Python for Statistical Analysis (İstatistiksel Analiz için Python)

İstatistik, verilerin toplanması, düzenlenmesi, görüntülenmesi, analiz edilmesi, yorumlanması ve sunulmasını inceler. İstatistik, veri bilimi ve makine öğrenimi için ön koşul olması önerilen bir Matematik dalıdır. İstatistik çok geniş bir alandır ama biz bu bölümde sadece en alakalı kısma odaklanacağız. Bu zorluğu tamamladıktan sonra web geliştirme, veri analizi, makine öğrenimi ve veri bilimi yoluna gidebilirsiniz.

Data(Veri)

Veri, genellikle analiz olmak üzere, bir amaç için toplanan ve çevrilen herhangi bir karakter kümesidir. Metin ve sayılar, resimler, ses veya video dahil herhangi bir karakter olabilir. Veriler bir bağlama yerleştirilmezse, bir insan veya bilgisayar için bir anlam ifade etmez. Verilerden anlam çıkarmak için farklı araçlar kullanarak veriler üzerinde çalışmamız gerekir. Veri analizi, veri bilimi veya makine öğreniminin iş akışı verilerden başlar. Veriler bazı veri kaynaklarından sağlanabilir veya oluşturulabilir. Yapılandırılmış ve yapılandırılmamış veriler vardır. Veriler küçük veya büyük formatta bulunabilir. Alacağımız veri türlerinin çoğu, dosya işleme bölümünde ele alınmıştır.

Statistics Module(İstatistik Modülü)

Python istatistik modülü, sayısal verilerin matematiksel istatistiklerini hesaplamak için işlevler sağlar. Modül, NumPy, SciPy gibi üçüncü taraf kitaplıklarına veya Minitab, SAS ve Matlab gibi profesyonel istatistikçileri hedefleyen tescilli tam özellikli istatistik paketlerine rakip olmayı amaçlamaz. Grafik ve bilimsel hesap makineleri seviyesine yöneliktir.

NumPy

İlk bölümde Python'u başlı başına harika bir genel amaçlı programlama dili olarak tanımladık, ancak diğer popüler kütüphanelerin (numpy, scipy, matplotlib, pandas vb.) yardımıyla bilimsel hesaplama için güçlü bir ortam haline geldi. Numpy,Python'daki temel kitaplıklardan biridir dilerseniz detaylı NumPy kütüphanesini inceleyelim ve uygulayalım.

Importing NumPy

NumPy kütüphanesini içe aktaralım.

# \* Creating numpy array using

## Creating int numpy arrays(int numpy dizileri oluşturma)

import numpy as np

print('numpy:', np.\_\_version\_\_)

#numpy: 1.20.3

print(dir(np))

# python liste oluşturma

python\_list = [1,2,3,4,5]

# Veri türleri kontrol

print('Type:', type (python\_list)) # <class 'list'>

print(python\_list) # [1, 2, 3, 4, 5]

two\_dimensional\_list = [[0,1,2], [3,4,5], [6,7,8]]

print(two\_dimensional\_list)  # [[0, 1, 2], [3, 4, 5], [6, 7, 8]]

#  Python listesinden Numpy(Sayısal Python) dizisi oluşturma

numpy\_array\_from\_list = np.array(python\_list)

print(type (numpy\_array\_from\_list))   # <class 'numpy.ndarray'>

print(numpy\_array\_from\_list) # array([1, 2, 3, 4, 5])

### **Creating float numpy arrays(Float numpy dizileri oluşturma)**

Bir kayan nokta veri türü parametresiyle listeden bir kayan nokta dizisi oluşturma

Örnek:

python\_list = [1,2,3,4,5]

numy\_array\_from\_list2 = np.array(python\_list, dtype=float)

print(numy\_array\_from\_list2) # array([1., 2., 3., 4., 5.])

