

Numpy

3 main benefits of numpy array over a python list:

① Less memory

② Fast

③ Convenient

! What is numpy?

It is a Python extension module that provides efficient operation on arrays of homogeneous data. It allows Python to serve as a high-level language for manipulating numerical data, much like IDL, MATLAB, Fortran.

a = np.array([1, 2, 3], [4, 5, 6])

* a.ndim (Kaç boyutlu?)
→ 2

* a.shape (Sütun, satır) Demek ki 2 boyutlu
→ (2, 3)

* a.dtype (Kapladığı yes)
→ dtype('int16') → 16 bit
→ 4 byte

* a.itemsize (Her bir elementin kapladığı yes)
→ 4

NumPy → Numerical Python

Subject:

Date: _____ / _____ / _____

* a.size (İçindeki eleman sayısı)

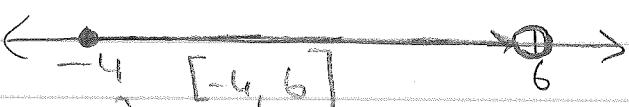
* np.arange (start, stop, step)

Belirtme 2sen 0'dan başlar

np.arange(6)

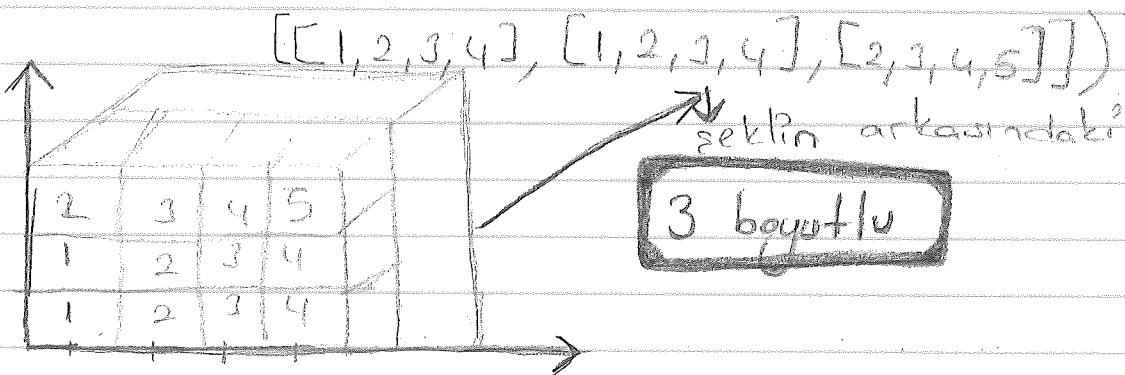
→ array([0, 1, 2, 3, 4, 5])

* np.arange(-4, 6)



→ array([-4, -3, -2, -1, 0, 1, 2, 3, 4, 5])

* a = np.array ([[[1, 2, 3, 4], [1, 2, 3, 4], [2, 3, 4, 5]]])



* np.arange(-4, 6, 3) → -4 ile 6 arası 3'ler 3'ler sayı

→ array([-4, -1, 2, 5])

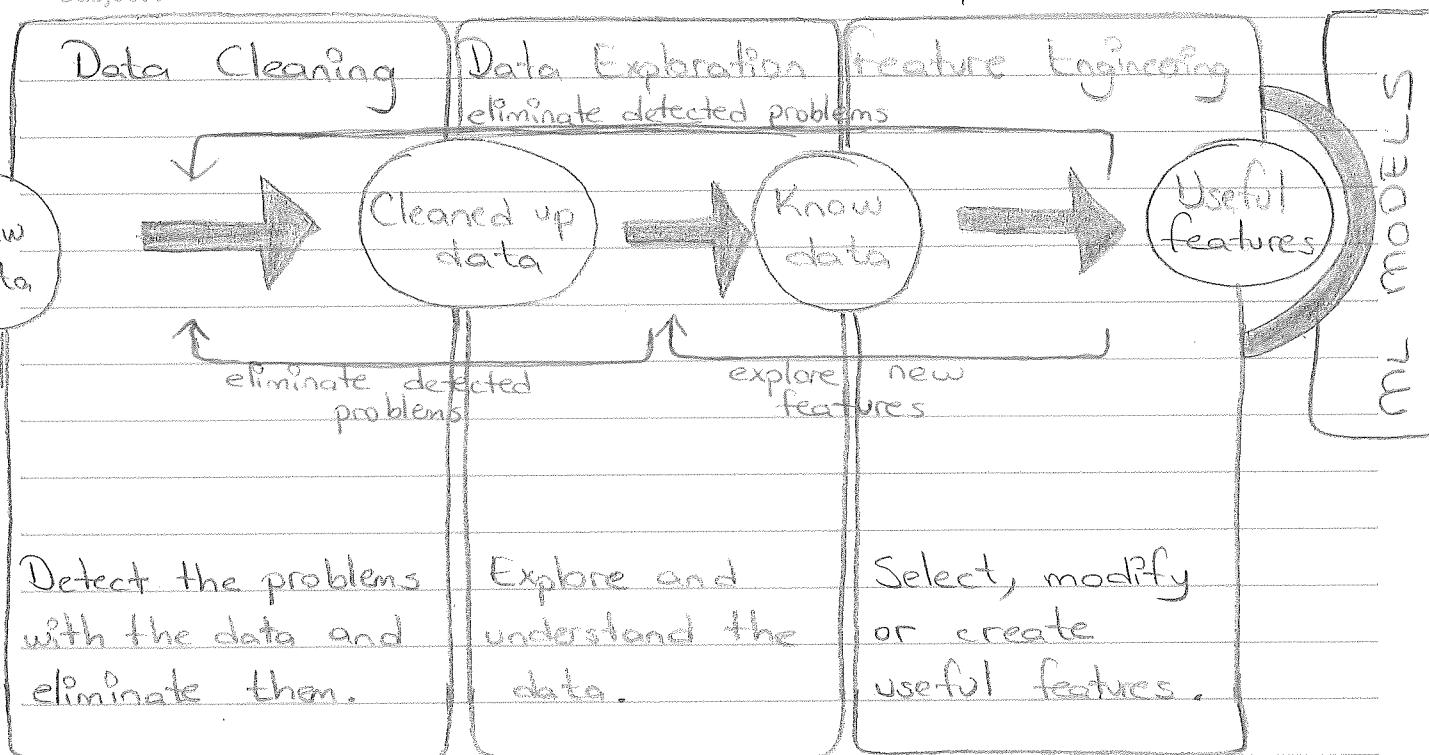
* np.linspace(1, 10) → Start-stop dahil 50 eşt parçaya böler.
(default olarak)

np.linspace(1, 10, 10) → 10 parçaya böle diye özetlikle belirttilik
9 molas artı, 10 parçaya böldü.

Subject:

Sınavlar

Date:/.....



ÖZELLİKLER

Aşağıdaki tablodan

	Float	Float	F...
Cat1					
Cat2					
Cat3					
Cat4					
Cat5					

→ Özelliğin yapısına göre kedi olduğunu tahmin eder ve aklı var. Bu machine learning'in temeli.

- Numpy is a Linear Algebra Library for Python.
- (Kare veya küp yoktur, tek boyutlu dir, bu yüzden linear sebile kullanılır.)

Numpy ile sistem yapışımında bunu usayarak bir array oluşturuyoruz. Bir kelime için 100 boyutlu bir array olusur. Bu yüzden kullanacığınız kütüphane çok hızlı olur. Bu yüzden Numpy tercih edilir.

Elementler tek tek
değişmeme gerek yok

Subject

LIST

* $a = [1, 2, 3]$

$[q * 2 \text{ for } q \text{ in } a]$

$\rightarrow [2, 4, 6]$

ARRAY

* $a = np.array([1, 2, 3])$

$a * 2$

$\rightarrow \text{array}([2, 4, 6])$

* $a = [1, 2, 3]$

$b = [4, 5, 6]$

$[q+r \text{ for } q, r \text{ in } \text{zip}(a, b)]$

$\rightarrow [5, 7, 9]$

* $a = np.array([1, 2, 3])$

$b = np.array([4, 5, 6])$

$a+b$

$\rightarrow \text{array}([5, 7, 9])$

List

* Ram'de daha çok yer kaplar. Aynı veri daha az yer kaplar.

* Daha yavaş

* $[1, 3.2, \text{True}, \text{"Ahmet"}, 5]$

Birden fazla veri tipi tutabılır.
Bu yüzden daha yavaş

* `dtype`'na bakamaz. Her element farklı türde olabilir.

! Pandas da her tür veri kabul eder.

Numpy Array

* Daha hızlı

Bir veri tipi tutar. Bu yüzden hızlı. (Ya hepsi str, ya hepsi bool...)

* `dtype`'a bakabilir. Her element aynı türde olabilir.

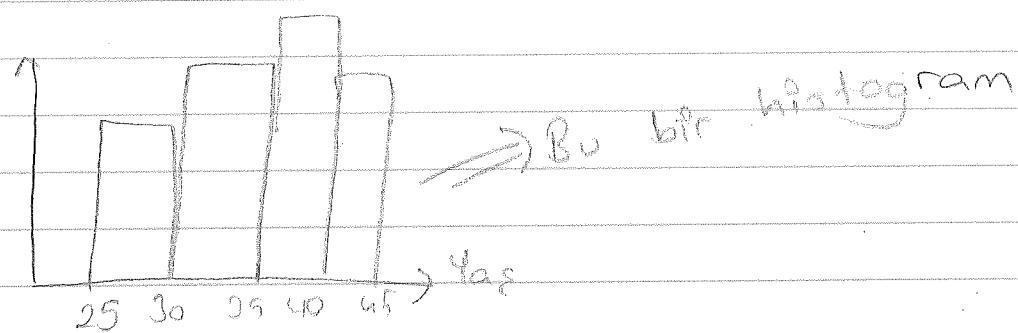
Subject:

Linspace / zeros / ones

* np.linspace(1, 10, dtype=int) \rightarrow 1 ile 10 arasındaki 50 parçaya böl ve hepsi int yap.

* np.linspace(1, 10, dtype=str) \rightarrow str'e çevirir.

* np.linspace(0, 10, 25, dtype=bool) \rightarrow True - False venir



* np.zeros(5) \rightarrow 5 tane 0 oluşturur.

* np.zeros((5, 3)) \rightarrow Tuple'a denkstür since kabul eder.

* np.zeros((5, 3, 2), dtype=int) \rightarrow 3 satır 2 sütün
5 derinliği olan bir array oluştur.
bool yazarsam false olur çünkü sıfır.
str yazarsam " boş satır olusur.

* np.ones(5) \rightarrow 5 tane 1 oluştur.

* np.ones((2, 4)) \rightarrow 1'inin hepsini 5 ile çarp.

* np.full(4, "sonsuz") \rightarrow içinde ne yazarsan onu doldurur.
 \rightarrow array(["sonsuz", "sonsuz", "sonsuz", "sonsuz"])

Subject:

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

Men 3 tane Renkler
Verdiğimizde 3 tane Renkler
3 renk 35x35 oluyor.

* np.full((3, 32, 32), 255) \rightarrow 32 erinde, 32 boyunda
255'lerden oluşan büyük
bir kare oluştur.
3 yaprak oluştur.

import matplotlib.pyplot as plt

% matplotlib inline

a = np.full((32, 32, 3), 255, dtype=int)

a[:, :, 0] = 0

0 lari, değiştirilecek renkleri

a[:, :, 1] = 0

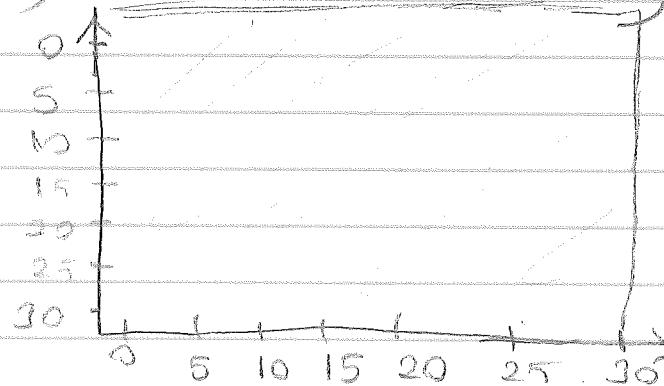
görür. En büyük değer

255 hepsinden istedim.

a[:, :, 2] = 0

gibi değiştirebilirim.

plt.imshow(a)



Subject:

Date:/..../..

* $b = np.array([36])$

b.shape

$\rightarrow (36,)$ \Rightarrow 36 tane veri var. Bu bir n shape
ni degistirmek istiyorum.

b.reshape(6,6) \Rightarrow Yine 36 elemen var ama
6 satir 6 sutun olustur.

b.reshape(3,3,4) \Rightarrow Gariplari hep 36 ediyor.
Günks b'yi o sekilde tanimla-
dik. Garip mi 36 etmisse
hata verir.

b.reshape(1,36) veya

b.reshape(36,1) tabii ki.

* c = np.array([20, 21, 22, 23, 24, 25, 26, 34, 35, 0, 1, 2, 3, 7])

c.max()

$\rightarrow 35$

c.argmax()

$\rightarrow 8$ (Max elemen indeksi)

c.min()

$\rightarrow 0$

c.argmin()

$\rightarrow 9$ (Min elemenin indexi)

, 2D ARRAY WITH Linspace

a = np.linspace(start=[0,1,2,3,4], stop=[20,21,22,23,24], num=5)

5 x 5 matris

KAHOO T!

