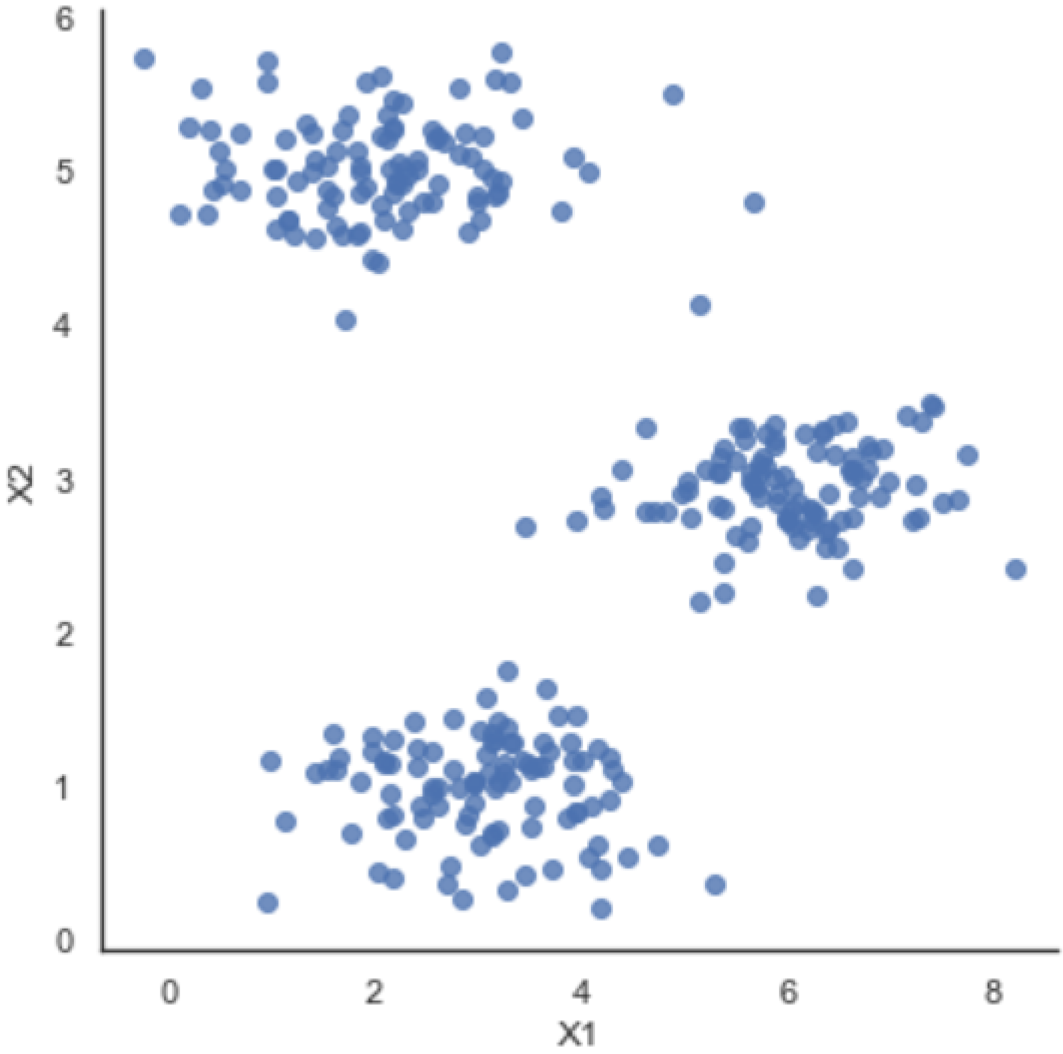
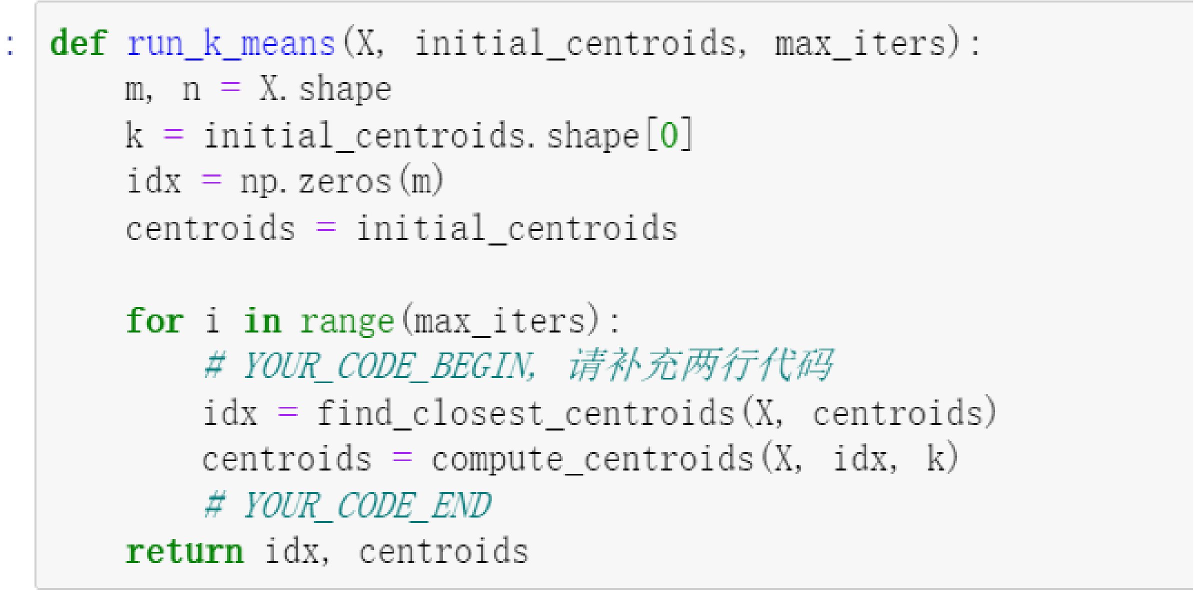
hw7

Kmeans

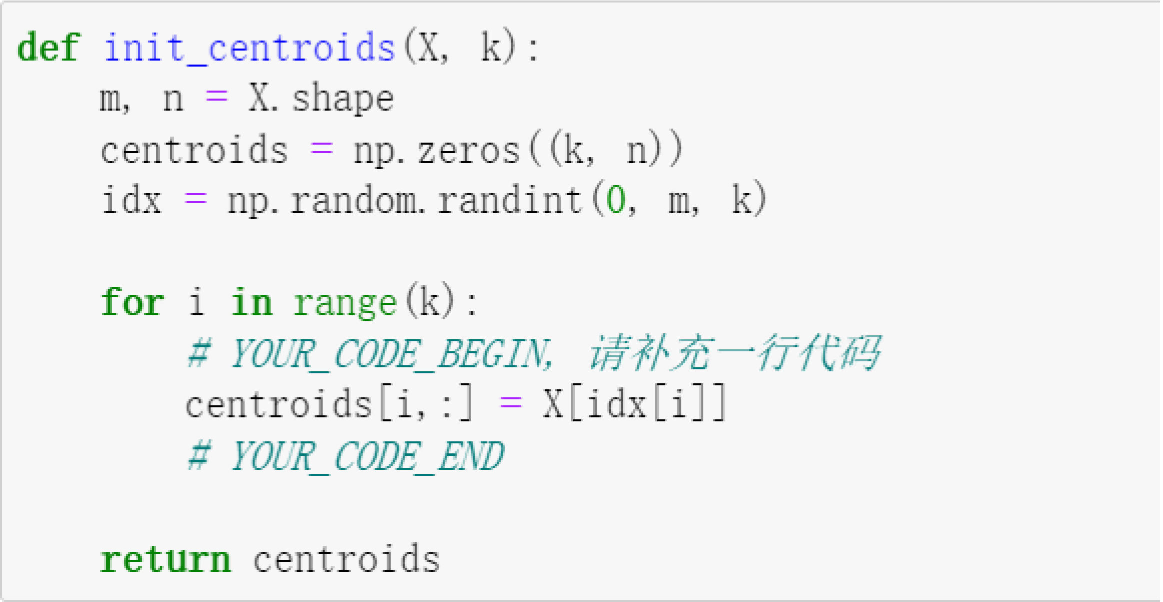
1. data overview



2. code

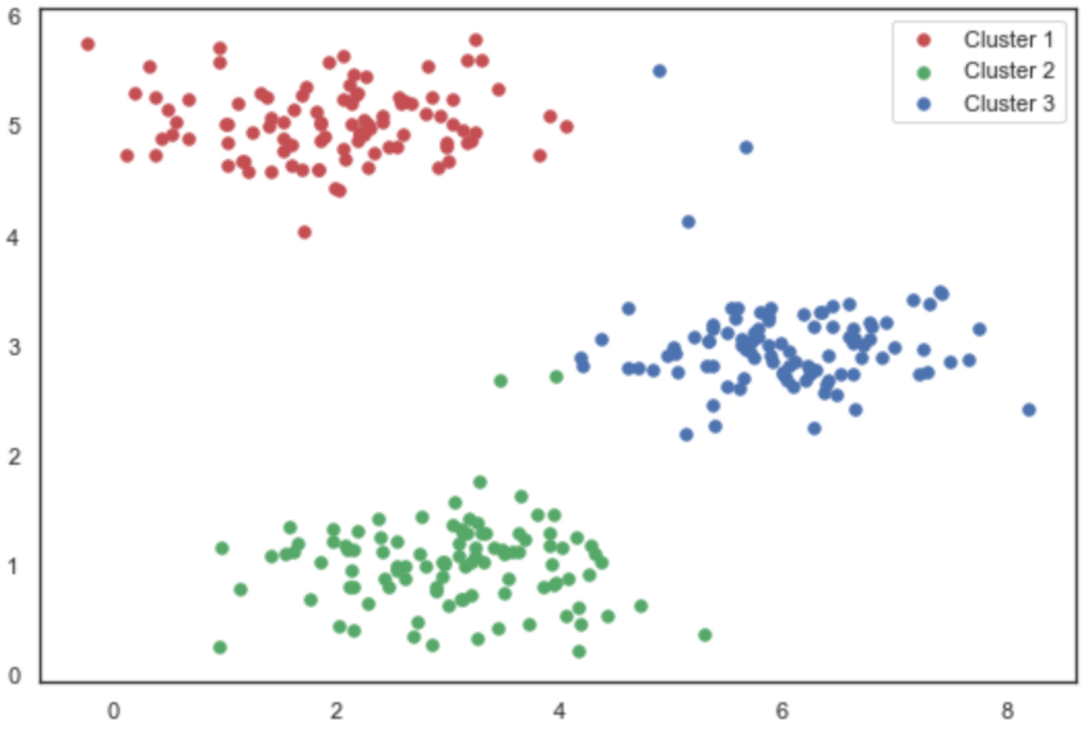


```python
def run_k_means(X, initial_centroids, max_iters):
    m, n = X.shape
    k = initial_centroids.shape[0]
    idx = np.zeros(m)
    centroids = initial_centroids

    for i in range(max_iters):
        # YOUR_CODE_BEGIN, 请补充两行代码
        idx = find_closest_centroids(X, centroids)
        centroids = compute_centroids(X, idx, k)
        # YOUR_CODE_END
    return idx, centroids
```



```python
def init_centroids(X, k):
    m, n = X.shape
    centroids = np.zeros((k, n))
    idx = np.random.randint(0, m, k)

    for i in range(k):
        # YOUR_CODE_BEGIN, 请补充一行代码
        centroids[i,:] = X[idx[i]]
        # YOUR_CODE_END

    return centroids
```

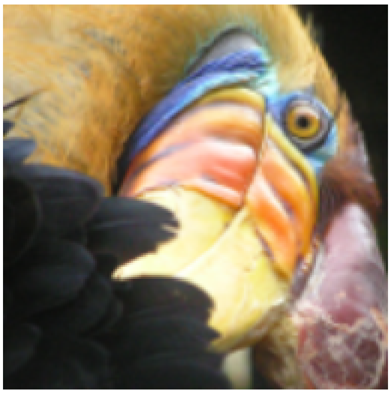
3. result



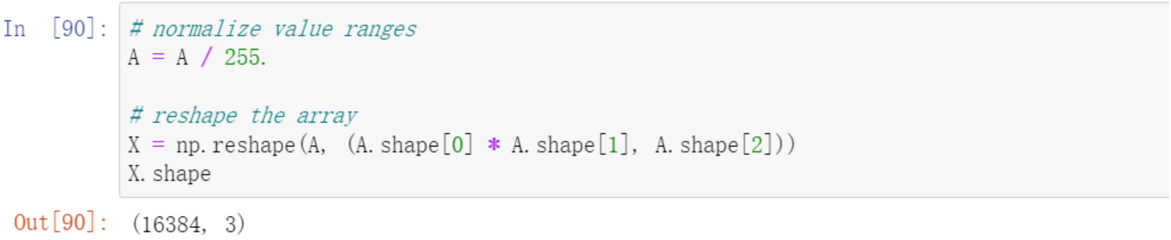
Compress picture using Kmeans

1. original pic

```
from IPython.display import Image
Image(filename='data/bird_small.png')
```



2. data overview

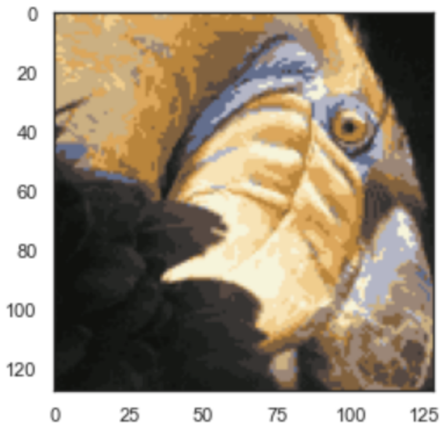


```
In [90]: # normalize value ranges
         A = A / 255.

         # reshape the array
         X = np.reshape(A, (A.shape[0] * A.shape[1], A.shape[2]))
         X.shape

Out[90]: (16384, 3)
```

3. compressed pic

In [19]:

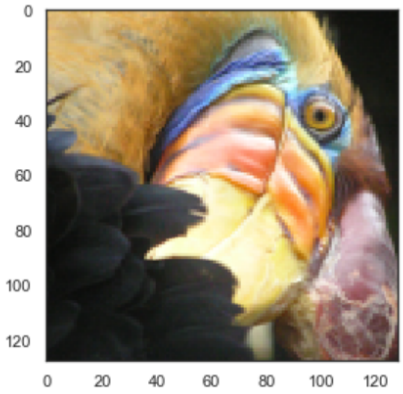
```
plt.imshow(X_recovered)
plt.show()
```



4. using lib

In [20]:

```
from skimage import io

# cast to float, you need to do this otherwise the color would be weird after clustring
pic = io.imread('data/bird_small.png') / 255.
io.imshow(pic)
plt.show()
```



In [101]:

```
pic.shape
```

Out[101]:

```
(128, 128, 3)
```

In [102]:

```
# serialize data
data = pic.reshape(128*128, 3)
```

In [103]:

```
data.shape
```

Out[103]:

```
(16384, 3)
```

In [104]:

```
from sklearn.cluster import KMeans #导入K-Means库

model = KMeans(n_clusters=16, n_init=100)
```

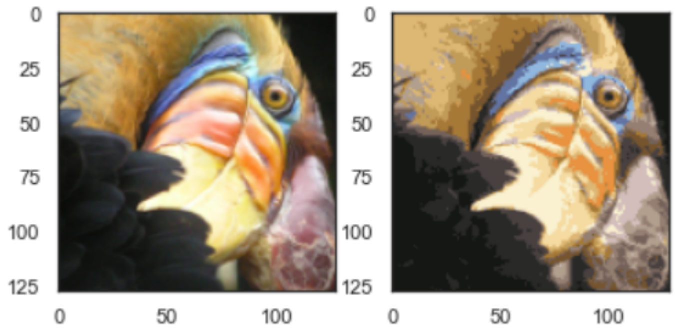
In [31]:

```
model.fit(data)
```

Out[31]:

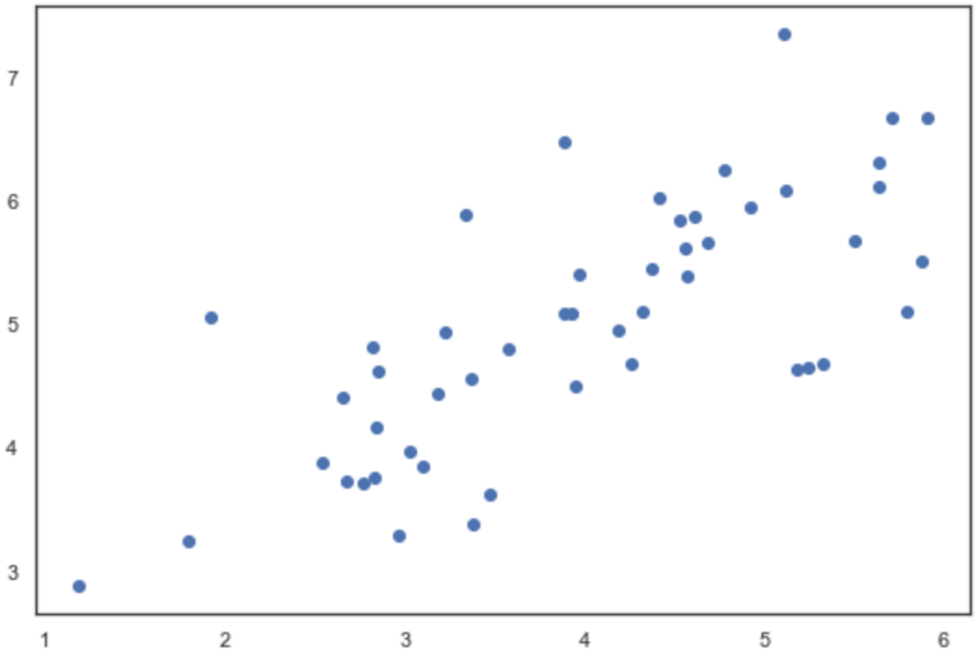
```
KMeans(n_clusters=16, n_init=100)
```

```
fig, ax = plt.subplots(1, 2)
ax[0].imshow(pic)
ax[1].imshow(compressed_pic)
plt.show()
```

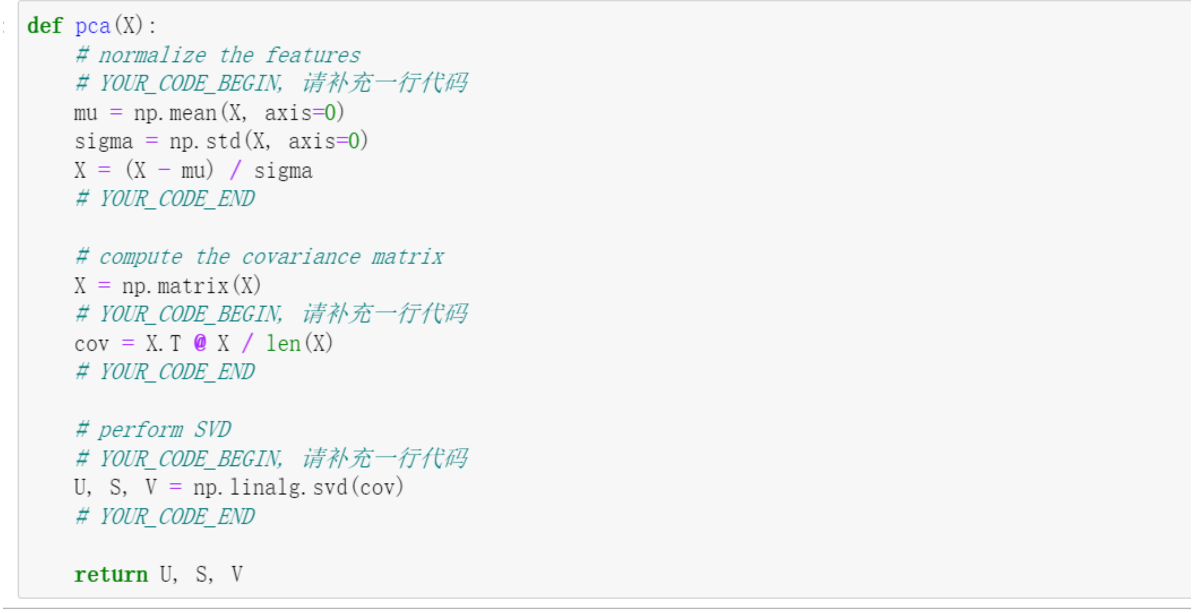


pca

1. data overview

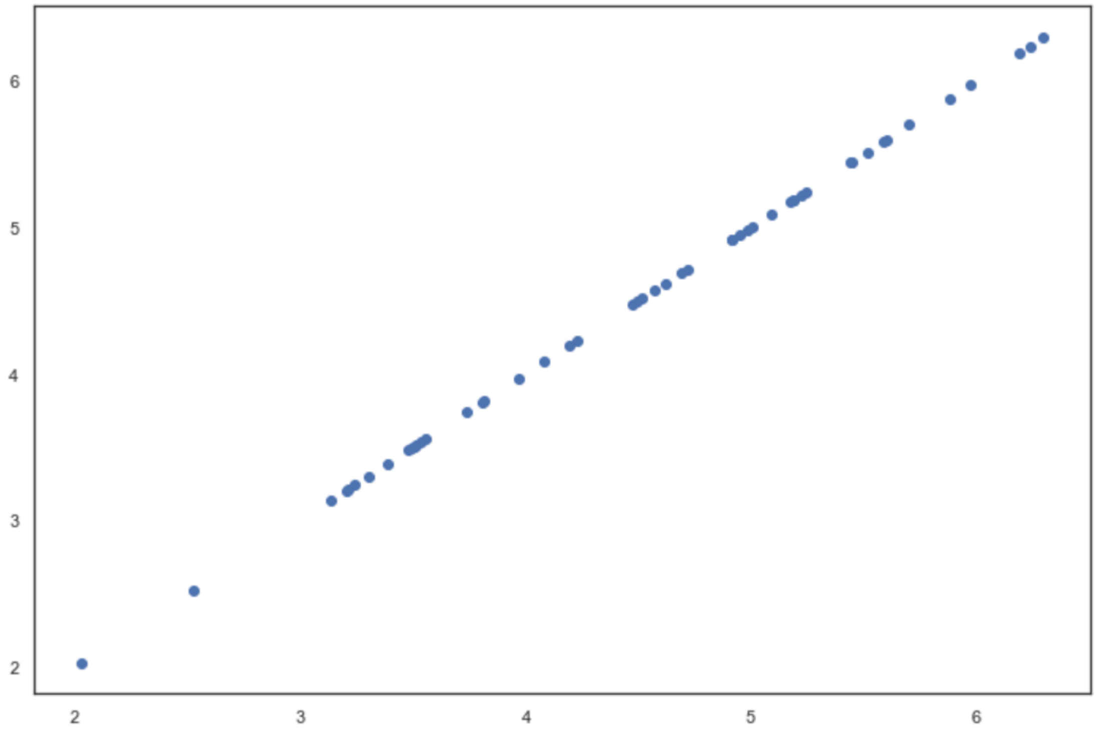


2. algorithm details



```
def pca(X):
    # normalize the features
    # YOUR_CODE_BEGIN, 请补充一行代码
    mu = np.mean(X, axis=0)
    sigma = np.std(X, axis=0)
    X = (X - mu) / sigma
    # YOUR_CODE_END

    # compute the covariance matrix
    X = np.matrix(X)
    # YOUR_CODE_BEGIN, 请补充一行代码
    cov = X.T @ X / len(X)
    # YOUR_CODE_END

    # perform SVD
    # YOUR_CODE_BEGIN, 请补充一行代码
    U, S, V = np.linalg.svd(cov)
    # YOUR_CODE_END

    return U, S, V
```

3. result


6
5
4
3
2
2
3
4
5
6