### **Creating boolean numpy arrays(Boolean numpy dizileri oluşturma)**

Listeden bir **boole** dizisi oluşturma

numpy\_bool\_array = np.array([0, 1, -1, 0, 0], dtype=bool)

print(numpy\_bool\_array)# [False  True  True False False]

### **Creating multidimensional array using numpy(Numpy kullanarak çok boyutlu dizi oluşturma)**

Numpy dizisinde bir veya birden çok satır ve sütun olabilir

two\_dimensional\_list = [[0,1,2], [3,4,5], [6,7,8]]

numpy\_two\_dimensional\_list = np.array(two\_dimensional\_list)

print(type (numpy\_two\_dimensional\_list))

print(numpy\_two\_dimensional\_list)

<class 'numpy.ndarray'>

[[0 1 2]

 [3 4 5]

 [6 7 8]]

### **Converting numpy array to list(Numpy dizisini listeye dönüştürme)**

Bir diziyi her zaman **tolist()** kullanarak bir python listesine dönüştürebiliriz.

np\_to\_list  =  numpy\_array\_from\_list . tolist ()

print( type ( np\_to\_list ))

print( 'tek boyutlu dizi:' , np\_to\_list )

print( 'iki boyutlu dizisi' , numpy\_two\_dimensional\_list . tolist ())

### **Creating numpy array from tuple(Tuple'dan numpy dizisi oluşturma)**

#NumPy Dizisi

python\_tuple = (1,2,3,4,5)

print(type (python\_tuple)) # <class 'tuple'>

print('python\_tuple: ', python\_tuple) # python\_tuple:  (1, 2, 3, 4, 5)

#Python'da NumPy Dizisi

numpy\_array\_from\_tuple = np.array(python\_tuple)

print(type (numpy\_array\_from\_tuple)) # <class 'numpy.ndarray'>

print('numpy\_array\_from\_tuple: ', numpy\_array\_from\_tuple) # numpy\_array\_from\_tuple:  [1 2 3 4 5]

### **Shape of numpy array(Numpy dizisinin şekli)**

Şekil yöntemi, dizinin şeklini bir tanımlama grubu olarak sağlar. Birincisi satır, ikincisi sütundur. Dizi tek boyutluysa dizinin boyutunu döndürür.

Örnek:

nums = np.array([1, 2, 3, 4, 5,6,7])

print(nums)

print('shape of nums: ', nums.shape)

print(numpy\_two\_dimensional\_list)

print('shape of numpy\_two\_dimensional\_list: ', numpy\_two\_dimensional\_list.shape)

three\_by\_four\_array = np.array([[0, 1, 2, 3],

        [4,5,6,7],

        [8,9,10, 11]])

print(three\_by\_four\_array.shape)

[1 2 3 4 5 6 7]

shape of nums:  (7,)

[[0 1 2]

 [3 4 5]

 [6 7 8]]

shape of numpy\_two\_dimensional\_list:  (3, 3)

(3, 4)

### **Data type of numpy array(Numpy dizisinin veri türü)**

Veri türleri türü: str, int, float, Complex, bool, list, None

int\_lists = [-5,-4,-3, -2, -1, 0, 1, 2,3,4,5]

int\_array = np.array(int\_lists)

float\_array = np.array(int\_lists, dtype=float)

print(int\_array)

#[-5 -4 -3 -2 -1  0  1  2  3  4  5]

print(int\_array.dtype)#int32

print(float\_array)

#[-5. -4. -3. -2. -1.  0.  1.  2.  3.  4.  5.]

print(float\_array.dtype)#float64

### **Size of a numpy array(Numpy dizisinin boyutu)**

Numpy'de, bir numpy dizi listesindeki öğelerin sayısını bilmek için "size" kullanırız.

numpy\_array\_from\_list = np.array([1, 3, 5, 7,9])

two\_dimensional\_list = np.array([1, 2], [3, 4],[6, 7])

print('The size:', numpy\_array\_from\_list.size) # 5

print('The size:', two\_dimensional\_list.size)  # 3

## Mathematical Operation using numpy(Numpy kullanarak matematiksel işlem)

NumPy dizisi, tam olarak python listesi gibi değildir. Python listesinde matematiksel işlem yapmak için öğeler arasında döngü yapmalıyız, ancak numpy herhangi bir matematiksel işlemi döngü yapmadan yapmamıza izin verebilir. Matematiksel operasyon:

Toplama (+)

Çıkarma (-)

Çarpma işlemi (\*)

Bölünme (/)

Modüller (%)

Kat Bölümü (//)

Üstel (\*\*)

### **Addition(Toplama)**

numpy\_array\_from\_list = np.array([1, 2, 3, 4, 5])

print('original array: ', numpy\_array\_from\_list)

ten\_plus\_original = numpy\_array\_from\_list  + 10

print(ten\_plus\_original)

#original array:  [1 2 3 4 5]

#[11 12 13 14 15]

### **Subtraction(Çıkarma)**

numpy\_array\_from\_list = np.array([1, 2, 3, 4, 5])

print('original array: ', numpy\_array\_from\_list)

ten\_plus\_original = numpy\_array\_from\_list  + 10

print(ten\_plus\_original)

#original array:  [1 2 3 4 5]

#[11 12 13 14 15]

### **Multiplication(Çarpma)**

numpy\_array\_from\_list = np.array([1, 2, 3, 4, 5])

print('original array: ', numpy\_array\_from\_list)

ten\_times\_original = numpy\_array\_from\_list % 3

print(ten\_times\_original)

### **Division(Bölünme)**

numpy\_array\_from\_list = np.array([1, 2, 3, 4, 5])

print('original array: ', numpy\_array\_from\_list)

#original array:  [1 2 3 4 5]

ten\_times\_original = numpy\_array\_from\_list // 10

print(ten\_times\_original)#[0 0 0 0 0]

### **Modulus(Modül)**

numpy\_array\_from\_list = np.array([1, 2, 3, 4, 5])

print('original array: ', numpy\_array\_from\_list)

ten\_times\_original = numpy\_array\_from\_list % 3

print(ten\_times\_original)

#original array:  [1 2 3 4 5]

#[1 2 0 1 2]

### **Floor Division(Kat Bölümü)**

numpy\_array\_from\_list = np.array([1, 2, 3, 4, 5])

print('original array: ', numpy\_array\_from\_list)

#original array:  [1 2 3 4 5]

ten\_times\_original = numpy\_array\_from\_list // 10

print(ten\_times\_original)

[0 0 0 0 0]

### **Exponential(Üstel)**

numpy\_array\_from\_list = np.array([1, 2, 3, 4, 5])

print('original array: ', numpy\_array\_from\_list)

ten\_times\_original = numpy\_array\_from\_list  \*\* 2

print(ten\_times\_original)

original array:  [1 2 3 4 5]

[ 1  4  9 16 25]

### **Checking data types(Veri türünü kontrol etme)**

numpy\_int\_arr = np.array([1,2,3,4])

numpy\_float\_arr = np.array([1.1, 2.0,3.2])

numpy\_bool\_arr = np.array([-3, -2, 0, 1,2,3], dtype='bool')

print(numpy\_int\_arr.dtype)

print(numpy\_float\_arr.dtype)

print(numpy\_bool\_arr.dtype)

## Converting types(Türleri Değiştirme)

Numpy dizisinin veri türlerini dönüştürebiliriz

1. Int to Float

numpy\_int\_arr = np.array([1,2,3,4], dtype = 'float')

numpy\_int\_arr

#array([1., 2., 3., 4.])

