U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence
Lecture 5b: Machine Learning Tools

Luís Paulo Reis

lpreis@fe.up.pt

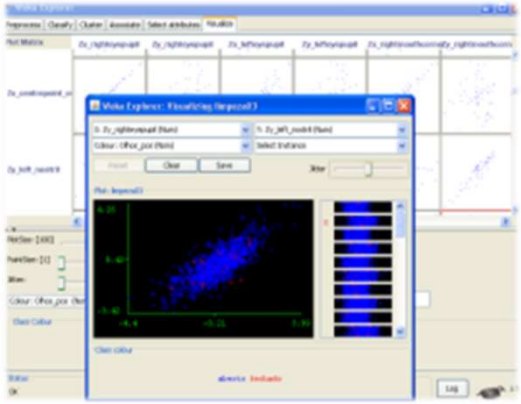
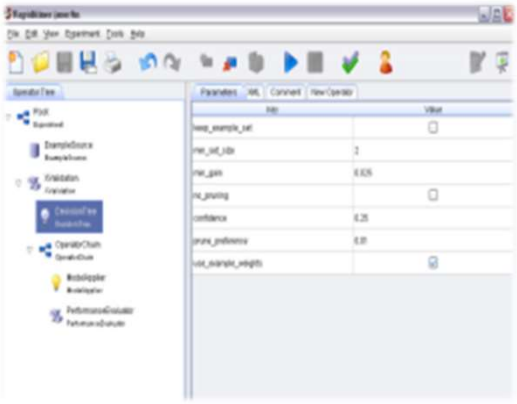
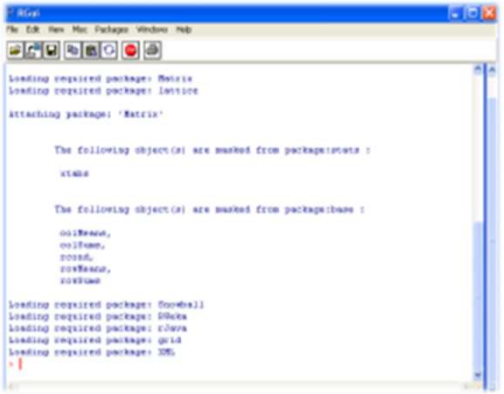
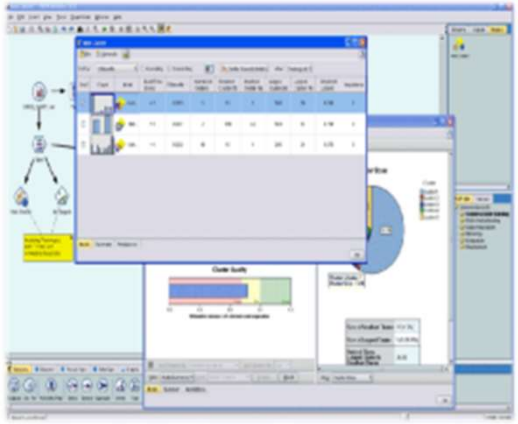
Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence

Agenda

- Data Science/Machine Learning Tools
- Python Programming Language
- Anaconda
- Jupyter Notebook and JupyterLab
- NumPy and SciPy
- Pandas
- Scikit-Learn
- MapPlotLib and Seaborn
- Conclusions

Interesting Machine Learning Tools/Libraries/Software Packages

Python Libraries; R; RapidMiner; WEKA; IBM SPSS Modeler; KNIME; H20.AI, SAS Enterp.Miner; Statsoft Statistica Data Miner; Insightful Miner; Pandas; Scikit-Learn; Theano; Keras; TensorFlow; PyTorch

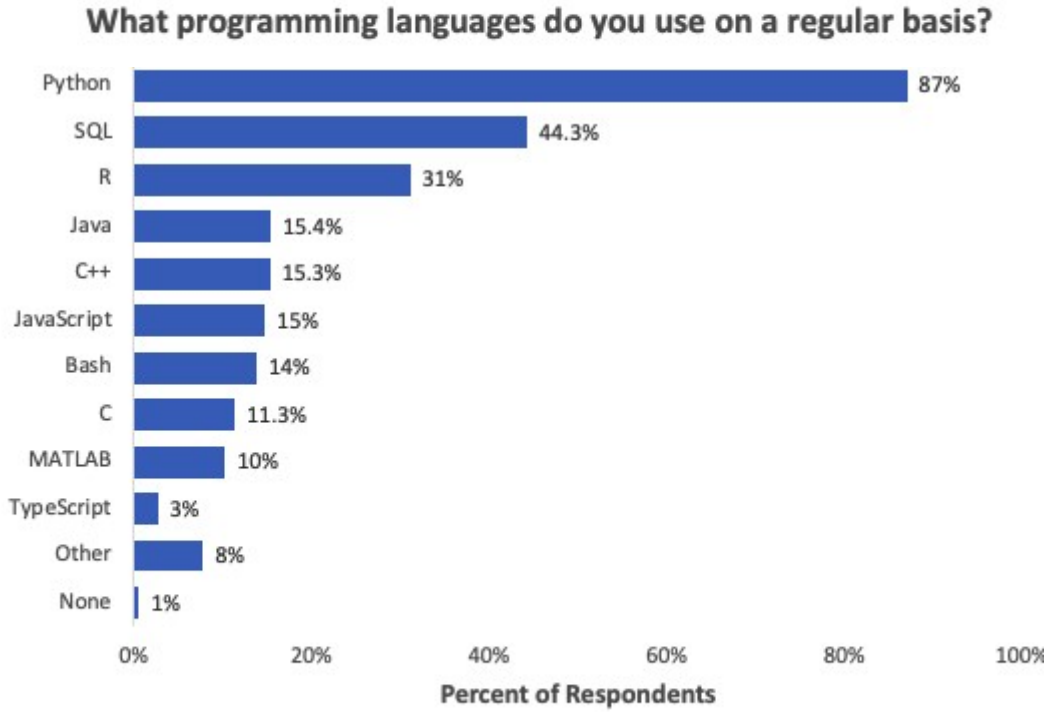

Python

- Python is an interpreted high-level general-purpose programming language

- Python's design philosophy emphasizes code readability with its notable use of significant indentation.
- Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects
- Python is dynamically-typed and garbage-collected
- It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming
- Python is often described as a "batteries included" language due to its comprehensive standard library

Python

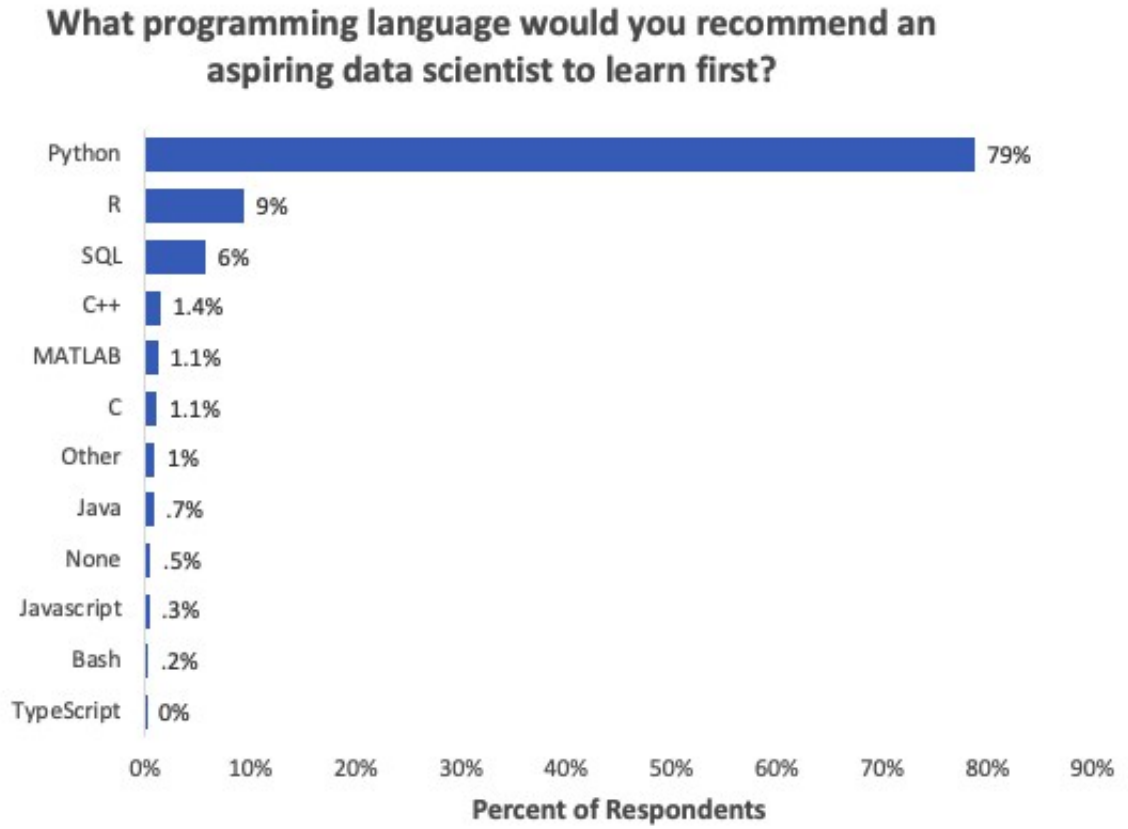
- Python is a perfect choice for computer science students trying to get started in Data Science/ Machine Learning
- **Python is the most popular programming language choice among Data Scientists**



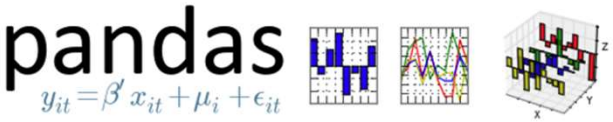
Note: Data are from the 2019 Kaggle ML and Data Science Survey. You can learn more about the study here: https://www.kaggle.com/c/kaggle-survey-2019.

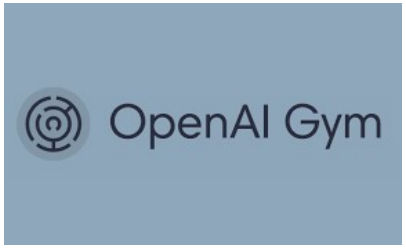
Python

- **Python is the most recommended programming language for Data Science**
- Nearly 8 in 10 professionals would recommend Python as the programming language for data science to learn first

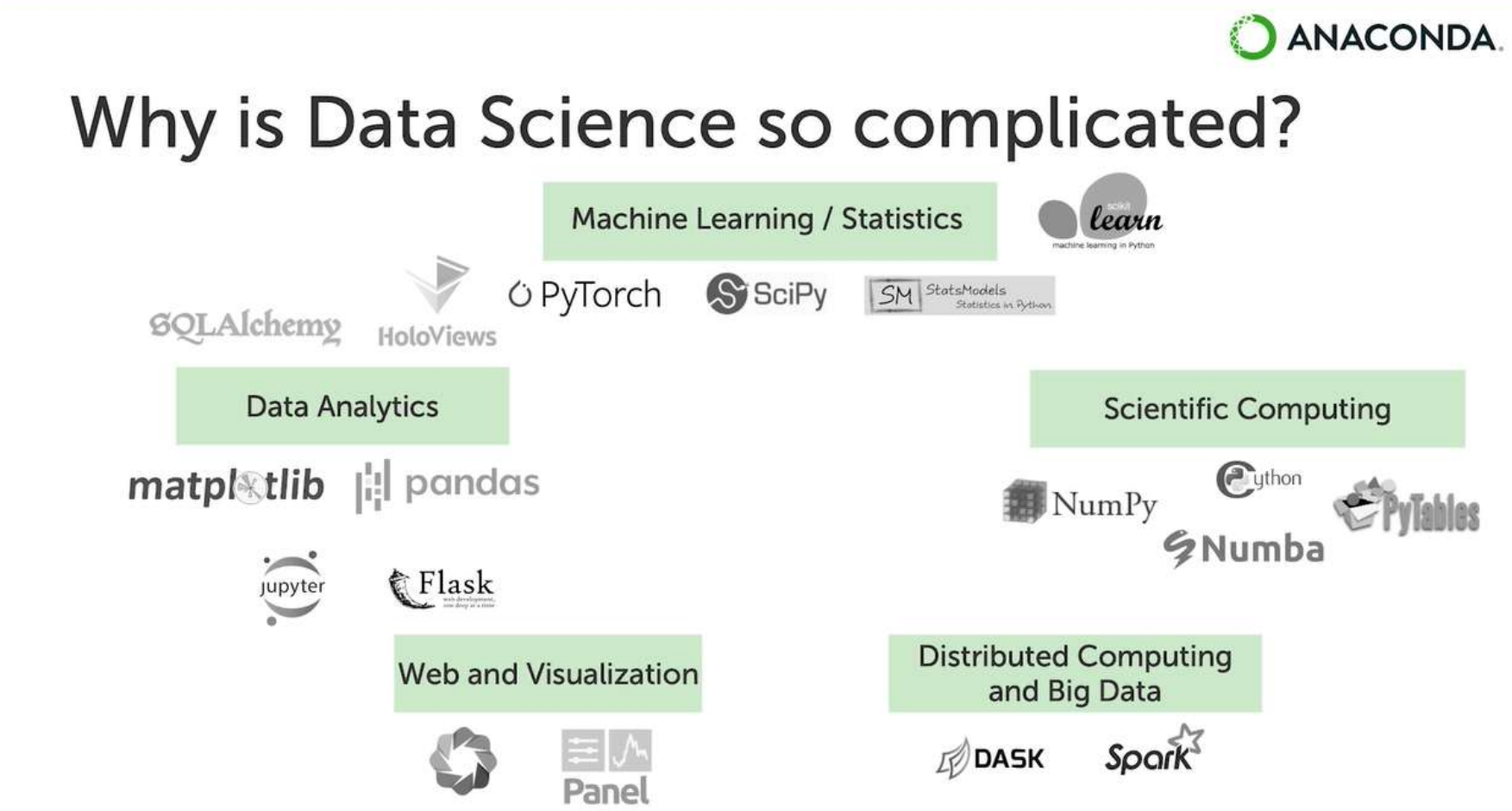


Interesting Machine Learning Tools





Anaconda


ANACONDA.
Why is Data Science so complicated?
Machine Learning / Statistics
scikit learn
machine learning in Python
SQLAlchemy
HoloViews
PyTorch
SciPy
SM StatsModels
Statistics in Python
Data Analytics
Scientific Computing
matplotlib
pandas
NumPy
Cython
PyTables
Numba
jupyter
Flask
Web and Visualization
Distributed Computing and Big Data
Panel
DASK
Spark

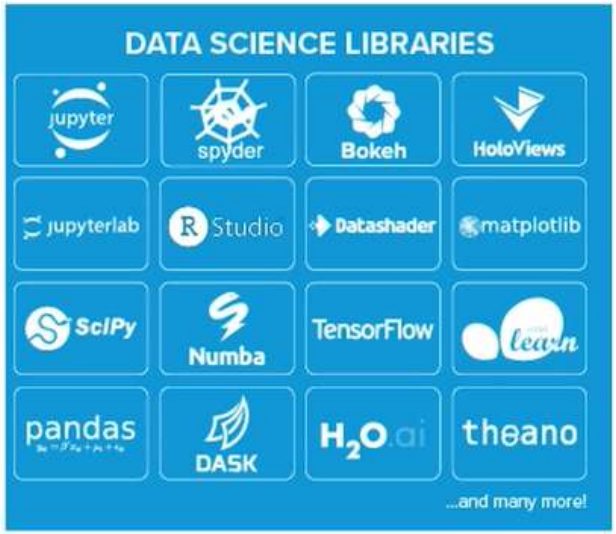
Anaconda

- Anaconda includes most of the useful Packages!
- https://www.anaconda.com/products/individual

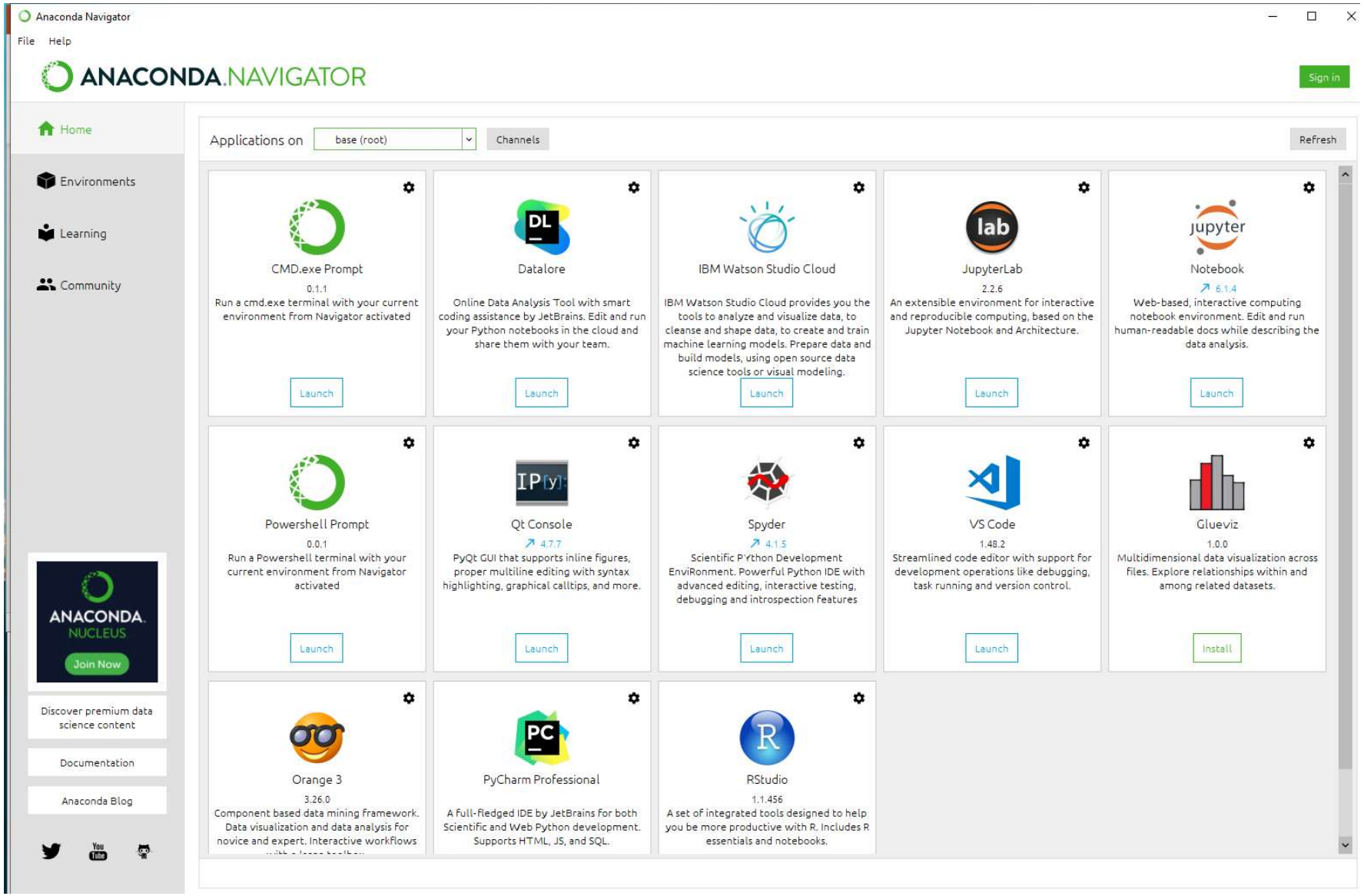
ANACONDA

Anaconda Distribution

- Thousands of curated packages
  - Analysis
  - Visualization
  - Modeling
- Mac OS, Linux, and Windows
  - 200+ packages pre-installed
  - It "just works"



