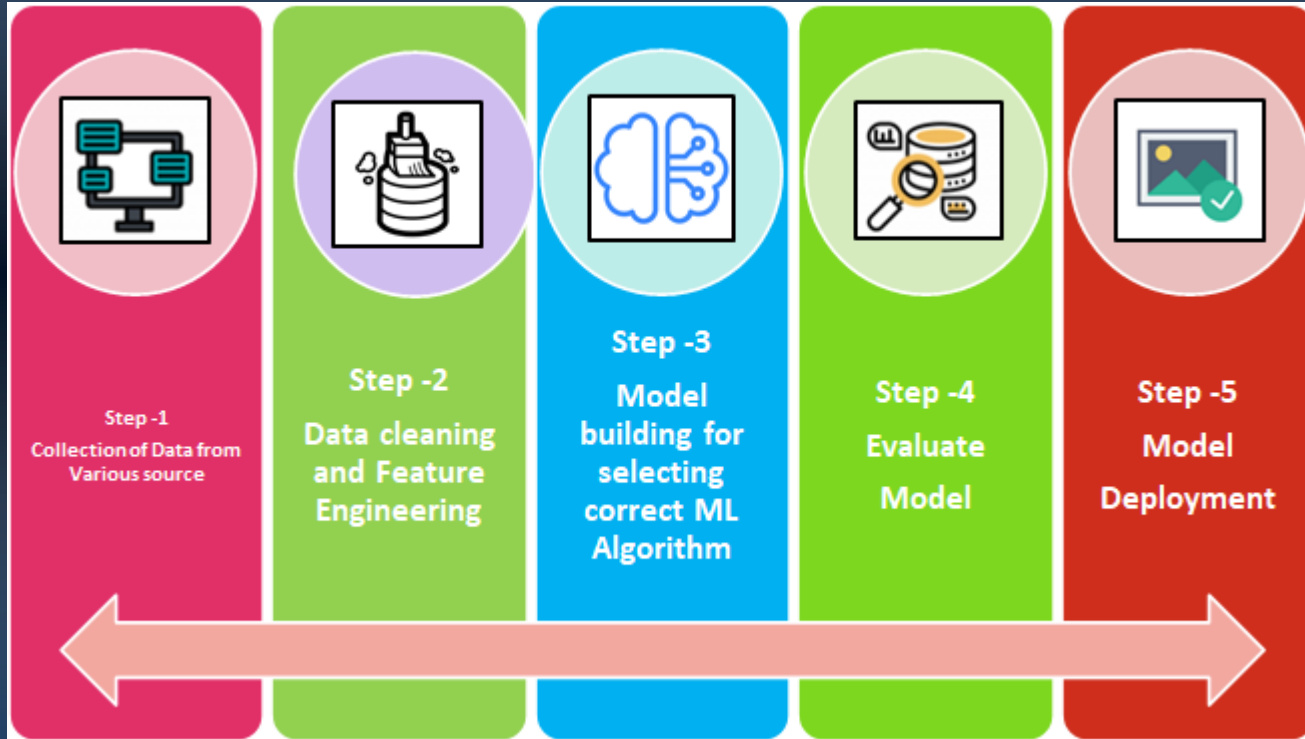




Data Management & Visualization

Eng- Mohamed Khaled Idris
Eng- Mayar Swilam

Review:



Python Libraries

1. Scientifics Computing Libraries



Pandas

(Data structures & tools)



NumPy

(Arrays & matrices)



SciPy

(Integrals, solving differential equations, optimization)

Python Libraries

2. Visualization Libraries



Matplotlib

(plots & graphs, most popular)



Seaborn

(plots : heat maps, time series, violin plots)

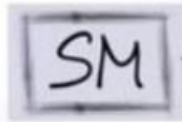
Python Libraries

3. Algorithmic libraries



Scikit-learn

(Machine Learning : regression, classification,...)



Statsmodels

(Explore data, estimate statistical models, and perform statistical tests.)

