Ortalama-Varyans-Standart Sapma Hesaplama

Soru:

3x3 boyutlarında bir matrisin elemanlarının ortalamasını, varyansını, standart sapmasını, minimum ve maksimum elemanlarını ve toplamlarını hesaplayan **hesapla()** adında Numpy dizisi döndüren bir fonksiyon oluşturma.

Fonksiyonun inputu(girdisi) 9 elemanı olan bir dizi olmalıdır. Fonksiyon diziyi 3x3'lük Numpy dizisine çevirmeli ve matrisin ortalamasını, varyansını, standart sapmasını, minimum ve maksimum elemanlarını ve toplamını dictionary olarak return etmelidir.

Fonksiyonun return edeceği dictionary şu şekilde olmalıdır:

```
{
  'mean': [axis1, axis2, flattened],
  'variance': [axis1, axis2, flattened],
  'standard deviation': [axis1, axis2, flattened],
  'max': [axis1, axis2, flattened],
  'min': [axis1, axis2, flattened],
  'sum': [axis1, axis2, flattened]
}
```

Eğer fonksiyonun input dizisinin boyutu 9'dan küçük ise "Dizinin 9 elemanı olmak zorundadır." ValueError mesajıyla exception oluşturulmalıdır.

Örneğin `hesapla([0,1,2,3,4,5,6,7,8])` fonksiyonunun return edeceği dictionary:

```
{
  'mean': [[3.0, 4.0, 5.0], [1.0, 4.0, 7.0], 4.0],
  'variance': [[6.0, 6.0, 6.0], [0.6666666666666666, 0.6666666666666666,
0.6666666666666666], 6.666666666666667],
  'standard deviation': [[2.449489742783178, 2.449489742783178,
2.449489742783178], [0.816496580927726, 0.816496580927726,
0.816496580927726], 2.581988897471611],
  'max': [[6, 7, 8], [2, 5, 8], 8],
  'min': [[0, 1, 2], [0, 3, 6], 0],
  'sum': [[9, 12, 15], [3, 12, 21], 36]
}
```

Cevap:

Numpy dizisi kullanacağımız için ilk adımda Numpy kütüphanesini import ediyoruz.

```
import numpy as np
```

Fonksiyonun input dizisinin 9 boyutlu olacağı söyleniyor. Exception oluşturarak dizinin boyutuna kontrol edelim.

```
def hesapla(dizi):
 if len(dizi) != 9:
   raise Exception("Dizinin 9 elemanı olmak zorundadır.")
 return "Dizi 9 elemanlı"
hesapla([0,1,2,3,4,5,6,7,8])
=>Dizi 9 elemanlı
hesapla([0,1,2,3,4,5,6,7])
=> ValueError: Dizinin 9 elemanı olmak zorundadır.
```

Yukarıdaki örnekleri çalıştırdığımızda Exception durumun çalıştığını görüyoruz. Şimdi girilen diziyi 3x3'lük matrise çevirmeye geçebiliriz. Numpy'ın reshape fonksiyonunu kullanarak yapacağız.

```
def hesapla(dizi):
 if len(dizi) != 9:
   raise ValueError("Dizinin 9 elemanı olmak zorundadır.")
 matrix = np.reshape(dizi,(3,3))
 return matrix
hesapla([0,1,2,3,4,5,6,7,8])
=> array([[0, 1, 2],
          [3, 4, 5],
          [6, 7, 8]])
```

Numpy kütüphanesi birçok matematiksel işlemleri yapmamızı sağlayan fonksiyonlar içerir. Verilen matrisin ortalamasını, varyans ve standart sapma hesaplamasını kolayca yapabiliriz. Örneğin matrisin ortalamasını hesaplamak için **mean()** fonksiyonunu kullanalım.

```
def hesapla(dizi):
 if len(dizi) != 9:
   raise ValueError("Dizinin 9 elemanı olmak zorundadır.")
 matrix = np.reshape(dizi,(3,3))
 mean=np.mean(matrix)
 return mean
=> 4.0
```

Yukarıdaki soruda çıktı olarak beklenen sözlük yapısını incelediğimizde her bir işlem için 3 farklı cevap beklenmektedir. Matrisimize tekrardan bakalım.

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

Ortalama hesaplaması için 3 sonucumuzun olması gerekiyor.

```
'mean': [[3.0, 4.0, 5.0], [1.0, 4.0, 7.0], 4.0]
```

Buradaki ilk liste matrisin sütunlarının ortalaması: `[3.0, 4.0, 5.0] -> [sütun1_ortalaması,sütun2_ortalaması,sutün3_ortalaması]`

ikinci liste matrisin satırlarının ortalaması: `[1.0, 4.0, 7.0] -> [satır1_ortalaması,satır2_ortalaması,satır3_ortalaması]`

sonuncu değer ise matrisin tüm elemanlarının ortalaması: `4.0`

Matrisin satır ve sütun ortalamasını hesaplamak için **mean()** fonksiyonuna **axis** değerini girmemiz gerekmektedir.

```
def hesapla(dizi):
 if len(dizi) != 9:
   raise ValueError("Dizinin 9 elemanı olmak zorundadır.")
 matrix = np.reshape(dizi,(3,3))
 mean= [np.mean(matrix, axis=0),
        np.mean(matrix, axis=1),
        np.mean(matrix)
        ]
 return mean
hesapla([0,1,2,3,4,5,6,7,8])
=> [array([3., 4., 5.]), array([1., 4., 7.]), 4.0]
```

Çıktımız Numpy arrayi olarak return etti. Fakat çıktı liste olarak return etmesi gerekiyor çünkü bizden beklenen sonucun liste tipinde olduğunu görüyoruz. Sonuçları **tolist()** fonksiyonu kullanarak listeye çevirelim.

```
#'mean': [[3.0, 4.0, 5.0], [1.0, 4.0, 7.0], 4.0]
def hesapla(dizi):
 if len(dizi) != 9:
   raise ValueError("Dizinin 9 elemanı olmak zorundadır.")
 matrix = np.reshape(dizi,(3,3))
 mean= [np.mean(matrix, axis=0).tolist(),
```

```
        np.mean(matrix, axis=1).tolist(),
        np.mean(matrix).tolist()
        ]
 return mean
hesapla([0,1,2,3,4,5,6,7,8])
=> [[3.0, 4.0, 5.0], [1.0, 4.0, 7.0], 4.0]
```

Yapacağımız işlemler dictionary olarak istenmektedir. Dictionarylerde anahtar/değer ilişkisi vardır. Değerler herhangi bir veri türünden olabilir ve tekrarlanabilirken, anahtarlar değişmez türde olmalıdır. Dictionary kullanarak yapacağımız tüm işlemleri tek bir çıktı olarak return edebileceğiz. Nasıl mı?

```
def hesapla(dizi):
 if len(dizi) != 9:
   raise Exception("List must contain nine numbers.")
 matrix = np.reshape(dizi,(3,3))
 islemler = {
     "mean":[
             np.mean(matrix, axis=0).tolist(),
             np.mean(matrix, axis=1).tolist(),
             np.mean(matrix).tolist()
             ],
     "variance":[
             np.var(matrix, axis=0).tolist(),
             np.var(matrix, axis=1).tolist(),
             np.var(matrix).tolist()
             ]
 }
 return islemler
hesapla([0,1,2,3,4,5,6,7,8])

=> {'mean': [[3.0, 4.0, 5.0], [1.0, 4.0, 7.0], 4.0],
 'variance': [[6.0, 6.0, 6.0],
  [0.6666666666666666, 0.6666666666666666, 0.6666666666666666],
  6.666666666666667]}
```

islemler adından bir dictionary oluşturarak 2 adet anahtar belirledik. Bunlar mean ve variance değerleridir. Böylece tek bir çıktıda birden fazla işlemi görmüş olduk. Hadi, bizden istenen tüm işlemleri de dictionary'e ekleyelim ve cevabımızı tamamlayalım :)

```
def calculate(list):
  if len(list) != 9:
   raise Exception("List must contain nine numbers.")
  matrix = np.reshape(list,(3,3))
  operations = {
     "mean":[
             np.mean(matrix, axis=0).tolist(),
             np.mean(matrix, axis=1).tolist(),
             np.mean(matrix).tolist()
             ],
     "variance":[
             np.var(matrix, axis=0).tolist(),
             np.var(matrix, axis=1).tolist(),
             np.var(matrix).tolist()
             ],
```

```
    "standart deviation":[
            np.std(matrix, axis=0).tolist(),
            np.std(matrix, axis=1).tolist(),
            np.std(matrix).tolist()
            ],
    "max":[
            np.max(matrix, axis=0).tolist(),
            np.max(matrix, axis=1).tolist(),
            np.max(matrix).tolist()
            ],
    "min":[
            np.min(matrix, axis=0).tolist(),
            np.min(matrix, axis=1).tolist(),
            np.min(matrix).tolist()
            ],
    "sum":[
            np.sum(matrix, axis=0).tolist(),
            np.sum(matrix, axis=1).tolist(),
            np.sum(matrix).tolist()
            ],
 }
 return operations

calculate([0,1,2,3,4,5,6,7,8])

=> {'max': [[6, 7, 8], [2, 5, 8], 8],
 'mean': [[3.0, 4.0, 5.0], [1.0, 4.0, 7.0], 4.0],
 'min': [[0, 1, 2], [0, 3, 6], 0],
 'standart deviation': [[2.449489742783178,
   2.449489742783178,
   2.449489742783178],
  [0.816496580927726, 0.816496580927726, 0.816496580927726],
  2.581988897471611],
 'sum': [[9, 12, 15], [3, 12, 21], 36],
 'variance': [[6.0, 6.0, 6.0],
  [0.6666666666666666, 0.6666666666666666, 0.6666666666666666],
  6.666666666666667]}
```