{Python Cheat Sheet}

Machine Learning in Bioengineering | IST

by Ana Sofia Carmo github.com/anascacais in /in/anasofiacarmo (via LaTeX template)

Python Basics

- String operations:

You can join strings by using a sum operator.

```
>>> aux = 2
>>> print(f'I have {aux} dogs')
'I have 2 dogs'
```

>>> s.replace(a,b)	Replace every occurrence of character a with b.
>>> s.split('chr')	Splits string by the chosen chr and returns a list with the elements.

- List operations:

You can join lists by using a sum operator.

You can use list indexing by (1) selecting a single index, (2) providing a *start*, *end* and *step* or (3) an iteration of this, e.g.:

```
>>> list[start:end:step]
>>> list[start::] ≡ list[start:len(list):1]
>>> list[::step] ≡ list[0:len(list):step]
```

>>> l.index(item)	Get the index of an item.
>>> l.append(item)	Append item to the end.
>>> l.remove(item)	Remove first instance of item.
>>> l.pop(ind)	Remove and print item in index ind.
>>> l.sort()	Sort list.
>>> l.insert(ind,item)	Insert item in index ind.

- Dictionaries:

```
>>> dict = {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
>>> dict['brand']
'Ford'
```

>>> d.update(d2)	Add key-value pairs from d2 to d1 and overwrite the ones in common.
>>> d.get(key)	Get corresponding value if exists.
>>> d.keys()	Get list of keys.
>>> d.values()	Get list of values.
>>> d.items()	Get list of tuples with key and value (key, value).

Python Basics

- List comprehension:

```
>>> aux = [1, 2, 3, 4]
>>> [item for item in aux]
[1, 2, 3, 4]
>>> [item for item in aux if (item % 2) == 0]
[2, 4]
>>> {str(item): item for item in aux}
{'1': 1, '2': 2, '3': 3, '4': 4}
```

- File handling:

>>> f = open(path)	Open file for reading.
>>> f = open(path, 'w')	Open file for writing.
>>> f.close()	Close file.

```
>>> with open(path) as f:
        # perform file operations
```

- File operations:

>>> f.write(str)	Writes a string to a file (to add a paragraph add '\n').
>>> f.readlines()	Returns all lines as elements of a list.
>>> f.readline()	Returns a single line (read file line-by-line).

- json module:

JSON is a lightweight format for saving data in a human-readable format.

```
>>> import json
```

>>> json.dump(dict, f)	Save dict to JSON file.
>>> json.load(f)	Reads JSON file into a dict.

- pickle module:

Pickle serializes objects so they can be saved to a file, not restricted by a specific format.

```
>>> import pickle
```

>>> pickle.dump(obj, f)	Save object to file.
>>> pickle.load(f)	Read file.

Python Cool Features

- os module:

Provides functions to interact with the corresponding operating system.

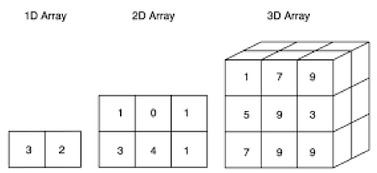
```
>>> import os
>>> os.path.join('C:/path/to', 'file.txt')
'C:/path/to/file.txt'
>>> os.path.basename('C:/path/to/file.txt')
'file.txt'
```

>>> os.listdir(dir)	List files and folders in dir (if dir is not provided, it defaults to the cwd).

- numpy module:

Provides powerful data structures, implementing multi-dimensional arrays and matrices.

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




```
>>> a1 = np.array([1,2,3])
>>> a1.shape
(3,)
>>> len(a1)
3
>>> a2 = np.array([[1,2,3], [4,5,6]])
>>> a2.shape
(2,3)
>>> len(a2)
2
```

>>> np.zeros((d1,d2))	Creates array of zeros with the given dimensions.
>>> np.ones((d1,d2))	Creates array of ones with the given dimensions.
>>> np.arange(a,b,c)	Creates an array of evenly spaced values, from a to b, with step value c.
>>> np.linspace(a,b,c)	Creates an array of evenly spaced values, from a to b, with c elements.

Python Cool Features

- pandas module:

Intuitive, table-like data structure. Can receive as input ndarray, Iterable (e.g. list), dict, or DataFrame.

```
>>> import pandas as pd
>>> df = pd.DataFrame(data, columns=['col1', 'col2',...])
```

I/O:

>>> pd.read_csv(path)	Reads file into a DataFrame.
>>> pd.to_csv(path)	Saves DataFrame to file; path must end in .csv.