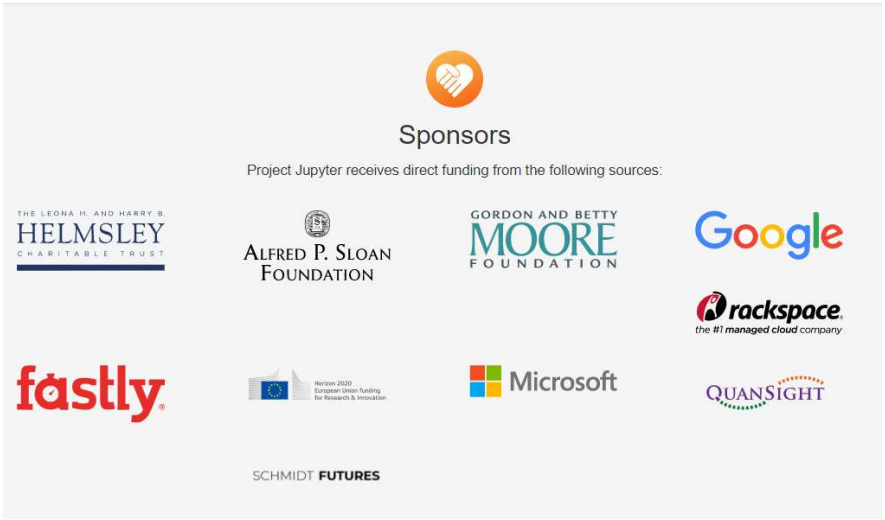
Anaconda Navigator



Project Jupyter

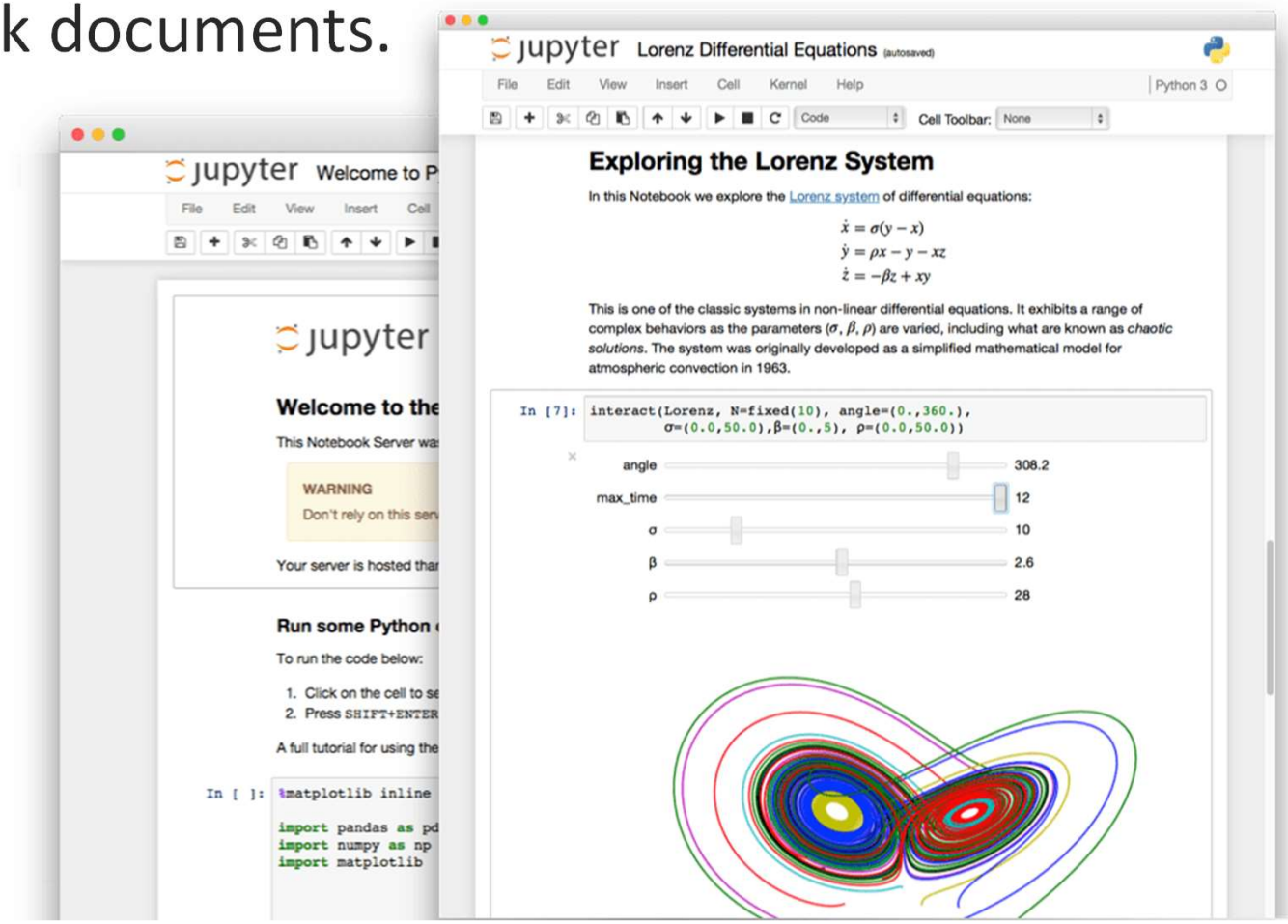
"Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages."

https://jupyter.org/



Jupyter Notebook

- Jupyter Notebook is a web-based interactive computational environment for creating Jupyter notebook documents.

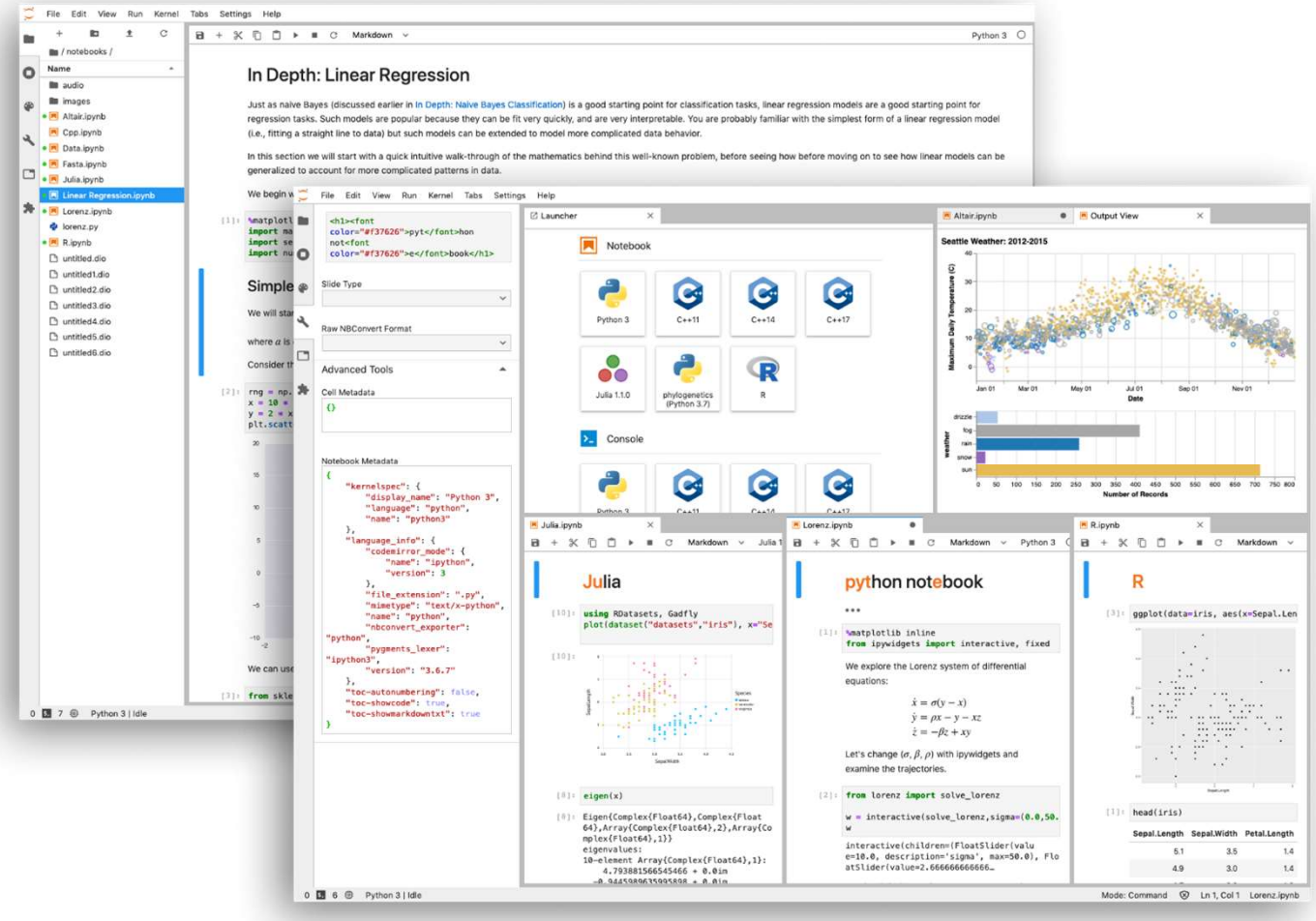


Jupyter Notebook

- Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
- It supports several languages like Python (IPython), Julia, R, etc.
- Mostly used for data analysis, data visualization, and other interactive, exploratory computing.
- Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
- For beginners in data science, jupyter notebook is more preferred;
  - It only consists of a file browser and a (notebook) editor view, which is easier to use.
- When you get familiar with it and need more features, you can switch to JupyterLab.

