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

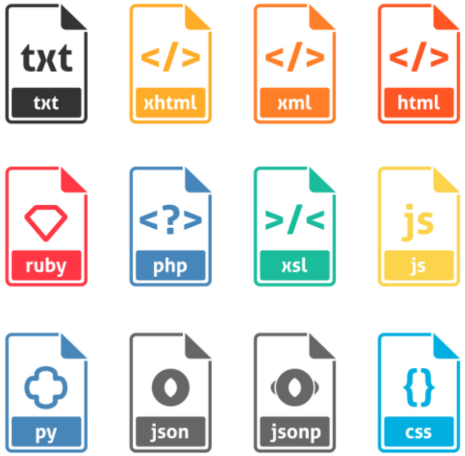
Sumary

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.



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. As a reference we have extracted from a .csv file available in the Kaggle repository.

kaggle

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

```python
# 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.

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

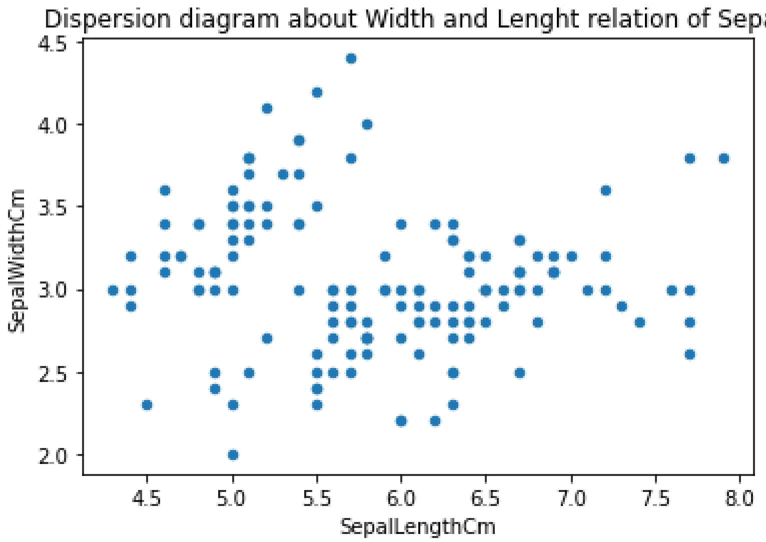
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

Correlation between the parameters of each species in the dataset.

