Experiment 4

Objective:

To implement k means clustering.

Theory

K-means clustering is a popular unsupervised machine learning algorithm used to identify and group similar data points into clusters. The algorithm is commonly applied in exploratory data analysis, customer segmentation, pattern recognition, and image compression. Unlike supervised learning algorithms that rely on labeled data, K-means clustering operates without labels, identifying inherent patterns in the data based solely on the similarities among data points.

The K-means Algorithm aims to partition data points into k distinct clusters, where k is a user-defined parameter. Each cluster is represented by its centroid (the average of all points in the cluster). The algorithm works as follows:

Initialization: Choose k initial centroids, either by selecting random data points or by using methods like the K-means++ algorithm, which ensures initial centroids are spread out.
Assignment Step: Each data point is assigned to the nearest centroid based on a distance metric (typically Euclidean distance). This step forms k clusters, where each cluster contains the points closest to a specific centroid.

Update Step: After assigning data points to clusters, the centroids are recalculated by taking the mean of all points in each cluster. This updated centroid represents the new center of the cluster.
Iterate: The assignment and update steps are repeated until the centroids stabilize (i.e., they no longer change significantly) or until a maximum number of iterations is reached.
The objective of K-means is to minimize the within-cluster sum of squares (WCSS), which measures the variance within each cluster. By minimizing this, the algorithm ensures that data points within each cluster are as close as possible to their respective centroid, creating compact clusters.

Choosing the Number of Clusters (k) is a critical step in K-means. One commonly used technique is the elbow method, which involves running the algorithm for a range of k values and plotting the WCSS for each. The "elbow point," or the point where the WCSS reduction slows significantly, often suggests an optimal k. Another method is the silhouette score, which measures how similar points are within clusters compared to points in other clusters.

Advantages and Limitations: K-means is computationally efficient, especially for large datasets, and is relatively easy to implement. It performs well with spherical or well-separated clusters. However, K-means has limitations: it requires the number of clusters to be specified in advance, which is not always straightforward in unsupervised tasks. Additionally, it is sensitive to outliers and may perform poorly with non-spherical or overlapping clusters.
Image Compression: By clustering pixel colors, K-means can reduce the number of colors in an
In practical implementation, libraries like scikit-learn in Python provide an efficient K-means function that automates many aspects of the process, including initialization and convergence monitoring. By adjusting the parameter k and applying techniques like the elbow method, K-means can help reveal valuable insights from complex datasets.

Code & OUTPUT

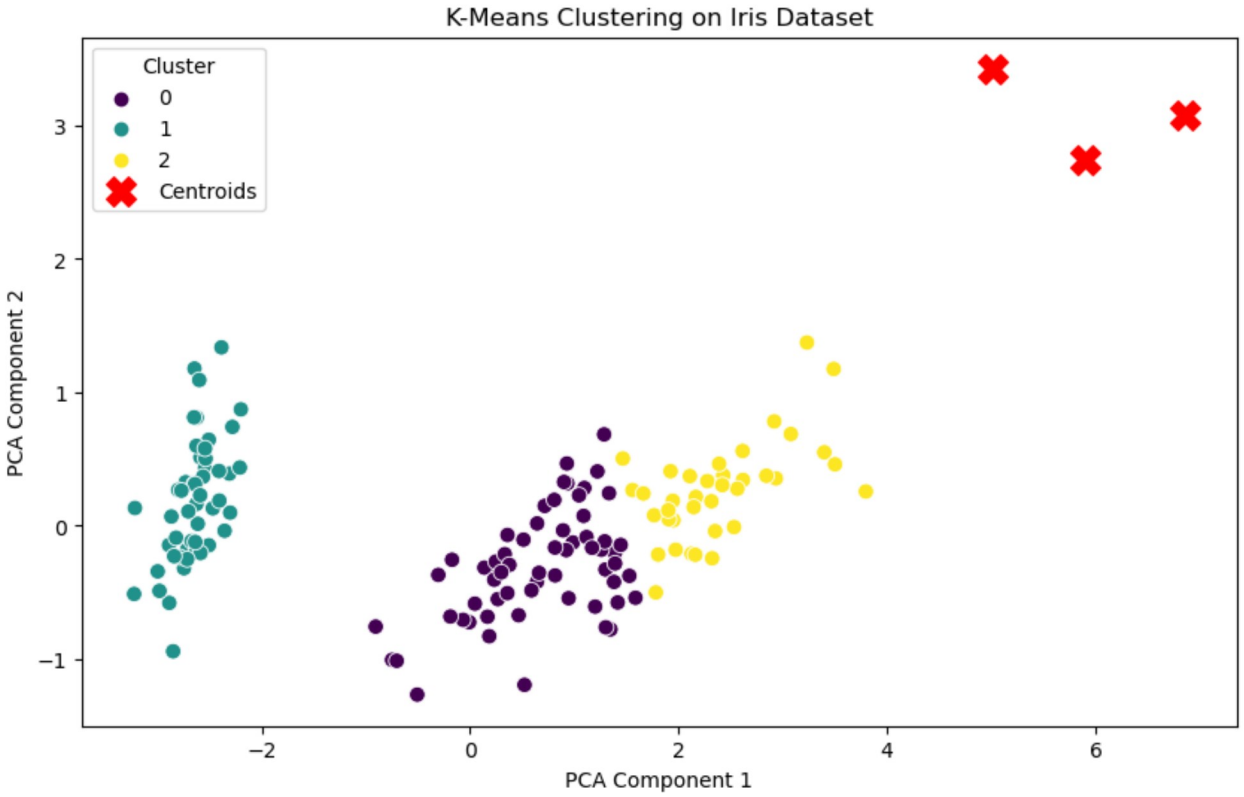
In [1]:
```
print("Experiment No 04 : To implement k means clustering.")
```

```
Experiment No 04 : To implement k means clustering.
```

In [5]:
```
# Import necessary libraries
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA

print("OUTPUT:\n\n")

# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # True labels for comparison

# Initialize the KMeans model
# Assuming 3 clusters as there are 3 species in the Iris dataset
kmeans = KMeans(n_clusters=3, random_state=42)

# Fit the model and predict clusters
y_kmeans = kmeans.fit_predict(X)

# Show cluster centers
print("Cluster Centers:")
print(kmeans.cluster_centers_)

# Visualize the clusters in a 2D plot using PCA (Principal Component Analysis) for dimensionality reduction
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Plot the results
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=y_kmeans, palette="viridis", s=60)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red', marker='X', s=200, label='Centroid
plt.title("K-Means Clustering on Iris Dataset")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.legend(title="Cluster")
plt.show()

# Evaluate by comparing with true labels
# Note: K-Means does not use true labels, this is just for evaluation purposes.
from sklearn.metrics import accuracy_score, confusion_matrix

# Map clusters to actual labels (for evaluation purposes only)
# In this case, we need to manually assign the clusters to labels for evaluation
# This can vary each time you run it
label_map = {0: 'setosa', 1: 'versicolor', 2: 'virginica'}
predicted_labels = [label_map[label] for label in y_kmeans]

# Display confusion matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y, y_kmeans))
```

OUTPUT:

```
c:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init
` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\hp\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memo
ry leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
Cluster Centers:
[[5.9016129  2.7483871  4.39354839 1.43387097]
 [5.006      3.428      1.462      0.246     ]
 [6.85       3.07368421 5.74210526 2.07105263]]
```




```
Confusion Matrix:
[[ 0 50  0]
 [48  0  2]
 [14  0 36]]
```

In [ ]:

Result

As a result of this Experiment, we successfully wrote and executed the program to implement k means clustering.

Learning Outcomes

Understand and apply the K-means clustering algorithm to group data points into clusters, selecting optimal k and interpreting results effectively.