

«به نام خدا»



هوش مصنوعی و محاسبات زیستی – دکتر حاجی پور
ترم 1402_01

پرنیان طاهری 99106352

تمرین کامپیوتری 3 – سوال 2

در این سوال ابتدا kmeans را به صورت دستی پیاده‌سازی کرده و با تعداد خوش‌های [4, 5, 6] امتحان می‌کنیم و این کار را برای نقاط اولیه مختلف انجام می‌دهیم. سپس این مراحل را برای توابع AgglomerativeClustering و kmeans از کتابخانه sklearn.metrics نیز انجام می‌دهیم و نتایج را با هم مقایسه می‌کنیم.

ابتدا برای درک بهتر، دیتا را پس از لود کردن نمایش می‌دهیم.

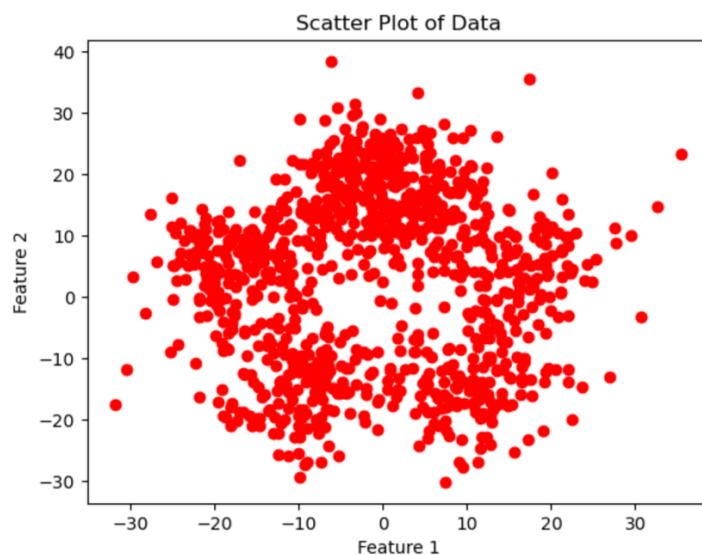
Load Data:

```
# Load .mat file
mat_data = scipy.io.loadmat('DataNew.mat')

DataNew = mat_data['DataNew'].T
print(DataNew.shape)
```

Plot Data:

```
plt.scatter(DataNew[:,1],DataNew[:,0], color='r')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Scatter Plot of Data')
plt.show()
```



Self Written Kmeans Function:

```
def initialize_centers(data, num_clusters):  
  
    # Random initialization  
    indices = np.random.choice(len(data), num_clusters,  
    replace=False)  
    centers = data[indices]  
    return centers  
  
def assign_to_clusters(data, centers):  
  
    # Assign each data point to the closest center  
    distances = np.linalg.norm(data[:, np.newaxis] - centers,  
    axis=2)  
    labels = np.argmin(distances, axis=1)  
    return labels  
  
def update_centers(data, labels, num_clusters):  
  
    # Update the cluster centers  
    centers = np.array([np.mean(data[labels == i], axis=0) for i in  
    range(num_clusters)])  
    return centers  
  
def k_means(data, num_clusters, max_iterations=100):  
  
    # Initialize  
    first_centers = initialize_centers(data, num_clusters)  
    labels = np.zeros(len(data))  
    centers = first_centers  
  
    for _ in range(max_iterations):  
  
        labels = assign_to_clusters(data, centers)  
  
        # Update centers  
        new_centers = update_centers(data, labels, num_clusters)  
  
        # Check for convergence  
        if np.all(centers == new_centers):  
            break  
  
        centers = new_centers  
  
    return labels, centers, first_centers
```

