Python: Fundamentals for Exploratory Data Analysis

Eduardo Destefani Stefanato[1]*
Vitor Souza Premoli Pinto de Oliveira[1]*

[1]Universidade Federal do Espírito Santo

Artificial Intelligence Applied to Images

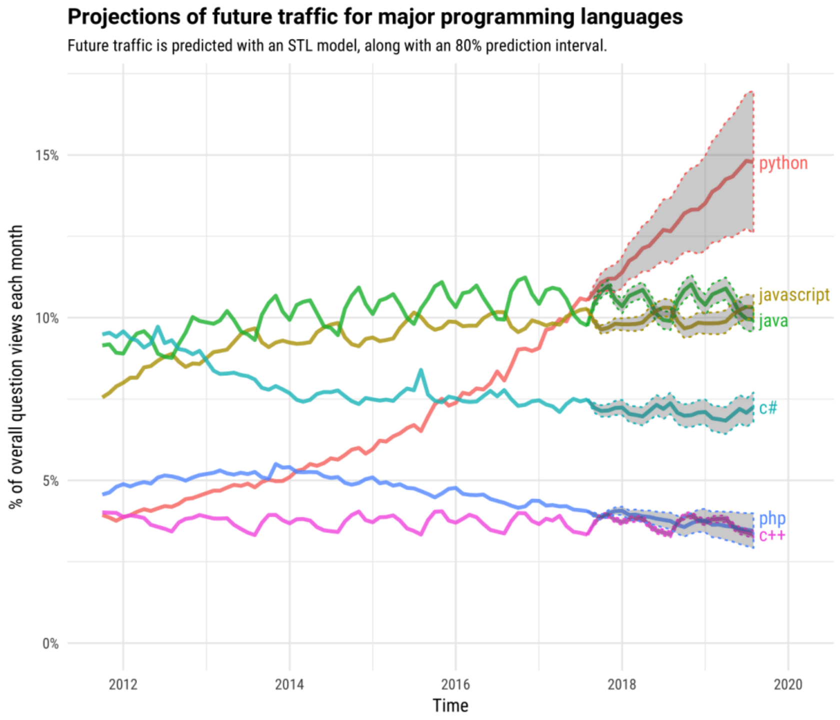
Sumary

INTRODUCTION

Python Language

What’s Python?

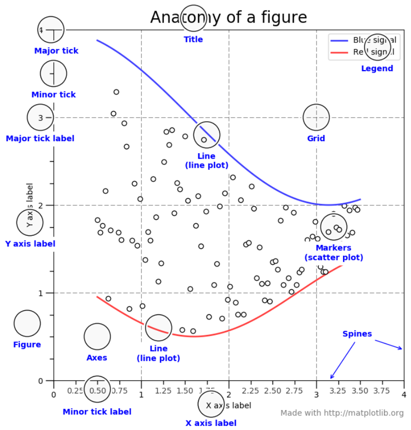
Python is a high-level, interpreted, scripting, interactive, object-oriented, functional, dynamically typed, strong programming language. It was released by Guido van Rossum in 1991, but only became more popular two decades later.

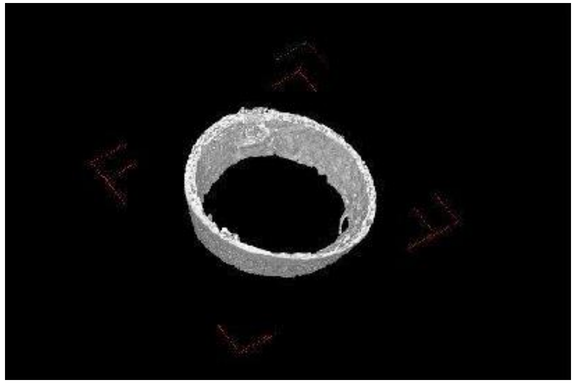


As an Open Source language, its use is widespread among researchers in various fields. Such a programming language is known for its simplicity and ease of learning, this conclusion indicates that the use of Python and its tools, especially **Numpy**, **Matplotlib** and **Pandas** are extremely advantageous for research in Physics.

The great of the Python language is its wealth of libraries, called modules, which can be official or community libraries. A Python module is a file with Python source code, data and function definitions, usually with a certain purpose. Python currently has hundreds of thousands of modules, mostly from the community, see the **PiPy** website, which as of 08/08/2021 listed **320,306** Python modules/packages.

Therefore, a Python program typically imports modules with the appropriate functionality to solve certain problems.





PYTHON LIBRARIES AND MODULES

Numpy

What's Numpy and why do we use it?

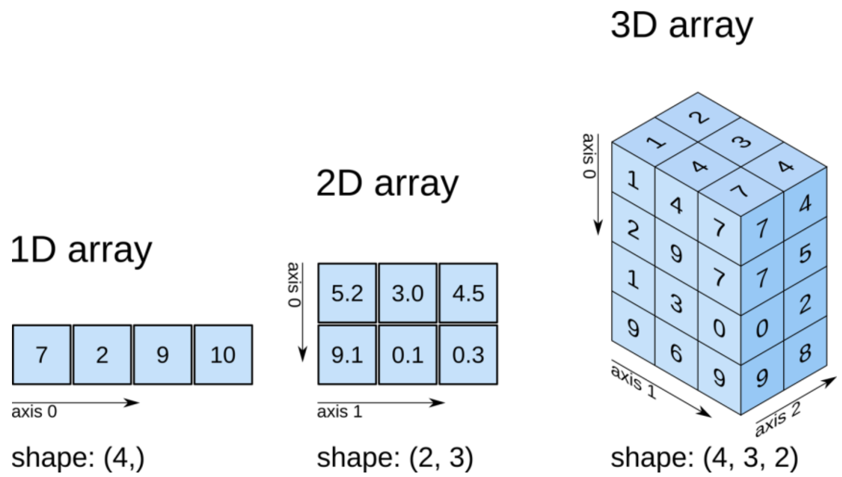
Numpy is a Python library that gathers several packages and functions targeted at multidimensional arrays. Numpy was developed for high-performance numerical computing, so it has optimizations in its source code to take advantage of the hardware's full potential. Numpy is also known as array-oriented computing.

What are arrays anyway?

