NUMPY INTRODUCTION WITH EXAMPLES

If we study Pandas, we have to study NumPy, because Pandas includes NumPy. Here, I'll introduce NumPy and share some basic functions.

(This tutorial is part of our Pandas Guide. Use the right-hand menu to navigate.)

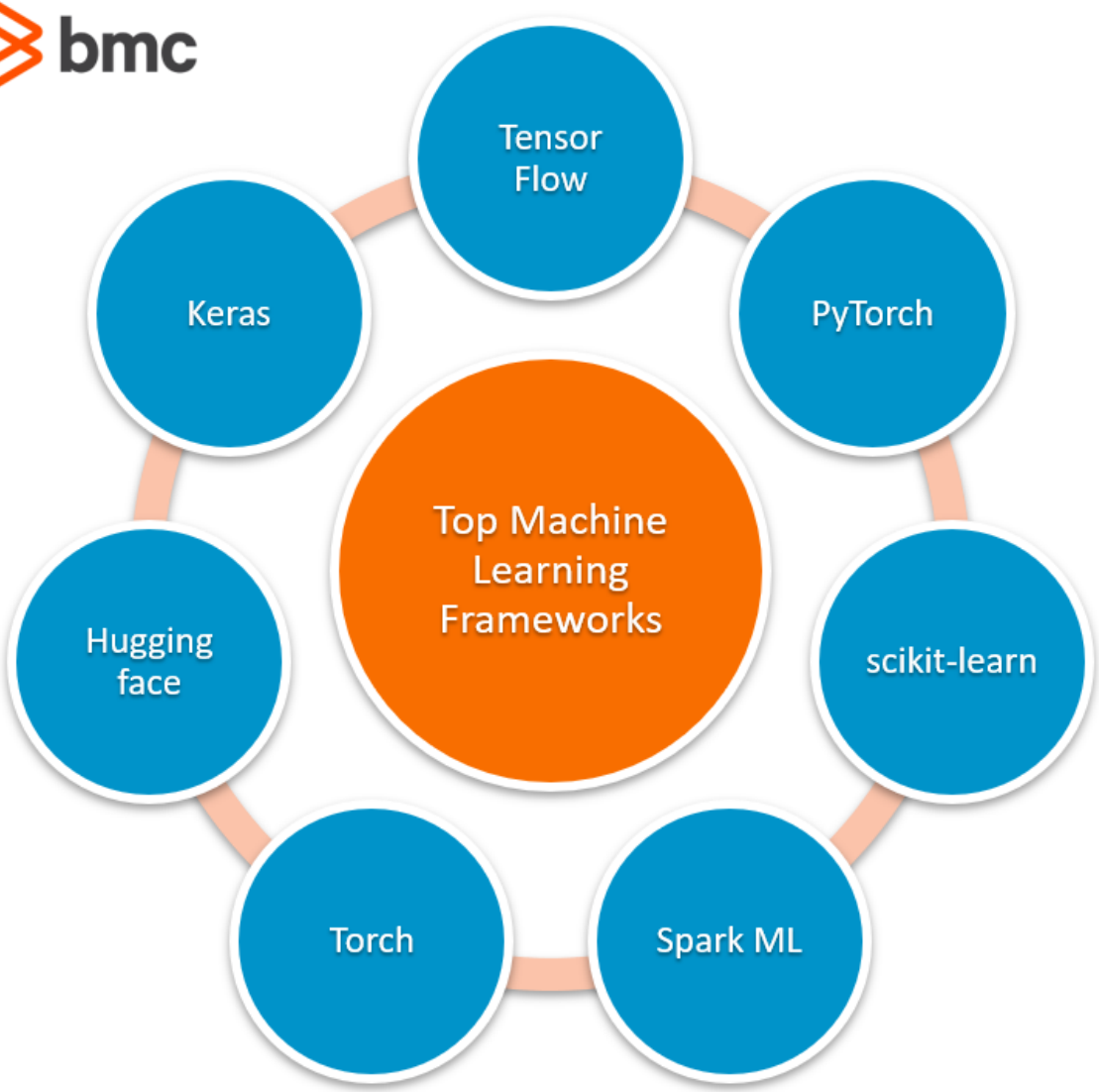
What is NumPy?

NumPy is a package that create arrays. It lets you make arrays of numbers with different precision and scale, plus string, so it is especially useful for scientific computing.

Python by itself only has floats, integers, and imaginary numbers. But NumPy expands what Python can do because it handles:

- 32-bit numbers
- 15 big numbers
- Signed numbers
- Unsigned numbers
- And more

But that's not the only reason to use NumPy. It's designed for efficiency and scale, making it the workhouse for large machine learning (ML) libraries like TensorFlow.



Now, let's take a look at some basic functions of NumPy arrays.

Creating a NumPy array

Create an array with **np.array(<array>)**.

Don't put np.array(1,2,3,4,5) as 1,2,3,4,5 is not an array. NumPy would interpret the items after the commas as parameters to the **array()** function.

This creates an array:

```
import numpy as np
arr = np.array()
arr
```

Results:

```
array()
```

Array shape

An array has shape, just like, for example, a 2x2 array, 2x1 array, etc.

Query the shape like this:

```
arr.shape
```

You should call this a **vector** if you want to understand this better as it's not 3x1—because it only has one dimension, and a blank is not a dimension.

```
(3,)
```

This is 3x1 since it is an array of 3 arrays of dimension 1x1.

```
arr = np.array(,,])
arr.shape
```

Results:

```
(3, 1)
```

Reshaping an array

You can reshape an array of shape m x n into any combination that is a divisor of m x n. This array of shape (6,) can be reshaped to 2x3 since 2*3=6 divides 6.

```
import numpy as np
arr = np.array().reshape(2,3)
print(arr)
```

Results:

```


]
```

Arange

Notice that this function is not **arrange** but **arange**, as in **array range**. Use it to file an array with numbers. (There are lots of ways to do that, a topic that we will cover in a subsequent post.)

```
import numpy as np
arr = np.arange(5)
arr
```

Results in:

```
array()
```

Slice

Slicing an array is a difficult topic that becomes easier with practice. Here are some simple examples.

Take this array.

```
arr = np.array().reshape(2,3)
arr
```

Which looks like this:

```
array(,
])
```

(While you could say this has 2 rows and 3 columns to make it easier to understand, that's not technically correct. When you have more than two dimensions, the concept or rows and columns goes away. So that's why it's better to say dimensions and axes.)

This **slice** operations means start at the second position of the first axis and go to the end:

```
arr
```

Results in:

```
array(])
```

This starts at the beginning and goes to the end:

```
arr
```

Results in:

```
array(,
])
```

Add a comma to specify which column:

```
arr
```

Results in:

```
array()
```

Select along the other axis like this:

```
arr
```

Results in:

```
array()
```

Select a single element.

```
arr
```

Results:

```
4
```

Step

```
arr = np.array()
arr
```

```
array()
```

That concludes this introduction.

Related reading

- BMC Machine Learning & Big Data Blog
- Pandas Data Types
- Python Development Tools: Your Python Starter Kit
- Data Visualization Guide, a series of tutorials
- Guide to ML with TensorFlow & Keras
- Snowflake Guide