Sprint02_Tasca04

April 1, 2022

1 Sprint 02. Tasca 04

1.1 By José Manuel Castaño

1.2 Exercici 1

Crea una funció que donat un Array d'una dimensió, et faci un resum estadístic bàsic de les dades. Si detecta que l'array té més d'una dimensió, ha de mostrar un missatge d'error.

[6]:
```
import numpy as np
from numpy import random

def resum_estadistic(arr):
    if arr.ndim != 1:
        print("Error. La array suministrada té més d'una dimensió")
        return
    print("El tamany de la mostra es de {0}".format(len(arr)))
    print("El valor màxim és: {0}".format(np.max(arr)))
    print("El valor mínim és: {0}".format(np.min(arr)))
    print("La mitjana és: {0}".format(np.mean(arr)))
    print("La mediana és: {0}".format(np.median(arr)))
    print("La desviació estàndard és: {0}".format(np.std(arr)))
    print("La variancia és: {0}".format(np.var(arr)))
    print("El percentil 75 és: {0}".format(np.percentile(arr,75)))
    return

#Crea array de 20 enters aleatoris entre 0 i 20
arr = np.array([random.randint(20)for i in range(20)], dtype ='int64')
print(arr)

resum_estadistic(arr)    #Crida a la funció resum_estadistic
```

```
[18 10  0 17  4 19 13 10 14  0 10  3 10  7  4 15 14 15 15  6]
El tamany de la mostra es de 20
El valor màxim és: 19
El valor mínim és: 0
La mitjana és: 10.2
La mediana és: 10.0
La desviació estàndard és: 5.7236352085016735
```

```
La variancia és: 32.76
El percentil 75 és: 15.0
```

1.3 Exercici 2

Crea una funció que et generi un quadrat NxN de nombres aleatoris entre el 0 i el 100.

[9]:
```
def matriu_nxn(n):
    arr = np.random.randint(0, 100, (n,n))
    matriu = arr.reshape(n,n)
    return matriu

matriu_nxn(5)   #Crida a la funció matriu_nxn
```

[9]:
```
array([[76, 61, 32, 16, 19],
       [52, 89, 18, 57,  1],
       [35, 71, 88, 72, 46],
       [35, 56,  6, 13, 96],
       [ 1, 33, 49, 70,  9]])
```

1.4 Exercici 3

Crea una funció que donada una taula de dues dimensions, et calculi els totals per fila i els totals per columna.

[37]:
```
def totals_matriu(matriu):
    print("Totals per fila: {0}".format(matriu.sum(axis=1)))
    print("Totals per columna: {0}".format(matriu.sum(axis=0)))

matriu = matriu_nxn(5)   #Crida a la funció matriu_nxn de l'exercisi 2 per tal␣
 ↪de generar la matriu
print(matriu)
totals_matriu(matriu)    #Crida a la funció totals_matriu
```

```
[[65 88 28 36 68]
 [78 50 60 45 98]
 [86 90 32 66 97]
 [29 97 17 79 47]
 [12 30 10 99 62]]
Totals per fila: [285 331 371 269 213]
Totals per columna: [270 355 147 325 372]
```

1.5 - Exercici 4

Implementa manualment una funció que calculi el coeficient de correlació. Informa't-en sobre els seus usos i interpretació.

[16]:
```
arr1 = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
arr2 = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

def coefcorrelacio(variable1, variable2):
    mean1 = variable1.mean()        #Calcula mitja de la variable1
    mean2 = variable2.mean()        #Calcula mitja de la variable2

    dif1 = variable1 - mean1        #Calcula la diferencia entre la variable1 i␣
 ↪la seva mitja
    dif2 = variable2 - mean2        #Calcula la diferencia entre la variable2 i␣
 ↪la seva mitja

    covarianza = sum(dif1*dif2)/(len(dif1)-1)          #Calcula la covariança
    varianza1 = (sum(dif1*dif1)/(len(dif1)-1))**0.5     #Calcula la variança 1
    varianza2 = (sum(dif2*dif2)/(len(dif2)-1))**0.5     #Calcula la variança 2

    return covarianza/(varianza1*varianza2)

print(coefcorrelacio(arr1, arr2))
print ("Calculat amb la funció np.corrcoef")
print (np.corrcoef(x, y))
```

```
-0.758591524376155
Calculat amb la funció np.corrcoef
[[ 1.         -0.75859152]
 [-0.75859152  1.        ]]
```

[ ]:

[ ]: