Experiment 7

Aim: To implement different clustering algorithms.

Theory:
Clustering is an unsupervised machine learning technique used to group similar data points together. The objective is to discover natural groupings within a dataset without prior knowledge of class labels. Clustering is widely used in customer segmentation, anomaly detection, image segmentation, and bioinformatics.

Types of Clustering
1. Partition-based Clustering (e.g., K-Means)
   ● Divides data into a predefined number of clusters.
   ● Each data point belongs to exactly one cluster.
2. Density-based Clustering (e.g., DBSCAN)
   ● Forms clusters based on dense regions in the data.
   ● Can identify clusters of arbitrary shape and detect noise.
3. Hierarchical Clustering
   ● Builds a tree of clusters using a bottom-up (agglomerative) or top-down (divisive) approach.
4. Model-based Clustering
   ● Assumes data is generated by a mixture of underlying probability distributions (e.g., Gaussian Mixture Models).

K Means Clustering:
K-Means is a partition-based clustering algorithm that divides data into K clusters.

Algorithm Steps:
1. Choose the number of clusters, K.
2. Randomly initialize K centroids.
3. Assign each data point to the nearest centroid.
4. Update centroids by computing the mean of points in each cluster.
5. Repeat steps 3 & 4 until convergence (when centroids stop changing significantly).

Key Considerations:
Choosing K: The Elbow method and Silhouette Score help determine the optimal number of clusters.
Limitations: Sensitive to initial centroid placement and does not work well with non-spherical clusters or varying densities.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise):
DBSCAN is a density-based clustering algorithm that groups points closely packed together while marking low-density points as noise.

Algorithm Steps:
1. Set parameters:
2. ε (epsilon): Maximum distance to be considered a neighbor.
3. MinPts: Minimum points required to form a dense region.
4. Choose an unvisited point and determine its ε-neighborhood.
5. If the point has at least MinPts neighbors, a new cluster is formed.
6. Expand the cluster by adding density-reachable points.
7. Repeat for all points, marking outliers as noise.

Advantages:
● Handles clusters of arbitrary shape.
● Automatically detects noise and outliers.
● No need to specify the number of clusters.

Disadvantages:
● Choosing optimal ε and MinPts is challenging.
● Struggles with varying density clusters.

Steps:

1. **Load and preprocess the dataset.**

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment	On-board service	Leg room service	Baggage handling
0	1	0	13	1	2	460	3	4	3	1	...	5	4	3	4
1	1	1	25	0	0	235	3	2	3	3	...	1	1	5	3
2	0	0	26	0	0	1142	2	2	2	2	...	5	4	3	4
3	0	0	25	0	0	562	2	5	5	5	...	2	2	5	3
4	1	0	61	0	0	214	3	3	3	3	...	3	3	4	4

Checkin service	Inflight service	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
4	5	5	25	18.0	0
1	4	1	1	6.0	0
4	4	5	0	0.0	1
1	4	2	11	9.0	0
3	3	3	0	0.0	1

The dataset represents airline passenger satisfaction, containing demographic details (Gender, Age, Customer Type, Type of Travel), flight-related attributes (Class, Flight Distance, Gate Location), in-flight service ratings (Seat Comfort, Inflight WiFi, Entertainment, Food & Drink), and delay metrics (Departure/Arrival Delay). Satisfaction is labeled as 0 (Unsatisfied) or 1 (Satisfied), making it suitable for classification and clustering.

