University of Colima.

Faculty of Mechanical and Electrical Engineering.

Intelligent Computing Engineering.

UNIVERSIDAD DE COLIMA

Data Analysis and Visualization.

6°D.

Date: 11/04/2024.

Place: Mexico, Colima, Coquimatlan.

CRISTIAN ARMANDO LARIOS BRAVO.

Data Visualization with Matplotlib

In this exercise we analize a dataset really common that is the flowers dataset of Iris variety. There are three types of Iris:

- Iris setosa.
- Iris versicolor.
- Iris virginica.

By Cristian Armando Larios Bravo.

🎨CONNECT. We can apply data science to analyze and visualize characteristics of iris flowers

👽 OVERVIEW

Iris is a flowering plant genus of 310 accepted species with showy flowers. As well as being the scientific name, iris is also widely used as a common name for all Iris species, as well as some belonging to other closely related genera. A common name for some species is flags, while the plants of the subgenus Scorpiris are widely known as junos, particularly in horticulture. It is a popular garden flower. The often-segregated, monotypic genera Belamcanda (blackberry lily, I. domestica), Hermodactylus (snake's head iris, I. tuberosa), and Pardanthopsis (vesper iris, I. dichotoma) are currently included in Iris. Three Iris varieties are used in the Iris flower data set outlined by Ronald Fisher in his 1936 paper.
A Bubble Chart is a multi-variable graph that is a cross between a Scatterplot and a Proportional Area Chart.

Bubble Charts are typically used to compare and show the relationships between categorised circles, by the use of positioning and proportions. The overall picture of Bubble Charts can be used to analyse for patterns/correlations. A violin plot depicts distributions of numeric data for one or more groups using density curves. The width of each curve corresponds with the approximate frequency of data points in each region. Densities are frequently accompanied by an overlaid chart type, such as box plot, to provide additional information.

In [ ]:
```
# Import the libraries
import sklearn.datasets as ds
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

The Iris Dataset

In [ ]:
```
# We load the Iris dataset from sklearn
iris = ds.load_iris()
```

In [ ]:
```
# Print the dataset.
# In a dictionary with different fields
# Where is the data dictionary of the dataset

print(iris)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
```

```
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.8, 2.7, 3.9, 1.2],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3. , 4.5, 1.5],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
       [6.3, 2.3, 4.4, 1.3],
       [5.6, 3. , 4.1, 1.3],
       [5.5, 2.5, 4. , 1.3],
       [5.5, 2.6, 4.4, 1.2],
       [6.1, 3. , 4.6, 1.4],
       [5.8, 2.6, 4. , 1.2],
       [5. , 2.3, 3.3, 1. ],
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
```

```
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'targe
t_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. _
iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteris
tics:**\n\n    :Number of Instances: 150 (50 in each of three classes)\n    :Number
of Attributes: 4 numeric, predictive attributes and the class\n    :Attribute Inform
ation:\n        - sepal length in cm\n        - sepal width in cm\n        - petal l
ength in cm\n        - petal width in cm\n        - class:\n                - Iris-S
etosa\n                - Iris-Versicolour\n                - Iris-Virginica\n
\n    :Summary Statistics:\n\n    ============== ==== ==== ======= ===== ==========
=========\n                    Min  Max   Mean    SD   Class Correlation\n    ======
======== ==== ==== ======= ===== ====================\n    sepal length:   4.3  7.9
5.84   0.83    0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194\n    pet
```

al length: 1.0 6.9 3.76 1.76 0.9490 (high!)\n petal width: 0.1 2.5 1.20 0.76 0.9565 (high!)\n ============== ==== ==== ======= ===== ========== ==========\n\n :Missing Attribute Values: None\n :Class Distribution: 33.3% for each of 3 classes.\n :Creator: R.A. Fisher\n :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n :Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhaps the best known database to be found in the\npattern recognition literature. Fisher\'s paper is a classic in the field and\nis referenced frequently to this day. (See Duda & Hart, for example.) The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant. One class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each other.\n\n|details-start|\n**References**\n|details-split|\n\n- Fisher, R.A. "The use of multiple measurements in taxonomic problems"\n Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n Mathematical Statistics" (John Wiley, NY, 1950).\n- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.\n- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n Structure and Classification Rule for Recognition in Partially Exposed\n Environments". IEEE Transactions on Pattern Analysis and Machine\n Intelligence, Vol. PAMI-2, No. 1, 67-71.\n- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions\n on Information Theory, May 1972, 431-433.\n- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al"s AUTOCLASS II\n conceptual clustering system finds 3 classes in the data.\n- Many, many more ...\n\n|details-end|', 'feature_names': ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data_module': 'sklearn.datasets.data'}

Separate the data contained in the dataset

In [ ]:
```
# Separate the data contained in the dataset (Dictionary)
# Save the data of the dataset in the variable iris_x
iris_x = iris['data']
# Flowers classes, 0 = setosa, 1 = versicolor, 2 = virginica
iris_y = iris['target']
# Label dimesions: petal width, petal length, sepal width, sepal length
# 'sepal_lenght (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'

iris_x_labels = iris['feature_names']
# Labels of the names of the flowers setosa, versicolor, virginica
iris_y_labels = iris['target_names']
```

Separate the flowers into three variables

In [ ]:
```
# Separate the flowers into three variables
setosa = np.where(iris_y == 0)
versicolor = np.where(iris_y == 1)
virginica = np.where(iris_y == 2)
# Print the variables
print("Setosa:", setosa)
print("Versicolor:", versicolor)
print("Virginica", virginica)
```

```
Setosa: (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
      dtype=int64),)
Versicolor: (array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
66,
       67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
       84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99],
      dtype=int64),)
Virginica (array([100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
       113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125,
       126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
       139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149], dtype=int64),)
```

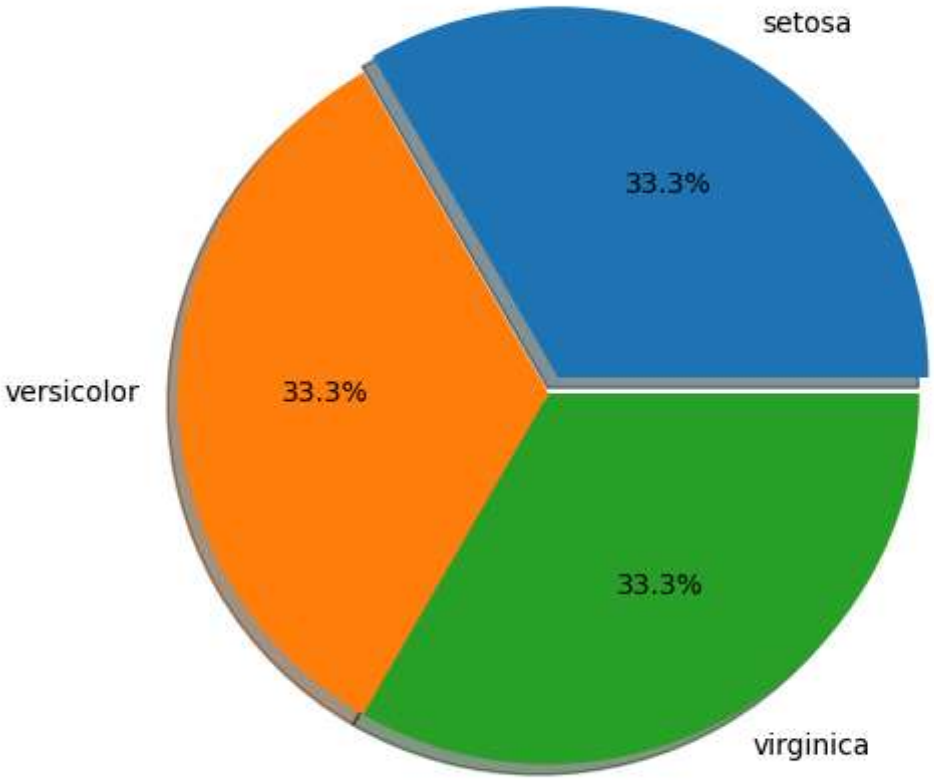
Unique method of Numpy

In [ ]:
```
# Unique method of numpy
# Find the unique values and count their frequency in an array.
# Return two arrays, the first one with the unique values and the second one with t
(unique, frequency) = np.unique(iris_y, return_counts=True)
print(unique)
print(frequency)
```

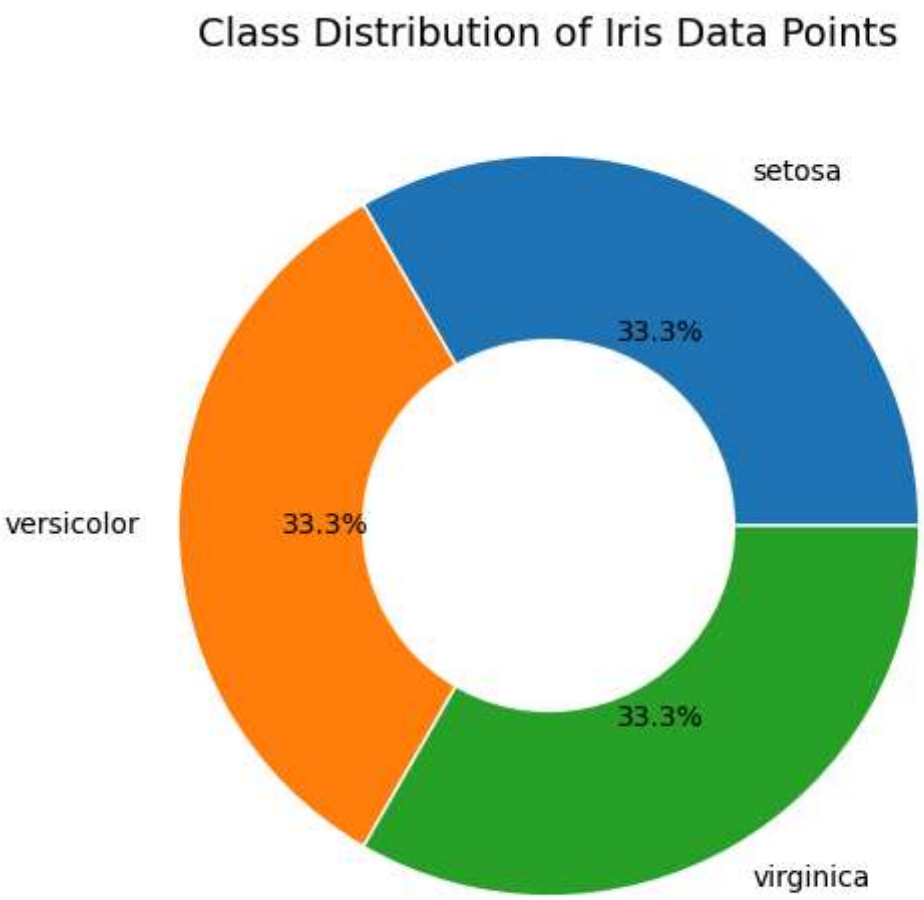
```
[0 1 2]
[50 50 50]
```

Show proportions on a pie chart.

In [ ]:
```
# Circle or pie chart to show the proportion of the data
# Describe on a standing graph to know how many flowers there are in each type
plt.figure(figsize=(6, 6))
# Build the graph
plt.pie(frequency, labels=iris_y_labels, autopct='%1.1f%%', shadow=True, explode=(0
# To see all the options of the library
plt.show()
```

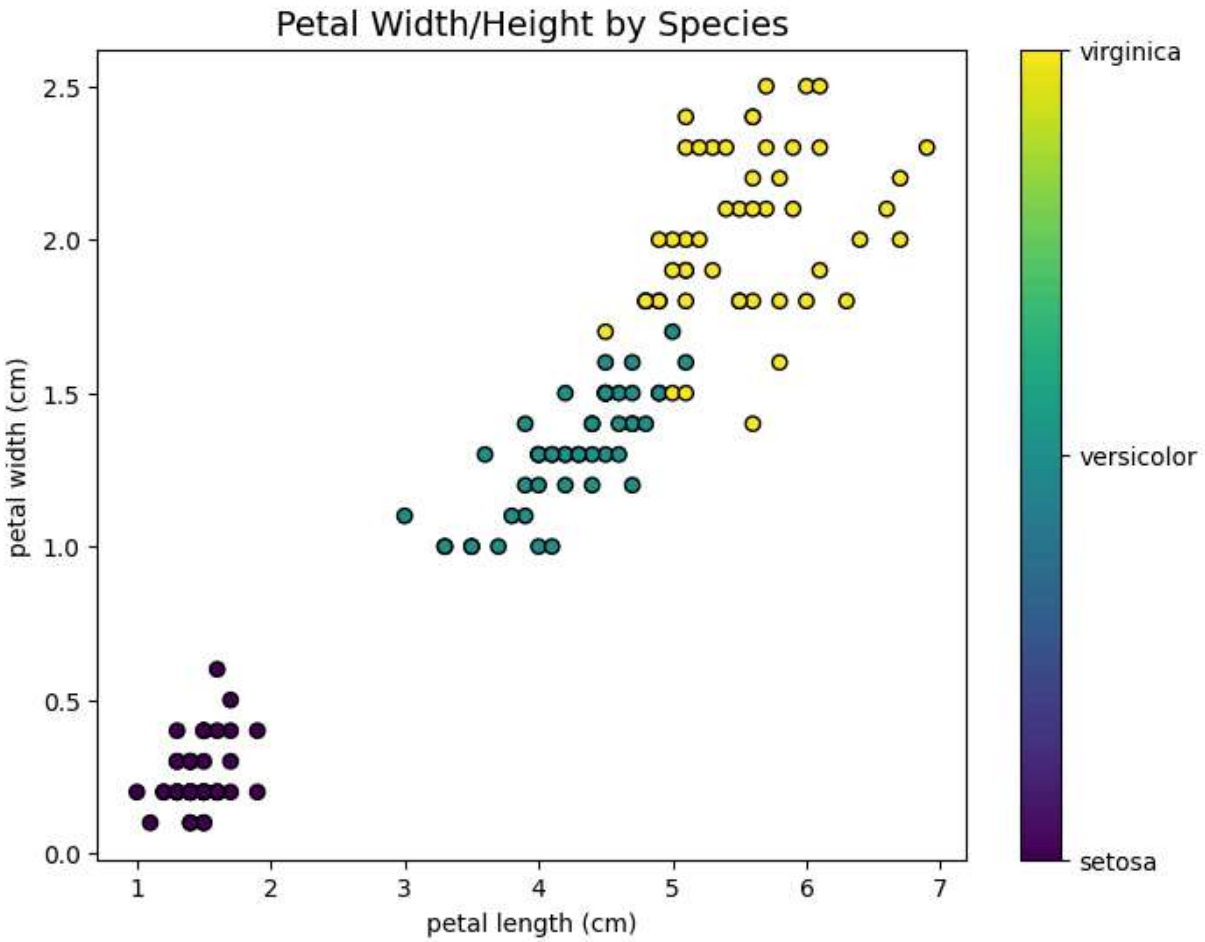


In [ ]:
```
# We create a second representation of them
# It is convenient to present the same information in different formats
plt.figure(figsize=(6,6))
plt.pie(frequency, labels=iris_y_labels, autopct="%1.1f%%",
        textprops={'fontsize': 10}, wedgeprops={'edgecolor':'#ffffff'})

# We create a circle to draw the donut
circle = plt.Circle(xy=(0,0), radius=0.5, color='white')
# We add the circle with the graphic context and the artist method
plt.gca().add_artist(circle)
# Add a title
plt.title('Class Distribution of Iris Data Points', size=14)
# Show the graph
plt.show()
```

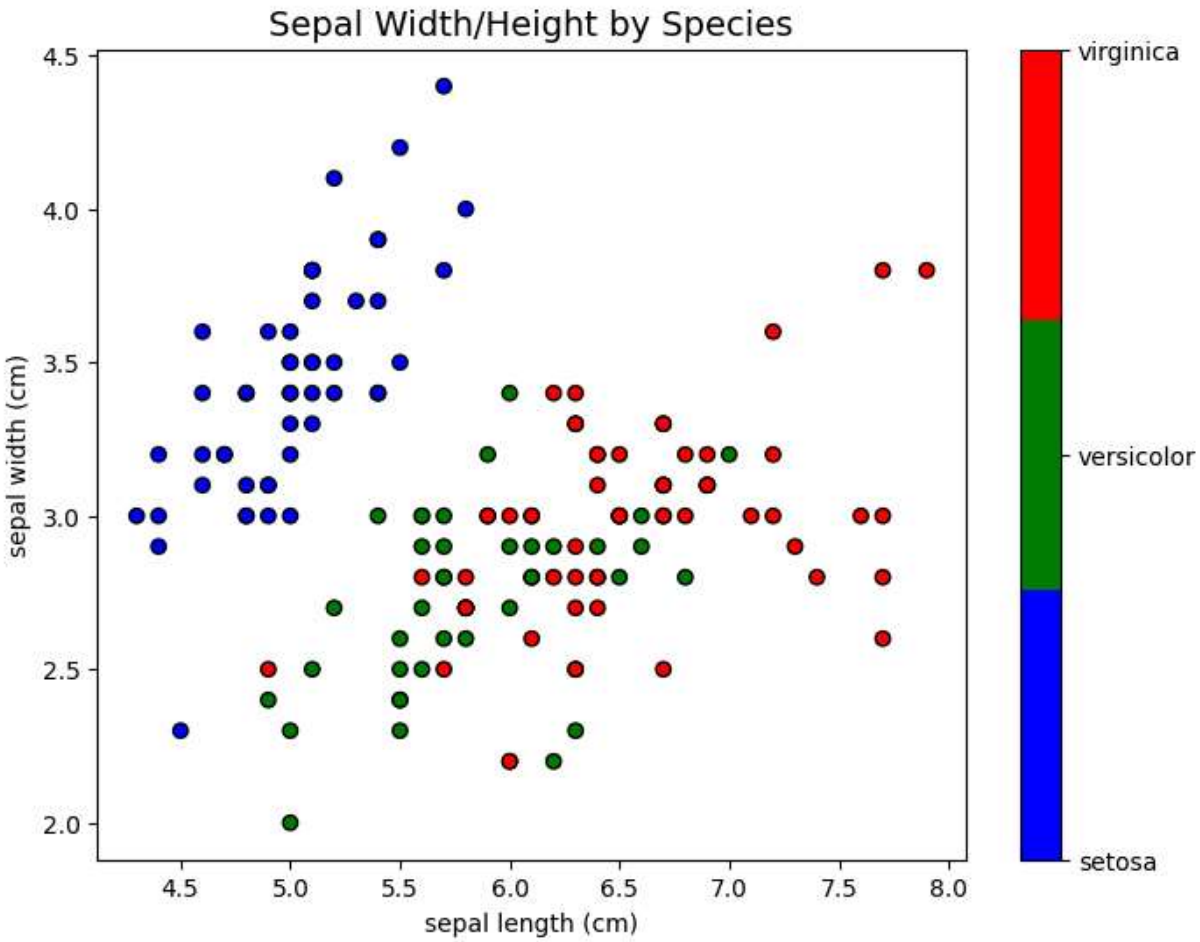


In [ ]:
```
from matplotlib.ticker import FuncFormatter
formatter = FuncFormatter(lambda v, _: iris_y_labels[v])

# Lambda function example. They are simple, short and only do one function
summ = lambda x,y: x+y
# This would be the equivalence of the lambda function
def summ2(x,y):
    return x+y
```

In [ ]:
```
# Build a figure
plt.figure(figsize=(8,6))
plt.scatter(iris_x[:,2], iris_x[:,3], c= iris_y, edgecolors='black')
# Add the labels contained in the iris_y_labels variable in
# iris_x_labels
plt.xlabel(iris_x_labels[2])
plt.ylabel(iris_x_labels[3])

plt.title("Petal Width/Height by Species", size=14)
plt.colorbar(ticks=[0,1,2], format=formatter)
plt.show()
```

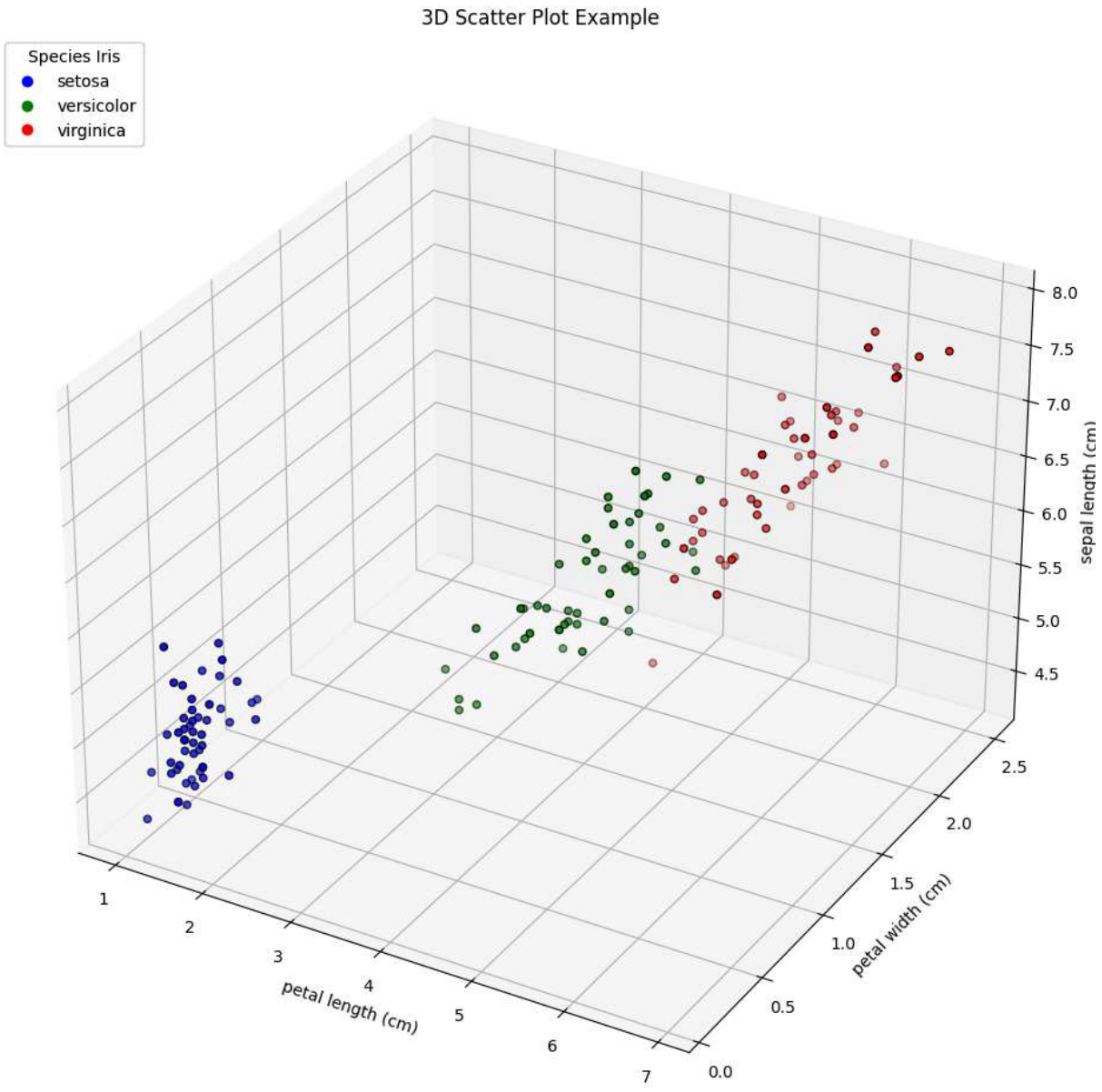


In [ ]:
```
# CHALLENGE
# Create a similar bubble or point graph taking the parameters
# Length and width of the sepals contained in the iris_x dataset

from matplotlib.colors import ListedColormap

# Build a figure
plt.figure(figsize=(8,6))
colors = ['blue', 'green', 'red']
color = ListedColormap(colors)
plt.scatter(iris_x[:,0], iris_x[:,1], c= iris_y, edgecolors='black', cmap=color)

# Add labels contained in the iris_y_labels variable in
# iris_x_labels
plt.xlabel(iris_x_labels[0])
plt.ylabel(iris_x_labels[1])

plt.title("Sepal Width/Height by Species", size=14)
plt.colorbar(ticks=[0,1,2], format=formatter)
plt.show()
```

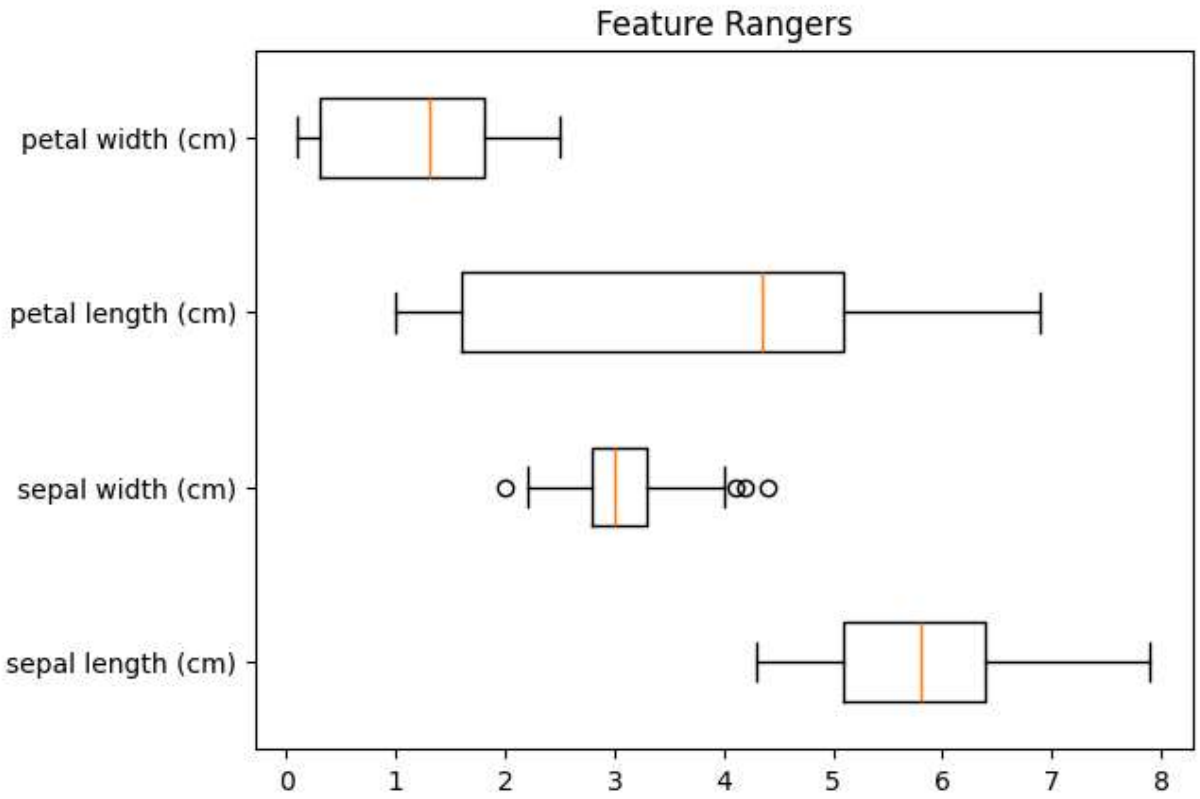


3D Graph

In [ ]:
```
# Create a figure of 8 by 6
fig = plt.figure(figsize=(16,12))
# Create a subplot in the fig object
ax = fig.add_subplot(111, projection='3d')
# Add the data to the figure, the labels
ax.set_xlabel(iris_x_labels[2]) # X axis
ax.set_ylabel(iris_x_labels[3]) # Y axis
ax.set_zlabel(iris_x_labels[0]) # Z axis
# Create the 3D figure
scatter = ax.scatter(iris_x[:, 2], iris_x[:, 3], iris_x[:, 0],
                     c=iris_y, cmap=color, edgecolors='black', s=20)

# Create a legend to differentiate the classes of flowers
handles,_ = scatter.legend_elements()
plt.title("3D Scatter Plot Example")
# Create the legends for each of the axes
legend = ax.legend(handles, iris_y_labels, title="Species Iris", loc='upper left')
# Add the legend to the figure with the artist method
ax.add_artist(legend)
plt.show()
```



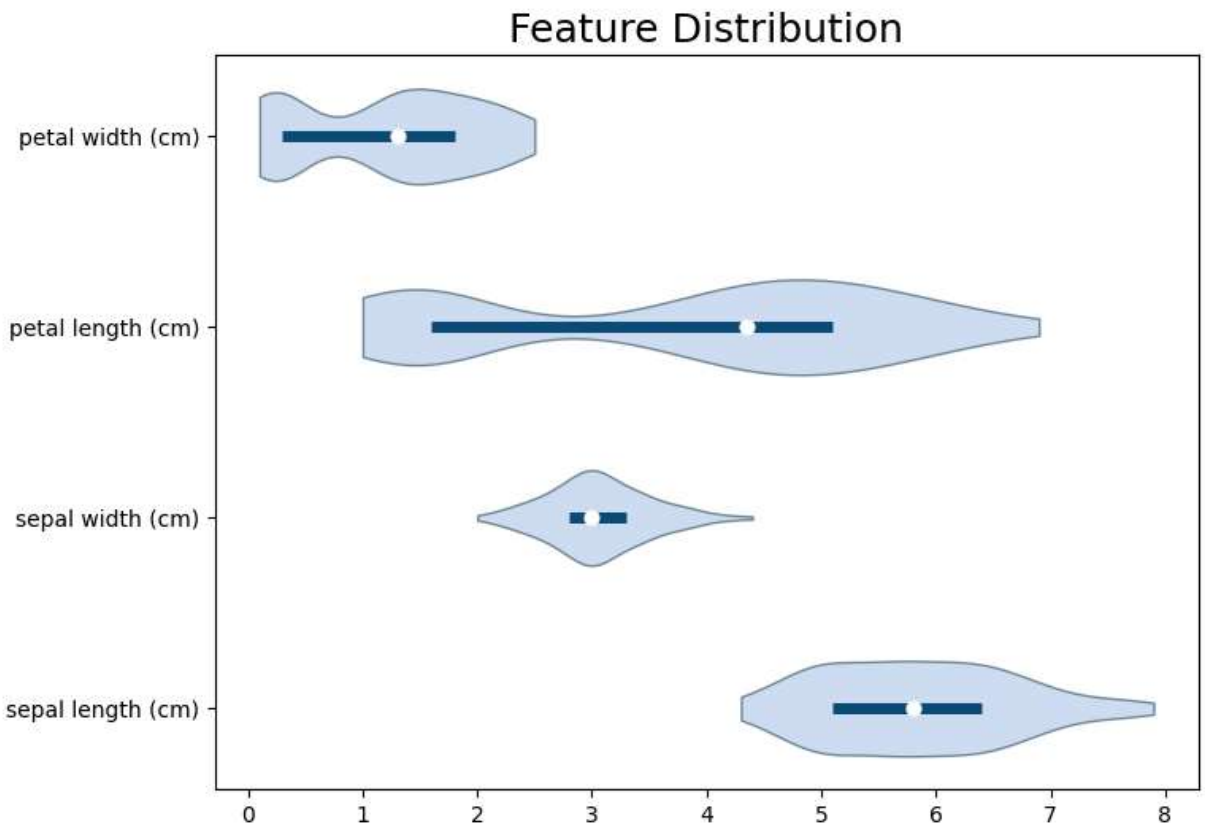
In [ ]:
```
plt.boxplot(iris_x, labels=iris_x_labels, vert=False)
plt.title("Feature Rangers")
plt.show()
```



Violin diagrams.

In [ ]:
```
# Calculate the percentiles of the data
q1, median, q3 = np.percentile(iris_x[:,], [25, 50, 75], axis=0)
```

In [ ]:
```
plt.figure(figsize=(8,6))
# Draw violin graph
parts = plt.violinplot(iris_x, vert=False, showextrema=False)
plt.title("Feature Distribution", size=18)
plt.yticks([1,2,3,4], iris_x_labels)

for pc in parts['bodies']:
    pc.set_facecolor('#9ec2e6')
    pc.set_edgecolor('#01263a')
    pc.set_alpha(0.5)

plt.gca().scatter(median, [1,2,3,4], color="white", zorder=3, s=40)

plt.gca().hlines([1,2,3,4], q1, q3, color='#0a4b78', lw=5)
plt.show()
```

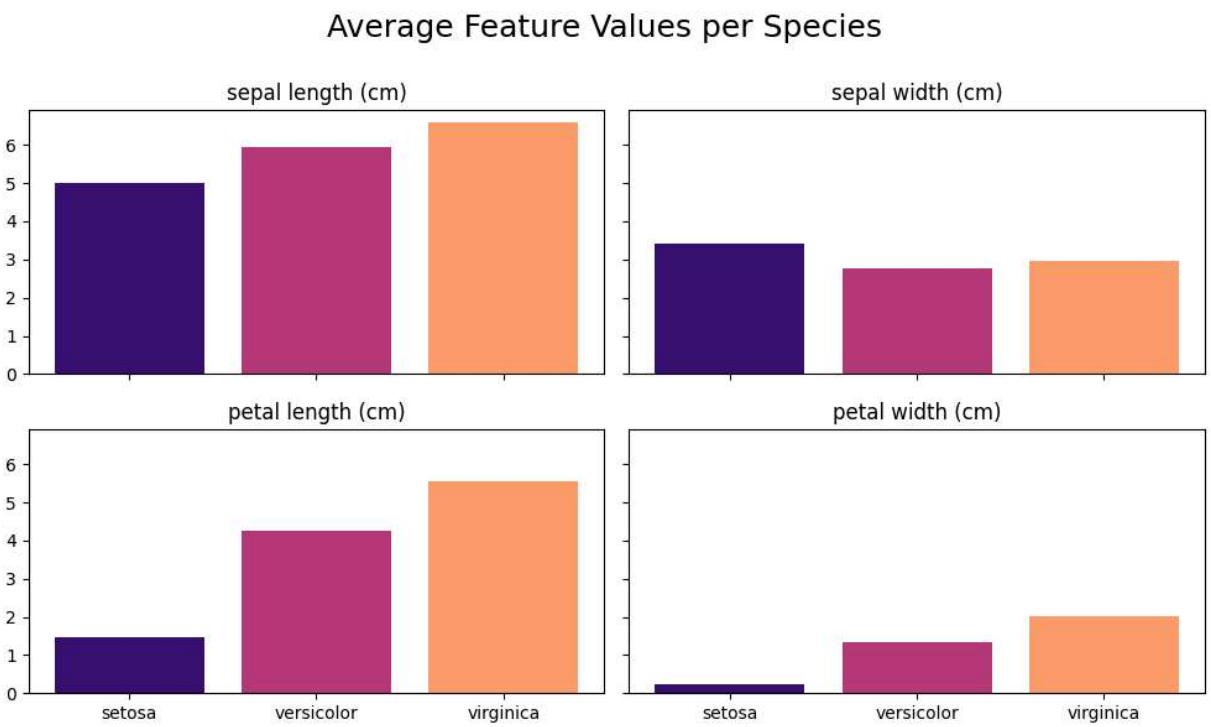


In [ ]:
```
# We calculate the averages of each species: setosa, virginica and versicolor
setosa_means = iris_x[setosa].mean(axis=0)
virginica_means = iris_x[virginica].mean(axis=0)
versicolor_means = iris_x[versicolor].mean(axis=0)
```

In [ ]:
```
# Color map to color the bars, we use the magma class
cmap = plt.cm.magma
color = [cmap(0.2), cmap(0.5), cmap(0.8)]
```

In [ ]:
```
# Create a figure with 2 rows and 2 columns
fig, axes = plt.subplots(2,2, sharex=True, sharey=True, figsize=(10,6))
for i, lab in enumerate(iris_x_labels):
    ax = axes[int(i/2), i%2]
    #Creamos una gráfica de barras, con tres barras en 0, 1 y 2
    # Create a bar graph, with three bars at 0, 1 and 2
    ax.bar([0,1,2], [setosa_means[i], versicolor_means[i], virginica_means[i]], col
    # Place titles on each graph
    ax.set_title(iris_x_labels[i])
    # Reduce the ticks to 3 because there are three bars
    ax.set_xticks([0,1,2])
    # We put labels on the ticks
    ax.set_xticklabels(iris_y_labels)
    # Titulo
    fig.suptitle('Average Feature Values per Species', size = 18)
    # Matplotlib allows you to organize the graphs
    fig.tight_layout()
    # We separate the title from the graph
    fig.subplots_adjust(top=0.85)
```

```
plt.show()
```



A summary of the work developed.

In this Data Analysis and Visualization activity I learned how to effectively employ bubble charts to examine data distribution and identify patterns among data points. These charts provide a clear representation of how data scatters and clusters based on its characteristics.

Moreover, I mastered the creation of violin plots, which enable me to visualize the data's density distribution. The central black line within the violin plot indicates the median, while the quartiles represent the distribution of data points and identify outliers or noisy data.

Furthermore, I delved into the creation of 3D visualizations, which allow me to explore the behavior of data in relation to multiple variables and identify potential correlations. These visualizations provide a comprehensive understanding of the interplay between different data dimensions.

Additionally, I gained proficiency in constructing multi-plots, which enable me to simultaneously visualize multiple charts, each displaying distinct data patterns. This technique facilitates the identification of inconsistencies and anomalies within the data.

Overall, this activity has significantly enhanced my ability to utilize the matplotlib library for effective data visualization. I have gained a comprehensive understanding of various chart types and their applications, enabling me to effectively communicate data insights and uncover hidden patterns within complex datasets.