- Arrays are basically a structure and organization of data aligned and divided into rows and columns in a one-, two-, three-, and n-dimensional way;

The data can be:

- Pixels of an image (grayscale or color);
- Discrete and/or continuous signals from a sensor, simulation, or experiment;
- 3D data for each axis, such as data from an MRI scan;
- And among others.

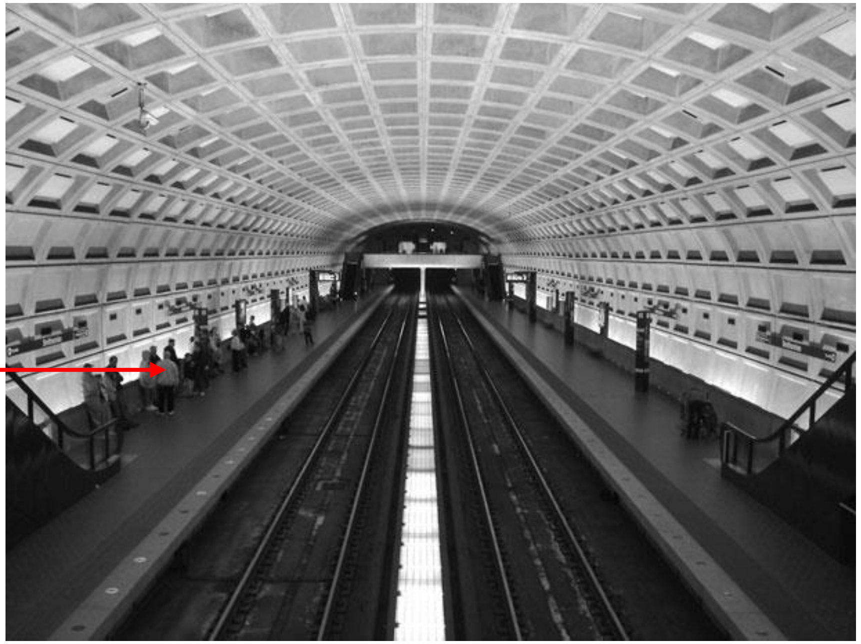


Array attributes and applications

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

```python
img = plt.imread('dc_metro.png'); print('Temos um %s: \n\n %s \n\ncom shape (dimensão) igual à %s, portanto uma matriz 2D.'
%(type(img), img, np.shape(img)))
```

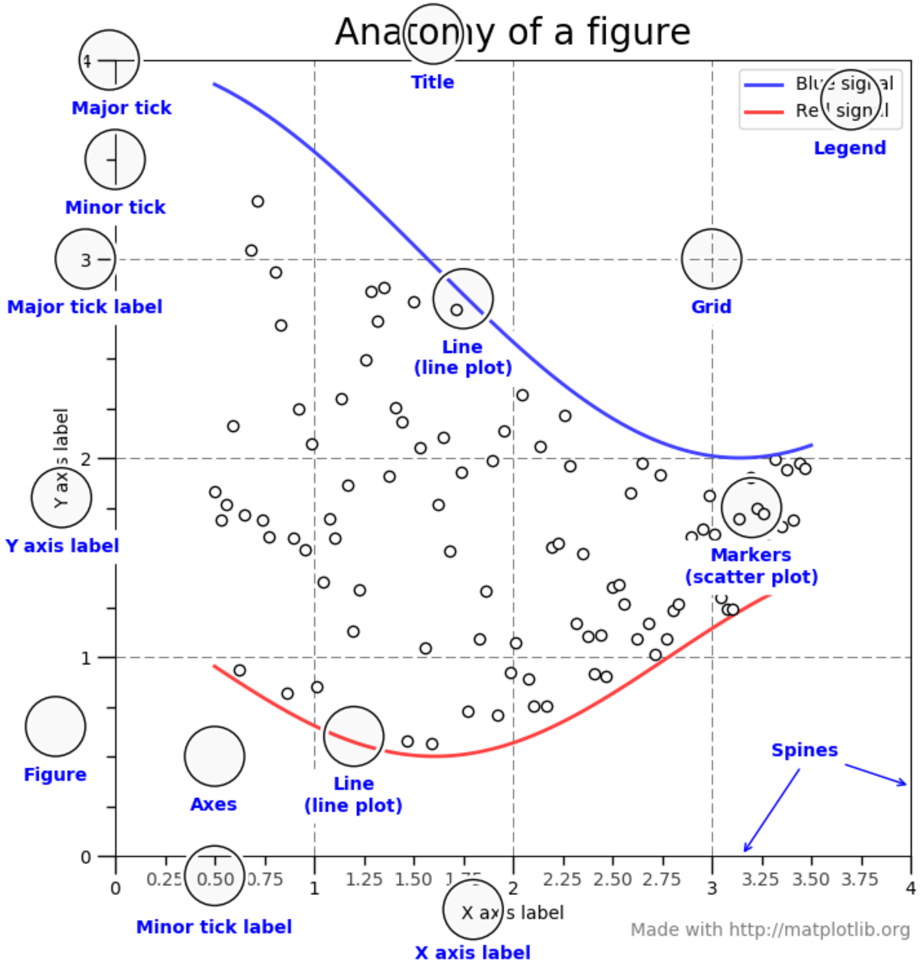
```
Temos um <class 'numpy.ndarray'>:

 [[0.6039216  0.6431373  0.6784314  ... 0.59607846 0.5921569  0.63529414]
 [0.6666667  0.654902   0.654902   ... 0.654902   0.64705884 0.6392157 ]
 [0.68235296 0.6784314  0.6666667  ... 0.61960787 0.57254905 0.5764706 ]
 ...
 [0.14509805 0.15294118 0.16078432 ... 0.1254902  0.1254902  0.1254902 ]
 [0.14117648 0.15686275 0.16470589 ... 0.12156863 0.1254902  0.12156863]
 [0.14117648 0.15294118 0.16470589 ... 0.12156863 0.1254902  0.12156863]]

com shape (dimensão) igual à (461, 615), portanto uma matriz 2D.
```



PYTHON LIBRARIES AND MODULES

Matplotlib

What’s Matplotlib and why do we use it?

