

## Numpy Tutorial Part 1 – Introduction to Arrays

*This is part 1 of the numpy tutorial covering all the core aspects of performing data manipulation and analysis with numpy's ndarrays. Numpy is the most basic and a powerful package for scientific computing and data manipulation in python.*



Numpy Tutorial Part 1: Introduction to Arrays. Photo by Bryce Canyon.

2. How to create a numpy array?
3. How to inspect the size and shape of a numpy array?
4. How to extract specific items from an array?
  - 4.1 How to reverse the rows and the whole array?
  - 4.2 How to represent missing values and infinite?
  - 4.3 How to compute mean, min, max on the ndarray?
5. How to create a new array from an existing array?
6. Reshaping and Flattening Multidimensional arrays
  - 6.1 What is the difference between flatten() and ravel()?
7. How to create sequences, repetitions, and random numbers?
  - 7.1 How to create repeating sequences?
  - 7.2 How to generate random numbers?
  - 7.3 How to get the unique items and the counts?

## 1. Introduction to Numpy

Numpy is the most basic and a powerful package for working with data in python.

If you are going to work on data analysis or machine learning projects, then having a solid understanding of numpy is nearly mandatory.

Because other packages for data analysis (like pandas) is built on top of numpy and the scikit-learn package which is used to build machine learning applications works heavily with numpy as well.

So what does numpy provide?

At the core, numpy provides the excellent ndarray objects, short for n-dimensional arrays.

In a 'ndarray' object, aka 'array', you can store multiple items of the same data type. It is the facilities around the array object that makes numpy so convenient for performing math and data manipulations.

You might wonder, *'I can store numbers and other objects in a python list itself and do all sorts of computations and manipulations through list comprehensions, for-loops etc. What do I need a numpy array for?'*

Well, there are very significant advantages of using numpy arrays over lists.

To understand this, let's first see how to create a numpy array.

## 2. How to create a numpy array?

There are multiple ways to create a numpy array, most of which will be covered as you read this. However one of the most common ways is to create one from a list or a list like an object by passing it to the `np.array` function.

```
# Create an 1d array from a list
import numpy as np
list1 = [0,1,2,3,4]
arr1d = np.array(list1)

# Print the array and its type
print(type(arr1d))
arr1d

#> class 'numpy.ndarray'
#> array([0, 1, 2, 3, 4])
```

The key difference between an array and a list is, arrays are designed to handle vectorized operations while a python list is not.

That means, if you apply a function it is performed on every item in the array, rather than on the whole array object.

Let's suppose you want to add the number 2 to every item in the list. The intuitive way to do it is something like this:

```
list1 + 2  # error
```

That was not possible with a list. But you can do that on a ndarray.

```
# Add 2 to each element of arr1d
arr1d + 2

#> array([2, 3, 4, 5, 6])
```

Another characteristic is that, once a numpy array is created, you cannot increase its size. To do so, you will have to create a new array. But such a behavior of extending the size is natural in a list.

Nevertheless, there are so many more advantages. Let's find out.

So, that's about 1d array. You can also pass a list of lists to create a matrix like a 2d array.

```
# Create a 2d array from a list of lists
list2 = [[0,1,2], [3,4,5], [6,7,8]]
arr2d = np.array(list2)
arr2d

#> array([[0, 1, 2],
#>         [3, 4, 5],
#>         [6, 7, 8]])
```

You may also specify the datatype by setting the dtype argument. Some of the most commonly used numpy dtypes are: 'float', 'int', 'bool', 'str' and 'object'.

To control the memory allocations you may choose to use one of 'float32', 'float64', 'int8', 'int16' or 'int32'.

```
# Create a float 2d array
arr2d_f = np.array(list2, dtype='float')
arr2d_f

#> array([[ 0.,  1.,  2.],
#>         [ 3.,  4.,  5.],
#>         [ 6.,  7.,  8.]])
```

The decimal point after each number is indicative of the float datatype. You can also convert it to a different datatype using the `astype` method.

```
# Convert to 'int' datatype
arr2d_f.astype('int')

#> array([[0, 1, 2],
#>         [3, 4, 5],
#>         [6, 7, 8]])
```

```
# Convert to int then to str datatype
arr2d_f.astype('int').astype('str')

#> array([[ '0', '1', '2'],
#>         [ '3', '4', '5'],
#>         [ '6', '7', '8']],
#>        dtype='U21')
```

A numpy array must have all items to be of the same data type, unlike lists. This is another significant difference.

However, if you are uncertain about what datatype your array will hold or if you want to hold characters and numbers in the same array, you can set the `dtype` as `'object'`.

```
# Create a boolean array
arr2d_b = np.array([1, 0, 10], dtype='bool')
arr2d_b

#> array([ True, False,  True], dtype=bool)
```

```
# Create an object array to hold numbers as well as strings
arr1d_obj = np.array([1, 'a'], dtype='object')
arr1d_obj

#> array([1, 'a'], dtype=object)
```

Finally, you can always convert an array back to a python list using `tolist()`.

```
# Convert an array back to a list
arr1d_obj.tolist()

#> [1, 'a']
```

To summarise, the main differences with python lists are:

1. Arrays support vectorised operations, while lists don't.
2. Once an array is created, you cannot change its size. You will have to create a new array or overwrite the existing one.
3. Every array has one and only one dtype. All items in it should be of that dtype.
4. An equivalent numpy array occupies much less space than a python list of lists.

### 3. How to inspect the size and shape of a numpy array?

Every array has some properties I want to understand in order to know about the array.

Let's consider the array, arr2d. Since it was created from a list of lists, it has 2 dimensions that can be shown as rows and columns, like in a matrix.

Had I created one from a list of list of lists, it would have 3 dimensions, as in a cube. And so on.

Let's suppose you were handed a numpy vector that you didn't create yourself. What are the things you would want to explore in order to know about that array?

Well, I want to know:

- <li>If it is a 1D or a 2D array or more. (ndim)</li>
- <li>How many items are present in each dimension (shape)</li>
- <li>What is its datatype (dtype)</li>
- <li>What is the total number of items in it (size)</li>
- <li>Samples of first few items in the array (through indexing)</li>

```
# Create a 2d array with 3 rows and 4 columns
list2 = [[1, 2, 3, 4],[3, 4, 5, 6], [5, 6, 7, 8]]
arr2 = np.array(list2, dtype='float')
arr2

#> array([[ 1.,  2.,  3.,  4.],
#>         [ 3.,  4.,  5.,  6.],
#>         [ 5.,  6.,  7.,  8.]])
```

```
# shape
print('Shape: ', arr2.shape)

# dtype
print('Datatype: ', arr2.dtype)

# size
print('Size: ', arr2.size)

# ndim
print('Num Dimensions: ', arr2.ndim)

#> Shape: (3, 4)
#> Datatype: float64
#> Size: 12
#> Num Dimensions: 2
```

## 4. How to extract specific items from an array?

```
arr2

#> array([[ 1.,  2.,  3.,  4.],
#>         [ 3.,  4.,  5.,  6.],
#>         [ 5.,  6.,  7.,  8.]])
```

You can extract specific portions on an array using indexing starting with 0, something similar to how you would do with python lists.

But unlike lists, numpy arrays can optionally accept as many parameters in the square brackets as there is number of dimensions.

```
# Extract the first 2 rows and columns
arr2[:2, :2]
list2[:2, :2] # error

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

Additionally, numpy arrays support boolean indexing.

A boolean index array is of the same shape as the array to be filtered and it contains only True and False values. The values corresponding to True positions are retained in the output.

```
# Get the boolean output by applying the condition to each element.
b = arr2 > 4
b

#> array([[False, False, False, False],
#>        [False, False,  True,  True],
#>        [ True,  True,  True,  True]], dtype=bool)
```

```
arr2[b]

#> array([ 5.,  6.,  5.,  6.,  7.,  8.])
```

## 4.1 How to reverse the rows and the whole array?

Reversing an array works like how you would do with lists, but you need to do for all the axes [dimensions] if you want a complete reversal.

```
# Reverse only the row positions
arr2[::-1, ]

#> array([[ 5.,  6.,  7.,  8.],
#>        [ 3.,  4.,  5.,  6.],
#>        [ 1.,  2.,  3.,  4.]])
```

```
# Reverse the row and column positions
arr2[::-1, ::-1]

#> array([[ 8.,  7.,  6.,  5.],
#>        [ 6.,  5.,  4.,  3.],
#>        [ 4.,  3.,  2.,  1.]])
```





## 4.2 How to represent missing values and infinite?

Missing values can be represented using `np.nan` object, while `np.inf` represents infinite. Let's place some in `arr2d`.

```
# Insert a nan and an inf
arr2[1,1] = np.nan # not a number
arr2[1,2] = np.inf # infinite
arr2

#> array([[ 1.,  2.,  3.,  4.],
#>         [ 3., nan, inf,  6.],
#>         [ 5.,  6.,  7.,  8.]])
```

```
# Replace nan and inf with -1. Don't use arr2 == np.nan
missing_bool = np.isnan(arr2) | np.isinf(arr2)
arr2[missing_bool] = -1
arr2

#> array([[ 1.,  2.,  3.,  4.],
#>         [ 3., -1., -1.,  6.],
#>         [ 5.,  6.,  7.,  8.]])
```

## 4.3 How to compute mean, min, max on the ndarray?

The ndarray has the respective methods to compute this for the whole array.

```
# mean, max and min
print("Mean value is: ", arr2.mean())
print("Max value is: ", arr2.max())
print("Min value is: ", arr2.min())

#> Mean value is:  3.583333333333
#> Max value is:  8.0
#> Min value is:  -1.0
```

However, if you want to compute the minimum values row wise or column wise, use the `np.amin` version instead.

```
# Row wise and column wise min
print("Column wise minimum: ", np.amin(arr2, axis=0))
print("Row wise minimum: ", np.amin(arr2, axis=1))

#> Column wise minimum: [ 1. -1. -1.  4.]
#> Row wise minimum:   [ 1. -1.  5.]
```

Computing the minimum row-wise is fine. But what if you want to do some other computation/function row-wise? It can be done using the `np.apply_over_axis` which you will see in the upcoming topic.

```
# Cumulative Sum
np.cumsum(arr2)

#> array([ 1.,  3.,  6., 10., 13., 12., 11., 17., 22., 28., 35., 43.]
```

## 5. How to create a new array from an existing array?

If you just assign a portion of an array to another array, the new array you just created actually refers to the parent array in memory.

That means, if you make any changes to the new array, it will reflect in the parent array as well.

So to avoid disturbing the parent array, you need to make a copy of it using `copy()`. All numpy arrays come with the `copy()` method.