JupyterLab

- Next-generation user interface, including notebooks. It offers more of an IDE-like experience.



JupyterLab

- Web-based interactive development environment for Jupyter notebooks, code, and data.
- JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning.
- JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones., open several notebooks or files (e.g., HTML, Text, Markdowns, etc.) as tabs in the same window.
- JupyterLab uses the same Notebook server and file format as the classic Jupyter Notebook to be fully compatible with the existing notebooks and kernels.
- The Classic Notebook and Jupyterlab can run side to side on the same computer (we can easily switch between the two interfaces).
- Interface of both Lab and notebook are similar, except the panel of the file system on the left side in Jupyter lab.

NumPy

Differences between NumPy arrays and standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically)
- Changing the size of ndarray creates a new array deleting original
- Elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. Exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences
- New scientific/mathematical Python-based packages are using NumPy arrays
- To efficiently use scientific/mathematical Python-based software needs to know how to use NumPy arrays
- https://numpy.org

SciPy

- Free and open-source Python library
- Used for scientific computing and technical computing
- Collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics, signal and image processing, and more
- Part of SciPy Stack, built on NumPy
- SciPy is a community-driven project
- Development happens on GitHub
- https://www.scipy.org/scipylib/

Pandas

- Pandas is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language

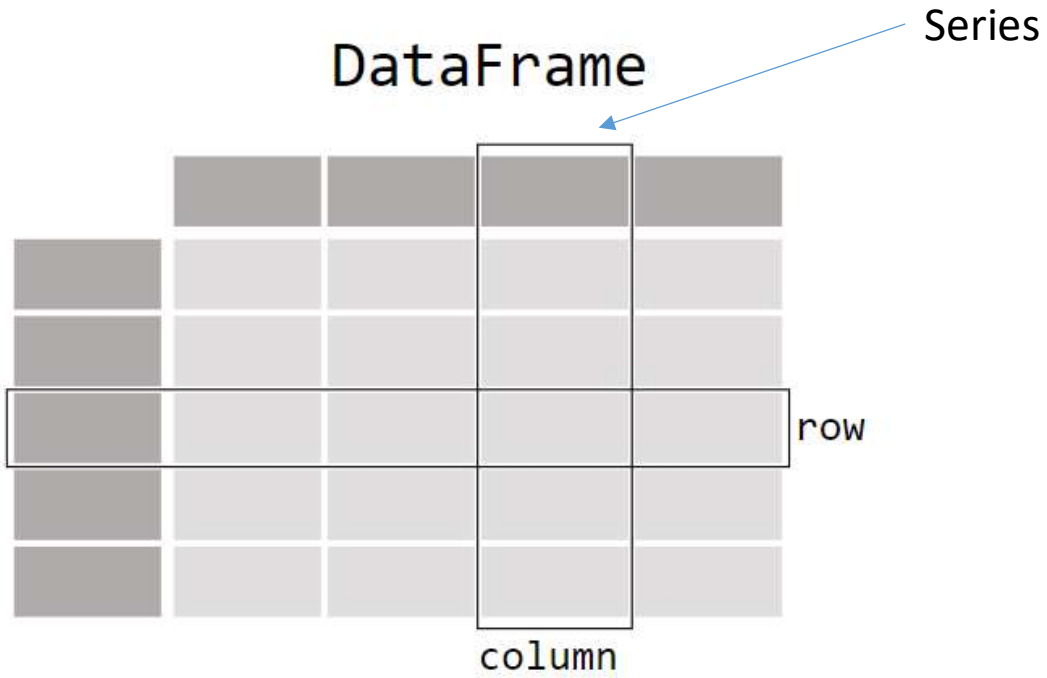
- Adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R)
- Provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- Allows handling missing data
- Highly optimized for performance
- https://pandas.pydata.org/

Pandas

```
In [1]: import pandas as pd
```

To load the pandas package and start working with it, import the package. The community agreed alias for pandas is `pd`, so loading pandas as `pd` is assumed standard practice for all of the pandas documentation.

pandas data table representation



A DataFrame is a 2-dimensional data structure that can store data of different types (including characters, integers, floating point values, categorical data and more) in columns. It is similar to a spreadsheet, a SQL table or the data.frame in R

Pandas

- A **DataFrame** is a two-dimensional array of values with both a row and a column index.
- A **Series** is a one-dimensional array of values with an index (essentially a single column within that **DataFrame)**

	Value
0	NJ
1	CA
2	TX
3	MD
4	OH
5	IL

Column Index

	State	City	Shape
0	NJ	Towaco	Square
1	CA	San Francisco	Oval
2	TX	Austin	Triangle
3	MD	Baltimore	Square
4	OH	Columbus	Hexagon
5	IL	Chicaco	Circle

Pandas

- Creating a Data Frame

```
pd.DataFrame(data = [
['NJ', 'Towaco', 'Square'],
['CA', 'San Francisco', 'Oval'],
['TX', 'Austin', 'Triangle'],
['MD', 'Baltimore', 'Square'],
['OH', 'Columbus', 'Hexagon'],
['IL', 'Chicago', 'Circle']
],
             columns = ['State', 'City', 'Shape'])


pd.Series(data=['NJ', 'CA', 'TX', 'MD', 'OH', 'IL'])


* Side Note: Python is able to make use of "white space" so the code
is easier to read - hence the new lines and spacing.
```

Pandas

```
auto.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

```
(397, 9)
```

The shape of our dataset, in rows and columns.

```
auto.tail(3)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
394	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
395	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
396	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

A look at the last three rows of our dataset.

```
auto.shape
```

```
(397, 9)
```

The shape of our dataset, in rows and columns.

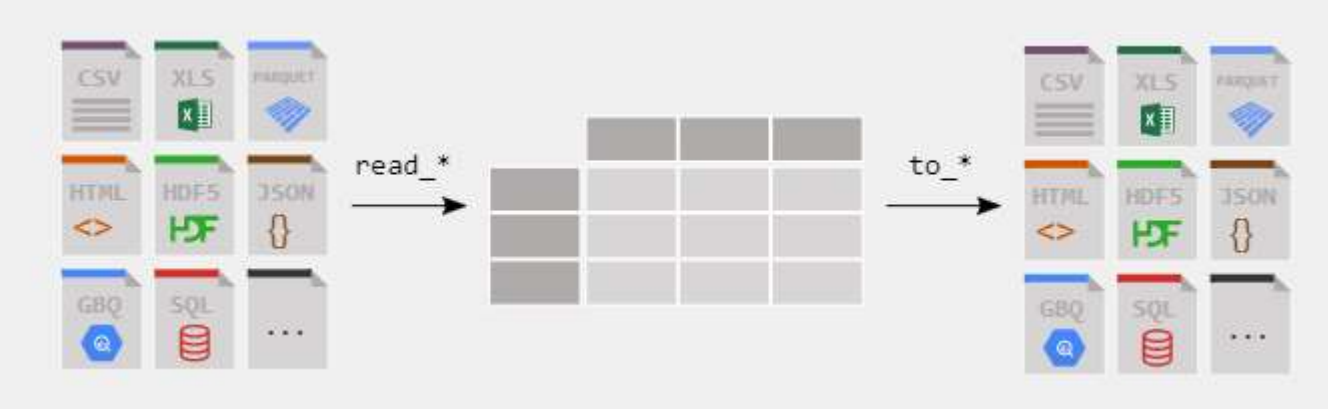
Pandas

```
auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 9 columns):
mpg             397 non-null float64
cylinders       397 non-null int64
displacement    397 non-null float64
horsepower      397 non-null object
weight          397 non-null int64
acceleration    397 non-null float64
year            397 non-null int64
origin          397 non-null int64
name            397 non-null object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.0+ KB
```

