

1_Basics_students

December 23, 2020

1 HaMLeT

2 Session 1: Basics

2.1 Goals of this session

After this session, you will have an understanding of: - basic python programming - python data-structures - some important python packages: NumPy, matplotlib.pyplot, scikit-learn - working with images - some common terms used in ML - the Iris dataset - PCA

2.1.1 Structure of the Notebook:

You need to perform the tasks and answer the questions in the notebook. For most of the tasks you will find either examples or hints to help you. For this notebook the hints are mostly in the form of commands that you will be required to use. Feel free to search the internet and find out how to use the required command, eg. which inputs need to be provided and how many outputs are to be expected, etc.

Let's start with some basic python programming. Have fun!

Python requires you to `import` the package you want to work on. Let's `import` an important package that motivates us to learn python!

```
[1]: # importing necessary packages
import antigravity
```

2.2 1. Basic NumPy operations

```
[2]: # importing necessary packages
import numpy as np
```

Task: Create a 1D array with 10 elements Hint: you may use any of NumPy built-in array generators, eg. `np.zeros`, `np.ones`, `np.arange`, `np.random`, etc.

```
[3]: # Your code here:
arr1d = np.arange(10)
```

```
arr1d.shape
```

```
[3]: (10,)
```

Task: Reshape the above array (`np.reshape`)

```
[17]: # Your code here:
a = np.reshape(arr1d, (2,5))
a
```

```
[17]: array([[0, 1, 2, 3, 4],
           [5, 6, 7, 8, 9]])
```

Task: Create another 1D array and combine it with the previous array (a) vertically (b) horizontally (eg. using `np.concatenate`, `np.vstack`, etc.)

```
[20]: # Your code here:
b = np.ones((1,5), dtype = int)

# a)
#numpy.concatenate((a1, a2, ...), axis=0, out=None)
arr1d_vcom1 = np.concatenate((a, b))
print("concatenate: ",arr1d_vcom1)

#numpy.vstack(tup)
arr1d_vcom2 = np.vstack((a, b))
print("vstack: ",arr1d_vcom2)

# b)
#numpy.hstack(tup)
arr1d_hcom2 = np.hstack((b, b))
print("hstack: ",arr1d_hcom2)
```

```
concatenate:  [[0 1 2 3 4]
               [5 6 7 8 9]
               [1 1 1 1 1]]
vstack:  [[0 1 2 3 4]
          [5 6 7 8 9]
          [1 1 1 1 1]]
hstack:  [[1 1 1 1 1 1 1 1 1 1]]
```

Task: Get positions where first and second array have the same elements (eg. `np.where`)

```
[33]: # Your code here:
w1 = np.arange(0, 10)
w2 = np.ones(10).astype(int)
w2[-1] = 9
print(" w1:",w1, '\n', "w2:",w2)
```

```
pos = np.where(w1 == w2)
pos
```

```
w1: [0 1 2 3 4 5 6 7 8 9]
w2: [1 1 1 1 1 1 1 1 1 9]
```

```
[33]: (array([1, 9]),)
```

Task: Extract the following from the matrix you created by horizontal concatenation: (a) the first element of the matrix, (b) the first row, (c) the first column, (d) any subset of your choice, eg. the four center elements.

This is called ‘Slicing’

```
[45]: # Your code here:
print("original array:\n",arr1d_vcom2)
# a)
s_a = arr1d_vcom2[0, 0]
# b)
s_b = arr1d_vcom2[:, 0]
# c)
s_c = arr1d_vcom2[0, :]
# d)
s_d = arr1d_vcom2[1, 0:3]

print("\n[0, 0]: ", s_a)
print("[:, 0]: ", s_b)
print("[0, :]: ", s_c)
print("[1, 0:3]: ", s_d)
```

original array:

```
[[0 1 2 3 4]
 [5 6 7 8 9]
 [1 1 1 1 1]]
```

```
[0, 0]: 0
[:, 0]: [0 5 1]
[0, :]: [0 1 2 3 4]
[1, 0:3]: [5 6 7]
```

2.3 2. Python data types

- numbers (eg. int, float32)
- booleans (True, False)
- ‘null’ type (None)
- strings (eg. ‘hello’, ‘world’)
- lists (eg. [8, ‘hello’, 5>7])

- tuples (eg. (8,'hello',5>7))

Hopefully you are already familiar with the common datatypes. We are going to discuss the last two:

2.3.1 List

Python lists allow you to store a sequence of different objects, as in the above example, [int, string, bool].

Note: Lists are created using square brackets [] and comma separated values

Tasks on List 1. Create a list containing a string, a bool, and an int 2. Change any one object of your choice from the list 3. Add new objects to the end of the existing list using (a) **append** (b) **extend**.

```
[50]: # Your code here:
#1
list1 = ['string1', True, 1]
#2
list1[1] = False
#3
list1.append(2)
list1.extend([3])
list1
```

```
[50]: ['string1', False, 1, 2, 3]
```

Question: What is the difference between **append** and **extend**?

Answer:

append: Append object to the end of the list.

extend: Extend list by appending elements from the a list (iterable).

Tasks on List (cont.) 5. Try out the examples using the functions: **pop**, **remove**, and **del**

```
[66]: myList = [0,5,3,4,3]
print("pop output: ",myList.pop(3))
print("new list after pop: ",myList)

myList = [0,5,3,4,3]
print("remove output: ",myList.remove(3))
print("new list after remove: ",myList)

myList = [0,5,3,4,3]
print("del output: ")
del myList[3]
print("new list after del: ",myList)
```

```
pop output: 4
new list after pop: [0, 5, 3, 3]
remove output: None
new list after remove: [0, 5, 4, 3]
del output:
new list after del: [0, 5, 3, 3]
```

Question: What is the difference among functions from task 5?

Answer: pop and del delete with index, while remove handles directly the value of the object. The difference between del and pop is pop returns the deleted value and del returns nothing.

2.3.2 Tuple

Similar to list, tuples can also store a sequence of arbitrary objects. They are constructed using parentheses.

Important: List can be changed after construction (mutable). Tuple cannot be changed (immutable).

Tasks on Tuple 1. Create a tuple object and replace the first object with the int object 0 2. Convert the above tuple to list, change the first object, convert back to tuple

Hint: Use functions `list()` and `tuple()` to convert datatype Tip: You can check datatype using either `type()` or `isinstance(var, dtype)`

```
[69]: # Your code here:
tuple1 = tuple(np.arange(1,10))
print("original tuple:", tuple1)
#can not change tuple's value directly
tu_list1 = list(tuple1)
tu_list1[0] = 0
tuple1 = tuple(tu_list1)
print("changed tuple: ", tuple1)
```

```
original tuple: (1, 2, 3, 4, 5, 6, 7, 8, 9)
changed tuple: (0, 2, 3, 4, 5, 6, 7, 8, 9)
```

2.3.3 Sequence types

- list
- string
- tuple
- NumPy array

Task on sequence types (handy when programming in python!)

- Checking if object is contained within sequence or not.

Understand the following example code and then perform the given tasks.

```
[70]: # Example code:
x = (1,2,3)
y1 = 3 in x
y2 = 5 not in x
print('y1 =',y1)
print('y2 =',y2)
```

```
y1 = True
y2 = True
```

Task: Perform the tasks according to the instructions in the following three cells and answer the question at the end

```
[72]: # Given:
task1 = 'let us learn python'
# Task: Check if 'us' is present in task1 or not
# Your code here:
'us' in task1
```

[72]: True

```
[73]: # Given:
task2 = [10,20,30,40]
# Task: Check if [10,20] is present in task2 or not
# Your code here:
list_test1 = [10, 20]
[10, 20] in task2
```

[73]: False

```
[74]: # Given:
task3 = [True,[2,3],'hello']
# Task: Check if [2,3] is present in task3 or not
# Your code here:
[2,3] in task3
```

[74]: True

Question: Explain the different outputs in task 2 and task 3 above.

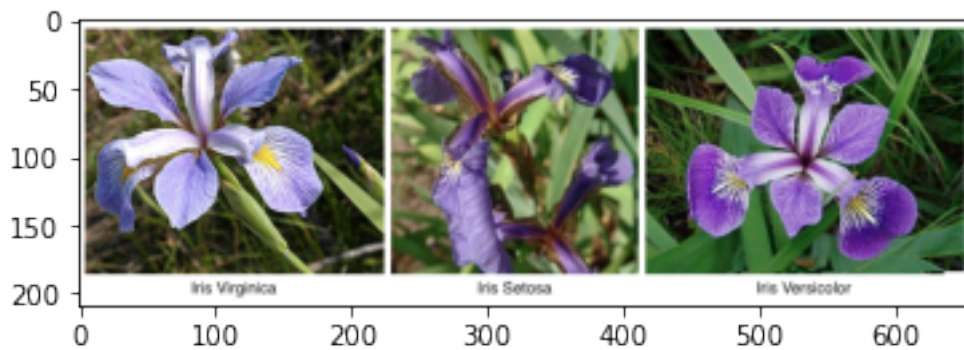
Answer: The list[10, 20] is not an element in task2, as the list task2 did not contain such a list but list [2,3] is an element of task3.

2.4 3. Working with images

We will learn to load, display and manipulate images using the `matplotlib` package.

```
[75]: # importing necessary packages
      %matplotlib inline
      import matplotlib.pyplot as plt
      import cv2
```

```
[76]: # Load image
      irisImg = cv2.imread('irisImage.png')
      irisImg = cv2.cvtColor(irisImg, cv2.COLOR_BGR2RGB) # because cv2 reads images
      ↪ as bgr!
      # Display image
      plt.imshow(irisImg) # display in RGB
      plt.show()
      irisImg.shape
```



```
[76]: (209, 658, 3)
```

Task: Find out the datatype of the image. Hint: You’ve already used this command above!

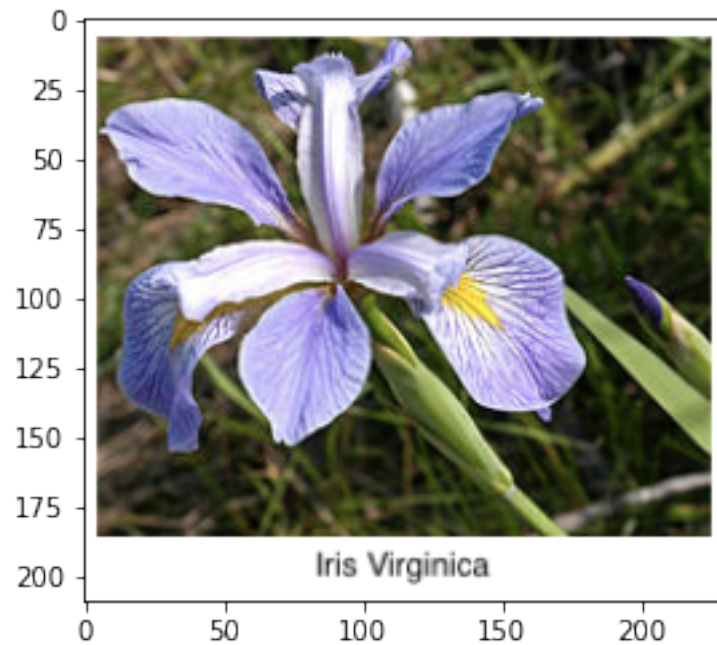
```
[77]: # Your code here:
      irisImg.dtype
```

```
[77]: dtype('uint8')
```

2.4.1 Let’s use the NumPy manipulations we learned to manipulate the image

Task: Extract and display only one of the species Hint: You can do so by the array slicing method you learned above

```
[89]: # Your code here:
      # Please name the image 'iris1'
      iris1 = irisImg[:, :230, :]
      plt.imshow(iris1) # display in RGB
      plt.show()
```



Task: Delete the white border from all the sides of the extracted image

```
[90]: # Your code here:  
# You may use plt.axis('off') to display images without the axis  
  
testImage = irisImg[:, :230, :]  
  
plt.axis("off")  
plt.imshow(testImage)  
plt.show()
```

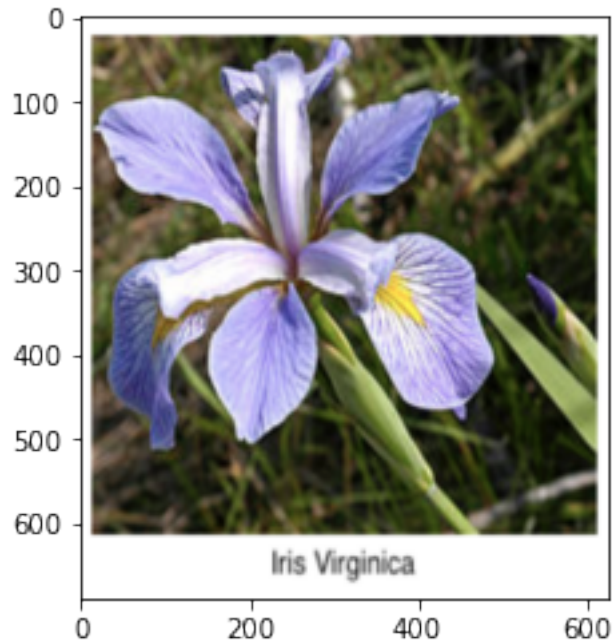



Iris Virginia

2.4.2 Some useful image manipulations

Task: Resize the image by three times (use `cv2.resize`)

```
[91]: # Your code here:  
# Please name the image 'resizedImg'  
w = testImage.shape[0]  
h = testImage.shape[1]  
resizedImg = cv2.resize(testImage, (w*3, h*3))  
  
plt.imshow(resizedImg)  
plt.show()
```



Task: Convert the image to grayscale. This can be done by taking the sum of weighted RGB values as follows: $0.2989R + 0.5870G + 0.1140B$

Hint: Use slicing to extract individual channels

```
[92]: # Your code here:
      # Please name your image 'irisGray'
      irisGray = iris1[:, :, 0]*0.2989 + iris1[:, :, 1]*0.5870 + iris1[:, :, 2]*0.1140
      plt.imshow(irisGray, cmap='gray')
      plt.axis('off')
      plt.show()
```



Iris Virginia

Alternatively, we could use the built-in function `cv2.COLOR_RGB2GRAY`

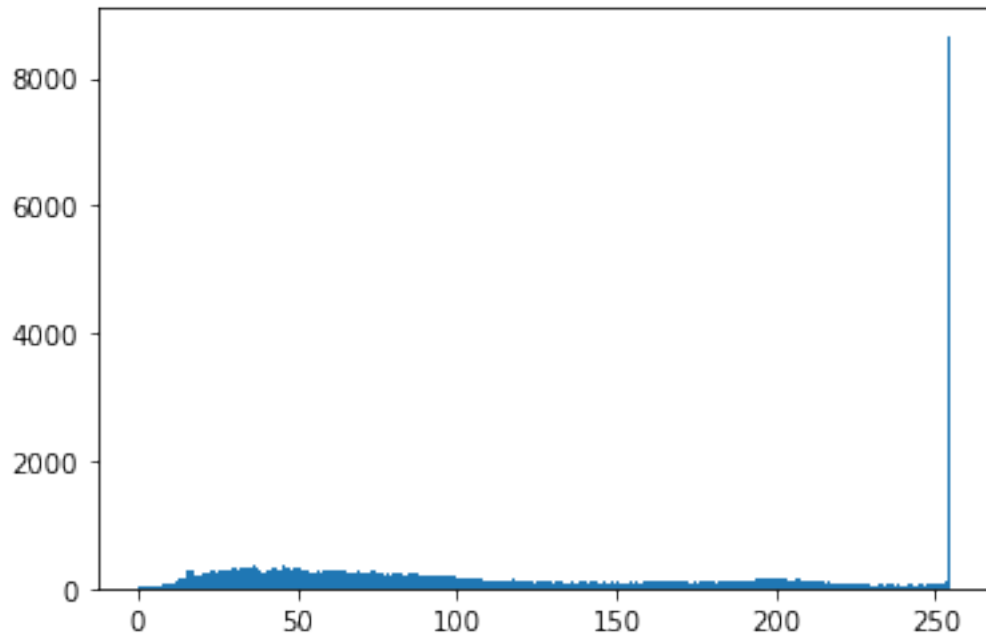
```
[93]: irisGray = cv2.cvtColor(iris1, cv2.COLOR_RGB2GRAY)
plt.imshow(irisGray, cmap='gray')
plt.axis('off')
plt.show()
```



Iris Virginia

Task: Plot the histogram of the gray scale image (use `plt.hist()` with `ravel()`)

```
[95]: # Your code here:  
plt.hist(irisGray.ravel(), bins=255)  
plt.show()
```



2.4.3 Normalization

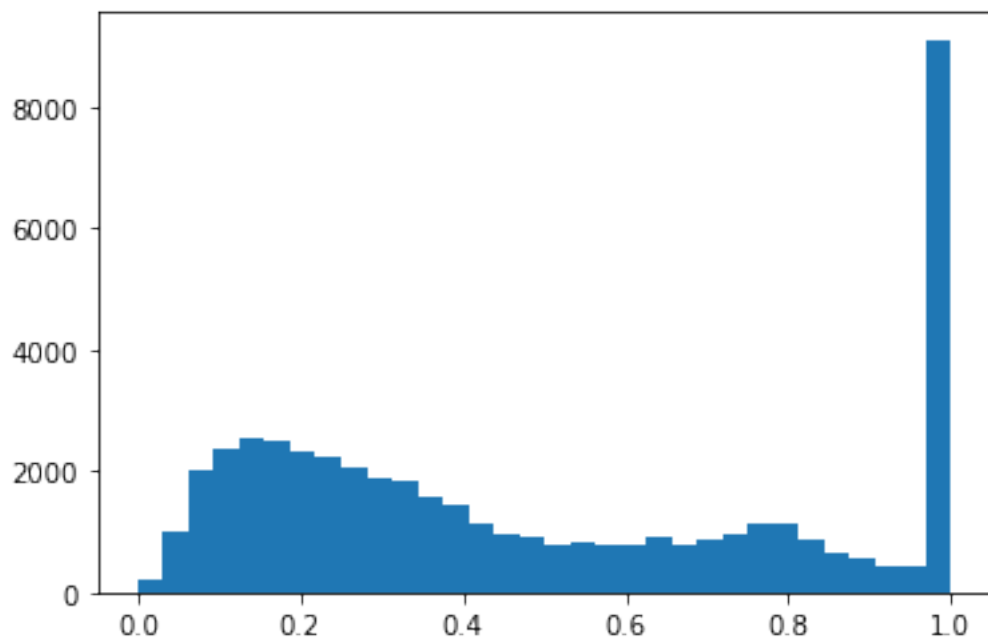
In Machine Learning tasks, normalization is often the first step. It essentially means scaling and centering the data values for faster convergence and improved accuracy. Here we show one method of normalizing by subtracting the minimum value from all data points and then dividing by the range of values.

2.4.4 Normalization method 1:

```
[97]: # Normalization method 1:  
irisNorm1 = (irisGray-np.min(irisGray))/(np.max(irisGray)-np.min(irisGray))  
plt.imshow(irisNorm1, cmap='gray')  
plt.axis('off')  
plt.show()  
# To see the histogram of the normalized image  
plt.hist(irisNorm1.ravel(), bins=32)  
plt.show()
```



Iris Virginia



Note that after normalization the image is rescaled between 0 and 1.

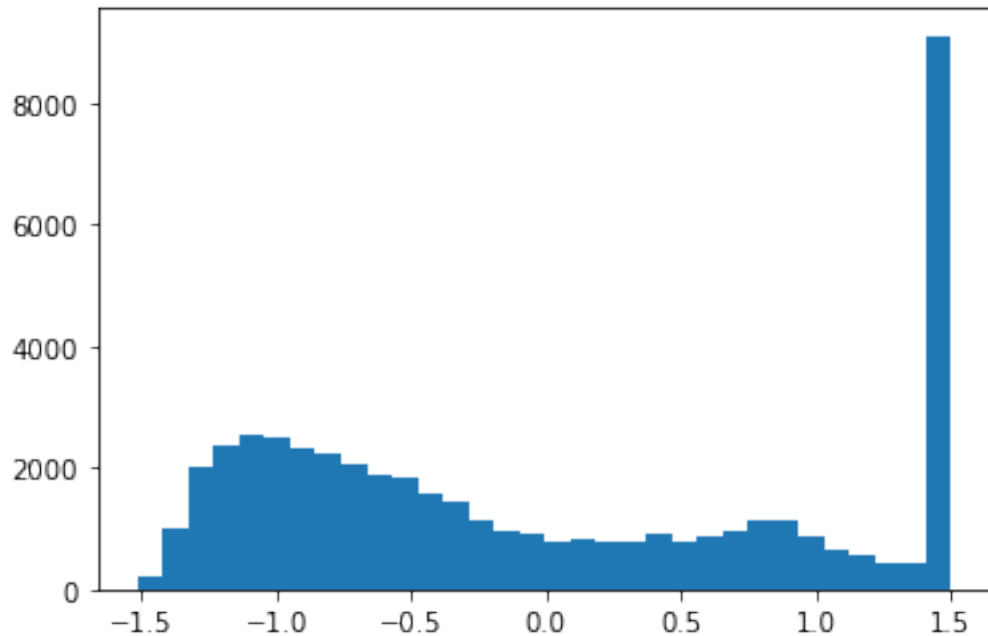
2.4.5 Normalization method 2:

Task: Perform normalization on the grayscale image by subtracting the mean and scaling by the standard deviation

```
[98]: # Normalization method 2:  
# Your code here:  
# Please name your image 'irisNorm2'  
irisNorm2 = (irisGray - np.mean(irisGray)) / np.std(irisGray)  
plt.imshow(irisNorm2, cmap='gray')  
plt.axis('off')  
plt.show()  
# To see the histogram of the normalized image  
plt.hist(irisNorm2.ravel(), 32)  
plt.show()
```



Iris Virginia



Now we're a little comfortable with python and handling different kinds of data. We now move on to the next step!

2.5 4. Machine Learning

In ML, we try to create a model based on the data we have at hand. So the first most important thing is to learn to represent that data such that the computer understands it. In our course, we will use python's Scikit-Learn package.

2.5.1 The dataset

The Iris flower dataset (or Fisher's or Anderson's Iris data set) is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. The aim was to quantify the morphologic variation of Iris flowers of three related species (the three flowers we saw above!).

- The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor).
- Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

We will analyze and visualize the data using **seaborn**. It is a handy python package for visualising statistic data and it offers a plethora of nice plotting tools. It also has the aforementioned Iris dataset included. The Iris dataset in seaborn does not use **numpy** or **cv2**, as we don't work on the

images directly. Instead, we work with ‘features,’ as mentioned above. This information is stored in a so called `pandas dataframe`, another useful tool for statistical analyses.

```
[99]: # We will now load this dataset and display the first few elements.  
# importing necessary packages  
import seaborn as sns  
iris = sns.load_dataset('iris')  
iris.head() #displaying the data (partially)
```

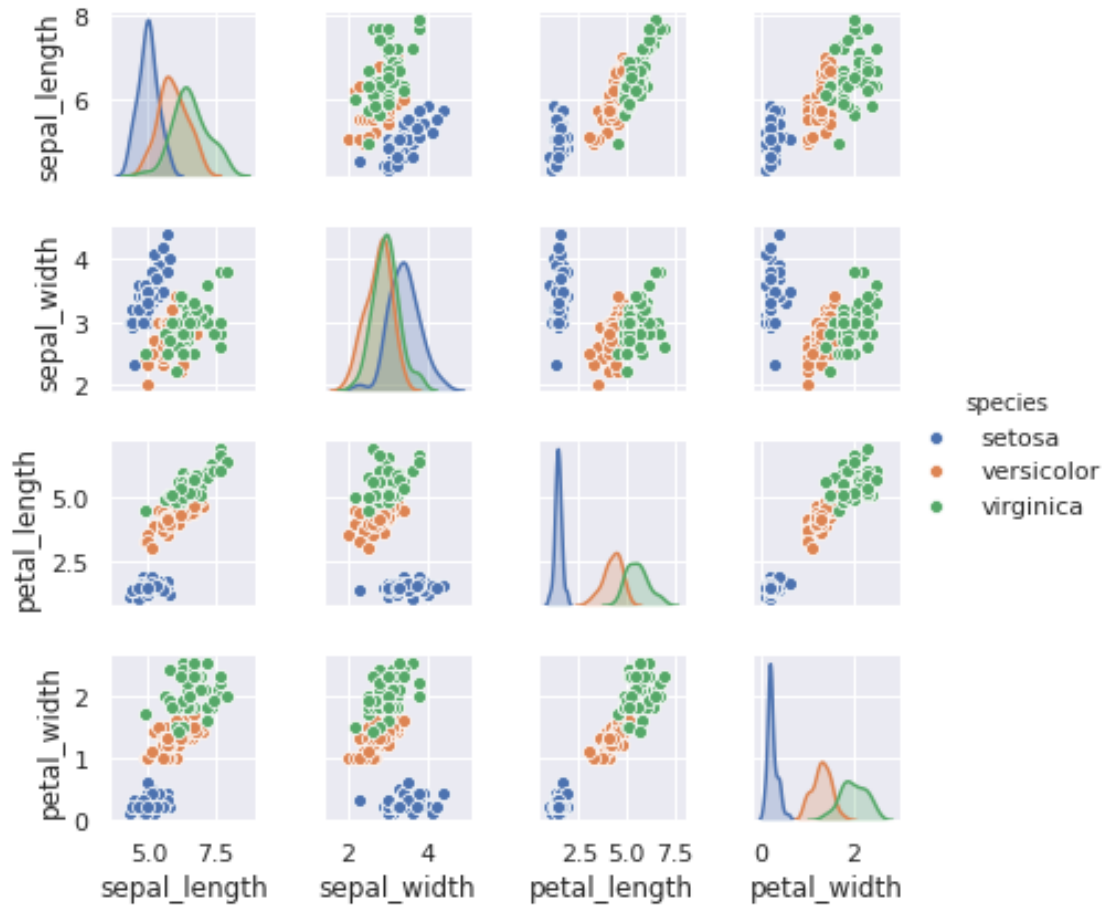
```
[99]:   sepal_length  sepal_width  petal_length  petal_width  species  
0         5.1         3.5         1.4         0.2   setosa  
1         4.9         3.0         1.4         0.2   setosa  
2         4.7         3.2         1.3         0.2   setosa  
3         4.6         3.1         1.5         0.2   setosa  
4         5.0         3.6         1.4         0.2   setosa
```

Note: `pandas dataframe` displays the data in a very intuitive tabular format. Here we see the four features our data has as individual columns. The species on the last column are formed by all the features from the corresponding rows.

2.5.2 Data visualization

```
[100]: %matplotlib inline  
sns.set()  
sns.pairplot(iris, hue='species',height=1.5)
```

```
[100]: <seaborn.axisgrid.PairGrid at 0x7ffaf2fcf090>
```

Note: Here we see a visual pair-wise comparison of all the features

2.5.3 Working with our dataset using sklearn

We will now load the data using the Python ML library `sklearn`. It includes many ML algorithms, some of which we will be learning in this course.

```
[101]: # importing necessary packages
from sklearn import datasets
iris = datasets.load_iris()
```

```
[102]: # Displaying the dataset
print(iris.data)
# Displaying the datatype
print(type(iris.data))
np.shape(iris.data)
```

```
[[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]]
<class 'numpy.ndarray'>

```

```
[102]: (150, 4)
```

Note: 1. The dataset was loaded as a numpy array 2. We do not see the species information here

In `sklearn`, the dataset can be separated into ‘features’ and ‘targets’ as follows (`sklearn` stores species information as targets)

```
[103]: features = iris.data[:, [0, 1, 2, 3]]
       targets = iris.target
       targets
```

```
[103]: 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, 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, 2, 2])
```

Task: Extract all the features of the first flower species Hint: You will find some more information when you inspect the shape of ‘features’ (and ‘targets’)

```
[104]: # Your code here:
       features[targets==0]
```

```
[104]: 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]])

```

Now let's try to visualize the relationship between some of the features

```

[105]: # Plotting the relationship between the Sepal Length and Sepal Width
feature1 = features[:, 0] # Sepal Length
feature2 = features[:, 1] # Sepal Width

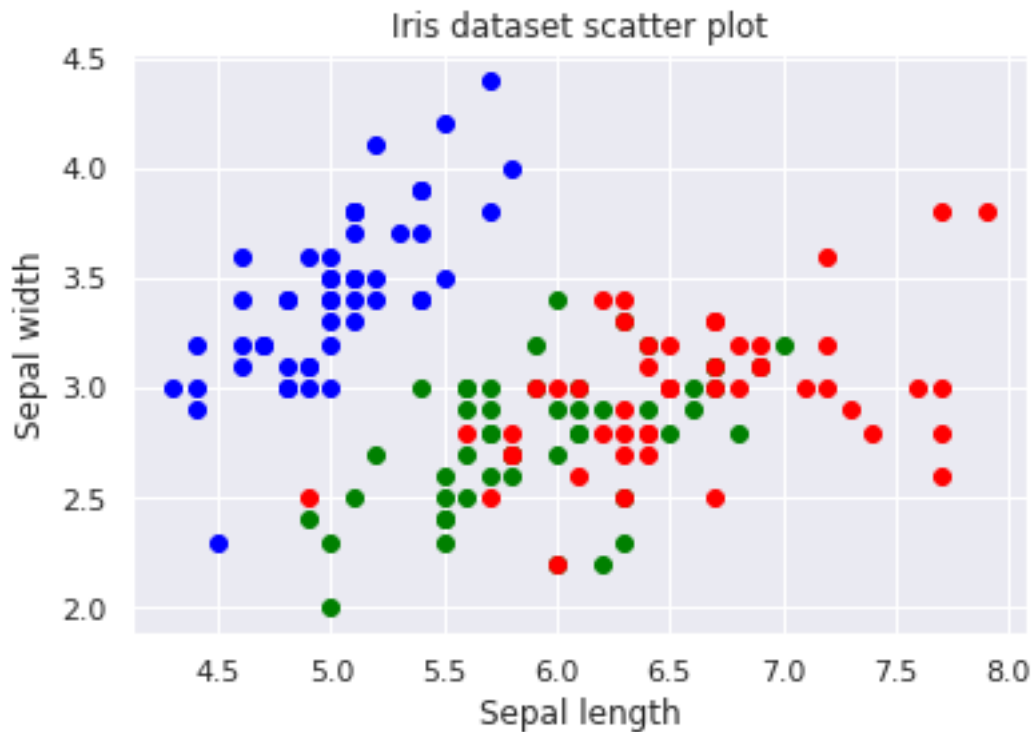
species = ('Iris-setosa', 'Iris-versicolor', 'Iris-virginica')
colors = ('blue', 'green', 'red')

data = [[features[np.where(targets == target)][:, feature] for feature in [0, 1]
        ↪ 1]] for target in range(3)]

for item, color, group in zip(data, colors, species):

