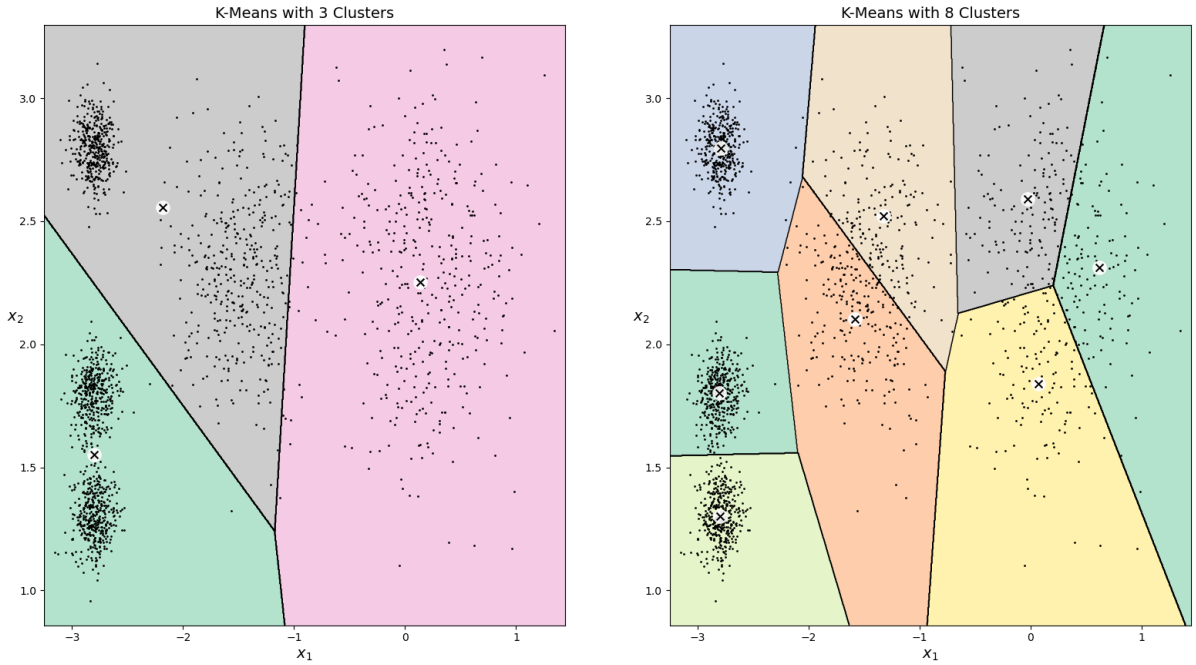
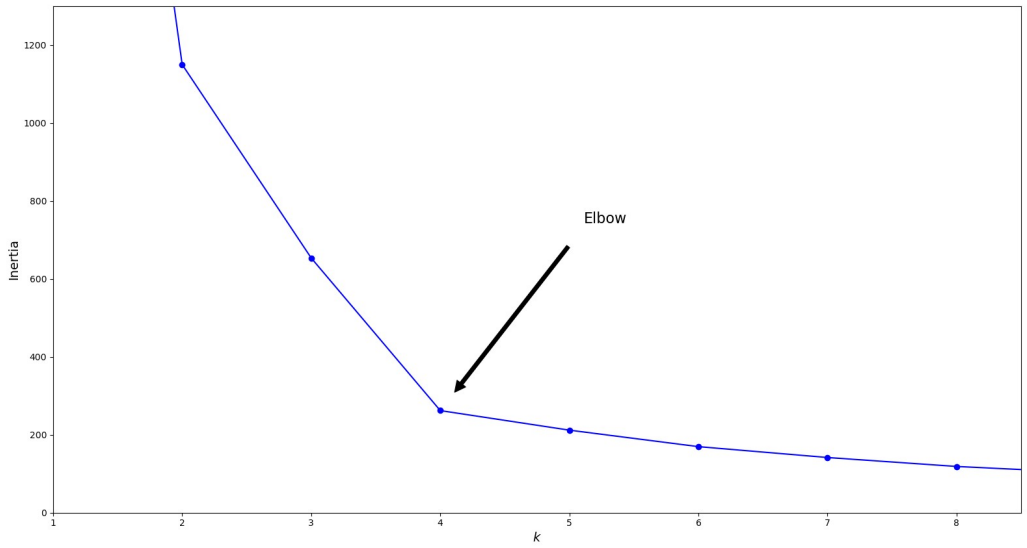
Q1-3. Finding an Optimum Number of Clusters for K-Means Clustering

```
import numpy as np
import os

import matplotlib as mpl
import matplotlib.pyplot as plt

import sklearn
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

########################################################################
### Determine Optimum Number of Clusters for Effective K-Means Clustering ###
########################################################################

### Prepare the dataset ##############################################################################################################################
# Blob centers
blob_centers = np.array([[ 0.2,  2.3],
                         [-1.5 ,  2.3],
                         [-2.8,  1.8],
                         [-2.8,  2.8],
                         [-2.8,  1.3]])

# Blob standard deviation to define the distribution of dataset
blob_std = np.array([0.4, 0.3, 0.1, 0.1, 0.1])

# Produce the blob with centers and standard deviation
X, y = make_blobs(n_samples=2000, centers=blob_centers,
                  cluster_std=blob_std, random_state=7)

### Prepare K-Means clusterer with diffferent number of clusters #####################################################################################
kmeans_k3 = KMeans(n_clusters=3, random_state=42)   # K-Means clusterer with 3 clusters
kmeans_k8 = KMeans(n_clusters=8, random_state=42)   # K-Means clusterer with 8 clusters

kmeans_k3.fit(X)    # Fit
kmeans_k8.fit(X)

### K-Means clustering reuslt plotting ###############################################################################################################
# Segment K-Means clusters by clustering all the points within X range and Y range
# Acquire the value range of x1 and x2
mins = X.min(axis=0) - 0.1
maxs = X.max(axis=0) + 0.1

# Acquire all the (x1, x2) points within the range
xx, yy = np.meshgrid(np.linspace(mins[0], maxs[0], 1000),
                     np.linspace(mins[1], maxs[1], 1000))

# Subplot 1 ##########################################################################################################################################
plt.subplot(121)
plt.title("K-Means with 3 Clusters", fontsize=14)

# Cluster all the (x1, x2) points within the range / Cluster labels are used as height of contour
Z = kmeans_k3.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)                             # Reshape for plotting

# Color all the (x1, x2) points with height according to cluster label
plt.contourf(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]), cmap="Pastel2")

# Connect and draw the contour lines
plt.contour(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]), linewidths=1, colors='k')

plt.scatter(X[:, 0], X[:, 1], c='k', s=1)     # Plot the data on the contour

# Draw the centroids of K-Means clustering results
plt.scatter(kmeans_k3.cluster_centers_[:, 0], kmeans_k3.cluster_centers_[:, 1], marker='o', s=30, linewidths=8, color='w', zorder=10, alpha=0.9)
plt.scatter(kmeans_k3.cluster_centers_[:, 0], kmeans_k3.cluster_centers_[:, 1], marker='x', s=50, color='k', zorder=11, alpha=1)

plt.xlabel("$x_1$", fontsize=14)
plt.ylabel("$x_2$", fontsize=14, rotation=0)


# Subplot 2 ##########################################################################################################################################
plt.subplot(122)
plt.title("K-Means with 8 Clusters", fontsize=14)

# Cluster all the (x1, x2) points within the range / Cluster labels are used as height of contour
Z = kmeans_k8.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)                             # Reshape for plotting

# Color all the (x1, x2) points with height according to cluster label
plt.contourf(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]), cmap="Pastel2")

# Connect and draw the contour lines
plt.contour(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]), linewidths=1, colors='k')

plt.scatter(X[:, 0], X[:, 1], c='k', s=1)     # Plot the data on the contour

# Draw the centroids of K-Means clustering results
plt.scatter(kmeans_k8.cluster_centers_[:, 0], kmeans_k8.cluster_centers_[:, 1], marker='o', s=30, linewidths=8, color='w', zorder=10, alpha=0.9)
plt.scatter(kmeans_k8.cluster_centers_[:, 0], kmeans_k8.cluster_centers_[:, 1], marker='x', s=50, color='k', zorder=11, alpha=1)

plt.xlabel("$x_1$", fontsize=14)
plt.ylabel("$x_2$", fontsize=14, rotation=0)

plt.show()

### Determine optimum number of cluster with the inertia of K-Means clustering ###########################################################################
print('Inertia of K-Means with 3 clusters : {}'.format(kmeans_k3.inertia_))
print('Inerita of K-Means with 8 clusters : {}'.format(kmeans_k8.inertia_))

# Acquire an optimum number of clusters using elbow of K-Means inertia graph

# K-Means clustering with various number of clusters
kmeans_per_k = [KMeans(n_clusters=k, random_state=42).fit(X) for k in range(1, 10)]

# Save the inertia of each K-Means results
inertias = [model.inertia_ for model in kmeans_per_k]

plt.figure(figsize=(8, 3.5))
plt.plot(range(1, 10), inertias, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Inertia", fontsize=14)
plt.annotate('Elbow',
             xy=(4, inertias[3]),
             xytext=(0.55, 0.55),
             textcoords='figure fraction',
             fontsize=16,
             arrowprops=dict(facecolor='black', shrink=0.1)
            )
plt.axis([1, 8.5, 0, 1300])
plt.show()
# This plot shows that 4 cluster is an optimum number of clusters for current K-Means settting

### Optimum K-Means clustering reuslt plotting ###############################################################################################################
plt.title("K-Means with 4 Clusters (Inertia {})".format(kmeans_per_k[3].inertia_), fontsize=14)

# Cluster all the (x1, x2) points within the range / Cluster labels are used as height of contour
Z = kmeans_per_k[3].predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)          # Reshape for plotting

# Color all the (x1, x2) points with height according to cluster label
plt.contourf(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]), cmap="Pastel2")

# Connect and draw the contour lines
plt.contour(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]), linewidths=1, colors='k')

plt.scatter(X[:, 0], X[:, 1], c='k', s=1)     # Plot the data on the contour

# Draw the centroids of K-Means clustering results
plt.scatter(kmeans_per_k[3].cluster_centers_[:, 0], kmeans_per_k[3].cluster_centers_[:, 1], marker='o', s=30, linewidths=8, color='w', zorder=10, alpha=0.9)
```

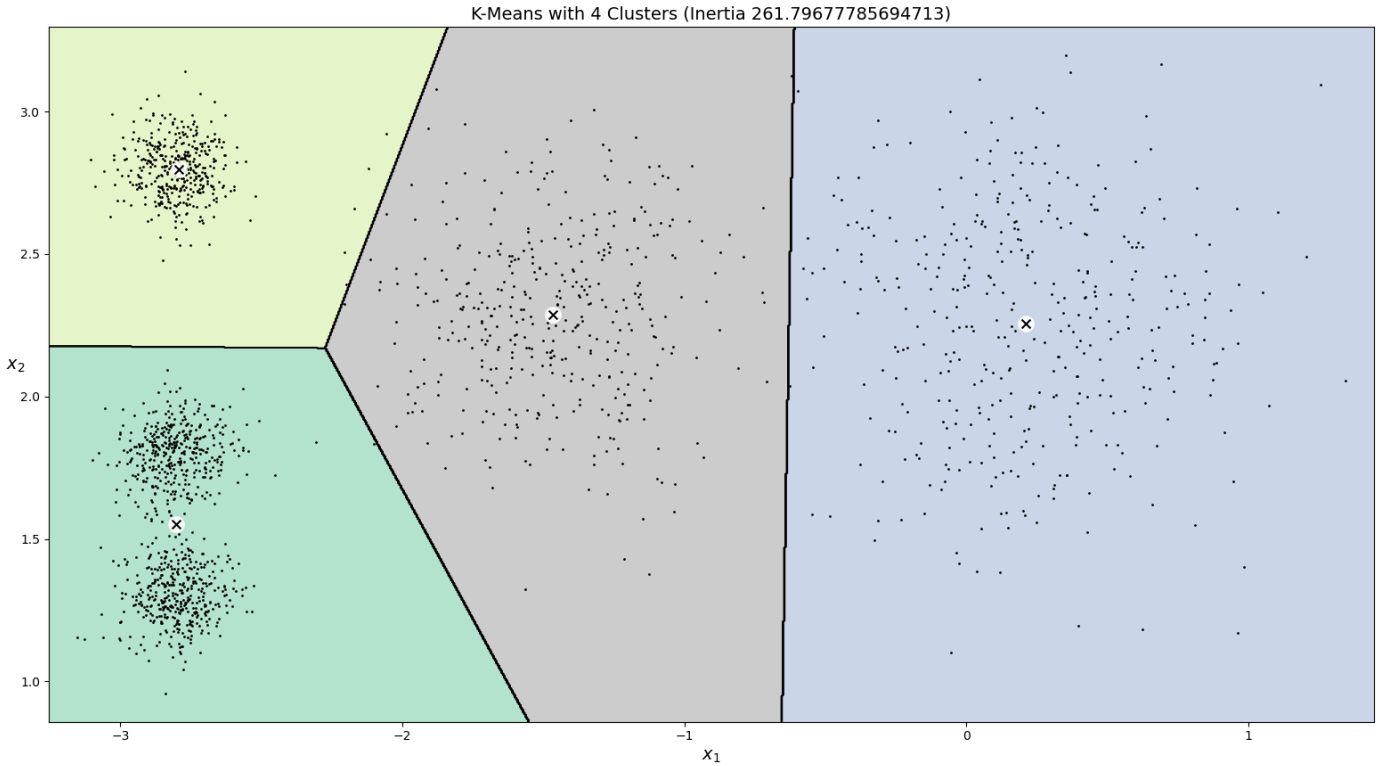
```
plt.scatter(kmeans_per_k[3].cluster_centers_[:, 0], kmeans_per_k[3].cluster_centers_[:, 1], marker='x', s=50, color='k', zorder=11, alpha=1)

plt.xlabel("$x_1$", fontsize=14)
plt.ylabel("$x_2$", fontsize=14, rotation=0)

plt.show()
```



- **K-Means Clustering 에서는 사용하는 Cluster/Centroid 의 개수에 따라 Clustering 결과가 달라짐**
- **Cluster 3 개를 사용한 경우에는 그보다 많은 데이터의 밀집을 표현하지 못하는 상황이 발생함 (1, 1 Subplot 의 회색 영역 & 녹색 영역)**
- **Cluster 8 개를 사용한 경우에는 하나의 밀집이 여러 구역으로 나눠지는 것을 볼 수 있음 (1, 2 Subplot 의 주황색 영역 vs 살색 영역 / 회색 영역 vs 노란색 영역 vs 우측 녹색 영역)**



- **K-Means Clustering 에서 최적의 Cluster 개수를 판단하기 위해 Objective Function (Inertia) (클러스터 내 데이터의 포함 여부를 결정하는 함수 (ex : Euclidean Distance)) 의 변화를 확인함**
- **K-Means Clustering 의 Objective Function 이 급격하게 감소하는 순간을 ‘ Elbow’ 라 지칭하며 이 지점이 데이터가 제대로 분리되지 못하는 순간과 과도하게 분리되는 경계지점으로 해석할 수 있음**
- **본 예제에서는 Cluster 4 개가 Elbow 지점의 Cluster 개수이며 최적의 Cluster 개수라 추정함**



- **Cluster 4 개를 사용한 경우 3 개를 사용한 경우보다 각 Centroid 가 주요 데이터 밀집 지점에 잘 배치되는 것을 볼 수 있음 . 그리고 8 개를 사용한 경우보다 밀집 지점을 과도하게 나누지 않는 것을 볼 수 있음 .**

- **그러나 주요 밀집 지점이 총 5 개인 상황에서 그보다 적은 4 개를 사용하기 때문에 좌측 하단 녹색 영역과 같이 Centroid 가 애매하게 배치되는 것을 볼 수 있음 .**

- **Cluster 개수에 따른 Objective Function (Inertia) 그래프에서 Elbow 는 무조건 최적의 Cluster 개수가 아니지만 찾는 최초 지표로서 사용될 수 있음 .**