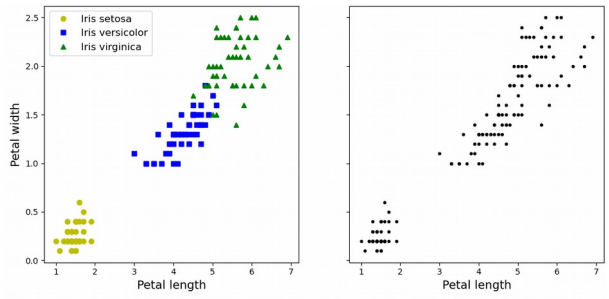
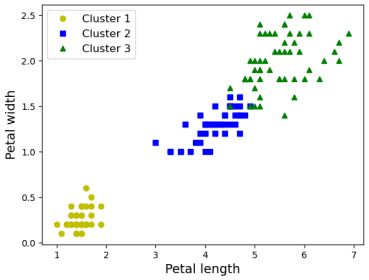
Q2. Gaussian Mixture Model

```
import numpy as np
import os

import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm

import sklearn
from sklearn.datasets import load_iris
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_blobs

########################################################
### Iris Data Clustering using Gaussain Mixture Model ###
########################################################

### Prepare Iris dataset ###############################################################################################################################
data = load_iris()
X = data.data                # X : Input data of Iris dataset
y = data.target              # y : Target label of Iris dataset

print(data.target_names)     # Print target names

# Plot Iris dataset
plt.figure(figsize=(9, 3.5))

plt.subplot(121)
plt.plot(X[y==0, 2], X[y==0, 3], "yo", label="Iris setosa")
plt.plot(X[y==1, 2], X[y==1, 3], "bs", label="Iris versicolor")
plt.plot(X[y==2, 2], X[y==2, 3], "g^", label="Iris virginica")
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.legend(fontsize=12)

plt.subplot(122)
plt.scatter(X[:, 2], X[:, 3], c="k", marker=".")
plt.xlabel("Petal length", fontsize=14)
plt.tick_params(labelleft=False)
plt.show()
```




```
### Gaussian Mixture Model (GMM) for clustering Iris dataset ###########################################################################################
GMM = GaussianMixture(n_components=3, n_init=10, random_state=42).fit(X)     # Fit GMM with current Iris dataset

y_pred = GMM.predict(X)      # Cluster Iris dataset

# Re-Organize the clustering label according to target data label
mapping_index = [np.argmax(np.bincount(y_pred[n:n+50])) for n in range(0, 150, 50)]
mapping = {mapping_index[i]:i for i in [0, 1, 2]}
y_pred = np.array([mapping[cluster_id] for cluster_id in y_pred])

# Plot Iris dataset differently according to its cluster number
plt.plot(X[y_pred==0, 2], X[y_pred==0, 3], "yo", label="Cluster 1")
plt.plot(X[y_pred==1, 2], X[y_pred==1, 3], "bs", label="Cluster 2")
plt.plot(X[y_pred==2, 2], X[y_pred==2, 3], "g^", label="Cluster 3")
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.legend(loc="upper left", fontsize=12)
plt.show()

print('Number of Correct Predictions : {}'.format(np.sum(y==y_pred)))

print('Accuracy : {}'.format(np.sum(y==y_pred) / len(y_pred)))
```



GMM 을 통한 Iris Dataset Clustering 에 의한 결과가 원본 데이터셋의 Label 분포와 유사한 것을 볼 수 있음

```
########################################################
### Random Data Clustering using Gaussian Mixture Model ###
########################################################

### Prepare random dataset with blob distribution ######################################################################################################
X1, y1 = make_blobs(n_samples=1000, centers=((4, -4), (0, 0)), random_state=42)
X1 = X1.dot(np.array([[0.374, 0.95], [0.732, 0.598]]))
X2, y2 = make_blobs(n_samples=250, centers=1, random_state=42)
X2 = X2 + [6, -8]
X = np.r_[X1, X2]
y = np.r_[y1, y2]

### Gaussian Mixture Model (GMM) for given random dataset ##############################################################################################
GMM = GaussianMixture(n_components=3, n_init=10, random_state=42)

GMM.fit(X)  # Fit GMM with given random dataset

print('GMM weights : {}'.format(GMM.weights_))
print('GMM means : {}'.format(GMM.means_))
print('Is GMM converged? : {}'.format(GMM.converged_))
print('Number of convergence in GMM : {}'.format(GMM.n_iter_))
print('Probability Density Function of each point in dataset : {}'.format(GMM.score_samples(X)))

### Plotting GMM data clusters as countours according to their probability distribution ################################################################
# Segment GMM 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))

plt.title("GMM with 3 Components", fontsize=14)

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

# Color all the (x1, x2) points with normalized color according to cluster label
plt.contourf(xx, yy, Z, norm=LogNorm(vmin=1.0, vmax=30.0), levels=np.logspace(0, 2, 12))

# Connect and draw the contour lines
plt.contour(xx, yy, Z, norm=LogNorm(vmin=1.0, vmax=30.0), levels=np.logspace(0, 2, 12), 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(GMM.means_[:, 0], GMM.means_[:, 1], marker='o', s=30, linewidths=8, color='w', zorder=10, alpha=0.9)
plt.scatter(GMM.means_[:, 0], GMM.means_[:, 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()
```