Lab 10. Clustering using Kmeans and Gaussian Mixture Model (GMM)

The **MNIST database** is a large database of handwritten digits (0-9) that is commonly used for training various image processing systems. The original MNIST database contains 60,000 training images and 10,000 testing images. You can use the function "**`sklearn.datasets.load_digits`**" to download a subset of this data set.

1. Apply PCA and select first two directions to convert the data in to 2D. (Code snippet is given below).

2. Apply K-means (K=10) clustering on the reduced data. Plot the data points in these clusters (use different colours for each cluster). Obtain the sum of squared distances of samples to their closest cluster centre.
(Use **`kmeans.fit`** to train the model and **`kmeans.labels_`** to obtain the cluster labels).

3. Build a GMM with 10 components (use **`GMM.fit`**) on the reduced data. Use this GMM to cluster the data points (Use **`GMM.predict`**). Plot the points in these clusters.

4. Obtain the purity score for the clustering (K=10).

5. Repeat part 2 and 3 for K= 5, 8, 10, 12, 15, 17 and Find the optimum number of clusters (K) using Elbow method for K-means and GMM.

```
##########################################

# To load and reduce MNIST data

digits = load_digits()
data = scale(digits.data)

pca = PCA(n_components=10).fit(data)
reduced_data = PCA(n_components=2).fit_transform(data)

###############
```

 # Purity score
Get y_true using the original file (y_true = digits.target)

```
from sklearn import metrics

def purity_score(y_true, y_pred):
  # compute contingency matrix (also called confusion matrix)
  contingency_matrix = metrics.cluster.contingency_matrix(y_true, y_pred)
  # return purity
  Return np.sum(np.amax(contingency_matrix,axis=0))/np.sum(contingency_matrix)
```