Pandas

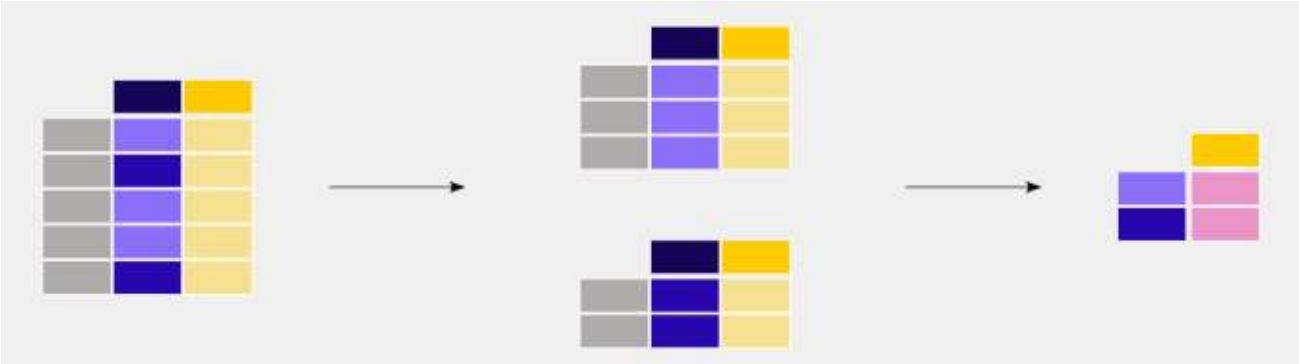
- Pandas supports the integration with many file formats or data sources out of the box (csv, excel, sql, json, parquet,…). Importing data from each of these data sources is provided by function with the prefix read_*. Similarly, the to_* methods are used to store data.



Pandas

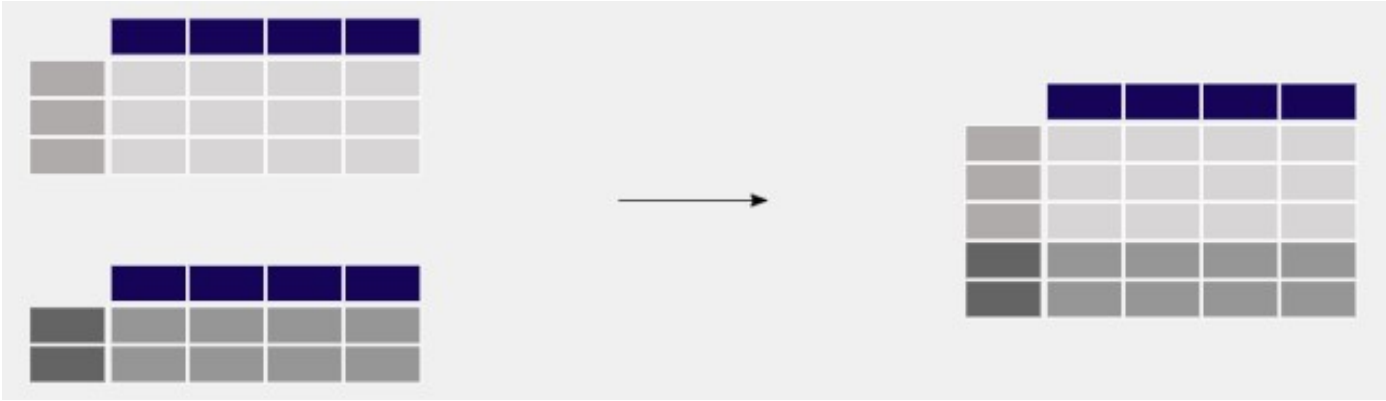
- Create new columns derived from existing columns
  (There is no need to loop over all rows of your data table to do calculations. Data manipulations on a column work elementwise.)

- Summary Statistics (Basic statistics (mean, median, min, max, counts) are easily calculable. These or custom aggregations can be applied on the entire data set, a sliding window of the data or grouped by categories)



Pandas

- Combine data from multiple tables (Multiple tables can be concatenated both column wise as row wise and database-like join/merge operations are provided to combine multiple tables of data)



- Handle time series data (extensive set of tools for working with dates, times, and time-indexed data)
- Manipulate textual data (Data sets do not only contain numerical data. pandas provides a wide range of functions to clean textual data and extract useful information from it)

SciKit-Learn

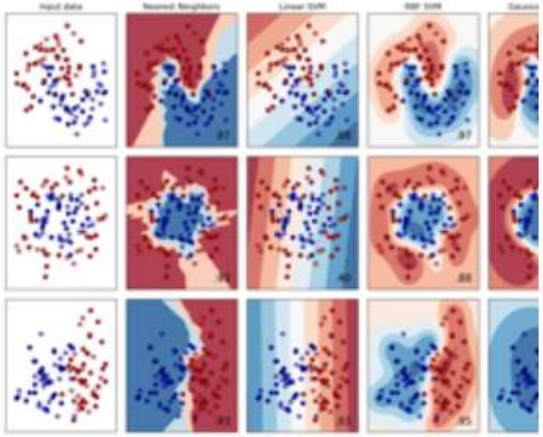
- Provides machine learning algorithms: classification, regression, clustering, model validation, preprocessing, etc.
- Built on NumPy, SciPy and Matplotlib
- Simple and efficient tools for predictive data analysis
- Open source, commercially usable - BSD license
- Most popular Python machine learning library for developing machine learning algorithms?
- Wide range of supervised and unsupervised learning algorithms that works on a consistent interface in Python
- Data-mining and data analysis
- Classification, regression, clustering, dimensionality reduction, model selection, and preprocessing
- https://scikit-learn.org/

SciKit-Learn

Classification

Identifying which category an object belongs to.