1. Float to Int

numpy\_int\_arr = np.array([1., 2., 3., 4.], dtype = 'int')

numpy\_int\_arr

#array([1, 2, 3, 4])

1. Int ot boolean

np.array([-3, -2, 0, 1,2,3], dtype='bool')

#array([ True,  True, False,  True,  True,  True])

1. Int to str

numpy\_array\_from\_list.astype('int').astype('str')

#array(['1', '2', '3', '4', '5'], dtype='<U11')

## Multi-dimensional Arrays(Çok Boyutlu Diziler)

Örnek:

two\_dimension\_array = np.array([(0,2,4),(1,3,5), (6,7,8)])

print(type (two\_dimension\_array))

print(two\_dimension\_array)

print('Shape: ', two\_dimension\_array.shape)

print('Size:', two\_dimension\_array.size)

print('Data type:', two\_dimension\_array.dtype)

Çıktı:

<class 'numpy.ndarray'>

[[0 2 4]

 [1 3 5]

 [6 7 8]]

Shape:  (3, 3)

Size: 9

Data type: int32

### **Getting items from a numpy array(Numpy dizisinden öğe alma)**

Örnek:

two\_dimension\_array = np.array([[0,2,4],[3,5,7], [-1,-8,-9]])

first\_row = two\_dimension\_array[0]

second\_row = two\_dimension\_array[1]

third\_row = two\_dimension\_array[2]

print('Even nubers row:', first\_row)

print('odd row:', second\_row)

print('negative numbers row: ', third\_row)

Çıktı:

Even nubers row: [0 2 4]

odd row: [3 5 7]

negative numbers row:  [-1 -8 -9]

Örnek:2

first\_column= two\_dimension\_array[:,0]

second\_column = two\_dimension\_array[:,1]

third\_column = two\_dimension\_array[:,2]

print('First column:', first\_column)

print('Second column:', second\_column)

print('Third column: ', third\_column)

print(two\_dimension\_array)

Çıktı:2

First column: [ 0  3 -1]

Second column: [ 2  5 -8]

Third column:  [ 4  7 -9]

[[ 0  2  4]

 [ 3  5  7]

 [-1 -8 -9]]

### **Slicing Numpy array(Numpy Dizisini Dilimleme)**

Numpy'de dilimleme, python listesinde dilimlemeye benzer

Örnek:

two\_dimension\_array = np.array([[1,2,3,1],[4,5,6,2], [7,8,9,4]])

first\_two\_rows\_and\_columns = two\_dimension\_array[0:4, 0:4]

print(first\_two\_rows\_and\_columns)

Çıktı:

[[1 2 3 1]

 [4 5 6 2]

 [7 8 9 4]]

### **How to reverse the rows and the whole array?([::])**

(Satırları ve tüm diziyi nasıl tersine çevirebilirim?) i

two\_dimension\_array[::]

#array([[1, 2, 3, 1],

      # [4, 5, 6, 2],

       #[7, 8, 9, 4]])

### **Reverse the row and column positions(Satır ve sütun konumlarını ters çevirin)**

two\_dimension\_array = np.array([[1,2,3],[4,5,6], [7,8,9]])

two\_dimension\_array[::-1,::-1]

array([[9, 8, 7],

       [6, 5, 4],

       [3, 2, 1]])

## How to represent missing values ?(Eksik değerler nasıl temsil edilir?)

print(two\_dimension\_array)

two\_dimension\_array[1,1] = 55

two\_dimension\_array[1,2] =44

print(two\_dimension\_array)

[[1 2 3]

 [4 5 6]

 [7 8 9]]

[[ 1  2  3]

 [ 4 55 44]

 [ 7  8  9]]

* **Numpy Zeroes**

numpy\_zeroes = np.zeros((3,3),dtype=int,order='C')

numpy\_zeroes

array([[0, 0, 0],

       [0, 0, 0],

       [0, 0, 0]])

numpy\_ones = np.ones((4,4),dtype=int,order='C')

print(numpy\_ones)

[[1 1 1 1]

 [1 1 1 1]

 [1 1 1 1]

 [1 1 1 1]]

* Reshape:Genel olarak değiştirmek istediğimiz dizin için kullanırız.

numpy.reshape(), numpy.flatten(),aşağıda göreceğimiz üzere diziyi değiştirebiliriz.

first\_shape  = np.array([(1,2,3), (4,5,6)])

print(first\_shape)

reshaped = first\_shape.reshape(3,2)

print(reshaped)