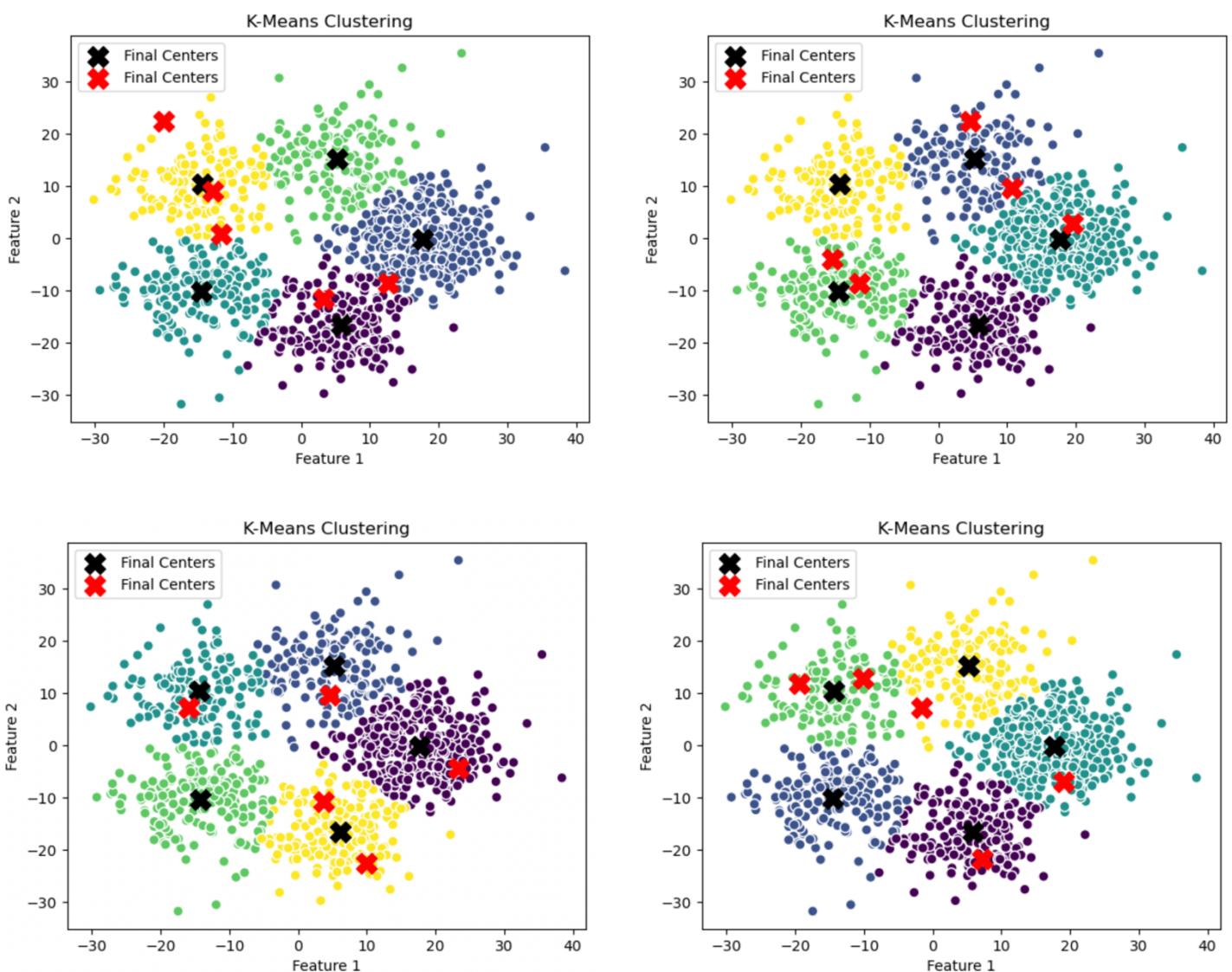
در تابع بالا در قسمت `initialize_centers` به صورت رندم به اندازه تعداد خوشهای مورد نظر نقطه اولیه از نقاط داخل دیتا انتخاب می‌کند. در نتیجه نقاط اولیه ما نقاط جدیدی نیستند بلکه از بین دیتاهای انتخاب می‌شوند. سپس در قسمت `assign_to_clusters` فاصله هر دیتا را تا هر کدام از `center` ها بدست می‌آوریم و مینیمم آن را به عنوان `label` در نظر می‌گیریم سپس برای آپدیت کردن `center` ها در تابع `update_centers` میانگین دیتاهای هر خوش را بدست آورده و مرکز جدید را برابر آن قرار می‌دهیم.

این کار را به اندازه 100 بار انجام می‌دهیم.

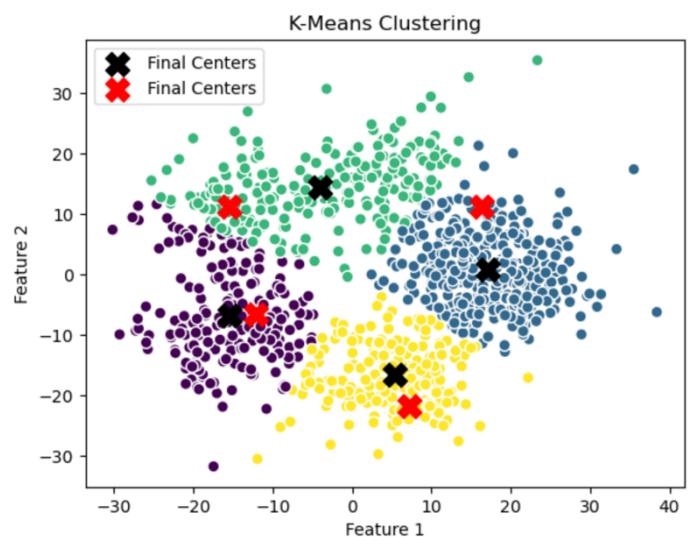
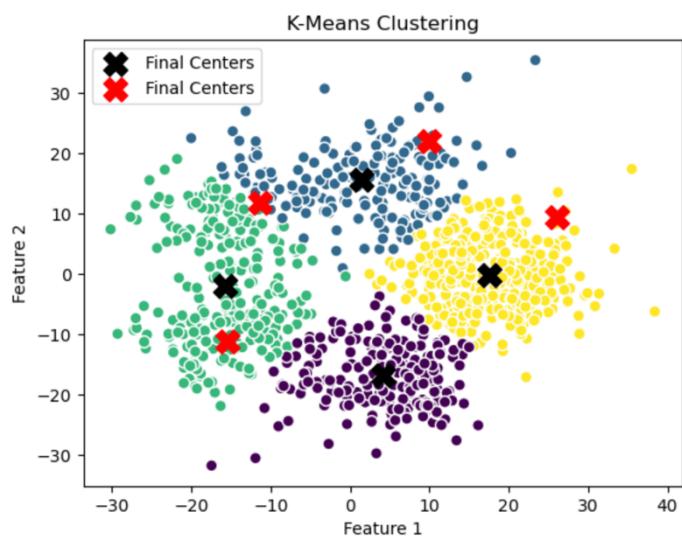
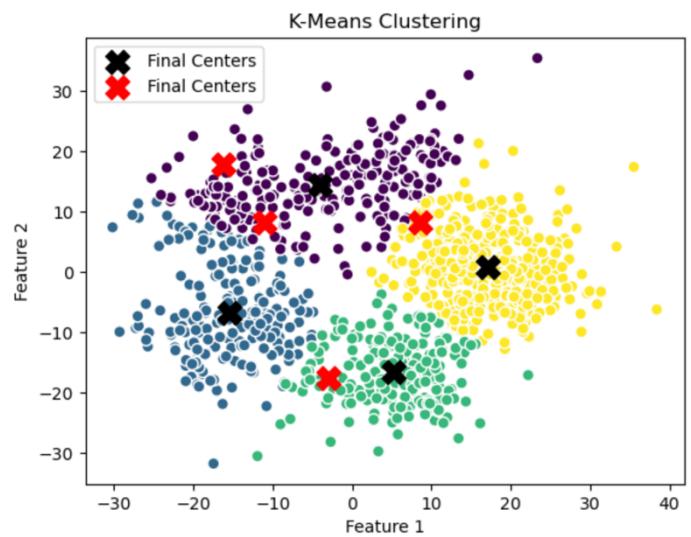
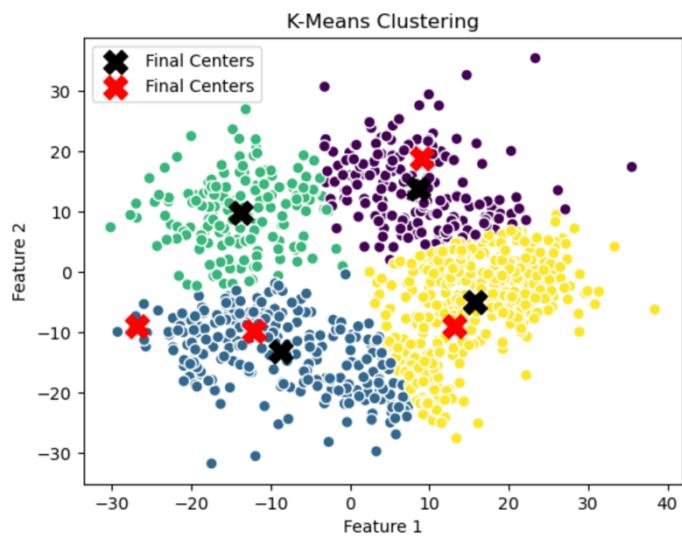
نتایج:

برای هر کدام از دسته بندی ها با تعداد خوشهای [4, 5, 6] ، 4 بار ران می‌کنیم.

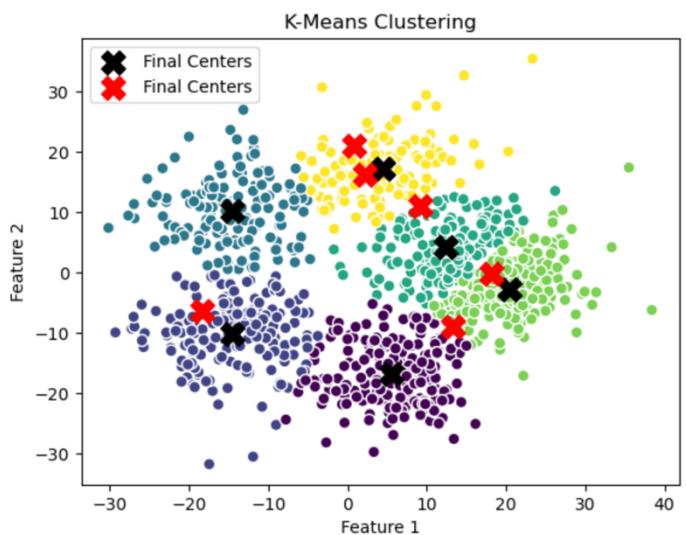
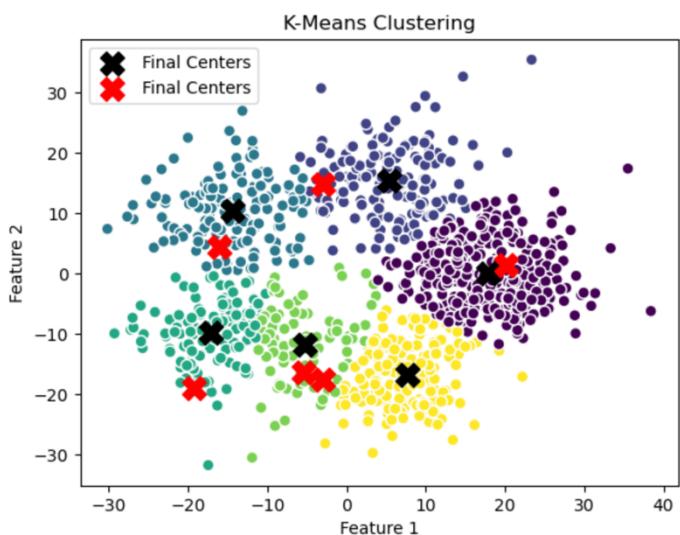
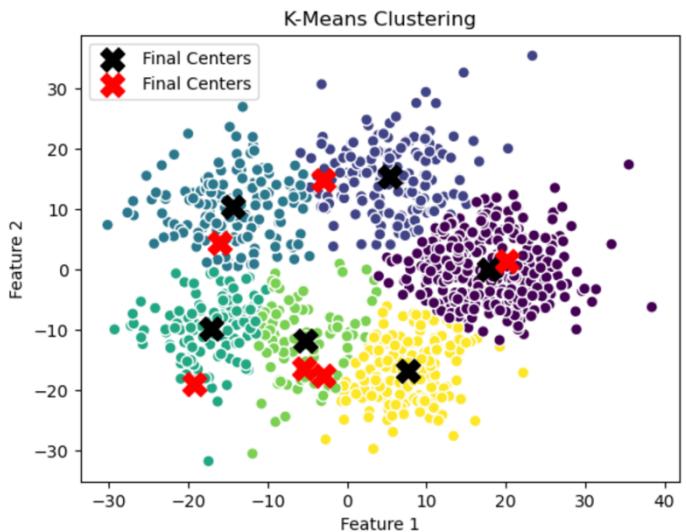
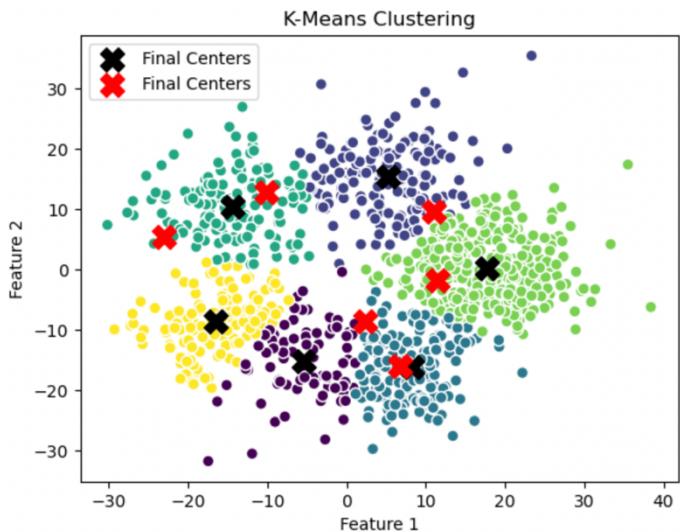
`num_clusters = 5:`



`num_clusters = 4:`



`num_clusters = 6:`



Built-in Kmeans Function

```
def k_means_algorithm(data, num_clusters, initial_centers):  
  
    kmeans = KMeans(n_clusters=num_clusters, init=initial_centers,  
    n_init=1, random_state=42)  
    labels = kmeans.fit_predict(data)  
    centers = kmeans.cluster_centers_  
    return labels, centers
```

num_clusters = 5:

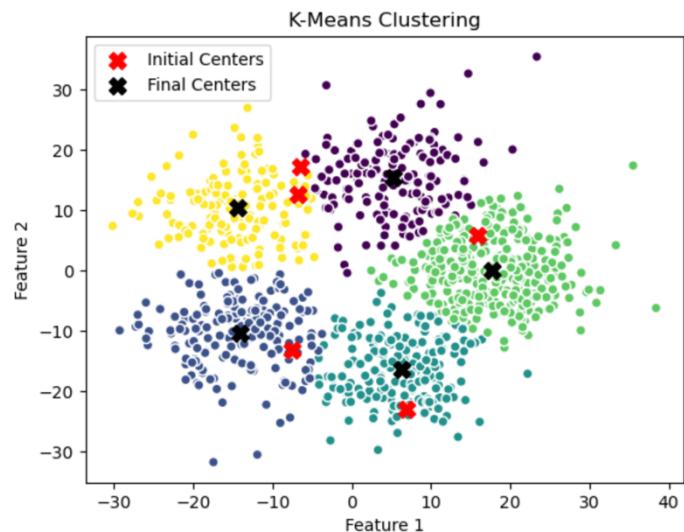
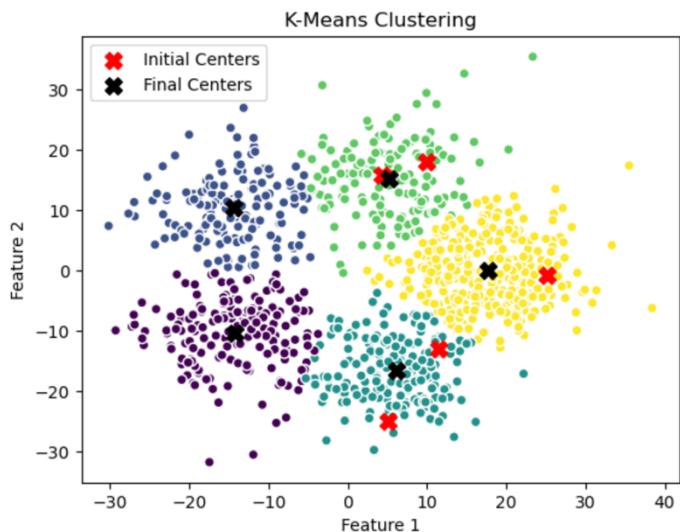
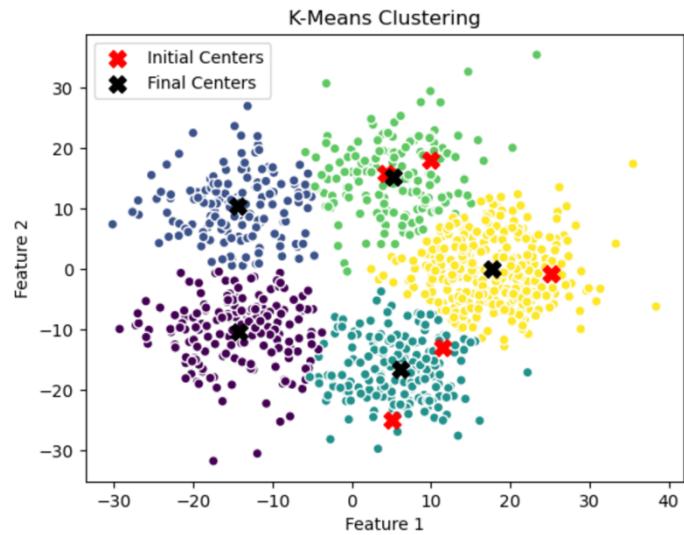
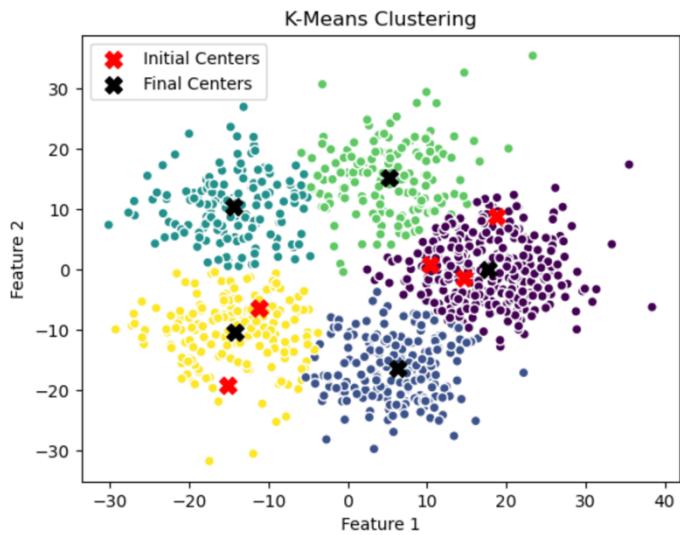
```
num_clusters = 5  
indices = np.random.choice(len(DataNew), num_clusters,  
replace=False)  
initial_centers = DataNew[indices]  
  
cluster_labels, cluster_centers = k_means_algorithm(DataNew,  
num_clusters, initial_centers)  
  
plt.scatter(DataNew[:, 0], DataNew[:, 1], c=cluster_labels,  
edgecolors='w')  
plt.scatter(initial_centers[:, 0], initial_centers[:, 1], c='r',  
marker='X', s=100, label='Initial Centers')  
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1],  
c='black', marker='X', s=100, label='Final Centers')  
plt.title('K-Means Clustering')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.legend()  
plt.show()
```

در این قسمت از تابع Kmeans از کتابخانه `sklearn.cluster` پایتون استفاده می‌کنیم.
ابتدا در تابع `k_means_algorithm`، تعداد خوشها و نقاط اولیه را به تابع Kmeans می‌دهیم و `label` و `center` را می‌گیریم.

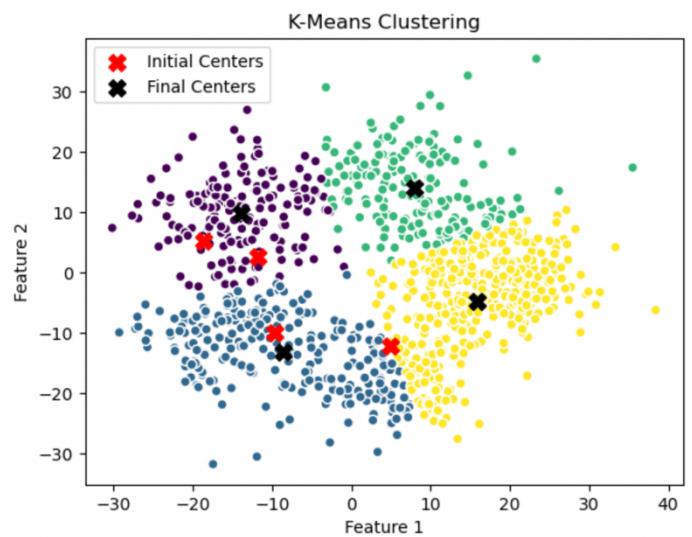
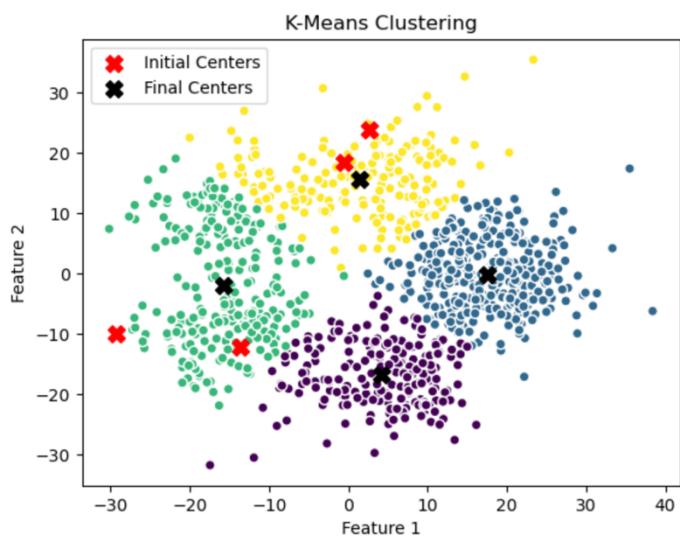
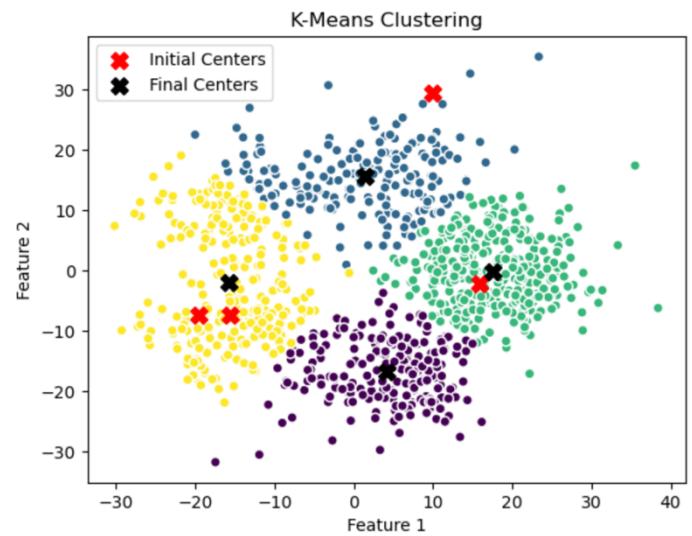
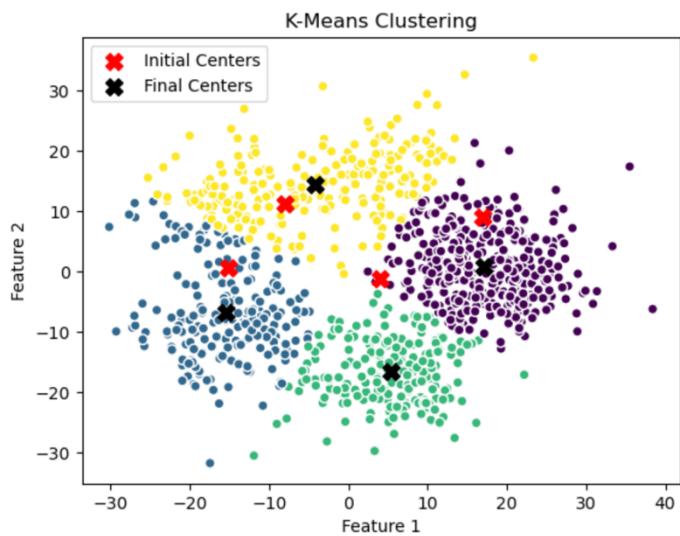
در این جا تابعی که برای قسمت 5 کلاستر نوشتیم را آوردم بای بقیه کلاسترها هم به همین ترتیب است.
ابتدا مانند قبل نقاط اولیه را به صورت رندم از بین دیتاهای انتخاب می‌کنیم و به تابعی که توضیح داده شد
پاس می‌دهیم.

نتائج:

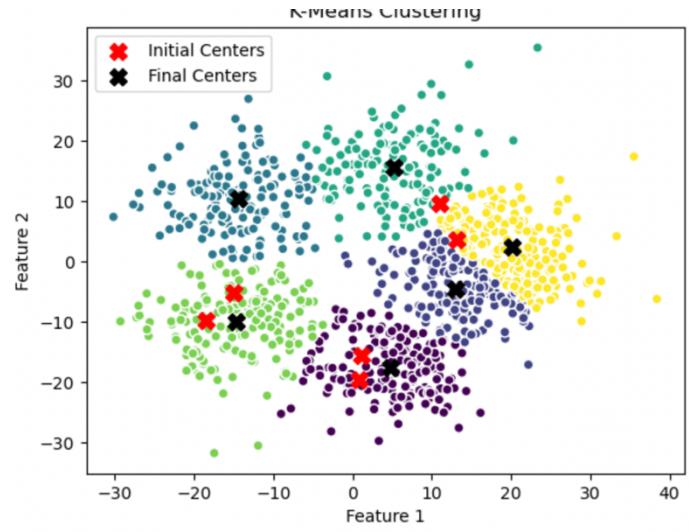
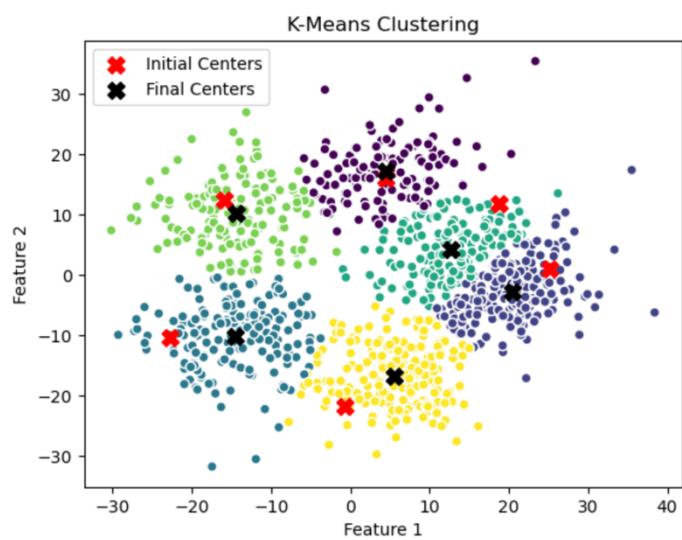
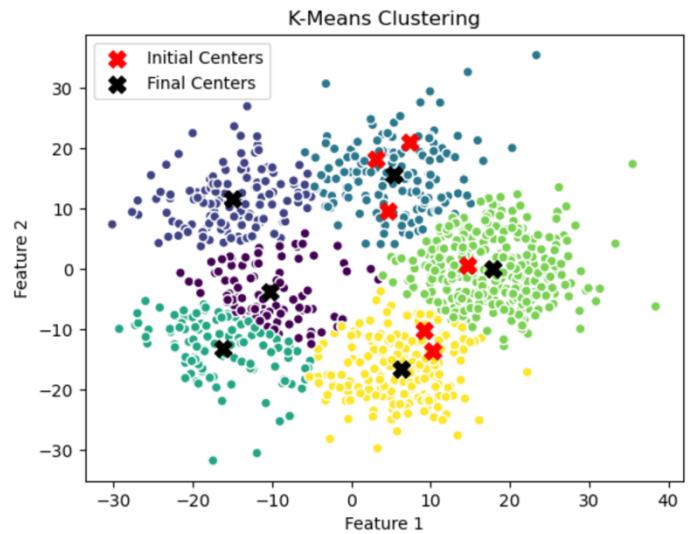
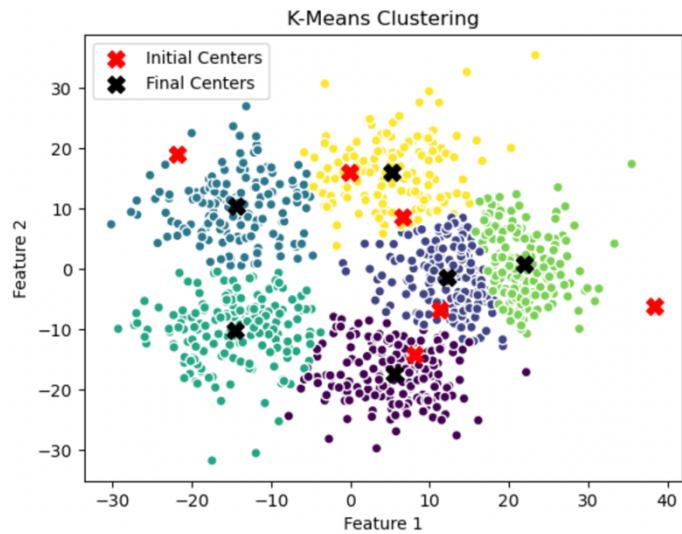
num_clusters = 5:



`num_clusters = 4`



`num_clusters = 6`



مقایسه:

در هر دو روش نتایج تقریباً یکسان است و بهترین دسته بندی برای دسته بندی با 5 خوش است که در هر مرحله ران شدن ثابت مانده است.

در قسمت 'ه' از تابع آماده `AgglomerativeClustering` از کتابخانه `sklearn.cluster` استفاده می‌کنیم.

```
def get_cluster_representatives(data, labels):  
  
    unique_labels = np.unique(labels)  
    representatives = []  
  
    for label in unique_labels:  
        cluster_points = data[labels == label]  
        cluster_representative = np.mean(cluster_points, axis=0)  
        representatives.append(cluster_representative)  
    return np.array(representatives)
```

`num_clusters = 5:`

```
num_clusters = 5  
  
# Hierarchical clustering  
model = AgglomerativeClustering(n_clusters=num_clusters)  
labels = model.fit_predict(DataNew)  
  
# Get centroid  
cluster_centroids = get_cluster_representatives(DataNew, labels)  
  
# Plot  
plt.scatter(DataNew[:, 0], DataNew[:, 1], c=labels,  
cmap='viridis', marker='o', edgecolors='w')  
plt.scatter(cluster_centroids[:, 0], cluster_centroids[:, 1],  
c='red', marker='X', s=100, label='Cluster Centroids')  
plt.title('Agglomerative Hierarchical Clustering')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.show()
```

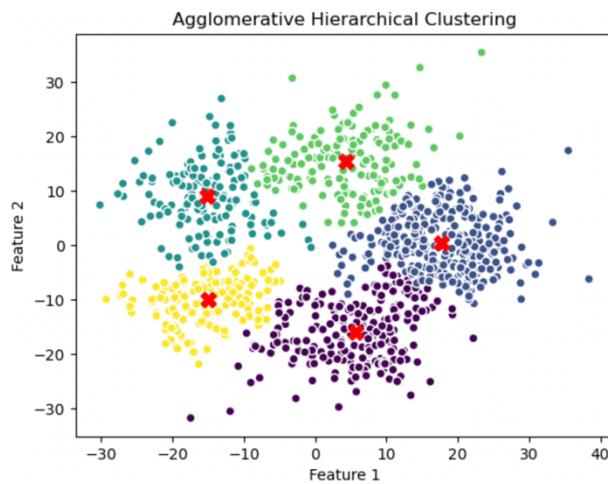
در تابع `get_cluster_representatives` می‌خواهیم `centroid`‌ها را بدست بیاوریم. برای اینکار میانگین دیتاهای داخل هر خوش را به عنوان `centroid` انتخاب می‌کنیم.

در این جا تابعی که برای قسمت 5 کلاستر نوشتیم را آوردم بای بقیه کلاسترها هم به همین ترتیب است. ابتدا مدل را برای تعداد کلاسترهای موردنظر بدست می‌آوریم و سپس روی دیتاهای موردنظرمان فیت می‌کنیم. سپس با استفاده از تابعی که توضیح داده شد `centroid`‌ها را بدست می‌آوریم و رسم می‌کنیم.

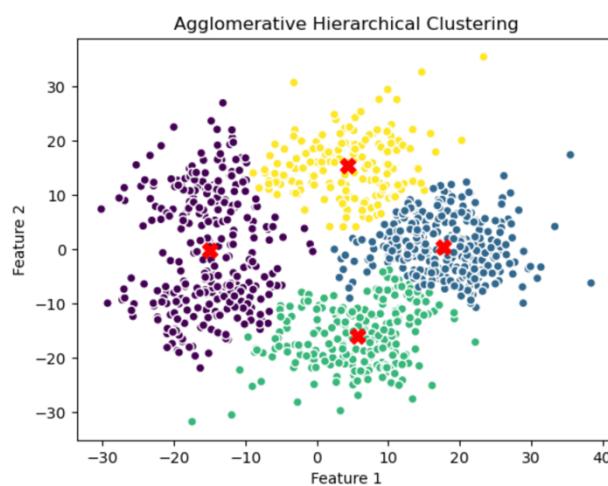
در این روش به دلیل اینکه نقاط اولیه رندم انتخاب نمی‌شود، هرچه ران کنیم نتیجه یکسان است.

نتائج:

num_clusters = 5



num_clusters = 4



num_clusters = 6

