Group of
Horribly
Optimistic
STatisticians

NumPy

Intro to Data Science

Maksymilian Norkiewicz & Jędrzej Ogrodowski

Goal for today’s meeting

To introduce you to idea of how data is interpreted at low level, grasp what tools do we use and why.

At the end I will talk about hardware and GPU computing and how you can prepare your environments for comfortable work.

To configure workspace:

- install VSCode extension for Jupyter Notebook
- create and activate venv
- install requirements.txt

Jupyter Notebooks

For learning experimentation purpose of these classes we will use **Jupyter Notebooks**.

Jupyter Notebooks allows to combine markdown notes with executable code, making it easy to document your workflow by running code in sections.

I recommend this format while experimenting with code. For final implementations it is better to use .py files.

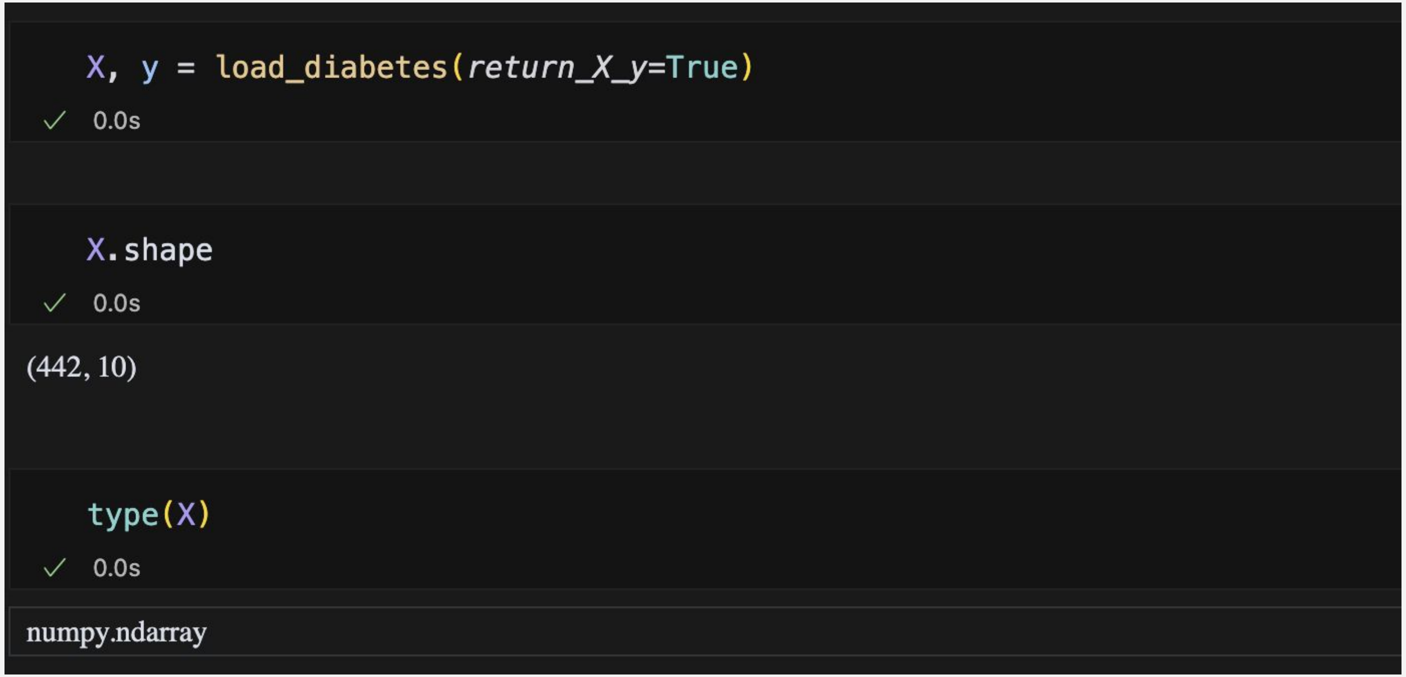
How computers perceive data?

As Data Scientists, it’s important to view data at different levels of abstraction. Raw data can come in many forms - text, images, sensor readings. Each observation can be represented as a vector of features.

Dataset can be viewed as a matrix composed of feature vectors.

Because of that we can use that as an advantage to perform calculations and analysis faster.

How computers perceive data?



```
X, y = load_diabetes(return_X_y=True)
✓ 0.0s
```

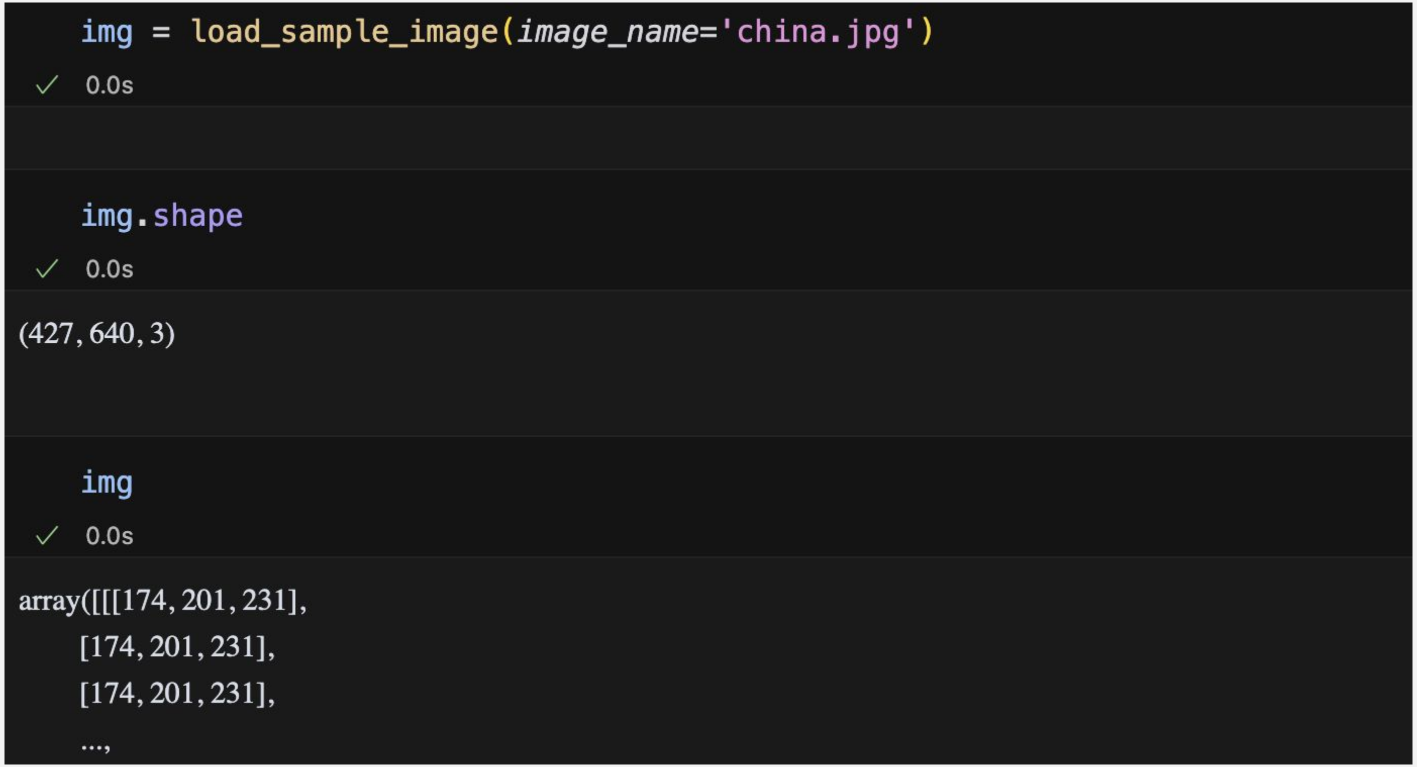
```
X.shape
✓ 0.0s
```

```
(442, 10)
```

```
type(X)
✓ 0.0s
```

```
numpy.ndarray
```

How computers perceive data?



```
img = load_sample_image(image_name='china.jpg')
✓ 0.0s
```

```
img.shape
✓ 0.0s
```

```
(427, 640, 3)
```

```
img
✓ 0.0s
```

```
array([[[174, 201, 231],
        [174, 201, 231],
        [174, 201, 231],
        ...,
```

NumPy - foundation of all other tools

NumPy is Python library used to work with multidimensional arrays and advanced mathematical functions.

Some of these functions are:
- shape manipulation
- linear algebra
- basic statistics
- generating sequences, random values etc.

Why NumPy?

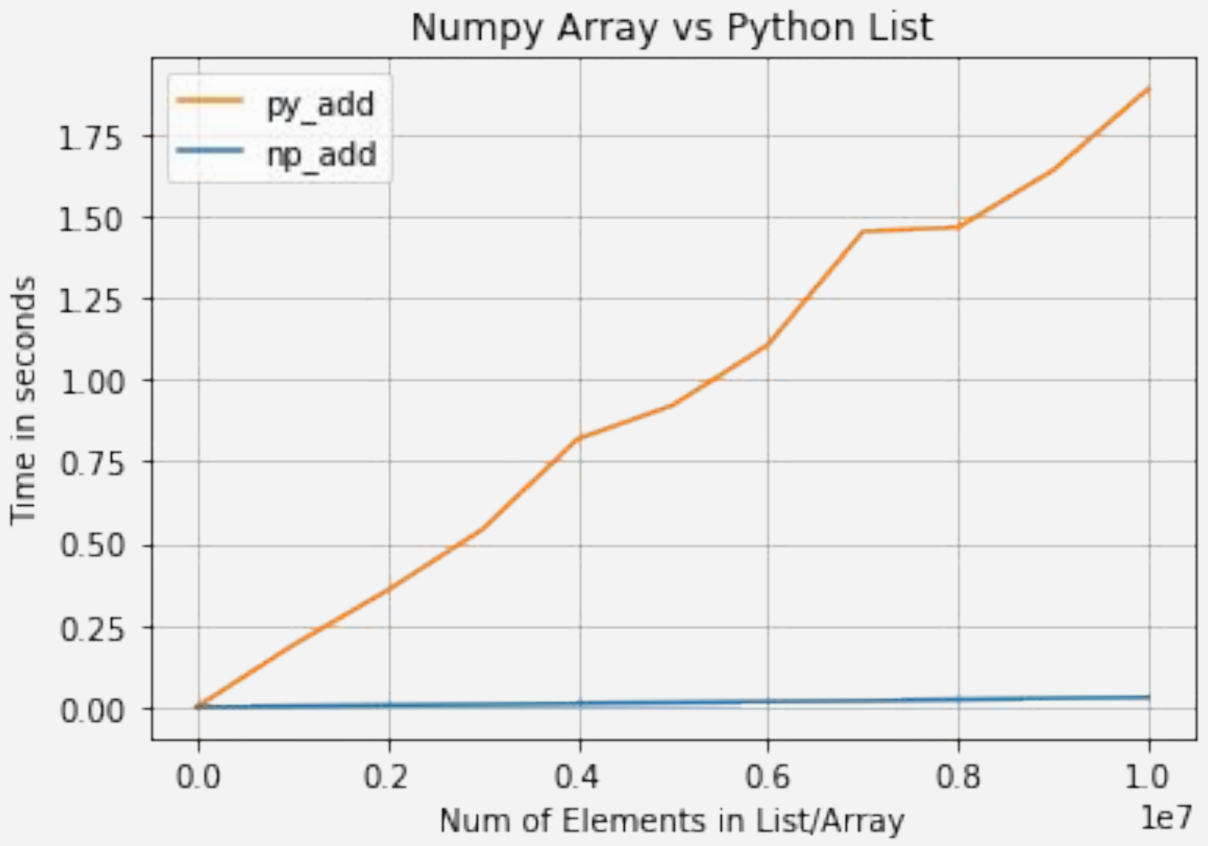
I mentioned that NumPy is used to work with arrays. Arrays plays crucial role in Data Science as they are foundation of storing data. Whether we are working with raw data, tabular data, time series, images or tensors - arrays are primary data structures for computation.

NumPy arrays are optimized, faster and memory efficient. They are solving pythons speed and performance issues.

Under the hood it takes advantage of vectorization, contiguous memory block and low-level iterations.

Advantages of NumPy

	Python List	**NumPy Array**
Size	Allocates memory for pointer, ref counter, value for each element	Compact, fixed-type values
Convenience	Many situations requires using loops	Most work replaced by vectorized operations
Functionality	Limited built-in math	Set of transformations and tools (e.g. FFT)
Speed	Comes with flexibility but slow	C-optimized operations - fast


Numpy Array vs Python List
py_add
np_add
Time in seconds
Num of Elements in List/Array
1e7
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0.0
0.2
0.4
0.6
0.8
1.0

NumPy built-in functions

```
np.sum(array)
np.mean(array)
np.max(array), np.min(array)
np.std(array)
np.var(array)
np.isnan(array)
np.power(array)
```

Linear algebra with NumPy

NumPy provides a wide range of tools that make algebraic and numerical calculations much easier. Algorithms rely heavily on matrix operations.

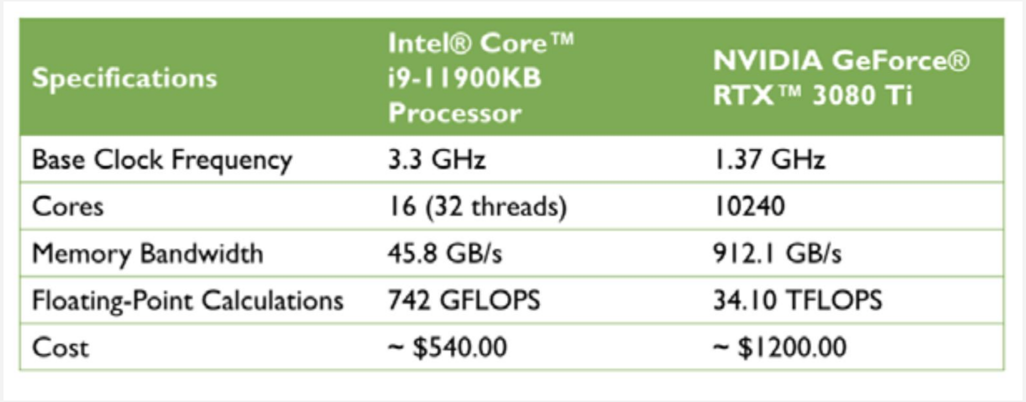
```
rankOfMatrix = np.linalg.matrix_rank(matrix)
traceOfMatrix = np.trace(matrix)
detOfMatrix = np.linalg.det(matrix)
inversedMatrix = np.linalg.inv(matrix)
dotProduct = np.dot(vector1, vector2)
arrTransposed = array.T
```

Examples in Notebook

GPU

Due to optimizations of computational performance, when training models, processing large datasets etc. gpu’s are used.

You can think of graphic cards as a small cluster inside your machine. They offer more floating-point calculations and bigger memory bandwidth.



Specifications	Intel® Core™ i9-11900KB Processor	NVIDIA GeForce® RTX™ 3080 Ti
Base Clock Frequency	3.3 GHz	1.37 GHz
Cores	16 (32 threads)	10240
Memory Bandwidth	45.8 GB/s	912.1 GB/s
Floating-Point Calculations	742 GFLOPS	34.10 TFLOPS
Cost	~ $540.00	~ $1200.00

Utilizing GPU and distributed programming

In modern Data Science where we deal with large amounts of data and sometimes huge models, platforms that allow us to compute on GPU or on distributed compute nodes are widely used.

Good starting point is **Google Colab**.

In commercial work one can use all-in-one cloud environments like **Azure Machine Learning** or **AWS Sagemaker** for developing models and **Databricks** or **Spark** for big data handling.

How to use GPU in code

We can improve NumPy performance by using open-source library CuPy. It allows programmers to leverage GPU without rewriting code (same syntax as NumPy, different import).

There are more popular libraries that aims at GPU acceleration such as PyTorch and Tensorflow. They are mainly used for Deep Learning tasks where high computational resources are required. However due to time constraints, these libraries are out of scope.

How to use GPU in code

Using NumPy:

```
import numpy as np
x = np.random.rand(1000, 1000)
y = np.dot(x, x.T)
```

Using CuPy:

```
import cupy as cp
x_gpu = cp.random.rand(1000, 1000)
y_gpu = cp.dot(x_gpu, x_gpu.T)
```

How to use Colab

Thank you