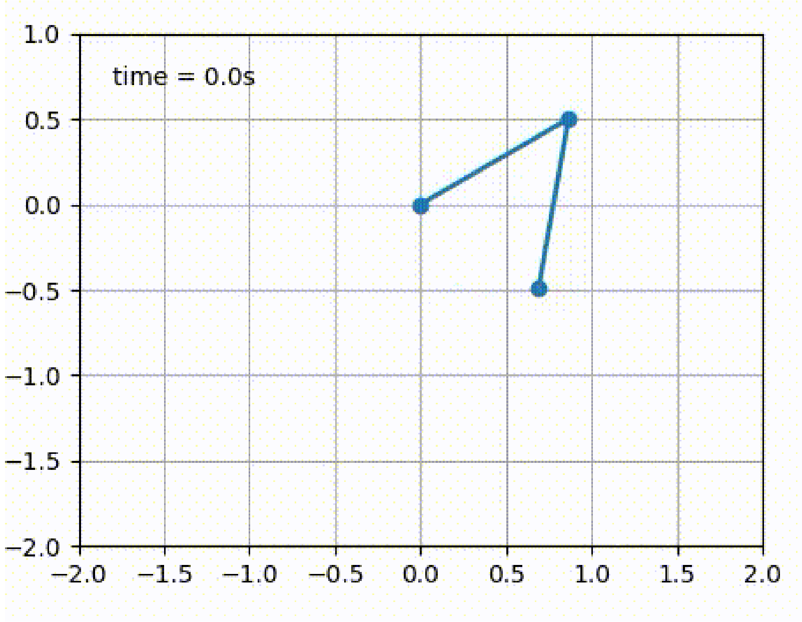


Matplotlib is a python library for creating graphs and data visualizations in general, targeted at its math extension Numpy.

Using Matplotlib, we can:

- Analyze images;
- Render images;
- Create simulations;
- Create interactive graphics;
- And among others.

Example animation, illustrating the problem of a double pendulum.



PYTHON LIBRARIES AND MODULES

Pandas

What's Pandas and why do we use it?

Pandas is a library that provides tools to read, write and manipulate data between structures in memory, in different formats: CSV and text files, Microsoft Excel, SQL databases and the fast HDF5 format.

It is very common that the data we analyze is arranged in a table format, like a neural network or any machine learning algorithms, especially when we talk about statistical data analysis.

This data in a table format can have, for example, columns containing different attributes of the data, and rows containing a set of observations.

DataFrame

To help us deal with table data, Pandas provides us with an object, called DataFrame, that is able to store and manipulate this type of data in a way equivalent to an Excel spreadsheet, where its rows and columns are called Series.

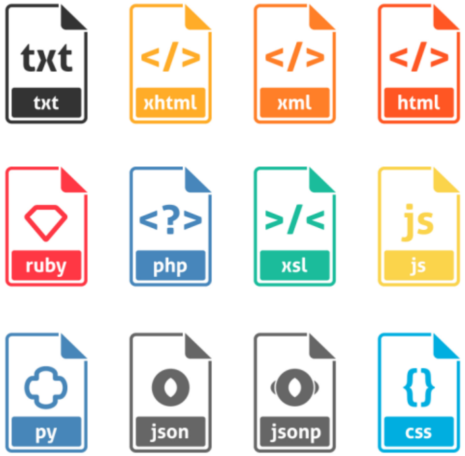
pandas

EXPLORATORY DATA ANALYSIS

Dataset extraction #1

What’s dataset and its purpose?

Dataset is a set of tabular data that is represented in a spreadsheet format where the rows are the records of events and the columns are the characteristics of these events.



Anderson's Iris data set

The data set used consists of 50 samples from each of the three Iris species (Iris setosa, Iris virginica and Iris versicolor). Four features were measured in each sample: the length and width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish species from each other. As a reference we have extracted from a .csv file available in the Kaggle repository.

Iris flower dataset

Iris data set

```
# library data sets
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# primal data
data = pd.read_csv('data/Iris.csv').copy(); data

data['Species'] = [i.split('-')[1] for i in data['Species']]; data = data.drop(columns=['Id']); data
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

The rows being the samples and the columns being: Sepal length (cm), Sepal width (cm), Petal length (cm) and Petal width (cm).

Basic statistics and variables correlation

```
# basic statistics
data.describe()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Shows the basic statistics of the dataset primitive, such as mean, standard deviation, and quartile.

```
# variables correlation
corr = data.corr(); corr
```

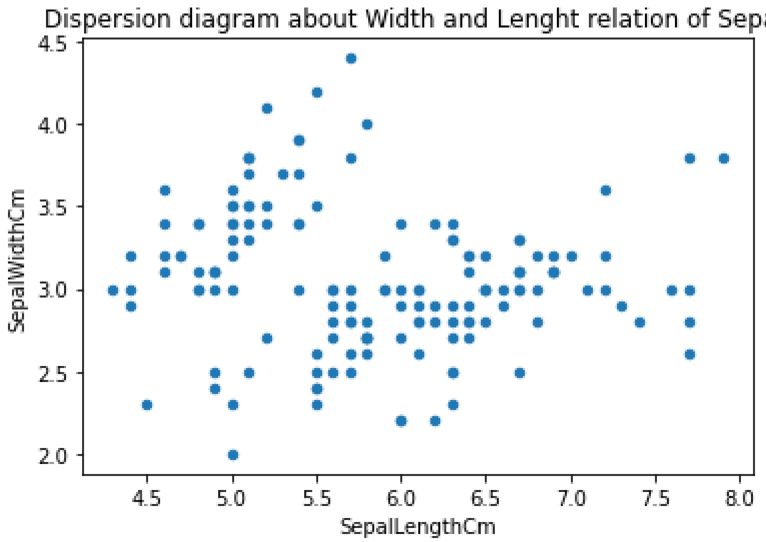
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

Correlation between the parameters of each species in the dataset.

plot data

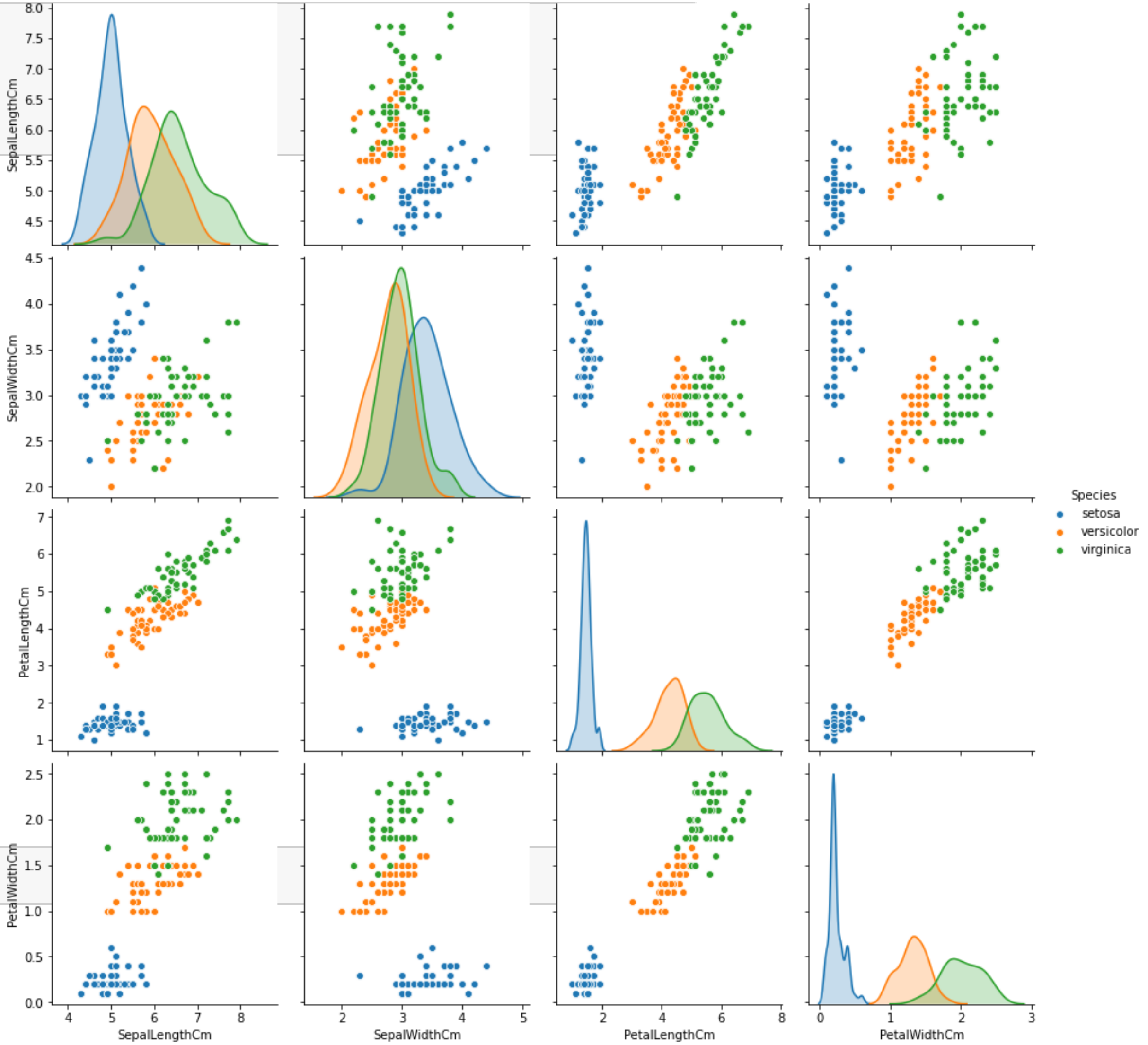
```python
#Gráfico de dispersão
data.plot(x="SepalLengthCm", y="SepalWidthCm", kind="scatter")
plt.title("Dispersion diagram about Width and Lenght relation of Sepal ")
```

```
Text(0.5, 1.0, 'Dispersion diagram about Width and Lenght relation of Sepal ')
```




```python
sns.pairplot(data, hue="Species", height=3)
```

```
<seaborn.axisgrid.PairGrid at 0x20ae075e760>
```

