NumPy

Makine öğrenmesinde büyük verileri yönetecegimiz için pythondaki list veri tipi bunun için yavaş ve maliyetli olacaktır.(Hafıza açısından). Daha hızlı ve az hafıza kullanarak veri setlerini yönetmek için "NumPy" kütüphanesini kullanacağız.

İndirmek için CMD'ye girip, "pip install numpy" yazmak yeterlidir.

info → NumPy = Numeric Python
info → ndarray = n-dimensional array (This is the data type of NumPy)

1) Introduction To Numpy

```
import numpy as np
```

* `np1 = np.array([0,1,2,3,4])`

```
print(np1.shape)  # This function gives us the shape of array.
>>> (5,)
```

* `np2 = np.arange(10)` `#.arange(x,y,z)` gives us a array from x to y. (z kadar atlayarak)

```
print(np2)
>>> [0,1,2,3,4,5,6,7,8,9]
```

* `np3 = np.array(0,10,2)`

```
print(np3)
>>> [0,2,4,6,8]
```

* #Zeros

```
np4 = np.zeros(5)
print(np4)
>>> [0,0,0,0,0]
```

* #Multidimensional Zeros

```
np5 = np.zeros((2,5))
```

→ How many rows is
→ How many columns

```
>>> [[0,0,0,0,0]
     [0,0,0,0,0]]
```

* #Full

```
np6 = np.full((10),6)
print(np6)
>>> [6,6,6,6,6,6,6,6,6,6]
```

CamScanner ile tarandı

* # Multidimensional Full

```
np7 = np.full((2,10),6)
print(np7)
>>> [[6,6,6,6,6,6,6,6,6,6]
     [6,6,6,6,6,6,6,6,6,6]]
```

* # Convert python lists to np

```
my_list = [1,2,3,4,5]
np8 = np.array(my_list)
print(np8)
print(np8[0])

>>> [1,2,3,4,5]
    1
```

---

2) Slicing Numpy Arrays

```
np1 = np.array([1,2,3,4,5,6,7,8,9])

# Return [2,3,4,5]
print(np1[1:5])

# Return from something till the end of array
# [4,5,6,7,8,9]
print(np1[3:])

# Return negative slices
# [7,8]
print(np1[-3:-1])

# Steps
print(np1[1:5:2])
>>> [2,4]

# Steps on entire array
print(np1[::2])
>>> [1,3,5,7,9]

# Slice a 2-D array

np2 = np.array([[1,2,3,4,5],[6,7,8,9,10]])

# Pull out a single item
print(np2[1,2])
>>> 8
# Slice a 2-d array → >>> [2,3]
print(np2[0:1, 1:3])

print(np2[0:2, 1:3])
>>> [[2,3]
     [7,8]]
```

info →

	0	1	2
	x	y	z
	-3	-2	-1

CamScanner ile tarandı

3) Numpy Universal Functions

```
np1 = np.array([0,1,2,3,4,5,6,7,8,9])

# Square root (karekök) of each element
print(np.sqrt(np1))

# Absolute Value → (Gelişması için -3,-2,-1 sayılarını ekledik varsayalım.)
print(np.absolute(np1))
>>> [3,2,1,0,1,2,3,4,5,6,7,8,9]

# Exponentials → e üzeri olarak ekrana yazdırır. (e≈2,718)
print(np.exp(np1))

# Min /Max
print(np.min(np1))   >>> -3
print(np.max(np1))   >>> 9

# Sign positive or negative
print(np.sign(np1))
>>> [-1,-1,-1,0,1,1,1,1,1,1,1,1,1]

# Trigonometric sin cos log
print(np.sin(np1))  → Arraydaki değerleri radyan olarak alarak sin ve cos
print(np.cos(np1))    değerlerini döndürür.

print(np.log(np1))  → log_e^np1 = olacak şekilde her karakterin logaritmasını alır.
```

If you want to learn more Google → Numpy.org universal functions (ufunc)

4) Copy Vs. View

```
np1 = np.array([0,1,2,3,4,5])

np2 = np1.view()   → viewde np1 ve np2 bağlantılı olur ve birinde
                     yapılan değişiklik ikisini de etkiler
np3 = np1.copy()   → Copy'de np3, np1'in kopyası fakat bağımsız
                     bir array olur. Değişiklikler sadece ilgili array'i değiştirir.
```

CamScanner ile tarandı

5) Shape and Reshape Arrays

```
# Create 1-D array

np1 = np.array([1,2,3,4,5,6,7,8,9,10,11,12])

# Create 2-D array

np2 = np.array([[1,2,3,4,5,6],[7,8,9,10,11,12]])
print(np2.shape)
>>> (2,6)

# Reshape 2-D

np3 = np1.reshape(3,4)
print(np3)
>>> [[ 1,2,3,4]
     [ 5,6,7,8]
     [ 9,10,11,12]]

# Reshape 3-D

np4 = np1.reshape(2,3,2)
print(np4)
>>> [[[ 1,2]
      [3,4]
      [5,6]]

     [[ 7,8]
      [9,10]
      [11,12]]]

# np5 = np4.reshape(-1)   → (2,3,2) lik np4 ü (12,) lik tek boyutlu hale çevirir
```

6) Iterating Through Numpy Arrays

```
# np1 = np.array([1,2,3,4,5,6,7,8,9,10])
for x in np1:
    print(x)

>>> teker teker elemanları yazar, alt alta olacak şekilde.


# np2 = np.array([[1,2,3,4,5],[6,7,8,9,10]])
for x in np2:
    #print(x) ⟶ [1,2,3,4,5. ve .6,7,8,9,10 u yazar alt alta.
    for y in x:
        print(y) → 1,2,3,4,5,6,7,8,9,10 sayılarını alt alta yazar.
```

CamScanner ile tarandı

```
# np3 = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
  for x in np3:
      for y in x:
          for z in y:
              print(z)   → alt alta 1'den 12'ye kadar yazar.


# np.nditer()

for x in np.nditer(np3):
    print(x)

>>> alt alta 1'den 12'ye kadar yazar.
```

---

7) Sorting Numpy Arrays The Right Way

```
np1 = np.array([6,7,9,0,2,1])

# np.sort()

print(np.sort(np1))   → np1 arrayini numerik olarak sıralar.

np2 = np.array(["John","Tino","Aaron","Zed"])
print(np.sort(np2))   → Alphabetic olarak sıralar.

np3 = np.array([True, False, False, True])
print(np.sort(np3))   → Boolean olarak sıralar. Önce Falseler sonra Trueler.

# Return a copy not change the original
print(np1)
print(np.sort(np1))
print(np1)

>>> [6, 7, 9, 0, 2, 1]
    [0, 1, 2, 6, 7, 9]
    [6, 7, 9, 0, 2, 1]
```

→ np1'in yapısı değişmedi çünkü 2. printte sort yaparken copy'ini edip gösterdi. Bu yüzden orijinalde değişim olmadı.

```
# 2-d sorting

np4 = np.array([[6,7,1,9],[8,3,5,0]])
print(np.sort(np4))   → Sıralama öğeler arasında bireysel olarak yapılır.

>>> [[1,6,7,9],[0,3,5,8]]
```

CamScanner ile tarandı

8) Searching Numpy Arrays The Easy Way

Search

```
npl = np.array([1,2,3,4,5,6,7,8,9,10])

x= np.where(npl==3)
print(x)
```

integer bu sit

```
>>> (array([2], dtype=int64),) → this is a tuple
```

index of "3" — data type of "3"

```
print(x[0])
>>> 2
```

---

there are two "3"

```
npl = np.array([1,2,3,4,5,6,7,8,9,10,3])

x= np.where(npl==3)
print(x[0])
>>> [2,10] → yani "3" ün bulundugu indexleri verir. x[0] dedik çünkü tuplenin 0. indexinde sokli yerler.

print(npl[x[0]])
>>> [3,3]
```

Return even items

```
y = np.where(npl%2==0)
print(y) → tek sayilari yazar.
print(y[0]) → Tek sayilerin indexlerini yazar.
```

Return odd items

```
z= np.where(npl%2==1)
```

CamScanner ile tarandı

9) How to Filter Numpy Arrays

Filtering Numpy Arrays with Boolean Index Lists

```
npl = np.array([1,2,3,4,5,6,7,8,9,10])
x = [True, True, False, F, F, F, F, F, F, F]    F = False

print(npl[x])

>>> [1,2]


# filtered = [ ]
  for thing in npl:
      if thing >5 :
        filtered.oppend(True)
      else :
        filtered.oppend(False)

print(filtered)
print(npl[filtered])

>>>   [F,F,F,F,F,T,T,T,T,T]
      [6,7,8,9,10]


# Shortcut

filtered = npl >5
print(filtered)
print(npl[filtered])

>>> Yukardaki çıktının aynısını verir.
```

CamScanner ile tarandı