2. Elbow method for number of clusters

```python
# Determine optimal clusters using Elbow Method
inertia = []
k_values = range(1, 11)
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_scaled)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Method
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("Inertia")
plt.title("Elbow Method for Optimal K")
plt.show()
```
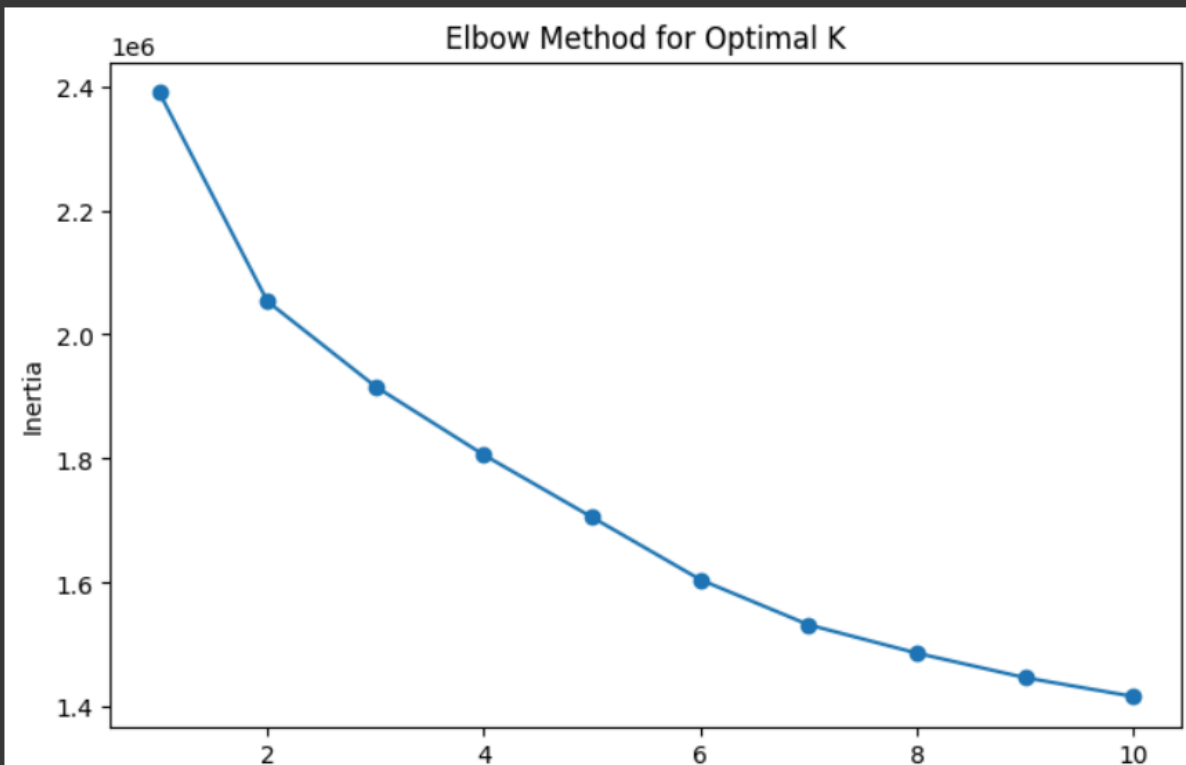
The Elbow Method plot helps determine the optimal number of clusters (K) in K-Means by plotting inertia (sum of squared distances to cluster centers) against different K values. The "elbow" point, where inertia reduction slows significantly, indicates the ideal K. In this plot, the elbow appears around K = 4 or 5.

3. K means clustering

```python
# Apply K-Means with optimal K (assume 3 based on elbow method)
kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(df_scaled)

# Display cluster distribution
print(df['Cluster'].value_counts())
```

```
Cluster
1    35729
3    25697
0    22216
2    20262
Name: count, dtype: int64
```

We try out clusters for k = 4 and 5. With k = 5, we find that cluster size of last cluster is very small as compared to the other clusters formed. So we use k = 4, we can observe that the clusters formed are similar in size. The four cluster formed are:

0 - 22216
1 - 35729
2 - 20262
3 - 25697

Visualization of clusters:

```python
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA

# Reduce dimensions to 3
pca_3d = PCA(n_components=3)
df_pca_3d = pca_3d.fit_transform(df_scaled)

df_plot_3d = pd.DataFrame(df_pca_3d, columns=['PC1', 'PC2', 'PC3'])
df_plot_3d['Cluster'] = df['Cluster']

# 3D scatter plot
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot with cluster colors
scatter = ax.scatter(df_plot_3d['PC1'], df_plot_3d['PC2'], df_plot_3d['PC3'],
                     c=df_plot_3d['Cluster'], cmap='viridis', alpha=0.6)

ax.set_title('K-Means Clustering Visualization (3D PCA)')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')

plt.legend(*scatter.legend_elements(), title='Cluster')
plt.show()

# Ensure feature names match df_scaled
feature_names = df.select_dtypes(include=[np.number]).columns.tolist()[:df_scaled.shape[1]]

# Create PCA components DataFrame
pca_components = pd.DataFrame(pca_3d.components_, columns=feature_names,
                             index=['PC1', 'PC2', 'PC3'])

# Display how features contribute to each principal component
print(pca_components.T)
```
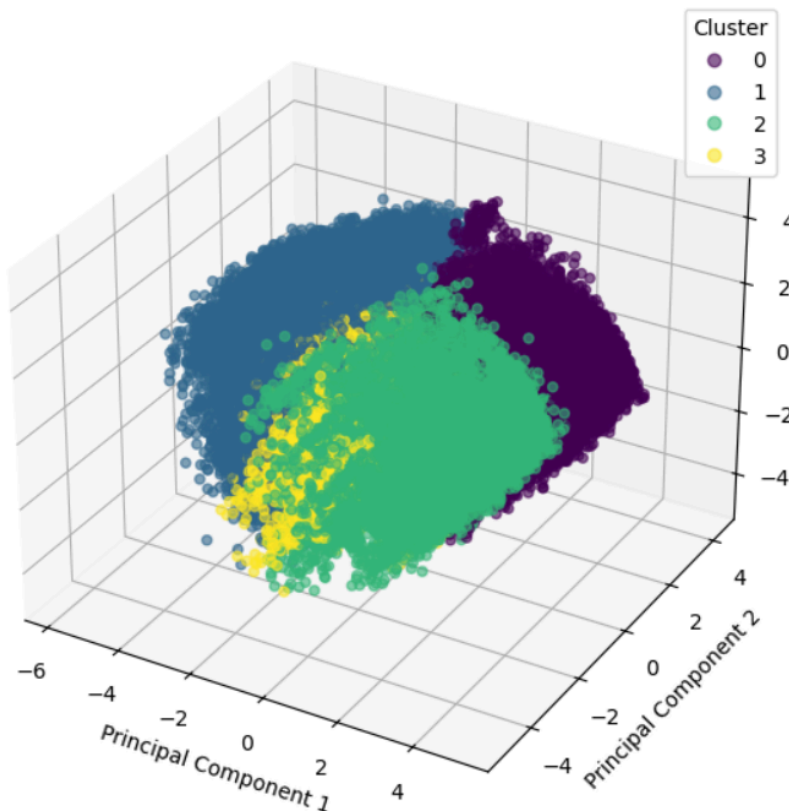

K-Means Clustering Visualization (3D PCA)

To visualize the clusters we perform Principal Component Analysis (PCA) to reduce the dataset dimensions to three, making it easier to visualize the clusters obtained from K-Means clustering. The 3D scatter plot visualizes the clusters in the transformed

feature space, where different colors represent different clusters. The clusters appear to be well-defined, indicating that the K-Means algorithm successfully grouped similar data points. The PCA components help analyze the contribution of original features to the new principal components, providing insights into how data is distributed across clusters.

4. DBSCAN clustering

```python
from sklearn.cluster import DBSCAN

# Apply DBSCAN clustering
dbscan = DBSCAN(eps=2, min_samples=50)  # Adjust eps and min_samples as needed
clusters = dbscan.fit_predict(df_scaled)

# Add cluster labels to the dataset
df["Cluster"] = clusters

# Display cluster distribution
print(df['Cluster'].value_counts())
```

```
Cluster
-1     70501
 1     11596
 0     11462
 2      4136
 3      3677
 4       695
13       259
11       219
 8       202
 5       178
 7       162
14       129
15       113
 6       107
12       100
10        75
 9        52
19        51
20        50
16        49
17        47
18        44
Name: count, dtype: int64
```

We apply DBSCAN clustering with eps=2 and min_samples=50, grouping data points based on density. It assigns cluster labels using dbscan.fit_predict(df_scaled) and stores them in the dataset. The output shows that most points (70501) are classified as noise (-1), while the remaining data forms multiple clusters of varying sizes.

The high number of noise points (-1) in the output suggests that the chosen eps=2 is too small to form dense clusters, causing many points to be classified as outliers. Additionally, min_samples=50 requires at least 50 points in a neighborhood to form a cluster, which may be too restrictive for some regions of the dataset. As a result, only a few dense clusters are detected, while most points remain unclustered.

Visualization:
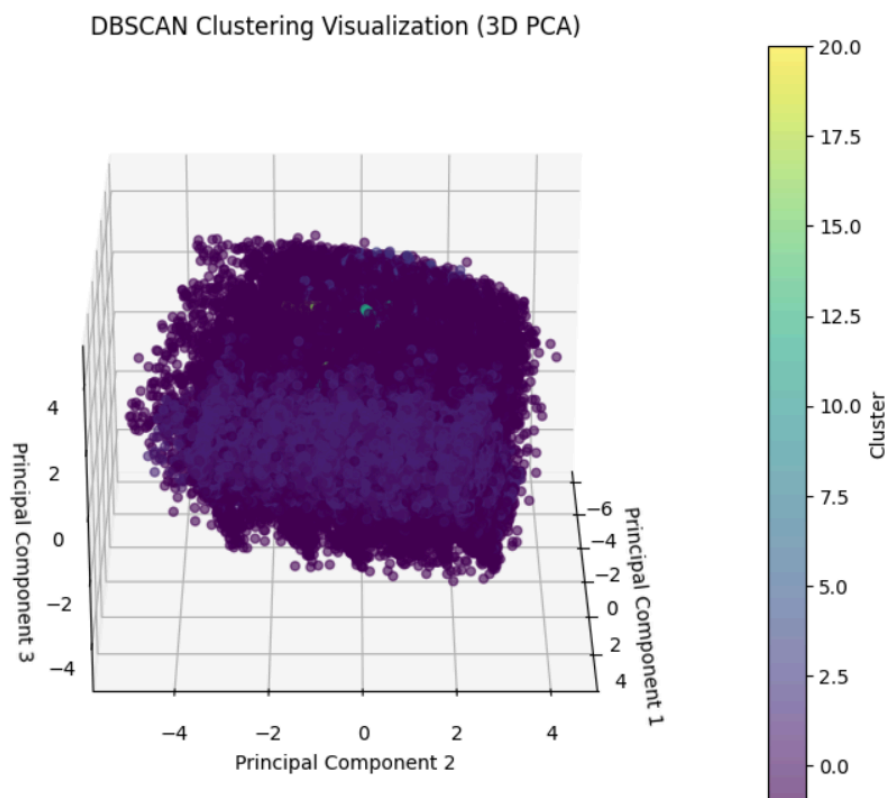
The DBSCAN clustering visualization in 3D PCA space shows that the majority of points are assigned to cluster -1 (noise) with only a few points belonging to distinct smaller clusters. This suggests that DBSCAN's parameters (epsilon and min_samples) may not be well-tuned for clear separation, leading to most points being grouped together while only a few scattered regions are identified as separate clusters.

Overall we can see that the performance of K means is better than DBSCAN for clustering the dataset.
K-Means was a better choice for this dataset because the data is well-distributed without high-density regions, and clusters can be separated by distance rather than density.
DBSCAN struggled due to high dimensionality and the lack of naturally dense clusters, causing excessive noise detection.

Conclusion:

In this experiment, we implemented and analyzed different clustering techniques, including K-Means and DBSCAN, to identify meaningful patterns in an airline passenger satisfaction dataset.

K-Means clustering successfully grouped the dataset into four well-defined clusters (K=4), ensuring a balanced distribution of data points across clusters. The Elbow Method helped determine the optimal number of clusters, and PCA visualization confirmed that the clusters were well-separated.

DBSCAN clustering, on the other hand, struggled with this dataset. The chosen parameters (eps=2, min_samples=50) resulted in a large number of data points being classified as noise (-1), indicating that DBSCAN was unable to effectively form clusters. This suggests that the dataset does not contain well-defined dense regions, making DBSCAN less suitable for this case.