```

```
plt.scatter(item[0], item[1], color=color)
plt.title('Iris dataset scatter plot')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```



Task: In a similar manner, plot petal length versus petal width

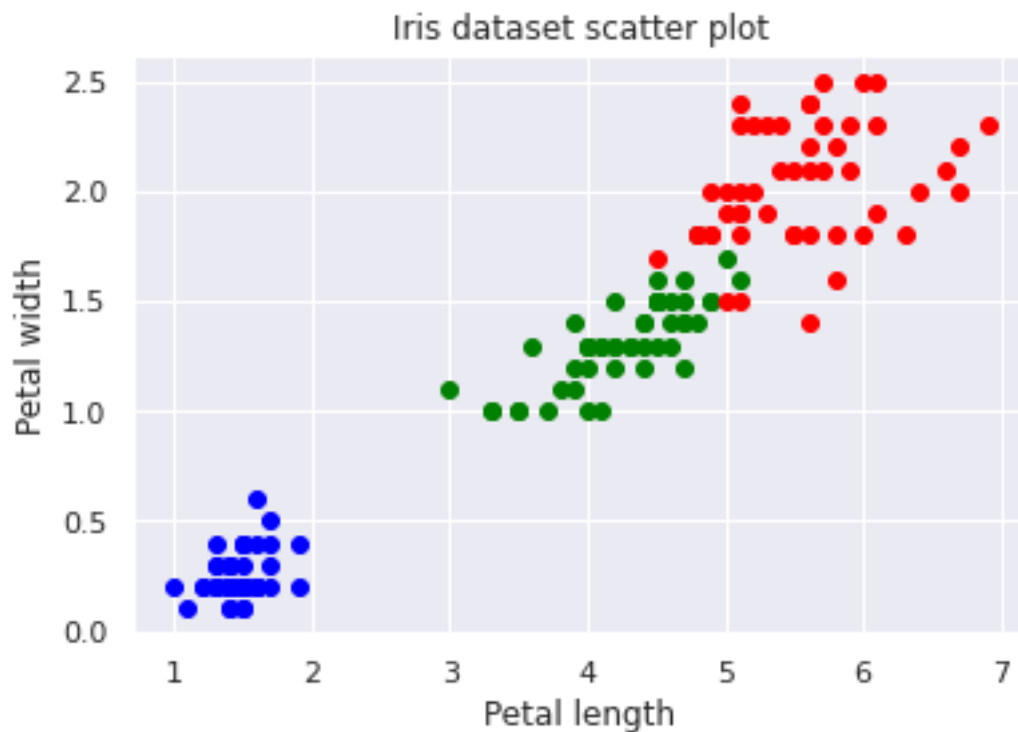
```
[106]: # Your code here:
# Plotting the relationship between the Petal Length and Petal Width
feature1 = features[:, 2] # Petal Length
feature2 = features[:, 3] # Petal Width

species = ('Iris-setosa', 'Iris-versicolor', 'Iris-virginica')
colors = ('blue', 'green', 'red')

data = [[features[np.where(targets == target)][:, feature] for feature in [2, 3]
        for target in range(3)]

for item, color, group in zip(data, colors, species):
    plt.scatter(item[0], item[1], color=color)
    plt.title('Iris dataset scatter plot')
plt.xlabel('Petal length')
```

```
plt.ylabel('Petal width')
plt.show()
```



Question: State your findings, eg. do these two features provide a better species separation?

Answer: Yes, Sepal is a good condition to separate Iris-setosa and other two species, while Petal is a good condition to separate all species.

We have been comparing only two features at a time only because it is easier to visualize. In ML, however, it is quite common to have hundreds of features. In such a case, we might have some features that are redundant by, for eg., being just a linear combination of some other features! Hence, it is common to perform dimensionality reduction to retain only the most representative features from the entire set. One way of doing is by Principal Component Analysis (PCA). Here we try to understand and implement PCA:

2.6 5. Principal Component Analysis (PCA)

2.6.1 PCA Summary:

- Standardize the data.
- Calculate Eigenvectors and Eigenvalues from the covariance matrix.
- Sort eigenvalues in descending order
- Choose the k eigenvectors that correspond to the k largest eigenvalues (k=number of dimensions of the new feature subspace (k d)).

- Construct the projection matrix W from the selected k eigenvectors.
- Transform the original dataset X via W to obtain a k -dimensional feature subspace Y .

```
[107]: X = features
Y = targets
# importing necessary packages for standardizing
from sklearn.preprocessing import StandardScaler
X = StandardScaler().fit_transform(X)
X
X
```

