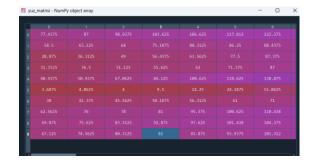
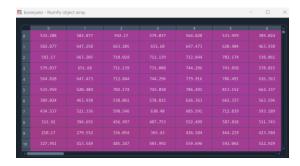
a- Ödevde kullanmak için aşağıdaki fotoğrafta belirtilen kütüphaneleri import ettim. "yuzler" adında oluşturduğum klasöre .pgm uzantılı olan dosyaları topladım. Boyut yüksek olmasından dolayı işlem süresi uzun olacağından boyutları düşürdüm. yuz_matrisi ismi adlı matrisle değerleri tutarak array haline getirdim. Yukarıdaki açıklamaya ilişkin kod aşağıda belirtilmiştir.

```
import <mark>numpy</mark> as np
from skimage.io import imread
from skimage.measure import block_reduce
import glob
from sklearn.decomposition import PCA
from PIL import Image
From sklearn.decomposition import PCA
foto_yolu =r"C:\Users\yigit\Desktop\odev2\yuzler\*.pgm"
foto_yolu = glob.glob(foto_yolu)
yuz matrisi = []
for face in foto yolu:
    foto = imread(face)
    foto = block_reduce(foto,block_size=(4,4), func=np.mean)
    foto = np.array(foto.flatten())
    yuz_matrisi.append(foto)
yuz_matrisi = np.array(yuz_matrisi)
```

10*2016 boyutundaki öznitelik matrisinin kovaryansını almamız gerekiyor. Aşağıda yazdığım kodda bu işlemi yapabiliriz. Ardından bu kodunun çıktısında ise 2016*2016 boyutunda kovaryans matrisi çıktısı alıyoruz. Aşağıda ilgili kodlar ve çıktılar belirtilmiştir.

```
yuz_matrisi = np.array(yuz_matrisi)
pca = PCA()
pca.fit(yuz_matrisi)
kovaryans = pca.get covariance()
```





10*2016 öznitelik matrisi

2016*2016 kovaryans matrisi

b- A şıkkında çıktı olarak aldığımız kovaryans matrisinin özvektörlerini ve özdeğerlerini aşağıda kodda belirtildiği gibi elde ediyorum. (PCA kütüphanesi.)

```
vektor = pca.components_
value = pca.explained_variance
```

23	Vã	alue - NumPy object				
,						
	_	0				
		604547				
		263250				
		221252				
	3	197951				
	4	148602				
	5	129402				
		93748.5				
		77558.7				
	8	51470.5				
	9	7.06528e-26				

ve	ktor - NumPy obje	ect array					- 0		
4									
	-0.0146191	-0.0184074	-0.0207721	-0.0215319	-0.0241067	-0.0252245	-0.023436		
	0.0317136	0.0335855	0.034609	0.033177	0.0303339	0.0301719	0.021133		
2	0.0212575	0.0202149	0.0134653	0.0113718	0.00779474	0.000649829	-0.004845		
3	-0.00994696	-0.0124325	-0.0194987	-0.0240575	-0.0263728	-0.0284298	-0.020108		
	0.00949727	0.0108955	0.00897406	0.00432148	0.00440762	-0.000327315	-0.006156		
5	0.000612021	0.00483263	-0.00212215	-0.00414184	0.00187359	0.000555871	0.001417		
	-0.00400663	-0.00345941	-0.00446669	0.0038916	0.0121821	0.0142972	0.014869		
7	-0.00697739	-0.00587558	-0.00705888	-0.0120898	-0.0151657	-0.0106247	-0.005214		
	-0.000682764	-0.000221628	0.00194014	-0.00148575	0.00170574	-0.00389983	4.19845e-		
	0.273323	0.125216	0.054229	-0.110598	0.0375402	-0.221672	0.170164		

Özvektör matrisi

Özdeğer matrisi

Kullandığım PCA kütüphanesiyle birlikte çıktı olarak seçilmiş özvektörler verildi. Fotoğrafların siyah beyazlağını daha görünür kılmak adına aşağıda görüldüğü üzere 2048 ile çarptım. İlgili kod ve koda ait çıktılar belirtilmiştir.



c- Bu aşamadan sonra ise elimizdeki projeksiyon hesaplama formülünü kullanarak seçilmiş özvektörler ve kovaryans matrisiyle çarpıp fotoğrafların izdüşümünü buldum. Bu hesaplamayı yapmak için numpy ve PIL kütüphanesinden yararlandım.

```
cikti = np.matmul(vektor,kovaryans)

cikti = cikti.T

ters = np.linalg.ters(np.matmul(cikti.T,cikti))

project = np.matmul(cikti, ters)

project = np.matmul(project, cikti.T)

cikti_yeni = np.matmul(yuz_matrisi, project)

for i in range(10):
    foto_cikti = Image.fromarray(cikti_yeni[i].reshape(48,42))
    foto_cikti.show()
```



d- Her bir izdüşürülmüş fotoğrafın oklid uzaklığını hesaplamak için aşağıdaki kodu kullandım. Kod sonucu 10 fotoğrafın da öklid uzaklığı eşit çıktı.

```
oklid = []
for i in range(10):
    temp = np.linalg.norm(yuz_matrisi[i]- cikti_yeni[i])
    oklid.append(temp)
for i in range(10):
    print("",i+1,". ","",oklid[i])
```

```
1 . 1898.272458501607

2 . 1898.272458501607

3 . 1898.272458501607

4 . 1898.272458501607

5 . 1898.272458501607

6 . 1898.272458501607

7 . 1898.272458501607

8 . 1898.272458501607

9 . 1898.272458501607

10 . 1898.272458501607
```

e- Orijinal fotoğraflar ile izdüşürülen görüntüler arasında anlaşılabilirliklerin yüksek olduğunu ve benzerliklerinin yüksek olduğunu düşünüyorum bu yüzden 10 yüz görüntüsünün yeterli olduğunu düşünüyorum. Daha fazla fotoğrafın modellenmesi benzerlik açısından pozitif etki gösterecektir. Daha fazla model kullanması durumunda baz vektörlerleri daha fazla yer kaplayacaktır. Bu durumda ise negatif etki gösterecek hem bellek açısından hem işlem hızı açısından negatif etki gösterecektir.

KODUN TAMAMI

```
import numpy as np
      from skimage.io import imread
      from skimage.measure import block reduce
      import glob
      from sklearn.decomposition import PCA
     from PIL import Image
7
     from sklearn.decomposition import PCA
     foto_yolu = r"C:\Users\yigit\Desktop\odev2\yuzler\*.pgm"
11
     foto yolu = glob.glob(foto yolu)
12
     yuz matrisi = []
15
     for face in foto_yolu:
         foto = imread(face)
17
          foto = block_reduce(foto,block_size=(4,4), func=np.mean)
         foto = np.array(foto.flatten())
         yuz matrisi.append(foto)
     yuz matrisi = np.array(yuz matrisi)
22
     pca = PCA()
     pca.fit(yuz_matrisi)
25
     kovaryans = pca.get covariance()
     vektor = pca.components_
     value = pca.explained_variance_
     for i in range(10):
         foto_cikti = Image.fromarray(vektor[i].reshape(48,42)*2048)
         foto_cikti.show()
      cikti = np.matmul(vektor,kovaryans)
      cikti = cikti.T
      ters = np.linalg.inv(np.matmul(cikti.T,cikti))
      project = np.matmul(cikti, ters)
      project = np.matmul(project, cikti.T)
      cikti_yeni = np.matmul(yuz_matrisi, project)
      for i in range(10):
          foto_cikti = Image.fromarray(cikti_yeni[i].reshape(48,42))
          foto cikti.show()
      oklid = []
      for i in range(10):
          temp = np.linalg.norm(yuz_matrisi[i]- cikti_yeni[i])
```