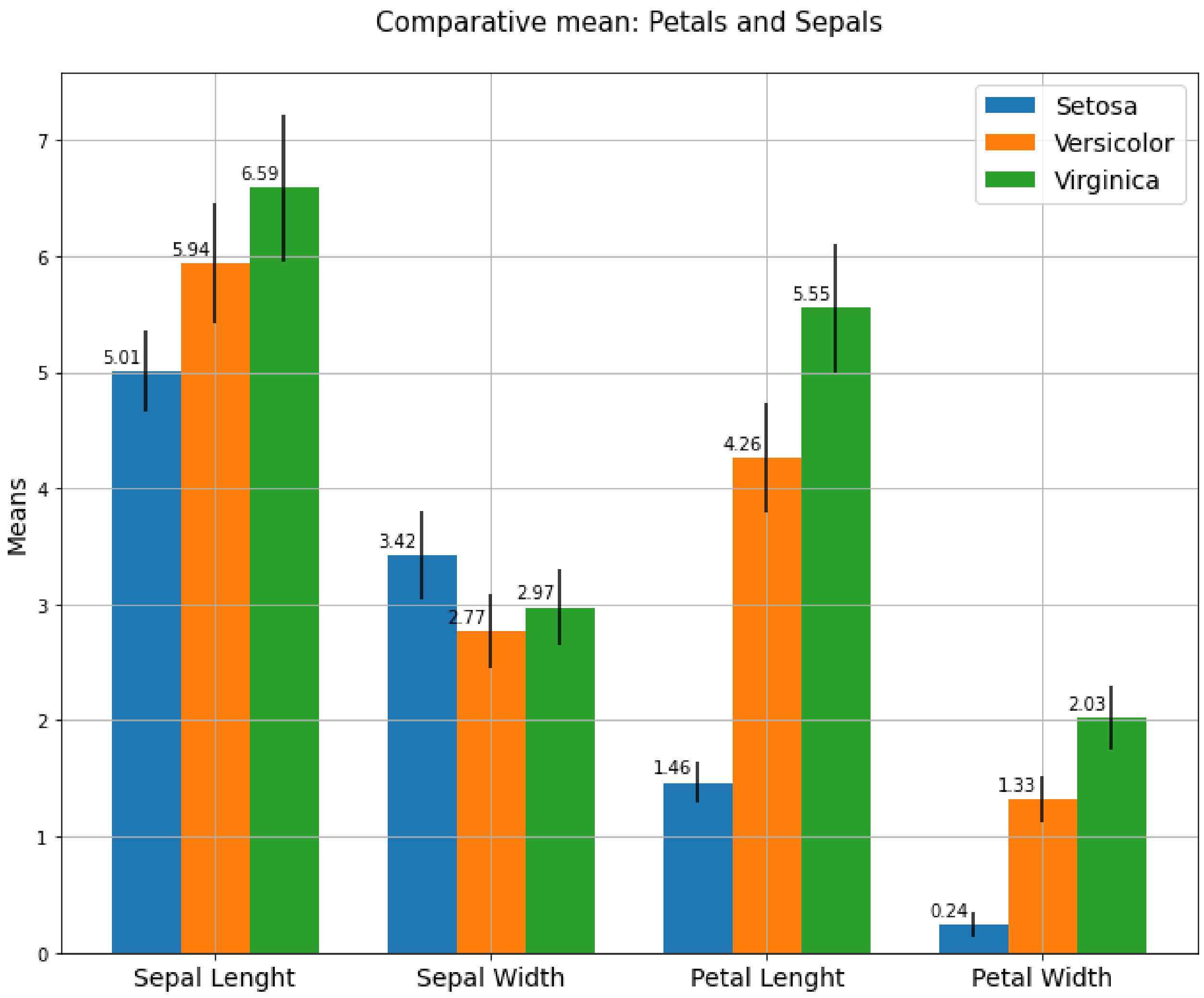


Aggregation and transformation

```
data_group = data.groupby(['Species'])[['SepalLengthCm',
                                        'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']].agg([np.mean, np.std]).reset_index(); data_group
```

	Species	SepalLengthCm		SepalWidthCm		PetalLengthCm		PetalWidthCm	
		mean	std	mean	std	mean	std	mean	std
0	setosa	5.006	0.352490	3.418	0.381024	1.464	0.173511	0.244	0.107210
1	versicolor	5.936	0.516171	2.770	0.313798	4.260	0.469911	1.326	0.197753
2	virginica	6.588	0.635880	2.974	0.322497	5.552	0.551895	2.026	0.274650

```
fig, ax = plt.subplots(1,2, figsize=(19,8))
ax[0].set_title('Comparative mean: Petals and Sepals\n', fontsize=15)

rects1 = ax[0].bar(x - 0.125 - width/2, se, width, label='Setosa',
                   yerr=se_err)
rects2 = ax[0].bar(x - 0.125  + width/2, ve, width, label='Versicolor',
                   yerr=ve_err)
rects3 = ax[0].bar(x - 0.125 + 1.5*width, vi, width, label='Virginica',
                   yerr=vi_err)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

ax[0].set_xticks(x)
ax[0].set_xticklabels(labels, fontsize=14)
ax[0].set_ylabel('Means', fontsize=14)
ax[0].legend(fontsize=14)
ax[0].grid(True)

ax[1].set_title('Correlatation Iris\n', fontsize=15)
sns.heatmap(corr, annot=True, ax=ax[1])
# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.xticks(fontsize=12, color='black')
plt.yticks(fontsize=12, color='black')

plt.show()
```

