Лабораторная работа №7 “Метод главных компонент”

Долматович Алина, 858641

In [1]:
```
from scipy.io import loadmat
import matplotlib.pyplot as pyplot
import numpy as np
from scipy import misc
from mpl_toolkits.mplot3d import Axes3D
import math
```

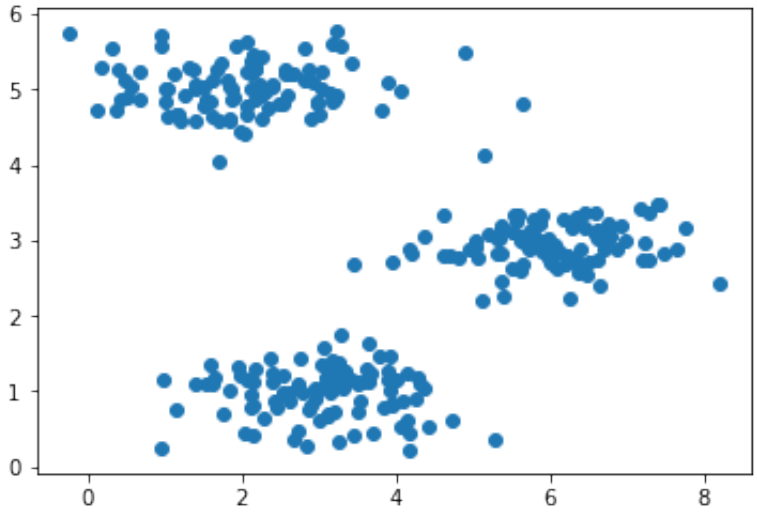
Загрузите данные ex7data1.mat из файла.

In [2]:
```
data = loadmat('ex7data1.mat')

x = data['X']
print(x.shape)
```

```
(300, 2)
```

Постройте график загруженного набора данных.

In [3]:
```
pyplot.scatter(x[:,0], x[:, 1])
pyplot.show()
```



Реализуйте функцию вычисления матрицы ковариации данных.

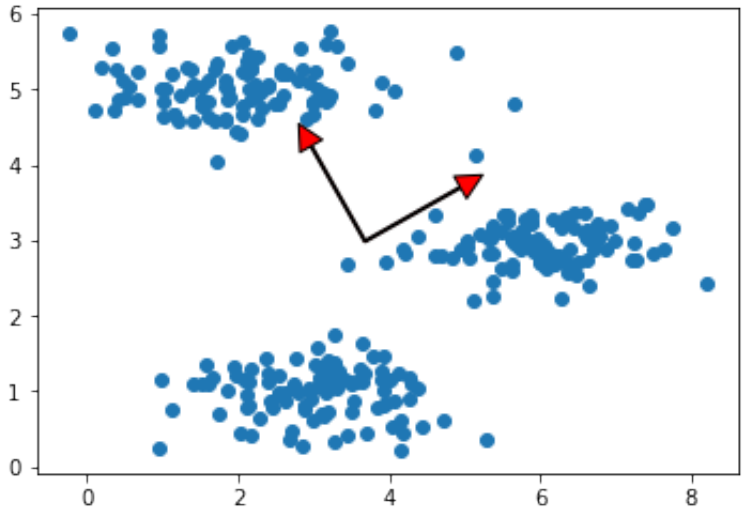
In [4]:
```
def getCovMatrix(x):
    return np.cov(x.T)
```

Вычислите координаты собственных векторов для набора данных с помощью сингулярного разложения матрицы ковариации (разрешается использовать библиотечные реализации матричных разложений).

In [5]:
```
def getEigenvectors(x):
    covMatrix = getCovMatrix(x)
    eigenvalues, eigenvectors = np.linalg.eig(covMatrix)
    return eigenvectors

eigenvectors = getEigenvectors(x)
```

Постройте на графике из пункта 2 собственные векторы матрицы ковариации.

In [6]:
```
data = x
mu = data.mean(axis=0)
data = data - mu

projected_data = np.dot(data, eigenvectors)
sigma = projected_data.std(axis=0).mean()

fig, ax = pyplot.subplots()
ax.scatter(x[:,0], x[:, 1])
for axis in eigenvectors:
    start, end = mu, mu + sigma * axis
    ax.annotate('', xy=end, xycoords='data',
                xytext=start, textcoords='data',
                arrowprops=dict(facecolor='red', width=1.0))
ax.set_aspect('equal')
pyplot.show()
```



Реализуйте функцию проекции из пространства большей размерности в пространство меньшей размерности с помощью метода главных компонент.

In [7]:
```
def dimensionalityReduction(x):
    eigenvectors = getEigenvectors(x)
    v = -eigenvectors[:,1]
    return np.dot(v, x.T), v

xNew, v = dimensionalityReduction(x)
```

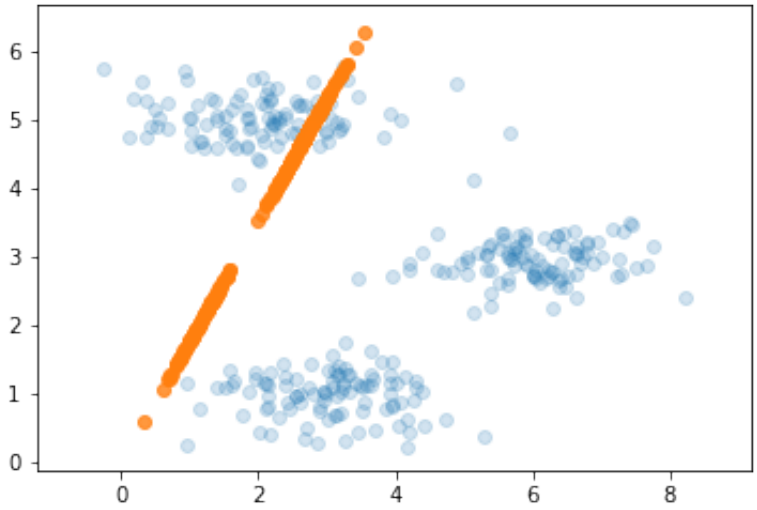
Реализуйте функцию вычисления обратного преобразования.

In [8]:
```
def restored(x):
    xNew, v = dimensionalityReduction(x)
    xRestored = []
    for n in range(len(xNew)):
        xRestored.append(np.dot(xNew[n], v))

    return np.array(xRestored)

xRestored = restored(x)
print(xRestored)
```

```
[ 2.67413059  4.73627272]
[ 2.94928918  5.22361846]
[ 2.37515599  4.20674539]
[ 2.60777935  4.61875507]
[ 2.63701735  4.6705398 ]
[ 2.53388722  4.4878814 ]
[ 2.97626259  5.27139228]
[ 2.88945434  5.11764229]
[ 3.11685324  5.5203987 ]
[ 2.85478206  5.05623266]
[ 2.05009561  3.63101637]
[ 2.50891134  4.4436455 ]
[ 2.65020716  4.69390087]
[ 2.58302576  4.57491287]
[ 2.14331113  3.79611455]
[ 2.85219494  5.0516505 ]
[ 2.37558904  4.20751238]
[ 2.34455228  4.15254178]
[ 2.37779543  4.21142022]
[ 2.50085438  4.429375461
```

Постройте график исходных точек и их проекций на пространство меньшей размерности (с линиями проекций).

In [9]:
```
pyplot.scatter(x[:, 0], x[:, 1], alpha=0.2)
pyplot.scatter(xRestored[:, 0], xRestored[:, 1], alpha=0.8)
pyplot.axis('equal')
pyplot.show()
```



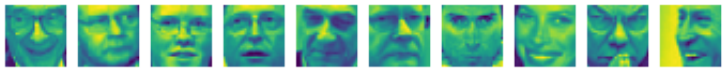
Загрузите данные ex7faces.mat из файла.

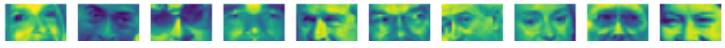
In [10]:
```
data = loadmat('ex7faces.mat')
x = data['X']

print(x.shape)
```

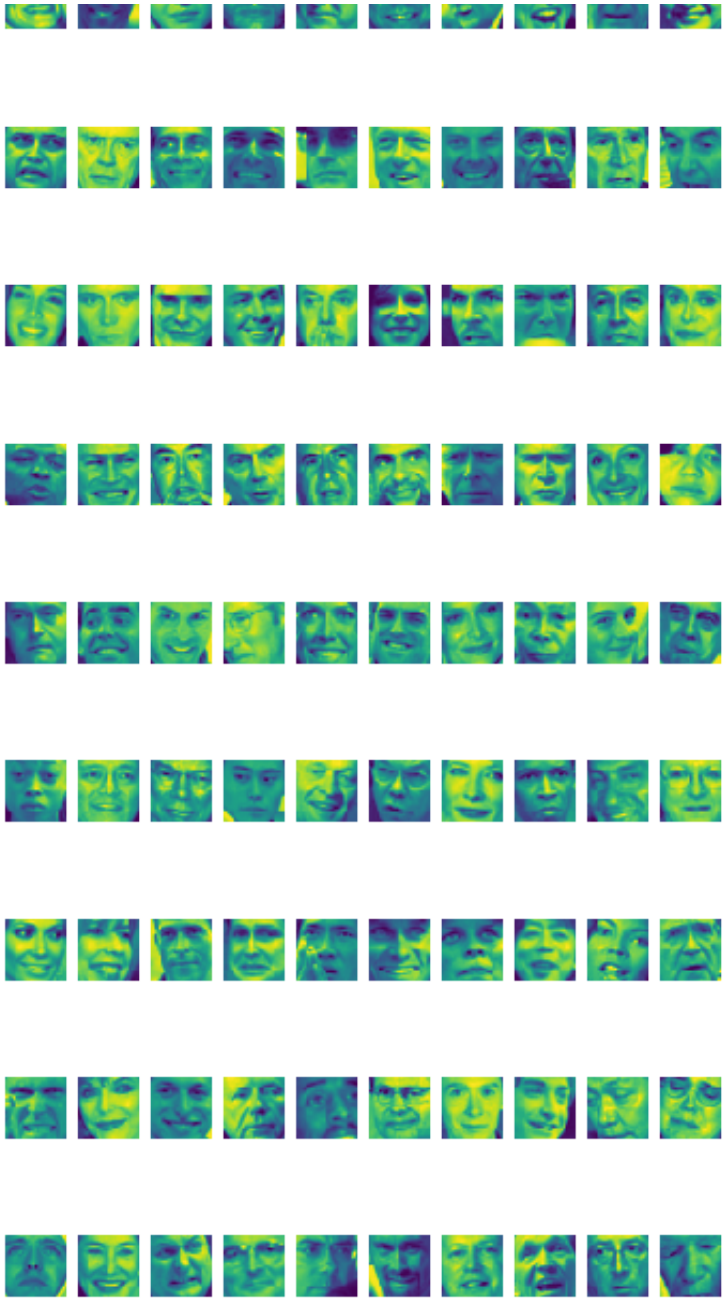
```
(5000, 1024)
```

Визуализируйте 100 случайных изображений из набора данных.

In [11]:
```
def visualization(x, count=100):
    lines = int(math.ceil(count / 10.0))
    for i in range(lines):
        _, axis = pyplot.subplots(1, 10)
        for j in range(10):
            index = 10*i + j
            if index < count:
                matrix = x[index].reshape(32, 32, order="F")
                axis[j].imshow(matrix)
            axis[j].axis("off")
        pyplot.show()

visualization(x)
```







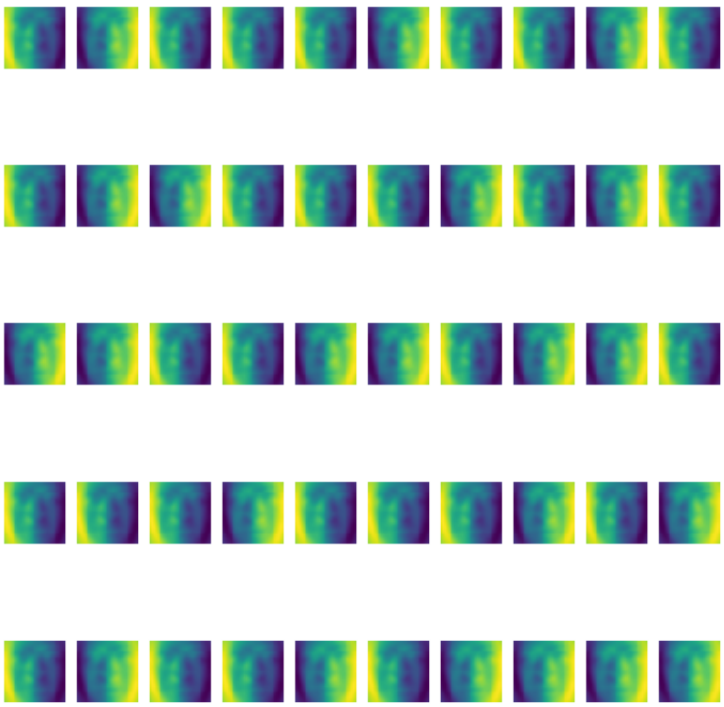
С помощью метода главных компонент вычислите собственные векторы.

In [12]:
```
eigenvectors = getEigenvectors(x)
print(eigenvectors.shape)

```

```
(1024, 1024)
```

Визуализируйте 36 главных компонент с наибольшей дисперсией. Визуализируйте 100 главных компонент с наибольшей дисперсией. Как изменилось качество выбранных изображений?

In [13]:
```
def imagesVisualization(x, count):
    xRestored = restored(x)
    dispersions = np.array([np.var(value) for value in xRestored])
    indexes = dispersions.argsort()[-count:][::-1]
    values = np.array([xRestored[index] for index in indexes])
    visualization(values, count)

imagesVisualization(x, 50)
```



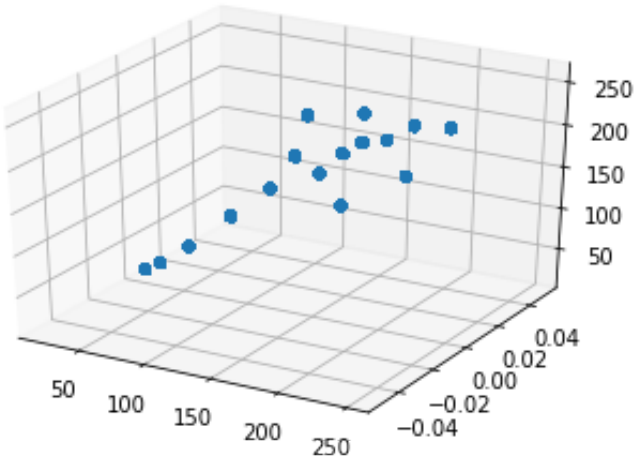
Используйте изображение, сжатое в лабораторной работе №6 (Кластеризация).

In [14]:
```
compressedImage = misc.imread('compressedImage.png')
print(compressedImage.shape)

compressedImage = compressedImage.reshape(16384, 3)
print(compressedImage)

```

```
(128, 128, 3)
[[221 186   0]
 [221 186   0]
 [221 186   0]
 ...,
 [ 58  55   0]
 [ 58  55   0]
 [ 58  55   0]]
```

С помощью метода главных компонент визуализируйте данное изображение в 3D и 2D.

In [15]:
```
fig = pyplot.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(compressedImage[:, 1], compressedImage[:, 2], compresse
pyplot.show()
```



In [16]:
```
def dimensionalityReduction(x):
    m = x.shape[1]
    eigenvectors = getEigenvectors(x)
    v = -eigenvectors[:,:m-1]
    return np.dot(v.T, x.T), v
```

Соответствует ли 2D изображение какой-либо из проекций в 3D?

In [17]:
```
xNew, v = dimensionalityReduction(compressedImage)
pyplot.scatter(xNew[0], xNew[1])
pyplot.show()
```