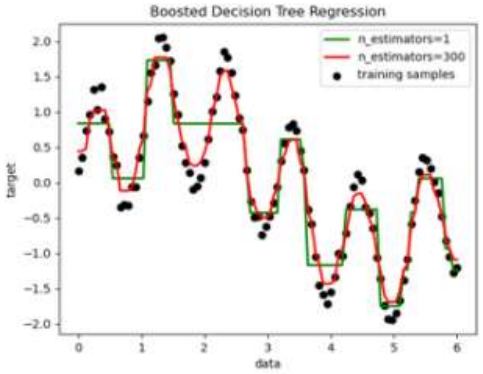
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

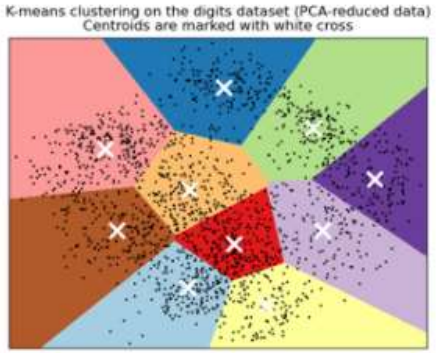
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

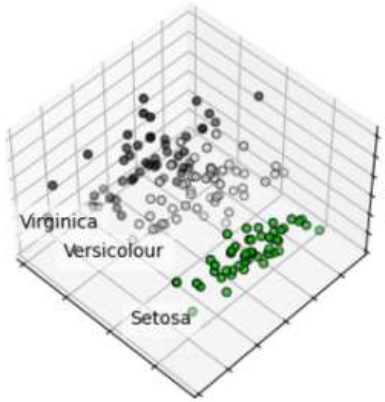


SciKit-Learn

Dimensionality reduction

Reducing the number of random variables to consider.

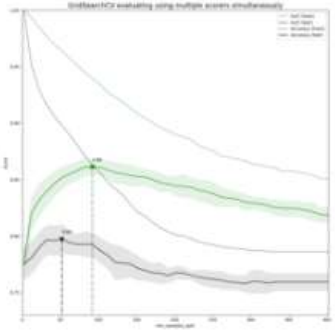
Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



Model selection

Comparing, validating and choosing parameters and models.

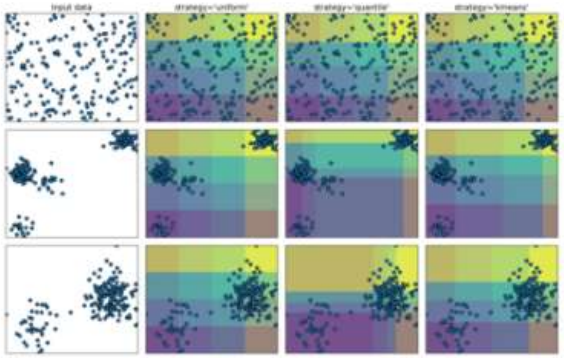
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...



Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



MapPlotLib

- Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python
- A set of functionalities similar to those of MATLAB
- Line plots, scatter plots, barcharts, histograms, pie charts etc.
- Relatively low-level; some effort needed to create advanced visualization
- https://matplotlib.org

matplotlib

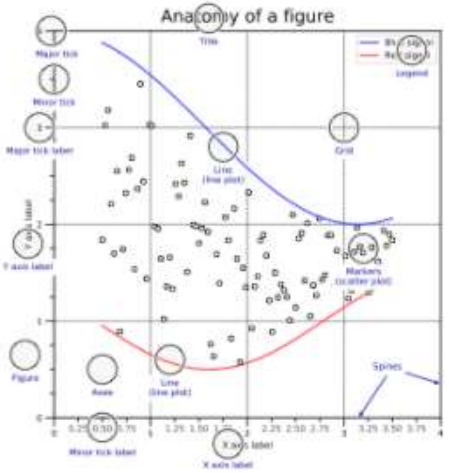
Cheat sheet

Version 3.2

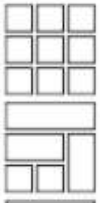
Quick start

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt


X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X,Y,color='C1')

fig.savefig("figure.pdf")
fig.show()
```

Anatomy of a figure



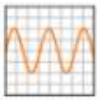
Subplots layout



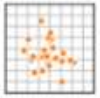
`subplot[s](cols,rows,…)`
`fig, axs = plt.subplots(3,3)`

`G = gridspec(cols,rows,…)`
`ax = G[0,:]`

Basic plots



`plot([X],Y,[fmt],…)`
X, **Y**, fmt, color, marker, linestyle



`scatter(X,Y,…)`
X, **Y**, [s]izes, [c]olors, markers, cmap

`bar[h](x,height,…)`
x, **height**, width, bottom, align, color

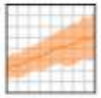
`imshow(Z,[cmap],…)`
Z, cmap, interpolation, extent, origin

`contour[f]([X],[Y],Z,…)`
X, Y, **Z**, levels, colors, extent, origin

`quiver([X],[Y],U,V,…)`
X, Y, **U**, **V**, C, units, angles

`pie(X,[explode],…)`
Z, explode, labels, colors, radius

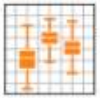
`text(x,y,text,…)`
x, **y**, **text**, va, ha, size, weight, transform



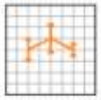
`fill[_between][x]( … )`
X, Y1, Y2, color, where

Advanced plots

`step(X,Y,[fmt],…)`
X, **Y**, fmt, color, marker, where



`boxplot(X,…)`
X, notch, sym, bootstrap, widths



`errorbar(X,Y,xerr,yerr,…)`
X, **Y**, xerr, yerr, fmt

`hist(X, bins, …)`
X, bins, range, density, weights

`violinplot(D,…)`

Scales

`ax.set_[xy]scale(scale,…)`

linear any values

log values > 0

symlog any values

logit 0 < values < 1

Projections

`subplot(…,projection=p)`

p='polar'

p='3d'

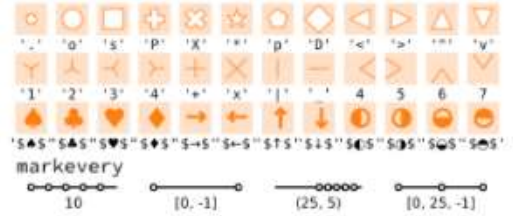
`p=Orthographic()`
from cartopy.crs import Cartographic

Lines

`linestyle or ls`

`capstyle or dash_capstyle`

Markers



Colors

Colormaps

`plt.get_cmap(name)`

Tick locators

```
from matplotlib import
ax.[xy]axis.set_[mino
```

`ticker.NullLocator()`

`ticker.MultipleLocato`

`ticker.FixedLocator([`

`ticker.LinearLocator(`

`ticker.IndexLocator(b`

`ticker.AutoLocator()`

`ticker.MaxNLocator(n=`

`ticker.LogLocator(bas`

Tick formatters

```
from matplotlib import
ax.[xy]axis.set_[mino
```

`ticker.NullFormatter()`

`ticker.FixedFormatter`

`ticker.FuncFormatter(`

`ticker.FormatStrForma`

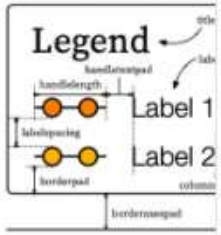
`ticker.ScalarFormatte`

`ticker.StrMethodForma`

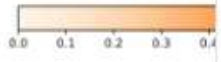
`ticker.PercentFormatt`

Ornaments

`ax.legend(…)`
handles, labels, loc,

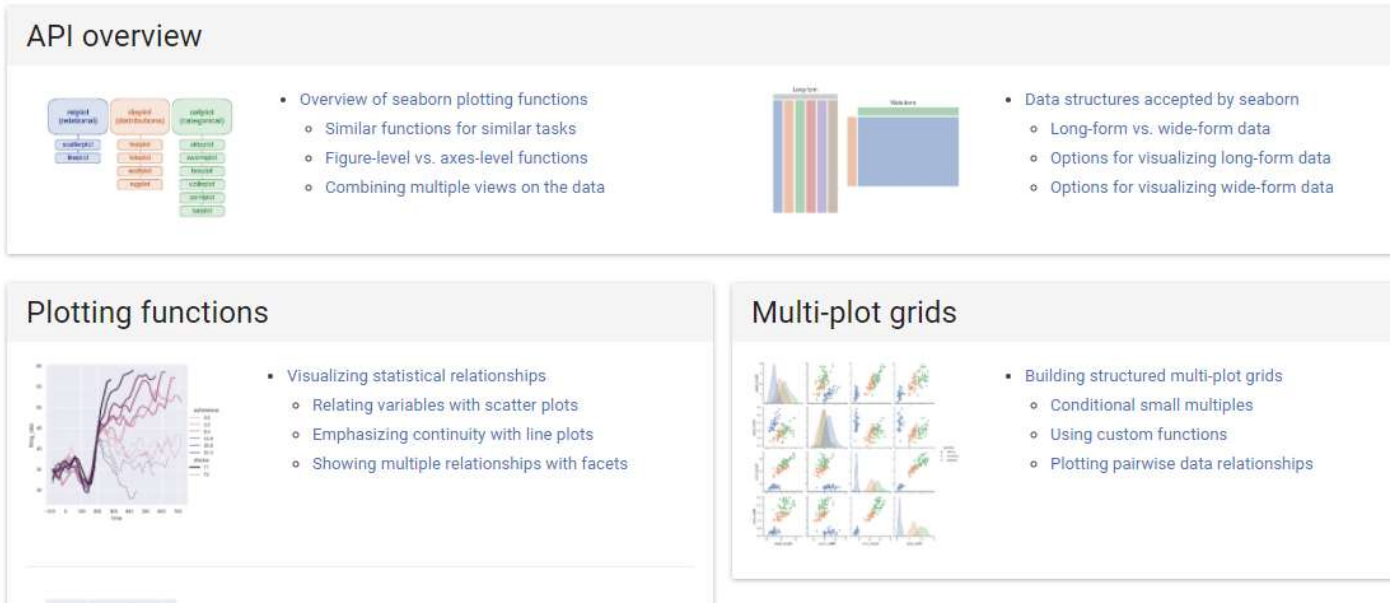


`ax.colorbar(…)`
mappable, ax, cax,

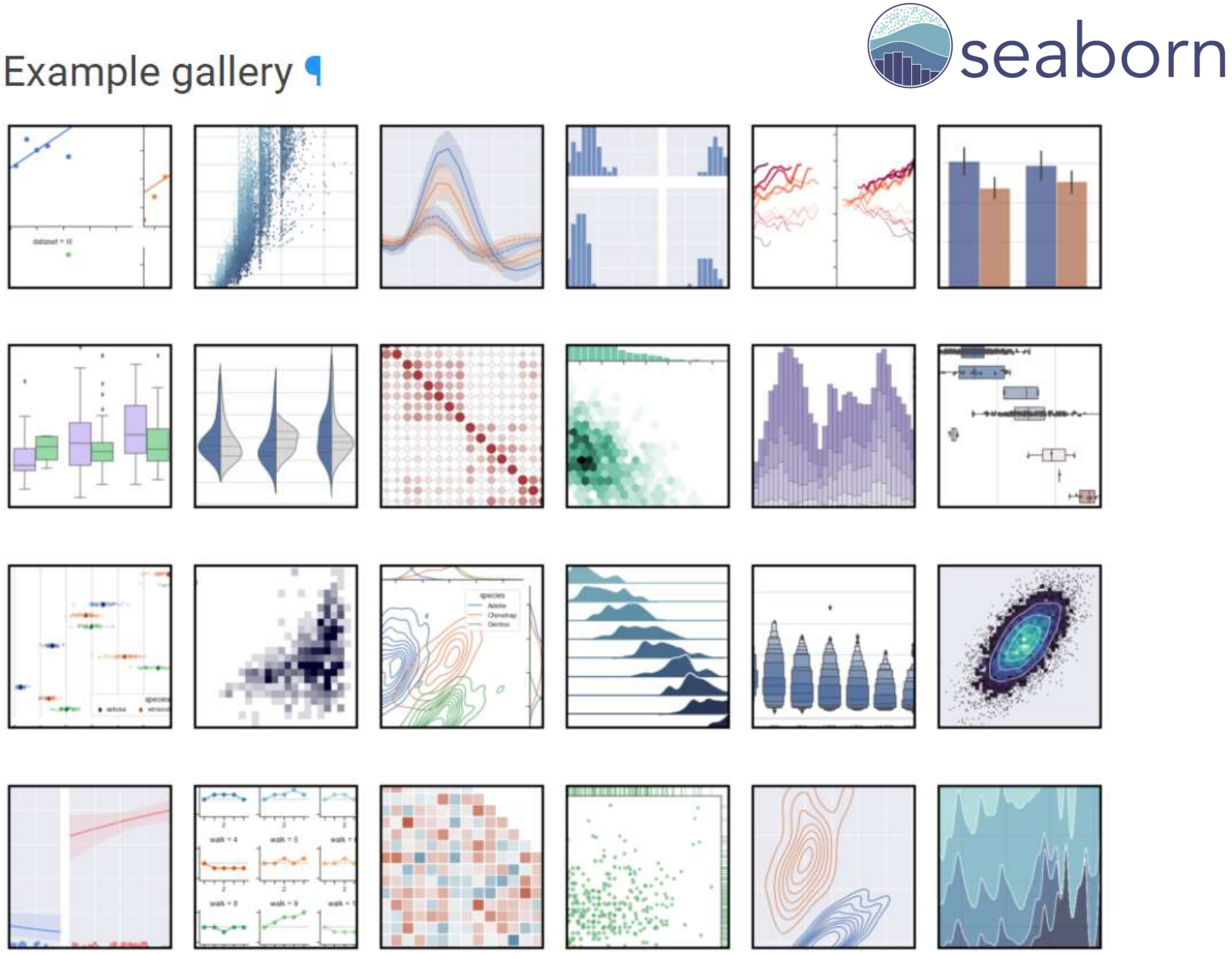


Seaborn

- Based on matplotlib
- Provides high level interface for drawing attractive and informative statistical graphics
- Similar (in style) to the popular ggplot2 library in R
- https://seaborn.pydata.org/



Seaborn



Conclusions

- Overview of the Machine Learning/Data Science Tools for the AI Course
- Python, Anaconda, Jupyter Notebook, JupyterLab, NumPy, SciPy, Pandas, Scikit-Learn, MatPlotLib, Seaborn
- Large Number of Other Powerful Machine Learning Libraries and Tools: PyTorch, Theano, TensorFlow, Keras, etc.

References

- Bruno Santos, https://towardsdatascience.com/an-introduction-to-pandas-in-python-b06d2dd51aba
- DataCamp, https://campus.datacamp.com/courses/data-science-for-everyone/
- Katia Oleinik, Python for Data Analysis, Boston University
- Python Website, https://www.python.org/
- Anaconda Website, https://www.anaconda.com/
- Project Jupyter Website, https://jupyter.org/
- NumPy Website, https://numpy.org/
- SciPy Website, https://www.scipy.org/
- Pandas Website, https://pandas.pydata.org/
- Scikit-Learn Website, https://scikit-learn.org/
- MatPlotLib Website, https://matplotlib.org
- Seaborn Website, https://seaborn.pydata.org/

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence
Lecture 5b: Machine Learning Tools

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence