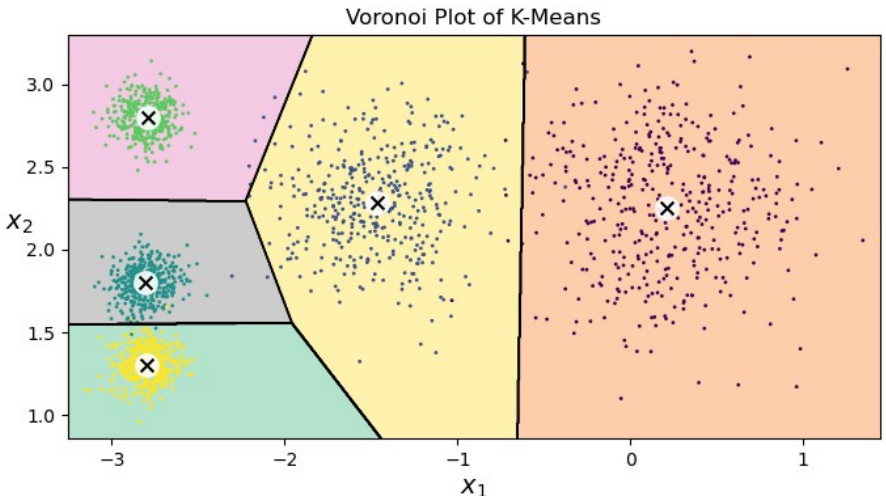
Q1-1. K-Means Clustering on Random Dataset

```
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

##########################################
### K-Means clustering on random dataset ###
##########################################

### Prepare blob 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)

### K-Means Clustering Fitting ######################################################################################################################
k = 5
kmeans = KMeans(n_clusters=k, random_state=42)
y_pred = kmeans.fit_predict(X)

print(y_pred)
print(kmeans.labels_)
print(y_pred is kmeans.labels_)

print(kmeans.cluster_centers_)

### K-Means Clustering Result Plotting ##############################################################################################################
plt.figure(figsize=(8, 4))

# 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))

# Cluster all the (x1, x2) points within the range / Cluster labels are used as height of contour
Z = kmeans.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=y, s=1)     # Plot the data on the contour

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

plt.show()

### Hard Clustering of K-Means ######################################################################################################################
X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]])  # Prepare a new random dataset

# Show each input data's distance to each cluster
print(kmeans.transform(X_new))

# Show each input data's euclidean distance to each cluster
print(np.linalg.norm(np.tile(X_new, (1, k)).reshape(-1, k, 2) - kmeans.cluster_centers_, axis=2))

# This shows that KMeans.transfrom() shows each data's euclidean distance to each cluster
```



- **K-Means Clustering 의 경우 Cluster 중심점 (Centroid, Codebook) 과 데이터간의 거리 (Distance or Error) 가 최소가 될 때까지 주변 데이터의 Mean 을 향해서 움직이게됨 .**
- **K-Means 를 이용하여 계속 Clustering 을 수행하게되면 Cluster 의 중심점이 주변 데이터 집합의 평균 /Mean 이 되는 중심점 근처로 이동하여 배치되는 것을 볼 수 있음 .**
- **각 데이터는 제일 가까운 Cluster 중심점에 Assign 되어 Labeling 됨 .**