plot data

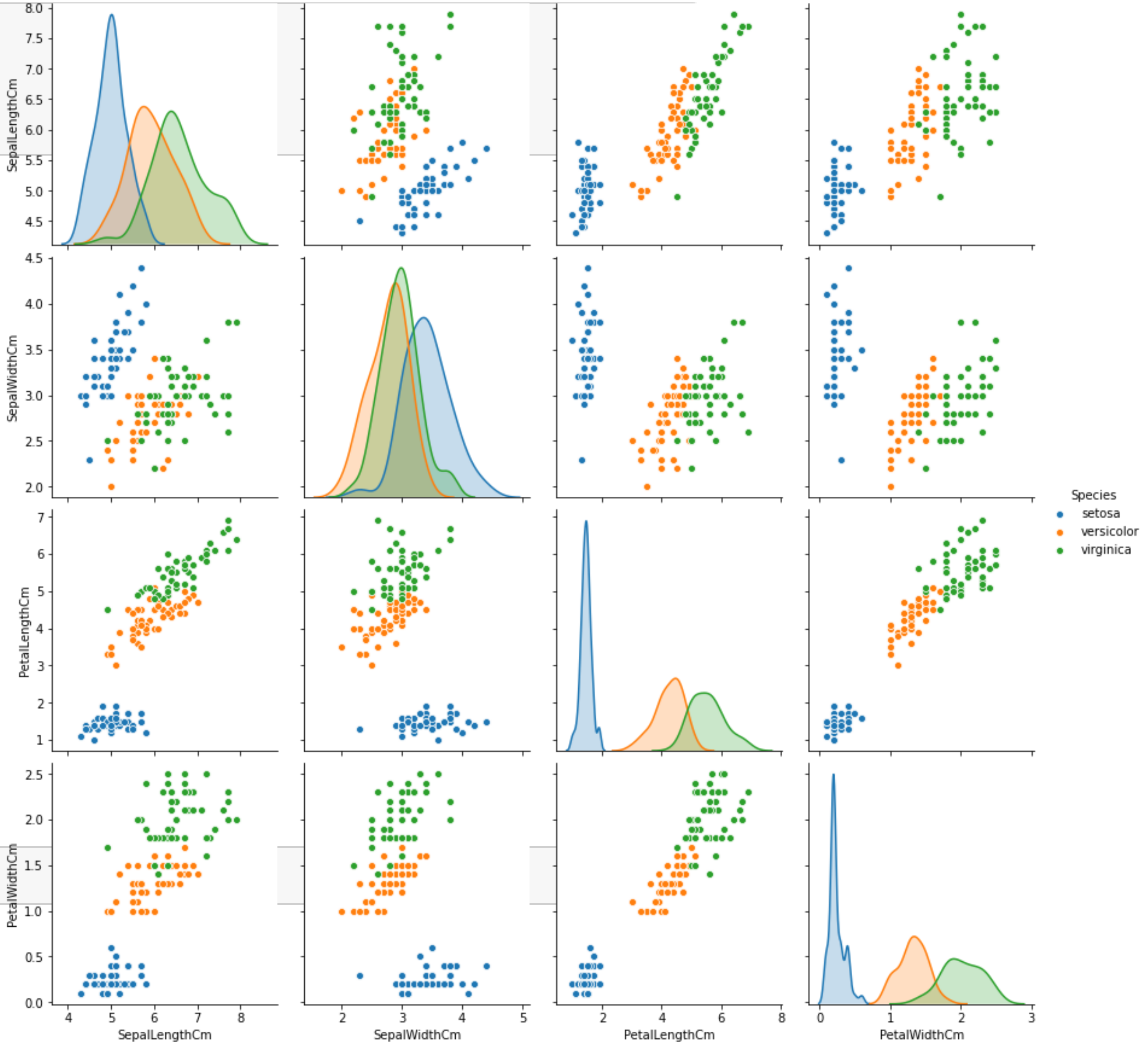
```
#Gráfico de dispersão
data.plot(x="SepalLengthCm", y="SepalWidthCm", kind="scatter")
plt.title("Dispersion diagram about Width and Lenght relation of Sepal ")
```

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




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