[[1 2 3]

 [4 5 6]]

[[1 2]

 [3 4]

 [5 6]]

* Yukarıdaki dizinin array ile yazalım.

flattened = reshaped.flatten()

flattened

#array([1, 4, 3, 4, 4, 6])

* "array” leri birleştirelim

np\_list\_one = np.array([1,2,3])

np\_list\_two = np.array([4,5,6])

print(np\_list\_one + np\_list\_two)

print('Horizontal Append:', np.hstack((np\_list\_one, np\_list\_two)))

#[5 7 9]

#Horizontal Append: [1 2 3 4 5 6]

* Vertical Stack(Dikey Yığın)

print('Vertical Append:', np.vstack((np\_list\_one, np\_list\_two)))

#Vertical Append: [[1 2 3]

#[4 5 6]]

* Generating Random Numbers(Rastgele Sayılar Üretme)

Örnek:1

random\_float = np.random.random()

random\_float

#0.10062022391087522

Örnek:2

random\_floats = np.random.random(5)#5 ADET FLOAT SAYI ÜRET

random\_floats

#array([0.54871705, 0.61188574, 0.14481353, 0.63867363, 0.68573411])

Örnek:3

  #0 ile 10 arasında rastgele bir tamsayı oluşturma

       random\_int = np.random.randint(0, 11)

       print(random\_int)

      #9 rasgele 0 ile 10 arasında  bir sayı geldi

Örnek:4

#2 ile 10 arasında rasgele 4 adet tam sayı üret

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

random\_int

#array([5, 4, 2, 9])

Örnek:4

#3'e 3 lük matris olsun ve 2ile 10 arası rassal sayılardan oluşsun.

random\_int = np.random.randint(2,10, size=(3,3))

random\_int

#array([[2, 4, 6],

       [6, 9, 2],

       [4, 2, 2]])

* Generationg random numbers(Rasgele Sayı Oluşturma)

Genel gösterim: np.random.normal(mu, sigma, size)(ortalama,varyans,adet)

Örnek:

normal\_array = np.random.normal(79, 15, 80)

normal\_array

Çıktı:

array([ 65.79162175,  52.87867978,  78.67320333,  52.61137418,

       108.93041318,  73.88803797,  98.40779342,  82.45002868,

        82.85260741,  64.99056632,  65.38146986,  48.36771228,

        49.04167286,  47.09471977,  54.94069096,  85.54996565,

        86.82911375,  64.6712154 ,  62.7470582 ,  60.22290748,

        78.70096193,  91.332835  ,  96.87060371,  79.63618362,

        88.38094156,  93.04927569,  90.43887856,  85.44640763,

        64.38383598,  81.21542455,  47.20465619,  80.2904986 ,

        65.16934088,  67.27709637,  76.83120845,  74.79379359,

        72.95407443,  90.49216529,  95.71560489,  57.5762741 ,

        57.55055755,  92.74968067,  56.1174167 ,  83.96339255,

        36.68622187,  84.49278091,  59.89850725,  73.36244983,

        61.27726988,  89.37622428,  89.22865229, 107.11481286,

        71.53717157,  72.96655027, 113.60741613,  72.96181181,

        67.54535122,  82.36553804,  88.61826315,  81.0347691 ,

        85.30413514,  71.34114649,  83.3867504 ,  61.90216392,

        75.64732757,  83.73429924,  67.1720353 ,  90.38365832,

        82.4072446 ,  84.25445911,  91.39229014,  81.98140515,

        77.11038093, 101.34846151,  78.80529281,  65.62182063,

        62.83452668,  98.49867943,  74.47540675,  82.57894099])

## Numpy and Statistics(NumPy ve İstatistik)

Öncelikle gerekli kütüphaneleri çağıralım “seaborn” ve “ matplotlip” çağıralım. Aşağıda örnekte bir set oluşturalım ve histogramını Tablo:1’de inceleyelim.

Örnek:

     import matplotlib.pyplot as plt

     import seaborn as sns

     sns.set()

     plt.hist(normal\_array, color="grey", bins=50)

Çıktı:

 (array([1., 0., 0., 0., 0., 0., 2., 1., 1., 0., 2., 1., 1., 2., 0., 3., 3.,

        0., 7., 2., 1., 0., 2., 4., 3., 1., 2., 4., 3., 5., 5., 4., 1., 2.,

        5., 2., 2., 0., 1., 1., 2., 0., 1., 0., 0., 1., 1., 0., 0., 1.]),

    array([ 36.68622187,  38.22464575,  39.76306964,  41.30149352,

         42.83991741,  44.37834129,  45.91676518,  47.45518906,

         48.99361295,  50.53203683,  52.07046072,  53.6088846 ,

         55.14730849,  56.68573237,  58.22415626,  59.76258015,

         61.30100403,  62.83942792,  64.3778518 ,  65.91627569,

         67.45469957,  68.99312346,  70.53154734,  72.06997123,

         73.60839511,  75.146819  ,  76.68524288,  78.22366677,

         79.76209065,  81.30051454,  82.83893842,  84.37736231,

         85.91578619,  87.45421008,  88.99263397,  90.53105785,

         92.06948174,  93.60790562,  95.14632951,  96.68475339,

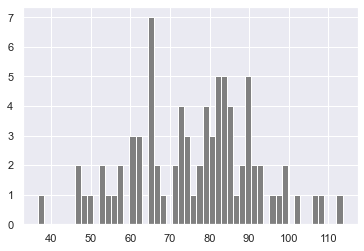
         98.22317728,  99.76160116, 101.30002505, 102.83844893,

        104.37687282, 105.9152967 , 107.45372059, 108.99214447,

        110.53056836, 112.06899224, 113.60741613]),

    <BarContainer object of 50 artists>)

Tablo:1



## Matrix in numpy(NumPy'da Matris)

Örnek:

four\_by\_four\_matrix = np.matrix(np.ones((4,4), dtype=float))

four\_by\_four\_matrix

Çıktı:

matrix([[1., 1., 1., 1.],

        [1., 1., 1., 1.],

        [1., 1., 1., 1.],

        [1., 1., 1., 1.]])

Örnek:

np.asarray(four\_by\_four\_matrix)[2] = 2

four\_by\_four\_matrix

matrix([[1., 1., 1., 1.],

        [1., 1., 1., 1.],

        [2., 2., 2., 2.],

        [1., 1., 1., 1.]])

## Numpy numpy.arange()

Arange Nedir? Bazen, tanımlanmış bir aralık içinde eşit aralıklarla yerleştirilmiş değerler oluşturmak istersiniz. Örneğin, 1'den 10'a kadar değerler oluşturmak istiyorsunuz; numpy.arange() işlevini kullanabiliriz.

lst  =  range ( 0 , 11 , 2 )

lst

range(0, 11, 2)

for l in lst:

    print(l)

0

2

4

6

8

10

Sayı oluşrutalım başlangış noktası artış, miktarı ve bitiş noktasını ele alalım genel gösterimi: range arange numpy.arange(start, stop, step)

whole\_numbers = np.arange(0, 20, 1)

whole\_numbers

#array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,

       17, 18, 19])

## Creating sequence of numbers using linspace(Linspace kullanarak sayı dizisi oluşturma)

Virgül sonrası olan değerler oluşturmak isterseniz “linspace” kullanabiliriz.

np.linspace(1.0, 5.0, num=10)#1.0'dan 5.0'a 10 tane sayı oluşturalım.

array([1.        , 1.44444444, 1.88888889, 2.33333333, 2.77777778,

       3.22222222, 3.66666667, 4.11111111, 4.55555556, 5.        ])

Son değeri dahil etmemek:

Örnek:

np.linspace(1.0, 5.0, num=5, endpoint=False)

#array([1. , 1.8, 2.6, 3.4, 4.2])

LogSpace, bir günlük ölçeğinde çift aralıklı sayıları döndürür. Logspace, np.linspace ile aynı parametrelere sahiptir.

Genel Gösterim: numpy.logspace(start, stop, num, endpoint)

np.logspace(2, 4.0, num=4)

array([  100.        ,   464.15888336,  2154.43469003, 10000.

#Liste Parçalayalım

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

np\_list

#array([[1, 2, 3],

     #  [4, 5, 6]])

# Birinci satır İkinci satır olarak parçalayalım.

print('First row: ', np\_list[0])

print('Second row: ', np\_list[1])

#First row:  [1 2 3]

#Second row:  [4 5 6]

print('First column: ', np\_list[:,0])

print('Second column: ', np\_list[:,1])

print('Third column: ', np\_list[:,2])

#First column:  [1 4]

#Second column:  [2 5]

#Third column:  [3 6]

# NumPy Statistical Functions with Example(Örnekle NumPy İstatistiksel İşlevleri)

NumPy, dizideki verilen öğelerden minimum, maksimum, ortalama, medyan, yüzdelik, standart sapma ve varyans vb. bulmak için oldukça kullanışlı istatistiksel işlevlere sahiptir. Fonksiyonlar aşağıdaki gibi açıklanmıştır. İstatistiksel fonksiyon Numpy, aşağıda listelendiği gibi sağlam istatistiksel fonksiyonla donatılmıştır.

Numpy Fonksiyonları

* Min np.min()
* Maks. np.maks()
* Ortalama np.mean()
* Medyan np.medyan()
* varyans
* Yüzdelik
* Standart sapma np.std()

Örnek:

np\_normal\_dis = np.random.normal(5, 0.5, 100)

print(np\_normal\_dis)

## min, max, mean, median, sd

print('min: ', two\_dimension\_array.min())

print('max: ', two\_dimension\_array.max())

print('mean: ',two\_dimension\_array.mean())

# print('median: ', two\_dimension\_array.median())

print('sd: ', two\_dimension\_array.std())

Çıktı:

[4.51973653 5.18170615 5.23094397 4.63975923 4.59576707 5.02698729

 5.09340393 4.3287569  4.46867527 5.33168575 5.59652313 5.11937362

 5.76146516 5.35608593 4.84339967 3.98936354 6.06551548 5.78428081

 5.32085472 4.89176266 5.46897312 4.65958169 4.79368098 4.77237791

 4.58107749 4.45556191 5.31848723 5.08140703 5.79405462 4.56529556

 4.85768253 5.41523643 4.98410963 5.67512735 5.45741322 4.4874891

 5.0118196  4.85863187 5.04024498 5.33453937 4.81134298 5.10023393

 5.19300245 5.26911795 4.40981663 4.60511726 5.54421921 4.88264242

 5.40697379 5.59951588 3.98545032 5.27245213 4.50138931 4.57302668

 5.23252824 4.76266865 4.85592048 3.81270713 4.84154585 4.1692835

 4.96705909 5.5088007  5.3629441  5.3067828  4.05250016 5.17702812

 5.25085436 4.96604939 5.32144566 4.96029394 5.33009874 4.46877043

 5.06710144 5.67853687 4.8099848  5.03698568 4.98310454 5.4705493

 5.44757937 5.19089288 4.09847691 4.97248168 3.84739249 5.62994409

 5.33869893 5.0621495  5.47524526 4.18595334 5.59694376 5.62553196

 5.59988312 4.1149021  4.40305664 4.93819563 4.76673277 5.30279311

 4.15688708 4.39303436 5.33966837 5.09817038]

min:  1

max:  55

mean:  14.777777777777779

sd:  18.913709183069525

Örnek:

print(two\_dimension\_array)

print('Column with minimum: ', np.amin(two\_dimension\_array,axis=0))

print('Column with maximum: ', np.amax(two\_dimension\_array,axis=0))

print('=== Row ==')

print('Row with minimum: ', np.amin(two\_dimension\_array,axis=1))

print('Row with maximum: ', np.amax(two\_dimension\_array,axis=1))

Çıktı:

[[ 1  2  3]

 [ 4 55 44]

 [ 7  8  9]]

Column with minimum:  [1 2 3]

Column with maximum:  [ 7 55 44]

=== Row ==

Row with minimum:  [1 4 7]

Row with maximum:  [ 3 55  9]

### **How to create repeating sequences?(Tekrar eden diziler nasıl oluşturulur?)**

a = [1,2,3]

print('Tile:   ', np.tile(a, 2)) # 2 kere tekrar etsin

#Tile:    [1 2 3 1 2 3]

print('Repeat: ', np.repeat(a, 2))

#Repeat:  [1 1 2 2 3 3]

### **How to generate random numbers?()**

Örnek:1

one\_random\_num = np.random.random()

one\_random\_in = np.random

print(one\_random\_num)

#0.22885345753592512

Örnek:2

r = np.random.random(size=[2,3])# 2'ye 3 Matris şeklinde

print(r)

#[[0.43326235 0.06563784 0.37499954]

# [0.87698826 0.11264158 0.81280818]]

Örnek:3

from scipy import stats

np\_normal\_dis = np.random.normal(5, 0.1, 1000)

print(np\_normal\_dis)

## min, max, mean, median, sd

print('min: ', np.min(np\_normal\_dis))

print('max: ', np.max(np\_normal\_dis))

print('mean: ', np.mean(np\_normal\_dis))

print('median: ', np.median(np\_normal\_dis))

print('mode: ', stats.mode(np\_normal\_dis))

print('sd: ', np.std(np\_normal\_dis))

Çıktı:

max:  5.329041899264341

mean:  4.998723087998381

median:  5.002055137562557

mode:  ModeResult(mode=array([4.60482159]), count=array([1]))

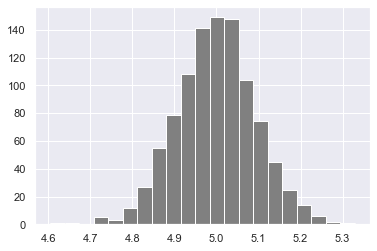
sd:  0.09456348661589764

Histogram Grafiğini Tablo:2’de görebiliriz.

plt.hist(np\_normal\_dis, color="grey", bins=21)

plt.show()

Tablo:2



## Lineer Cebir

Doğrusal iki dizinin çarpımı

f = np.array([1,2,3])

g = np.array([4,5,3])

### 1\*4+2\*5 + 3\*6

np.dot(f, g)  # 23

### **NumPy Matrix Multiplication with np.matmul()**

h = [[1,2],[3,4]]

i = [[5,6],[7,8]]

### 1\*5+2\*7 = 19

np.matmul(h, i)

### array([[19, 22],

       #[43, 50]])

Determinant :

np.linalg.det(i)

#-2.000000000000005

Örnek:

# x+2 olsun ve  0'dan başlasın ve 11'e kadar 1'er artsın

new\_list = [ x + 2 for x in range(0, 11)]

new\_list

#[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

Örnek:

# x+2 yerine bir başka şekilde aynı işlemi yapabiliriz.

np\_arr = np.array(range(0, 11))

np\_arr + 2

#array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])

Doğrusal ilişkisi olan nicelikler için doğrusal denklem kullanırız. Aşağıdaki örneği görelim:

* Numpy kullanarak Gauss normal dağılımını çizmek. Aşağıda görebileceğiniz gibi, numpy rastgele sayılar üretebilir. Rastgele örnek oluşturmak için ortalama(mu), sigma(standart sapma), veri noktalarının sayısına ihtiyacımız var.

Örnek:

mu = 28

sigma = 15

samples = 100000

x = np.random.normal(mu, sigma, samples)

ax = sns.distplot(x);

ax.set(xlabel="x", ylabel='y')

plt.show()