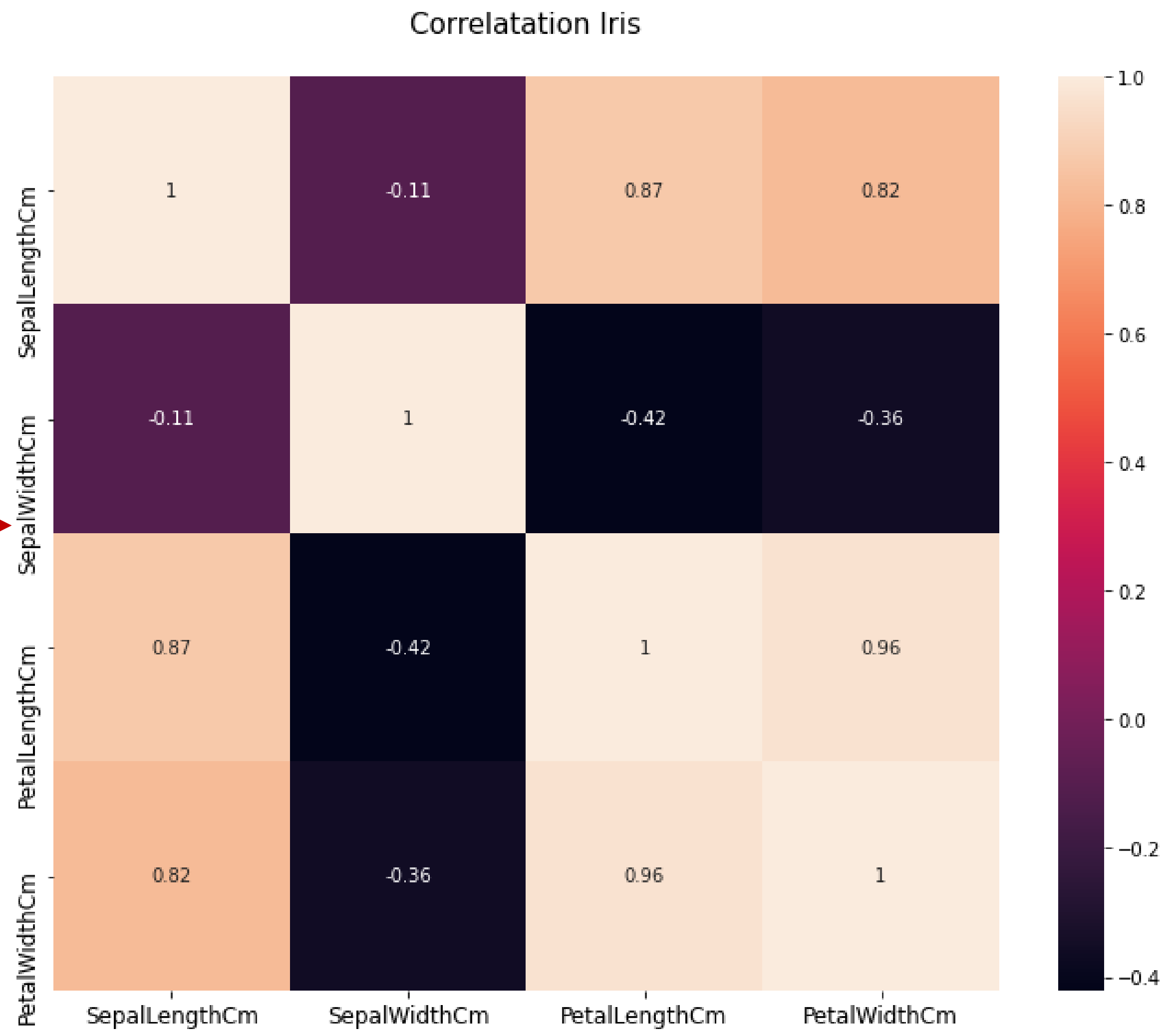


Variables correlation

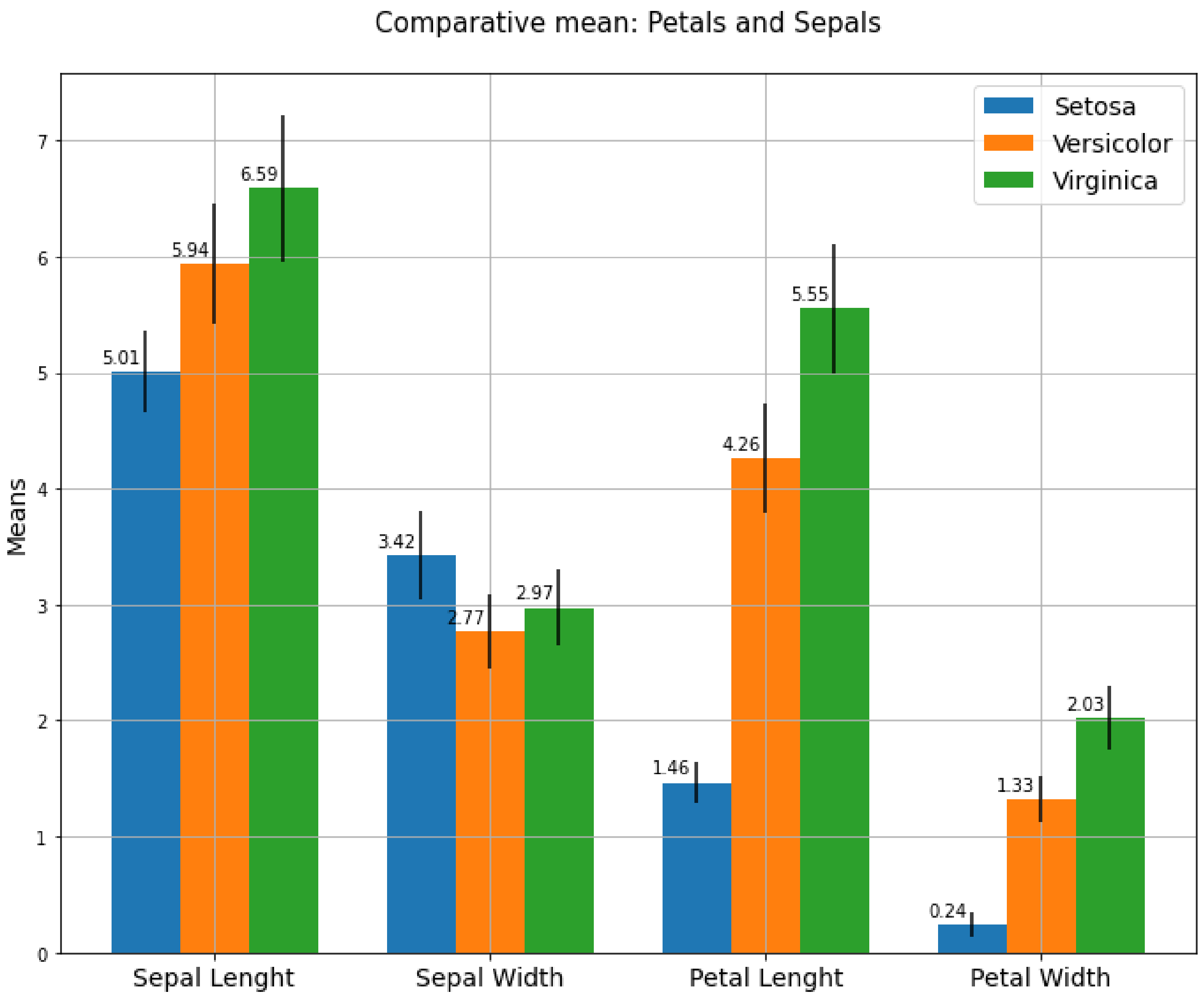
```
# variables correlation
corr = data.corr(); corr
```