```
# Assign portion of arr2 to arr2a. Doesn't really create a new array.
arr2a = arr2[:2,:2]
arr2a[:1, :1] = 100 # 100 will reflect in arr2
arr2

#> array([[ 100.,   2.,   3.,   4.],
#>        [   3.,  -1.,  -1.,   6.],
#>        [   5.,   6.,   7.,   8.]])
```

```
# Copy portion of arr2 to arr2b
arr2b = arr2[:2, :2].copy()
arr2b[:1, :1] = 101 # 101 will not reflect in arr2
arr2

#> array([[ 100.,    2.,    3.,    4.],
#>         [    3.,   -1.,   -1.,    6.],
#>         [    5.,    6.,    7.,    8.]])
```

## 6. Reshaping and Flattening Multidimensional arrays

Reshaping is changing the arrangement of items so that shape of the array changes while maintaining the same number of dimensions.

Flattening, however, will convert a multi-dimensional array to a flat 1d array. And not any other shape.

First, let's reshape the `arr2` array from 3x4 to 4x3 shape.

```
# Reshape a 3x4 array to 4x3 array
arr2.reshape(4, 3)

#> array([[ 100.,    2.,    3.],
#>         [    4.,    3.,   -1.],
#>         [   -1.,    6.,    5.],
#>         [    6.,    7.,    8.]])
```

### 6.1 What is the difference between `flatten()` and `ravel()`?

There are 2 popular ways to implement flattening. That is using the `flatten()` method and the other using the `ravel()` method.

The difference between `ravel` and `flatten` is, the new array created using `ravel` is actually a reference to the parent array. So, any changes to the new array will affect the parent as well. But is memory efficient since it does not create a copy.

```
# Flatten it to a 1d array
arr2.flatten()

#> array([ 100.,    2.,    3.,    4.,    3.,   -1.,   -1.,    6.,    5.,  6.,    7.,
  8.])
```

```
# Changing the flattened array does not change parent
b1 = arr2.flatten()
b1[0] = 100 # changing b1 does not affect arr2
arr2

#> array([[ 100.,    2.,    3.,    4.],
#>         [    3.,   -1.,   -1.,    6.],
#>         [    5.,    6.,    7.,    8.]])
```

```
# Changing the raveled array changes the parent also.
b2 = arr2.ravel()
b2[0] = 101 # changing b2 changes arr2 also
arr2

#> array([[ 101.,    2.,    3.,    4.],
#>         [    3.,   -1.,   -1.,    6.],
#>         [    5.,    6.,    7.,    8.]])
```

## 7. How to create sequences, repetitions and random numbers using numpy?

The `np.arange` function comes handy to create customised number sequences as `ndarray`.

```
# Lower limit is 0 be default
print(np.arange(5))

# 0 to 9
print(np.arange(0, 10))

# 0 to 9 with step of 2
print(np.arange(0, 10, 2))

# 10 to 1, decreasing order
print(np.arange(10, 0, -1))

#> [0 1 2 3 4]
#> [0 1 2 3 4 5 6 7 8 9]
#> [0 2 4 6 8]
#> [10 9 8 7 6 5 4 3 2 1]
```

You can set the starting and end positions using `np.arange`. But if you are focussed on the number of items in the array you will have to manually calculate the appropriate step value.

Say, you want to create an array of exactly 10 numbers between 1 and 50, Can you compute what would be the step value?

Well, I am going to use the `np.linspace` instead.

```
# Start at 1 and end at 50
np.linspace(start=1, stop=50, num=10, dtype=int)

#> array([ 1,  6, 11, 17, 22, 28, 33, 39, 44, 50])
```

Notice since I explicitly forced the `dtype` to be `int`, the numbers are not equally spaced because of the rounding.

Similar to `np.linspace`, there is also `np.logspace` which rises in a logarithmic scale. In `np.logspace`, the given start value is actually  $\text{base}^{\text{start}}$  and ends with  $\text{base}^{\text{stop}}$ , with a default based value of 10.

```
# Limit the number of digits after the decimal to 2
np.set_printoptions(precision=2)

# Start at 10^1 and end at 10^50
np.logspace(start=1, stop=50, num=10, base=10)

#> array([ 1.00e+01,  2.78e+06,  7.74e+11,  2.15e+17,  5.99e+22,
#>         1.67e+28,  4.64e+33,  1.29e+39,  3.59e+44,  1.00e+50])
```

The `np.zeros` and `np.ones` functions lets you create arrays of desired shape where all the items are either 0's or 1's.

```
np.zeros([2,2])
#> array([[ 0.,  0.],
#>         [ 0.,  0.]])
```

```
np.ones([2,2])
#> array([[ 1.,  1.],
#>         [ 1.,  1.]])
```

## 7.1 How to create repeating sequences?

`np.tile` will repeat a whole list or array `n` times. Whereas, `np.repeat` repeats each item `n` times.

```
a = [1,2,3]

# Repeat whole of 'a' two times
print('Tile: ', np.tile(a, 2))

# Repeat each element of 'a' two times
print('Repeat: ', np.repeat(a, 2))

#> Tile:      [1 2 3 1 2 3]
#> Repeat:    [1 1 2 2 3 3]
```

## 7.2 How to generate random numbers?

The `random` module provides nice functions to generate random numbers (and also statistical distributions) of any given shape.

```

# Random numbers between [0,1) of shape 2,2
print(np.random.rand(2,2))

# Normal distribution with mean=0 and variance=1 of shape 2,2
print(np.random.randn(2,2))

# Random integers between [0, 10) of shape 2,2
print(np.random.randint(0, 10, size=[2,2]))

# One random number between [0,1)
print(np.random.random())

# Random numbers between [0,1) of shape 2,2
print(np.random.random(size=[2,2]))

# Pick 10 items from a given list, with equal probability
print(np.random.choice(['a', 'e', 'i', 'o', 'u'], size=10))

# Pick 10 items from a given list with a predefined probability 'p'
print(np.random.choice(['a', 'e', 'i', 'o', 'u'], size=10, p=[0.3, .1, 0.1, 0.4, 0.1])) # picks more o's

#> [[ 0.84  0.7 ]
#> [ 0.52  0.8 ]]

#> [[-0.06 -1.55]
#> [ 0.47 -0.04]]

#> [[4 0]
#> [8 7]]

#> 0.08737272424956832

#> [[ 0.45  0.78]
#> [ 0.03  0.74]]

#> ['i' 'a' 'e' 'e' 'a' 'u' 'o' 'e' 'i' 'u']
#> ['o' 'a' 'e' 'a' 'a' 'o' 'o' 'o' 'a' 'o']

```

Now, everytime you run any of the above functions, you get a different set of random numbers.

If you want to repeat the same set of random numbers every time, you need to set the seed or the random state. The seed can be any value. The only requirement is you must set the seed to the same value every time you want to generate the same set of random numbers.



Once `np.random.RandomState` is created, all the functions of the `np.random` module becomes available to the created randomstate object.

```
# Create the random state
rn = np.random.RandomState(100)

# Create random numbers between [0,1) of shape 2,2
print(rn.rand(2,2))

#> [[ 0.54  0.28]
#>   [ 0.42  0.84]]
```

```
# Set the random seed
np.random.seed(100)

# Create random numbers between [0,1) of shape 2,2
print(np.random.rand(2,2))

#> [[ 0.54  0.28]
#>   [ 0.42  0.84]]
```

## 7.3 How to get the unique items and the counts?

The `np.unique` method can be used to get the unique items. If you want the repetition counts of each item, set the `return_counts` parameter to `True`.

```
# Create random integers of size 10 between [0,10)
np.random.seed(100)
arr_rand = np.random.randint(0, 10, size=10)
print(arr_rand)

#> [8 8 3 7 7 0 4 2 5 2]
```

```
# Get the unique items and their counts
unqs, counts = np.unique(arr_rand, return_counts=True)
print("Unique items : ", unqs)
print("Counts       : ", counts)

#> Unique items : [0 2 3 4 5 7 8]
#> Counts       : [1 2 1 1 1 2 2]
```





## 8.0 Conclusion

This completes the part 1 of the numpy series. The next one is [advanced numpy for data analysis](https://www.machinelearningplus.com/numpy-tutorial-python-part2/) [\[https://www.machinelearningplus.com/numpy-tutorial-python-part2/\]](https://www.machinelearningplus.com/numpy-tutorial-python-part2/), where I will explicit on the functionalities that is an essential toolkit of data analysis.

### Related



[\[https://www.machinelearningplus.com/numpy-tutorial-python-part2/\]](https://www.machinelearningplus.com/numpy-tutorial-python-part2/)

Numpy Tutorial Part 2 - Vital Functions for Data Analysis

[\[https://www.machinelearningplus.com/numpy-tutorial-python-part2/\]](https://www.machinelearningplus.com/numpy-tutorial-python-part2/)

February 14, 2018

In "Python"



[https://www.machinelearningplus.com/numpy-exercises-python/\]](https://www.machinelearningplus.com/numpy-exercises-python/)

101 NumPy Exercises for Data Analysis [Python]

[https://www.machinelearningplus.com/numpy-exercises-python/\]](https://www.machinelearningplus.com/numpy-exercises-python/)

February 26, 2018

In "Python"



[https://www.machinelearningplus.com/pandas-exercises-python/\]](https://www.machinelearningplus.com/pandas-exercises-python/)

101 Pandas Exercises for Data Analysis

[https://www.machinelearningplus.com/pandas-exercises-python/\]](https://www.machinelearningplus.com/pandas-exercises-python/)

April 27, 2018

In "Python"

You may also like:

[output\\_22\\_1](#)

[22\\_Density\\_Plot\\_Matplotlib](#)

[Kolmogorov-Smirnov](#)

[101 NumPy Exercises for Data Analysis \[Python\] - ML+](#)

[Time Series Analysis in Python - A Comprehensive Guide with Examples - ML+](#)

[How Naive Bayes Algorithm Works? \(with example and full code\) | ML+](#)

[Python debugging with pdb](#)

[Gensim Tutorial - A Complete Beginners Guide](#)



Tags: [Data Manipulation](https://www.machinelearningplus.com/tag/data-manipulation/) (<https://www.machinelearningplus.com/tag/data-manipulation/>), [Numpy](https://www.machinelearningplus.com/tag/numpy/) (<https://www.machinelearningplus.com/tag/numpy/>), [Python](https://www.machinelearningplus.com/tag/python/) (<https://www.machinelearningplus.com/tag/python/>)

#### Sponsored Links

**Become a Data scientist. No technical background required.**

Great Learning

**2 bed homes in Borivali(E) starting ₹1.8 Cr + Taxes.**

Rustomjee Summit, Borivali (E)

**Ready 2.5/3 BHK Apartments in Pimpri-Chinchwad, Pune from Rs. 95 Lacs\***

Mahindra Lifespaces, Pune

**This Photo Has Not Been Edited, Look Closer At The Hole**

Trendchaser

**Hair care in monsoon? These tips will tell you how!**

Skin & Hair Academy

**People from India cannot believe these flight prices**

Travel Deals Shop

23 Comments   machinelearningplus.com

 Login ▾

 Recommend    Tweet    Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



**Ankit Kumar** • 3 months ago

One of the best blogs on the internet...thanks for sharing your work with us. :)

^ | ▾ • Reply • Share ›



**Rivanda** • a year ago

Aciu uz straipsni!

^ | ▾ • Reply • Share ›

Feedback





**Selva Prabhakaran** Mod ➔ Rivanda • 10 months ago

Ačiū!

^ | v • Reply • Share ›



**Inayatullah Baloch** • a year ago

sir, you are great. I like your all posts . all are very informative and easy to understand.

^ | v • Reply • Share ›



**Azhar** • a year ago

well explained with examples.  
was very useful.

^ | v • Reply • Share ›



**arvind kumar** • a year ago

lots of thank for Numpy

^ | v • Reply • Share ›



**Anton** • a year ago

Author, I am not native english speaker, but really, it's great. I understand everything without reread. Information clear and exactly tell what for what. Thanks. because i buy and read some books, but it looks like that people who write them, don't want that somebody understand them. Thanks, thanks, thanks, thanks.

P.S If you have book i would like to have it on my shelf, if not, you are the one why must write it.

^ | v • Reply • Share ›



**Esquitossomose** • a year ago

Thank You, very usefull

^ | v • Reply • Share ›



**manju** • a year ago

Very well explained !! Thanks

^ | v • Reply • Share ›



**Arun Ajay** • a year ago

Love it!

^ | v • Reply • Share ›



**Saran** • a year ago

Thanks Selva. Can you please explain how dimension is calculated?

^ | v • Reply • Share ›



**Selva Prabhakaran** ➔ Saran • a year ago

Hi Saran,

Dimension refers to the number of levels of involved in he array/list.

For example:

[1,2,3,4] --> 1D

[[1,2,3][4,5,6]] --> 2D

[[[1,2], [3,4]], [[5,6][7,8]]] --> 3D

Feedback



^ | v • Reply • Share ›



**Saran** • a year ago

It is mentioned that numpy array cannot have different datatypes but as mentioned above through dtype =object ,it allows us to create values of different datatypes.

