NumPy For Machine Learning

Video 1 :
NumPy array takes only one type of data, that's mean that every element of the array should be the same data type.

NumPy array is an n-dimensional array that's mean that it can be 2-dimensional array or 3-dimensional array and so on.

To make a NumPy array:
Import NumPy as np.
Np1 = np.array([0,1,2,3,4,5,6,7,8,9])
Or
Np2 = np.arange(10)

** arange(start,end,step)

So we can write that np3 = arange(2,10,2) that will make an array from 2 to 10 with step = 2.

Print(np1.shape): the shape function tells us the length of the array

Np4 = np.zeros(10): that will create a numpy array of 10 zeros
** Zeros((number of dimensions , size of each))

Np5 = np.full((10),6): that will create a numpy array of size 10 consisting of 6.

Np6 = np.full((2,10),6): that will create a 2-dimensional NumPy array of size 10 consisting of 6.
** full((n-dimensions,size),fill-number)

We also can convert a normal python list to a NumPy array.
Ex:
Arr = [1,2,3,4,5]
Np7 = np.array(Arr)

NumPy arrays are zero base indexed so we can access any element by its own index.

Video 2 : Slicing:
Slicing is the same as in a normal python list.
[start , end , step] “end not included”
Ex:
Import numpy an np

```
Np1 = np.array([1,2,3,4,5,6,7,8,9])
Print(np1[1:5])          # that will return 2,3,4,5

Print(np1[3:])           # that will return 4,5,6,8,8,9

Print(np1[-3,-1])        # that will return 7,8

Print(np1[ : :2])        #that will return the entire array with step of 2



Np2 = np.array([[1,2,3,4,5],[6,7,8,9]])
Print([1,2])        # that will return 8
Print([0:1 , 1:3])# that will return 2,3
Print([0:2 , 1:3])# that will slice from both rows so that will return
                        2,3 and 7,8.
```

Video 3 : Universal functions:

Import numpy as np

Np1 = np.array([-3,-2,-1,0,1,2,3,4,5,6,7,8,9])

# to take the square root of each element:
Print(np.sqrt(Np1))

# to take the absolute value of each element of the array:
Print(np.absolute(Np1))

# Exponentials: $e^{each\ element\ of\ the\ array}$:
Print(np.exp(Np1))

# min / max:
Print(np.min(Np1))
Print(np.max(Np1))

sign:
Print(np.sign(Np1)) # that will return (-1) for each negative number in the array and (1) for each positive and (0) for zero.

#log , sin , cos, tan ,…
Print(np.log(Np1)) # that will return (nan) for negative numbers and (inf) for zero.

Video 4 : View and copy.

Copy: is a shallow copy of the original array that's mean its totally different of the original array so, if we changed any element of the original array the copy will not affect and vice versa.

View: is a copy of the original array but its totally related to it so, if we changed any element of the original array the view will be affected and vice versa.

Ex: view.

```
import numpy as np
arr1 = np.array([1,2,3,4])
arr2 = arr1.view()
print(arr1) # 1 2 3 4
print(arr2) # 1 2 3 4
arr1[0]=5
arr2[1]=8
print(arr1) # 5 8 3 4
print(arr2) # 5 8 3 4
```

Ex: copy.

```
arr1 = np.array([1,2,3,4])
arr2 = arr1.copy()
print(arr1) # 1 2 3 4
print(arr2) # 1 2 3 4
arr1[0]=5
arr2[1]=8
print(arr1) # 5 2 3 4
print(arr2) # 1 8 3 4
```

Video 5 : Shape and Reshape Numpy Arrays.
Import numpu as np
#1-d array:
arr1= np.array([1,2,3,4,5,6])

#2-d array:
Arr2=np.array([[1,2,3],[4,5,6]])

Print(arr2.shape()) # that will return (2,3) where 2 = numbers of rows and 3 = numbers of columns.

 # reshape an array: reshape(rows,columns).

Arr3 = arr1.reshape(2,3) # that will reshape the 1-d arr1 to a 2-d arr3 array >> [[1,2,3]
[4,5,6]]

#reshape to 3d:
Arr4 = arr1.reahape(2,3,1) ## [[[1]
[2]
[3]
[4]
[5]
[6]]]

 # flatten to 1d array:
Arr5 = arr4.reshape(-1) # [1,2,3,4,5,6]

Video 6 : Iterating Through Numpy Arrays.

```
Import numpy as np

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

# Now to print each single element of this 3-d array we can use 3 nested for loops:

For x in arr:
    For y in x:
        For z in y:
            Print(z)


# or we can just use nditer()  method:

For I in np.nditer(arr):
    Print(I)

# that will give us the same result.
```

Video 7 : Sorting Numpy Arrays.

```
Import numpy as np

Arr = np.array([6,7,4,9,0,2,1])

Print(np.sort(arr)) #0 1 2 4 6 7 9

Arr2 = np.array([d,c,b,a])

Print(np.sort(arr2)) # a b c d

Arr3 = np.array([True,False,False,True])

Print(np.sort(arr3)) # False False True True

#2-d array:

Arr4 = np.array([[3,2,1],[6,5,4]])

Print(np.sort(arr4)) # that will sort each individual row
                              [1 2 3]
                              [4 5 6]
```

***** sort() function does not change the original array its just returns a copy of array but sorted.**

Video 8 : Searching Numpy Arrays.

Import numpy as np

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

if I want to know the index of 3 for example so, I should use **where()** function.

X= np.where(arr)

Print(x) # that will return (array([2] , dtype = int64))

if I just need the index of 3:

Print(x[0]) #that will return [2] which is the index of 3

#What if I have more than a 3

Ex: Arr3 = np.array([1,2,3,4,5,6,7,8,9,10,3])

X= np.where(arr)

Print(x[0]) #that will return [2,10]

if I want to get index of even numbers:

Y = np.where(arr%2==0)
Print(y[0]) # 1 3 5 7 9

# if I want actual numbers:
Print(arr[y[0]])

Video 9 : Filter Numpy Arrays.

```
Import numpy as np

Arr = np.array([1,2,3,4,5,6,7,8,9,10])
# if I want to print all even numbers:
Filteres=[]
For x in arr:
        If x%2==0:
                Filteres.append(True)
        Else:
                Filteres.append(False)

Print(arr)          # 1 2 3 4 5 6 7 8 9 10

Print(filteres)  # False True False True False True False True False
True

Print(arr[filteres]) #[2,4,6,8,10]

# or we can write it like that:

Filters2 = arr%2 ==0

# we will get the same results

Print(arr)          # 1 2 3 4 5 6 7 8 9 10

Print(filteres2)  # False True False True False True False True
False True

Print(arr[filteres2]) #[2,4,6,8,10]
```

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

Print(arr.ndim) **>> ndim** returns number of dimensions of your array which here =1.
Arr.**dtype** >> returns data type ‘int64’

#we can specify the data type of arrays’ element when we create it like: a=np.array([1,2,3],**dtype=’int16’**)

---

to get how many bytes of the array: **arr.nbytes**

---

to create a numpy array full of ones: **s=np.ones()**

---

to create an array full of random numbers between 0 : 1:
X=np.random.rand(n-rows,n-columns)

# to create an array full of random integer numbers between start , end :
X= np.random.randint(start,end ,size=(rows,columns))
Ex: x = np.random.randint(2,8 , size=(2,3))
** end not included

#to create a matrix of main diagonal=1 and the rest of the matrix =0

Np.identity(number)

Ex: x= np.identity(3) >> that will return a 3*3 matrix

Loading data from a file:

File_data = np.**genfromtxt('file_name.txt' , delimiter = ',')**

# the data type of file_data is float if I want to change it to int:
File_data = file_data.**astype**('int32')

Print(file_data[file_data>50]) >> that will return every number in file_data that is greater than 50.

Len(arr) >> returns the number of rows.

Tolist() >> convert an numpy array to list

Import numpy as np
A = np.array([1,2,3],float)
S = **a.tosting() >>** converts the array elements to unreadable output 'encryption'

Print(np.**fromstring**(S)) >> 'decryption'

Np.transpose() >> بتقلب الصفوف لاعمده و العكس

Np.concatenate() >> merge arrays.
Ex:
A = np.array([1,2])

B = np.array([3,4])

Np.cocatenate((A,B)) >>> [1,2,3,4]

np.newaxis

Ex:

```
>>> a = np.array([1, 2, 3], float)
>>> a
array([1., 2., 3.])
>>> a[:,np.newaxis]
array([[ 1.],
[ 2.],
[ 3.]])
>>> a[:,np.newaxis].shape
(3,1)
>>> b[np.newaxis,:]
array([[ 1., 2., 3.]])
>>> b[np.newaxis,:].shape
(1,3)
```