Python Libraries



TensorFlow



TENSORFLOW & PYTORCH

Data Management

- In order to manage data through python, there are many ways to do it !
- Also, you may need to do some mathematical operations to matrices, lists or any structure
- You can simply manage your data as the same as SQL

HOW THEN ?

NumPy



What is NumPy ?

- NumPy is an open-source Python library that facilitates efficient numerical operations on large quantities of data.
- There are a few functions that exist in NumPy that we use on pandas DataFrames.
- For us, the most important part about NumPy is that pandas is built on top of it. So, NumPy is a dependency of Pandas.

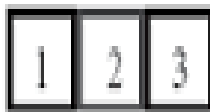


NumPy Arrays

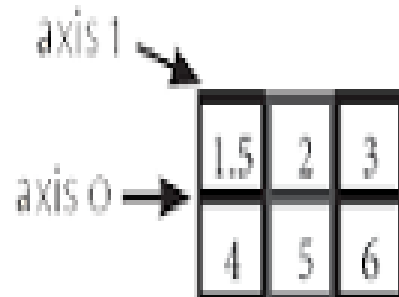
- **NumPy arrays are unique in that they are more flexible than normal Python lists.**
- **They are called ndarrays since they can have any number (n) of dimensions (d).**
- **They hold a collection of items of any one data type and can be either a vector (one-dimensional) or a matrix (multi-dimensional).**
- **NumPy arrays allow for fast element access and efficient data manipulation.**

NumPy Arrays

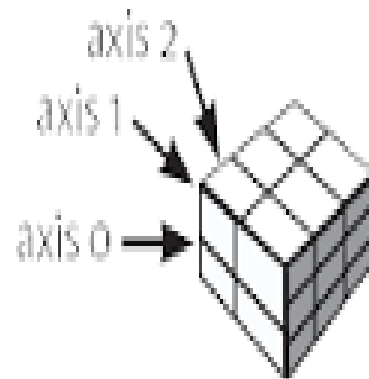
1D array



2D array



3D array



Python For Data Science NumPy Cheat Sheet

Learn NumPy online at www.DataCamp.com

Numpy

The NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays

1D array



2D array



3D array



> Creating Arrays

```
## s = np.array([1,2,3])
## s = np.array([1,2,3,4], dtype = float)
## s = np.array([1,2,3,4,5,6,7,8,9,10,11,12], dtype = float)
```

Initial Placeholders

```
## a.shape #Array dimensions
## len(a) #length of array
## a.ndim #Number of array dimensions
## a.size #Number of array elements
## a.dtype #Data type of array elements
## a.dtype.name #Name of data type
## a.astype(int) #Convert an array to a different type
```

> Data Types

```
## np.int8 #Signed byte-integer type
## np.float64 #Standard double-precision floating point
## np.complex #Complex numbers represented by 128 floats
## np.bool #Boolean type storing TRUE and FALSE values
## np.object #Python object type
## np.str_ #Fixed-length string type
## np.ndarray #Fixed-length ndarray type
```

> Array Mathematics

```
## a + b #Addition
## a - b #Subtraction
## a * b #Element-wise multiplication
## a / b #Element-wise division
## a // b #Floor division
## a % b #Modulus
## a ** b #Power
```

https://images.datacamp.com/image/upload/v1676302459/Marketing/Blog/Numpy_Cheat_Sheet.pdf

Pandas



What is Pandas ?

- *Pandas* is a very popular library for working with data (its goal is to be the most powerful and flexible open-source tool, and in our opinion, it has reached that goal).
- DataFrames are at the center of pandas. A DataFrame is structured like a table or spreadsheet. The rows and the columns both have indexes, and you can perform operations on rows or columns separately.

DataFrame

- A pandas DataFrame can be easily changed and manipulated. Pandas has helpful functions for handling missing data, performing operations on columns and rows, and transforming data.
- If that wasn't enough, a lot of SQL functions have counterparts in pandas, such as join, merge, filter by, and group by.
- With all of these powerful tools, it should come as no surprise that pandas is very popular among data scientists.

Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

Use the following import convention:

```
>>> import pandas as pd
```

> Pandas Data Structures

Series

A one-dimensional labeled array capable of holding any data type

```
>>> s = pd.Series([2, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

Index →

a	2
b	-5
c	7
d	4

Dataframe

A two-dimensional labeled data structure with columns of potentially different types

Columns →

	Country	Capital	Population
Index →	Belgium	Brussels	11000000
	India	New Delhi	1300000000
	Brazil	Brasilia	200000000

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [11000000, 1300000000, 200000000]}
>>> df = pd.DataFrame(data, columns=['Country', 'Capital', 'Population'])
```

> Dropping

```
>>> s.drop(['a', 'c']) #Drop values from series (axis=0)
>>> df.drop('Country', axis=1) #Drop values from columns (axis=1)
```

> Asking For Help

```
>>> help(pd.Series.iloc)
```

> Sort & Rank

```
>>> df.sort_index() #Sort by labels along an axis
>>> df.sort_values('Country') #Sort by the values along an axis
>>> df.rank() #Assign ranks to entries
```

```
>>> pd.read_excel('File.xlsx')
>>> df.to_excel('d:/myDataFrom.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xls = pd.ExcelFile('File.xlsx')
>>> df = df.read_excel(xls, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite://memory:')
>>> pd.read_sql('SELECT * FROM my_table', engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query('SELECT * FROM my_table', engine)

read_sql() is a convenience wrapper around read_sql_table() and read_sql_query()
>>> df.to_sql('mydb', engine)
```

Also see NumPy Arrays

> Selection

Getting

```
>>> df.iloc[2] #Select single row of subset of rows
Country Brazil
Capital Brasilia
Population 207000000
>>> df.iloc[:, 'Capital'] #Select a single column of subset of columns
0 Brasilia
1 New Delhi
2 Brasilia
>>> df.ix[2, 'Capital'] #Select rows and columns
'New Delhi'
```

Boolean indexing

```
>>> s[s > 5] #Returns a where value is not 0
>>> s[s < -5] | (s > 10) #s where value is < -5 or > 10
>>> df[df['Population'] > 1000000000] #Use Zfilter to subset DataFrame
```

Sorting

```
>>> s['a'] = 6 #Set index 0 of Series s to 6
```

Summary

```
>>> df.sum() #Sum of values
>>> df.mean() #Calculate sum of values
>>> df.std()/df.max() #Standard/Maximum values
>>> df.describe() #Minimum/Maximum index values
>>> df.describe() #Summary statistics
>>> df.mean() #Mean of values
>>> df.median() #Median of values
```

> Applying Functions

```
>>> f = lambda x: x**2
>>> df.apply(f) #Apply function
>>> df.applymap(f) #Apply function element-wise
```

> Data Alignment

Internal Data Alignment

The indices that don't overlap:

```
2, 3], index=['a', 'b', 'c'])
```

Operations with Fill Methods

The alignment yourself with the help of the fill methods:

```
s-r()
```

```
s-r()
```

```
s-r()
```

Datacamp Cheat Sheet (Pandas)

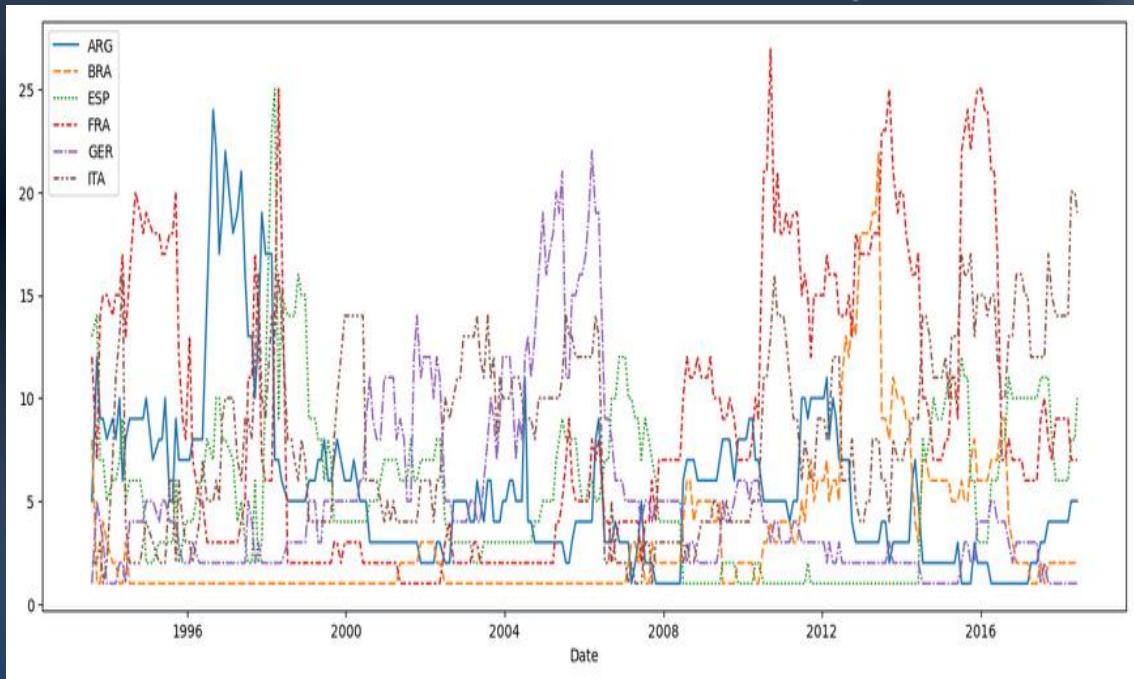
https://images.datacamp.com/image/upload/v1676302204/Marketing/Blog/Pandas_Cheat_Sheet.pdf

- A picture is worth a thousand words
- For exploratory data analysis
- Communicate data clearly
- Share unbiased representation of data
- Support recommendations to different stakeholders

Why Visualization ?

The important of Visualization

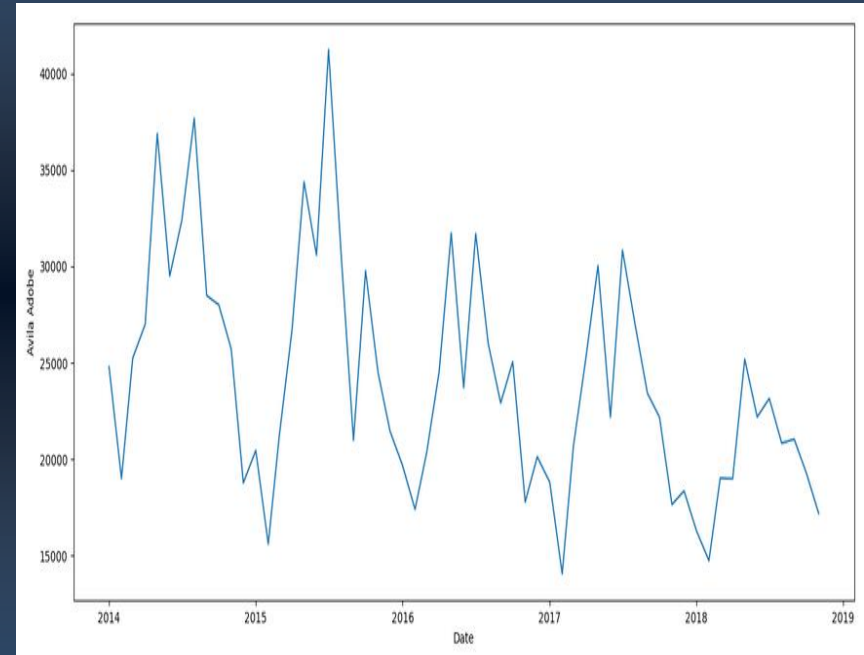
- Considering only the years represented in the figure, which countries spent at least 5 consecutive years in the #1 ranked spot?

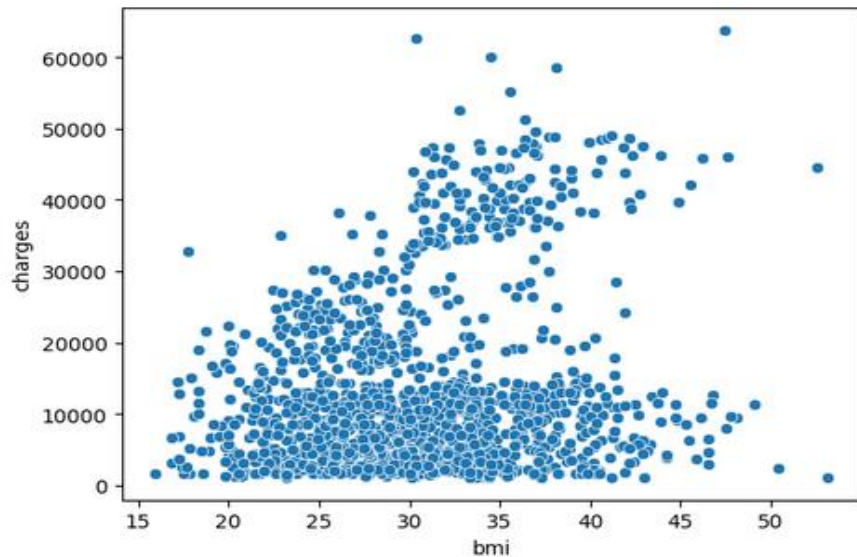


Does Avila Adobe get more visitors:

- **in September-February (in LA, the fall and winter months), or**
- **in March-August (in LA, the spring and summer)?**

Using this information, when should the museum staff additional seasonal employees?

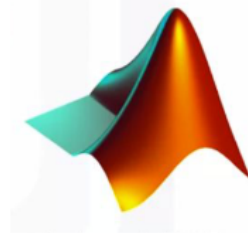




IS THERE ANY
RELATION
BETWEEN BODY
MASS INDEX
AND INSURANCE
CHARGES ?

Matplotlib

- Neurobiologist
- Part of a team analyzing **Electrocorticography Signals (ECoG)**
 - **Electrocorticography** is the process of recording electrical activity in the brain
- The team
 - used a proprietary software (**MATLAB** based version) for analysis
 - had only one license and were taking turns in using it
- **John** replace the proprietary software with **Matplotlib**



Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of backends formats and interactive environments across platforms.

> Prepare The Data

1D Data

```
import numpy as np
x = np.linspace(0, 10, 100)
y = np.sin(x)
z = np.cos(x)
```

2D Data or Images

```
import numpy as np
x = np.linspace(0, 10, 100)
y = np.linspace(0, 10, 100)
z = np.sin(x) * np.cos(y)
fig = plt.figure()
ax = fig.gca()
ax.imshow(z.reshape((100, 100)))
```

> Create Plot

```
fig = plt.figure(figsize=(10, 10))
```

Figure

```
fig = plt.Figure()
fig = plt.Figure(figsize=(10, 10))
```

Axes

Adding a data series to a plot is as easy as using the `plot()` method of the `Figure` object.

2D Data

```
fig, ax = plt.subplots()
ax = ax.imshow(z.reshape((100, 100)))
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_title('2D Data')
ax.grid(True)
```

Vector Fields

```
import numpy as np
x = np.linspace(0, 10, 100)
y = np.linspace(0, 10, 100)
```

Data Distributions

```
fig, ax = plt.subplots()
ax = ax.hist(x)
```

Plot Anatomy

```
fig, ax = plt.subplots()
ax = ax.plot(x, y)
```

Workflow

The basic steps to creating plots with Matplotlib are:

1. Import Data 2. Create Plot 3. Plot 4. Customize Plot 5. Save Plot 6. Show Plot

Markers

```
fig, ax = plt.subplots()
ax = ax.scatter(x, y)
```

Linestyles

```
fig, ax = plt.subplots()
ax = ax.plot(x, y, linestyle='solid')
```

Text & Annotations

```
fig, ax = plt.subplots()
ax = ax.text(x, y, 'Text')
```

Datacamp Cheat Sheet (Matplotlib)

https://images.datacamp.com/image/upload/v1676360378/Marketing/Blog/Matplotlib_Cheat_Sheet.pdf



Questions

Thank You