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

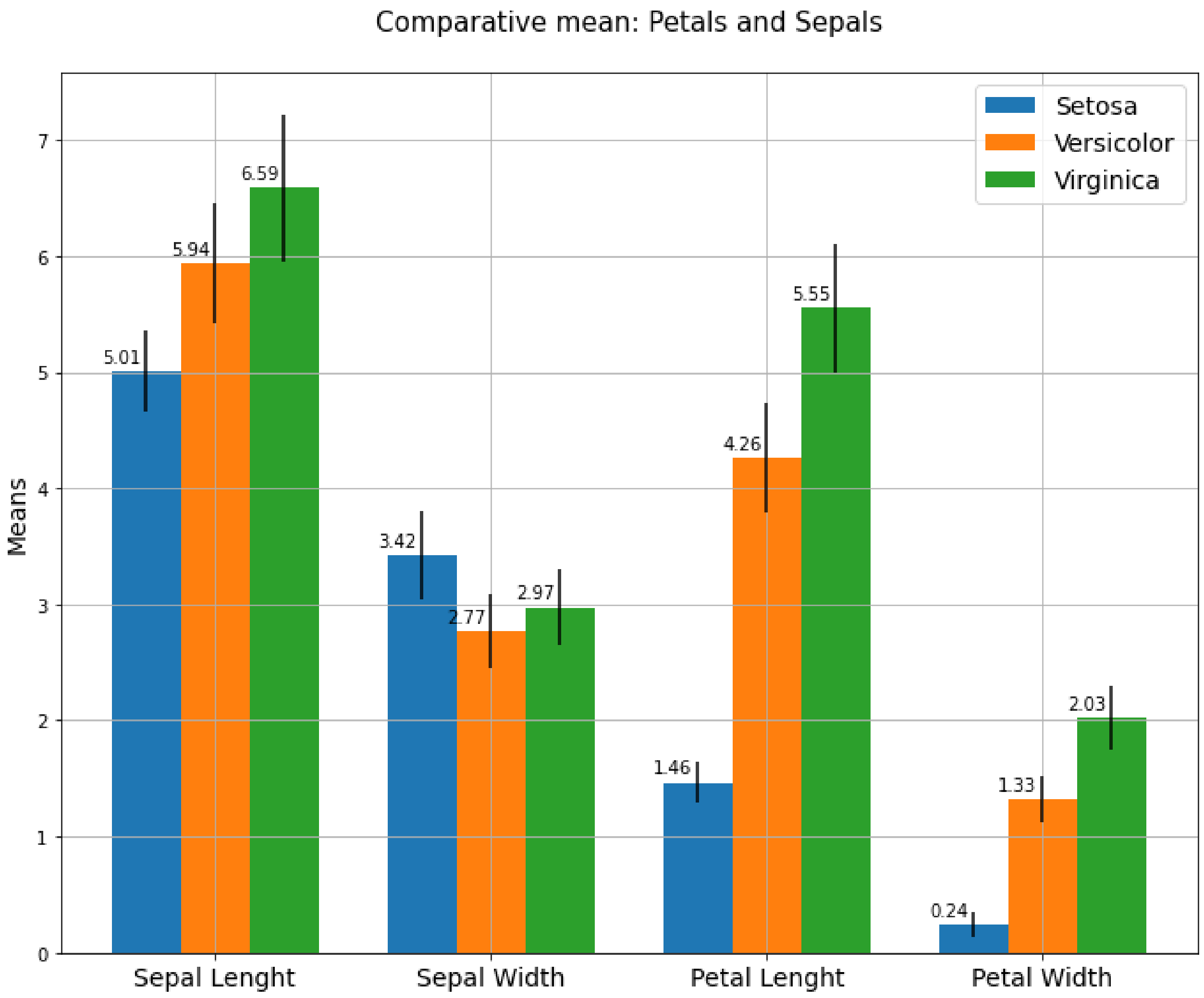


Aggregation and transformation

```
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()
```

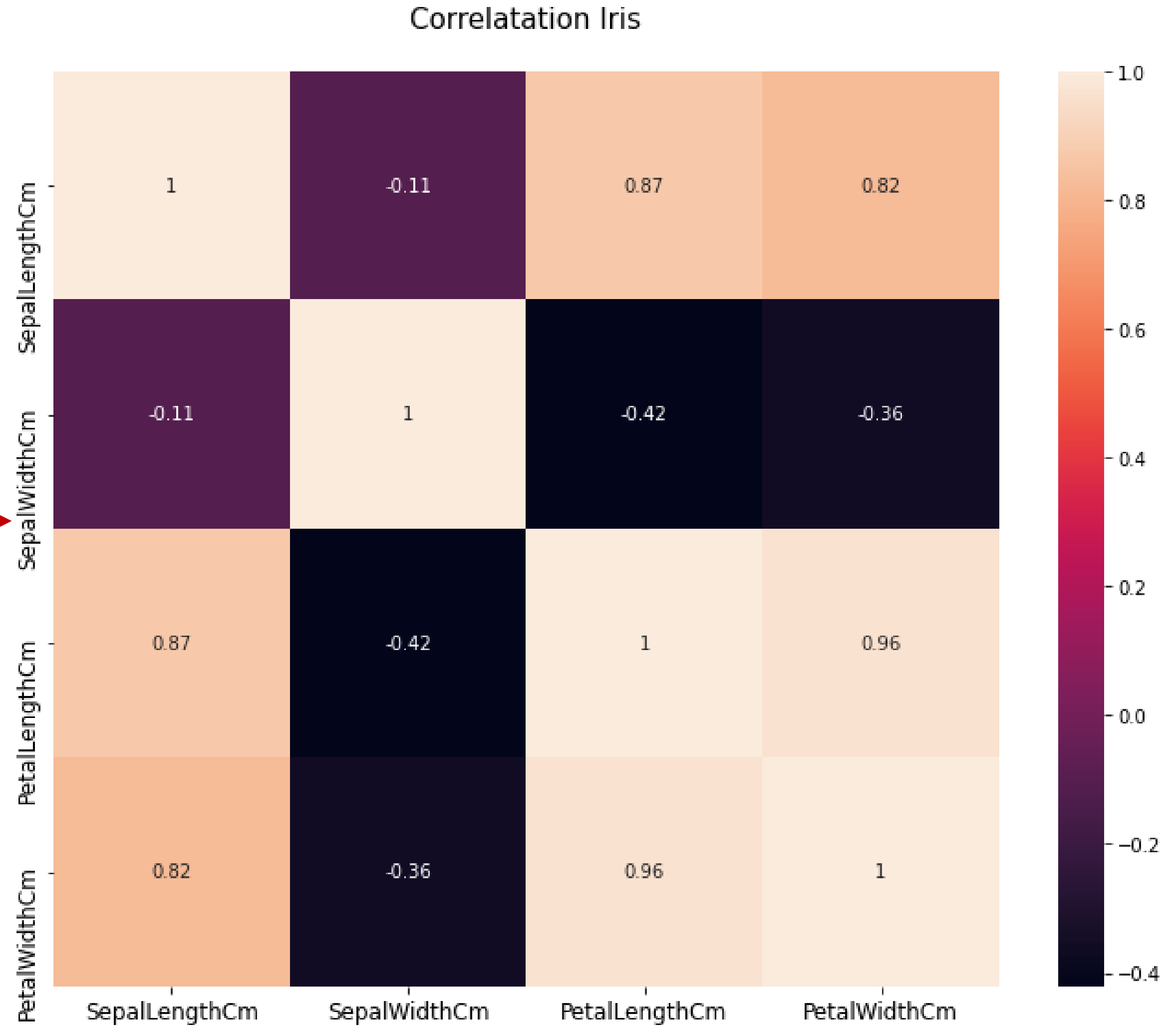


Variables correlation

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