```
[107]: array([[ -9.00681170e-01,  1.01900435e+00, -1.34022653e+00,
        -1.31544430e+00],
       [ -1.14301691e+00, -1.31979479e-01, -1.34022653e+00,
        -1.31544430e+00],
       [ -1.38535265e+00,  3.28414053e-01, -1.39706395e+00,
        -1.31544430e+00],
       [ -1.50652052e+00,  9.82172869e-02, -1.28338910e+00,
        -1.31544430e+00],
       [ -1.02184904e+00,  1.24920112e+00, -1.34022653e+00,
        -1.31544430e+00],
       [ -5.37177559e-01,  1.93979142e+00, -1.16971425e+00,
        -1.05217993e+00],
       [ -1.50652052e+00,  7.88807586e-01, -1.34022653e+00,
        -1.18381211e+00],
       [ -1.02184904e+00,  7.88807586e-01, -1.28338910e+00,
        -1.31544430e+00],
       [ -1.74885626e+00, -3.62176246e-01, -1.34022653e+00,
        -1.31544430e+00],
       [ -1.14301691e+00,  9.82172869e-02, -1.28338910e+00,
        -1.44707648e+00],
       [ -5.37177559e-01,  1.47939788e+00, -1.28338910e+00,
        -1.31544430e+00],
       [ -1.26418478e+00,  7.88807586e-01, -1.22655167e+00,
        -1.31544430e+00],
       [ -1.26418478e+00, -1.31979479e-01, -1.34022653e+00,
        -1.44707648e+00],
       [ -1.87002413e+00, -1.31979479e-01, -1.51073881e+00,
        -1.44707648e+00],
       [ -5.25060772e-02,  2.16998818e+00, -1.45390138e+00,
        -1.31544430e+00],
       [ -1.73673948e-01,  3.09077525e+00, -1.28338910e+00,
        -1.05217993e+00],
       [ -5.37177559e-01,  1.93979142e+00, -1.39706395e+00,
        -1.05217993e+00],
       [ -9.00681170e-01,  1.01900435e+00, -1.34022653e+00,
        -1.18381211e+00],
```

[-1.73673948e-01, 1.70959465e+00, -1.16971425e+00,
 -1.18381211e+00],
 [-9.00681170e-01, 1.70959465e+00, -1.28338910e+00,
 -1.18381211e+00],
 [-5.37177559e-01, 7.88807586e-01, -1.16971425e+00,
 -1.31544430e+00],
 [-9.00681170e-01, 1.47939788e+00, -1.28338910e+00,
 -1.05217993e+00],
 [-1.50652052e+00, 1.24920112e+00, -1.56757623e+00,
 -1.31544430e+00],
 [-9.00681170e-01, 5.58610819e-01, -1.16971425e+00,
 -9.20547742e-01],
 [-1.26418478e+00, 7.88807586e-01, -1.05603939e+00,
 -1.31544430e+00],
 [-1.02184904e+00, -1.31979479e-01, -1.22655167e+00,
 -1.31544430e+00],
 [-1.02184904e+00, 7.88807586e-01, -1.22655167e+00,
 -1.05217993e+00],
 [-7.79513300e-01, 1.01900435e+00, -1.28338910e+00,
 -1.31544430e+00],
 [-7.79513300e-01, 7.88807586e-01, -1.34022653e+00,
 -1.31544430e+00],
 [-1.38535265e+00, 3.28414053e-01, -1.22655167e+00,
 -1.31544430e+00],
 [-1.26418478e+00, 9.82172869e-02, -1.22655167e+00,
 -1.31544430e+00],
 [-5.37177559e-01, 7.88807586e-01, -1.28338910e+00,
 -1.05217993e+00],
 [-7.79513300e-01, 2.40018495e+00, -1.28338910e+00,
 -1.44707648e+00],
 [-4.16009689e-01, 2.63038172e+00, -1.34022653e+00,
 -1.31544430e+00],
 [-1.14301691e+00, 9.82172869e-02, -1.28338910e+00,
 -1.31544430e+00],
 [-1.02184904e+00, 3.28414053e-01, -1.45390138e+00,
 -1.31544430e+00],
 [-4.16009689e-01, 1.01900435e+00, -1.39706395e+00,
 -1.31544430e+00],
 [-1.14301691e+00, 1.24920112e+00, -1.34022653e+00,
 -1.44707648e+00],
 [-1.74885626e+00, -1.31979479e-01, -1.39706395e+00,
 -1.31544430e+00],
 [-9.00681170e-01, 7.88807586e-01, -1.28338910e+00,
 -1.31544430e+00],
 [-1.02184904e+00, 1.01900435e+00, -1.39706395e+00,
 -1.18381211e+00],
 [-1.62768839e+00, -1.74335684e+00, -1.39706395e+00,

-1.18381211e+00],
 [-1.74885626e+00, 3.28414053e-01, -1.39706395e+00,
 -1.31544430e+00],
 [-1.02184904e+00, 1.01900435e+00, -1.22655167e+00,
 -7.88915558e-01],
 [-9.00681170e-01, 1.70959465e+00, -1.05603939e+00,
 -1.05217993e+00],
 [-1.26418478e+00, -1.31979479e-01, -1.34022653e+00,
 -1.18381211e+00],
 [-9.00681170e-01, 1.70959465e+00, -1.22655167e+00,
 -1.31544430e+00],
 [-1.50652052e+00, 3.28414053e-01, -1.34022653e+00,
 -1.31544430e+00],
 [-6.58345429e-01, 1.47939788e+00, -1.28338910e+00,
 -1.31544430e+00],
 [-1.02184904e+00, 5.58610819e-01, -1.34022653e+00,
 -1.31544430e+00],
 [1.40150837e+00, 3.28414053e-01, 5.35408562e-01,
 2.64141916e-01],
 [6.74501145e-01, 3.28414053e-01, 4.21733708e-01,
 3.95774101e-01],
 [1.28034050e+00, 9.82172869e-02, 6.49083415e-01,
 3.95774101e-01],
 [-4.16009689e-01, -1.74335684e+00, 1.37546573e-01,
 1.32509732e-01],
 [7.95669016e-01, -5.92373012e-01, 4.78571135e-01,
 3.95774101e-01],
 [-1.73673948e-01, -5.92373012e-01, 4.21733708e-01,
 1.32509732e-01],
 [5.53333275e-01, 5.58610819e-01, 5.35408562e-01,
 5.27406285e-01],
 [-1.14301691e+00, -1.51316008e+00, -2.60315415e-01,
 -2.62386821e-01],
 [9.16836886e-01, -3.62176246e-01, 4.78571135e-01,
 1.32509732e-01],
 [-7.79513300e-01, -8.22569778e-01, 8.07091462e-02,
 2.64141916e-01],
 [-1.02184904e+00, -2.43394714e+00, -1.46640561e-01,
 -2.62386821e-01],
 [6.86617933e-02, -1.31979479e-01, 2.51221427e-01,
 3.95774101e-01],
 [1.89829664e-01, -1.97355361e+00, 1.37546573e-01,
 -2.62386821e-01],
 [3.10997534e-01, -3.62176246e-01, 5.35408562e-01,
 2.64141916e-01],
 [-2.94841818e-01, -3.62176246e-01, -8.98031345e-02,
 1.32509732e-01],

[1.03800476e+00, 9.82172869e-02, 3.64896281e-01,
 2.64141916e-01],
 [-2.94841818e-01, -1.31979479e-01, 4.21733708e-01,
 3.95774101e-01],
 [-5.25060772e-02, -8.22569778e-01, 1.94384000e-01,
 -2.62386821e-01],
 [4.32165405e-01, -1.97355361e+00, 4.21733708e-01,
 3.95774101e-01],
 [-2.94841818e-01, -1.28296331e+00, 8.07091462e-02,
 -1.30754636e-01],
 [6.86617933e-02, 3.28414053e-01, 5.92245988e-01,
 7.90670654e-01],
 [3.10997534e-01, -5.92373012e-01, 1.37546573e-01,
 1.32509732e-01],
 [5.53333275e-01, -1.28296331e+00, 6.49083415e-01,
 3.95774101e-01],
 [3.10997534e-01, -5.92373012e-01, 5.35408562e-01,
 8.77547895e-04],
 [6.74501145e-01, -3.62176246e-01, 3.08058854e-01,
 1.32509732e-01],
 [9.16836886e-01, -1.31979479e-01, 3.64896281e-01,
 2.64141916e-01],
 [1.15917263e+00, -5.92373012e-01, 5.92245988e-01,
 2.64141916e-01],
 [1.03800476e+00, -1.31979479e-01, 7.05920842e-01,
 6.59038469e-01],
 [1.89829664e-01, -3.62176246e-01, 4.21733708e-01,
 3.95774101e-01],
 [-1.73673948e-01, -1.05276654e+00, -1.46640561e-01,
 -2.62386821e-01],
 [-4.16009689e-01, -1.51316008e+00, 2.38717193e-02,
 -1.30754636e-01],
 [-4.16009689e-01, -1.51316008e+00, -3.29657076e-02,
 -2.62386821e-01],
 [-5.25060772e-02, -8.22569778e-01, 8.07091462e-02,
 8.77547895e-04],
 [1.89829664e-01, -8.22569778e-01, 7.62758269e-01,
 5.27406285e-01],
 [-5.37177559e-01, -1.31979479e-01, 4.21733708e-01,
 3.95774101e-01],
 [1.89829664e-01, 7.88807586e-01, 4.21733708e-01,
 5.27406285e-01],
 [1.03800476e+00, 9.82172869e-02, 5.35408562e-01,
 3.95774101e-01],
 [5.53333275e-01, -1.74335684e+00, 3.64896281e-01,
 1.32509732e-01],
 [-2.94841818e-01, -1.31979479e-01, 1.94384000e-01,

1.32509732e-01],
 [-4.16009689e-01, -1.28296331e+00, 1.37546573e-01,
 1.32509732e-01],
 [-4.16009689e-01, -1.05276654e+00, 3.64896281e-01,
 8.77547895e-04],
 [3.10997534e-01, -1.31979479e-01, 4.78571135e-01,
 2.64141916e-01],
 [-5.25060772e-02, -1.05276654e+00, 1.37546573e-01,
 8.77547895e-04],
 [-1.02184904e+00, -1.74335684e+00, -2.60315415e-01,
 -2.62386821e-01],
 [-2.94841818e-01, -8.22569778e-01, 2.51221427e-01,
 1.32509732e-01],
 [-1.73673948e-01, -1.31979479e-01, 2.51221427e-01,
 8.77547895e-04],
 [-1.73673948e-01, -3.62176246e-01, 2.51221427e-01,
 1.32509732e-01],
 [4.32165405e-01, -3.62176246e-01, 3.08058854e-01,
 1.32509732e-01],
 [-9.00681170e-01, -1.28296331e+00, -4.30827696e-01,
 -1.30754636e-01],
 [-1.73673948e-01, -5.92373012e-01, 1.94384000e-01,
 1.32509732e-01],
 [5.53333275e-01, 5.58610819e-01, 1.27429511e+00,
 1.71209594e+00],
 [-5.25060772e-02, -8.22569778e-01, 7.62758269e-01,
 9.22302838e-01],
 [1.52267624e+00, -1.31979479e-01, 1.21745768e+00,
 1.18556721e+00],
 [5.53333275e-01, -3.62176246e-01, 1.04694540e+00,
 7.90670654e-01],
 [7.95669016e-01, -1.31979479e-01, 1.16062026e+00,
 1.31719939e+00],
 [2.12851559e+00, -1.31979479e-01, 1.61531967e+00,
 1.18556721e+00],
 [-1.14301691e+00, -1.28296331e+00, 4.21733708e-01,
 6.59038469e-01],
 [1.76501198e+00, -3.62176246e-01, 1.44480739e+00,
 7.90670654e-01],
 [1.03800476e+00, -1.28296331e+00, 1.16062026e+00,
 7.90670654e-01],
 [1.64384411e+00, 1.24920112e+00, 1.33113254e+00,
 1.71209594e+00],
 [7.95669016e-01, 3.28414053e-01, 7.62758269e-01,
 1.05393502e+00],
 [6.74501145e-01, -8.22569778e-01, 8.76433123e-01,
 9.22302838e-01],

[1.15917263e+00, -1.31979479e-01, 9.90107977e-01,
 1.18556721e+00],
 [-1.73673948e-01, -1.28296331e+00, 7.05920842e-01,
 1.05393502e+00],
 [-5.25060772e-02, -5.92373012e-01, 7.62758269e-01,
 1.58046376e+00],
 [6.74501145e-01, 3.28414053e-01, 8.76433123e-01,
 1.44883158e+00],
 [7.95669016e-01, -1.31979479e-01, 9.90107977e-01,
 7.90670654e-01],
 [2.24968346e+00, 1.70959465e+00, 1.67215710e+00,
 1.31719939e+00],
 [2.24968346e+00, -1.05276654e+00, 1.78583195e+00,
 1.44883158e+00],
 [1.89829664e-01, -1.97355361e+00, 7.05920842e-01,
 3.95774101e-01],
 [1.28034050e+00, 3.28414053e-01, 1.10378283e+00,
 1.44883158e+00],
 [-2.94841818e-01, -5.92373012e-01, 6.49083415e-01,
 1.05393502e+00],
 [2.24968346e+00, -5.92373012e-01, 1.67215710e+00,
 1.05393502e+00],
 [5.53333275e-01, -8.22569778e-01, 6.49083415e-01,
 7.90670654e-01],
 [1.03800476e+00, 5.58610819e-01, 1.10378283e+00,
 1.18556721e+00],
 [1.64384411e+00, 3.28414053e-01, 1.27429511e+00,
 7.90670654e-01],
 [4.32165405e-01, -5.92373012e-01, 5.92245988e-01,
 7.90670654e-01],
 [3.10997534e-01, -1.31979479e-01, 6.49083415e-01,
 7.90670654e-01],
 [6.74501145e-01, -5.92373012e-01, 1.04694540e+00,
 1.18556721e+00],
 [1.64384411e+00, -1.31979479e-01, 1.16062026e+00,
 5.27406285e-01],
 [1.88617985e+00, -5.92373012e-01, 1.33113254e+00,
 9.22302838e-01],
 [2.49201920e+00, 1.70959465e+00, 1.50164482e+00,
 1.05393502e+00],
 [6.74501145e-01, -5.92373012e-01, 1.04694540e+00,
 1.31719939e+00],
 [5.53333275e-01, -5.92373012e-01, 7.62758269e-01,
 3.95774101e-01],
 [3.10997534e-01, -1.05276654e+00, 1.04694540e+00,
 2.64141916e-01],
 [2.24968346e+00, -1.31979479e-01, 1.33113254e+00,

```

1.44883158e+00],
[ 5.53333275e-01,  7.88807586e-01,  1.04694540e+00,
 1.58046376e+00],
[ 6.74501145e-01,  9.82172869e-02,  9.90107977e-01,
 7.90670654e-01],
[ 1.89829664e-01, -1.31979479e-01,  5.92245988e-01,
 7.90670654e-01],
[ 1.28034050e+00,  9.82172869e-02,  9.33270550e-01,
 1.18556721e+00],
[ 1.03800476e+00,  9.82172869e-02,  1.04694540e+00,
 1.58046376e+00],
[ 1.28034050e+00,  9.82172869e-02,  7.62758269e-01,
 1.44883158e+00],
[-5.25060772e-02, -8.22569778e-01,  7.62758269e-01,
 9.22302838e-01],
[ 1.15917263e+00,  3.28414053e-01,  1.21745768e+00,
 1.44883158e+00],
[ 1.03800476e+00,  5.58610819e-01,  1.10378283e+00,
 1.71209594e+00],
[ 1.03800476e+00, -1.31979479e-01,  8.19595696e-01,
 1.44883158e+00],
[ 5.53333275e-01, -1.28296331e+00,  7.05920842e-01,
 9.22302838e-01],
[ 7.95669016e-01, -1.31979479e-01,  8.19595696e-01,
 1.05393502e+00],
[ 4.32165405e-01,  7.88807586e-01,  9.33270550e-01,
 1.44883158e+00],
[ 6.86617933e-02, -1.31979479e-01,  7.62758269e-01,
 7.90670654e-01]])

```

Task: Calculate the covariance matrix of the **transposed** feature matrix (use NumPy's cov function)

```

[108]: # Your code here:
# Please name the matrix as 'cov_mat'
# Remember to transpose the matrix before computing the covariance
cov_mat = np.cov(X.T)
cov_mat

```

```

[108]: array([[ 1.00671141, -0.11835884,  0.87760447,  0.82343066],
 [-0.11835884,  1.00671141, -0.43131554, -0.36858315],
 [ 0.87760447, -0.43131554,  1.00671141,  0.96932762],
 [ 0.82343066, -0.36858315,  0.96932762,  1.00671141]])

```

Task: Perform eigenvalue decomposition on cov_mat (use NumPy's linear algebra function eig)

```

[109]: # Your code here:
# Please name the matrix as 'cov_mat'
# Please call your variables eig_vecs and eig_vals

```

```
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
eig_vals, eig_vecs
```

```
[109]: (array([2.93808505, 0.9201649 , 0.14774182, 0.02085386]),
       array([[ 0.52106591, -0.37741762, -0.71956635,  0.26128628],
              [-0.26934744, -0.92329566,  0.24438178, -0.12350962],
              [ 0.5804131 , -0.02449161,  0.14212637, -0.80144925],
              [ 0.56485654, -0.06694199,  0.63427274,  0.52359713]]))
```

Question: How many PC values should we retain?

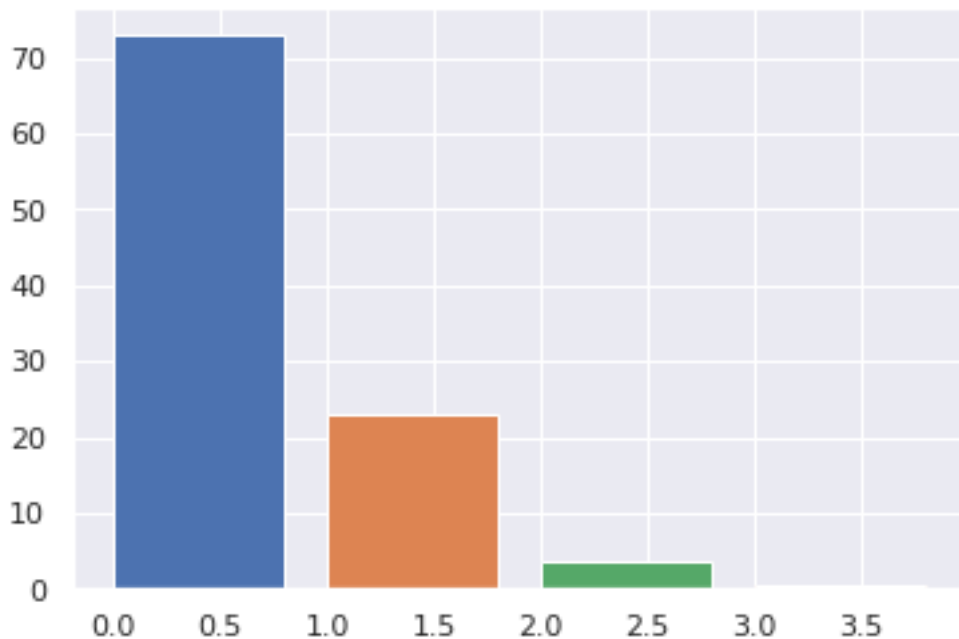
Answer: I don't know either ;)

But we can both find it out by calculating the 'Explained Variance.'

```
[110]: total = sum(eig_vals)
       var_exp = [(i/total)*100 for i in sorted(eig_vals, reverse=True)]

       for pos, data in enumerate(var_exp):
           plt.bar(pos, data, align='edge')
       plt.show
```

```
[110]: <function matplotlib.pyplot.show(*args, **kw)>
```



This plot shows that around 73% of the variance is captured by the first PC and almost 23% by the second. We can safely ignore the third and the fourth component without losing much information. In principle, we are reducing the 4D feature space to a 2D feature subspace, by choosing the “top

2” eigenvectors with the highest eigenvalues to construct a $d \times k$ -dimensional eigenvector matrix W . So let’s construct W .

```
[111]: # First, make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eig_pairs.sort()
eig_pairs.reverse()

matrix_w = np.hstack((eig_pairs[0][1].reshape(4,1),
                      eig_pairs[1][1].reshape(4,1)))

print('Matrix W:\n', matrix_w)
```

Matrix W:

```
[[ 0.52106591 -0.37741762]
 [-0.26934744 -0.92329566]
 [ 0.5804131  -0.02449161]
 [ 0.56485654 -0.06694199]]
```

Lastly, we will use the 4×2 -dimensional projection matrix W to transform our samples onto the new subspace via the equation $Y=X \times W$, where Y is a 150×2 matrix of our transformed samples.

```
[112]: #np.cross(X,matrix_w)
Y = X.dot(matrix_w)
Y
```

```
[112]: array([[ -2.26470281, -0.4800266 ],
               [ -2.08096115,  0.67413356],
               [ -2.36422905,  0.34190802],
               [ -2.29938422,  0.59739451],
               [ -2.38984217, -0.64683538],
               [ -2.07563095, -1.48917752],
               [ -2.44402884, -0.0476442 ],
               [ -2.23284716, -0.22314807],
               [ -2.33464048,  1.11532768],
               [ -2.18432817,  0.46901356],
               [ -2.1663101 , -1.04369065],
               [ -2.32613087, -0.13307834],
               [ -2.2184509 ,  0.72867617],
               [ -2.6331007 ,  0.96150673],
               [ -2.1987406 , -1.86005711],
               [ -2.26221453, -2.68628449],
               [ -2.2075877 , -1.48360936],
               [ -2.19034951, -0.48883832],
               [ -1.898572  , -1.40501879],
               [ -2.34336905, -1.12784938],
```

[-1.914323 , -0.40885571],
 [-2.20701284, -0.92412143],
 [-2.7743447 , -0.45834367],
 [-1.81866953, -0.08555853],
 [-2.22716331, -0.13725446],
 [-1.95184633, 0.62561859],
 [-2.05115137, -0.24216355],
 [-2.16857717, -0.52714953],
 [-2.13956345, -0.31321781],
 [-2.26526149, 0.3377319],
 [-2.14012214, 0.50454069],
 [-1.83159477, -0.42369507],
 [-2.61494794, -1.79357586],
 [-2.44617739, -2.15072788],
 [-2.10997488, 0.46020184],
 [-2.2078089 , 0.2061074],
 [-2.04514621, -0.66155811],
 [-2.52733191, -0.59229277],
 [-2.42963258, 0.90418004],
 [-2.16971071, -0.26887896],
 [-2.28647514, -0.44171539],
 [-1.85812246, 2.33741516],
 [-2.5536384 , 0.47910069],
 [-1.96444768, -0.47232667],
 [-2.13705901, -1.14222926],
 [-2.0697443 , 0.71105273],
 [-2.38473317, -1.1204297],
 [-2.39437631, 0.38624687],
 [-2.22944655, -0.99795976],
 [-2.20383344, -0.00921636],
 [1.10178118, -0.86297242],
 [0.73133743, -0.59461473],
 [1.24097932, -0.61629765],
 [0.40748306, 1.75440399],
 [1.0754747 , 0.20842105],
 [0.38868734, 0.59328364],
 [0.74652974, -0.77301931],
 [-0.48732274, 1.85242909],
 [0.92790164, -0.03222608],
 [0.01142619, 1.03401828],
 [-0.11019628, 2.65407282],
 [0.44069345, 0.06329519],
 [0.56210831, 1.76472438],
 [0.71956189, 0.18622461],
 [-0.0333547 , 0.43900321],
 [0.87540719, -0.50906396],
 [0.35025167, 0.19631173],

[0.15881005, 0.79209574],
 [1.22509363, 1.6222438],
 [0.1649179 , 1.30260923],
 [0.73768265, -0.39657156],
 [0.47628719, 0.41732028],
 [1.2341781 , 0.93332573],
 [0.6328582 , 0.41638772],
 [0.70266118, 0.06341182],
 [0.87427365, -0.25079339],
 [1.25650912, 0.07725602],
 [1.35840512, -0.33131168],
 [0.66480037, 0.22592785],
 [-0.04025861, 1.05871855],
 [0.13079518, 1.56227183],
 [0.02345269, 1.57247559],
 [0.24153827, 0.77725638],
 [1.06109461, 0.63384324],
 [0.22397877, 0.28777351],
 [0.42913912, -0.84558224],
 [1.04872805, -0.5220518],
 [1.04453138, 1.38298872],
 [0.06958832, 0.21950333],
 [0.28347724, 1.32932464],
 [0.27907778, 1.12002852],
 [0.62456979, -0.02492303],
 [0.33653037, 0.98840402],
 [-0.36218338, 2.01923787],
 [0.28858624, 0.85573032],
 [0.09136066, 0.18119213],
 [0.22771687, 0.38492008],
 [0.57638829, 0.1548736],
 [-0.44766702, 1.54379203],
 [0.25673059, 0.5988518],
 [1.84456887, -0.87042131],
 [1.15788161, 0.69886986],
 [2.20526679, -0.56201048],
 [1.44015066, 0.04698759],
 [1.86781222, -0.29504482],
 [2.75187334, -0.8004092],
 [0.36701769, 1.56150289],
 [2.30243944, -0.42006558],
 [2.00668647, 0.71143865],
 [2.25977735, -1.92101038],
 [1.36417549, -0.69275645],
 [1.60267867, 0.42170045],
 [1.8839007 , -0.41924965],
 [1.2601151 , 1.16226042],

```
[ 1.4676452 ,  0.44227159],
[ 1.59007732, -0.67624481],
[ 1.47143146, -0.25562182],
[ 2.42632899, -2.55666125],
[ 3.31069558, -0.01778095],
[ 1.26376667,  1.70674538],
[ 2.0377163 , -0.91046741],
[ 0.97798073,  0.57176432],
[ 2.89765149, -0.41364106],
[ 1.33323218,  0.48181122],
[ 1.7007339 , -1.01392187],
[ 1.95432671, -1.0077776 ],
[ 1.17510363,  0.31639447],
[ 1.02095055, -0.06434603],
[ 1.78834992,  0.18736121],
[ 1.86364755, -0.56229073],
[ 2.43595373, -0.25928443],
[ 2.30492772, -2.62632347],
[ 1.86270322,  0.17854949],
[ 1.11414774,  0.29292262],
[ 1.2024733 ,  0.81131527],
[ 2.79877045, -0.85680333],
[ 1.57625591, -1.06858111],
[ 1.3462921 , -0.42243061],
[ 0.92482492, -0.0172231 ],
[ 1.85204505, -0.67612817],
[ 2.01481043, -0.61388564],
[ 1.90178409, -0.68957549],
[ 1.15788161,  0.69886986],
[ 2.04055823, -0.8675206 ],
[ 1.9981471 , -1.04916875],
[ 1.87050329, -0.38696608],
[ 1.56458048,  0.89668681],
[ 1.5211705 , -0.26906914],
[ 1.37278779, -1.01125442],
[ 0.96065603,  0.02433167]])
```

```
[177]: species = ('Iris-setosa', 'Iris-versicolor', 'Iris-virginica')
       colors = ('blue','green','red')

       # Plotting the relationship between PCA1 and PCA2
       feature1 = Y[:, 0]
       feature2 = Y[:, 1]

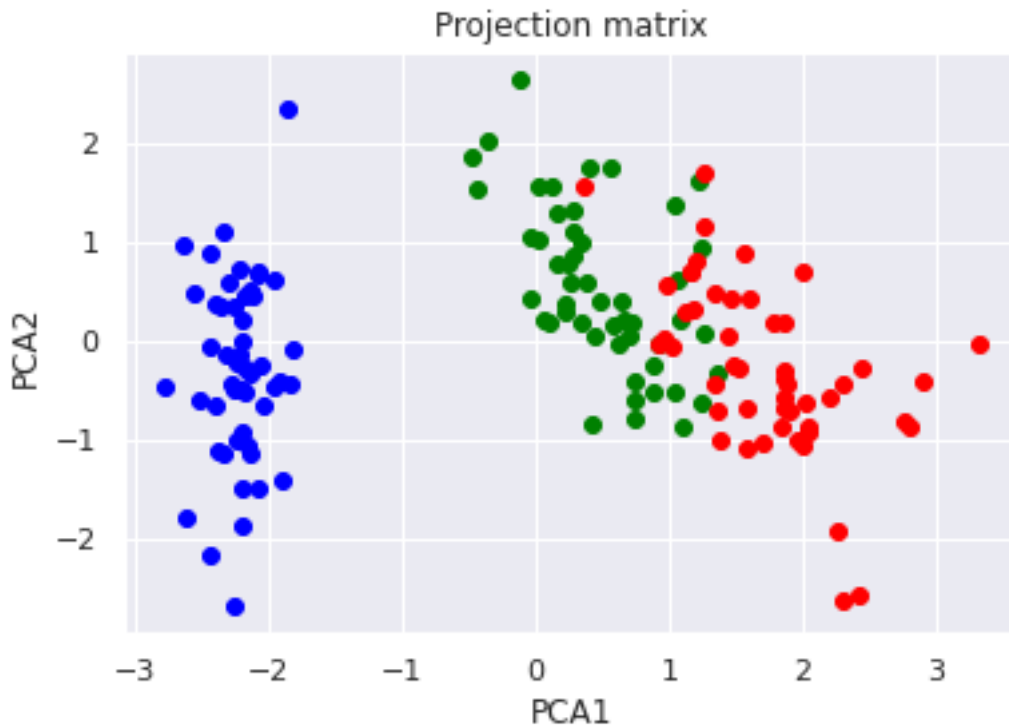
       species = ('Iris-setosa', 'Iris-versicolor', 'Iris-virginica')
       colors = ('blue','green','red')
```

```

data = [[Y[np.where(targets == target)][:, feature] for feature in [0, 1]] for
    target in range(3)]

for item, color, group in zip(data, colors, species):
    plt.scatter(item[0], item[1], color=color)
    plt.title('Projection matrix')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.show()

```



2.6.2 PCA using scikit-learn

```

[113]: # importing necessary packages
from sklearn.decomposition import PCA as sklearnPCA

```

Task: Perform PCA using the built-in function sklearnPCA

```

[114]: # Your code here:
#n_components == min(n_samples, n_features) - 1
pca = sklearnPCA('mle')
Y = pca.fit_transform(X)
Y

```

```

[114]: array([[ -2.26470281,  0.4800266 , -0.12770602],
               [ -2.08096115, -0.67413356, -0.23460885],
               [ -2.36422905, -0.34190802,  0.04420148],
               [ -2.29938422, -0.59739451,  0.09129011],
               [ -2.38984217,  0.64683538,  0.0157382 ],
               [ -2.07563095,  1.48917752,  0.02696829],
               [ -2.44402884,  0.0476442 ,  0.3354704 ],
               [ -2.23284716,  0.22314807, -0.0886955 ],
               [ -2.33464048, -1.11532768,  0.14507686],
               [ -2.18432817, -0.46901356, -0.25376557],
               [ -2.1663101 ,  1.04369065, -0.2686811 ],
               [ -2.32613087,  0.13307834,  0.09375924],
               [ -2.2184509 , -0.72867617, -0.23091124],
               [ -2.6331007 , -0.96150673,  0.18079608],
               [ -2.1987406 ,  1.86005711, -0.472901  ],
               [ -2.26221453,  2.68628449,  0.03052661],
               [ -2.2075877 ,  1.48360936, -0.00534409],
               [ -2.19034951,  0.48883832, -0.04421532],
               [ -1.898572  ,  1.40501879, -0.37434327],
               [ -2.34336905,  1.12784938,  0.13263047],
               [ -1.914323  ,  0.40885571, -0.42129259],
               [ -2.20701284,  0.92412143,  0.15986528],
               [ -2.7743447 ,  0.45834367,  0.3321791 ],
               [ -1.81866953,  0.08555853,  0.0344886 ],
               [ -2.22716331,  0.13725446,  0.11799354],
               [ -1.95184633, -0.62561859, -0.30564098],
               [ -2.05115137,  0.24216355,  0.08636401],
               [ -2.16857717,  0.52714953, -0.20681625],
               [ -2.13956345,  0.31321781, -0.27115024],
               [ -2.26526149, -0.3377319 ,  0.06843578],
               [ -2.14012214, -0.50454069, -0.07500844],
               [ -1.83159477,  0.42369507, -0.27046738],
               [ -2.61494794,  1.79357586,  0.04722842],
               [ -2.44617739,  2.15072788, -0.08266805],
               [ -2.10997488, -0.46020184, -0.17027486],
               [ -2.2078089 , -0.2061074 , -0.22544158],
               [ -2.04514621,  0.66155811, -0.48453741],
               [ -2.52733191,  0.59229277,  0.01943581],
               [ -2.42963258, -0.90418004,  0.19325466],
               [ -2.16971071,  0.26887896, -0.17588382],
               [ -2.28647514,  0.44171539,  0.03489491],
               [ -1.85812246, -2.33741516, -0.20423422],
               [ -2.5536384 , -0.47910069,  0.30576645],
               [ -1.96444768,  0.47232667,  0.30960132],
               [ -2.13705901,  1.14222926,  0.24843356],
               [ -2.0697443 , -0.71105273, -0.06392983],
               [ -2.38473317,  1.1204297 ,  0.05721786],

```

[-2.39437631, -0.38624687, 0.1394679],
 [-2.22944655, 0.99795976, -0.18149278],
 [-2.20383344, 0.00921636, -0.15302949],
 [1.10178118, 0.86297242, -0.68458616],
 [0.73133743, 0.59461473, -0.09412172],
 [1.24097932, 0.61629765, -0.55400684],
 [0.40748306, -1.75440399, -0.02310177],
 [1.0754747 , -0.20842105, -0.39825552],
 [0.38868734, -0.59328364, 0.12419155],
 [0.74652974, 0.77301931, 0.1489694],
 [-0.48732274, -1.85242909, 0.24926527],
 [0.92790164, 0.03222608, -0.59616936],
 [0.01142619, -1.03401828, 0.53889939],
 [-0.11019628, -2.65407282, -0.04679044],
 [0.44069345, -0.06329519, 0.20507382],
 [0.56210831, -1.76472438, -0.76577139],
 [0.71956189, -0.18622461, -0.06865895],
 [-0.0333547 , -0.43900321, 0.19493289],
 [0.87540719, 0.50906396, -0.50351138],
 [0.35025167, -0.19631173, 0.49087307],
 [0.15881005, -0.79209574, -0.30203717],
 [1.22509363, -1.6222438 , -0.48230402],
 [0.1649179 , -1.30260923, -0.17283781],
 [0.73768265, 0.39657156, 0.61652631],
 [0.47628719, -0.41732028, -0.26495223],
 [1.2341781 , -0.93332573, -0.36841227],
 [0.6328582 , -0.41638772, -0.29189625],
 [0.70266118, -0.06341182, -0.44602701],
 [0.87427365, 0.25079339, -0.47257895],
 [1.25650912, -0.07725602, -0.727155],
 [1.35840512, 0.33131168, -0.26082658],
 [0.66480037, -0.22592785, 0.08586389],
 [-0.04025861, -1.05871855, -0.31957333],
 [0.13079518, -1.56227183, -0.14998348],
 [0.02345269, -1.57247559, -0.24155228],
 [0.24153827, -0.77725638, -0.15121196],
 [1.06109461, -0.63384324, 0.10531139],
 [0.22397877, -0.28777351, 0.66524972],
 [0.42913912, 0.84558224, 0.45063407],
 [1.04872805, 0.5220518 , -0.39578638],
 [1.04453138, -1.38298872, -0.68829596],
 [0.06958832, -0.21950333, 0.29157927],
 [0.28347724, -1.32932464, 0.08941002],
 [0.27907778, -1.12002852, 0.0944876],
 [0.62456979, 0.02492303, -0.02048115],
 [0.33653037, -0.98840402, -0.19938976],
 [-0.36218338, -2.01923787, 0.10582105],

[0.28858624, -0.85573032, 0.13088969],
 [0.09136066, -0.18119213, 0.12897834],
 [0.22771687, -0.38492008, 0.15621315],
 [0.57638829, -0.1548736 , -0.27165036],
 [-0.44766702, -1.54379203, 0.19040093],
 [0.25673059, -0.5988518 , 0.09187916],
 [1.84456887, 0.87042131, 1.00540102],
 [1.15788161, -0.69886986, 0.53016015],
 [2.20526679, 0.56201048, -0.20291417],
 [1.44015066, -0.04698759, 0.16363011],
 [1.86781222, 0.29504482, 0.39562837],
 [2.75187334, 0.8004092 , -0.5823091],
 [0.36701769, -1.56150289, 0.98689327],
 [2.30243944, 0.42006558, -0.65170644],
 [2.00668647, -0.71143865, -0.39399057],
 [2.25977735, 1.92101038, 0.3975519],
 [1.36417549, 0.69275645, 0.28461207],
 [1.60267867, -0.42170045, 0.02318641],
 [1.8839007 , 0.41924965, 0.02633841],
 [1.2601151 , -1.16226042, 0.58024929],
 [1.4676452 , -0.44227159, 1.00386957],
 [1.59007732, 0.67624481, 0.63842871],
 [1.47143146, 0.25562182, 0.03743126],
 [2.42632899, 2.55666125, -0.12788146],
 [3.31069558, 0.01778095, -0.7033053],
 [1.26376667, -1.70674538, -0.26753689],
 [2.0377163 , 0.91046741, 0.23479948],
 [0.97798073, -0.57176432, 0.8281272],
 [2.89765149, 0.41364106, -0.85742182],
 [1.33323218, -0.48181122, -0.00542836],
 [1.7007339 , 1.01392187, 0.29845061],
 [1.95432671, 1.0077776 , -0.41998472],
 [1.17510363, -0.31639447, 0.12993776],
 [1.02095055, 0.06434603, 0.33771597],
 [1.78834992, -0.18736121, 0.27065801],
 [1.86364755, 0.56229073, -0.71563412],
 [2.43595373, 0.25928443, -0.72781615],
 [2.30492772, 2.62632347, -0.49347381],
 [1.86270322, -0.17854949, 0.35414871],
 [1.11414774, -0.29292262, -0.18348839],
 [1.2024733 , -0.81131527, -0.16472376],
 [2.79877045, 0.85680333, -0.5429065],
 [1.57625591, 1.06858111, 0.94585382],
 [1.3462921 , 0.42243061, 0.18087548],
 [0.92482492, 0.0172231 , 0.41682619],
 [1.85204505, 0.67612817, -0.01267211],
 [2.01481043, 0.61388564, 0.42833284],

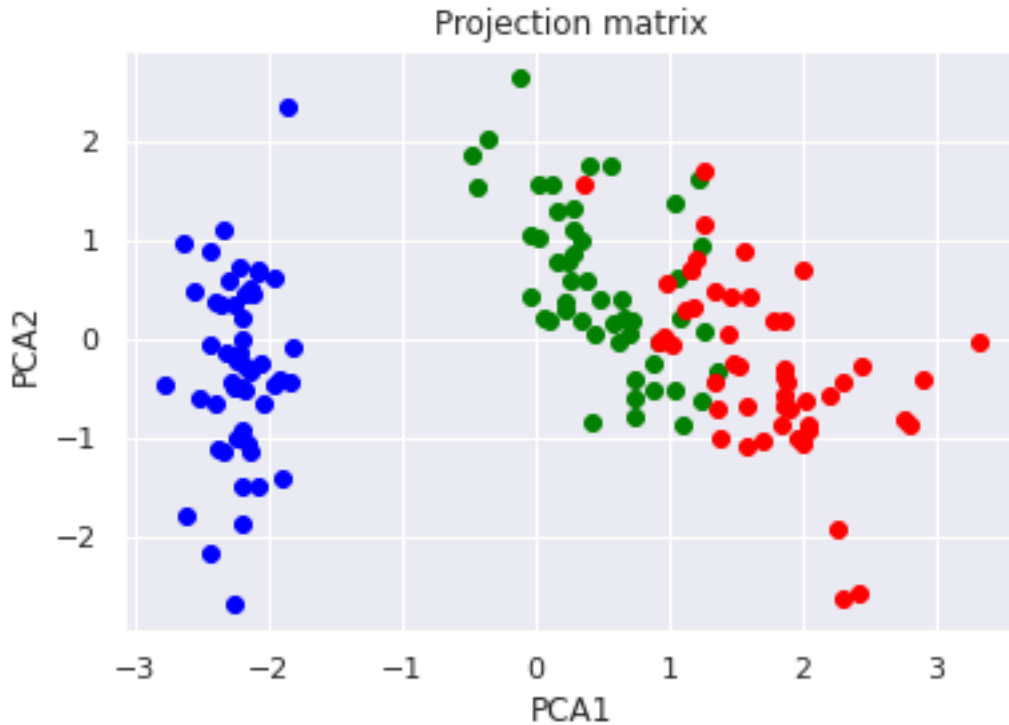

```
[ 1.90178409,  0.68957549,  0.13007501],
[ 1.15788161, -0.69886986,  0.53016015],
[ 2.04055823,  0.8675206 ,  0.338144  ],
[ 1.9981471 ,  1.04916875,  0.63241344],
[ 1.87050329,  0.38696608,  0.25627385],
[ 1.56458048, -0.89668681, -0.02637135],
[ 1.5211705 ,  0.26906914,  0.18017838],
[ 1.37278779,  1.01125442,  0.93339524],
[ 0.96065603, -0.02433167,  0.52824881]])
```

Task: Plot the features you found using `sklearn`'s implementation of the PCA in a similar way as we showed previously using the hand-implemented method. This will also verify if our implementation is correct and gives the same result as the built-in function.

```
[103]: # Your code here:
# Plotting the relationship between PCA1 and PCA2
feature1 = Y[:, 0]
feature2 = Y[:, 1]

species = ('Iris-setosa', 'Iris-versicolor', 'Iris-virginica')
colors = ('blue','green','red')
data = [[Y[np.where(targets == target)][:, feature] for feature in [0, 1]] for
        target in range(3)]
print("data shape:", np.shape(data))
for item, color, group in zip(data, colors, species):
    plt.scatter(item[0], item[1], color=color)
    plt.title('Projection matrix')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.show()
```

data shape: (3, 2, 50)



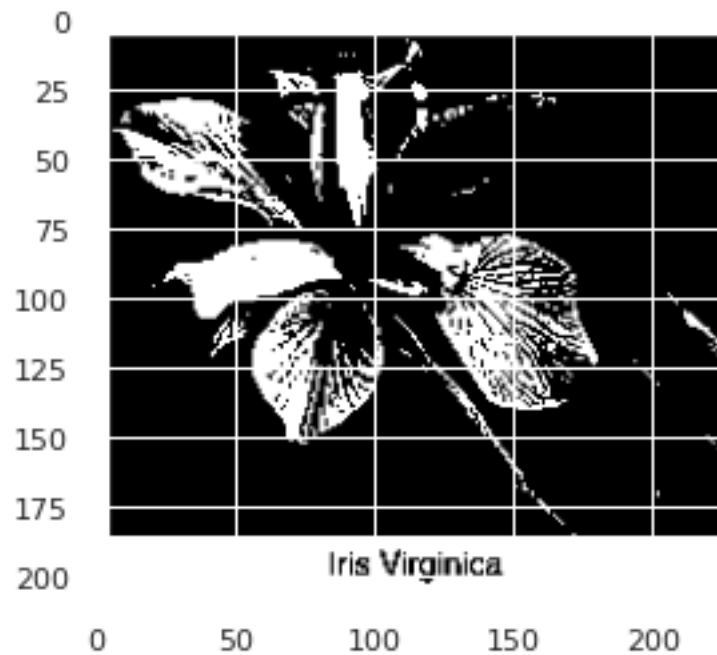
2.7 End Remark about coding in Python ;)

```
[ ]: # Uncomment the code below
      # import this
```

2.8 Bonus Task: Segmentation by simple thresholding

Consider the image `irisNorm1` we obtained previously. Extract a segmentation mask for the flower using a basic thresholding based on the histogram of the image that we displayed. A mask is basically a binary image with pixel values as 0 for the region of interest, and all other pixels as 1.

```
[115]: # Your code here:
        # Please name the mask segMask
        plt.imshow(irisNorm1>0.75, cmap='gray')
        plt.show()
```



2.9 Feedback Cell:

I hope this Notebook gave you a good start into python and Machine Learning. Let us know how you liked it. Any suggestions/ criticism are also welcome!

Your Feedback: