1- Machine Learning with scikit-learn

December 28, 2024

In the chain of processes that make up data analysis, the construction phase of predictive models and their validation are done by powerful library called **scikit-learn** .
In this chapter you will see some examples that will illustrate the basic construction of predictive models with some different models.

0.1 The scikit-learn Library:

scikit-learn is a Python module that integrates many of machine learning algorithms. This library was developed initally by Cornapeu in 2007, but the first real release was in 2010.
This library is part of the SciPy (Scientific Python) group, a set of libraries created for scientific computing and specially for data analysis, many of which are discussed in this book. Generally these libraries are definedas **SciKits**, hence the first part of the name of this library.
The second part of the library's nameis derived from **Machine Learning**, the discipline pertaining to this library.

0.2 Machine Learning:

Machine Learning is a discipline that deals with the study of methods for pattern recognation in data sets undergoing data analysis. In particular, it deals with development of algorithms that learn from data and make predictions. Each methodology is based on building a specific model.
There are very many methods that belong to the learning machine, each with its unique characteristics, which are specific to the nature of the data and the predictive model that you want to build. The choice of which method is to be applied is called **learning problem**.

The data to be subjected to a pattern in the learning phase can be arrays composed by a single value per element, or by multivariate value, These values are often referred to as **feature** or **attributes**.

0.3 Supervised and Unsupervised Learning:

Depending on the type of data and the model to be built, you can separate the learning problems into two broad categories ….
Supervised learning :
In supervised learning you know what the answer is, and you try to build a model to generate correct answer based on the answer which you already knew and prepared for the model…..
They are the methods in which the training set contains additional attributes that you want to predict **(target)**.
Thanks to these values, you can instruct the model to provide similar values when you have to submit new values **(test set)**.
- **Classification:**

(I have a photo)

the data in the training set belong to two or more classes or categories; then, the data, already being labeled, allow us to teach the system to recognize the characteristics that distinguish each class. When you will need to consider a new value unkown to the system, the system will evaluate its class according to its characteristics.

• **Regression:**
(I have a historical data)

when the value to be predicted is a continuous variable. The simplest case to understand is when you want to find the line which describes the trend from a series of points represented in a scatterplot.

0.3.1 Unsupervised learning:

In an unsupervised learning setting there is no answer, your model tries to find strucutre in your data, OR generate representations of your data based on GROUPING between the data intact with each other.
There are the methods in which the training set consists of input values x without any corresponding target value.

- **Clustering:**

the goal of these methods is to discover GROUPS of similar examples in a dataset.

- **Dimensionality reduction:**

reduction of high-dimwnsional dataset to one with only two or three dimensions is useful not just for data visualization, but for converting data of very high dimensionality into data of much lower dimensionality such that each of the lower dimensions conveys much more information.

In addition to these two main categories, there is further group of methods which have the purpose of validation and evaluation of the models.

0.3.2 Training Set & Testing Set:

Machine learning enables learning some properties by a model from a data set and applying them to new data. This is because a common practice in machine learning is to evaluate an algorithm. This evaluation consists of splitting the data into two parts, one called **training set**, with which we will learn the properties of the data, and the other is called the **testing set**, in which to test these properties.

1 Supervised Learning with scikit-learn:

In this chapter you will see a number of examples of **supervised learning**.

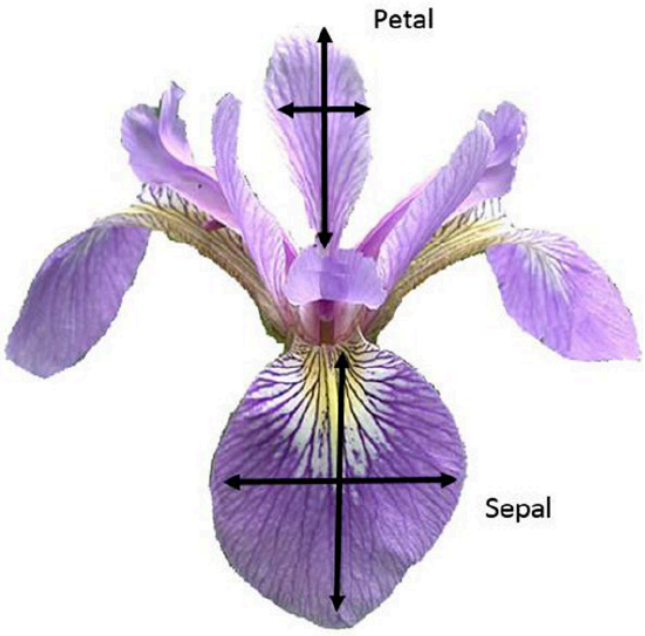
- Classification, using the Iris dataset
- K- Nearest Neighbors Classefier
- Support Vector Machine (SVC)
- Regression, using the Diabetes Dataset
- Linear Regression
- Support Vector Machines (SVR)

supervised learning consists of learning possible patterns between two or more features reading values from a training set; the learning is possible because the training set contains known results (target or labels).
All models in scikit-learning are referred to as **supervised estimators**, using the **fir(x,y)** function

that makes their training.
x comprises the features observed, while y indicates the target.
Once the estimator has carried out the training, it will be able to predict the value of y for any new observation x not labeled.
This operation will make it through the **predict (x)** function.

2 The Iris Flower Dataset:

The **Iris Flower Dataset** is a particular dataset used for the first time by Sir Ronald Fisher in 1936. It is often also called Andrson Iris Dataset, after the person who collected the data directly measuring the size of the different parts of the iris flowers. In this dataset, data from three different species of iris are collected and exactely these data correspond to the length and width of the sepals and the length and width of the petals.



This dataset is currently being used as a good example to utilize for many types of analysis , in particular as regards the problems of **classification** that can be approached by means of machine learning methodologies. It is no coincidence then that this dataset is provided along with the scikit-learn library as a 150*4 numpy array.
Now you will study this dataset in dtail importing it in your python working session.

[2]:
```
from sklearn import datasets
iris = datasets.load_iris()
```

In this way you loaded all the data and metadata concerning the Iris dataset in the *iris* variable. In order to see the values of the data collected in it, it is sufficient to call the attribute **data** of the variable *iris*.

[14]:
```
import pandas as pd
import numpy as np
```

[27]:
```
iris.data
```

[27]:
```
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]])
```

[21]:
```
iris_Table = pd.DataFrame(iris.data,␣
 ↪columns=['sepal_length','sepal_width','petal_length','petal_width'])
iris_Table
```

[21]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
..	…	…	…	…
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

[150 rows x 4 columns]

As you can see you will get an array of 150 elements, each containing 4 numeric values: the size of the sepals and petals repectively.
To know instead what kind of flower belongs each item you will refer to the **target** attribute.

[22]:
```
iris.target
```

[22]:
```
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, 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])
```

You obtain 150 items with three possible integer values (0,1 and 2) which correspond to the three species of iris. To know the correspondence between the species and number you have to call the **target_names** attribute.

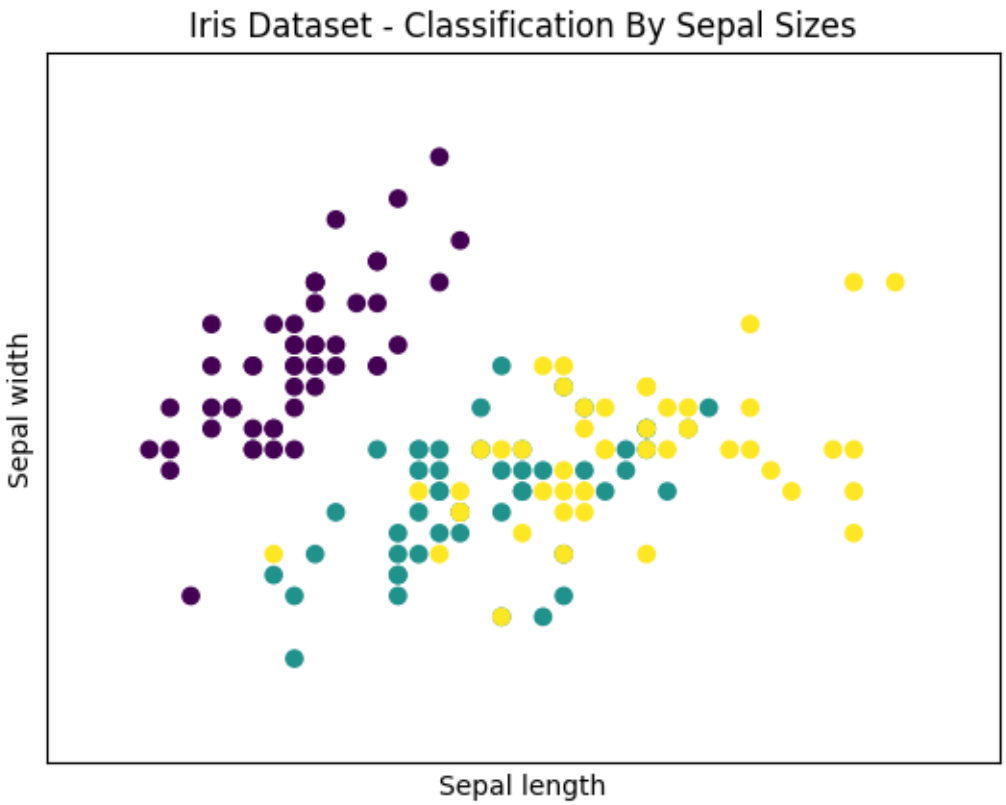
[24]:
```
iris.target_names
```

[24]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

To better understand this data set you can use the matplotlib library, using the techniques you learned in Chapter 7. Therefore create a scatterplot that displays the three different species in three different colors.
X-axis will represent the length of the sepal while the y-axis will represent the width of the sepal.

[30]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from sklearn import datasets

iris = datasets.load_iris()
x = iris.data[:,0] #X-Axis - sepal length
y = iris.data[:,1] #Y-Axis - sepal length
species = iris.target #Species

#Setting Axis limits....
#x.min() Finds the minimum value in the x array (smallest sepal length)
#x.max() Finds the maximum value in the x array (largest sepal length)
#x.min()-0.5 & x.max()+0.5 Adds a margin of 0.5 units below the minimum
# and above the maximum, ensuring the scatterplot is not tightly bounded
#to the data points.
#Add padding around the data points to improve plot aesthetics....
x_min, x_max = x.min() - .5,x.max() + .5
y_min, y_max = y.min() - .5,y.max() + .5

#SCATTERPLOT
plt.figure()
plt.title('Iris Dataset - Classification By Sepal Sizes')
plt.scatter(x,y, c=species)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
```

[30]:
```
([], [])
```

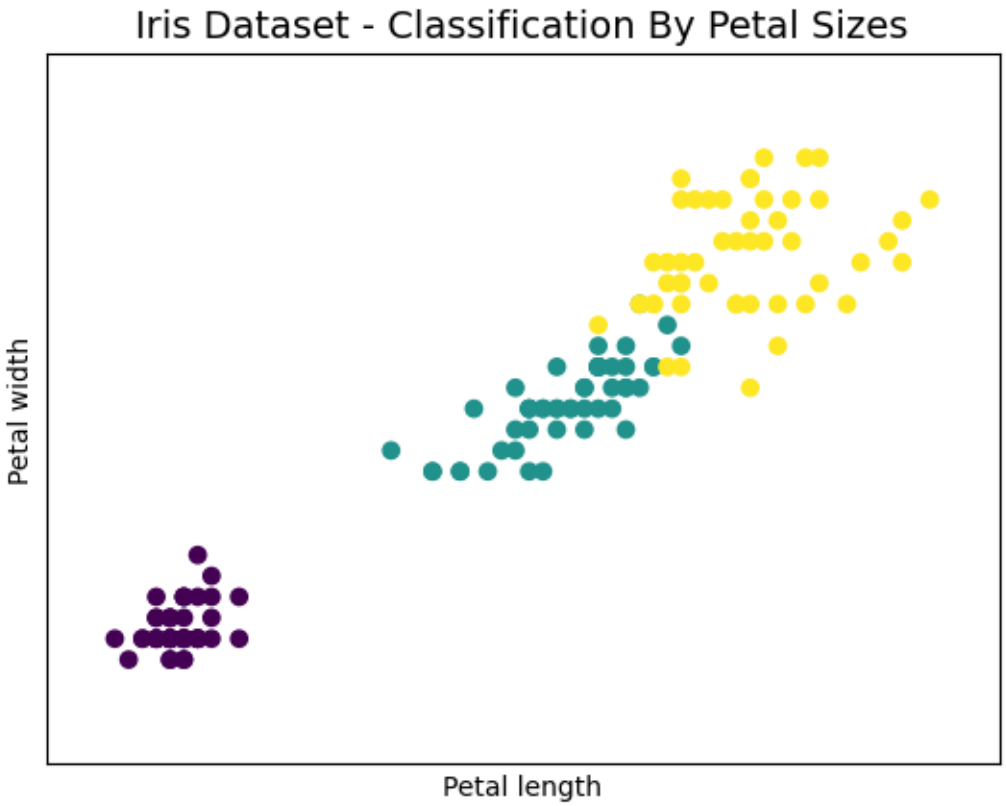


As a result you get a scatterplot as shown above, each species of iris is represented bt a specific color.
From the figure above you can see how the Iris sestosa features differ from the other two species, forming a cluster of violet dots separate from the other two species.
Try to follow the same procedure, but this time using the other two variables, that is the measure of the length and width of the petal.You can use the same code above only by changing some values.

[26]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from sklearn import datasets

iris = datasets.load_iris()
x = iris.data[:,2] #X-Axis - sepal length
y = iris.data[:,3] #Y-Axis - sepal length
species = iris.target #Species

x_min, x_max = x.min() - .5,x.max() + .5
y_min, y_max = y.min() - .5,y.max() + .5

#SCATTERPLOT
plt.figure()
```

```
plt.title('Iris Dataset - Classification By Petal Sizes', size=14)
plt.scatter(x,y, c=species)
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
```

[26]: ([], [])



The result is a scatterplot as shown above, In this case the division between the three species is much more evident. As you can see you have three different clusters.

3 The PCA Decomposition:

You have seen how the three species could be characterized taking into account four measuerments of the petals and sepals size.
We represented two scatterplots, one for the Petals and one for Sepals, but how can we unify the whole thing?Four dimensions are a problem that even a Scatterplot 3D is not able to solve.

In this regard a special technique called **Principle Component Analysis** has been developed.

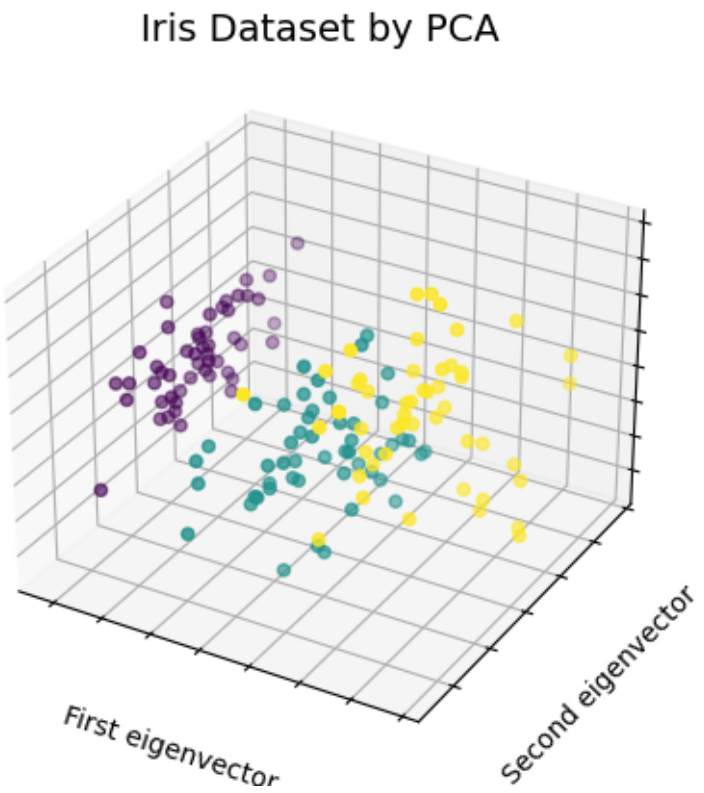
This technique allows you to reduce the number of dimensions of a system keeping all the information of the characterization of the various points, the new dimensions generated are called **principal components**.
In our case, so you can reduce the system from 4 to 3 dimensions and then plot the results within a 3D scatterplot.
In this way you can use measures both of sepals and petals for characterizing the various species of iris of the test elements in the dataset.

The Scikit-learn function which allows you to do the demensional reduction, is the **fir__transform()** function which belongs to the **PCA** object.
In order to use it, first you need to import the PCA module **sklearn.decomposition**.
Then you have to define the object constructor using **PCA()** defining the number of new dimensions (principal components) as valueof the **n__components** option.
In your case it is 3.
Finally you have to call the **fit__transform()** function passing the four-dimensional Iris Dataset as argument.

[32]:
```
from sklearn.decomposition import PCA
x_reduce = PCA(n_components=3).fit_transform(iris.data)
```

In addition, in order to visualize the new values you will use a scatterplot 3D using the **mpl__toolkits.mplot3d** module of matplotib.
If you don't remmeber how to do it, see the Scatterplot section in chapter 7.

[41]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.decomposition import PCA
iris = datasets.load_iris()
x = iris.data[:,1] #X-Axis - petal length
y = iris.data[:,2] #Y-Axis - petal length
species = iris.target #Species
x_reduced = PCA(n_components=3).fit_transform(iris.data)
#SCATTERPLOT 3D
fig = plt.figure()
#ax = Axes3D(fig) depracted line
ax = fig.add_subplot(111, projection='3d')
ax.set_title('Iris Dataset by PCA', size=14)
ax.scatter(x_reduced[:,0],x_reduced[:,1],x_reduced[:,2], c=species)
ax.set_xlabel('First eigenvector')
ax.set_ylabel('Second eigenvector')
ax.set_zlabel('Third eigenvector')
ax.xaxis.set_ticklabels(())
ax.yaxis.set_ticklabels(())
ax.zaxis.set_ticklabels(())
fig = plt.figure(figsize=(48,16))
```




```
<Figure size 4800x1600 with 0 Axes>
```

Good Bye …