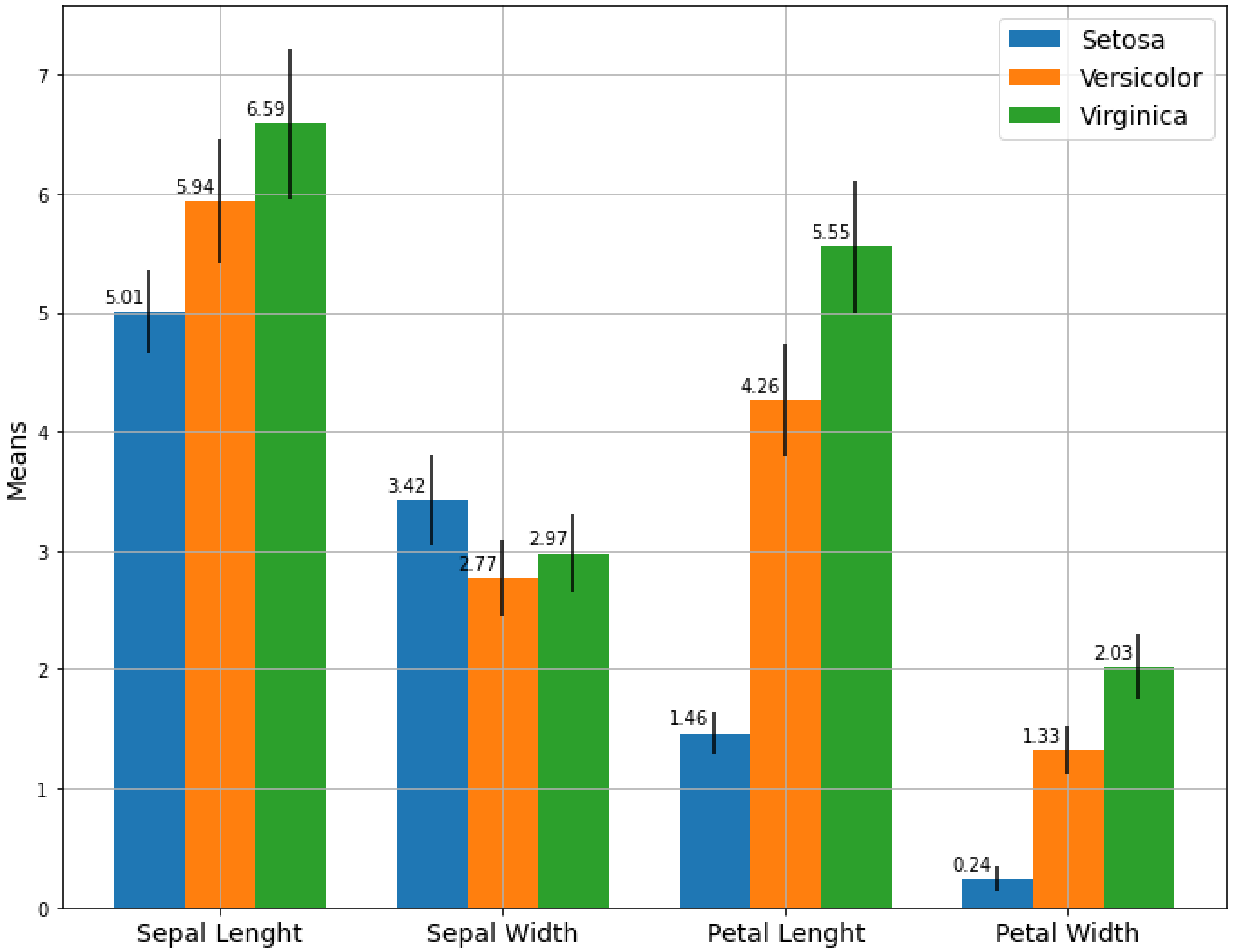
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

```
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()
```

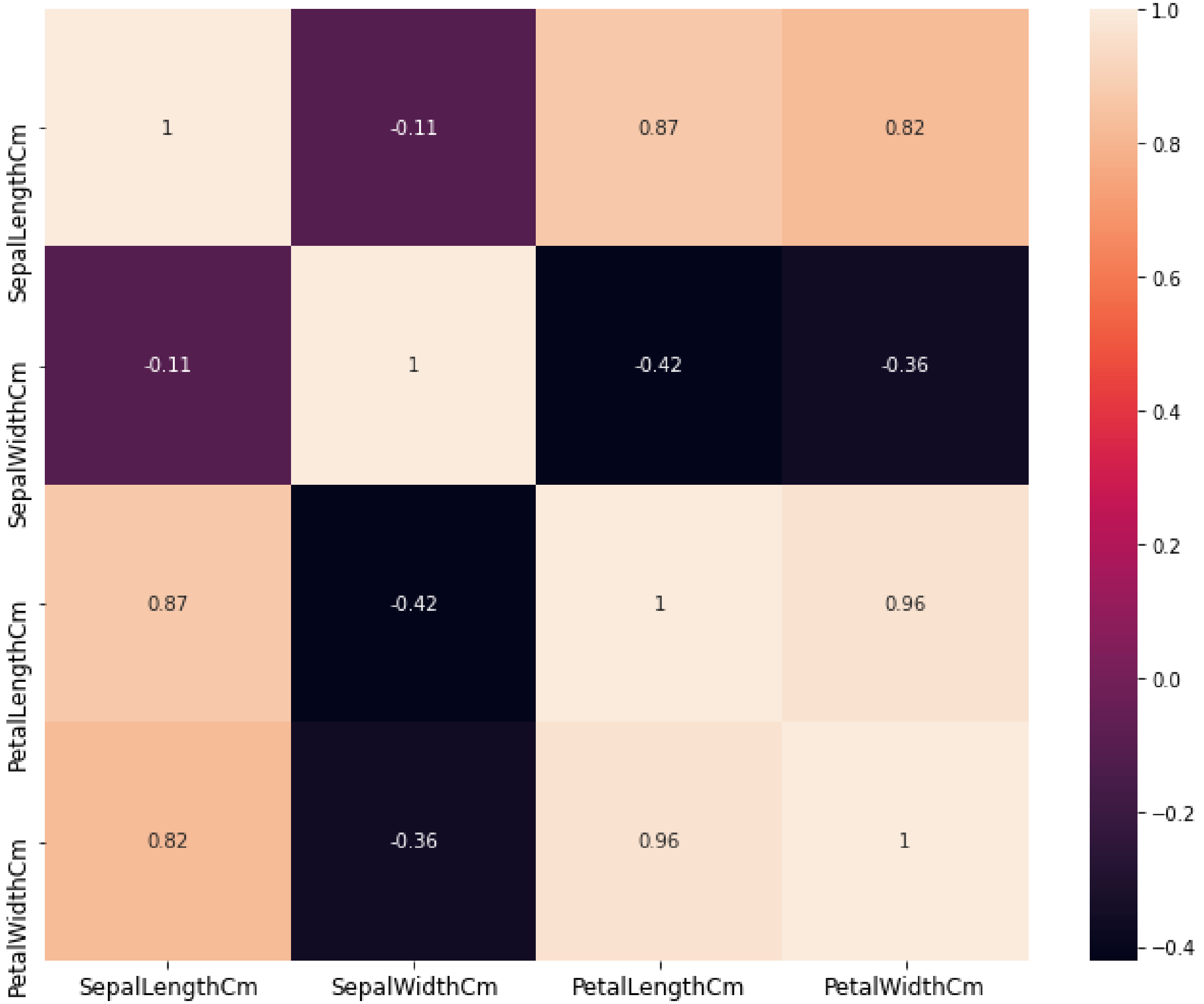


Variables correlation

Comparative mean: Petals and Sepals



Correlatation Iris



EXPLORATORY DATA ANALYSIS

