümü



```
def knapsack_by_profit(item_list, capacity):
   sorted_items = sorted(item_list, key=lambda x: x[1],
reverse=True)
   knapsack(sorted_items, capacity)
```

üstteki kodun içinde sadece bu kısım değişecek

```
def knapsack_by_weight (item_list, capacity):
    sorted_items = sorted(item_list, key=lambda x: x[2])
    knapsack_greedy (sorted_items, capacity)
        ⋮
```

2 dedim çünkü weight'le bakıcak

Ascending (artan) istediği için zaten, "reverse" kullanmaya gerek yok.

yukarıdaki büyük kodun en sonunda bu sefer,

knapsack_by_weight (item_list, C) ⇒ Bu fonk'u çağır

Output:

```
book is taken
vase is taken
radio is taken
painting is taken
Total profit is 170
```

- By Profit - Weight Ratio kod Çözümü:

```
def knapsack_by_profit_per_weight (item_list, capacity):
    sorted_items = Sorted(item_list, key=lambda x: x[1] / x[2], reverse = True)
    knapsack_greedy (sorted_items, capacity)
        ⋮
```

buna göre sort et (sırala) diyor yani item'ları

knapsack_by_profit_per_weight (item_list, C) → Çağırdım

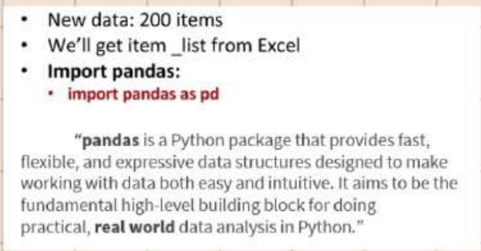
Output:

```
Vase is taken
Clock is taken
book is taken
radio is taken
Total profit is 255
```

Optimal → By giving order ile bulduğumuz çıktı (şansa)

Generally, with larger data sets, "profit-weight ratio" is the winner.

Büyük Veriler İçin



- New data: 200 items
- We'll get item _list from Excel
- **Import pandas:**
  - **import pandas as pd**

"**pandas** is a Python package that provides fast, flexible, and expressive data structures designed to make working with data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python."

* Verilerin old. excel dosyasını ve spyder kodunu (dosyasını) aynı yere kaydet önce (masaüstü vs...). Sonra koda excel'deki verileri uyarlayabilirsin.

pandas

* pd read_excel

Excel dosyasını okuyor

```
✓ import pandas as pd

# Read Excel File
✓ df = pd.read_excel("kpData.xlsx")

# Convert Data Frame into tuple list

✓ item_list = list(df.itertuples(index=False, name=None))
```

şu ismi sen verebilirsin

Excel file'in ismi

Excel file

EZBERLE!

Excel'deki verileri tuple list haline getirir üst örneklerdeki listlerdeki gibi.

kod:

```
import pandas as pd
     ⋮
```

* (üstteki örneklerdeki kodun list'e kadar olan kısmını al buraya yapıştır.)

```
df = pd.read_excel("kpData.xlsx")

item_list = list(df.itertuples(index=False, name=None))

knapsack_by_profit(item_list, 2250)
```

→ üsttekilerin arasından bu fonk'u çağırdık!

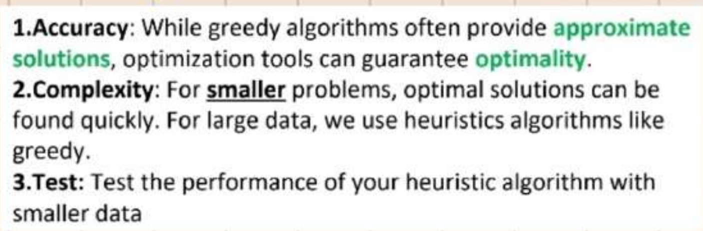
Output:

```
a is taken
b is taken
  ⋮
```

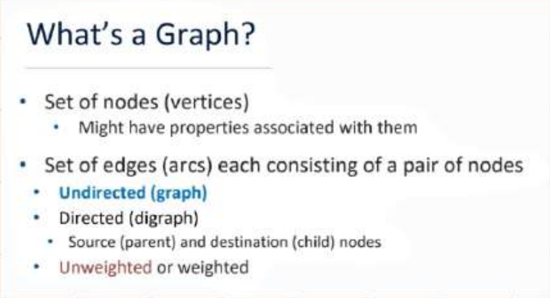
yüksek profitten başlayarak alabildiğini alıyor.

- WEEK 4 -

→ Greedy Algorithm ; → large data setlerde de iyi



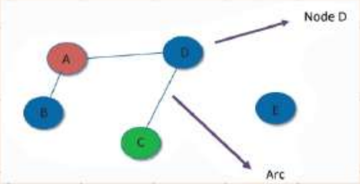
1.Accuracy: While greedy algorithms often provide **approximate solutions**, optimization tools can guarantee **optimality**.
2.Complexity: For **smaller** problems, optimal solutions can be found quickly. For large data, we use heuristics algorithms like greedy.
3.Test: Test the performance of your heuristic algorithm with smaller data



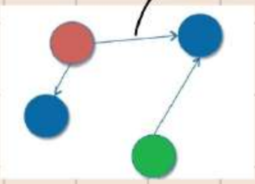
What's a Graph?

- Set of nodes (vertices)
  - Might have properties associated with them
- Set of edges (arcs) each consisting of a pair of nodes
  - **Undirected (graph)**
  - Directed (digraph)
    - Source (parent) and destination (child) nodes
  - Unweighted or weighted

Graphs (Shortest Paths ...)



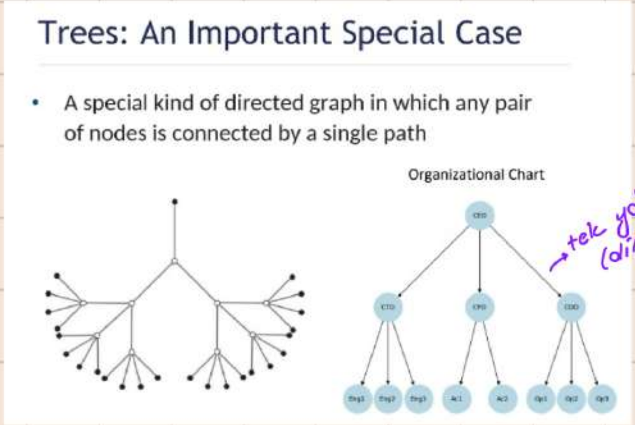
* Undirected graph



* Directed graph

→ arcların üzerinde yazan herhangi bir sayı (km, cost, time, aver. speed ....) "weight" olarak adlandırılır.

!!
①

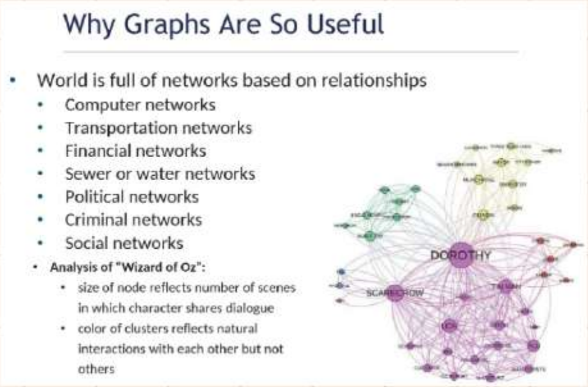


Trees: An Important Special Case

- A special kind of directed graph in which any pair of nodes is connected by a single path

→ tek yönlü (directed)

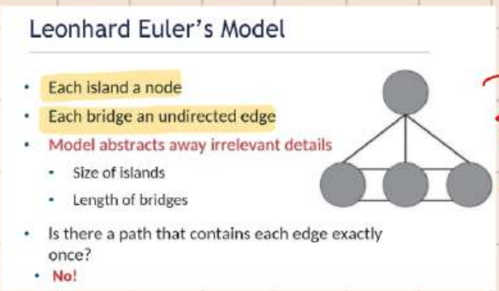
* why graphs are so useful ?



Why Graphs Are So Useful

- World is full of networks based on relationships
  - Computer networks
  - Transportation networks
  - Financial networks
  - Sewer or water networks
  - Political networks
  - Criminal networks
  - Social networks
  - **Analysis of "Wizard of Oz":**
    - size of node reflects number of scenes in which character shares dialogue
    - color of clusters reflects natural interactions with each other but not others

⊛



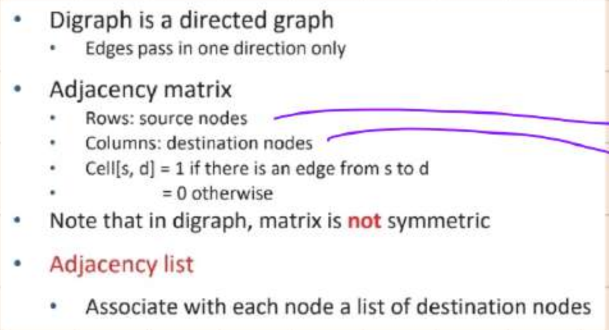
Leonhard Euler's Model

- Each island a node
- Each bridge an undirected edge
- Model abstracts away irrelevant details
  - Size of islands
  - Length of bridges
- Is there a path that contains each edge exactly once?
  - **No!**

} path "bu şekilde daha net"

Shortest Path Problems

- shortest path from n1 to n2
- Shortest sequence of edges such that
    - Source node of first edge is n1
    - Destination of last edge is n2
    - For edges, e1 and e2, in the sequence, if e2 follows e1 in the sequence, the source of e2 is the destination of e1
- Shortest weighted path
    - Minimize the sum of the weights of the edges in the path



- Digraph is a directed graph
    - Edges pass in one direction only
- Adjacency matrix
    - Rows: source nodes
    - Columns: destination nodes
    - Cell[s, d] = 1 if there is an edge from s to d
    -                = 0 otherwise
- Note that in digraph, matrix is **not** symmetric
- Adjacency list
    - Associate with each node a list of destination nodes

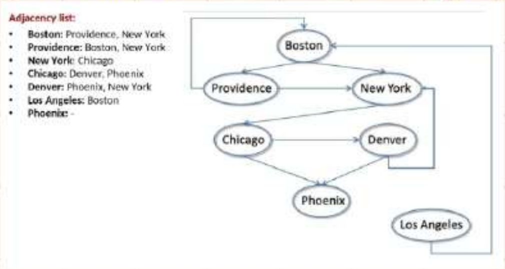
$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & & & 1 & 0 \\ 2 & & & & \\ 3 & & & & \\ 4 & & & & \end{array}$$

→source nodes

destination nodes

2.3 nodeları arasını seçmiş✓

Exp: "Depth-First Search" Algorithm:

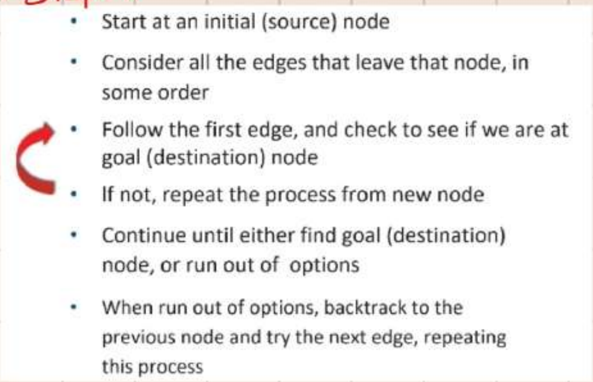


Adjacency list:

- Boston: Providence, New York
- Providence: Boston, New York
- New York: Chicago
- Chicago: Denver, Phoenix
- Denver: Phoenix, New York
- Los Angeles: Boston
- Phoenix: -

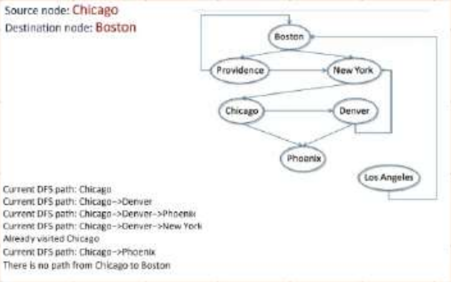
* Graph might have cycles, so we must keep track of the nodes we have visited, to avoid going in infinite loops.

* Steps:



- Start at an initial (source) node
- Consider all the edges that leave that node, in some order
- Follow the first edge, and check to see if we are at goal (destination) node
- If not, repeat the process from new node
- Continue until either find goal (destination) node, or run out of options
- When run out of options, backtrack to the previous node and try the next edge, repeating this process

Exp:

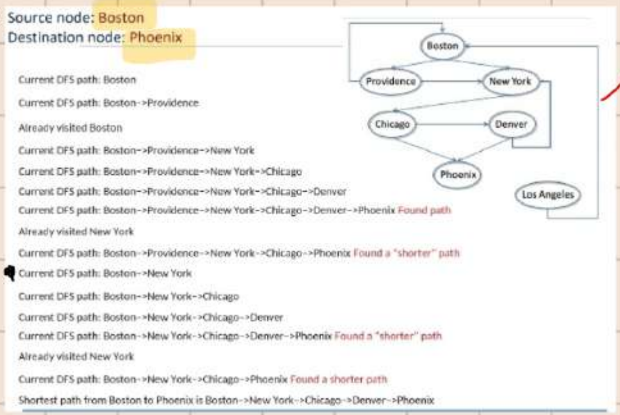


Source node: Chicago

Destination node: Boston

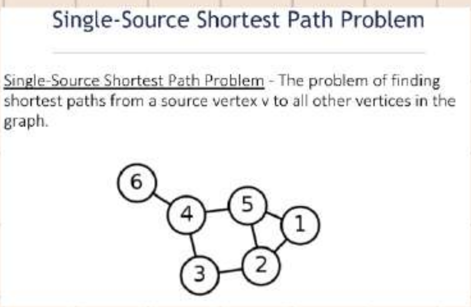
```
Current DFS path: Chicago
Current DFS path: Chicago->Denver
Current DFS path: Chicago->Denver->Phoenix
Current DFS path: Chicago->Denver->New York
Already visited Chicago
Current DFS path: Chicago->Phoenix
There is no path from Chicago to Boston
```

Exp:


Source node: Boston
Destination node: Phoenix

Current DFS path: Boston
Current DFS path: Boston->Providence
Already visited Boston
Current DFS path: Boston->Providence->New York
Current DFS path: Boston->Providence->New York->Chicago
Current DFS path: Boston->Providence->New York->Chicago->Denver
Current DFS path: Boston->Providence->New York->Chicago->Denver->Phoenix Found path
Already visited New York
Current DFS path: Boston->Providence->New York->Chicago->Phoenix Found a "shorter" path
Current DFS path: Boston->New York
Current DFS path: Boston->New York->Chicago
Current DFS path: Boston->New York->Chicago->Denver
Current DFS path: Boston->New York->Chicago->Denver->Phoenix Found a "shorter" path
Already visited New York
Current DFS path: Boston->New York->Chicago->Phoenix Found a shorter path
Shortest path from Boston to Phoenix is Boston->New York->Chicago->Denver->Phoenix

if these arcs are weighted, we want to minimize the sum of weights.


Single-Source Shortest Path Problem

Single-Source Shortest Path Problem - The problem of finding shortest paths from a source vertex v to all other vertices in the graph.

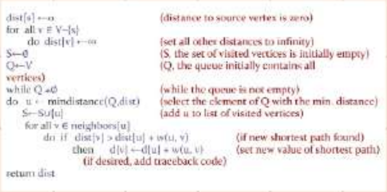
- Dijkstra's Algorithm -

is a solution to the single-source shortest path problem in graph theory.

* Works on both directed and undirected graphs. However, all edges must have nonnegative weights.

Approach = Greedy Algorithm

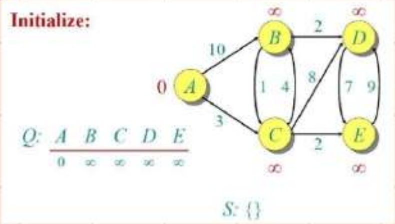
- Input = Weighted graph G= {E,V} and source vertex v ∈ V, such that all edge weights are nonnegative.
- Output = Lengths of shortest paths (or the shortest paths themselves) from a given source vertex v∈V to all other vertices.



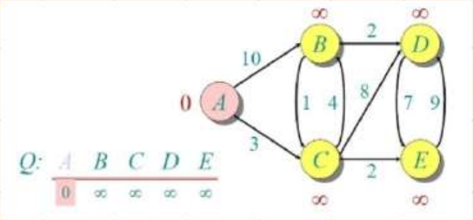
```
dist[s] ←0                          (distance to source vertex is zero)
for all v ∈ V–{s}
    do dist[v] ←∞                   (set all other distances to infinity)
S←∅                                 (S, the set of visited vertices is initially empty)
Q←V                                 (Q, the queue initially contains all
vertices)
while Q ≠∅                          (while the queue is not empty)
do  u ← mindistance(Q,dist)         (select the element of Q with the min. distance)
    S←S∪{u}                         (add u to list of visited vertices)
    for all v ∈ neighbors[u]
        do if  dist[v] > dist[u] + w(u, v)        (if new shortest path found)
              then   d[v] ←d[u] + w(u, v)        (set new value of shortest path)
              (if desired, add traceback code)
return dist
```

Dijkstra Animated Example;

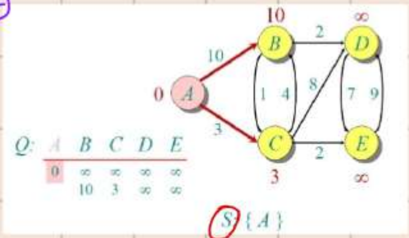
1-



2-

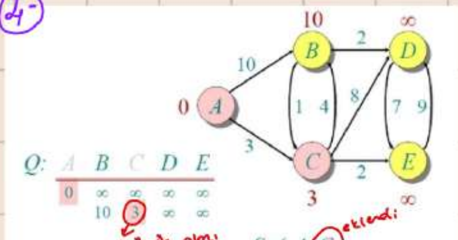


3-



set of visited vertices

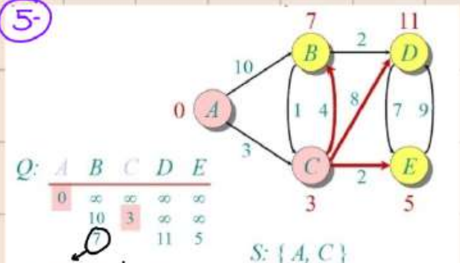
4-



küçük olanı seçerim (shortest-path) arıyorum

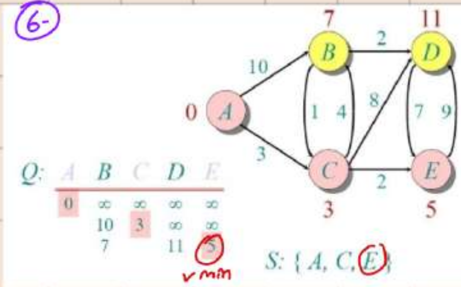
eklendi

5-



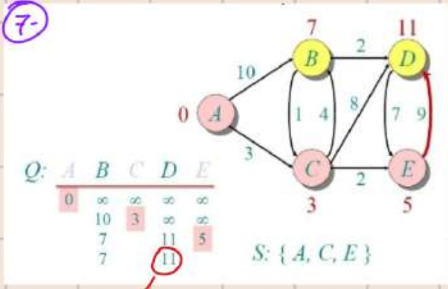
B'ye C üzerinden 7 km'yle gidilebiliyormuş.

6-



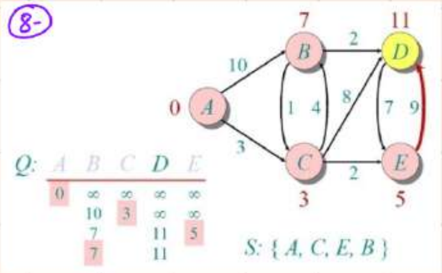
v min

7-



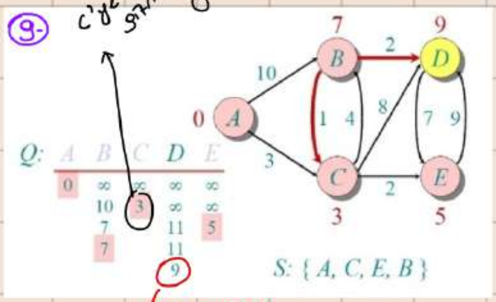
14 diye arttırmadım çünkü önceki adımda daha kısa gidiş yolunu bulmuşum "11".

8-



→ D mantıksız oldu o zmn B'ye hiç gitmedik oraya da bakalım

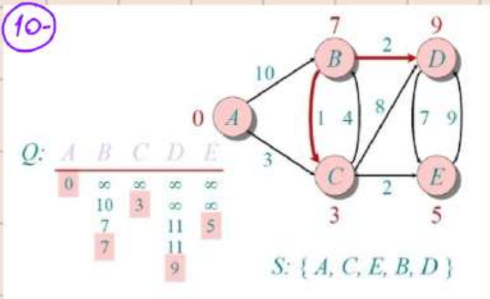
9- C'ye B üzerinden gitmek mantıksız yol uzuyor



A→C→B→D

D'ye B üzerinden gitmek en mantıklı oldu

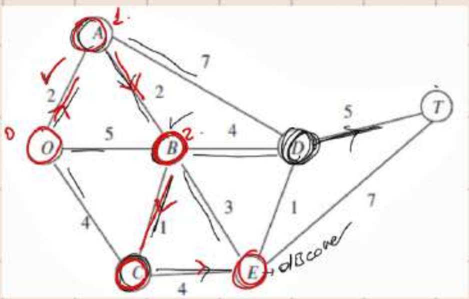
10-



**

Exp: Find the shortest path between the nodes O-T

?



O	A	B	C	D	E	T
0	∞	∞	∞	∞	∞	∞
0	2 ✓	5	4	∞	∞	∞
0	2	4	4	9	∞	∞
0	2	4	4	8	7	∞
0	2	4	4 ✓	8 ✓	7 ✓	∞
0	2	4	4	8	7	14
						13

"5" olarak güncellemedik çünkü hâli hazırdaki hâli 5'ten küçük

Mesela O'dan D'ye en kısa yol "8" miş.

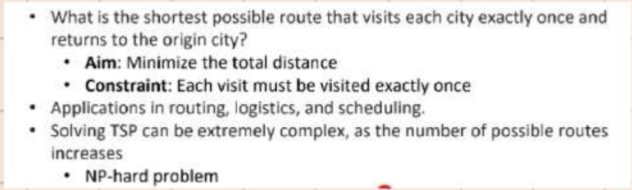
S = {O, A, B, C, E, D}

Solution: O-A-B-E-D-F

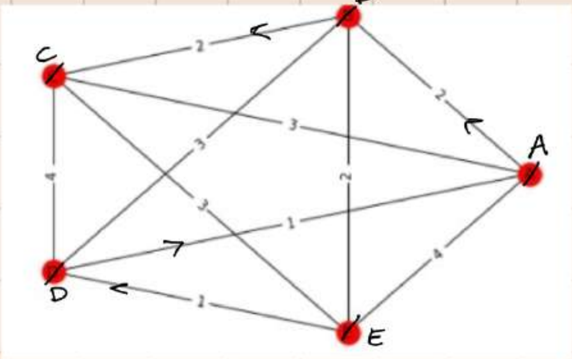
$\boxed{\text{Toplam maaliyet} = 13}$

- Travelling Salesman Problem (TSP) -

Exp:

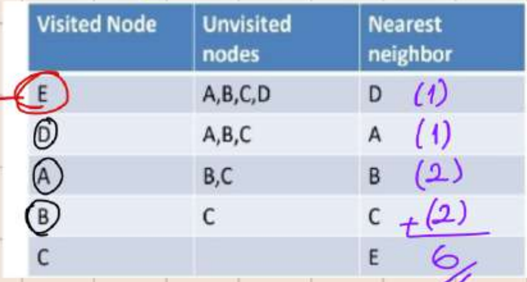


- What is the shortest possible route that visits each city exactly once and returns to the origin city?
  - **Aim**: Minimize the total distance
  - **Constraint**: Each visit must be visited exactly once
- Applications in routing, logistics, and scheduling.
- Solving TSP can be extremely complex, as the number of possible routes increases
  - NP-hard problem

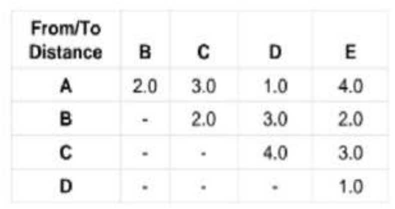


Anladım

Bunu başlangıçta random seçtik

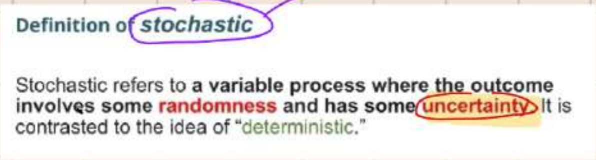


Visited Node	Unvisited nodes	Nearest neighbor
E	A,B,C,D	D (1)
D	A,B,C	A (1)
A	B,C	B (2)
B	C	C +(2)
C		E 6



From/To Distance	B	C	D	E
A	2.0	3.0	1.0	4.0
B	-	2.0	3.0	2.0
C	-	-	4.0	3.0
D	-	-	-	1.0

- WEEK 5 -



randomness

Definition of *stochastic*

Stochastic refers to **a variable process where the outcome involves some randomness and has some uncertainty.** It is contrasted to the idea of "deterministic."

* So far, we've seen "deterministic" optimization problems:

- Knapsack Problem
- Graphs

deterministic × stochastik

sonuç sabittir

↳ sonuç olasılığa bağlıdır ve rastgele değişkenlik gösterir.

* import → içe aktarmak

→ Random Module in Python - Examples:

random module — generates number between 0-1 → 0-1 arası herhangi bir sayı print eder

1) print(random.random()) = x=random.random()
print(x)

→ random numbers between 1-100 (integer) — 1 ve 100 dahil!

2) x= random.randint(1,100)

sadece sayı

list

3) print(random.choice[1,2,3]) → 1,2,3'ten birini random yazdırıyor

liste!

=
```
y=[1,2,3]
x= random.choice(y)
print(x)
```

4)
```
y= [1,2,6,7,8,5]
x=random.choices(y, k=5)
print("five numbers from a list:", x)
```

→ 5 tane seçer listeden

→ five numbers from a list: [6,1,6,2,5]

aynı seçebiliyor

5)
```
items= ['Alissa', 'Alice', 'Marco', 'Melissa']
x= random.sample(items, k=2)
print(x)
```

↳ 2 tane istedim listeden

→ ['Alissa', 'Marco']

Aynı şeyden yazdırmıyor!

* 2 kere para atılıyor :

Power set
- HH → 1/4
- TT → 1/4
- HT, TH → $\frac{1}{2}$ → Bunun olasılığı daha yüksek

ÖR : -Zar Atma-

```
import random
def rollDie():
    return random.choice([1,2,3,4,5,6])
print(rollDie())
```

liste verirken

=

```
import randolm
def rolldie():
    x= random.rondint(1,6)
    return x
print(rollDie())
```

bununla da yazabilirim

→ return demeden yazdırırsan "x" değerini hiçbir yerde kullanmadınız diye uyarı veriyor.

→ fonk'nu çağırdığımda return dediğim ifadeyi çağırmış olurum aslında

Ör : Zar 5 kere atılıyor :

```
import rodom
def rollDie(n):
    die = [1,2,3,4,5,6]
    result = random.choices(die, k=n)
    return result
x=rollDie(5)
print(x)
```

→ x'i aşağıda tekrar bir işlemde kullanmak istersen "return" kullanabilirsin.

→ Return yazmadan sadece print dersen ekranda görürsün. Bu da okey bu exp. için.

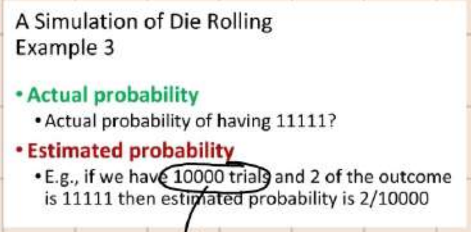
→ [2,6,5,6,5] > $\left(\frac{1}{6}\right)^5$ → probability

* Yukarıdaki deneyde Power set → 6^5 (total num. of combinations)

→ her zar atımında 6 olasılık var, 5 kere atıyorum zarı.

Devam!

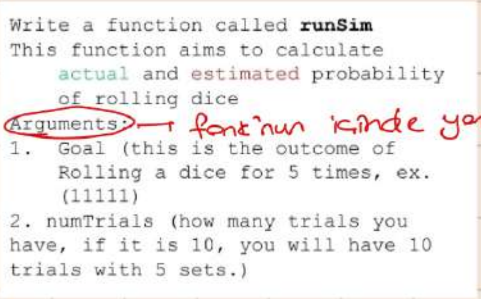
ÖR:



A Simulation of Die Rolling
Example 3

- **Actual probability**
  - Actual probability of having 11111?
- **Estimated probability**
  - E.g., if we have 10000 trials and 2 of the outcome is 11111 then estimated probability is 2/10000

fonk bu kadar kez çağrılır



```
Write a function called runSim
This function aims to calculate
    actual and estimated probability
    of rolling dice
Arguments:
1.  Goal (this is the outcome of
    Rolling a dice for 5 times, ex.
    (11111)
2. numTrials (how many trials you
have, if it is 10, you will have 10
trials with 5 sets.)
```

→ fonk'nun içinde yazacak parametreler

#num. of trials = 100 olsun

```
def runSim (goal, numTrials):
    total = 0
    for i in range (numTrials):
        x = rollDie (len(goal))
        if x == goal:
            total += 1
    estProb = total / numTrials
    return estProb
```

$\underbrace{len(goal)}_{5}$

!! Tanımlanan x'ler aynı "x" gibi gözükse de farklı bloklarda tanımlandıkları için birbirlerinden bağımsızlardır.

estProb = total / numTrials → Direk probability hesapladı

!! # of trials arttırıldıkça estimated, actual'a yaklaşır!

```
goal = [1, 1, 1, 1, 1]
x = runSim (goal, 100)
print ("Estimated prob:", x)
print ("Actual prob:", (1/6) ** len(goal))
```

(1/6) → 11111

1'in gelme olasılığı $\frac{1}{6}$, 5 tane 1 gelme olasılığı ⇒ $\left(\frac{1}{6}\right)^5$

	B	C	D	E
	10	3	∞	∞
C	7		11	5
E	7		11	
B			9	

kimsenin aynı değil durumu → $\frac{365}{(365-23)!}$ kombinasyon sayıdı → a

$\frac{a}{b}$ → kimse aynısını paylaşmıyor

$\boxed{1-\frac{a}{b}}$

* $\frac{1}{365}$ → bir kişinin dgsı

* en az 2 kişinin dg'si aynı

b ← 365^{23} → tüm kombinasyonlar

365 → bir kişinin olasılığı

365 günden random seçicem herkese!
→ bunlardan en az 2'si eşit çıkarsa ✓

0 → no one has a birthday day 1, day2 ...

⇒ Birthday Example ⚠ 23 kişide en az 2 kişinin dg'sinin aynı gün olma olasılığı

```
import random
def sameDate(numPeople, numSame):
        birthdays = [0] * 365   → initially (365 tane yan yana "0" varmış gibi düşün) (Başta 365 günden herhangi birinde doğan yok.)
       for i in range(numPeople):          } 23 kere (1,365) arasından
            birthdate = random.randint(0,364)  } random sayı seçiliyor
                                           → herkese bir dg atıyor (Bütün günlerin seçilme olasılığı eşit kabul edilir)
            birthdays[birthdate] + = 1   → Başlangıçta "0" olan birthday'i seçilen her günde, o günü "1" arttırıyor.
            return max(birthdays) >= numSame   → Bu koşul sağlanıyorsa "True" döndürür.
print(sameDate(23,2))
```

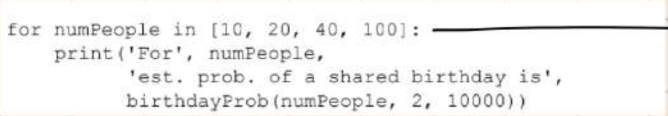
```
→ False   } run ettikçe böyle
→ True    }    cevaplar döndürür
   ⋮
```

→ Kodun Devamı :

```
def birthdayProb(numPeople, numSame, numTrials):
      total = 0
      for t in range(numTrials):
           if sameData(numPeople, numSame) == True:
                total += 1
      return total / numTrials   ⇒ Estimated-Sample Probability
print(birthdayProb(23, 2, 100)) → 23 kişilik 100 sınıfta check ediyorum. Bir sınıfta aynı dg'si olan insan var mı?
```

```
→ True
  0.55
→ False     } gibi sonuçlar
  0.51      }      verir
→ False
  0.5064
  ⋮
```

* 30 people min 3 kişi ✓

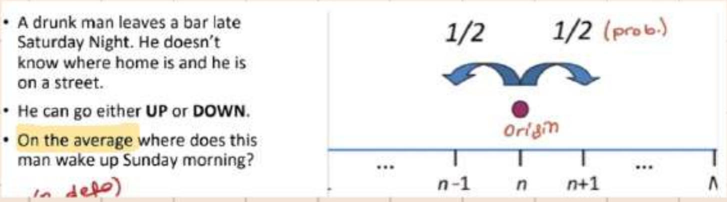
⊛ 

```
for numPeople in [10, 20, 40, 100]:
    print('For', numPeople,
          'est. prob. of a shared birthday is',
          birthdayProb(numPeople, 2, 10000))
```

→ for loop sayesinde farklı kişi sayılarında gerçekleşecek senaryoları görücez.

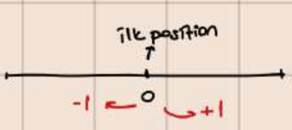
⊛ $\underbrace{\text{All models}}_{\text{only approximation}}$ are wrong but some are useful !

1-D Random Walk Exp



- A drunk man leaves a bar late Saturday Night. He doesn't know where home is and he is on a street.
- He can go either **UP** or **DOWN**.
- On the average where does this man wake up Sunday morning?

(n defa)



(x ekseni)

* Direction "+1" veya "-1" ⇒ n defa position değiştirdim en sonki position'ım neresi olur?

$\frac{1}{2}$ olasılıkları

```
import random

def random_walk_1D(n):          → n iteration yapıcaksın "for" kullan.
    position = 0
    for i in range(n):
        direction = random.choice([1,-1])
        position += direction     → Direction "-1"se mesela position "-1" çıkarılıp update olacak
    return position     → en sonda neyi bulmak istiyorsak

n = 100
final_position = random_walk_1D (n)
print (f"Position after {n} steps : {final_position}")
```

Output :

```
→ Position after 100 steps : 6
→ Position after 100 steps : -4
   ⋮
```

} run ettikçe

⇒ Üstteki sorunun Simulation kısmı : (üstteki koda devam ediyorum)

"Şu kadar kez deney yapılırsa average ne olur" → Expected Prob. ✓

sonunda amaç average bulmak!

```
def simulate_random_walks(num_trials, n):

        summation = 0

        for i in range(num_trials):

                x = random_walk_1D(n)

                summation += x

        return summation / num_trials   → Average'ı return'ün içinde bulabilirim
                                          (Probability)
x = simulate_random_walks(10000, 100)

print(x)
```

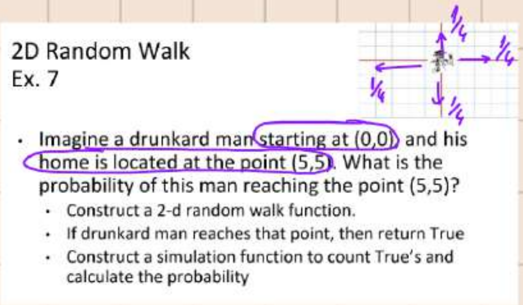
num. of trials ↑ , sample mean'e (gerçek probability'e) yaklaşılır.

↳ Central Limit Theorem (CLT)

2-D Random Walk (x-y axis)

(n=100 , num. of trials = 10000 olsun)

n → num of steps



2D Random Walk
Ex. 7

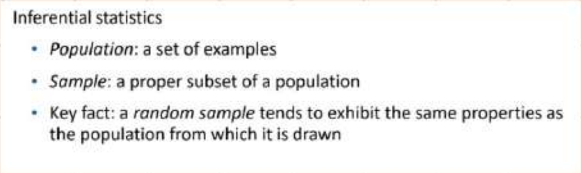
- Imagine a drunkard man starting at (0,0) and his home is located at the point (5,5). What is the probability of this man reaching the point (5,5)?
    - Construct a 2-d random walk function.
    - If drunkard man reaches that point, then return True
    - Construct a simulation function to count True's and calculate the probability

```
import random

def random_walk_2D(n):

        x = 0

        y = 0

        for i in range(n):

                direction = random.choice(['up', 'down', 'right', 'left'])

                if direction == 'up':

                        y = y + 1

                if direction == 'down':

                        y = y + 1
                          ⋮
                          ⋮
```

Devamı var! Daha simulation step var!

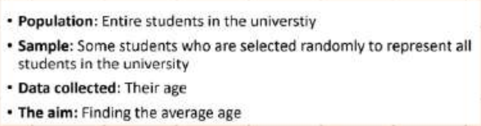
- WEEK 7 -

- Monte Carlo Simulation


Inferential statistics

- *Population*: a set of examples
- *Sample*: a proper subset of a population
- Key fact: a *random sample* tends to exhibit the same properties as the population from which it is drawn

Exp:


- **Population:** Entire students in the universtiy
- **Sample:** Some students who are selected randomly to represent all students in the university
- **Data collected:** Their age
- **The aim:** Finding the average age

Exp:

Flipping a Coin Twice

Do you think that the next flip will come up heads?

Exp:

Flipping a Coin 100 times

Now do you think that the next flip will come up heads?

Exp:

Flipping a Coin 100 times

Do you think that the probability of the next flip coming up heads is 52/100?

→ Why the Difference in Confidence

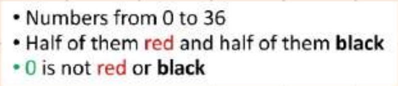
confidence in our estimate depends upon two things :

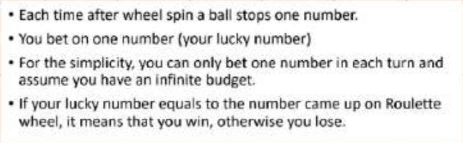
* Size of sample (100 vs 2)

* Variance of Sample (all heads vs. 52 heads out of 100)

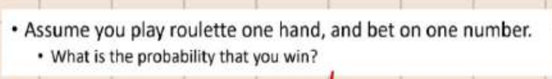
As the Variance ↑ , we need larger samples to have the same degree of confidence

Roulette Game


- Numbers from 0 to 36
- Half of them red and half of them **black**
- 0 is not red or **black**

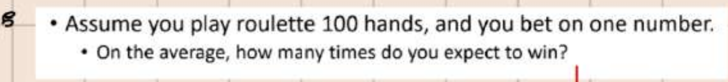

- Each time after wheel spin a ball stops one number.
- You bet on one number (your lucky number)
- For the simplicity, you can only bet one number in each turn and assume you have an infinite budget.
- If your lucky number equals to the number came up on Roulette wheel, it means that you win, otherwise you lose.

Exp:

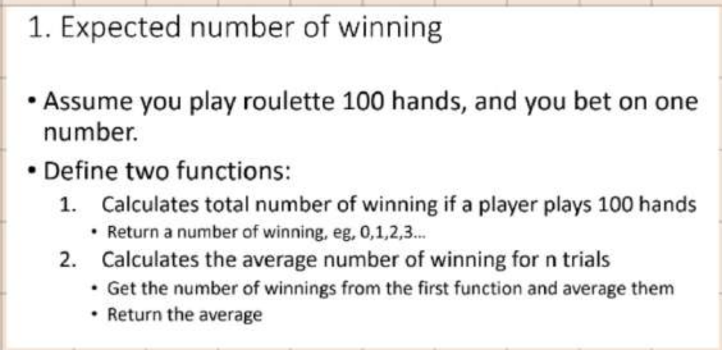

- Assume you play roulette one hand, and bet on one number.
  - What is the probability that you win?

↳ $1/37$

Exp:


- Assume you play roulette 100 hands, and you bet on one number.
  - On the average, how many times do you expect to win?

↳ $100 \times \frac{1}{37}$


1. Expected number of winning

- Assume you play roulette 100 hands, and you bet on one number.
- Define two functions:
  1. Calculates total number of winning if a player plays 100 hands
     - Return a number of winning, eg, 0,1,2,3...
  2. Calculates the average number of winning for n trials
     - Get the number of winnings from the first function and average them
     - Return the average

Roulette game çözümü :

1) ↗ kaç tanesi benim tuttuğum sayıya eşit olur.
→ 100 oyun oynadık kaç kez kazandık onu return edicez

2) Average num. of winning for n trials

Roulette Game

total num. of plays

```
import random

def numofWinning (luckynum, play)

    count = 0

    for i in range(play):

        if luckynum == random.randint (0,36)

            count += 1

    return count
```

→ kaç kere kazandığımı döndürüyorum

(Burda oyun kazanılmıştı artık, if statement içinde)

2.kısım (Simulation kısmı)

```
def simNumofWinning (luckynum, play, numTrials):

     countwin = 0

     for i in range (numTrials):

          countwin += numofWinning (luckynum, play)

     return countwin / numTrials

 X= simNumofWinning (13,100, 1000)

 print ("expected number of winning is", x)
```

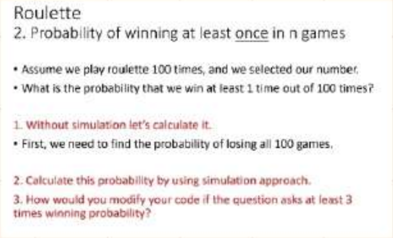
→
```
   win
   win
   win
   win
    |
    |
    |
   expected number of winning is  2.718
```

Real : $\frac{100}{37} = \boxed{2.72...}$

↓ expected number'ım buna çok yaklaşmış çünkü # of trial sayım fazla

100 oyunun en az 1 kere kazanma olasılığı

Exp:



Roulette

2. Probability of winning at least once in n games

- Assume we play roulette 100 times, and we selected our number.
- What is the probability that we win at least 1 time out of 100 times?

1. Without simulation let's calculate it.

- First, we need to find the probability of losing all 100 games.

2. Calculate this probability by using simulation approach.

3. How would you modify your code if the question asks at least 3 times winning probability?

$\frac{1}{37}$ ihtimal seçtiğim luckynum gelir you "win"

* Prob. of not winning in one game → $\left(\frac{36}{37}\right)^{100}$

* Prob. of winning = $1-\left(\frac{36}{37}\right)^{100}$

Determine 2 functions:

→ Determine if there is at least one win out of 100 games

- Return 1 or 0, or true or false

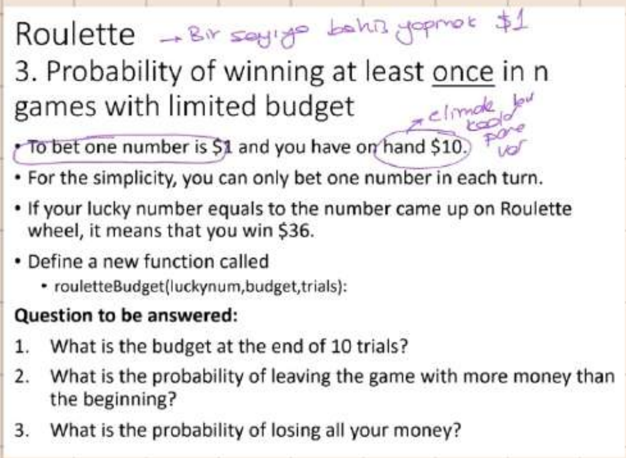
burası sadece (win lose win ⋮) yazdırıcak

→ Estimate the prob. of is at least one win

- Use the first func. to count number of winnings, then find the probability.

```
import random

def numofWinning(luckynum, play):        100 times I play
    flag = 0   → win sayısını kastediyor
    for i in range (play):
        if luckynum = random.randint (0,36):
            flag = 1
    return flag   → win or lose

def atleastoneProb (luckynum, play, numTrials):
    count = 0
    for i in range (numTrials):
        if numofWinning (luckynum, play) == 1:
            count += 1
    return count / numTrials

print (atleastoneProb (13, 100, 1000))
```

Exp: Roulette 3



Roulette → Bir sayıya bahis yapmak $1

3. Probability of winning at least once in n games with limited budget

- To bet one number is $1 and you have on hand $10. → elimde toplam $10 para var
- For the simplicity, you can only bet one number in each turn.
- If your lucky number equals to the number came up on Roulette wheel, it means that you win $36.
- Define a new function called
  - rouletteBudget(luckynum,budget,trials):

Question to be answered:

1. What is the budget at the end of 10 trials?
2. What is the probability of leaving the game with more money than the beginning?
3. What is the probability of losing all your money?

1. sorunun cevabı

```
import random
def rouletteBudget (luckynum, budget=10, play=10)
    budgetIncrease = 0
    budgetInitial = budget
    for i in range (play):          → 10 times
        budget -= 1
        if luckynum = random.randint(0,36):
            budget += 36
        if budget <= 0 :
            break   → para kalmadıysa oynayamaz artık
    if budget > budgetInitial :
        budgetIncrease = 1
        print('budget is increased")
    elif budget == budgetInitial:
        print ("budget is the same")
    else:
        print ("budget is decreased")
    return budgetIncrease

print(rouletteBudget(7,10,10))
```

Output:

```
→ budget is decreased
  0

→ budget is increased
  1
  ⋮
```

Simulation part

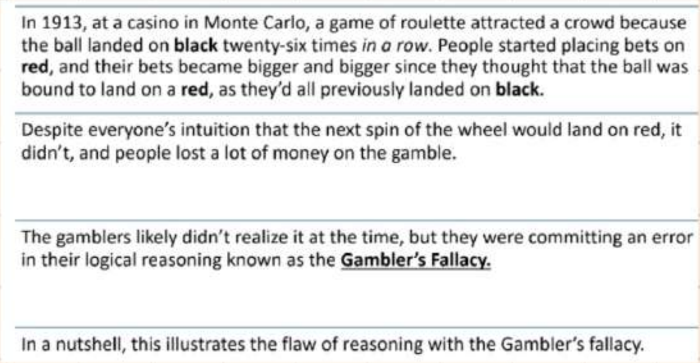
```
def budgetSim(luckynum, budget, play, numTrials)
    count = 0
    for i in range (numTrials):
        result = rouletteBudget(luckynum, budget, play)     ← 0 yada 1
        if result == 1:
            count += 1
    return count / numTrials
```

↱ bi örnek verip bu Gambler's Fallacy mi diye sorabilir!

Gambler's Fallacy → sözel soru!

Past experience'lardan dolayı insanların probability hesaplarken düştüğü yanılgılar.

→ 10 tane kırmızı top çektim → 11. de büyük ihtimal kırmızı olur.
→ 11. artık kırmızı çıkmaz.



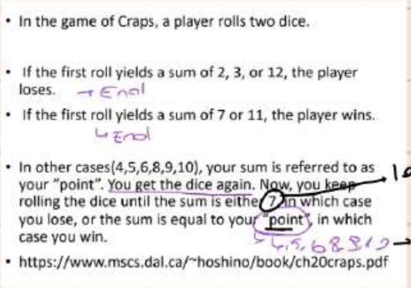
In 1913, at a casino in Monte Carlo, a game of roulette attracted a crowd because the ball landed on **black** twenty-six times *in a row*. People started placing bets on **red**, and their bets became bigger and bigger since they thought that the ball was bound to land on a **red**, as they'd all previously landed on **black.**

Despite everyone's intuition that the next spin of the wheel would land on red, it didn't, and people lost a lot of money on the gamble.

The gamblers likely didn't realize it at the time, but they were committing an error in their logical reasoning known as the **Gambler's Fallacy.**

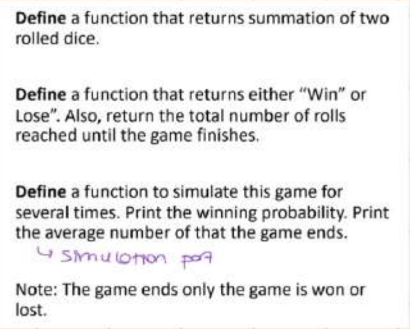
In a nutshell, this illustrates the flaw of reasoning with the Gambler's fallacy.

Game of Craps → geçen yılın vizesinde!



- In the game of Craps, a player rolls two dice.
- If the first roll yields a sum of 2, 3, or 12, the player loses. → End
- If the first roll yields a sum of 7 or 11, the player wins. ↳ End
- In other cases(4,5,6,8,9,10), your sum is referred to as your "point". You get the dice again. Now, you keep rolling the dice until the sum is either 7 in which case you lose, or the sum is equal to your "point", in which case you win. → lose → 4,5,6,8,9,10 → hangisini atıyorsa o aşamada
- https://www.mscs.dal.ca/~hoshino/book/ch20craps.pdf

"2 zar var toplamlarına göre kaybedip kazanıyorum"



Define a function that returns summation of two rolled dice.

Define a function that returns either "Win" or Lose". Also, return the total number of rolls reached until the game finishes.

Define a function to simulate this game for several times. Print the winning probability. Print the average number of that the game ends. ↳ simulation part

Note: The game ends only the game is won or lost.

The question is :

would you like to play the game considering the winning and losing probabilities?

(winning probability?)

```
import random
def rollDie():
    die1 = random.randint(1,6)
    die2 = random.randint(1,6)
    return die1 + die2
def playCraps():
    result = " "
    firstRoll = rollDie()
    if firstRoll ==7 or firstRoll == 11 :
        result = "win"
    elif firstRoll == 2 or firstRoll == 3 or firstRoll == 12 :
        result = "lose"
    else :
        while True :
            secondRoll = rollDie ()
            if secondRoll == firstRoll :
                result = "win"
                break
            elif secondRoll == 7 :
                result == "Lose"
                break
    return result
print (playCraps ())  → buraya kadar (win lose win ⋮) yazdırır


def crapsSimu (numTrials)
    count = 0
    for i in rage (numTrials) :
        result = playCraps()
        if result == "win"
            count += 1
    return count / numTrials


print("prob. of winning the game is :", crapsSimu (1000))
```

Output :

```
→ win
  prob. of winning the game is : 0.503
→ lose
  prob. of winning the game is : 0.488
```

The Pros and Cons of Greedy

→ Easy to implement

→ Computationally efficient

⚠ Not always yield the best solution, but approximation is good.

Law of Large Numbers

- As a sample size grows, its mean gets closer to the average of the whole population. (# of trials↑, the observed outcomes will closer to the "actual" probability.)

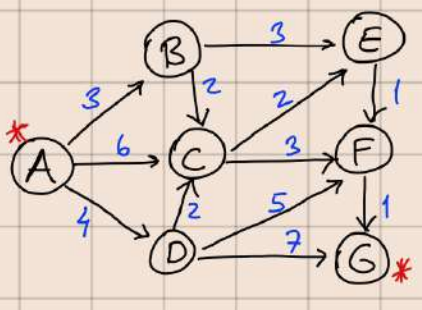
* Actual Probability = The true or observed value of the probability of an event

* Estimated Probability = The predicted value of the probability of an event.

When we use simulations:

- To model systems that are mathematically harder.
- To extract useful intermediate results

Dijkstra Algoritması



B'yi seçtim en küçük old. için B'nin komşularına baktım.

D'den bakınca "6" oldu ona 5'ten büyük o yüzden yazılmaz

7 yazamam

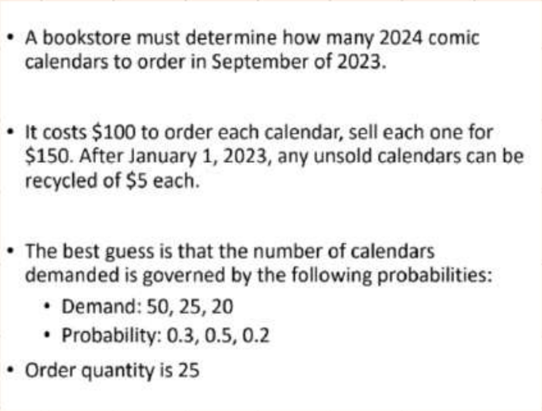
	B	C	D	E	F	G
	3✓	6	4	∞	∞	∞
B		5	4✓	6	∞	∞
D		5✓		6	9	11
C				6✓	8 (5+3)	11
E					7✓ (6+1)	11
F						8 (7+1)

E'ye 7'yle geldim

WEEK 9

→ Simulation

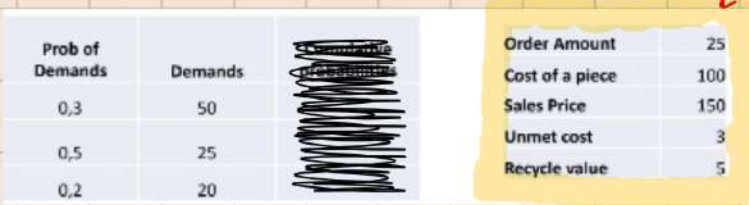
Newsvendor Problem :



- A bookstore must determine how many 2024 comic calendars to order in September of 2023.
- It costs $100 to order each calendar, sell each one for $150. After January 1, 2023, any unsold calendars can be recycled of $5 each.
- The best guess is that the number of calendars demanded is governed by the following probabilities:
  - Demand: 50, 25, 20
  - Probability: 0.3, 0.5, 0.2
- Order quantity is 25

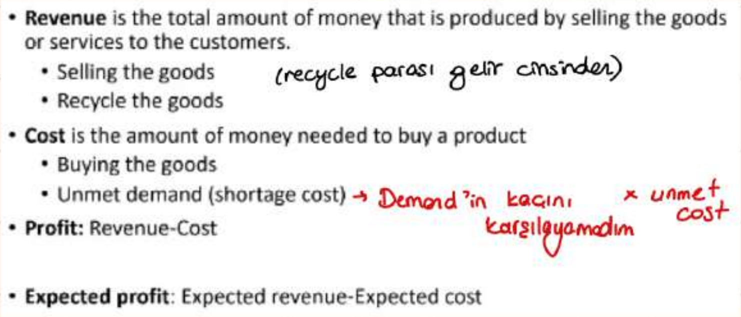
order > demand ise recycle edilir.

given



Prob of Demands	Demands	[redacted]
0,3	50	[redacted]
0,5	25	[redacted]
0,2	20	[redacted]

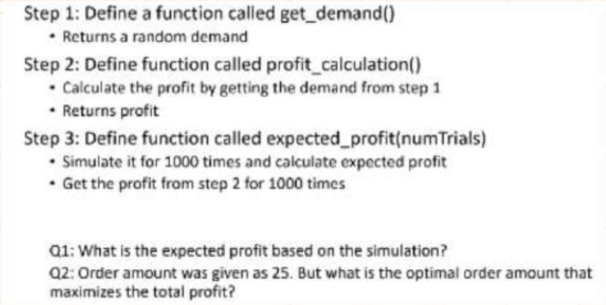
Order Amount	25
Cost of a piece	100
Sales Price	150
Unmet cost	3
Recycle value	5



- **Revenue** is the total amount of money that is produced by selling the goods or services to the customers.
  - Selling the goods (recycle parası gelir cinsinden)
  - Recycle the goods
- **Cost** is the amount of money needed to buy a product
  - Buying the goods
  - Unmet demand (shortage cost) → Demand'in kaçını karşılayamadım × unmet cost
- **Profit:** Revenue-Cost
- **Expected profit**: Expected revenue-Expected cost

↳ What is the expected profit? Conduct a simulation by hand for 20 trials.

1) Generate a random demands. func.
2) Calculate profit func. (and return)
3) Expected profit func. (simulation)



Step 1: Define a function called get_demand()
- Returns a random demand

Step 2: Define function called profit_calculation()
- Calculate the profit by getting the demand from step 1
- Returns profit

Step 3: Define function called expected_profit(numTrials)
- Simulate it for 1000 times and calculate expected profit
- Get the profit from step 2 for 1000 times

Q1: What is the expected profit based on the simulation?
Q2: Order amount was given as 25. But what is the optimal order amount that maximizes the total profit?

k=1 - 1 tane sayı seç
cumulative gerek yok

KOD

```
import random
def get_demand() :
    probabilities = [0.3, 0.5, 0.2]
    demands = [50, 25, 20]
    demand = random.choices(demands, probabilities, k=1)
    return demand
```

!!! ③

!!! ③

!!! ① → Demand değerlerini sahip oldukları olasılık değerine göre döndürüyor. k=1 de "1" sayı döndür demek

(hepsini sonda bir return et!)

```
def profit_calculation() :
    order_amount = 25
    cost_per_piece = 100
    Sales_price = 150
    unmet_cost = 3
    recycle_cost = 5
    d = get_demand()
    demand = d[0]
```

Burda liste olarak gönderdiği için "k=1"de onun "0." elemanını almak istiyorum ben.

→ hesaplama yapabilmek için fonk'un içine get_demand fonksiyonunu da çağırdık.

```
sold_pieces = min(order_amount, demand)
                      "-" de çıkabilir
unsold_pieces = max(order_amount - demand, 0)

unmet_demand = max(demad- order_amount, 0)


revenue = sold_pieces * sales_price + unsold_pieces * recycle_cost

cost = order_amount * cost_per_piece + unmet_demand * unmet_cost

return revenue - cost
```

* Demand > order
↳ sale, order kadar

* Order > Demand
↳ sale, demand kadar

Print etmeyi unutma

```
print(profit_calculation())
```

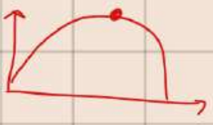
Simulation Part

```
def expected_profit(num_trials):
    total_profit = 0
    for i in range(num_trials):
        total_profit += profit_calculation()
    return total_profit / num_trials
```

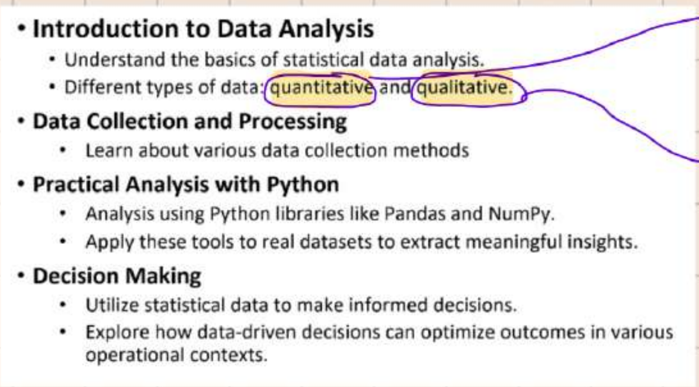
→ ortalamasını alıyor (expected)

```
print(expected_profit(10000))
```

!!! Order size'a 23, 24, 25, 26, 27 – gibi sayılar vermeyi denedik 25'ten sonra fark azalma eğiliminde devam etti
o yüzden "max profit": $\frac{\text{order}}{\text{size}}$ = 25 old. oldu.



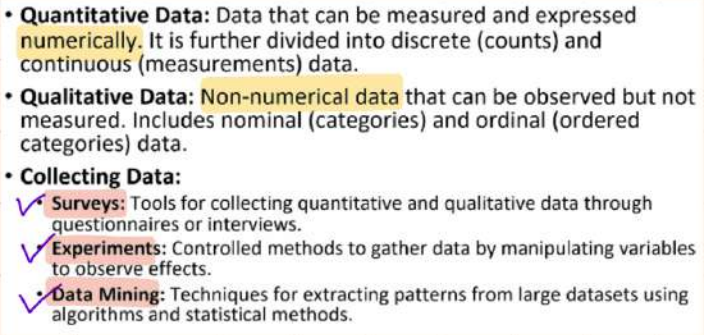
Experimental Data



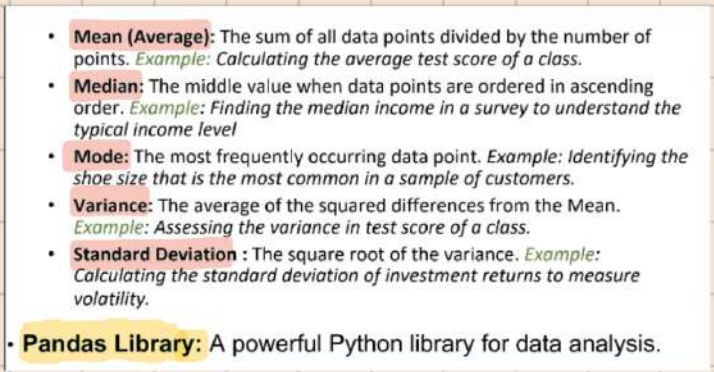
- **Introduction to Data Analysis**
  - Understand the basics of statistical data analysis.
  - Different types of data: quantitative and qualitative.
- **Data Collection and Processing**
  - Learn about various data collection methods
- **Practical Analysis with Python**
  - Analysis using Python libraries like Pandas and NumPy.
  - Apply these tools to real datasets to extract meaningful insights.
- **Decision Making**
  - Utilize statistical data to make informed decisions.
  - Explore how data-driven decisions can optimize outcomes in various operational contexts.

quantitative datas
4 yaş'ların olduğu bir data seti

qualitative (quality)
nonnumerical datas
(gender vs..)



- **Quantitative Data:** Data that can be measured and expressed numerically. It is further divided into discrete (counts) and continuous (measurements) data.
- **Qualitative Data:** Non-numerical data that can be observed but not measured. Includes nominal (categories) and ordinal (ordered categories) data.
- **Collecting Data:**
  - ✓ **Surveys:** Tools for collecting quantitative and qualitative data through questionnaires or interviews.
  - ✓ **Experiments**: Controlled methods to gather data by manipulating variables to observe effects.
  - ✓ **Data Mining:** Techniques for extracting patterns from large datasets using algorithms and statistical methods.

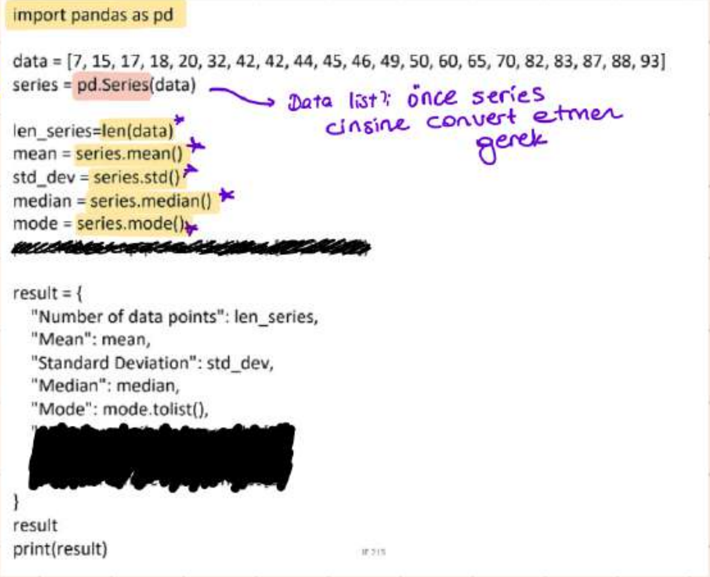


- **Mean (Average):** The sum of all data points divided by the number of points. *Example: Calculating the average test score of a class.*
- **Median:** The middle value when data points are ordered in ascending order. *Example: Finding the median income in a survey to understand the typical income level*
- **Mode:** The most frequently occurring data point. *Example: Identifying the shoe size that is the most common in a sample of customers.*
- **Variance:** The average of the squared differences from the Mean. *Example: Assessing the variance in test score of a class.*
- **Standard Deviation :** The square root of the variance. *Example: Calculating the standard deviation of investment returns to measure volatility.*

- **Pandas Library:** A powerful Python library for data analysis.

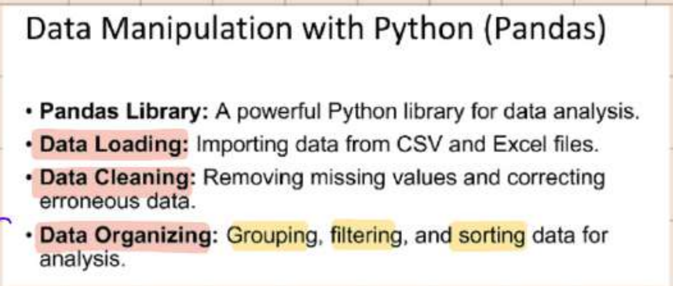
→ series. .... ile işlem yapıyor hep!

kod örneği :



```
import pandas as pd

data = [7, 15, 17, 18, 20, 32, 42, 42, 44, 45, 46, 49, 50, 60, 65, 70, 82, 83, 87, 88, 93]
series = pd.Series(data)

len_series=len(data)*
mean = series.mean()*
std_dev = series.std()*
median = series.median() *
mode = series.mode()*
[redacted]

result = {
   "Number of data points": len_series,
   "Mean": mean,
   "Standard Deviation": std_dev,
   "Median": median,
   "Mode": mode.tolist(),
[redacted]
}
result
print(result)
```

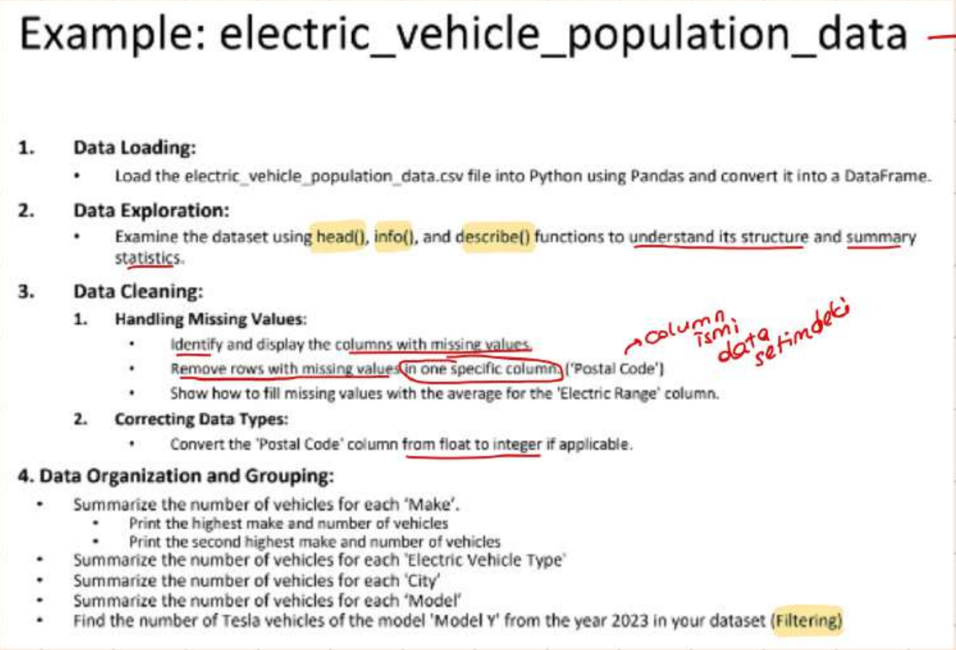
→ Data list'i önce series cinsine convert etmen gerek



Data Manipulation with Python (Pandas)

- **Pandas Library:** A powerful Python library for data analysis.
- **Data Loading:** Importing data from CSV and Excel files.
- **Data Cleaning:** Removing missing values and correcting erroneous data.
- **Data Organizing:** Grouping, filtering, and sorting data for analysis.

↳ Sırasıyla burdaki step'lerden geçerek ilerleyeceğiz!



Example: electric_vehicle_population_data

→ file'ının adı (dataların olduğu)

1. **Data Loading:**
   - Load the electric_vehicle_population_data.csv file into Python using Pandas and convert it into a DataFrame.
2. **Data Exploration:**
   - Examine the dataset using head(), info(), and describe() functions to understand its structure and summary statistics.
3. **Data Cleaning:**
   1. **Handling Missing Values:**
      - Identify and display the columns with missing values.
      - Remove rows with missing values in one specific column ('Postal Code') → column ismi data setindeki
      - Show how to fill missing values with the average for the 'Electric Range' column.
   2. **Correcting Data Types:**
      - Convert the 'Postal Code' column from float to integer if applicable.

4. Data Organization and Grouping:

- Summarize the number of vehicles for each 'Make'.
  - Print the highest make and number of vehicles
  - Print the second highest make and number of vehicles
- Summarize the number of vehicles for each 'Electric Vehicle Type'
- Summarize the number of vehicles for each 'City'
- Summarize the number of vehicles for each 'Model'
- Find the number of Tesla vehicles of the model 'Model Y' from the year 2023 in your dataset (Filtering)

* Önce spyder dosyanı file → save as → masaüstüne at!

* Data'ların yer aldığı dosyayı da → masaüstüne at

1) Data-Loading Step;

```
import pandas as pd

df = pd.read_csv('electric_vehicle_population_data.csv')
```

→ Data setini okudu ve "data frame" cinsine convert etti! (df) ismi öylesine

2) Data Exploration;

print(df.head()) → small infos, and summary about your long data (mesela data'nın ilk 5 satırını döndürür fikir olsun diye)

* print(df.info()) → column'lar hakkında bilgi verir (object sonucu veriyorsa o data hem int hem float sonucu verir.) - non-null

print(df.describe()) → some statistics for numerical columns only

↳ column'ların isimleri, non-null, object/float gibi genel bilgilendirme yapar.

3) Data Cleaning :

→ Handling Missing Values ←

first find the missing values

print (df.isna() . sum) → total number of NA's for each column → output : şu column'da ... kadar boş satır var

*** # remove rows with missing values in one specific column

df = df. dropna (subset = ['......']) → Direk column'ın ismini yaz. → Bu column'daki "na" satırlarını atıyor.

↳ en üstte tanımladığım df'i yeni hâline convert ediyorum. Çünkü data verilerimle oynamam lazım. → tekrar df.isna yaptık!

print (df.isna().sum()) → column'un yanında "0" görürüz çünkü o column'da NA kalmadı.

Show how to fill missing values with the average for the 'Electric Range' column.

data frame

column ismi böyle yazılır.

average_electric_range = df ['Electric Range'] . mean() → column'un mean'i bulduk önce.

df ['Electric Range'] = df ['Electric Range'] . fillna (average_electric_range)

bu columndaki na'ları dolduracak

↳ Buraya yazdığım şeyle doldurur. Sayı yazarsam "sayıyla" doldurur.

→ Correcting Data Types ←

int, float gibi

df ['Postal Code'] = df ['Postal Code'] . astype (int) → Postal Code column'ım float bir değerdi. o değeri int bir değerle değiştirdik.

print (df. info()) → info'yu tekrar çağırarak Postal Code'un int olarak revize edildiğini gördük.

4) Data Organization and Grouping

column name

(1) Display the counts for each make (brand)

** make_counts = df ['Make'] . value_counts () → Make columnunda şu kadar tane audi, şu kadar tesla gibi—

print (make_counts) → Her column için bakh (Tesladan şu kadar, audi'den şu kadar vs. çıktısı verir)

"1" dersem 2. sayısı en çok olan markayı söyler.

* top_make = make_counts . index [0] → sayısı en çok olan Markayı söyler

* top_make_count = make_counts . iloc [0] → Sayısı en çok olan markanın "sahip old. sayıyı verir"

print ("top brand is : ", top_make)
print ("top quantity sold is : ", top_make_count)

→ output : top brand is Tesla
top quantity sold is : 80819

hangi sütunu yazarsan ordaki veri çeşitlerinden her birinden ne kadar old. sayar.

Summarize the num. of vehicles for each 'Electric Vehicle Type' → Bu column için

```
type_counts = df['Electric Vehicle Type'].value_counts()
print(type'_counts)
```

Output:

Battery Elec. Vehicle → 14973
Plug-in Hybrid Vehicle → 3943

Type'ı verdi — kaç adet old.

- FILTERING -

iki tane filtreleme yapmayacaksan bu paranteze gerek yok.

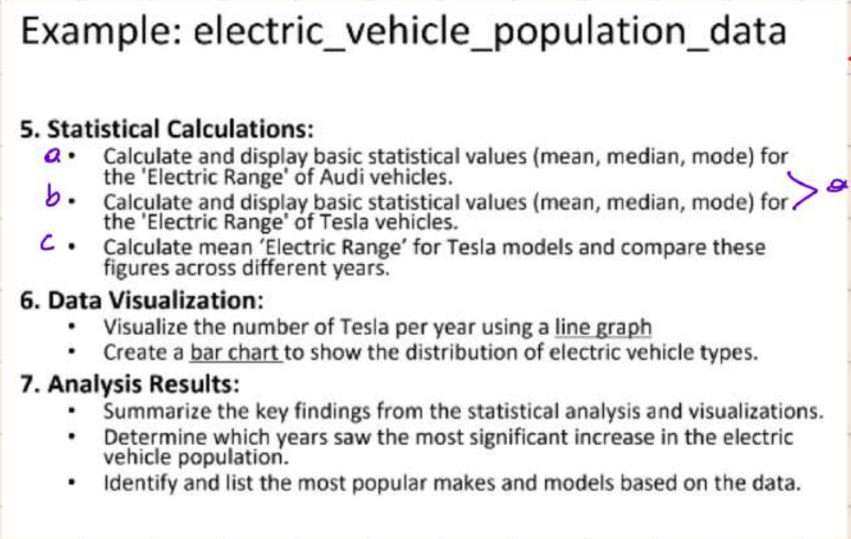
Find the number of Tesla vehicls of the model 'Model Y' from the year '2023'.

```
filtered_df = df[(df['Model']== 'Model Y') & (df['Model Year'] == 2023)]
```

1. criteria — 2. criteria

```
print(filtered_df)
```

✓ Model'lardan Model Y'leri ve 2023'te olanlarını saydı



Example: electric_vehicle_population_data

5. Statistical Calculations:

- a • Calculate and display basic statistical values (mean, median, mode) for the 'Electric Range' of Audi vehicles.
- b • Calculate and display basic statistical values (mean, median, mode) for the 'Electric Range' of Tesla vehicles.
- c • Calculate mean 'Electric Range' for Tesla models and compare these figures across different years.

6. Data Visualization:

- Visualize the number of Tesla per year using a line graph
- Create a bar chart to show the distribution of electric vehicle types.

7. Analysis Results:

- Summarize the key findings from the statistical analysis and visualizations.
- Determine which years saw the most significant increase in the electric vehicle population.
- Identify and list the most popular makes and models based on the data.

✱✱✱ > aynı şey

filter only this brand

5-a) Audi araçların (make column'u), Electric Range'leri üzerindeki hesaplamalar. (mean, median, mod)

(filter) (data frame'imde artık Audi'yle ilgili veri setlerim var)

```
audi_vehicle = df[(df['Make'] == 'AUDI') & (df['Electric Range'] > 0)]
```

→ Sadece audi setini filtreledim.
→ Electric range'in 0'dan büyükleri

```
audi_mean = audi_vehicle['Electric Range'].mean()
                                          .median()
```

→ print(audi_mean) → Electric range'lerin mean'ini yazdırır.

5-c) Tesla aracının yıllarını gruplandırıp o grupların Electric range'lerinin mean'ini verir ayrı ayrı. compare edebiliriz bakınca.

```
tesla_sales_by_year = df[df['Make']=='Tesla'].groupby('Model Year')['Electric Range'].mean()
print(tesla_sales_by_year)
```

↳ column name

6- Data Visualization

import pandas as pd

import matplotlib.pyplot as plt → Bunun eklemesi lazım

first filter

num. of Tesla per year using a line graph.

tesla_df = df [df['Make'] == 'Tesla']

vehicle_counts_per_year = tesla_df.groupby('Model Year').size() → 2024'te şu kadar var... gibi hepsini sayıyor size'da

Size direk data sayısını sayıyor.

Bunun yerine value_counts() kullanamayız çünkü o: şundan şu kadar var gibi döndürüyor.

print (vehicle_counts_per_year) Hangi yılda kaç tane Tesla olduğunu görürüm.

plt.figure()

$$\texttt{plt.plot(}\underbrace{\texttt{vehicle\_counts\_per\_year.index}}_{\text{bu parametrenin ilk indexini aldı}}\texttt{, }\underbrace{\texttt{vehicle\_counts\_per\_year.values}}_{\text{2.kısmını aldı (sayısal kısmı)}}\texttt{)}$$

tablonun x ve y ekseninden bahsettik

plt.xlabel ("num of vehicle")

plt.ylabel ("year")

eksenlere isim verir

plt.title ("tesla sales by year") → grafiğe bir title verdik!

create a bar chart distribution of electric vehicles

ev_type_counts = df.groupby('electric vehicle').size()

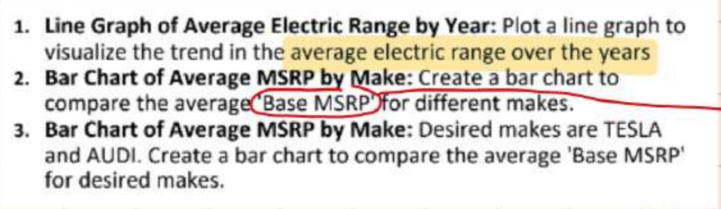
data frame electric vehicle'a göre gruplandır ve sayılarını döndürdü

plt.figure()

ev_type_counts.plot(kind='bar')

plt.show → En son bunu yazıp çalıştırdık

"Ör



1. **Line Graph of Average Electric Range by Year:** Plot a line graph to visualize the trend in the average electric range over the years
2. **Bar Chart of Average MSRP by Make:** Create a bar chart to compare the average 'Base MSRP' for different makes.
3. **Bar Chart of Average MSRP by Make:** Desired makes are TESLA and AUDI. Create a bar chart to compare the average 'Base MSRP' for desired makes.

Excel'deki columnlardan biri

orijinal data frame → mesela yukarıda df'i sadece teslalardan oluşacak şekilde güncellemiştik. Burda direk orijinal data frame'i kullanıyoruz.

1) average_electric_range_per_year = df.groupby('Model Year') ['Electric Range'].mean

$$\texttt{plt.plot(}\underbrace{\texttt{aver\_elec\_ran\_per\_year.index}}_{\text{x ekseni}}\texttt{, }\underbrace{\texttt{ave\_elec\_ran\_per\_year.values}}_{\text{y ekseni}}\texttt{)}$$

plt.show

eksenlere isim vermeden direk oluştur demiş.

Average lazım!

2)
```
average_msrp = df.groupby('Make')['Base MSRP'].mean
average_msrp.plot(kind = 'bar')
plt.show()
```

→ Böyle print edersek; sol tarafta name'lerini sağ tarafta average'ları görürüz.

I took the tesla and audi rows both. (Bir kategoriden 2 markayı seçiyor)

3)
```
desired_models = df.[['Make'].isin(['Tesla', 'Audi'])]
average_msrp_for TA = desired_models . groupby('Make')['Base MSRP'].mean()
average_msrp_forTA .plot (kind = 'bar')
plt. show
```

WEEK 12

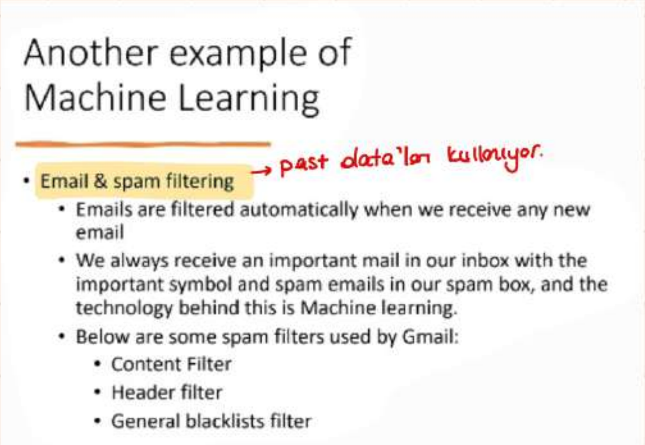
(AI = Artificial Intelligence)

Machine Learning

① AI uses "Machine learning techniques".

Examples of Machine Learning

- **AlphaGo** is the first computer program to defeat a professional human Go player, the first to defeat a Go world champion
- **Netflix/Spotify**
- **Self-driving cars**
  - Waymo

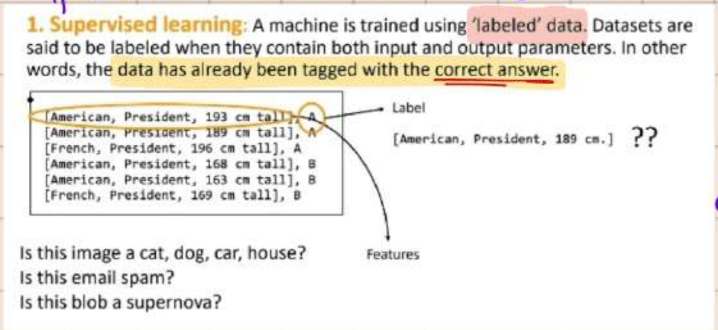


Another example of Machine Learning

- Email & spam filtering → past data'ları kullanıyor.
  - Emails are filtered automatically when we receive any new email
  - We always receive an important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning.
  - Below are some spam filters used by Gmail:
    - Content Filter
    - Header filter
    - General blacklists filter

Machine Learning requires Past Datas! (Data sets)

→ önceki outputlara bakarak prediction yapılıyor.
→ verimizin input ve output'unu biliyoruz.



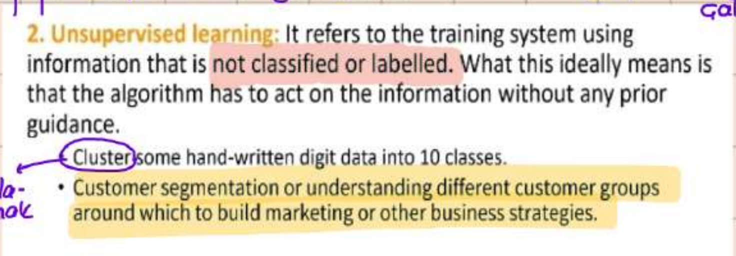
1. Supervised learning: A machine is trained using 'labeled' data. Datasets are said to be labeled when they contain both input and output parameters. In other words, the data has already been tagged with the correct answer.

```
[American, President, 193 cm tall], A
[American, President, 189 cm tall], A
[French, President, 196 cm tall], A
[American, President, 168 cm tall], B
[American, President, 163 cm tall], B
[French, President, 169 cm tall], B
```

[American, President, 189 cm.] ??

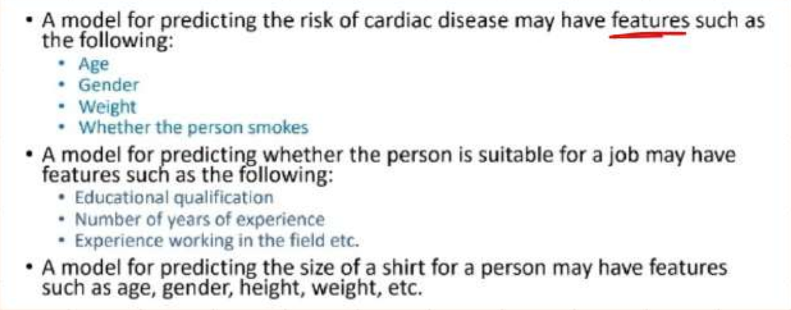
Is this image a cat, dog, car, house?
Is this email spam?
Is this blob a supernova?

→ tahminleme değil gruplama var!
→ output'u bilmiyoruz, verimizi cluster(gruplandırmaya) çalışıyoruz



2. Unsupervised learning: It refers to the training system using information that is not classified or labelled. What this ideally means is that the algorithm has to act on the information without any prior guidance.

- Cluster some hand-written digit data into 10 classes. ← gruplamak
- Customer segmentation or understanding different customer groups around which to build marketing or other business strategies.



- A model for predicting the risk of cardiac disease may have features such as the following:
  - Age
  - Gender
  - Weight
  - Whether the person smokes
- A model for predicting whether the person is suitable for a job may have features such as the following:
  - Educational qualification
  - Number of years of experience
  - Experience working in the field etc.
- A model for predicting the size of a shirt for a person may have features such as age, gender, height, weight, etc.

uygulamalar feedback alıyor. → bu sorular "label" olmuş oluyor.

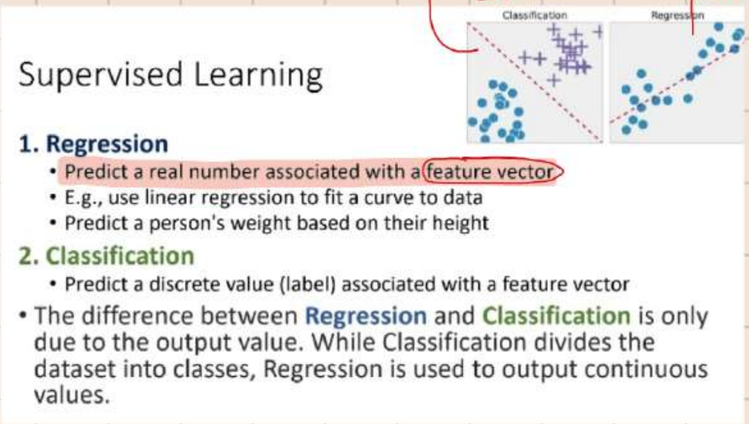
Supervised or Unsupervised Learning?

- Scenario 1: Facebook face recognition → Supervised
- Scenario 2: Netflix/spotify movie or song recommendation → Supervised
- Scenario 3: Document Clustering → Unsupervised → tahminleme yok gruplama var sadece.

→ Sen misin değil misin diye soruyor. Inputlarla bizden aldığı yanıtlarla output'lar elde ediyor.

Try to clarify date sets

data setleriyle prediction yapmaya çalışıyorum.

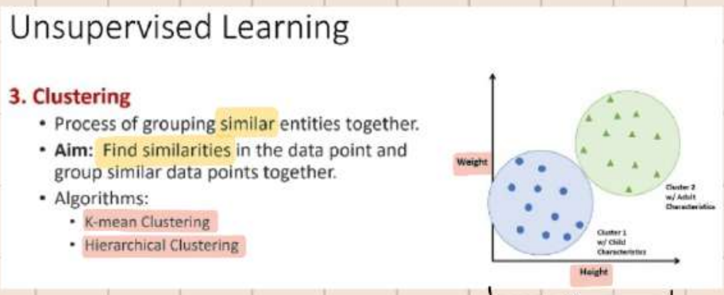


Supervised Learning

1. Regression
- Predict a real number associated with a feature vector
- E.g., use linear regression to fit a curve to data
- Predict a person's weight based on their height

2. Classification
- Predict a discrete value (label) associated with a feature vector

- The difference between **Regression** and **Classification** is only due to the output value. While Classification divides the dataset into classes, Regression is used to output continuous values.



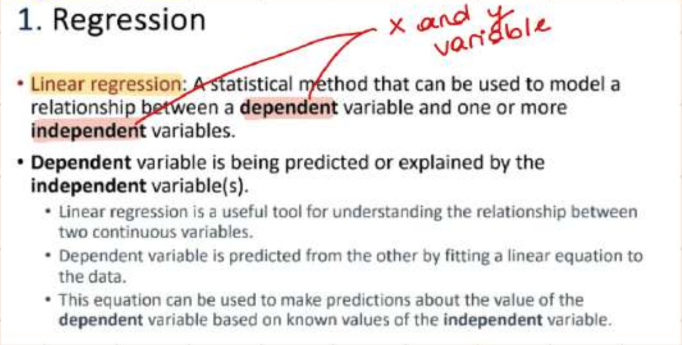
Unsupervised Learning

3. Clustering
- Process of grouping similar entities together.
- **Aim:** Find similarities in the data point and group similar data points together.
- Algorithms:
  - K-mean Clustering
  - Hierarchical Clustering

Weight
Height
Cluster 1 w/ Child Characteristics
Cluster 2 w/ Adult Characteristics

clustring

⊛ Regression



1. Regression

x and y variable

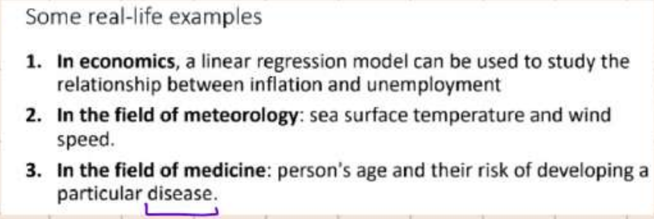
- Linear regression: A statistical method that can be used to model a relationship between a **dependent** variable and one or more **independent** variables.
- **Dependent** variable is being predicted or explained by the **independent** variable(s).
  - Linear regression is a useful tool for understanding the relationship between two continuous variables.
  - Dependent variable is predicted from the other by fitting a linear equation to the data.
  - This equation can be used to make predictions about the value of the **dependent** variable based on known values of the **independent** variable.

- Crop yield → dependent
- Based on rainfall → independent
- Predict crop yield

★★ Crop yield, yağmur yağışına bağlıdır. Dolayısıyla rainfall bağımsız değişkenken crop yield bağımlı değişken olur.

Dependent variable → genelde "y" ile ifade edilir. (crop yield)

Independent Variable → genelde "x" ile ifade edilir. (rainfall)

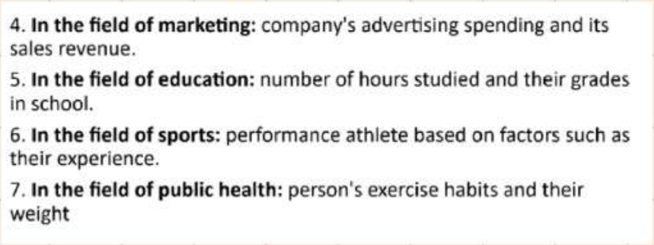


Some real-life examples

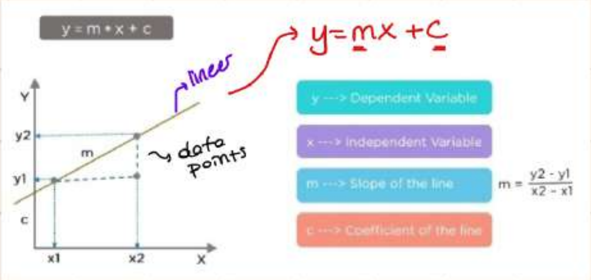
1. **In economics,** a linear regression model can be used to study the relationship between inflation and unemployment
2. **In the field of meteorology**: sea surface temperature and wind speed.
3. **In the field of medicine**: person's age and their risk of developing a particular disease.

- inflation → independent
- unemployment → dependent

dependent (to age)



4. **In the field of marketing:** company's advertising spending and its sales revenue.
5. **In the field of education:** number of hours studied and their grades in school.
6. **In the field of sports:** performance athlete based on factors such as their experience.
7. **In the field of public health:** person's exercise habits and their weight



y ---> Dependent Variable

x ---> Independent Variable

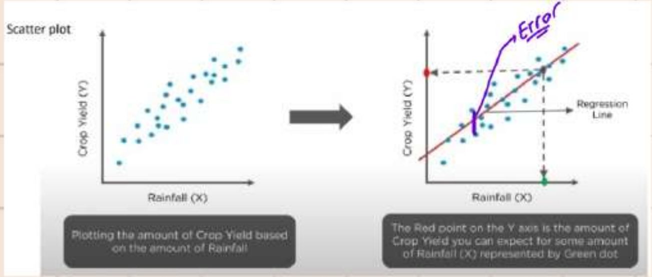
m ---> Slope of the line $m = \frac{y2 - y1}{x2 - x1}$

c ---> Coefficient of the line

$y = \underline{m}x + \underline{c}$

* lineer line ile predict some other y values. (prediction yapıyoruz)

⚠ Python'da bu line'i nasıl çizeceğimiz önemli!



Plotting the amount of Crop Yield based on the amount of Rainfall

The Red point on the Y axis is the amount of Crop Yield you can expect for some amount of Rainfall (X) represented by Green dot

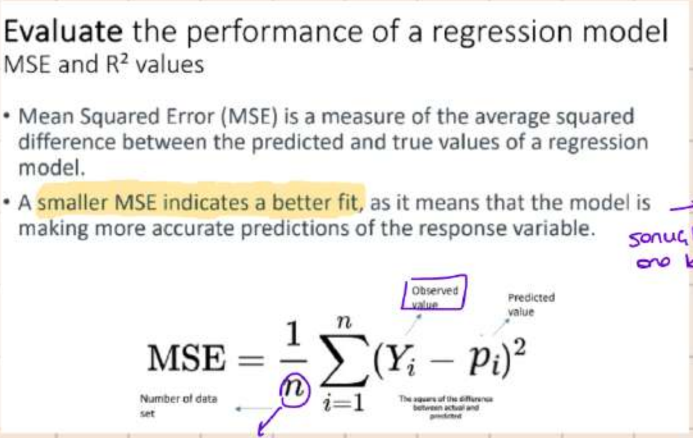
Erroru bütün data pointler için bulup toplayıp absolute value veya karesi yaptığımda o value'yu min. eden line'i buluyor. "Regression line"

* ilk stepte scatter plot yaratmalıyız çünkü line'ı çizmeden önce bazı observationlara ihtiyacımız var. observation'larım birbiriyle releated mi lineer çizgi ulaştırılabilir mi üstlerinden

kod 8

```
import pandas as pd

import matplolib . pyplot as plt  → plt. scatter kısmı için eklendi

import numpy as np → best regression line'ı oluşturmak için "min. sum of total error for each data set"


data = pd. read_excel ('rainfall-crops. xlsx') → Excel dosyasını okudu başta

print (data)

x = data ['rainfall']            Scatter plot
                                    için eksenler
y = data ['crop amount']

m, c = np. polyfit (x, y, 1) → regression line polinomum 1. dereceden olsun diye.

print (m, c) → m ve c değerlerini yazdırıyor. (denklemdeki slope ve coefficient'ı veren sayılar) → y = mx + c

y_line = m * x + c   Denklemi de yazdığımıza göre
                        plot'a çizebiliriz.

plt. scatter (x, y) → bunun için üstte 2. bir library kullanıldı. → Scatter plot çıktı

plt. plot (x, y_line, color = 'yellow') → istersen böyle özellik de yazabilirsin.

plt. xlabel ('rainfall')

plt. ylabel ('crop amount')
```

!! Çizdiğimiz line'ın amacı 'crop amount'u predict etmek!



Evaluate the performance of a regression model

MSE and R² values

- Mean Squared Error (MSE) is a measure of the average squared difference between the predicted and true values of a regression model.
- A smaller MSE indicates a better fit, as it means that the model is making more accurate predictions of the response variable.

→ bulduğumuz sonuçlar ne kadar iyi ona bakarız

Observed value

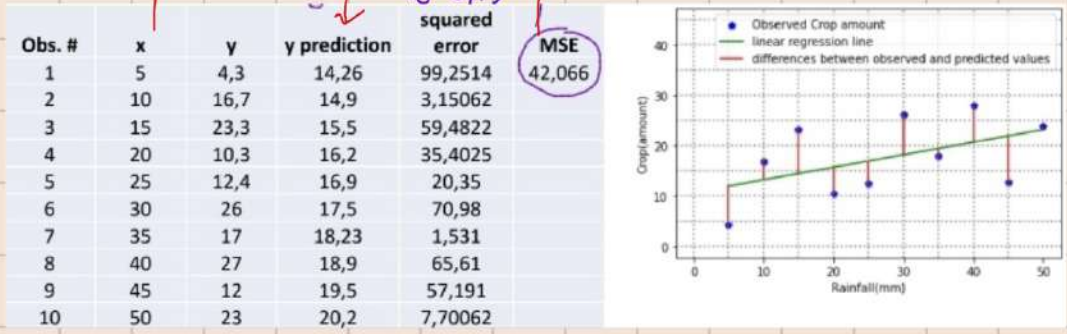
Predicted value

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - p_i)^2$$

Number of data set

The square of the difference between actual and predicted

obs. sayısı



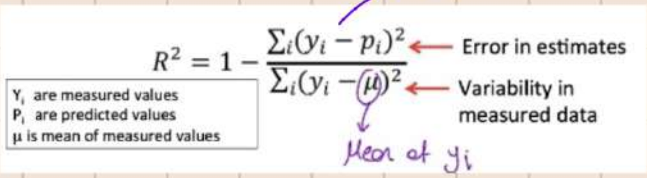
$(y - y_{pr.})^2$

Obs. #	x	y	y prediction	squared error	MSE
1	5	4,3	14,26	99,2514	42,066
2	10	16,7	14,9	3,15062	
3	15	23,3	15,5	59,4822	
4	20	10,3	16,2	35,4025	
5	25	12,4	16,9	20,35	
6	30	26	17,5	70,98	
7	35	17	18,23	1,531	
8	40	27	18,9	65,61	
9	45	12	19,5	57,191	
10	50	23	20,2	7,70062	

→ Bu bize sayılar başka bir şey ifade etmiyor. İyi mi kötü mü yorum yapamıyoruz.

(*) x'leri line denkleminde yerine koyup "y prediction" değerlerini buluruz.

R^2 Methed



gerçek y

$$R^2 = 1 - \frac{\sum_i (y_i - p_i)^2}{\sum_i (y_i - \mu)^2}$$

← Error in estimates

← Variability in measured data

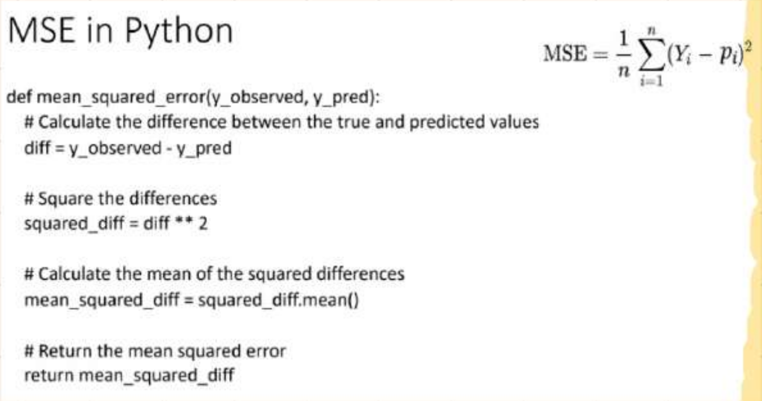
Mean of y_i

Y_i are measured values
P_i are predicted values
μ is mean of measured values

Relationship zayıf demek

$0 < R^2 < 1$

↳ 1'e yakın olması better iyi bir relationship var aralarında demek!



MSE in Python

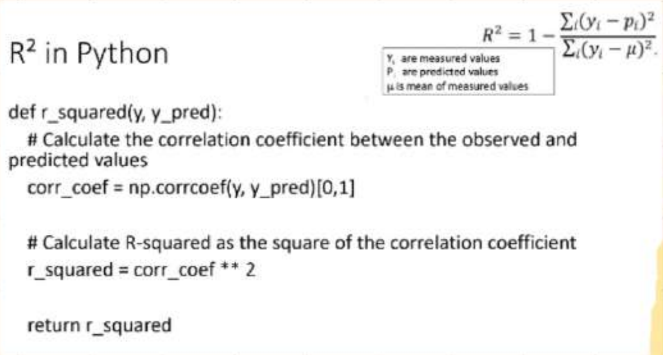
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - p_i)^2$$

```
def mean_squared_error(y_observed, y_pred):
  # Calculate the difference between the true and predicted values
  diff = y_observed - y_pred

  # Square the differences
  squared_diff = diff ** 2

  # Calculate the mean of the squared differences
  mean_squared_diff = squared_diff.mean()

  # Return the mean squared error
  return mean_squared_diff
```

önce tanımla!

```
def mean_squared_error(y_observed, y_pred):
    diff = y_observed - y_pred
    squared_diff = diff ** 2
    mean_squared_diff = squared_diff.mean()
    return mean_squared_diff
print(mean_squared_error(y, y_line))
```

→ 39.05



R² in Python

$$R^2 = 1 - \frac{\sum_i (y_i - p_i)^2}{\sum_i (y_i - \mu)^2}$$

Y_i are measured values
P_i are predicted values
μ is mean of measured values

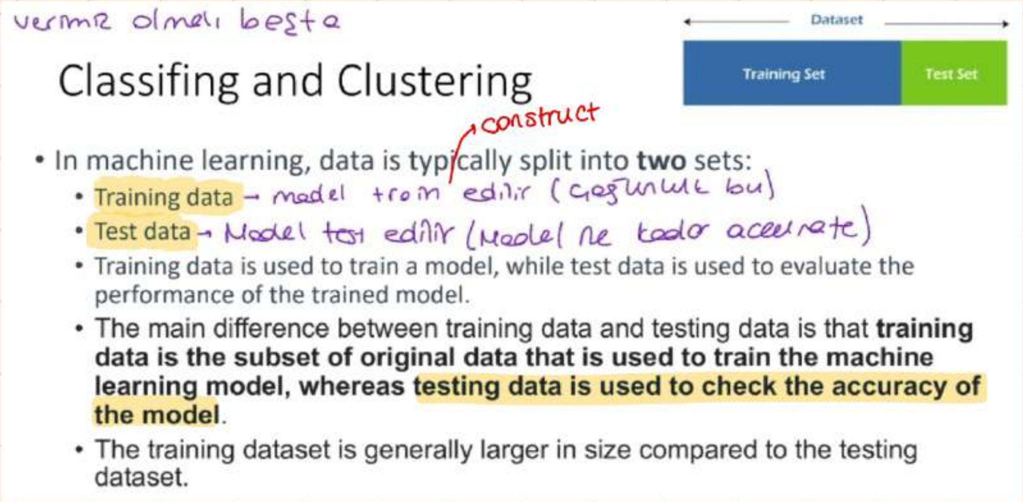
```
def r_squared(y, y_pred):
  # Calculate the correlation coefficient between the observed and
predicted values
  corr_coef = np.corrcoef(y, y_pred)[0,1]

  # Calculate R-squared as the square of the correlation coefficient
  r_squared = corr_coef ** 2

  return r_squared
```

y observed

```
def r_squared(y, y_pred):
    corr_coef = np.corrcoef(y, y_pred)[0,1]
    r_squared = corr_coef ** 2
    return r_squared
print(r_squared(y, y_line))
```

→ 0.84 (has a strong relationship)
(1'e yakın)

Classifing and Clustering



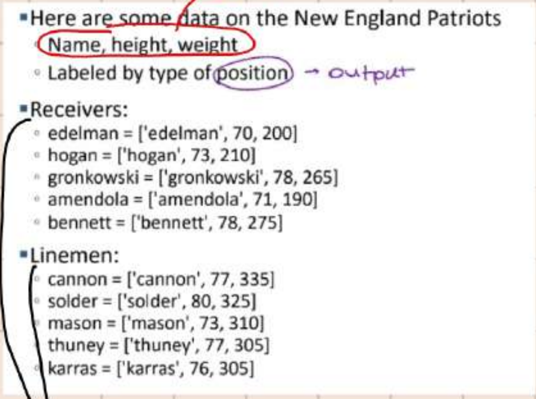
verimi olmalı beşte

Classifing and Clustering

- In machine learning, data is typically split into **two** sets: construct
  - Training data → model train edilir (çoğunluk bu)
  - Test data → Model test edilir (Model ne kadar accurate)
  - Training data is used to train a model, while test data is used to evaluate the performance of the trained model.
  - The main difference between training data and testing data is that **training data is the subset of original data that is used to train the machine learning model, whereas testing data is used to check the accuracy of the model**.
  - The training dataset is generally larger in size compared to the testing dataset.

1/ Feature seçimi önemli

4



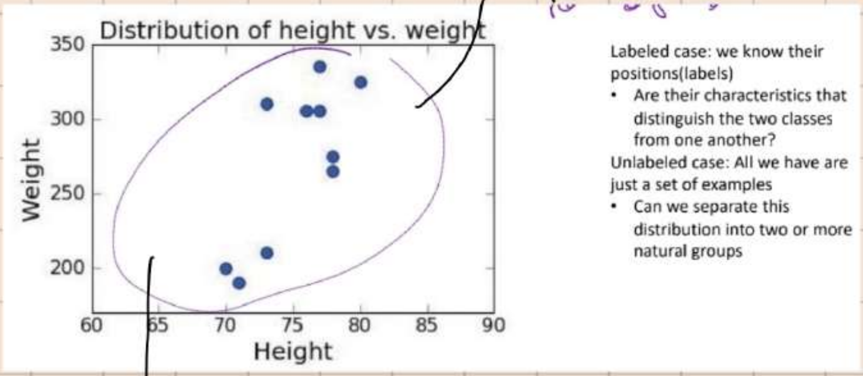
3 features (name önemsiz)

- Here are some data on the New England Patriots
  - Name, height, weight
  - Labeled by type of position → output
- Receivers:
  - edelman = ['edelman', 70, 200]
  - hogan = ['hogan', 73, 210]
  - gronkowski = ['gronkowski', 78, 265]
  - amendola = ['amendola', 71, 190]
  - bennett = ['bennett', 78, 275]
- Linemen:
  - cannon = ['cannon', 77, 335]
  - solder = ['solder', 80, 325]
  - mason = ['mason', 73, 310]
  - thuney = ['thuney', 77, 305]
  - karras = ['karras', 76, 305]

2 tane futbol position'ı var

- Unlabeled Data -

sadece data pointlerim var.



Labeled case: we know their positions(labels)

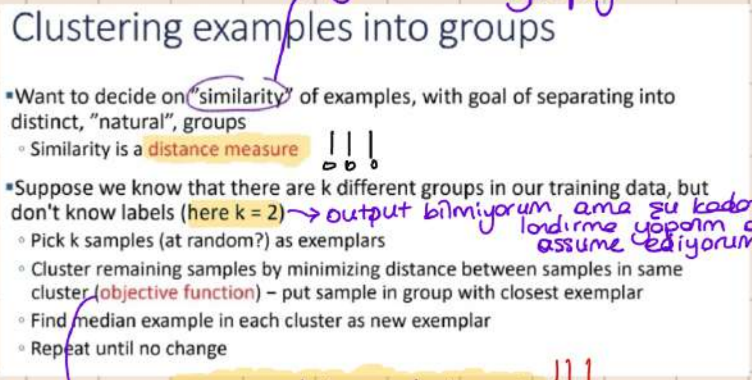
- Are their characteristics that distinguish the two classes from one another?

Unlabeled case: All we have are just a set of examples

- Can we separate this distribution into two or more natural groups

Bu karışık noktaları nasıl 2 gruba ayırabilirim?

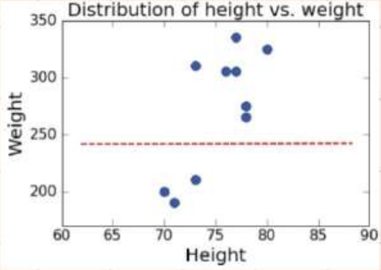
yakınlığa göre grupluyoruz



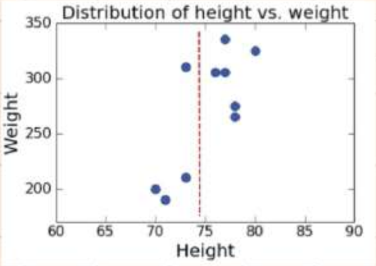
Clustering examples into groups

- Want to decide on "similarity" of examples, with goal of separating into distinct, "natural", groups
  - Similarity is a distance measure !!!
- Suppose we know that there are k different groups in our training data, but don't know labels (here k = 2) → output bilmiyorum ama şu kadar gruplandırma yaparım diye assume ediyorum.
  - Pick k samples (at random?) as exemplars
  - Cluster remaining samples by minimizing distance between samples in same cluster (objective function) – put sample in group with closest exemplar
  - Find median example in each cluster as new exemplar
  - Repeat until no change

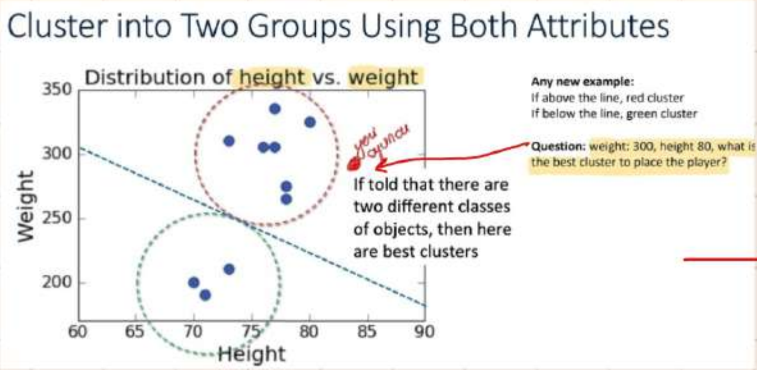
= Min the distance between !!! samples in the same cluster.



→ sadece weight'le göre gruplandırmış



→ sadece height'le göre



Cluster into Two Groups Using Both Attributes

If told that there are two different classes of objects, then here are best clusters

Any new example:
If above the line, red cluster
If below the line, green cluster

Question: weight: 300, height 80, what is the best cluster to place the player?

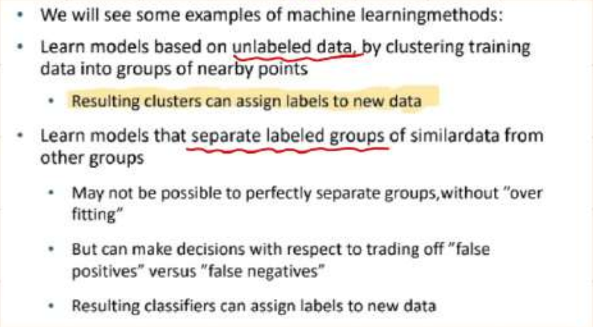
→ hem height hem weight'le göre gruplanmış

Supervised and Unsupervised Learning

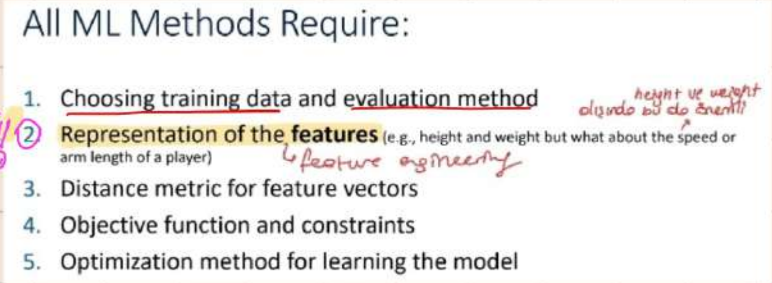
↳ we have only inputs, and group yapmaya çalışıyoruz.

↳ we have data set, we know input and output(label)

→ Machine Learning Methods

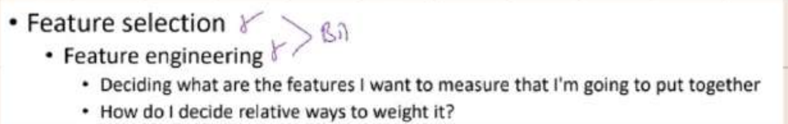


- We will see some examples of machine learningmethods:
- Learn models based on unlabeled data, by clustering training data into groups of nearby points
  - Resulting clusters can assign labels to new data
- Learn models that separate labeled groups of similardata from other groups
  - May not be possible to perfectly separate groups,without "over fitting"
  - But can make decisions with respect to trading off "false positives" versus "false negatives"
  - Resulting classifiers can assign labels to new data



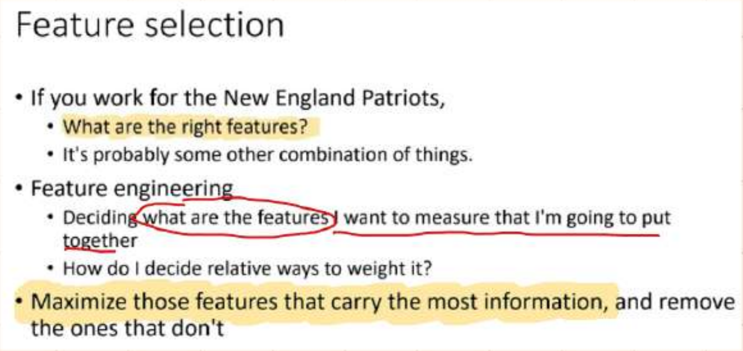
All ML Methods Require:

1. Choosing training data and evaluation method
2. Representation of the **features** (e.g., height and weight but what about the speed or arm length of a player)
   ↳ feature engineering
   height ve weight olduğunda bu da önemli
3. Distance metric for feature vectors
4. Objective function and constraints
5. Optimization method for learning the model



- Feature selection
  - Feature engineering
    - Deciding what are the features I want to measure that I'm going to put together
    - How do I decide relative ways to weight it?

Bil

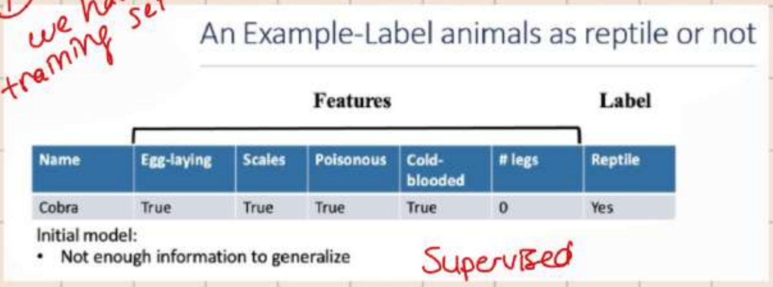


Feature selection

- If you work for the New England Patriots,
  - What are the right features?
  - It's probably some other combination of things.
- Feature engineering
  - Deciding what are the features I want to measure that I'm going to put together
  - How do I decide relative ways to weight it?
- Maximize those features that carry the most information, and remove the ones that don't

1

we have training set



An Example-Label animals as reptile or not

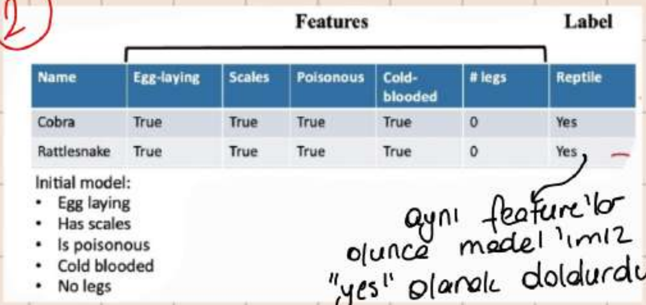
Features | Label

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes

Initial model:
- Not enough information to generalize

Supervised

2



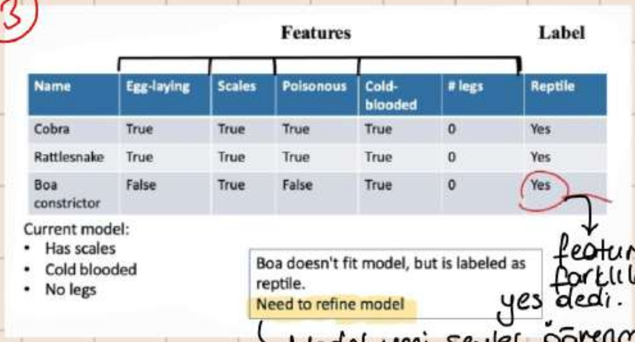
Features | Label

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes

Initial model:
- Egg laying
- Has scales
- Is poisonous
- Cold blooded
- No legs

aynı feature'lar olunca model'imiz "yes" olarak doldurdu

3



Features | Label

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes

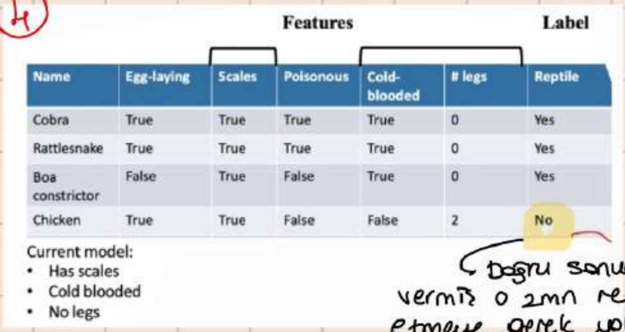
Current model:
- Has scales
- Cold blooded
- No legs

Boa doesn't fit model, but is labeled as reptile.
Need to refine model

feature'lar farklılaştı ama yes dedi.

Model yeni şeyler öğrenmeli

4



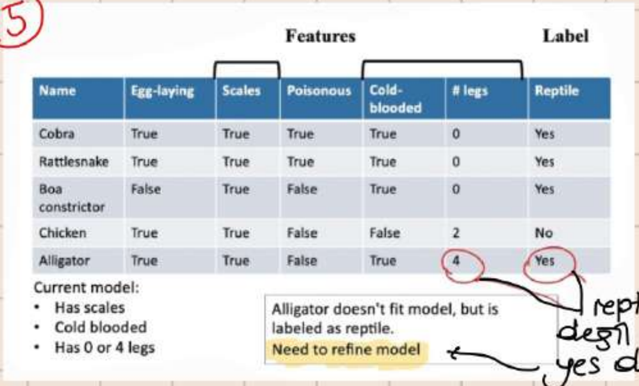
Features | Label

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No

Current model:
- Has scales
- Cold blooded
- No legs

Doğru sonuç vermiş o zmn refine etmeye gerek yok model'i.

5



Features | Label

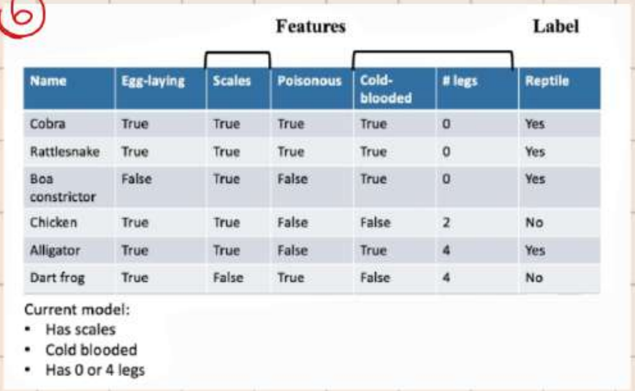
Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No
Alligator	True	True	False	True	4	Yes

Current model:
- Has scales
- Cold blooded
- Has 0 or 4 legs

Alligator doesn't fit model, but is labeled as reptile.
Need to refine model

reptile değil ama yes demiş.

6

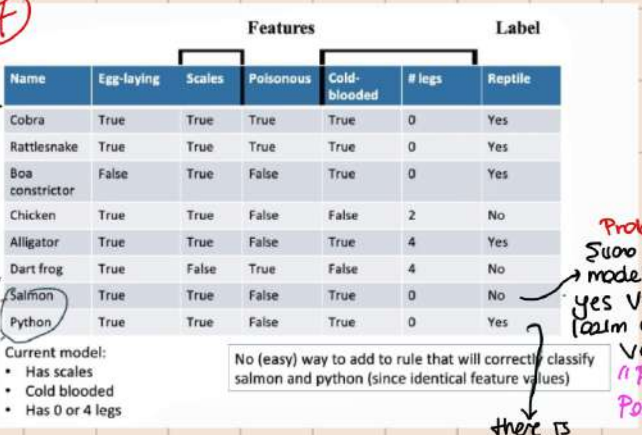


Features | Label

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No
Alligator	True	True	False	True	4	Yes
Dart frog	True	False	True	False	4	No

Current model:
- Has scales
- Cold blooded
- Has 0 or 4 legs

7



Features | Label

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No
Alligator	True	True	False	True	4	Yes
Dart frog	True	False	True	False	4	No
Salmon	True	True	False	True	0	No
Python	True	True	False	True	0	Yes

Assume it is training set

test set

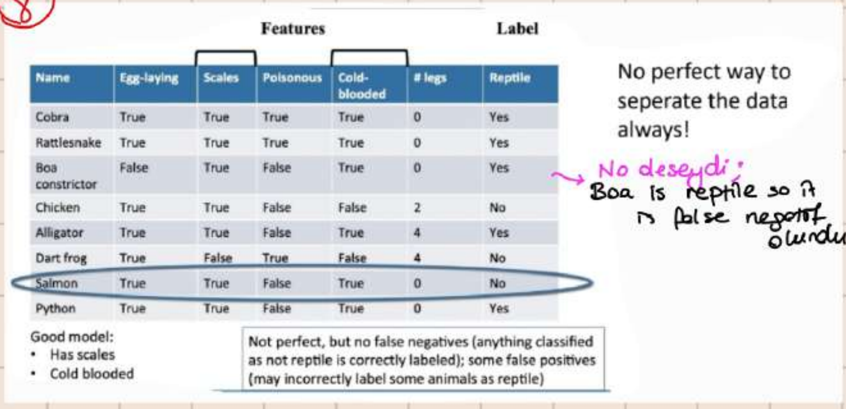
Current model:
- Has scales
- Cold blooded
- Has 0 or 4 legs

No (easy) way to add to rule that will correctly classify salmon and python (since identical feature values)

Problem! Şuan kodaki model'e göre yes vermesi lazım ama no vermiş. "False Pozitiv"

there is no problem

8



Features | Label

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No
Alligator	True	True	False	True	4	Yes
Dart frog	True	False	True	False	4	No
Salmon	True	True	False	True	0	No
Python	True	True	False	True	0	Yes

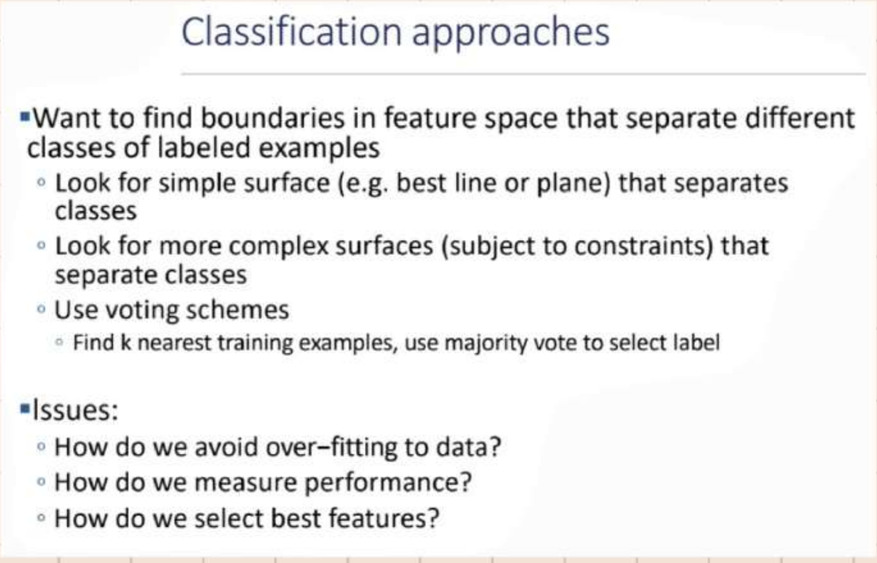
Good model:
- Has scales
- Cold blooded

Not perfect, but no false negatives (anything classified as not reptile is correctly labeled); some false positives (may incorrectly label some animals as reptile)

No perfect way to seperate the data always!

No deseydi: Boa is reptile so it is false negatif olurdu

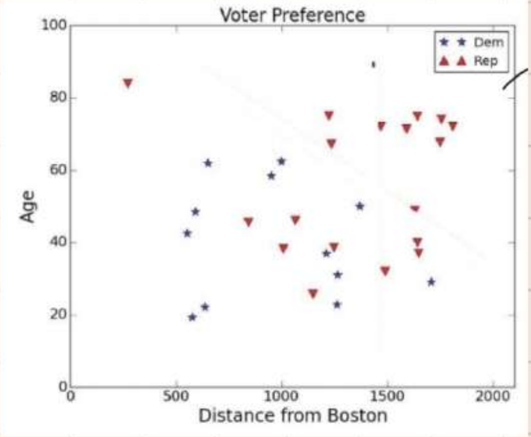
Reptile → +
non-Reptile → −



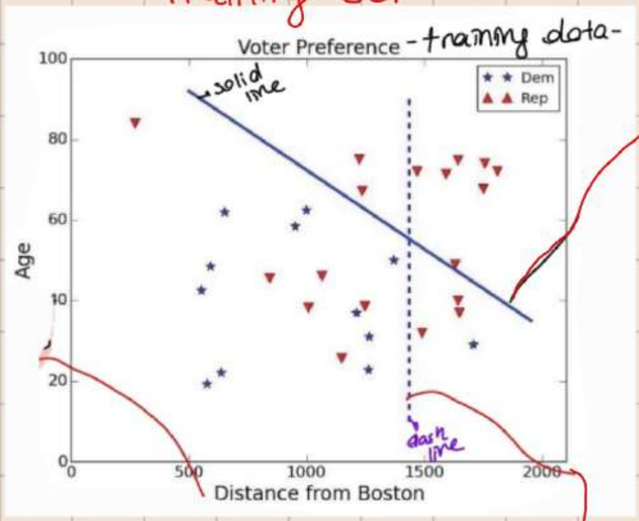
Classification approaches

- Want to find boundaries in feature space that separate different classes of labeled examples
  - Look for simple surface (e.g. best line or plane) that separates classes
  - Look for more complex surfaces (subject to constraints) that separate classes
  - Use voting schemes
    - Find k nearest training examples, use majority vote to select label
- Issues:
  - How do we avoid over-fitting to data?
  - How do we measure performance?
  - How do we select best features?

Output: Demokrat mı cumhuriyetçi mi? (Age ve distance from Boston verilerine göre output verilmiş.)



- Training Set -



üstünde kalanlar Rep
altında kalanlar dem

Sağ tarafı
Rep sol tarafı Dem
olacak

Hangi line daha accurate:

"Dash line" için;

sağ tarafındaki rep'leri saydık

sol tarafındaki dem'leri saydık

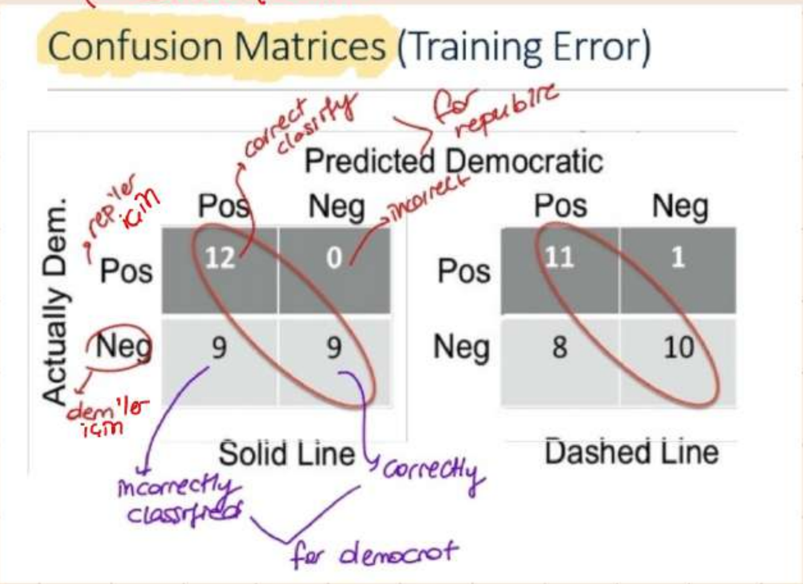
$$\frac{11 \text{ rep correct} + 10 \text{ dem correct}}{\text{Total num. of data (30)}} = \boxed{\frac{21}{30}}$$

"Solid line" için;

$$\frac{9 \text{ rep correct} + 12 \text{ dem correct}}{30} = \boxed{\frac{21}{30}}$$

EŞİT!

→ Classifaction



Confusion Matrices (Training Error)

Predicted Democratic

Solid Line

Actually Dem.	Pos	Neg
Pos	12	0
Neg	9	9

Dashed Line

Actually Dem.	Pos	Neg
Pos	11	1
Neg	8	10

correct classify

incorrect

for republic

rep'ler için

dem'ler için

incorrectly classified

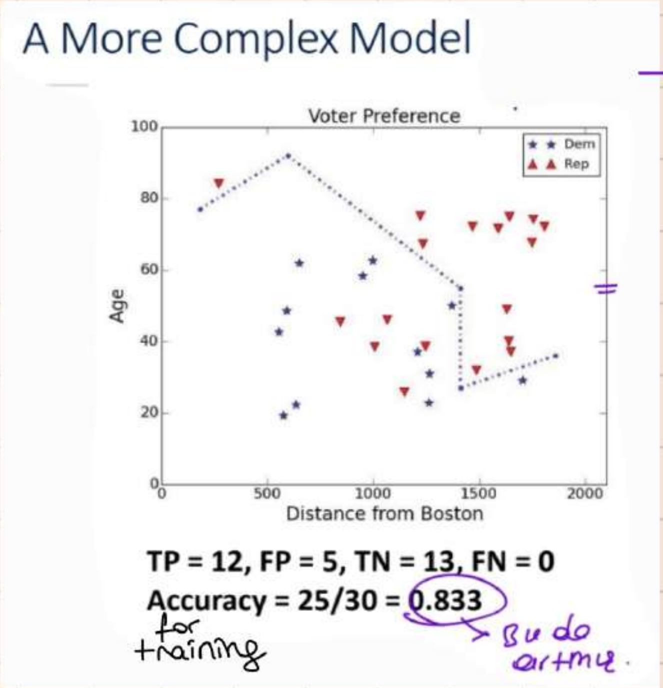
correctly

for democrat

Bunu bil!

Accuracy :

$$= \boxed{\frac{\text{true positive} + \text{true negative}}{\text{hepsi}}}$$



A More Complex Model

TP = 12, FP = 5, TN = 13, FN = 0

Accuracy = 25/30 = 0.833

for training

→ Bu da artmış.

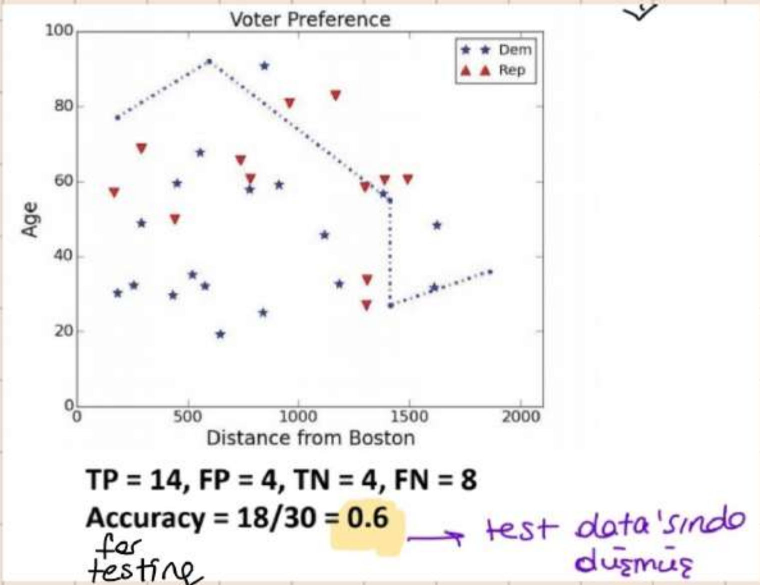
→ New model with dash lines

=

"Daha iyi" gözüküyor accuracy de daha fazla ama çok model odaklı. Bu yüzden training data da iyi sonuçlar verse de testing data'da bu kadar iyi sonuç vermez.

"Overfitting" bir model!

test data'sında uygulayınca accuracy düşmüş!



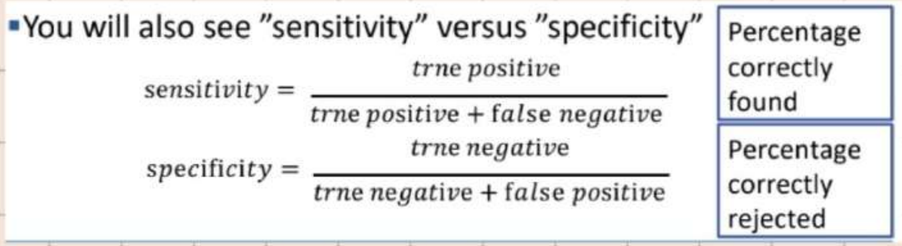
TP = 14, FP = 4, TN = 4, FN = 8

Accuracy = 18/30 = 0.6

for testing

→ test data'sında düşmüş

- Training accuracy → Training data özelinde
- Test accuracy → New data setinde (unseen) nasıl bir perfomans göstereceğini önemser.



- You will also see "sensitivity" versus "specificity"

$$sensitivity = \frac{trne\ positive}{trne\ positive + false\ negative}$$

Percentage correctly found

$$specificity = \frac{trne\ negative}{trne\ negative + false\ positive}$$

Percentage correctly rejected

select features carefully always!

Final sınavında

↳ k-means ve k nearest neighbor method ⇒ Çok ÖNEMLİ! → by hod'ik çözüm

Clustring eksik