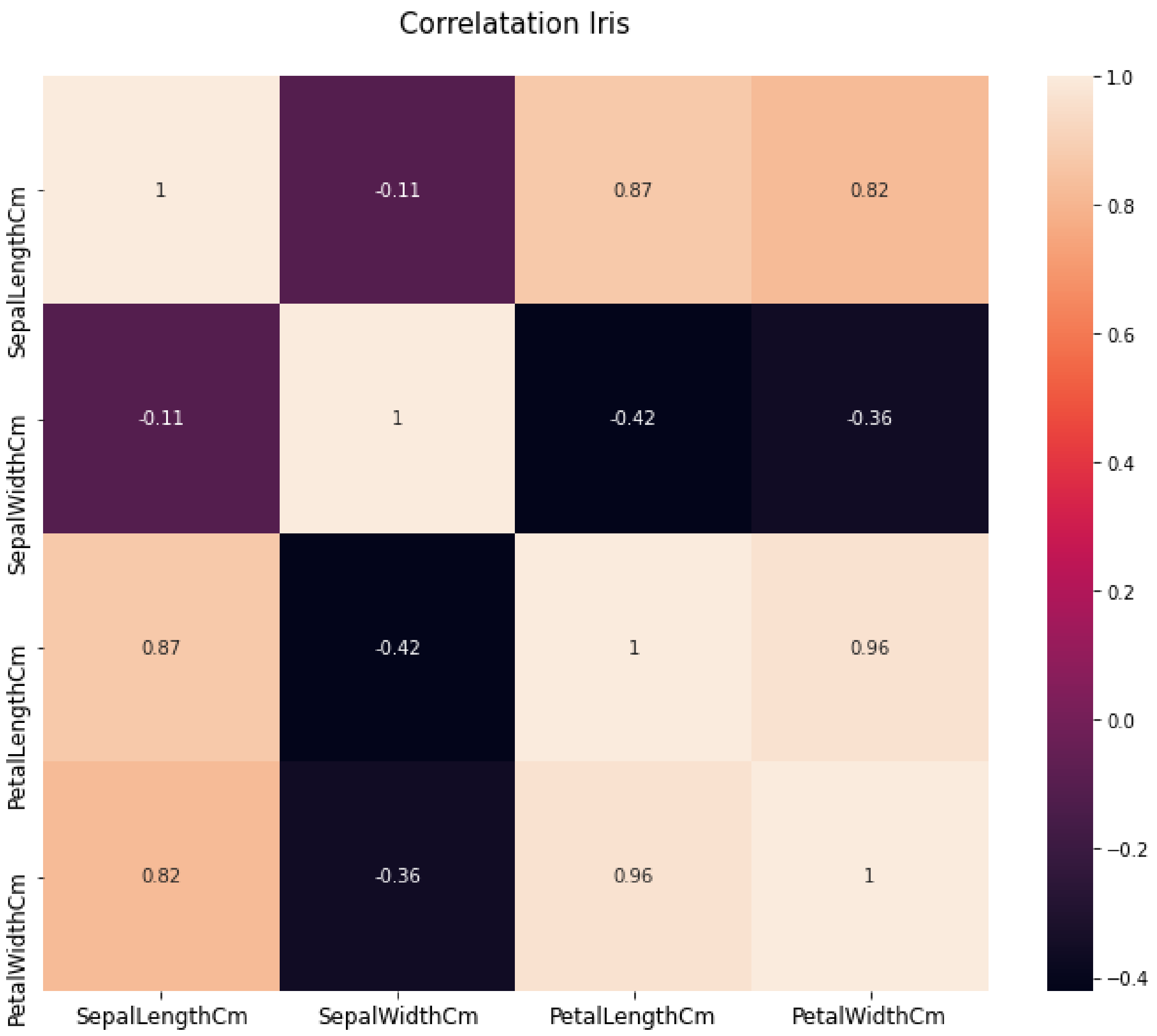
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

```
fig, ax = plt.subplots(1,2, figsize=(19,8))
ax[0].set_title('Comparative mean: Petals and Sepals\n', fontsize=15)

rects1 = ax[0].bar(x - 0.125 - width/2, se, width, label='Setosa',
                   yerr=se_err)
rects2 = ax[0].bar(x - 0.125  + width/2, ve, width, label='Versicolor',
                   yerr=ve_err)
rects3 = ax[0].bar(x - 0.125 + 1.5*width, vi, width, label='Virginica',
                   yerr=vi_err)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

ax[0].set_xticks(x)
ax[0].set_xticklabels(labels, fontsize=14)
ax[0].set_ylabel('Means', fontsize=14)
ax[0].legend(fontsize=14)
ax[0].grid(True)

ax[1].set_title('Correlatation Iris\n', fontsize=15)
sns.heatmap(corr, annot=True, ax=ax[1])
# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.xticks(fontsize=12, color='black')
plt.yticks(fontsize=12, color='black')

plt.show()
```



Variables correlation





Predictive Model

```python
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

def predict_iris():
    lst = []
    grand = ['Sepal lenght', 'Sepal widht', 'Petal lenght', 'Petal widht']
    for i, n in enumerate(grand):
        lst += [float(input('Insert {}:'. format(grand[i])))]
    insert = [lst]

    le = LabelEncoder()
    data['Species'] = le.fit_transform(data['Species'])

    X = data.drop(columns=['Species'])
    Y = data['Species']
    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)

    model = LogisticRegression()
    model.fit(x_train, y_train)

    caract = model.predict(insert)

    print('\nAccuracy model: {:.2f} % \n'. format(model.score(x_test, y_test)*100))
    if caract == 0:
        print('Your Iris specie is: Setosa')
    elif caract == 1:
        print('Your Iris specie is: Versicolor')
    elif caract == 2:
        print('Your Iris specie is: Viginica')
    else:
        print("This Iris isn't cataloged") # Always try put in a cluster

predict_iris()
```

out range parameters

```
Insert Sepal lenght:18
Insert Sepal widht:40
Insert Petal lenght:2.6
Insert Petal widht:100

Accuracy model: 100.00 %

Your Iris specie is: Viginica
```

```
Your Iris specie is: Versicolor
```

```
pecies
0
Your Iris specie is: Setosa
```

EXPLORATORY DATA ANALYSIS

Dataset extraction #2

Sum spot dataset

```
data = np.genfromtxt("SN_m_tot_V2.0.txt",skip_footer=0)
data = pd.DataFrame(data, columns=['Ano','NaN1','Data','Nums Spots', 'NaN2','NaN3']).copy(); data
```

	Ano	NaN1	Data	Nums Spots	NaN2	NaN3
0	1749.0	1.0	1749.042	96.7	-1.0	-1.0
1	1749.0	2.0	1749.123	104.3	-1.0	-1.0
2	1749.0	3.0	1749.204	116.7	-1.0	-1.0
3	1749.0	4.0	1749.288	92.8	-1.0	-1.0
4	1749.0	5.0	1749.371	141.7	-1.0	-1.0
...	...	...	...	...	...	...
3208	2016.0	5.0	2016.373	52.1	4.7	810.0
3209	2016.0	6.0	2016.456	20.9	2.2	886.0
3210	2016.0	7.0	2016.540	32.5	3.7	910.0
3211	2016.0	8.0	2016.624	50.7	4.4	879.0
3212	2016.0	9.0	2016.708	44.7	3.8	742.0

3213 rows × 6 columns

Example of a data table taken from the NASA domain, [Solar Cycle Prediction], from the .txt file format. It tells us the monthly sunspot averages over the years.

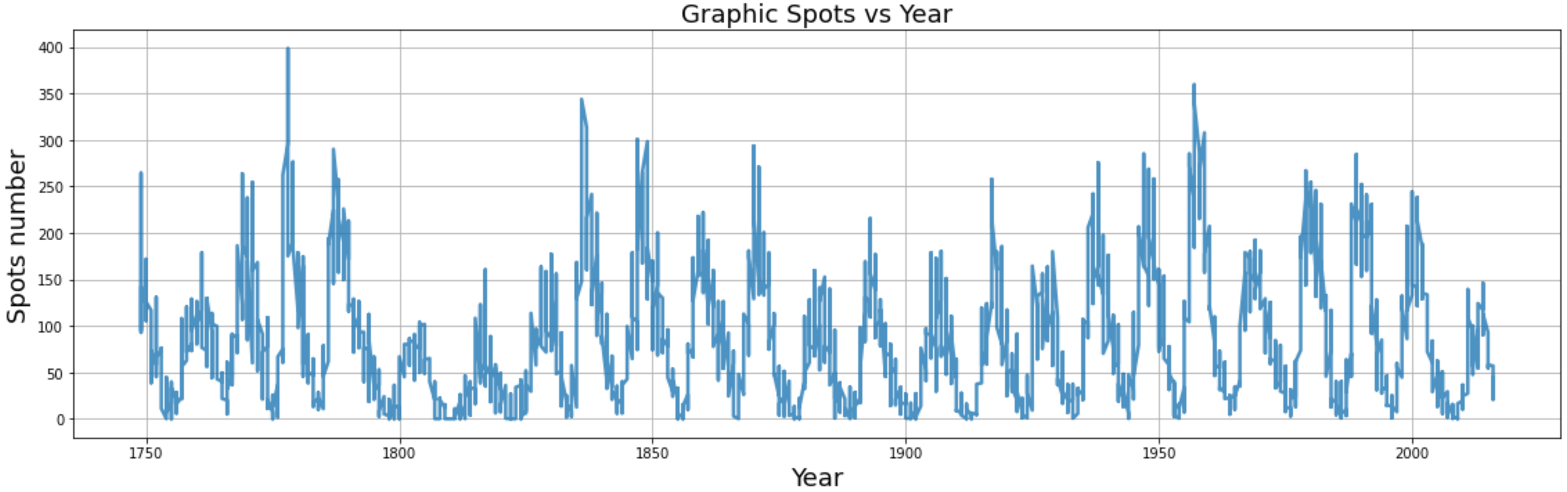
The table has 3213 rows and 6 columns.

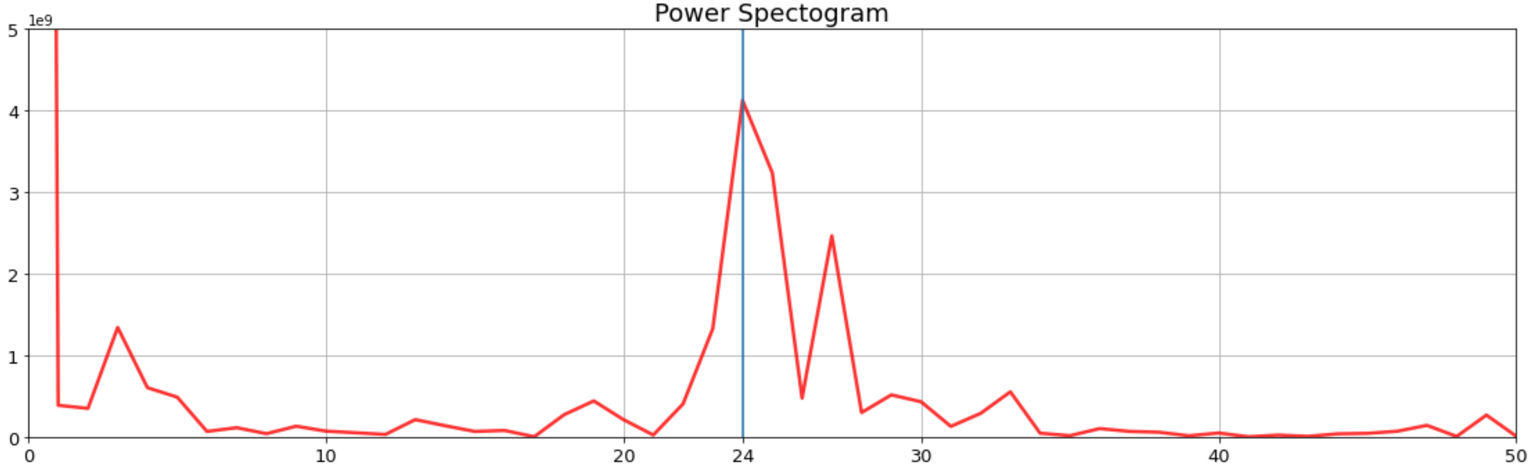
Sum spot dataset

```python
time = np.array(data_main['Ano'])
n_spots = np.array(data_main['Nums Spots'])

fft_spots = np.fft.fft(n_spots)

fig, ax = plt.subplots(2, 1, figsize=(16,10))

ax[0].set_title('Graphic Spots vs Year', fontsize=18)
ax[0].plot(time, n_spots,'-', linewidth=2.5, alpha=0.8)
ax[0].set_xlabel("Year", fontsize=18)
ax[0].set_ylabel("Spots number",fontsize=18)
ax[0].grid(True)

ax[1].set_title('Power Spectogram', fontsize=18)
ax[1].plot(abs((fft_spots)**2), linewidth=2.5,
         alpha=0.8, color='red')
ax[1].axvline(24)
ax[1].set_xlim(1,50)
ax[1].set_ylim(0,5e9)
plt.xticks(list(plt.xticks()[0]) + [24], fontsize=18)

plt.xticks(fontsize=13, color='black')
plt.yticks(fontsize=13, color='black')
fig.tight_layout()
plt.grid(True)
plt.show()
```

We can see a peak at 24, with that, taking into account the normalization factor N, we invert to get the period instead of the frequency, the period:

$$\frac{N}{k} = \frac{3213}{24} = 134\,months \cong 11\,years$$





Python: Fundamentals for Exploratory Data Analysis

Eduardo Destefani Stefanato[1]*
Vitor Souza Premoli Pinto de Oliveira[1]*

[1]Universidade Federal do Espírito Santo

Artificial Intelligence Applied to Images

REFERENCES

PiPy. **Find, install and publish Python packages with the Python Package Index**. Available in: https://pypi.org/. Access in: 22 Jul. 2021.

Matplotlib. **Matplotlib: Python plotting — Matplotlib 3.4.2 documentation.** Available in: https://matplotlib.org/. Access in: 22 Jul. 2021.

Numpy. **NumPy Reference — NumPy v1.21 Manual.** Available in: https://numpy.org/doc/stable/reference/. Access in: 22 Jul. 2021.

NumFocus. **Pandas.** Available in: https://pandas.pydata.org/. Access in: 29 Jul. 2021.

Kaggle. **Iris Species.** Available in: https://www.kaggle.com/uciml/íris/. Access in: 08 Aug. 2021.