Machine learning

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
```

```
Species
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

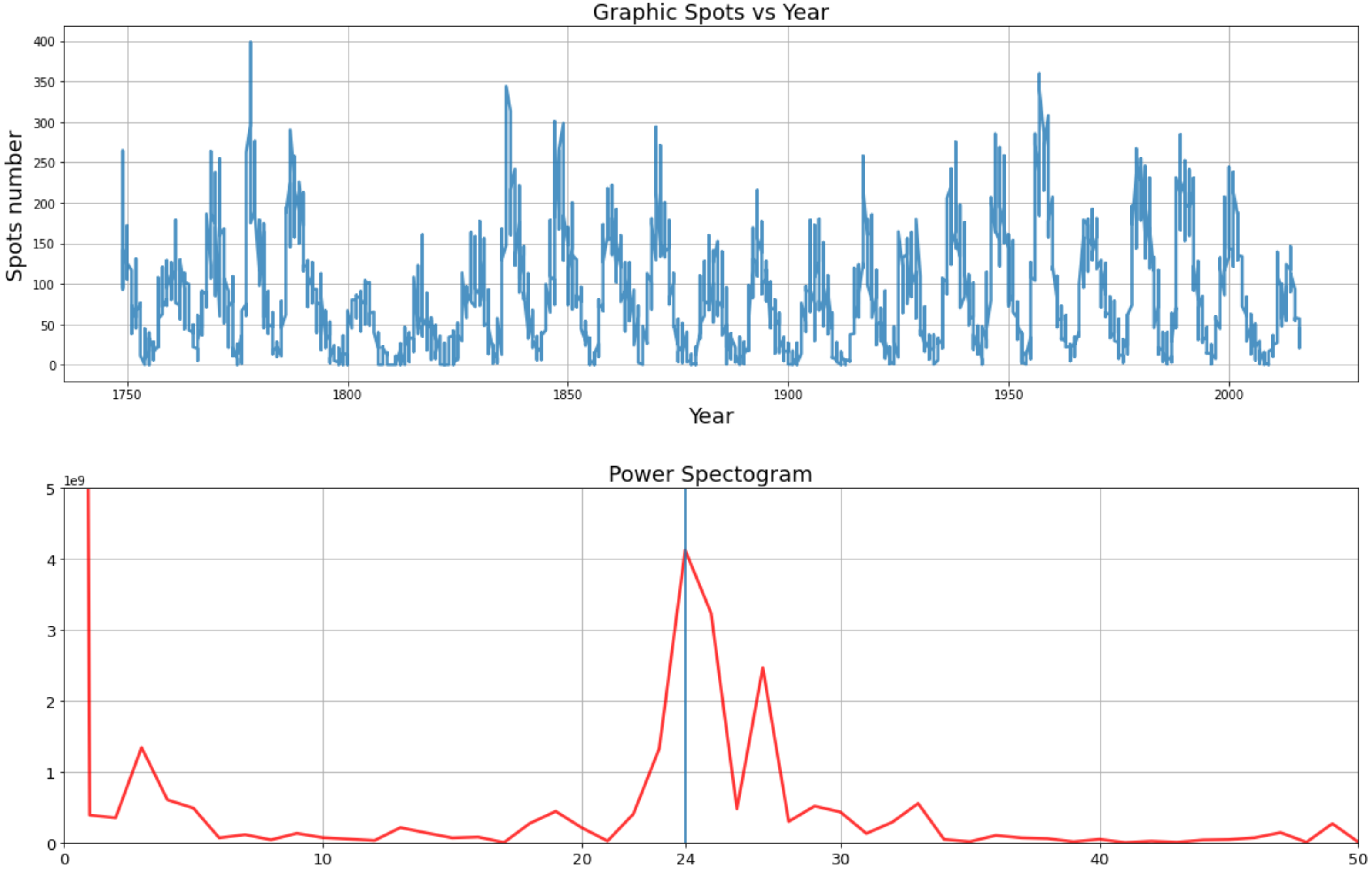
3213 rows × 6 columns

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

```
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.