DataFrame operations:

>>> df.shape	(rows, columns)
>>> df.columns	Returns list with names of cols.
>>> df.index	Returns list with indexes.
>>> df.drop(['col'], axis=1)	Remove col with name col.
>>> df.drop([0,1], axis=0)	Remove rows with indexes 0 and 1.

Data selection & examples:

```
>>> df.loc[<row selection>, <col selection>]
```

Can receive as input: single label, list/array of labels, slice object with labels or boolean / logical indexing.

```
>>> df.iloc[<row selection>, <col selection>]
```

Can receive as input: int, list/array, slice object with ints or boolean array.

>>> df['col']	Selects column with name col.
>>> df.values	Returns a numpy representation.
>>> df.loc['indx', 'col']	Selects value of row with index indx and column col.
>>> df.loc[df['col'] == val]	Selects rows whose column value equals val.
>>> df.iloc[i]	Selects ith row of df.
>>> df.iloc[:,i]	Selects ith col of df.

{Python Cheat Sheet}

Machine Learning in Bioengineering | IST

by Ana Sofia Carmo github.com/anascacais in /in/anasofiacarmo (via LaTeX template)

Python Visualization

- Standard Plots:

Comprehensive library for creating static, animated, and interactive visualizations in Python.

```
>>> import matplotlib.pyplot as plt

>>> plt.figure(figsize=(width,height))
```
If figsize is not provided, defaults to (6.4,4.8).

```
>>> plt.plot(x, y)
>>> plt.show()
```

The plot() function plots y versus x as lines and/or markers. But there are several other styles of plots:

- scatter
- histogram
- barchart

- Figure with Multiple Plots:

Subplot creates a grid with n=(rows*cols) plots.

```
>>> plt.figure(figsize=(width,height))
    for i in np.arange(n):
        plt.subplot(rows,cols,i)
        plt.plot(x, y)
```

- Customizing a Plot:

Attributes of plot():

linewidth or lw	Set width of line.
alpha	Set transparency of line.
linestyle or ls	'-', '--', '-.', ':', ...
marker	Set style of data points.
color	Set color of plots.

Additional customization:

>>> plt.title(str)	Set title of plot.
>>> plt.suptitle(str)	Set title of the set of figures when using subplot.
>>> plt.xlabel(str)	Set label of x axis.
>>> plt.ylabel(str)	Set label of y axis.
>>> plt.xlim((a,b))	Set limits of x axis.
>>> plt.ylim((a,b))	Set limits of y axis.
>>> plt.legend([str,...])	For multiple lines in the same plot, set legend.

Python Visualization

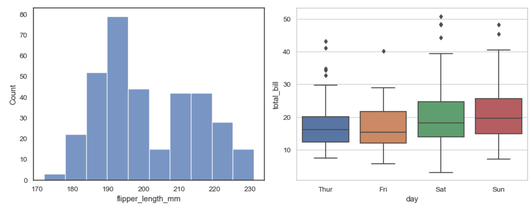
- seaborn module:

Based on matplotlib, provides attractive and informative statistical graphics.

```
>>> import seaborn as sns
```

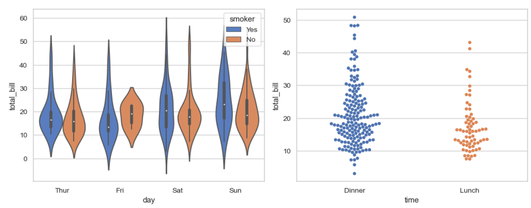
Here are some useful examples:

(a) sns.histplot: Plot univariate or bivariate histograms to show distributions of datasets.
(b) sns.boxplot: Draw a box and whiskers plot to show distributions with respect to categories.



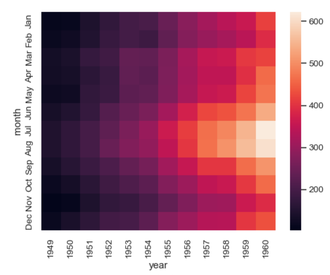
(a) (b)

(c) sns.violinplot: Draw a combination of boxplot and kernel density estimate.
(d) sns.swarmplot: Draw a categorical scatterplot with non-overlapping points.



(c) (d)

(e) sns.heatmap: Plot rectangular data as a color-encoded matrix.



(e)