Unit 3.3 Graded Assignment:
Muhammad Khan (2303.KHI.DEG.027)
Qadeer Hussain (2303.KHI.DEG.006)

Daily Assignment :

Perform k-means clusterization on the Iris dataset. Repeat the procedure on the dataset reduced with PCA, and then compare the results.

Answer:

First of all we import the matplotlib, numpy sklearn libraries & packages.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

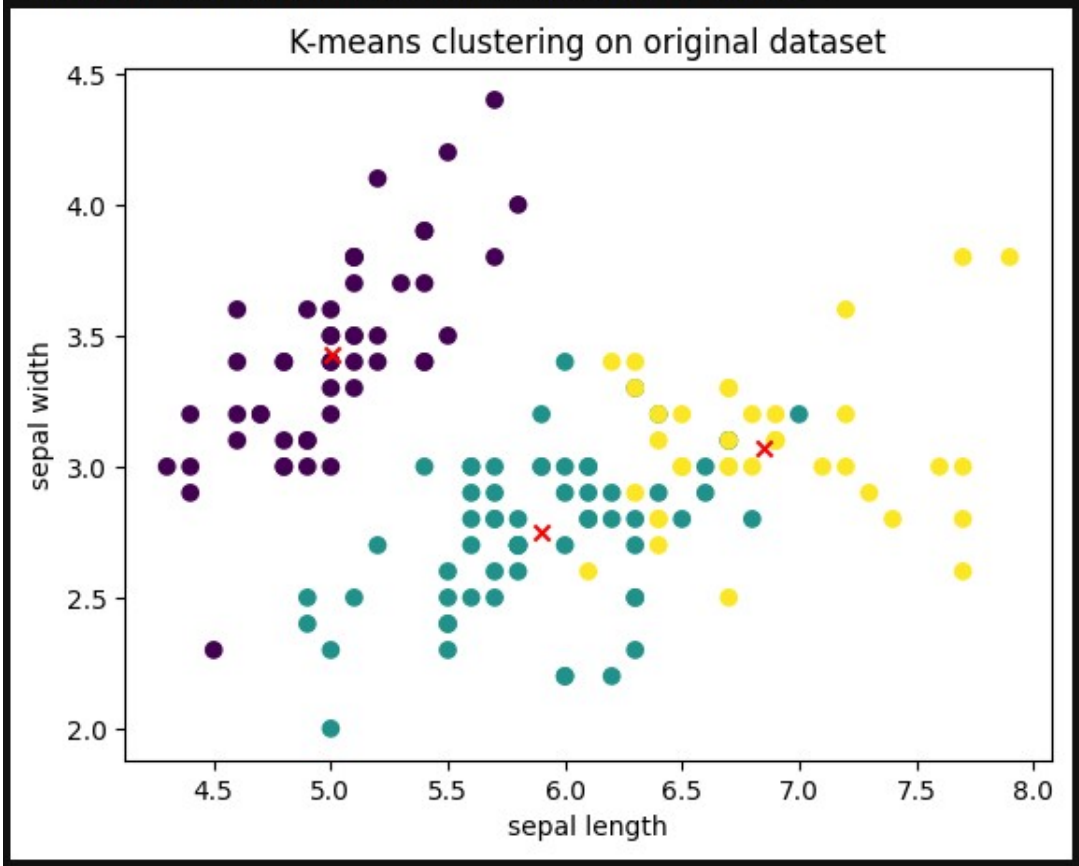
Then we performs k-means clustering on the Iris dataset. The dataset is loaded using the datasets.load_iris() function from scikit-learn and the data and target variables are assigned to x and y variables, respectively.
Then, an instance of the K-Means class from scikit-learn is created with n_clusters=3 specifying the number of clusters to form and n_init=1 and max_iter=100 specifying the number of times the algorithm will be run with different centroid seeds and the maximum number of iterations for each run, respectively. The fit() method of the K-Means instance is called on x to train the model and the predict() method is called to obtain the cluster labels for each data point in x. The cluster centers are obtained using the cluster_centers_ attribute of the K-Means instance. Finally, a scatter plot is created using plt.scatter() with x[:,0] and x[:,1] as the x and y coordinates, respectively, and c=all_predictions to assign a different color to each cluster. The centroids are also plotted on the same graph using plt.scatter() with centroids[:,0] and centroids[:,1] as the x and y coordinates, respectively, and marker='x' and color="red" to make them visible and red in color. The x and y axes are

labeled and the title of the graph is set using plt.xlabel(), plt.ylabel(), and plt.title()

```
iris = datasets.load_iris()
x = iris.data
y = iris.target

model = KMeans(n_clusters=3, n_init=1, max_iter=100)
model.fit(x)

all_predictions = model.predict(x)
centroids = model.cluster_centers_
print(centroids)

plt.scatter(x[:,0], x[:,1], c=all_predictions)
plt.scatter(centroids[:,0], centroids[:,1], marker='x', color="red")
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.title('K-means clustering on original dataset')
```

```
[[5.006      3.428      1.462      0.246     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]
 [6.85       3.07368421 5.74210526 2.07105263]]
```

```
Text(0.5, 1.0, 'K-means clustering on original dataset')
```



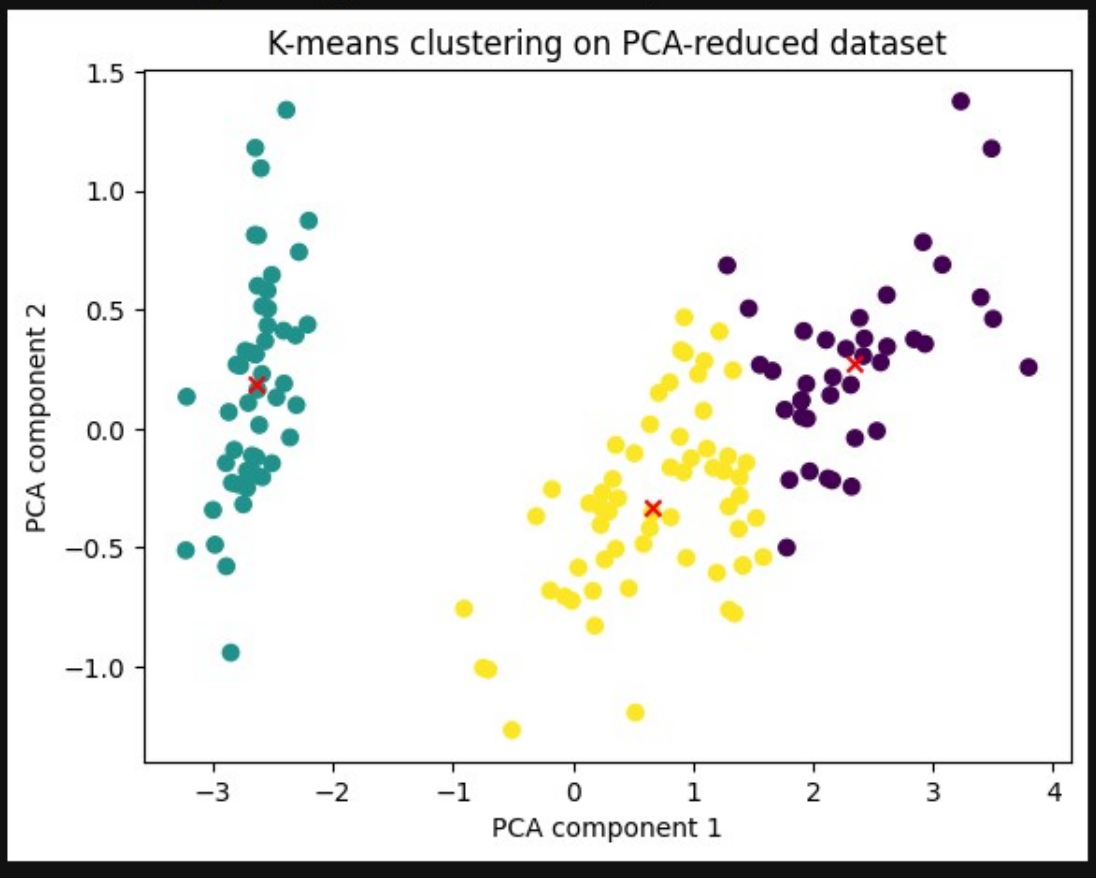
Then we performs k-means clustering on the Iris dataset after reducing its dimensionality using Principal Component Analysis (PCA).

pca = PCA(n_components=2) creates a PCA object that will reduce the dimensionality of the dataset to 2 dimensions. x_reduced = pca.fit_transform(x) applies PCA to the input feature matrix x, and reduces its dimensionality to 2. The result is stored in x_reduced.

```
iris = datasets.load_iris()
print(x.shape)
pca = PCA(n_components=2)
x_reduced = pca.fit_transform(x)

model = KMeans(n_clusters=3, n_init=1, max_iter=100)
model.fit(x_reduced)

all_predictions = model.predict(x_reduced)
centroids = model.cluster_centers_
print(centroids)

plt.scatter(x_reduced[:,0], x_reduced[:,1], c=all_predictions)
plt.scatter(centroids[:,0], centroids[:,1], marker='x', color="red")
plt.xlabel('PCA component 1')
plt.ylabel('PCA component 2')
plt.title('K-means clustering on PCA-reduced dataset')
plt.show
```

```
(150, 4)
[[ 2.34652659  0.27393856]
 [-2.64241546  0.19088505]
 [ 0.66567601 -0.3316042 ]]
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Comparison:

We creates a subplot with two columns using plt.subplot(1, 2, 1) and plt.subplot(1, 2, 2). The first column displays the scatter plot of the original dataset with sepal length on the x-axis and sepal width on the y-axis. The second column displays the scatter plot of the PCA-reduced dataset with PCA component 1 on the x-axis and PCA component 2 on the y-axis. Both scatter plots show the cluster labels obtained by the K-Means model and the centroids of the clusters.

Finally, the plt.subplots_adjust(wspace=0.5) we adds some horizontal padding between the two plots, and plt.tight_layout() adjusts the subplot parameters to fit the plot area, and plt.show() displays the plot.

```
iris = datasets.load_iris()
x = iris.data
y = iris.target

model = KMeans(n_clusters=3, n_init=1, max_iter=100)
model.fit(x)

all_predictions = model.predict(x)
centroids = model.cluster_centers_
# print(centroids)

plt.subplot(1, 2, 1)
plt.scatter(x[:,0], x[:,1], c=all_predictions)
plt.scatter(centroids[:,0], centroids[:,1], marker='x', color="red")
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.title('Original dataset')
plt.subplots_adjust(wspace=0.5)

iris = datasets.load_iris()
# print(x.shape)
pca = PCA(n_components=2)
x_reduced = pca.fit_transform(x)

model = KMeans(n_clusters=3, n_init=1, max_iter=100)
model.fit(x_reduced)

all_predictions = model.predict(x_reduced)
centroids = model.cluster_centers_
# print(centroids)
plt.subplot(1, 2, 2)

plt.scatter(x_reduced[:,0], x_reduced[:,1], c=all_predictions)
plt.scatter(centroids[:,0], centroids[:,1], marker='x', color="red")
plt.xlabel('PCA component 1')
plt.ylabel('PCA component 2')
plt.title('PCA-reduced dataset')
plt.tight_layout()
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```