DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Introduction to Python and Scikit-Learn

Machine Learning 2021
Slides P. Zanuttigh
Material from: M. Huenerfauth, G. van Rossum, R.P. Muller, P. Dragone, A. Passerini

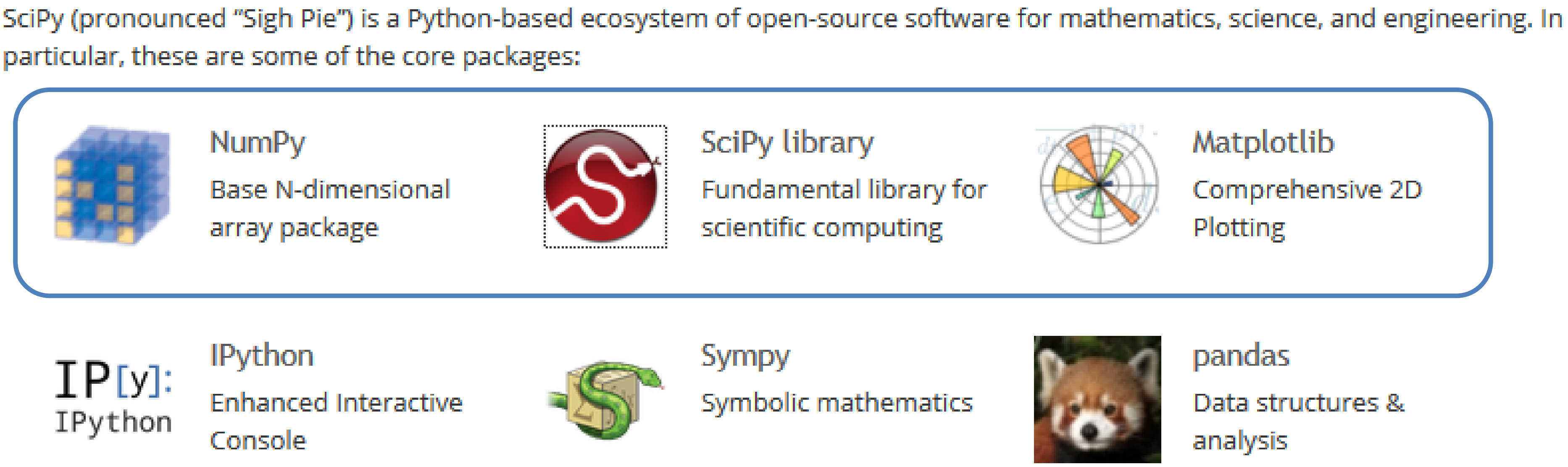
Python

- Interpreted high-level general-purpose programming language
- It is open source !
- Object Oriented programming model
- Current version is 3.9
  - There are relevant changes from Python 2.x to 3.x
  - For this course we'll use Python 3.x

Resources:

- Website: http://www.python.org
- Documentation: http://www.python.org/doc/

Modules: SciPy ecosystem



SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:

NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

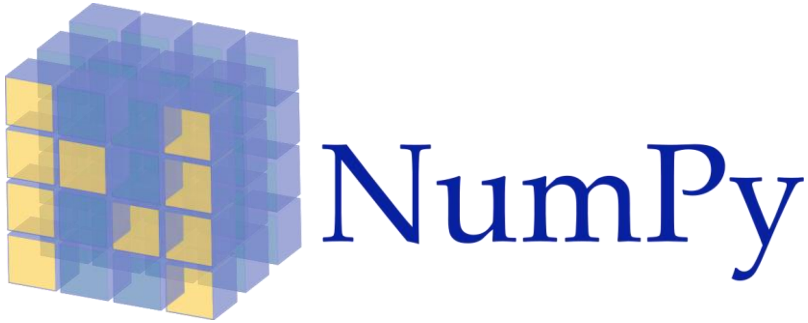
Matplotlib
Comprehensive 2D Plotting

IPython
Enhanced Interactive Console

Sympy
Symbolic mathematics

pandas
Data structures & analysis

Modules: NumPy



- Scientific computation capabilities within Python
  - Similar to Matlab functionalities
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra, etc…

Resources:

- Downloads: http://numpy.scipy.org/
- Tutorial: http://www.scipy.org/

Modules: scikit-learn

- Machine Learning library in Python
- Simple and efficient tools for data mining and data analysis
- Based on numpy and scipy
- Open source
- We'll use this library for the labs !!
- Documentation: http://scikit-learn.org/stable/documentation.html
- Reference Manual: http://scikit-learn.org/stable/modules/classes.html

scikit-learn: What's inside

1. Supervised learning
    1.1. Generalized Linear Models
    1.2. Linear and Quadratic Discriminant Analysis
    1.3. Kernel ridge regression
    1.4. Support Vector Machines
    1.5. Stochastic Gradient Descent
    1.6. Nearest Neighbors
    1.7. Gaussian Processes
    1.8. Cross decomposition
    1.9. Naive Bayes
    1.10. Decision Trees
    1.11. Ensemble methods
    1.12. Multiclass and multilabel algorithms
    1.13. Feature selection
    1.14. Semi-Supervised
    1.15. Isotonic regression
    1.16. Probability calibration
    1.17. Neural network models (supervised)

2. Unsupervised learning
    2.1. Gaussian mixture models
    2.2. Manifold learning
    2.3. Clustering
    2.4. Biclustering
    2.5. Decomposing signals in components
    2.6. Covariance estimation
    2.7. Novelty and Outlier Detection
    2.8. Density Estimation
    2.9. Neural network models (unsupervised)

3. Model selection and evaluation
4. Dataset transformations
5. Dataset loading utilities
6. Computing with scikit-learn

Setup: Your Home PC or Laptop

For your PC:

- □ Install ***Anaconda*** (with Python 3)
- □ Install ***scikit-learn*** (if not already installed by Anaconda)
  - Install scikit-learn with anaconda: `conda install scikit-learn`
    - or install with pip: `pip install -U scikit-learn`
  - It requires: Python (>= 3.4), NumPy (>= 1.8.2), SciPy (>= 0.13.3)
  - If required install the dependencies with pip or conda
- □ Install ***jupyter notebook***
  - □ With anaconda it is installed by default
  - □ Can be launched with : `jupyter notebook` or `jupyter-lab`

Setup Labs PCs

- Start the computer under linux
- To login you can use your DEI account or the temporary account provided by the instructor if you do not have a DEI account
- Setup Anaconda 3 environment with Python 3:

```
source /nfsd/opt/anaconda352/anaconda352.sh
```

- Launch jupyter notebook or lab

```
jupyter notebook   or   jupyter-lab
```

Tutorials

Useful resources to learn the basics of Python programming:

❑ See the provided *python_intro_labs* script

❑ Look at http://cs231n.github.io/python-numpy-tutorial/

❑ You can find a Jupyter notebook version of the tutorial at:
https://github.com/kuleshov/cs228-material/blob/master/tutorials/python/cs228-python-tutorial.ipynb

How to use:
1. Python Interpreter

- ❑ Interactive interface to Python (similar to matlab command window)

- ❑ Launch with the python command from the bash/command prompt

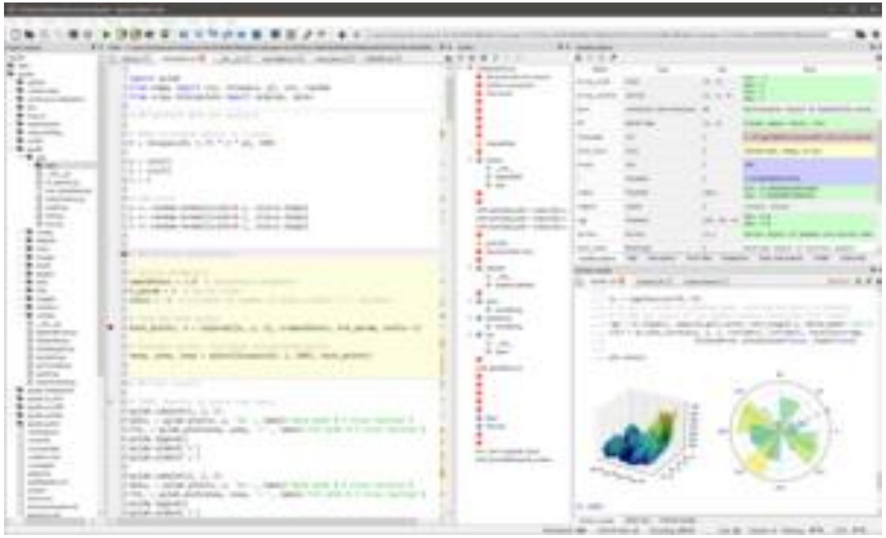
```
[python36] C:\Users\root>python
    Python 3.6.2 |Anaconda custom (64-bit)| (default, Jul 20 2017, 12:30:02)  [MSC v.1900 64
bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license" for more information.
    >>>
```

- ❑ Python interpreter evaluates inputs:

```
    >>> 3*(7+2)
    27
```

- ❑ Python prompts with '>>>'.
- ❑ To exit Python: `exit()`

How to use:
2. Write Source and Run



- ❑ Write your source code and save in a .py file
- ❑ You can use any editor or IDE of your choice
  - ○ e.g., PyCharm or Visual Studio Code
- ❑ Anaconda also provides the spyder environment that has some debugging tools
- ❑ Run the file:

```
python filename.py
```

How to use:
3. Jupyter notebook / lab

- ❑ Run with : `jupyter notebook` or `jupyter-lab`
    - ○ Jupyter lab has some extra features
- ❑ Interactive environment inside the web browser
- ❑ You can run each block of code and see the output
- ❑ Can combine code and text (comments / description)
- ❑ ***We'll use jupyter notebooks for the lab deliveries***

Basics:
Operators and Variables

□ Assignment uses = and comparison uses ==

□ For numbers: + - * / % are as expected

- Special use of + for string concatenation
- Special use of % for string formatting (as with *printf* in C)
- Logical operators are words (and, or, not) *not* symbols

□ The basic printing command is print

□ The first assignment to a variable creates it

□ Variable types don't need to be declared

□ Python figures out the variable types on its own

Basic Datatypes

Integers

x = 3 (x is an int)

z = 5 / 2   # Answer is 2.5 in Python 3 and 2 in Python 2 !!

Floats

x = 3.456 (x is a float)

Strings

Can use “ ” or ‘ ’ to specify : “abc” ‘abc’ are the same thing

Whitespaces

- Whitespace is meaningful in Python
  - especially indentation and placement of newlines
- Use a newline to end a line of code
- No braces { } to mark blocks of code in Python
  - … use consistent indentation instead !
  - The first line with more indentation starts a nested block
  - The first line with less indentation is outside of the block
- Often a colon (:) appears at the start of a new block
  - E.g., for function and class definitions
- Start comments with # – the rest of line is ignored

Assignments

❑ Binding a variable in Python means setting a name to hold a reference to some object

❑ Assignment creates references, not copies

❑ Names in Python do not have an intrinsic type

- Objects have types !
- Python determines the type of the reference automatically based on the data object assigned to it

❑ You create a name the first time it appears on the left side of an assignment expression: (e.g., x = 3)

❑ A reference is deleted via garbage collection after any names bound to it have passed out of scope

Arrays

- Handled through the numpy library
- A numpy array is a grid of values, all of the same type
- It is indexed by a tuple of non-negative integers
- The *shape* of an array is a tuple of integers giving the size of the array along each dimension

Examples:

```
import numpy as np
a = np.array([1, 2, 3])   # Create a rank 1 array
print(type(a))            # Prints "<class 'numpy.ndarray'>"
print(a.shape)            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array
print(b.shape)                     # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"
```

Sequence Types

1. Tuple

- A simple immutable ordered sequence of items
- Items can be of mixed types, including collection types

2. Strings

- Immutable
- Conceptually very much like a tuple

3. List

- Mutable ordered sequence of items of mixed types

4. (*Dictionaries*)

- Store a mapping between a set of keys and a set of values

Functions

Functions:

- *def* creates a function and assigns it a name
- *return* sends a result back to the caller
- Arguments are passed by assignment
- Arguments and return types are not declared

Examples:

```
def <name>(arg1, arg2, ..., argN):
   <statements>
   return <value>

def times(x,y):
   return x*y
```

Lab 0: Your First Program in Python

Develop a simple application in the last part of the lab:

1. Load the provided .csv file with the used car data
2. Use a linear regression to estimate the car prices from the year, kilometers or engine power
   - You can make a simple 1D regression from each one of the parameters independently
   - *(optional) If you like to experiment try a 2D or 3D regression combining multiple cues*
3. Firstly use the scipy *linregress* function
   - Alternatively you can use the sklearn.linear_model.LinearRegression class
4. Have a look at the correlation coefficient to see which of the 3 features works better
5. *(optional)* try to manually implement the least square algorithm
   - You should get exactly the same solution of *linregress* !
   - If never used least squares you can do it later after the lectures on linear models
6. Plot the data and the lines representing the output of the *linregress* and least square algorithms

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Linear Regression with scikit-learn

scipy.stats.linregress

- The function calculates a linear least-squares regression for two sets of measurements
- scipy.stats.linregress(x, y=None)[source]

Parameters:

- x, y : array_like Two sets of measurements. Both arrays should have the same length. If only x is given (and y=None), then it must be a two-dimensional array where one dimension has length 2. The two sets of measurements are then found by splitting the array along the length-2 dimension

Returns:

- slope : float slope of the regression line
- intercept : float intercept of the regression line
- rvalue : *float* *correlation coefficient (see box,* ± 1 *: total correlation, 0 no correlation)*
- pvalue : float two-sided p-value for a hypothesis test whose null hypothesis is that the slope is zero, using Wald Test with t-distribution of the test statistic
- stderr : float Standard error of the estimated gradient

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Least Squares (optional, presented later)

- Compute gradient of MSE on training set and set to 0

$$L_S = \frac{1}{m}\sum_{i=1}^{m}(<\boldsymbol{w},\boldsymbol{x}_i> -y_i)^2 \rightarrow \frac{\partial L_S}{\partial \boldsymbol{w}} = \frac{2}{m}\sum_{i=1}^{m}(<\boldsymbol{w},\boldsymbol{x}_i> -y_i)\boldsymbol{x}_i = 0$$

- Set

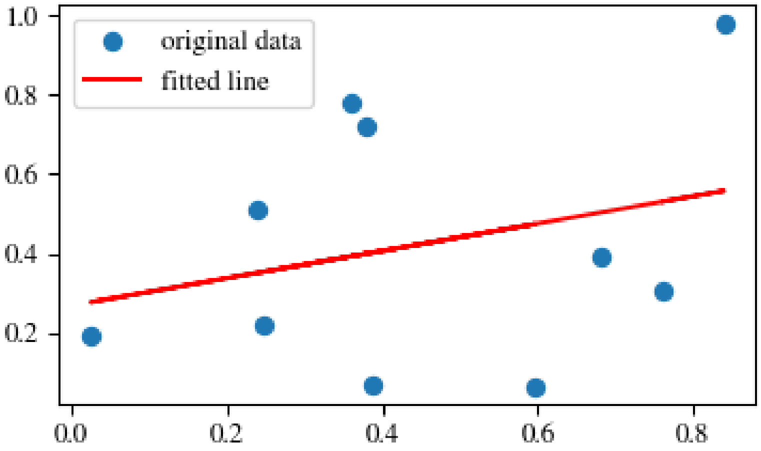
$$A = \left(\sum_{i=1}^{m}\boldsymbol{x}_i\boldsymbol{x}_i^T\right) \quad \boldsymbol{b} = \sum_{i=1}^{m} y_i\boldsymbol{x}_i$$

- The solution is:

$$\boxed{\boldsymbol{w} = A^{-1}\boldsymbol{b}}$$

- *w[0]: intercept* *w[1]: slope*
- The computation is done using homogeneous coordinates
- Python: 1D array and *m x 1* 2D array are different objects
- Inverse of a matrix: np.linalg.inv(M)

Plot Data with matplotlib



Plot the data along with the fitted line using matplotlib

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(x, y, 'o', label='original data')
>>> plt.plot(x, intercept + slope*x, 'r', label='fitted line')
>>> plt.legend()
>>> plt.show()
```

Task for Lab 0

1. Load a dataset with used car data
2. Use a linear regression to estimate the car prices from the year, kilometers or engine power
3. Understand which of the 3 features works better and visualize your results

For lab 0 there is no homework, it is just to get used with Python

For help ask to the instructor or to the TA

TA: F. Barbato, U. Michieli, G. Rizzoli, D. Shenaj