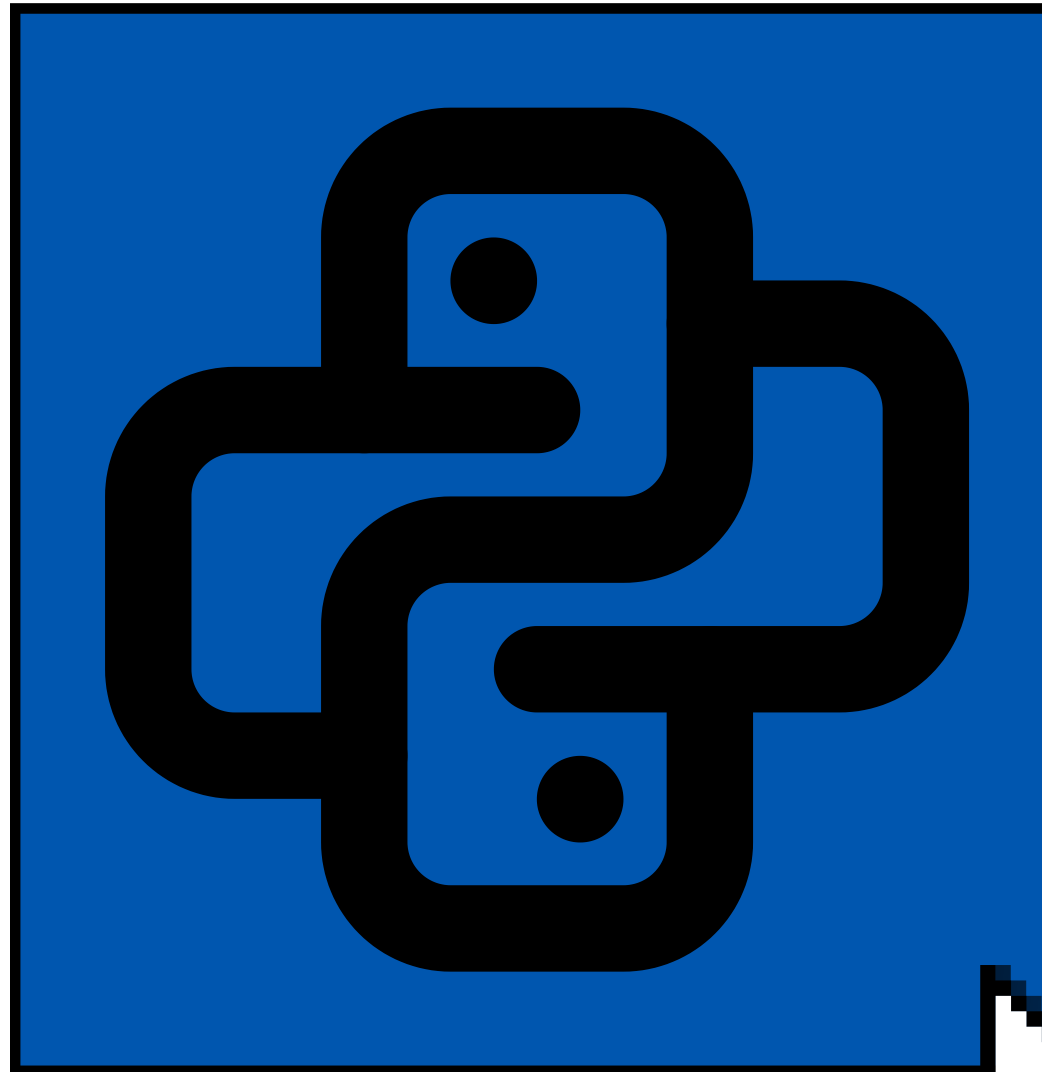
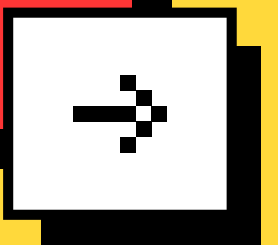


Una guida step-by-step

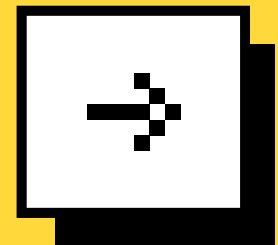


PCA CON PYTHON

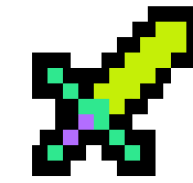
Con il supporto di Numpy



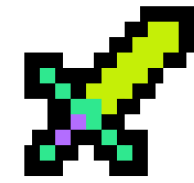
Numpy



...



Cos'è numpy?



NumPy (Numerical Python) è una **libreria** fondamentale per la programmazione scientifica in linguaggio Python. Contiene funzioni e moduli per affrontare una vasta gamma di operazioni matematiche.

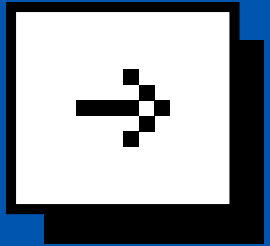


Moduli



Un modulo è un file contenente definizioni di funzioni e istruzioni Python.

Esempi di moduli NumPy



`numpy.random`

Contiene funzioni per la generazione di numeri casuali

`numpy.linalg`

Contiene funzioni per l'algebra lineare, come calcolo di determinanti, calcolo di autovalori e autovettori, e molte altre.

3/20



NumPy array object



Caratteristiche



01

Struttura di
dati base
fornita da
NumPy

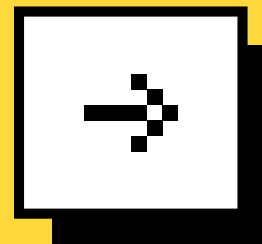
02

n-dimensionale

03

Istituito da
funzioni di NumPy

Creazione di array



Esempio 1



```
1 array = np.array([1, 4, 2, 5, 3])  
2 print(array)
```



```
array([1, 4, 2, 5, 3])
```



Esempio 2



```
1 lista = [1,2,3,4]  
2 array = np.array(lista)  
3 print(array)
```



```
[1 2 3 4]
```



Esempio 3



```
1 random_array = np.random.random((2, 3))  
2 print(random_array)
```



```
[[0.45315979 0.93884467 0.50065782]  
 [0.38647618 0.56890807 0.59955228]]
```

NumPy array object



Gli array NumPy hanno degli attributi:

✓ `array.shape`

Restituisce il numero di righe e di colonne dell'array.

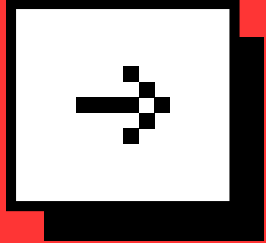
✓ `array.dtype`

Restituisce il tipo di dati degli elementi dell'array. Questo può essere "int", "float", ecc.

✓ `array.size`

Restituisce il numero totale di elementi nell'array.

Step 0: Prima di iniziare



Importare numpy e numpy.linalg

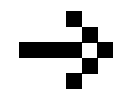


```
1 import numpy as np  
2 from numpy import linalg
```

7/20



Step 1: Importare il dataset

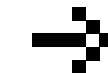


Iris Dataset

150 istanze di Iris classificate secondo tre specie: Iris setosa, Iris virginica e Iris versicolor. Le 4 variabili considerate sono la lunghezza e la larghezza del sepallo e del petalo.



```
1 from sklearn.datasets import load_iris
2 iris = load_iris()
3 X = iris.data
```



```
iris.data
iris.target
```





Step 2: Centrare i dati

`np.mean`



```
1 X_mean = np.mean(X)
2 X_centered = X - X_mean
```





Step 3: Calcolare la matrice di covarianza

`np.cov`



```
1 cov_matrix = np.cov(X_centered, rowvar=False)
```

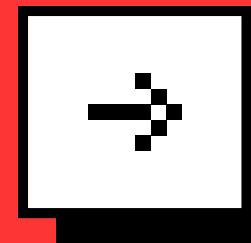


`rowvar = False`

Permette di calcolare la covarianza tra le quattro dimensioni del dataset, che sono disposte in colonna



Step 4: Autovalori e autovettori



`np.linalg.eig`



```
1 eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
```



La funzione restituisce 2 array:



1'array degli autovalori (4,)



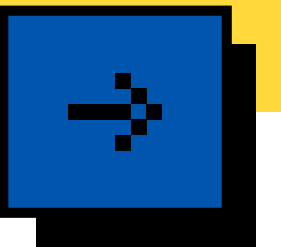
1'array degli autovettori (4, 4)

11/20



Step 5: Ordinare gli autovettori

Dobbiamo ordinare gli autovettori in ordine decrescente in base al loro autovalore corrispondente



`np.argsort`

Ordina gli indici di un array



```
1 sorted_eigenvalues_indexes = np.argsort(eigenvalues[::-1])
```



```
[::-1]
```

Impone che gli indici siano disposti in base all'ordine decrescente degli autovalori



```
1 sorted_eigenvectors = eigenvectors[:, sorted_eigenvalues_indexes]
```



Step 6: Selezionare gli autovettori



Osservare gli autovalori e scegliere quanti preservarne

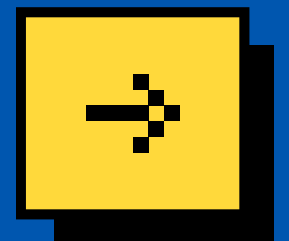
```
autovalori: [4.22824171 0.24267075 0.0782095 0.02383509]
```



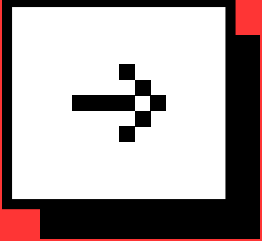
Selezionare gli autovettori corrispondenti



```
1 selected_eigenvectors = sorted_eigenvectors[:, :2]
```



Step 7: Calcolare le PC



np.dot



```
1 pc = np.dot(X_centered, selected_eigenvectors)
```



Le componenti principali si ottengono attraverso il prodotto scalare tra la matrice dei dati centrati e gli autovettori selezionati

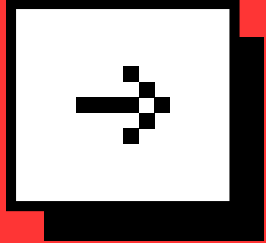


Otteniamo come output una matrice con tanti vettori quanti sono le PC

14/20



Step 8: Proiezione dei dati



np.dot



```
1 projected_data = np.dot(pc, selected_eigenvectors.T)
```

15/20



Calcoliamo il prodotto tra le componenti principali e la **trasposta** della matrice degli autovettori



Nel prodotto matriciale, il numero di colonne della prima matrice deve essere uguale al numero di righe della seconda matrice)



Plot dati

matplotlib.pyplot

Collezione di funzioni per la visualizzazione dei dati



```
1 import matplotlib.pyplot as plt
2 plt.scatter(X_centered[:,0], X_centered[:,1])
3 plt.xlabel('lunghezza del sepalo')
4 plt.ylabel('larghezza del sepalo')
```



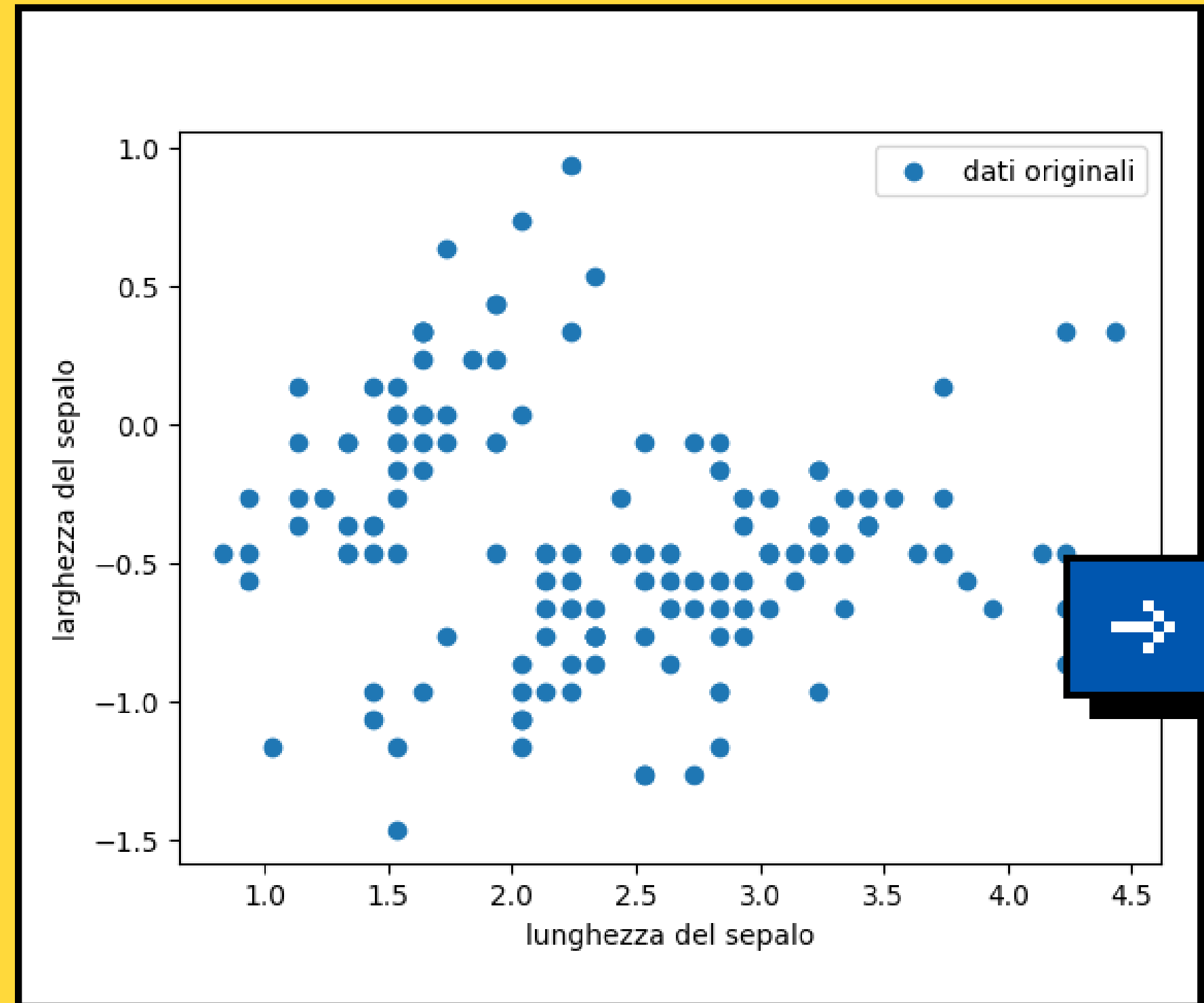
plt.scatter

Permette di visualizzare la relazione tra due variabili



plt.xlabel, plt.ylabel

Permettono di dare un nome alle ascisse e alle ordinate

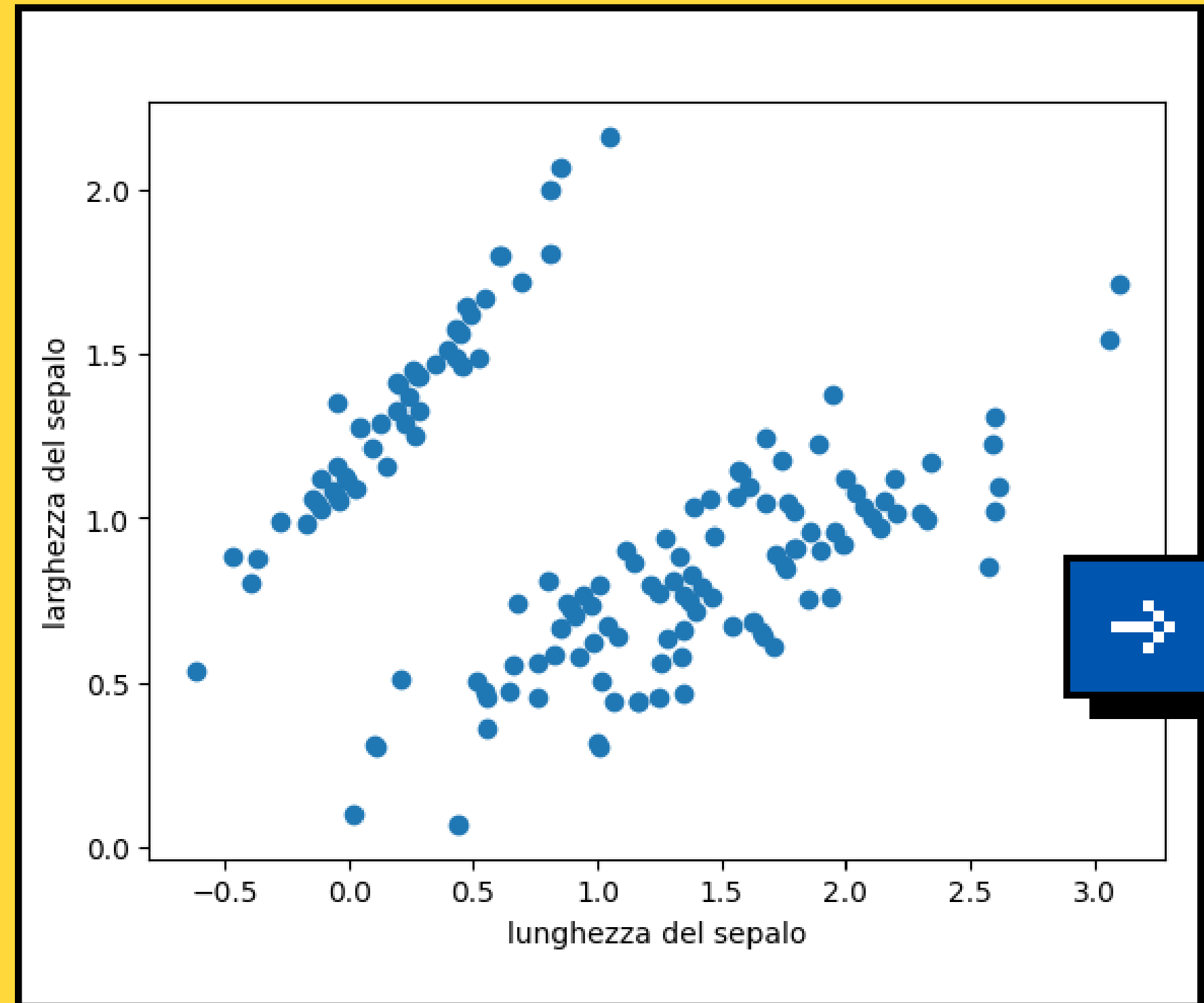


Plot dati



```
1 plt.figure() #serve a separare le due immagini
2 plt.scatter(projected_data[:,0], projected_data[:,1])
3 plt.xlabel('lunghezza del sepalo')
4 plt.ylabel('larghezza del sepalo')
```

Visualizziamo la stessa relazione di prima ma con i dati proiettati nel sottospazio delle componenti principali



Step 9: Varianza spiegata

`np.sum`



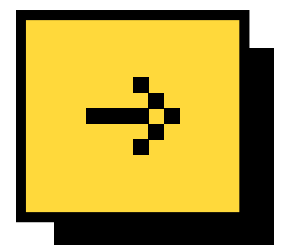
Somma di tutti autovalori



Somma degli autovalori corrispondenti agli autovettori selezionati



```
1 total_variance = np.sum(eigenvalues)
2 sum_selected_eigenvalues = np.sum(eigenvalues[:2])
```



Step 9: Varianza spiegata

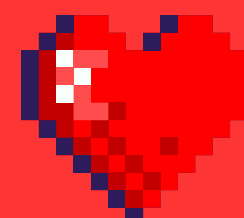


```
1 explained_variance = sum_selected_eigenvalues/total_variance  
2 print('varianza spiegata: ', explained_variance)
```

```
varianza spiegata: 0.9776852063187949
```



Rapporto tra la somma degli autovalori selezionati e la varianza totale, data dalla somma di tutti gli autovalori

 **GRAZIE!** 

<https://colab.research.google.com/drive/1GXvz8GZrdhkAgQKKWakrZZj4PgjtyDwY#scrollTo=UYukcw0ryfzf&uniqifier=1>