① numpy.array(list), what it does?

It convert list to array

② np.arange(x,y,z)

↓ ↓ ↓
start stop step

③ How can you get this output? array([3,4,5,6,7,8,9,10])

np.arange(3,12,1) → yarınak sonraki değerler.

! np.arange(3,12) Bu kod hata verecektir.

Tanıne kaydınız! İstediğiniz numpy array'ı evinize

④ Write a Numpy code to create a 3x3 matrix

with values ranging from 2 to 10.

np.array([2,3,4,5,6,7,8,9,10]).reshape(3,3)

np.array(2,11).reshape(3,3)

! np.array(2,11,3) ⇒ Cevap bu olmaz.

2 ile 11 arasında 3'er aralıkları, array oluşturucu demek

Subject :

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

⑤

What is the input of this syntax?

~~array([[(1., 0., 0., 0.),~~

~~[0., 1., 0., 0.],~~

~~[0., 0., 1., 0.],~~

~~[0., 0., 0., 1.]])~~

np.eye(4)



⑥

How do you get 50 integer numbers

between 0 to 1000 (0 and 1000 included) randomly?

np.random.randint(0, 1001, 50)

↳ Randomly get 50 says.
written

⑦

reshape(array, shape)

Subject :

25.09.2021

Date : / /

* np.random.rand(5) \rightarrow 5 tane 0-1 arasında rastgele sayı çıktı.

* np.random.rand(5)*10 \rightarrow 1 ile 10 arasında 5 tane rastgele sayı

* np.random.rand(5, 3) \rightarrow 5 satır 3 sütun 0-1 arası 15 sayı çıktı.
veya deşikke ettiğim

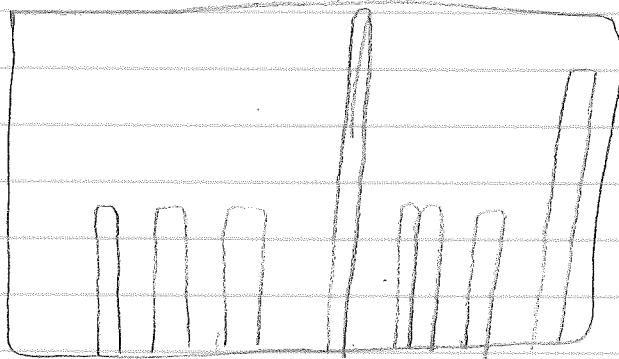
rand_float = np.random.rand(5, 3)

rand_float[0] \rightarrow ilk satırı verir.

* import matplotlib.pyplot as plt

%matplotlib inline

plt.hist(rand_float);



\rightarrow Bir grafik çıktı.

Subject:

arr = np.random.rand(99999)

Bu vadeden var. ve standart sapma olur.
Ve bu vadeden standart sapma olur.
Bu vadeden standart sapma olur.
Ortalama ortalama olur.
Yer yer de olur.

plt.hist(arr, bins=100);

abzuk sayısi
aralık sayısi

Noktalı virgül olmasa liste seklinde, kapasit grafik gösterir.

Noktalı virgül kaynasat plt.show() da yazabilirim.

→ Aynisini random ile yapalim.

np.random.rand(10) \Rightarrow 10 deger

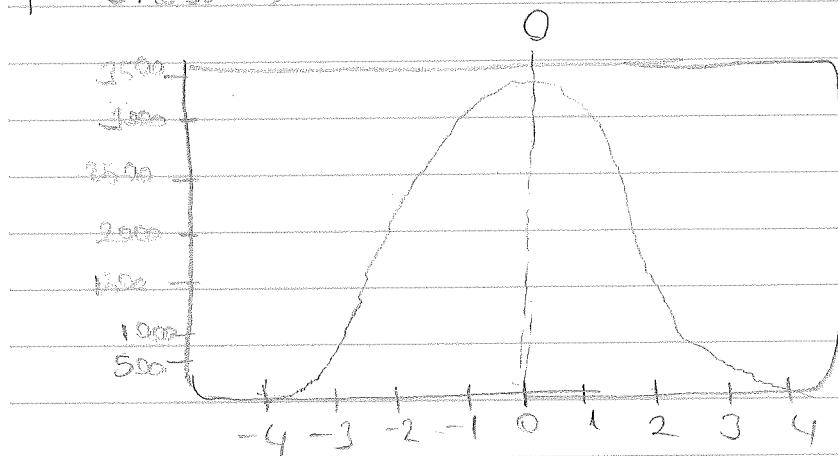
np.random.rand(5,5) \Rightarrow 25 deger

Burda 5 tane degerlerin sok büyük kismi -3, +3 araliginda dir.

* arr-n = np.random.rand(99999)

plt.hist(arr-n, bins=100)

plt.show()



$$\text{ort} = 0$$

$$\text{varyans} = 1$$

$$\text{standart sapma} = \sqrt{1}$$

$$\text{Varyans} \Rightarrow \text{arr-n.var()}$$

$$\text{Ort} \Rightarrow \text{arr-n.mean()} \\ \rightarrow 0.00292 \dots$$

$$\text{Standart sapma} \Rightarrow \text{arr-n.std()} \\ \rightarrow 1.0001230 \dots$$

Random fonk. genel olarak çok kullanılır.

Subject :

Date : / /

RANDINT

Return random integers from low (inclusive) to high (exclusive).

* np.random.randint(0, 10) → 0 ile 10 arasında rastgele sayı üret

* np.random.randint(0, 10, 5) → 0-10 arasında 5 sayı oluştur.

* np.random.randint(0, 10, size=(2, 4))
↓
8 tane sayı

* np.random.randint(0, [3, 10, 6, 2])
0-3 aralığında bir sayı
0-10 " " "
0-6 " " "
0-2 " " "
3 tane sayı
→ array([2, 2, 4, 1])

* np.random.randint([18, 60, 160, 0], [35, 100, 210, 5])
→ array([25, 64, 172, 4])
Bu aralıklarda rastgele sayı üretti.
ilk list low değerler
ikinci list upper değerler

Sınıf: 9. sınıf
Şenel Matematik

Subject:

axis=0 \Rightarrow x eklenir
axis=1 \Rightarrow y eklenir

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

Concatenation of the Arrays

ones = np.ones((4,4))

rand = np.random.randint(2, 10, size=(4,4))

np.concatenate([ones, rand]) \rightarrow ones'in sonuna,

rand'ı ekler

yani axis=0'a

göre yapar.

(Satırca göre)

np.concatenate([ones, rand], axis=1)

\rightarrow Birlesimme istenilen
süntagma göre yapar.

Splitting of the Arrays

concat = np.arange(1, 26).reshape(5, 5)

np.split(concat, 5) \rightarrow satır olarak 5'e böl

np.split(concat, [1, 3])

\rightarrow 1-3 satır arası ayrı böl

geri kalanlar ayrı

\rightarrow [2, 3]

np.split(concat, [2, 3], axis=1)

\rightarrow 2-3 satır arası ayrı
(yani sutunlar)

arasında 1, yani index=2

1, 2	3, 4, 5
6, 7	8, 9, 10
11, 12	13, 14, 15
16, 17	18, 19, 20
21, 22	23, 24, 25

$\text{axis}=0 = \text{row} = \text{record} = x \text{ axis} = \text{observation}$

$\text{axis}=1 = \text{column} = \text{feature} = y \text{ axis}$

Numpy Indexing and Selection

In this lecture we will discuss how to select elements or groups of elements from an array.

```
import numpy as np
```

```
import pandas as pd
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

```
sns.get_dataset_names()
```

`df = sns.load_dataset("diamonds")` → diamonds
 data setini
 seaborn kütüphaneinden aldık.

carat	cut	color	clarity	depth	price	x	y	z
0.23	Good	I	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	H	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	J	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	J	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	J	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	J	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	J	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	J	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	J	SI2	61.5	326	3.9	3.84	4.34
0.23	Poor	J	SI2	61.5	326	3.9	3.84	4.34

(10 satır yaklaşık 54.000 sıradan oluşur.)

Bunları numpy'a çeviriceğiz

```
nparray(df[["carat", "depth", "price", "x", "y", "z"]])
```

dataframe'i nparray'e çevirdim, bu sadece sayısal değerlerin oldu.

Subject:

diamond'a
atadık.

[i, i, i]
satır satır
step

Date:

(*) diamond = np.array (df[["carat", "depth", "price", "x", "y", "z"]])

diamond [0] \Rightarrow Sıfırıncı satır gelir.

diamond [-1] \Rightarrow Son satır gelir.

diamond [0:2] \Rightarrow 0-2 aralığında getirilir.

(*) diamond [0:2, 0:3]

↓
0-2 aralığında
satırlar

0	1	2
0	-	-
1	-	-

(*) diamond [0:2, 1] \Rightarrow 0 1 2
↓ ↓ ↓
1 - - - - -

when rightmost

(*) diamond [0, ::2] \Rightarrow 2 ser 2 ser attıyarak git

(*) carat = diamond [:, 0].reshape (len(diamond), 1)

Satırlarla ama sadece
sütunları carat + istiyorum
yok

(*) depth = diamond [:, 1].reshape (len(diamond), 1)

np.concatenate ([carat, depth], axis=1)
↓

carat ile depth'i birlestirdik

yanına eklesin alttaki
eklemesin diye axis=1

Subject:

son 3 satır
aldık

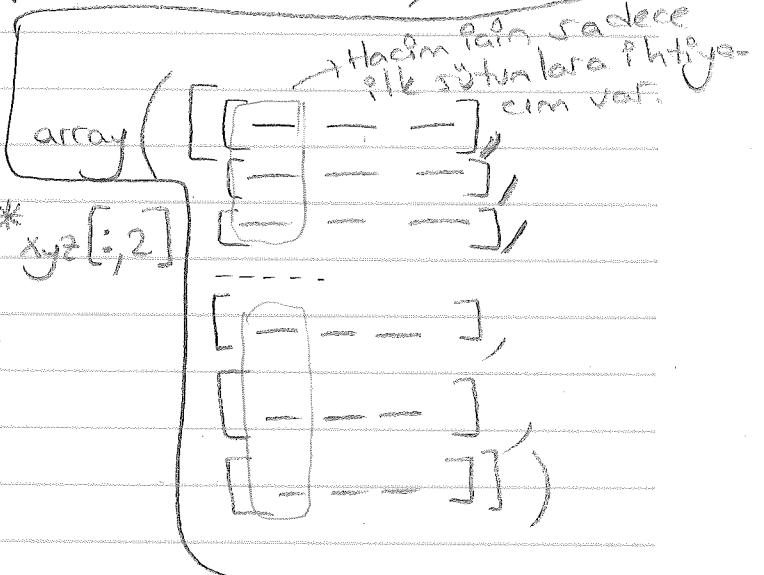
Date: / /

$xyz = \text{diamond}[:, -3:]$.reshape(len(diamond), 3)

Hacim hesaplayalım

$\text{volume} = xyz[:, 0] * xyz[:, 1] * xyz[:, 2]$

$\text{volume}.reshape(len(volume), 1)$



Indexing of 1D Arrays

import numpy as np

a = np.arange(20, 30)

→ array([20, 21, 22, 23, 24, 25, 26, 27, 28, 29])

① a[0:3] veya a[:3]

→ array([20, 21, 22])

② a[3:]

→ array([23, 24, 25, 26, 27, 28, 29])

→ Başlangıç ve bitiş indeksleri 2'ler atla

③ a[1:-2]

→ array([21, 23, 25, 27, 29])

Indexing of 2D Arrays

For DÖNGÜSÜ

for i in range(3):

 for j in range(3): → j sütun

1	2	3
4	5	6
7	8	9

[:, 1] → Satırlarda deşifrelik yapma.
1. sütunu al demek.

Subject :

Date : _____ / _____ / _____

Slicing of 2D Arrays

10	11	12	13	14
15	16	17	18	19
20	21	22	23	24
25	26	27	28	29

$[1:] [2:4]$
satır sütun

veya
 $[1:, 2:4] \rightarrow$ tek parantezde
virgülle de
ayrıca
yazabilirim.

UNIQUE

$a = np.array([10, 10, 20, 20, 30, 30])$

$q = np.unique(a)$

a
 $\rightarrow array([10, 20, 30])$

Fancy Indexing

Fancy indexing allows you to select entire rows or columns out of order, to show this, let's quickly build out a numpy array.

fancy = np.random.randint(0, 100, 10)

④ fancy[2:4]

→ array([15, 19])

⑤ fancy[[3, 5, 1, 9]]

→ array([19, 44, 35, 96])

⑥ fancy[0] = -5 ⇒ sıfırıncı eleman değiştir. -5 olur

f1 = fancy[[3, 5, 1, 9]] → fancy'in bir satırını değiştirm.

f1[0] = 1 → f1 değişt. ama fancy değişmedi



⑦ fancy2 = fancy → fancy'in bir satırını değişt.

fancy2[1] = 111 → fancy 2 değişt.
fancy değişt. ✘ ✘

⑧ fancy3 = fancy.copy()

fancy3[0] = 11111 → fancy 3 değişt.

fancy değişmedi. Çünkü kopyasını yaptı.

Subject: | Unicast - Multicast - Broadcast Date: / /

Ven' göndereceğim, bilgişayar asık mi diye sinyonu

Tek bir kişiye gitmeyen → Unicast

Birden fazla kişiye → Multicast

0 oğdakı herkese ise → Broadcast

| Peki Python'da BROADCAST nedir?

(*) $x = xyz [0:8, 0:3]$

$x [2:6, 1:3] = 0$ \Rightarrow Bu indexlerdeki her sayıya sıfır atadık.
satır sütun

(*) $x [2:6, 1:3] = [-1, 5] \Rightarrow$ Bu aralıklara -1 ve 5 kay

KAREKÖK

np.sqrt(x) → Eksi değerlere nan yazıyor.

KARESİ

np.power(x, 2) → Karesi

np.power(x, 3) → Küpü

np.power(x, 0.5) → Karekök veya $x^{**0.5}$

Subject:

Date:

SIN

np.sin(x) \Rightarrow Her elementin sinüsü olur.

np.sin(3.1428)

np.sin(np.pi) \rightarrow Gerçek π değerine böyle ulaşırız

np.sin(np.pi/2)

$\rightarrow 1$

LOG

\rightarrow Çok hızlı büyüyen

np.log(2.71) $\rightarrow \log e$

$\rightarrow 0.996\dots$

np.log(np.exp(1))

$\rightarrow 1.0$

MODULUS



arr = np.arange(0, 10)

np.mod(arr, 3) \rightarrow 1'den 10'a kadar sayıların 3 ile bölümünden kalanları eksiği olarak çıkar.

arr2 = np.random.randint(0, 100, 50)

• arr.mean()

$\rightarrow 49.56$

• arr2.min()

$\rightarrow 1$

• arr.std()

$\rightarrow 32.8256$

• arr2.sum()

$\rightarrow 2478$

• np.median(arr2)

$\rightarrow 53.0$

\rightarrow 0-100 arasında 59 sayıya geldi.

Subject :

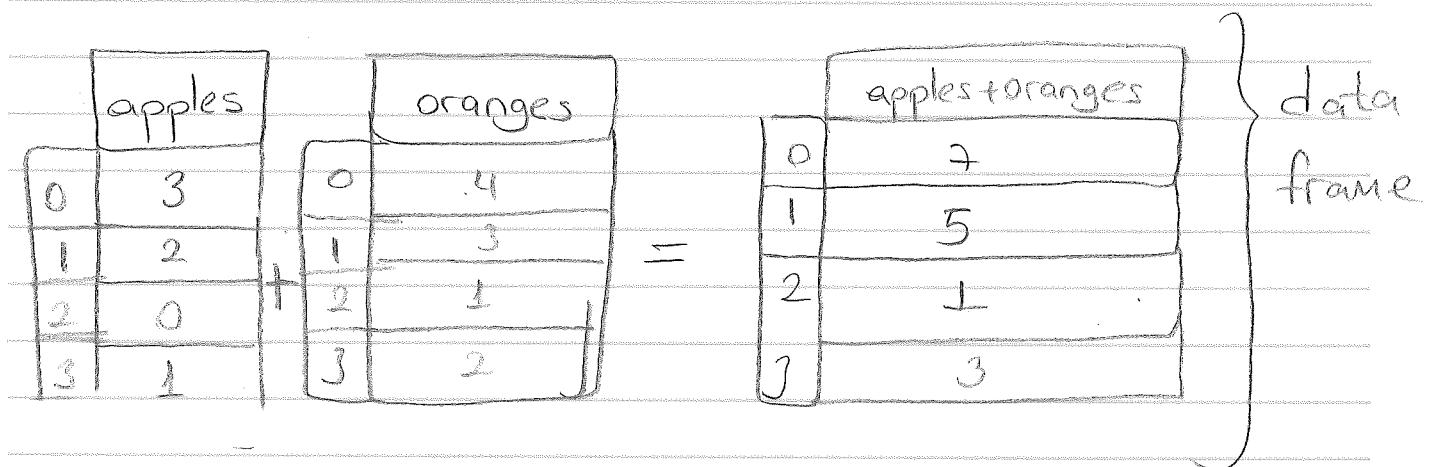
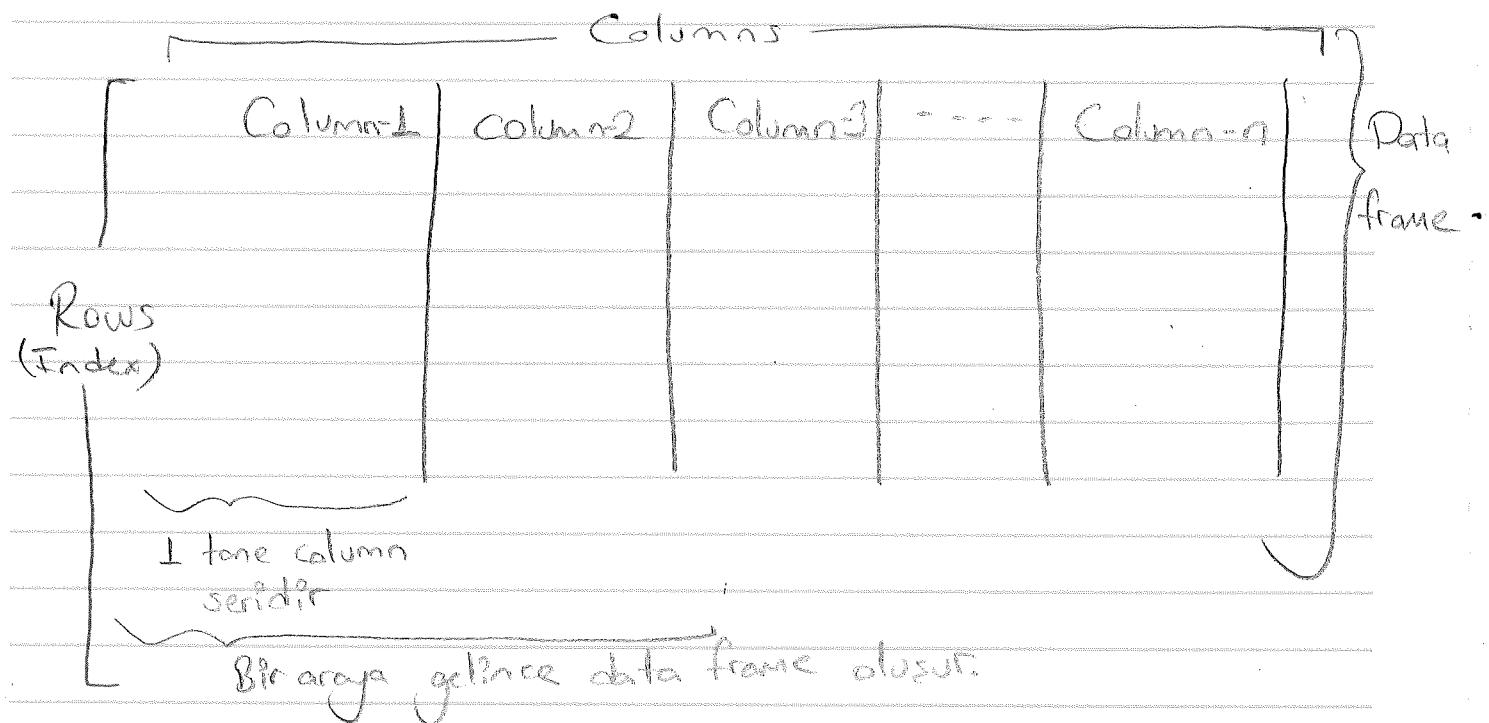
27.09.2021

Date : _____ / _____ / _____

PANDAS

Data frame deyince aklimza excell tabloları gelbilir. Ama excell'den çok daha hızlıdır.

Pandas consume more memory. Numpy is memory efficient.



Subject:

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

Import numpy as np

Import pandas as pd

ser = pd.Series([11, 21, 13, 41, 15, 61]) # np.array([1, 2, 1, 5]) gibi

(*) type(ser)

→ pandas.core.series.Series

(*) ser.size

→ 6

(*) len(ser)

→ 6

(*) ser.ndim # Bir liste olarak görünür.

→ 1

(*) ser.values

→ array([11, 21, 13, 41, 15, 61], dtype=int64)

numpy pretty. Yani iterable. Döngüsel olarak value'ları dolasabiliyim.

(*) [i for i in ser.values]

→ [11, 21, 13, 41, 15, 61]

Pandas numpy Uzerine
kullanımstır. Python kütüpha-
neleri birbirini arazinde konusur.

(*) ser2 = pd.Series([11, 21, 13, 41, 15, 61, 11, 21, 13, 41, 15, 61, 11, 21, 13, 41, 15, 61])

(*) ser2.head(10) # ilk 10'u getir.

(*) ser2.tail(3) # Son 3 eleman

22

Dear Dr. [REDACTED]
[REDACTED] [REDACTED] [REDACTED]

(*) `list("clarusway")` # Listeye atığınıza ısin iterable.
→ `[c, l, a, r, u, s, w, y]`

~~(*)~~ pd.Series(list(["classway"]))

 `data = ["paris", "london", "ankara"]`

```
index = [ "x", "y", 2 ] # label
```

`pd.Series(data, index)`

$\rightarrow x \text{ Paris}$ $y \text{ London}$ $z \text{ Ankara}$ } x, y, z index depth.

 Score = [70, 60, 90, 85, 99.5]

```
index = ["D", "E", "A", "B", "C"]
```

pd.Series(score, index) # Pandas serilne cevirdik

```
# Create a DataFrame  
df = pd.DataFrame([{'Name': 'John', 'Initial': 'J', 'Age': 30, 'Email': 'john@gmail.com', 'Weight': 195},  
                   {'Name': 'Evan', 'Initial': 'D', 'Age': 31, 'Email': 'evan@gmail.com', 'Weight': 180}],  
                   columns=['First Name', 'Last Initial', 'Age',  
                            'Email', 'Weight'])
```

Json → is like dictionary

Subject:

Date:/..../..

(*) dic = {"D": 10, "E": 20, "A": -30, "B": 99, "C": 1, } { }

pd.Series(dic) # Bir dict yapısını pandasa atık

→ D 10.0

E 20.0

A -30.0

B 99.0

C 1.1

(*) l1 = set([3, 2, 45, 4, 3, 4, 32, 6, 5, 5, 4])

a

→ {2, 3, 4, 5, 32, 45}

Küme (set) serîye atabilişiz

Set'ler unique değer alır.

pd.Series(l1 + (a))

0 32

1 2

2 3

3 4

4 5

5 45

(*) ser1 = pd.Series([1, 2, 3, 4]), index=['USA', 'Germany', 'USSR', 'Japan'])

ser2 = pd.Series([1, 2, 5, 4]), index=['USA', 'Germany', 'Italy', 'Japan'])

• ser1.index

{
Bunlarla baki tabii

• ser1.values

• ser1.sort_index()

Subject :

Date : _____ / _____ / _____

• ser2.sort_values()

ser1['deki German'], alam indexlere göre

① ser1[["Germany"]] # String olarak girip yapabiliyor.
→ 2

② ser1[["Japan"]]

→ 4

③ ser1[["Japan", "Germany"]] Birden fazla index varsa
→ Japan 4
Germany 2
liste olarak göndermemeliyiz.

④ ser1.Japan # Indexlemeyi bu şekilde yapabiliyor
→ 4
burda.

⑤ ser1 + ser2

→ Germany	4.0	} iki listede de olmayan NaN yazıyor.
Italy	NaN	
Japan	8.0	
USA	2.0	
USSR	NaN	

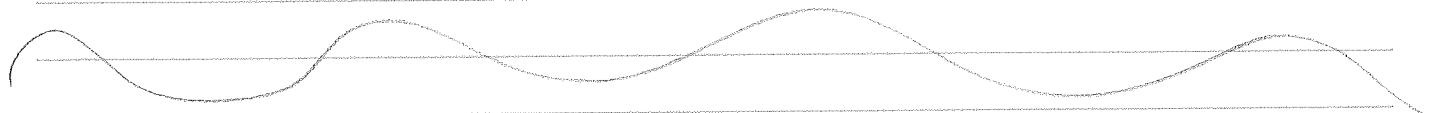
Subject:

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

④ parser = pd.Series(data=[121, 200, 150, 99],
index=["terry", "michael", "onion", "jason"])

REKLAM ALASI

⑤ np.eye(3)*33 \Rightarrow 3'e 3'lük bir matris } Kahoot



⑥ parser = pd.Series(data=[121, 200, 150, 99, 145, 120, 180],
index=["terry", "michael", "onion", "jason", "adam",
"thomas", "sack"])

⑦ parser["jason"]

\rightarrow 99

⑧ parser[["jason"]]

\rightarrow 99

⑨ parser[["onion", "adam"]]

\rightarrow onion 150

adam 145

string jason son value
dantı, numeric de değil.

⑩ parser[["onion": "adam"]]

\rightarrow onion 150

jason 99

adam 145

kısı aynı

⑪ parser[2:5]

\rightarrow onion 150

jason 99

adam 145

Attribute \Rightarrow Parentezi sagrilir.
Fonksiyonlar \Rightarrow Parentezi sagrilir.

Jelvelere
göre sırala

Subject:

Date: / /

① parser.keys()

② parser.sort_values()

③ parser.index

④ parser.sort_index()

⑤ parser.values

! Value'ler array ciktisi verit.

⑥ parser.items() # Memory'deki yesi belirtir. (Hexadecimal bir sayı)

⑦ list(parser.items()) # Liste ye cevresel ?aindeki? gorduk
ve istenek ?inak oblasabilme
aristik:

⑧ for key, value in parser.items()
print("key":key, "t value":value)

key ten buraya al ?velveleri? buraya al dedik.

⑨ "adam" in parser

\rightarrow True

⑩ 121 in parser # Value 'yi bulanadi - False verdi
 \rightarrow False

⑪ 121 in parser.values # Bu sekilde value buldur.
 \rightarrow True

⑫ parser.isin([121, 99]) # Var mi yok mu diye
value'ler de var mi
mi buluyor.

⑬ parser.json = 190 # gg yine 190 atadik,
degistirdik

! İzin sadece value'ler bakın.

Subject :

Date :/.....

① $\text{panser} \geq 180$

180'ün üstündekilerde
değisiklik yapınız.
True-false öner.

② $\text{panser}[\text{panser} \geq 180]$ # 180'ün büyüğü olan değerlerin
sıklıkını verin.

③  $m = np.arange(1, 30, 2).reshape((3, 5))$

$df = pd.DataFrame(m, columns=['col1', 'col2', 'col3', 'col4', 'col5'])$

Veriyi Column'ları
nöder
alın

$df.head(1)$ → ilk 1 satırın kaç satırı bakanı? 1
 $df.tail(2)$ → Son 2
 $df.sample(2)$ → Rastgele 2

30.09.2021

Subject :

Date : _____ / _____

PANDAS DATAFRAMES

Dataframes are the workhorse of Pandas.

- (*) df.reset_index →
- (*) df.set_index → Column'daki index'lere başlık yapıyor.
- (*) df.xs → İstener satırı getiriyor galiba
- (*) df.loc[] → Satır ve sütunları indexle çağırır.
- (*) df.iloc[] → Satır ve sütunları indexle çağırır.
- (*) df.index → Index isimleri ['a', 'b', 'c'] gibi
- (*) df.columns → column isimleri ['col1', 'col2', 'col3'] gibi
- (*) df.shape → Satır, sütun verisi (2, 3)
satır sütun
- (*) df.ndim → Shape'in içinde kaç elemen olduğunu gösterir.
 $(2, 3) \rightarrow 2$ elemen
- (*) df.info → Tablodaki sütunlar hakkında bilgi verir.
- (*) len(df) → Satır sayısını verir.
- (*) df.drop() → İstemezsen satır ya da sütunu düşür.
- (*) df.head() → İlk 5 columnu getir →
- (*) df.size → Eleman sayısını verir. (Satır x Sütun)
- (*) df.values → df'in içindeki değerler,
([1, 3, 5, 7],
[9, 11, 13, 15]) gibi

$df[df['Age'] > 35] \Rightarrow$ Age'i 35'den büyük olan öğrencileri getir.

Subject:

Date:/.....

- ④ df.keys() \Rightarrow column'ları [col1, col2...] gibi
- ④ df.col1 veya df["col1"] \Rightarrow Col1'i index olarak liste getirir.
- ④ df.describe \Rightarrow info'dan daha fazla bilgi verir.
- ④ df.value_counts \Rightarrow Hangi veri tipinden kaç tane olduğunu saydırır.
- ④ df.sum \Rightarrow ilgili sütunun toplamını toplar.
- ④ df.mean \Rightarrow Ort. alır.
- ④ df.unique \Rightarrow Unique değerleri gösterir.
- ④ df.isnull \Rightarrow Null değer mi değil mi (True-False)

Creating a DataFrame using a [dict]

`s1 = np.random.randint(2, 10, size=10)`

`s2 = np.random.randint(30, 100, size=40)`

`s3 = np.random.randint(100, 150, size=40)`

(*) `myDict = { 'var1': s1, 'var2': s2, 'var3': s3 }`

(*) `df_dict = pd.DataFrame(myDict)`

Dict'i? DataFrame'ye
geçirdik.

SLICING

(*) `df_dict[3:5]`

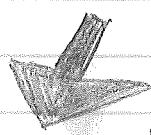
(*) `df_dict2 = df_dict[3:5]` # 3 ve 4. satırları aldık.

`df_dict2`

(*) `df_dict2.index = ["a", "b"]` # index'ı değiştirdik

(*) `df_dict2.columns = ["x", "y", "z"]` # dict2' deki
columnları değiştirdik.

(*) `pd.DataFrame(df_dict2.values, index=["c", "d"],
columns=["x", "y", "z"])`



Tek satırda index ve column değiştirdik.

Indexing, Selection and Slicing Methods and Several Attributes Using a Different

np.random.seed(101)

Böyle yazınca numpy.random
import etmeye
gerek kalmamış

df3 = pd.DataFrame(np.random.randn(5, 4),
index='A B C D E'.split('.'),
columns='W X Y Z'.split(''))

! 'A B C D E'.split('.') → Nokta ile bsl dedik

* df3['W'] # Column 'I' i verdi.
↑ Ayarla
↓ Type 'i' seridir.

* df3.W

* df3[['W']] # Type 'i' artık sen değil, DataFrame

* df3[['W', 'Y']] # Bu hata verdi, içine liste istenir

* df3[['W', 'Y']] # Birden fazla column seçtiğim

* df3.loc[:, 'W':'Y'] # W-Y arasındaki tüm sütunları
getir.

* df3.loc['B':'D', 'W':'Y'] # Satırda B-D arası, sütunda W-Y
arasını getir.

* df3[['X']] { df3['X'] + df3['Y'] } ** 2
df3[['Y']]

Seri oldugu için istedigim islemi yapabilmem
Y sütununu karsilari alip X sütunuyla
topluduk.

satır eklemek için .append
kullanılır.

Subject:

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

④ Dataframe'e yeni bir sütun eklemek:

$$df3["Result"] = df3["w"] * df3["x"] - df3["z"] ** 2$$

Result'a buları atıyorum ve yeni bir sütun ekliyorum. Önceliği sütunların değerlerini dört işlem yaparak yeni bir sütuna atadık

$$df3["Result"] = df3["w"] + df3["y"].std() - df3["z"].mean()$$

✓ std alabilirim (Tek bir değer için) ✓ orta alabilirim

Result'i düşürmek istiyorum:

df3.drop("Result", axis=1) # Değişiklik almadı çünkü kalıcı

- Aynı zamanda
1 df3.drop("Result", axis=1, inplace=True) # Değişiklik yapılmadı
2 df3=df3.drop("Result", axis=1) # Değişiklik yapıldı
iki yol

① df3.drop(["y", "Result"], axis=1, inplace=True)
df3

y'yi ve Result'i sevmemiştim atmak istiyorum. 2 sütun birde silindi

② df3.drop([{"C", "E"}, axis=0, inplace=True)

df3

$m = np.random.randint(1, 40, size=(8, 4))$

$df4 = pd.DataFrame(m, columns=[\text{"var1"}, \text{"var2"}, \text{"var3"}, \text{"var4"}])$

Subject:

Date:/.....

(*) $df4.loc[2]$ # 2. satırı getirdi.

(*) $df4.loc[2:6]$ # Birinci Fazla Satır getirildiği için dataframe olarak getirdi.

(*) $df4.index = [\text{'a'}, \text{'b'}, \text{'c'}, \text{'d'}, \text{'e'}, \text{'f'}].split()$
 $df4$

(*) $df4.loc[\text{"c"}]$

(*) $df4.loc[[\text{"c"}, \text{"f"}]]$ $df4.loc[\text{"c": "f"}, [\text{"var1"}, \text{"var2"}]]$

(*) $df4.loc[[\text{"c": "f"}][[\text{"var1"}, \text{"var2"}]]]$

Yanlış!

	var1	var2
c	13	18
d	34	30
e	20	38
f	21	28

(*) $df4.iloc[1:4, 2:4]$ # int olunca 4'te dahil olmadı.

(*) $df4.iloc[1:4, 2:3]$ # DataFrame'deki locationlara ulaşıyor.

(*) $df3.iloc[:, 2:4]$ # Tüm satırlar, 2-4 arası sütunlar gelir. (4 dahil değil)

loc \Rightarrow Veri yazılıyor

iLoc \Rightarrow Index'ı yazılıyor

İstemediklerini silmek $\Rightarrow .drop$

İsteklenenin seviyesi $\Rightarrow loc, iLoc$

Sartlı filtreleme

Subject:

Conditional selection

Date: _____ / _____ / _____

An important feature of pandas is conditional selection using bracket notation, very similar to numpy:

* df3[["x"] > 0] # True - False döndürür

* df3[df3[["x"] > 0]] # Conditional indexing
Sartlısı yapıyantları döndürür.

* df3[(df3[["x"] > 0) & (df3[["z"] > 0])] # İki sarti da
sağlayantları getirir.

* df3[df3[["z"] > 0][["w"]]] → Tablodan sadece w'yi istip-
nem. List getirir.

df3[df3[["z"] > 0][["w"]]] → Çift parantez yaparsan
dataframe getirir.

* df3[["x"] > 0]

* df3[True, False, True, False, True]
w x y z

- A True
- B False
- C True
- D False
- E True

A --- w x y z
C --- w x y z
E --- w x y z

→ (True olanları döndürdü).
Ter sayıı kacırı elemen sayısı
olmalı, 4 tane yazarak hata
alındı.)

* (df3[df3[["z"] > 0]]).iloc[2:5, :]

	w	x	y	z
D	---	---	---	---
E	---	---	---	---

Dataframe'e geldim.
0 yerde iloc olacak
kullanabilirim.

PRE- CLASS

Subject:

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

* $\text{df}['\text{Major}'] == \text{'ECOLOGY'}$

Tabloda 'Major' varsa True, değilse False döndür

* $\text{df}[\text{df}['\text{Major}']] == \text{'ECOLOGY'}$

ECOLOGY olan satırı döndür

Aynı!

* $\text{df.loc}[\text{df}['\text{Major}']] == \text{'ECOLOGY'}$

* $\text{df}[(\text{df}['\text{ShareWomen}'] > 0.5)]$

ShareWomen sütununda 0.5'ten büyük olan tüm satırları getir

* $\text{df}[(\text{df}['\text{ShareWomen}'] > 0.5) \&$

$(\text{df}['\text{Major_category}'] == \text{'Biology \& Life Science'})]$

İki şartı da sağlayanları getirir.

Ama araya & yerine | (veya) koysam her ikisini de getirir.

* $\text{df}[(\text{df}['\text{Major_category}'] == \text{'Health'})]$

$(\text{df}['\text{Major_category}'] == \text{'Biology \& Life Science'})]$

* $\text{df}[\text{df}['\text{Major_category}'].isin(['\text{Health}', '\text{Biology \& Life Science}'])]$

Daha kisa yol

$[('Health', 'Biology \& Life Science')]$

Subject :

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

outside = ['A1', 'A1', 'A1', 'A2', 'A2', 'A2'] .

?inside = [1, 2, 3, 1, 2, 3]

hier_index = list(zip(outside, ?inside))

hier_index = pd.MultiIndex.from_tuples(hier_index)

np.random.seed(101)

data = pd.DataFrame(np.random.rand(6, 2),

index=hier_index,

columns=['x', 'y'])

		x	y
A1	1	2.7068	0.628
	2	0.907	0.503
	3	0.651	-0.319
A2	1	-0.848	0.6059
	2	-2.018	0.7401
	3	0.528	-0.5890

! A2'ün 1. satırından
• y sütunu getir :
data.loc['A2'].loc[1, 'y']

data.loc['A1']

Output :

		x	y
A1	1
	2
	3

! Tek sütun getir : df['x'] (column)

! 1. satır getir : df[['x', 'y']]

! A1'den sadece x'i getir \Rightarrow data['x'][['A1']] AYNI
data.loc['A1', 'x'] (*)

* `df = pd.DataFrame(np.arange(12).reshape(3,4), columns=['A', 'B', 'C', 'D'])`

`df`

	A	B	C	D
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

* `df.drop(['B', 'C'], axis=1)`

	A	D
0	0	3
1	4	7
2	8	11

! 3'ten büyük veya eşit olan D'ler nasıl sağlanır?

`df[df['D'] >= 3]`

! Yeni bir sutun nasıl eklenir:

`data['new'] = data['city'].str[:2]`

↳ Baska bir tablodan ana tabloyu yontusla
Tablodaki 'city'ün ilk 2 harfini 'new' in
altına yaz demek

?inplace \Rightarrow Kalıcı değişiklik yapar

Subject :

Date : ____ / ____ / ____

! Index no değiştirmek :

data.set_index('section', inplace=True)

Index başlığı section oldu.

0-1-2... index no'ları

A-B-C... olarak değişti.

! Bir sütun tablodan nasıl kaldırılır?

data.drop('city', axis=1, inplace=True)

city sütununu kaldır.

Satır kaldırmak istersen axis=0

QUIZ - PRE-CLASS

! Yasi 15' ten büyük olanlar :

data[data['age'] > 15]

! country column 'da
age > 15 olanlar :

data[data['age'] > 15]['country']

! favourite_colors = red ve ages > 15 olanlar :

data[(data['age'] > 15) & (data['favourite_color'] == 'red')]

! iloc methodıyla row B 'yi seç

data.iloc[1]

↳ 1. index'teydi.

! loc methodıyla rows D ve E

columns city ve gender'i seç.

data.loc[['D', 'E'], ['city', 'gender']]

! iloc methodıyla rows D ve E

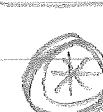
columns city ve gender'i seç.

data.iloc[[3,4], [1,2]]

↑
AYNI

Son indexler dahil *

↓
data.iloc[3:5, 1:3]



Son indexler dahil değil ✗

Subject :

Date / /

Subject:

Modül-2 De
"gereklenen"

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

Types of Machine Learning,

Supervised Learning (Dereceli Öğreme)

① Classification: Mesela kanser hücresi var = 0 Cevaplı binalar
 kanser hücresi yok = 1 varsa
 Vega X, XL, M classification

② Regression: Sayısal continuous değerler varsa regression
 (Borsa tahminleri gibi)
 Y float değerler

Unsupervised Learning

1 Dimensionality Reduction:

Ekimizde çok büyük veriler var. Binalar birbirce sıtzan olabilir.
 Bize en çok veri koordinatları sıtzuları seçmeli.
 Çok büyük verileri küçütmek

2 Clustering:

x_1 Country	x_2 Age	x_3 Salary	y Purchased	Tahmin sütunu
- - -	- - -	- - -	No	Burdakı sonclar var-yok gibi cevaplar \Rightarrow CLASSIFICATION
- - -	- - -	- - -	Yes	Float ise \Rightarrow REGRESSION
- - -	- - -	- - -	No	Cevap yoksa, yani y sütunu yoksa \Rightarrow CLUSTERING
- - -	- - -	- - -	No	

Reinforcement Learning

Günümüz sistemlerine yönelik çalışmalar. Drone, araba
 geliştirilmesi için yapılan bazı çalışmalar.

Subject: MISSING DATA Simple - Imputer (data.fillna(df.mean()))

Machine Learning

④ X. fillna (x.mean(), inplace=True)

Boslukları ort. ile doldur.



```
import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib as plt  
%matplotlib inline
```

```
data = [1, 3, 5, 7, 9]
```

```
pd.DataFrame(data, columns=["col1"])
```

Column ismi vermemektedir.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

! pd.Series(data)

Dataframe ile pandas.Series arasındaki farklılık ne?

Dataframe'de column name ve index verebiliriz.

Series'e yeni sutun gelmez.

Dataframe, pandas'ın sonucunda gelişmiş haliidir ve yenilikleri farklıdır.

Subject:

Date: /

data2 = np.arange(1, 24, 2).reshape(3, 4)

pd.DataFrame(data2, columns=["col1", "col2", "col3", "col4"],
index=["a", "b", "c"])

data2'yi DataFrame'e cevirdik. Column ve index,
isimler yazabilidik.

02.10.2021

Subject:

Date: / /

GROUP BY

`flights = sns.load_dataset("flights")`

(*) `flights.groupby("year").count()` # yila göre saydırma.

(*) `flights.groupby("year").sum()` asınca, içindeki sayıları topla.

(*) `flights[flights.year == 1949]` # flights'da yili 1949 olanları göster

- Mesela hafta içi arıza tweetlerin ort. 'sun' olabilme.
- Kategorik (label) olması.
- Yıllar da kategorikler.

Stack - Unstack

Company	Price			Price to earnings ratio (P/E)			Bu satırı sütun yapar
	Facebook	Google	Mc.	Facebook	Google	Mc.	
5-Jun-17	-	-	-	-	-	-	
6-Jun-17	-	-	-	-	-	-	

df.stack()

Company	Price			P/E		
	Facebook	Google	Mc.	Facebook	Google	Mc.
2017-06-05	-	-	-	-	-	-
	Facebook	Google	Mc.			
	-	-	-	-	-	-
2017-06-06	Facebook	Google				

! df.stack.unstack()

Eski haline geri venir.

Subject:

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

planets = sns.load_dataset("planets")

* planets.info() # Bir dataseti since ilk yapacağımız şey

* planets.describe()

* planets.mean() yerine bunu yap.

* planets.std()

* planets.describe(include="all") # Veri hakkında her şey.

* planets.isnull().sum() # null değerler.

* planets.dropna() # null değer olan satırı siler.

* planets.method.describe()

planets["method"]

süntasyon bittin
parametrenin yazıldığı
inclemek için yazın

* planets[["method"]].value_counts() { unique değerler
| idre bittiğim.
planets[["method"]].unique() } Group by 'i kimlere
yapabilirimğini görüyoruz.

* planets.groupby("method").sum()

* planets.groupby("method")["distance"].sum()

Seni dördüncü. Bir kışeli parantez daha
yaparsak df yapar.

Methodları grüpladık ve distance'ı buttik.

Indexler

Subject :



Hangi sütun gelsin
ontar, öyledik



Date : _____ / _____ / _____

std()

* planets.groupby(["method", "year"])[["distance", "orbital_period"]]

* planets.groupby("method")[[["orbital_period"]].describe()
Orbital period hakkında bilgiler

* planets.describe().T = planets.describe().transpose()
Satır ve sütunların yerlerini değiştir.

* planets[["method"]].T
2 kişeli parantez olursa T
alısır.

Sadece method sütunu
transpose kaydet.

* planets.groupby("method").max()
• min()
• mean()

$[0:7], [0:2]$

↓
Bütün satırlar

$[0:7], [0:2]$

↓
Sadece 0 ve 7. satırları al

Date: / /

Subject:

`df6 = pd.DataFrame([{"groups": ["A", "B", "C", "A", "B", "C", "A", "B", "C"],`

`"var1": [10, 27, 33, 22, 11, 99, 10, 20, 15],`

`"var2": [100, 253, 111, 262, 111, 969, 232, 343, 121]})`

(*) `df6.groupby("groups").mean()`

	var1	var2
A	24.0	198.
B	18.0	255.
C	49.0	477..

Aynı soruya
verdi.

(*) `df6.groupby("groups").agg([np.mean])`

`df6.groupby("groups").agg(["mean"])`

`df6.groupby("groups").agg(["mean", "median"])`

eval("4*3.2")

→ 12.8

eval("np.sqrt(4)")

→ 2

eval fonksiyonu, eğer str içinde bir
istem varsa yapar.

def koreal(a):

• return a*a

`eval(koreal(8))`

Büyük fonk.'u aynı
anda hesaplamayı sağlar.

Subject:

Date: _____ / _____ / _____

AGG FONK SİYONU

```
df6.groupby("groups").agg({ "var1": ( "mean", "median", "sum"),  
                            "var2": ( "std", "var")  
                          })
```

agg fonk. bunları hepsi hesaplar. Böyle bir esneklik sağlar.

np yazacaktan söyle:

{ "var1": (np.mean, np.median),
 np.std, np.var }
np'den çıktıdık

```
df6.groupby("groups").agg(['count', 'min', 'max', 'mean',  
                           np.median])
```

FILTER FONK.

```
df6.groupby("groups").filter(lambda x: x["var1"].mean() > 19)
```

Group by'a göre ort > 19 olanları getir.
(var1 > 19)

```
df6[df6.var1 > 19]
```

Eskişehir haca böyle yaparım.
Ama bu sadece filtreler
Ort. bilmiyoruz.

Subject:

TRANSFORM FONK

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

(*) df6.groupby("groups").filter(lambda x: x[["var2"]].std() > 10)

(*) df6, transform(lambda x: (x - x.mean()) / x.std())
[["var1", "var2"]].

→ Araya bu girecek.

APPLY FONK

(*) df6[["var1", "var2"]].apply(lambda x: (x - x.mean()) / x.std())
transform yerine apply kullandık
yine aynı sonucu verdi.

MAP FONK

df6[["groups"]].map({ "A": "adam", "B": "brace", "C": "cobweb"})
Aşağı parantez olmuyor, list'le satışıyor

Subject:

Date: _____

PRE-CLASS

Column pointers

(*) df.groupby('key')['data'].min()

→ key

A	0
B	1
C	2

→ data

(*) data = pd.DataFrame([{'bird': 389.0},
{'bird': 400},
{'bird': 405.0},
{'mammal': 8012},
{'mammal': 12.0},
{'mammal': 58}],
index=['falcon', 'parrot', 'eagle', 'lion',
'monkey', 'leopard'],
columns=('class', 'speed'))

	class	speed
falcon	bird	389.0
parrot	bird	400
eagle	bird	405.0
lion	mammal	8012
monkey	mammal	12.0
leopard	mammal	58.0

(*) data.groupby("class").max() # Birds and mammals' max speed

(*) data.groupby("class").mean() # Birds and mammals' avg speed

Subject:

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

Pivot Table

3 argumenten haben oder → Index
columns
values

Pivot allows you to transform or reshape data.

	date	city	temperature	humidity
0	5/1/2019	new york		
1	5/2/2019	newyork		
2		:		
3		mumbai		
4		:		
5				
6		beijing		

① df.pivot(index="date", columns="city")

	temperature			humidity		
city	beijing	mumbai	new york	beijing	mumbai	new york
date	80	-	-	-	-	-
5/1/2019	77					
5/2/2019						

→ values="humidity"

elsewhere same
humidity? geht nicht

② df.pivot_table(index="city", columns="date", aggfunc="mean")

	humidity		temperature	
date	5/1/2019	5/1/2019	5/1/2019	5/2/2019
city				
mumbai	81.5	55.5	1	- - -
new york	- -	- -	-	-

↓
aggfunc="sum"

aggfunc="count"

aggfunc="diff"

Tabloya All
ekler
Date: ↑ / /

Subject:

① df.pivot_table(index="city", columns="date", margins=True)

	humidity		temperature			
date	5/1/17	5/2/17	All	5/1/17	5/2/17	All
city						
mumbai						
new york						
All						

+ METHODS FOR PANDAS,

apply()

unique()

nunique()

value_counts()

sort_values()

MAP FONK,

train['Sex_num'] = train['Sex'].map({'female': 0, 'male': 1})

	Sex	Sex_num
0	male	1
1	female	0
2	female	0
3	female	0
4	male	1

Subject:

Date:/.....

(*) `train['Name_length'] = train.Name.apply(len)` { Harflen saydi
train.loc[0:4, ['Name', 'Name_length']] }

(*) `drinks.loc[:, 'beer_servings': 'whisky_servings'].apply(max, axis=1)`
bu indexler bu indexe tüm satırların
max'larını getir.

(np.argmax, axis=1)

max'ların hangi
satır olduğunu
 söyler

MISSING DATA

`NaN` \Rightarrow "Sayı değil" demek

`isnull()` { missing data için kullanılan fonksiyonlar
`notnull()` }

`fillna` \Rightarrow fill the missing values

`replace` \Rightarrow `df=df.replace({to_replace=np.nan, value=...})`

`dropna` \Rightarrow Drop the missing values

`interpolate` \Rightarrow Mean, median, ... hangi method yapsa
onu da kullanır

`df[columnname]=df[columnname].interpolate(method="linear")`

Subject :

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

④ $\text{df.isnull}() \Rightarrow$ Null vs True
yoksa false döndür

⑤ $\text{df.isnull().sum()} \Rightarrow$ Columnwise isnull topları

→ are a type	0
availability	0
location	1 gib?
:	:
:	:

⑥ $\text{df.isnull().sum().sum()}$

→ total (Hepsini toplar)

Filling Null Values,

$\text{df2 = df.fillna(value=0)}$

→ Null'ları 0 yap.

Drop Null Values

$\text{df.dropna(inplace=True)} = \text{df.dropna(axis=0, inplace=True)}$

⑦ $\text{data['B'].fillna(value=data['B'].mean(), inplace=True)}$

↓
B'ın ort. al

B'ı doldur

ort. ile doldur

AMACI: Sütunları kendi içinde ideal bir dağılıma sokmak

Subject:

04.10.2021

Date:/.....

MINMAX SCALER

$$x_{\text{new}} = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

\rightarrow Bu formülde aşağıdaki yazıldıkları
tekrar ettiğimiz gibi.

$$x_i \Rightarrow x'_i \in \text{kadisi}.$$

$df6[["var1"]].transform(\lambda x: x - df6[["var1"]].min()) / (df6[["var1"]].max() - df6[["var1"]].min())$

apply yazısının bu uygulaması 'apply' ile aynıdır.

Apply metodunu ile:

$df6[["var1"]].apply(\lambda x: (x - df6[["var1"]].min()) / (df6[["var1"]].max() - df6[["var1"]].min()))$

Apply ile transform arasındaki fark:

Transform'de groupby yapınca farklılık varmış.

Transform içinde lambda alabiliyor, normal fonk. alabiliyor
içine dic alır.

Apply içine tek bir fonk. alabiliyor.

Aralarındaki değişiklik groupby'da başlıyormus.

Transform içinde var ise veri varsa groupby'dan return eder.

Aynı sonucu

$df6.groupby('groups')[["var1", "var2"]].apply(np.mean)$

$df6.groupby('groups')[["var1", "var2"]].mean()$

Subject :

sum, count, min, max
calismaktı

Date: / /

df6[["Var1"]].transform([np.sqrt, np.exp, lambda x: x**2])

sum, count, min, max calismaktı

Güclü aggregate fonk. satırları

gruplar Ama transform her satırı

tek tek döndürür

FILTER

df6.groupby("groups").filter(lambda x: x["Var2"].mean() > 200)

groups'da ortası 200'den büyük olan Var2'ler

df6[df6["Var2"] > 200]

Bu da 200'den büyük var2'leri seçti ama
burda mean() yapmadı.

df6.groupby("groups").filter(lambda x: x["Var2"].mean() >
200).groupby("groups").mean()

Sırası groupby ekleyince
gruplu göreviniz.

Subject :

Date : / /

MAP and REPLACE

df6[["groups"]].map({ "A": "adam", "B": "bruce", "C": "cobweb" })

grupta satır satır doldasıyor

map yerine replace yazınca da aynı
sonuç çıkar.

map'de boş kalan yerin null ile doldurulur.
replace'de ayrı kalan

Subject :

Date : ____ / ____ / ____

TITANIC

titanic = sns.load_dataset('titanic')

① titanic.info()

② type(titanic)

③ titanic.groupby(['sex', 'pclass', 'deck'])['survived'].mean()

output →

survived	
sex	pclass
female	1
	2
	3
male	1
	2
	3

Kategorik olankor aldi.

→ Bu bir stacked table'dir.

index=["sex", "who"]
↑ da olabilir

titanic.pivot_table(values='survived', index='sex', columns='pclass')

output →

[pclass]

	1	2	3
sex			
female

male

Yukardaki gibi tablo aynı, sadace
satır-sütunlar farklı.

Subject :

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

Stack)



QUEUE



Son giren ilk eklər

ilk giren ilk eklər

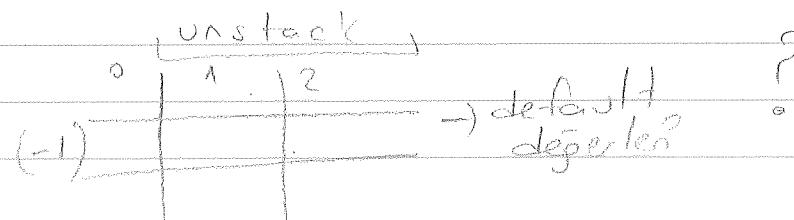
`titanic.groupby(["sex", "pclass"])[["survived"]].mean().unstack()`

`titanic.groupby(["who", "sex", "pclass"])[["survived"]].mean()
unstack().unstack()`

Onceki sayfada pivot_table koduna aggiqinc ekledik.

`titanic.pivot_table(values="survived", index=["sex", "who"],
columns="pclass", aggfunc="count")`

		pclass	1	2	3
	sex	who			
female	child		--	--	--
	women		--	--	--
male	child		--	--	--
	women		--	--	--



(*) .unique() → unique değer sayısı

(*) .del() → ilgili satırı siler

(*) .apply() } → satır satır tablodan değişiklik yapmak istenir.
(*) .map() } → satır satır tablodan değişiklik yapmak istenir.
sayılar, fonk. ve apply()

(*) .isnull() } → null mi değil mi diye sorır.

(*) .notnull() } → Boolean değer döndürür.

(*) .replace() → Map ile aynı mantıkta ama
replace()有期徒刑 uygulanır.
Değiştirme işlemi

(*) .fillna() → value doldurmak için (mesela n/a yerine)

(*) .dropna() → value düşürmek için

(*) .interpolate() → nın değerlerinin baş ve sonundaki
değerlere bakarak oralarak değer atar

(*) .value_counts() → value sayıları

(*) .sort_value → Verilen sıralar

Missing value 'lerin' ne yapabilmizi?

Removing missing values

Filling with mean values

Filling with median values

Filling with most frequent values

Hesgülüm

yapıtlacapına
dataadaki her bir

kolon tek tek
incelererek kard

ve filtre, hepsi 'nin

ayrı ayrı uygulanır
gerekçeli olur.

```
df = pd.DataFrame({'A': [1, 2, np.nan],
```

```
'B': [5, np.nan, np.nan],
```

```
'C': [1, 2, 3]})
```

(*) df.isnull()

(*) df.isnull().sum()

(*) df.dropna() → default axis=0

→ A: B C		→ NaN varsa düşürl
	0 1.0 5.0 1	

(*) df.dropna(axis=1) → NaN varsa düşürl

→ inplace=True dersen katici
değişiklik yapar.

Subject :

Date : _____ / _____ / _____

$df2 = df.append(pd.DataFrame([inp_sez]), reset_index())$

$df2$ (drop=True)

	A	B	C	D
0				NAN
1				NAN
2				NAN → Bu kodla NAN'ları, append ettiğimizde
3	NAN	NAN	NAN	NAN

$df2.dropna(thresh=2, axis=1)$

↳ Bu kadar True varsa but, diğer sütunları düşür.

$df2.dropna(axis=1) \rightarrow$ Hepsi 2'de bulundu



PRE-CLASSGroup-By

df

	key	data1	data2
0	A	0	5
1	B	1	0
2	C	2	3
3	A	2	3
4	B	4	4
5	C	5	3

min() } yaritabith
 max()

④ df.groupby('key')[['data1']].median()

'data1' in medianin

→ key

A	1.5
B	2.5
C	3.5

df

ayn' and 'data1' yaritabiyorum.
 birak

④ df.groupby('key').aggregate(['min', np.median, 'max'])

④ df.groupby('key').aggregate({'data1': 'min',
 'data1': 'max'})

✓

'data1' in 'min'
 'data2' in 'max'

Subject :

Date : _____ / _____ / _____

① df.groupby('key').std()

② # func. is True for only B, C, so A is False

```
def filter_func(x):  
    return x['data2'].std() > 4
```

```
display('df', "df.groupby('key').std()",  
       df.groupby('key').filter(filter_func))
```

→

	key	data1	data2	df.groupby('key').std()	df.groupby('key').filter(..)
0	A	0	5		
1	B	1	0		
2	C	2	3		
3	A	3	1		
4		1	1		

	key	data1	data2
1	B	1	0
2	C	2	3
3	B	4	7
4	C	5	9

3 ayrı farklı verdi

Subject:

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

TRANSFORM

`df.groupby('key').transform(lambda x: x - x.mean())`

APPLY

`def norm_by_data2(x):`

$x['data1'] / x['data2'].sum()$

`return x`

$\text{data1}' \in \text{data2}' \text{ ye bol}$

`df.groupby('key').apply(norm_by_data2)`



Basına `display('df', '---')` yazınca original
df ile birlikte gösteriliyor.

SPECIFYING THE SPLIT KEY

`L = [0, 1, 0, 1, 2, 0]`

`display('df', "df.groupby(L).sum()")` ?

Aşağıda

ANY PYTHON FUNCTION

`df2=df.set_index('key')`

\mapsto map gibi?

`df2.groupby(str.lower).mean()`

	data1	data2
a	1.5	4.0
b	2.5	3.5
c	2.5	6.0

MAP

* train['Sex_num'] = train.Sex.map({'female': 0, 'male': 1})
 train.loc[0:4, ['Sex', 'Sex_num']]

	Sex	Sex-num	
0	male	1	→ Sex columnun ismine
1	female	0	Sex-num yap ve
2	female	0	male → 1 female → 0 yap.
3	female	0	
4	male	1	

APPLY

* train['Name_length'] = train.Name.apply(len)

train.loc[0:4, ['Name', 'Name_length']]

→ Name_length sütunu eklenir ve isminin harf uzunluğunu
yaz.

ilk 4 satırı getir ve
Name ve Name_length sütunlarını
getir

* import numpy as np

train['Fare ceil'] = train.Fare.apply(np.ceil)

train.loc[0:4, ['Fare', 'Fare ceil']]

* train.Name.str.split(',') .head()

→ 0 [, , ,]
1 [, , ,]

Subject:

APPLY for SERIES

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

```
def get_element(my_list, position)
    return my_list[position]
```

```
train.Name.str.split(',').apply(get_element, position=0).head()
```

```
train.Name.str.split(',').apply(lambda x: x[0]).head()
```

Aynısı.

APPLY FOR DF

```
drinks.loc[:, 'beer_servings': 'wine_servings'].apply(
    np.argmax, axis=1)
```

Her sütbindung max'ları
column'ıni gelir.
dörek max: yagarsak
her sütbindung max
değerler gelir.

APPLYMAP

```
drinks.loc[:, 'beer_servings': 'wine_servings'].applymap(float)
```

↓
Tüm değerler floata
dönüyor.

HANDLING NULL VALUES

`df.isnull()` → True - False

`df.isnull().sum()` → Sütun toplamları (null olmayan)

`df.isnull().sum().sum()` → Her kisinin toplamı (Bir değer)

Filling Null Values

`df.fillna(value=0)` → Null'ları 0 ile doldur.

Filling Null Values with a previous value

`df4 = df.fillna(method='pad')`

→ Onceki değeri gösterir ve doldur.

filling Null Values with next value

`df5 = df.fillna(method='bfill')`

④ `df6 = df.fillna(method='pad', axis=1)`

→ Sol taki değeri Nan'a yazar

'pad' yerine 'bfill'
yazarsan 'sol taki' değer
yazar

Filling Different Values in Null in Different columns

```
df8 = df.fillna ({'society': 'abcd',
                  'balcony': 'defg'})
```

By column lardeki null laru buntan
yaz.

Filling Null Value with the mean of a column

```
df9 = df.fillna (value=df['balcony'].mean())
                ↓
                balcony deki null laru
                mean ite datadfr.
                ↑
                max()
                min()
```

DROPNA

↑ Null geren row lar
bigrdler.

* df10 = df.dropna()

* df11 = df.dropna (how='all') }

df12 = df.dropna (how='any')

REPLACE

* df12 = df.replace (to_replace=np.nan, value=875665)

* df13 = df.replace (to_replace=3.0, value=5.0)

Subject :

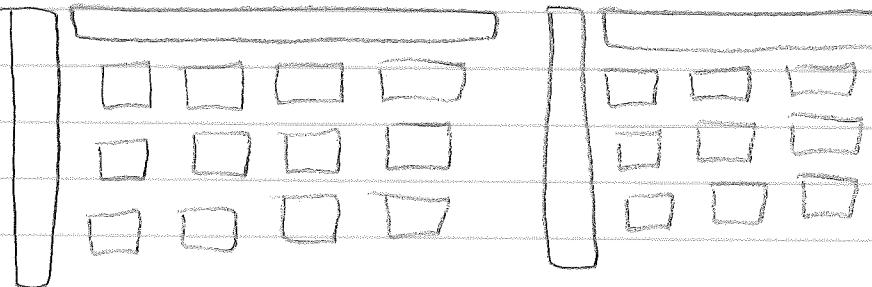
Date : / /

INTERPOLATE

`df['balcony'] = df['balcony'].interpolate(method='linear')`

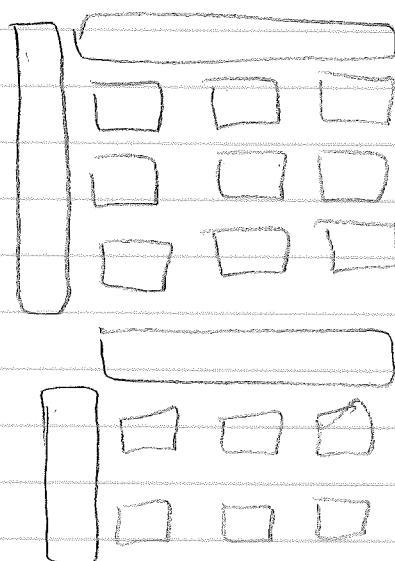
Hangi methodu.interpolate onu yarar?

MERGE - JOIN - CONCAT - APPEND



Tablolari yan yana eklenerek için : ① Merge
(axis=1) ② Join

Merge > Join



Tablolari alt alta eklenerek için (Stacking) : ① Concat
(axis=0) ② Append

Concat > Append

merge = pd.concat([df1, df2], axis=1).reset_index()

Subject:

inner = pd.merge(df1, df2)

inner = pd.merge(df1, df2, on='ID')
Merge islemi yapildi
ID column'una sahip
df1 ve df2

df1 (left)

ID	Col-1	Col-2	Col-3	Col-4
0	1	1	6	11
1	2	2	7	12
2	3	3	8	13
3	5	4	9	14
4	9	5	10	15

apple
orange
banana
kiwi
strawberry

df2 (right)

ID	Col-A	Col-B	Col-C
0	1	8	12
1	1	9	13
2	3	10	15
3	5	11	17

apple
orange
banana
kiwi

Output →

ID	Col-1	Col-2	Col-3	Col-4	Col-A	Col-B
0	1	1	6	11	apple	8
1	3	3	8	13	banana	10

df1.merge(df2, on='Col1', how='outer')

col-1'yeinden
birlesitir.

(Ornek olarak sonunda
deger hepini birlestir.

* pd.merge(df1, df2, on='ID')

↳ Merge istemini ID
yeinden yap

ID	Col-1	Col-2	Col-3	Col-4	X	Col-A	Col-B	Col-C
0	1	1	6	11	apple	8	12	apple
1	1	1	6	11	apple	9	13	orange
2	3	3	8	13	banana	10	15	banana
3	5	4	9	14	strawberry	11	17	kiwi

Subject :

Date : _____ / _____ / _____

Suffixes and Merging on Columns that are unique to each Dataframe :

pd.merge(df1, df2, suffixes=['-l', '-r'])

left_index=True,

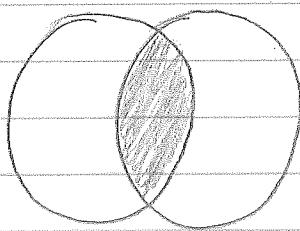
right_index=True)

Merge yapilan sutulara
-l koydu

-r koydu

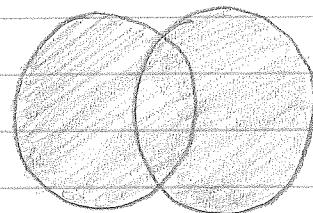
ID-l Col-1 Col-2 Col-3 Col-4-l ID-r Col-A Col-B Col-4-r

JOIN



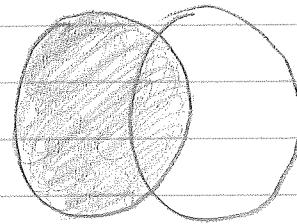
`pd.merge(df1, df2, on="col-name")`

INNER JOIN



`pd.merge(df1, df2, on="col-name", how="outer")`

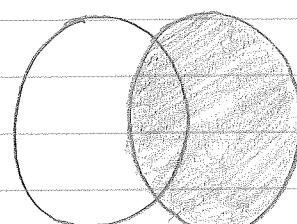
OUTER JOIN



LEFT JOIN

`pd.merge(df1, df2, on="col-name", how="left")`

(Boş olan yerlere NaN yazar)



RIGHT JOIN

`pd.merge(df1, df2, on="col-name", how="right")`



, suffixes = ['l', 'r']

! Tüm kodların sonuna ekleyebilirsin

Subject :

ERRORS

Date : / /

* pd.merge(df1, df2, on='Col_1')

Key Error verir çünkü bu column iki tabloda ortak değil

* pd.merge(df1, df2, on='Col_4')

Col_4 için tablonun birinci str bittiği int olarak deşifstirdi.

ValueError verdi

* pd.merge(df1, df2, on='ID', suffixes=['_L', '_R'],
left_on='Col_2',
right_on='Col_1')

MergeError. Çünkü left_on veya right_on 'ID' ya da
birini kullanamıyor.

Subject:

df.join():

PK data da da column
ay n' isimlerde suffix ve
varsayı ile column
rsuffix ile departure
isimlerde departure
date: date column

df1.join(df2, on='id', lsuffix='_l',
rsuffix='_r')

Default join is left

df1.join(df2, on='id', how='inner',
lsuffix='_l',
rsuffix='_r')

merge 'de yapılırsa
gibi yapabiliyoruz.

CONCAT(), df.concat()

Side-by-side and stacking Dataframes on top
of each other

* pd.concat([df1, df2]) \Rightarrow Default axis=0

This stacks the dataframes

* pd.concat([df1, df2], ignore_index=True)

↓
Reset the index

Subject :

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

* pd.concat([df1, df2], axis=1)

↓
Can side-by-side by specifying axis=1

* pd.concat([df1, df2], axis=1, join='inner')

Default join='outer' but you can specify inner join

! There is no option for left or right joins.

* pd.concat([df1, df2], axis=0, join='inner')

APPEND, df.append()

Concat ile yapılan her sevi yapabilmesi

* df1.append(df2)

* df1.append(df2, sort=True)

↓
! No way to specify type of join
Can specify ignore_index=True

Subject :

Date : / /

! You can think of joining as the same thing as merge except for the keys you want to join on are actually index instead of a column

Subject :

06.10.2021

Date : .. / .. / ..

(*) df.isnull().sum() / len(df)

↓
Her bir sütundan % kaçlık boşluk olduğunu söyle.

(*) df.salary.isnull().sum()

→ 3 (salary'de 3 boşluk var demek)

(*) df.isnull().any()

↓
Sütunlarda hiç null değer var mı?
(True-false döner.)

(*) df.isnull().all()

↓
Bütün hepsi null olan kim var?
(True-false döner.)

(*) df.isnull().any(axis=1)

↓
Satırarda herhangi bir null var mı?
(True-false döner.)

Subject: df ile filtreleme

Date: / /

(*) `df[df.isnull()]`

(*) `df[df.isnull().any(axis=1)]`

Sadece null satırları getir. Diğer satırları getirmey

(*) `df[n~(df.isnull().any(axis=1))]`

Yani demek. (Yukardaki kodu tersine çevirerek)

Sadece null olmayan satırları getir.

`Fillna`

(*) `df.fillna("other")`

df'deki tüm null'ları other ile doldurdu.

(*) `df[["class"]].fillna("other", inplace=True)`

class'daki boş değerleri other'la

(*) `df[["salary"]].fillna(2000)`

(*) `df[["salary"]].fillna(df[["salary"]].mean())`

Aynısı

`sal = df[["salary"]].mean()`

`df[["salary"]].fillna(sal)`

(Salary'deki null'lari
mean ile doldur.)

yada `median()`

Subject:

Date: / /

* df[df.salary.isnull()]=10
df

* df[df.salary.isnull()]
Salary'deki null'ları getir.

* df.loc[[0,6,10], "salary"]
Bu zahmetli! Her zaman loc veremeyez.

* df.loc[df.salary.isnull(), "salary"] = -111
Salary'deki null'ları -111 atadı.
Salary'deki null'ları getir

df.loc[df.salary.isnull(), "salary"] = df.salary.mean()

Eğer hâlen sonrası
değişiklik söyleyene
salary'in mean'ini
at dedik

* df.fillna({ "class": "other",
"var1": df.var1.mean(),
"var2": df.Var2.median(),
"dept": df.dept.mode()[0] })

df.dept.mode()
→ 0

Binden for/ba mode
olabiliyor. Bütün ilk
metin at dedik.

df[["class"]].mode()

Bunu str yazdıktan sonra class python için özet kelimeler

(*) df.notna()

↳ isnan'ın tersini döndürür.

(True - False)

Yani null → false oldu.

(*) df.where(df.notna(), df.median(), axis=1)

↓
object olantları döndürmez!

(*) df.dept.fillna(df.dept.mode()[0], inplace=True)

(*) df.loc[[0,6,10], "salary"] = np.nan

↳ Salary'deki 3 yerin nüv atadık.

(*) df.groupby(["gender", "dept"])["salary"].mean()

gender ve dept'e groupby yapılık.

Salary'ın mean'ını aldık.

(*) df.salary.fillna(df.groupby(["gender", "dept"])["salary"].transform("mean"))

?

11-6

Date _____

(*) df.groupby(["gender"])[["var1"]].mean()
gender'a göre groupby yapıyoruz
var1'i metik ediyoruz.

Subject:

(Seaborn'un flight seviyesi:

Date:

* flights = sns.load_dataset("flights")

Hıa null ver?
yoktu. Kadar
ortadık.

* flights.loc[np.random.randint(0, 1144, 30), "passengers"] = np.nan
flights

İki e kader
30 tane rota de

deger sezik ve nın ortadık

* flights.passenger.plot()

Grafikte gördük (np.nan yeri değiştirmek istenildi).

* flights.isnull().sum()

Null sayıstan rota

Yukarıda 30 değere verdigimiz için 30
çokalydi. Ama 2'si aynı oldugu için
28 ekti.

* flights.passenger.fillna(method="bfill")

Nullları alttakiyle doldur

Üstekine göre doldur dersen;
method="ffill"

! Nasıl doldur düüns bakmak ıdm

* kadin sozunu .plot() ekle

Subject :

Date : ____ / ____ / ____

⑧ flights.passenger.interpolate()

Üst ve alttakının ort.'sini null degerle yazdır.

! Null'ları önce groupby ile groupby sonra doldurmak daha sagittidir.
Bunun bir data da birde fact grp yapmak gerekebilir.

(*) `df.table.plot(kind = "box")`

→ outlier obantları bu deha
iyi gösteriyor mus panşas for

(*) `sns.boxplot(x=df.table)`

→ boxplot grafiğle outlierları görüldük

(*) `df_table = df.table`

→ df'in tableıyla bir eğilimle eklenmiş

(*) `q=df.table.quantile(0.25)` (*)

↓
 Q_1

(*) `Q_3=df.table.quantile(0.75)`

(*) `df.table.quantile(0.50)`

↳ mediani veric

$$IQR = Q_3 - Q_1$$

$$lower_lim = Q_1 - 1.5 * IQR$$

$$upper_lim = Q_3 + 1.5 * IQR$$

(*) `df.table > upper_lim` (True-false döndü)

$$df.table < lower_lim$$

Subject :

Date : / /

④ $(df_table < lower_lim) . value_counts()$

Sayıdır. (Önceki kodun sonuna
buu ekledik.)

④ $not_outlier = (df_table \geq lower_lim) \&$
 $(df_table \leq upper_lim)$

not_outlier

$$91 \leq x \leq 63.0$$

Outlier haricin dekileri
isteđik.

④ $df_drop_outlier = df[not_outlier]$

Outlier'lar
drop
outlierler kalkınca kaç tane var kaldırmış
istedik.

④ $df[df.table \geq lower_lim] \& (df.table \leq upper_lim)$

Yukardaki kodun aynısı

Outlier'ları whiskers'a esitlemek istiyoruz?

④ $df_table.loc[df_table > upper_lim] = upper_lim$

Filtreleyip True olantları $upper_lim$ 'e
esitledik.

④ df_boxplot(kind="box")

İki şartente de
grafikte gösterdi.

⑤ sns.boxplot(x=df_table)

⑥ df.loc[df.table < lower_lim, "Table"] = lower_lim

df

$\stackrel{?}{\rightarrow}$
df'de deşifrelik yaptı.

⑦ sns.histplot(x=df_table)

\downarrow
Histogramla gösterdi.



Correlation'a bakmak için: df.corr()

Bir sütunun IQR'ını hesaplamak:

Mesela tablodaki age column'ın IQR'si:

$$\textcircled{*} Q_1 = \text{df.age.quantile}(0.25)$$

$$\textcircled{*} Q_3 = \text{df.age.quantile}(0.75)$$

$$\textcircled{*} \text{IQR} = Q_3 - Q_1$$

$$\textcircled{*} \text{upper-lim-age} = Q_3 + 1.5 * \text{IQR} \quad \left. \begin{array}{l} \text{Outlier} \\ \text{hesabi} \end{array} \right\}$$

$$\textcircled{*} \text{lower-lim-age} = Q_1 - 1.5 * \text{IQR}$$

Limitation and Transformation of the Outliers

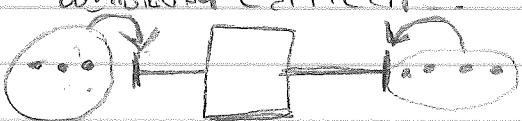
• winsorize() Method

from scipy.stats.mstats import winsorize

(*) df_win = winsorize(df.table, (0.01, 0.01))
 df_win

Outlierlerin %1'inin oldugu
 whiskersa esitledi.

(*) sns.boxplot(x=df_win)



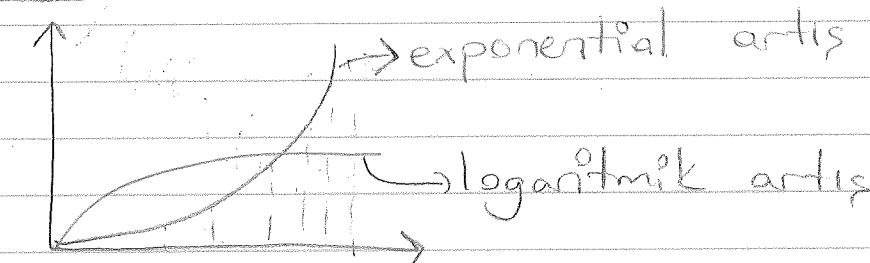
Tehlikeli bir method. Outlier olmayanlar da bozma ihtimalini var.

$\text{len}(df[df.table < \text{lower_lim}]) / \text{len}(df)$

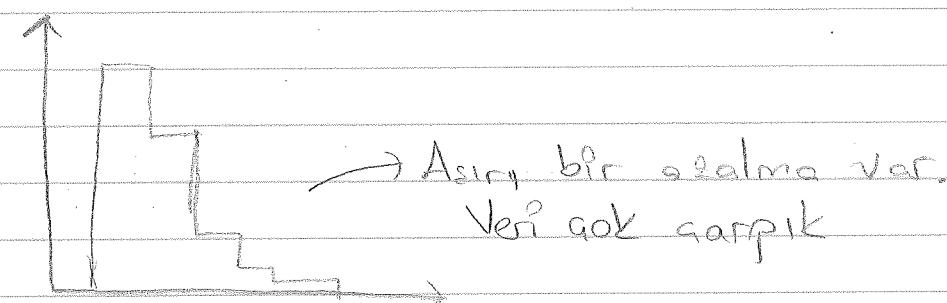
→ 0.0002...

→ Bu sayiyi alip yukarıdaki yıldızlı kırma da yaratabilirsin.

LOG



(*) df.carat.plot (kind="hist")



(*) sns.displot (df.carat, bins=15, kde=True)

(*) df[["log-carat"]] = np.log(df.carat)

Yarı bir düzgün oluşturup df'in karakterinin loguna esitledik. Veriyi normallestirdik. Çok az bir outlier kaldı.

(*) sns.boxplot (x=df.log-carat)

(*) sns.histplot (x=df.log-carat, kde=True)

Sürekli boxplot ve histogram
başka yorumları ki veri nasıl bir veri,
outlier durumu nasıl.

Subject :

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

* $df_carat_log = df_log_carat$

$Q_1_log = df_carat_log.quantile(0.25)$

$Q_3_log = df_carat_log.quantile(0.75)$

$IQR_log = Q_3_log - Q_1_log$

$IQR_log \rightarrow \text{Gen\k{e} kalan}$

* $\text{lower_lim_log} = Q_1_log - 1.5 * IQR_log$
 $\text{upper_lim_log} = Q_3_log + 1.5 * IQR_log$

* $df.loc[df.log_carat > upper_lim_log,$

\downarrow
[upper-limittler büyük
kan kişi farza ontara
upper-lim-log'aata]

"log_carat"] = upper_lim_log

* $sns.boxplot(x=df.log_carat)$

! Right skewed ise log ile,

Left skewed ise exp ile dağılımı
normalleştirilebilir.

Mesela TB'deki araba satısları :

2017 → 100 bin

2018 → 200 bin

2019 → 400 bin

2020 → 300 bin

} Yıllar birer birer artarken
fiyatlar exp ... artıyor.
Burda log uygulamamız gereklidir.

[07.10.2021]

pd.merge() veya left.merge() kullanabiliriz.

(*) pd.merge(left, right)

Ortak sütunları buldu, ortak satırları gösterdi.

(*) pd.merge(left, right, on="key")

} Ortak olan key
üzerinden birebirliği

(*) pd.merge(left, right, on=["key1", "key2"])

} 2 ortak sütun varsa
böyle.

(*) pd.merge(left, right, on="key", how="inner")

} Ortak sütundaki ortak
satırları aldı.

inner = Kesişenler

outer = Birleşenler

} how = "outer" yazarsak
unique olanları alır.

how = "left" → Soldakiin key'ine göre

how = "right" → Sağdakiin key'ine göre

Subject:

Date:/..../..

Left Join

Soldaki tür indexleri diğer sağdakinde de varsa getirir
kayıtları.



Birlestireceğimiz sütün, isimler aynıysa on kullanırız.

(*) pd.merge(df_left, df_right, left_index=True, right_index=True,
 how='outer')



) default olarak False'lar.

Index nötya göre birleştirir. Bize True'yu seç

Subject :

[09.10.2021]

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

df

(*) df.sex.str.islower()

(*) df.sex.str.islower().sample()

(*) df.sex.str.isupper().sum()

(*) df.sex.str.contains('male')

(*) df[df.sex.str.contains('female')]

(*) df.sex = df.sex.str.upper()
df.sex

Subject:

Date: / /

* df = df.replace('male', 'MALE')
df

* df['class'] = df['class'].str.swapcase()
df['class']

* Age numeric mi değil mi?

df['age'].str.isnumeric()

Str. methodu sadece str value'lerde kullanılır.
Bu yüzden hata verdi. Tipini astype ile
str yaparıyor.

df['age'].astype('string').str.isnumeric()

! NaN → Tipi float'tır.

* df.sex.apply(lambda x: x.lower() if x == 'MALE' else x)

MALE ise male yap
FEMALE ise female yap

* df.sex.apply(lambda x: x.lower() if x == 'MALE'
else x.capitalize())

(*) df.age = df.age.replace(np.nan, 'unk')

df.age.value_counts()

nan'ları 'unk' ye

değiştirerek ve
saydır.

(*) df.age = df.age.map({'unk': 'UNK'})

df.age.value_counts()



Replace → DF içerisinde egrilebilir.

Map → Sınırlı sayıda kullanılır. Dataya tam hâlin
değilsek kullanılamazız. İstem yapsa dfin
her değeri 'NaN' olsun.

(*) df.embarc_town.str.contains('bourg')

Sütundada 'bourg' geçen var mı yok mu
ona bakıyoruz.

True - False döndürür.

(*) df.embarc_town.str.endswith('town')

embarc_town sütundaki town ile
biten değer var mı?

Subject:

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

True-false döndü ama
tablo halinde NaN'lar da
için önce NaN'ları
kurtuluruz.

(*) df.embark_town.str.startswith('Ch')

"İlgili sütundaki Ch ile başlayan var mı?"

→ NaN'dan kurtul

(*) df.embark_town.fillna(df.embark_town.mode()[0],
inplace=True)

Sütun tüm NaN'larından temizlendi.

df[df.embark_town.str.startswith('Ch')]

↓ Tablodan göz

Nan'ıardan temizlenmeden önce bunu yapınca hata
verdi.

Str methodları kullanabilmek için Nan'ıardsa, kurtulmak
lazı.

(*) df[df.age.astype("string").str.startswith('2')]

Once tipini str yaptık

Sonra 2 ile başlayanlar, getir dedik.

(*) df.age.astype("string").str.find('ink')

age column', int degerler olusur. Bu yüzden önce
'str' e cevirdik.

! find ilk karakteri arayıp → Bulursa 0
bulamazsa -1 yazıyor.

Regek

re.research() re.findall()

Subject:

.count_values()

dörek hata veri

Yaz come yesterday or

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

* df.embark_town.str.findall('Queenstown')

embark_town sater str

O yıldız str yapmaya gerek yok.

! df.embark_town.str.contains("S*n")

Bu şekilde arama yapılabilir.

DATE

sns.get_dataset_names()

Data setlerinin isimleri sıktır. İstediğimi alıp
calışabiliyim.

df = sns.load_dataset('flights') → Bu çok sezik

* df.year = pd.to_datetime(df.year)

from datetime import datetime

* df.year.unique()

Sadece yıl varsa to_datetime'da formatı yazmak
göründüğü

Subject:

Date: _____ / _____ / _____

(*) df.year = pd.to_datetime(df.year,

format='%Y')

errors='ignore')

df.year

Formatı sadece '%Y' yazdık.
(Çünkü ay-gün istemedik.)
datetime formatting doldurduk.
Bu yıldan artık dt ile işlem yapabiliyim.

(*) df.year, dt.month

→ Ayı seçip görebilirim.

dt.week } tarihle ilgi herhangi
dt.weekday } bir değer böyle
dt.year } çağırabilirim.

(*) current_date = datetime.now()

current_time)

Şu anki zamanı verdi.

current_date.month)

Artık current_date üzerinden değerler
çAĞıRABİLİM.

now() } BUNLARI DA ÇAĞıRABİL-
.today() } İLE SIN Kİ



strftime formatı değişimi.

Subject:

Date:



current_date.strftime ('%d' + ' ' + 'B' + ' ' + '%Y')



date = current_date.strftime ('%d %B %Y')

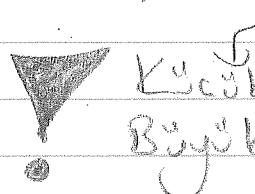
date



Yukardaki kodu date'e attadık.



datetime.strptime (date, '%d %B %Y')



Küçük harf olursa kisa isim
Büyük harf olursa uzun isim

TIMEDELTA

from datetime import timedelta

current_date.now() + timedelta(weeks=2)



2 hafta sonrasını hesapladık.



current_date.today() - timedelta(delta=3)



Bugünden 3 gün önceğini işaretlik



(current_date.now() + timedelta(delta=3)).strftime ('%d %B %Y')



Yukardaki ismini yazdıktan sonra

strftime ile formattıya yazdık

Subject:

a ve b
sülməvənələr

Date: 1/1/2023

data input - output

, usecols=['a', 'b']

(*) df = pd.read_csv('example.csv', skiprows=2)

df

2 atlayarak oku.

, nrows=3

↳ 3 satır getir

(*) titanic = pd.read_csv('titanic-train.csv', sep=';')

Separationını dəqiqədik
; ilə ayırmış. Bütün bunu
kaldırıp deşəteri ayırmayız.
(Emin deşəti)

(*) pd.read_csv('ornekCSV.csv', sep=';', index_col=0)

0'inci sülvə index
yap

(*) df.to_csv('example4')

Bulundığım klasöre daya kaydetti: 00

pwd (Dosyan nereye kaydoldu?) Bundan sonra
fazlıdan index

(*) df.to_csv('out/example5.csv', index=False) getirmiyor

pwd

out dosyama example5 klasörünü at

pip install lxml
pip install html5lib

{HTML}

Sayfaların 15'ini

Subject:

Date:

* pd.read_csv('example4.csv', index_col=0)

↓
example 4 dosyasını yazdırınak için
sonuna .csv yazdık

Okurken index=False
değeesen yazdırırken lounu
yazabilirsin aynı şey

EXCEL

* df=pd.read_excel('Excel-Sample.xlsx')

Excel bu şekilde okuyabiliğiz.

* df.to_excel('excel1.xlsx', sheet_name='sh1', index=False,
sayfa ismi)

* df.to_excel('excel2.xlsx', sheet_name='sh2')

df = pd.read_excel('excel1.xlsx')

df

excel'i okuduk

* with pd.ExcelWriter('combined.xlsx') as writer:
df.to_excel(writer, sheet_name='sh1', index=False)

df.to_excel(writer, sheet_name='sh2', index=True)

↓
combined isimli excel dosyası oluşturdu
writer objesi ile 2 tane sheet oluşturdu.

df=pd.read_excel('combined.xlsx', sheet_name='sh1')



! pip install sqlalchemy (Anaconda Jupyter'de indirmeye gerek yok. Zati var.)

! from sqlalchemy import create_engine

Bize - data base ile SQL arasındaki bağlantı kurar. Python ile SQL serverini kuracaktır.

(*) temp_db = create_engine('sqlite:///memory:')

Memory'de bize geçici bir yer oluştur demek. Hangi SQL server'a bağlanacağımıza yaradık (memory)

(*) temp_db.table_names()

Bakın temp_db içinde belli tablom var mı, yok. Tablo oluşturmanız lazım.

(*) tables = pd.read_html('https://...')

tables

Burdan table aldım.

(*) pop = table[5]

pop Tablolardan beşinciyi aldım.

İsime gerek mi diye bakıyorum.

Bir dehki kodda bu tabloyu SQL'e attım.

connection',
temp-db ile sqla

Subject:

Date:/..../..

(*) pop.to_sql(name='populations', con=temp_db)

populations ismini verdik.

(*) temp_db.table_names()

→ ['population'] → isimli table olustu.

(*) pd.read_sql(sql='population', con=temp_db)

Aritik burda normal SQL calisiyor.
SQL tablosu gibi okudu.

(*) pd.read_sql_query(sql='select Country, Population from Population', con=temp_db)

Populations tablosundan sadece
population ve country sutunlarini oku.

HTML

(*) df2 = pd.read_html('https://...')

df2

Siteden dataframe gettilik

(*) df[2]

2. tablunu oku. Bu baska bir dataframe
almus.

pip install lxml

pip install html5lib

} html dosyalarini okumak
icin bunları import ediyorum

(*) df[2].to_html('simple.html', index=False)

PRE-CLASS

Text and Time Methods

lower() ➔ Hepsi? Küçük harf yap

upper() ➔ Hepsi? Büyük harf yap

islower() ➔ Tüm karakterler küçük mü? (True-false)

isupper() ➔ " " büyük mü? (True-false)

isdigit() ➔ Rakam mı değil mi? (True-false)

isnumeric() ➔ Sayısal değer mi değil mi? (True-false)

replace() ➔ 'a' yerine 'b' yaz

split() ➔ Her string'i sunula bölg. (Mesela "-")

contains() ➔ Satır ısunu içen mi? (True-false)

strip() ➔ Helps strip whitespace (including new line) from each string

find() ➔ Returns the first position of the first occurrence of the pattern

findall() ➔ Returns a list of all occurrence of the pattern

String Method in Pandas

* df.column_name.str.upper()

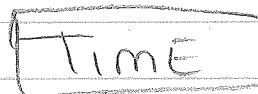
* df.column_name.str.contains('Chicken')

İlgili column 's' Zırhınla
chicken var mı?
(True - False)

* df.column_name.str.replace('I', 'Y')

Burun yerine buru kay

* df.column_name.str.replace('I', '').str.replace('Y', '')



* to_datetime() ➡ Parses many different kinds of date representations returning Timestamp object.

* strftime ➡ Convert object to a string according to a given format

* strptime ➡ Parse a string into a datetime object given a corresponding format

* timedelta ➡ Gives time difference.

Subject :

Date : _____ / _____ / _____

from datetime import datetime

* current_date = datetime.now()

- print(current_date) → 2021-07-17 08:21:32.236148

- print(current_date.date()) → 2021-07-17
↳ Sadere tarihi ver

- print(date.today()) → 2021-07-17

- print(current_date.day) → 17
· month → 7
· year → 2021

* print(current_date.strftime("%d")) → 17

("%" "B") → July

("%" "W") → 28

↳ Hafta

Subject:

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

Strptime

import datetime from datetime

str1 = '2022-12-31'

date1 = datetime.strptime(str1, "%Y-%m-%d")

print(date1) \Rightarrow 2022-12-31 00:00:00

\Rightarrow String'ın datetime
çevirildi.

print(date1.year) \rightarrow 2022
• month \rightarrow 12
• day \rightarrow 31

timedelta

from datetime import timedelta
import datetime

current_date = datetime.datetime.now()

date_after_10_days = current_date + timedelta(days=10)

Bu sonda 10 gün
soncaya git.

- ① `*read_csv()` \Rightarrow Read a comma-separated values (csv) file into DataFrame
- ② `DataFrame.to_csv()` \Rightarrow Write DataFrame to a comma-separated values (csv) file
- ③ `*read_excel()` \Rightarrow Read an Excel file into a Pandas DataFrame
- ④ `*DataFrame.to_excel()` \Rightarrow Write object to an Excel sheet
- ⑤ `*read_html()` \Rightarrow Read HTML tables into a list of DataFrame objects.