^ | v • Reply • Share ›



**Selva Prabhakaran** → Saran • a year ago

That's True and not. Because, all the items will be of dtype object.

^ | v • Reply • Share ›



**Hari** • a year ago

very good and it will cover all topics

^ | v • Reply • Share ›



**Nagendra** • a year ago

very nice tutorial

^ | v • Reply • Share ›



**Praveen K** • a year ago

Very detailed explanation.

^ | v • Reply • Share ›



**kulvinder** • a year ago

well explained and really helpful lot of thankz !!!

^ | v • Reply • Share ›



**Selva Prabhakaran** → kulvinder • a year ago

Glad to know, Thanks :)

^ | v • Reply • Share ›



**Tan** • a year ago

Excellent

^ | v • Reply • Share ›



**Selva Prabhakaran** → Tan • a year ago

Thanks Tan!

^ | v • Reply • Share ›



**Poovanna** • a year ago

Very well written. I think this tutorial series can be extended to include more important methods in numpy.

Also, it seems there's a mistake in Section 6.1- in the example that illustrates the difference between 'flatten()' and 'ravel()'

^ | v • Reply • Share ›



**Selva Prabhakaran** Mod → Poovanna • a year ago

Thanks Poovanna, the error has been corrected now.

By 'more important methods' what methods are you referring to?

Feedback



#### Sponsored Links

**Become a Data scientist. No technical background required.**

Great Learning

**2 bed homes in Borivali(E) starting ₹1.8 Cr + Taxes.**

Rustomjee Summit, Borivali (E)

**Ready 2.5/3 BHK Apartments in Pimpri-Chinchwad, Pune from Rs. 95 Lacs\***

Mahindra Lifespaces, Pune

**This Photo Has Not Been Edited, Look Closer At The Hole**

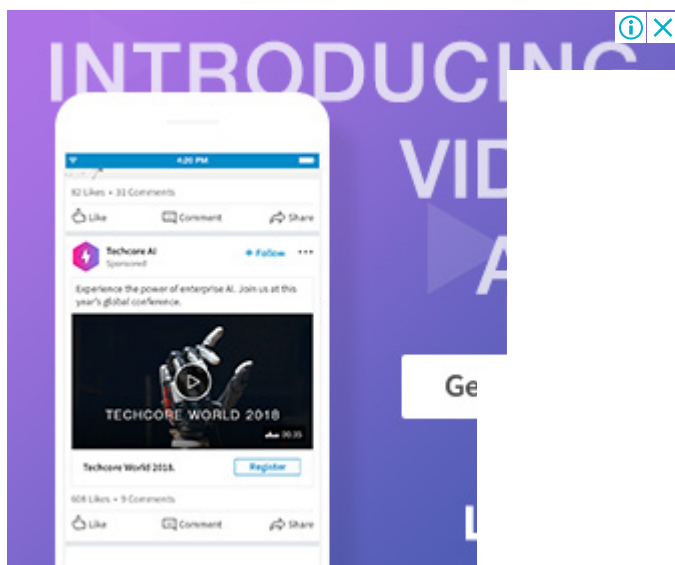
Trendchaser

**Hair care in monsoon? These tips will tell you how!**

Skin & Hair Academy

**People from India cannot believe these flight prices**

Travel Deals Shop



 <https://www.ezoic.com/what-is-ezoic/>

report this ad

## Recent Posts

[Vector Autoregression \(VAR\) – Comprehensive Guide with Examples in Python](https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/)  
(<https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/>).

Mahalonobis Distance – Understanding the math with examples (python).  
(<https://www.machinelearningplus.com/statistics/mahalanobis-distance/>).

[datetime in Python – Simplified Guide with Clear Examples](https://www.machinelearningplus.com/python/datetime-python-examples/)  
(<https://www.machinelearningplus.com/python/datetime-python-examples/>).

[Python Logging – Simplest Guide with Full Code and Examples](https://www.machinelearningplus.com/python/python-logging-guide/)  
(<https://www.machinelearningplus.com/python/python-logging-guide/>).

Matplotlib Histogram – How to Visualize Distributions in Python  
(<https://www.machinelearningplus.com/plots/matplotlib-histogram-python-examples/>).

[ARIMA Model – Complete Guide to Time Series Forecasting in Python](https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/)  
(<https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>).

Time Series Analysis in Python – A Comprehensive Guide with Examples  
(<https://www.machinelearningplus.com/time-series/time-series-analysis-python/>).

Matplotlib Tutorial – A Complete Guide to Python Plot w/ Examples  
(<https://www.machinelearningplus.com/plots/matplotlib-tutorial-complete-guide-python-plot-examples/>).

Topic modeling visualization – How to present the results of LDA models?  
(<https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/>),

[Top 50 matplotlib Visualizations – The Master Plots \(with full python code\).](https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/)  
(<https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/>).

List Comprehensions in Python – My Simplified Guide  
(<https://www.machinelearningplus.com/python/list-comprehensions-in-python/>).

Python @Property Explained – How to Use and When? (Full Examples).  
(<https://www.machinelearningplus.com/python/python-property/>).

How Naive Bayes Algorithm Works? (with example and full code).  
([https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-](https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works/)

[works-with-example-and-full-code/](#).

[Parallel Processing in Python – A Practical Guide with Examples](#)

[\(https://www.machinelearningplus.com/python/parallel-processing-python/\)](https://www.machinelearningplus.com/python/parallel-processing-python/).

[Cosine Similarity – Understanding the math and how it works \(with python codes\)](#)

[\(https://www.machinelearningplus.com/nlp/cosine-similarity/\)](https://www.machinelearningplus.com/nlp/cosine-similarity/).

[Gensim Tutorial – A Complete Beginners Guide](#)

[\(https://www.machinelearningplus.com/nlp/gensim-tutorial/\)](https://www.machinelearningplus.com/nlp/gensim-tutorial/).

[Lemmatization Approaches with Examples in Python](#)

[\(https://www.machinelearningplus.com/nlp/lemmatization-examples-python/\)](https://www.machinelearningplus.com/nlp/lemmatization-examples-python/).

[Feature Selection – Ten Effective Techniques with Examples](#)

[\(https://www.machinelearningplus.com/machine-learning/feature-selection/\)](https://www.machinelearningplus.com/machine-learning/feature-selection/).

[101 Pandas Exercises for Data Analysis](#) [\(https://www.machinelearningplus.com/python/101-pandas-exercises-python/\)](https://www.machinelearningplus.com/python/101-pandas-exercises-python/).

[LDA in Python – How to grid search best topic models?](#)

[\(https://www.machinelearningplus.com/nlp/topic-modeling-python-sklearn-examples/\)](https://www.machinelearningplus.com/nlp/topic-modeling-python-sklearn-examples/).

[Topic Modeling with Gensim \(Python\)](#) [\(https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/\)](https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/).

[Python debugging with pdb](#) [\(https://www.machinelearningplus.com/python/python-debugging/\)](https://www.machinelearningplus.com/python/python-debugging/).

[Caret Package – A Practical Guide to Machine Learning in R](#)

[\(https://www.machinelearningplus.com/machine-learning/caret-package/\)](https://www.machinelearningplus.com/machine-learning/caret-package/).

[101 NumPy Exercises for Data Analysis \(Python\)](#)

[\(https://www.machinelearningplus.com/python/101-numpy-exercises-python/\)](https://www.machinelearningplus.com/python/101-numpy-exercises-python/).

[Numpy Tutorial Part 2 – Vital Functions for Data Analysis](#)

[\(https://www.machinelearningplus.com/python/numpy-tutorial-python-part2/\)](https://www.machinelearningplus.com/python/numpy-tutorial-python-part2/).

## Tags

[@property](#) [\(https://www.machinelearningplus.com/tag/property/\)](https://www.machinelearningplus.com/tag/property/)

[Bigrams](#) [\(https://www.machinelearningplus.com/tag/bigrams/\)](https://www.machinelearningplus.com/tag/bigrams/)

[Classification](#) [\(https://www.machinelearningplus.com/tag/classification/\)](https://www.machinelearningplus.com/tag/classification/)

[Corpus](#) [\(https://www.machinelearningplus.com/tag/corpus/\)](https://www.machinelearningplus.com/tag/corpus/)

[Cosine Similarity](#) [\(https://www.machinelearningplus.com/tag/cosine-similarity/\)](https://www.machinelearningplus.com/tag/cosine-similarity/)

[Data Manipulation](#) [\(https://www.machinelearningplus.com/tag/data-manipulation/\)](https://www.machinelearningplus.com/tag/data-manipulation/)



[Debugging](https://www.machinelearningplus.com/tag/debugging/) (<https://www.machinelearningplus.com/tag/debugging/>)

[Doc2Vec](https://www.machinelearningplus.com/tag/doc2vec/) (<https://www.machinelearningplus.com/tag/doc2vec/>)

[Evaluation Metrics](https://www.machinelearningplus.com/tag/evaluation-metrics/) (<https://www.machinelearningplus.com/tag/evaluation-metrics/>)

[FastText](https://www.machinelearningplus.com/tag/fasttext/) (<https://www.machinelearningplus.com/tag/fasttext/>)

[Feature Selection](https://www.machinelearningplus.com/tag/feature-selection/) (<https://www.machinelearningplus.com/tag/feature-selection/>)

[Gensim](https://www.machinelearningplus.com/tag/gensim/) (<https://www.machinelearningplus.com/tag/gensim/>)

[klaR](https://www.machinelearningplus.com/tag/klar/) (<https://www.machinelearningplus.com/tag/klar/>)

[LDA](https://www.machinelearningplus.com/tag/lda/) (<https://www.machinelearningplus.com/tag/lda/>)

[Lemmatization](https://www.machinelearningplus.com/tag/lemmatization/) (<https://www.machinelearningplus.com/tag/lemmatization/>)

[Linear Regression](https://www.machinelearningplus.com/tag/linear-regression/) (<https://www.machinelearningplus.com/tag/linear-regression/>)

[Logistic](https://www.machinelearningplus.com/tag/logistic/) (<https://www.machinelearningplus.com/tag/logistic/>)

[LSI](https://www.machinelearningplus.com/tag/lsi/) (<https://www.machinelearningplus.com/tag/lsi/>)

[Matplotlib](https://www.machinelearningplus.com/tag/matplotlib/) (<https://www.machinelearningplus.com/tag/matplotlib/>)

[Multiprocessing](https://www.machinelearningplus.com/tag/multiprocessing/) (<https://www.machinelearningplus.com/tag/multiprocessing/>)

[Naive Bayes](https://www.machinelearningplus.com/tag/naive-bayes/) (<https://www.machinelearningplus.com/tag/naive-bayes/>)

[NLP](https://www.machinelearningplus.com/tag/nlp/) (<https://www.machinelearningplus.com/tag/nlp/>)

[NLTK](https://www.machinelearningplus.com/tag/nltk/) (<https://www.machinelearningplus.com/tag/nltk/>)

[Numpy](https://www.machinelearningplus.com/tag/numpy/) (<https://www.machinelearningplus.com/tag/numpy/>)

[Pandas](https://www.machinelearningplus.com/tag/pandas/) (<https://www.machinelearningplus.com/tag/pandas/>)

[Parallel Processing](https://www.machinelearningplus.com/tag/parallel-processing/) (<https://www.machinelearningplus.com/tag/parallel-processing/>)

[Phraser](https://www.machinelearningplus.com/tag/phraser/) (<https://www.machinelearningplus.com/tag/phraser/>)

[Practice Exercise](https://www.machinelearningplus.com/tag/practice-exercise/) (<https://www.machinelearningplus.com/tag/practice-exercise/>)

## Python

(<https://www.machinelearningplus.com/tag/python/>)

[R](https://www.machinelearningplus.com/tag/r/) (<https://www.machinelearningplus.com/tag/r/>)

[Regex](https://www.machinelearningplus.com/tag/regex/) (<https://www.machinelearningplus.com/tag/regex/>)

[Regression](https://www.machinelearningplus.com/tag/regression/) (<https://www.machinelearningplus.com/tag/regression/>)

[Residual Analysis](https://www.machinelearningplus.com/tag/residual-analysis/) (<https://www.machinelearningplus.com/tag/residual-analysis/>)

[Scikit Learn](https://www.machinelearningplus.com/tag/scikit-learn/) (<https://www.machinelearningplus.com/tag/scikit-learn/>)

[Significance Tests](https://www.machinelearningplus.com/tag/significance-tests/) (<https://www.machinelearningplus.com/tag/significance-tests/>)

[Soft Cosine Similarity](https://www.machinelearningplus.com/tag/soft-cosine-similarity/) (<https://www.machinelearningplus.com/tag/soft-cosine-similarity/>)

[spaCy](https://www.machinelearningplus.com/tag/spacy/) (<https://www.machinelearningplus.com/tag/spacy/>)

[Summarization](https://www.machinelearningplus.com/tag/summarization/) (<https://www.machinelearningplus.com/tag/summarization/>)

[TaggedDocument](https://www.machinelearningplus.com/tag/taggeddocument/) (<https://www.machinelearningplus.com/tag/taggeddocument/>)

Feedback





[TextBlob \(https://www.machinelearningplus.com/tag/textblob/\)](https://www.machinelearningplus.com/tag/textblob/)

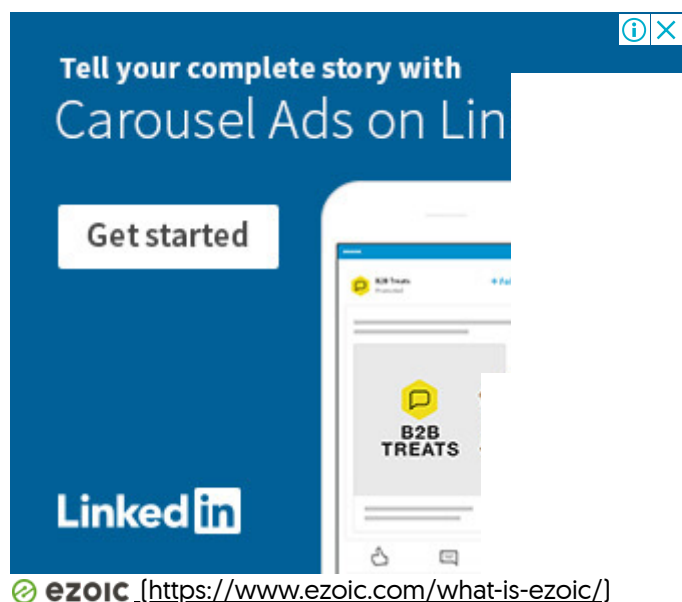
[TFIDF \(https://www.machinelearningplus.com/tag/tfidf/\)](https://www.machinelearningplus.com/tag/tfidf/)

[Time Series \(https://www.machinelearningplus.com/tag/time-series/\)](https://www.machinelearningplus.com/tag/time-series/)

[Topic Modeling \(https://www.machinelearningplus.com/tag/topic-modeling/\)](https://www.machinelearningplus.com/tag/topic-modeling/)

[Visualization \(https://www.machinelearningplus.com/tag/visualization/\)](https://www.machinelearningplus.com/tag/visualization/)

[Word2Vec \(https://www.machinelearningplus.com/tag/word2vec/\)](https://www.machinelearningplus.com/tag/word2vec/)



<https://www.ezoic.com/what-is-ezoic/>

report this ad



<https://www.ezoic.com/what-is-ezoic/>

report this ad

[HOME \(https://www.machinelearningplus.com/\)](https://www.machinelearningplus.com/)

[/ ALL POSTS \(https://www.machinelearningplus.com/blog/\)](https://www.machinelearningplus.com/blog/)

[/ DATA MANIPULATION \(https://www.machinelearningplus.com/category/data-manipulation/\)](https://www.machinelearningplus.com/category/data-manipulation/)

[/ PREDICTIVE MODELING](https://www.machinelearningplus.com/category/predictive-modeling/)

[\(https://www.machinelearningplus.com/category/predictive-modeling/\)](https://www.machinelearningplus.com/category/predictive-modeling/)

Feedback



/ [STATISTICS](https://www.machinelearningplus.com/category/statistics/) [\(https://www.machinelearningplus.com/category/statistics/\)](https://www.machinelearningplus.com/category/statistics/)  
/ [NLP](https://www.machinelearningplus.com/category/nlp/) [\(https://www.machinelearningplus.com/category/nlp/\)](https://www.machinelearningplus.com/category/nlp/)  
/ [PYTHON](https://www.machinelearningplus.com/category/python/) [\(https://www.machinelearningplus.com/category/python/\)](https://www.machinelearningplus.com/category/python/)  
/ [PLOTS](https://www.machinelearningplus.com/category/plots/) [\(https://www.machinelearningplus.com/category/plots/\)](https://www.machinelearningplus.com/category/plots/)  
/ [TIME SERIES](https://www.machinelearningplus.com/category/time-series/) [\(https://www.machinelearningplus.com/category/time-series/\)](https://www.machinelearningplus.com/category/time-series/)  
/ [CONTACT US](https://www.machinelearningplus.com/contact-us/) [\(https://www.machinelearningplus.com/contact-us/\)](https://www.machinelearningplus.com/contact-us/)

© Copyright by Machine Learning Plus | All rights reserved | [Privacy Policy](https://www.machinelearningplus.com/privacy-policy/)  
[\(https://www.machinelearningplus.com/privacy-policy/\)](https://www.machinelearningplus.com/privacy-policy/) | [Terms of Use](https://www.machinelearningplus.com/terms-of-use/)  
[\(https://www.machinelearningplus.com/terms-of-use/\)](https://www.machinelearningplus.com/terms-of-use/) | [About](https://www.machinelearningplus.com/about)  
[\(https://www.machinelearningplus.com/about\)](https://www.machinelearningplus.com/about)

