Experiment 8

Aim:

To design and implement a music recommendation system using unsupervised machine learning techniques, namely **K-Means Clustering** and **Principal Component Analysis (PCA)** on the `spotify.csv` dataset.

Theory:

The goal of this experiment is to group songs with similar characteristics and recommend songs from the same group. This is achieved using two key techniques:

1. **Principal Component Analysis (PCA)** – to reduce dimensionality and enable effective visualization.

2. **K-Means Clustering** – to form groups (clusters) of similar songs based on their audio features.

Dataset Description:

- **Name:** spotify.csv
- **Records:** Approximately 1100+ songs
- **Attributes:** Includes numerical attributes such as danceability, energy, loudness, acousticness, instrumentalness, tempo, etc.
- **Purpose:** These features are used to identify similarities between songs and cluster them accordingly.

Steps Involved:

1. Data Preprocessing:

- Selected relevant numeric features related to song characteristics.
- Applied **StandardScaler** to standardize the features, which is essential for distance-based models like K-Means and PCA.

```
 Dataset Loaded Successfully!
Columns in the dataset:
Index(['valence', 'year', 'acousticness', 'artists', 'danceability',
       'duration_ms', 'energy', 'explicit', 'id', 'instrumentalness', 'key',
       'liveness', 'loudness', 'mode', 'name', 'popularity', 'release_date',
       'speechiness', 'tempo'],
      dtype='object')
```

```
df = df.drop(['id', 'name', 'artists', 'release_date'], axis=1)
```

```
df.head()
```

	valence	year	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	loudness	mode	popularity	speechiness	tempo	Cluster
0	0.0594	1921	0.982	0.279	831667	0.211	0	0.878000	10	0.665	-20.096	1	4	0.0366	80.954	0
1	0.9630	1921	0.732	0.819	180533	0.341	0	0.000000	7	0.160	-12.441	1	5	0.4150	60.936	2
2	0.0394	1921	0.961	0.328	500062	0.166	0	0.913000	3	0.101	-14.850	1	5	0.0339	110.339	0
3	0.1650	1921	0.967	0.275	210000	0.309	0	0.000028	5	0.381	-9.316	1	3	0.0354	100.109	0
4	0.2530	1921	0.957	0.418	166693	0.193	0	0.000002	3	0.229	-10.096	1	2	0.0380	101.665	0

2. Dimensionality Reduction using PCA:

Objective:

To reduce the number of input features while retaining as much information (variance) as possible.

Process:

- PCA was applied with `n_components=2`.
- The explained variance ratio was checked to ensure that a significant portion of data variability is preserved.
- The 2D data was used for visualization of clusters.

Benefits:

- Helps visualize high-dimensional data.
- Reduces noise and computational complexity.

- Enhances the performance of clustering models.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Standardize the features
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Apply PCA (we'll keep 2 components for visualization)
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)
```

Scatter plot of PCA-reduced data before applying clustering.
This visualization represents the spread of songs across the first two principal components. It helps identify any natural grouping or separation in the data.

3. Clustering using K-Means:

Objective:
To group similar songs into `k=6` clusters using K-Means clustering.

Process:

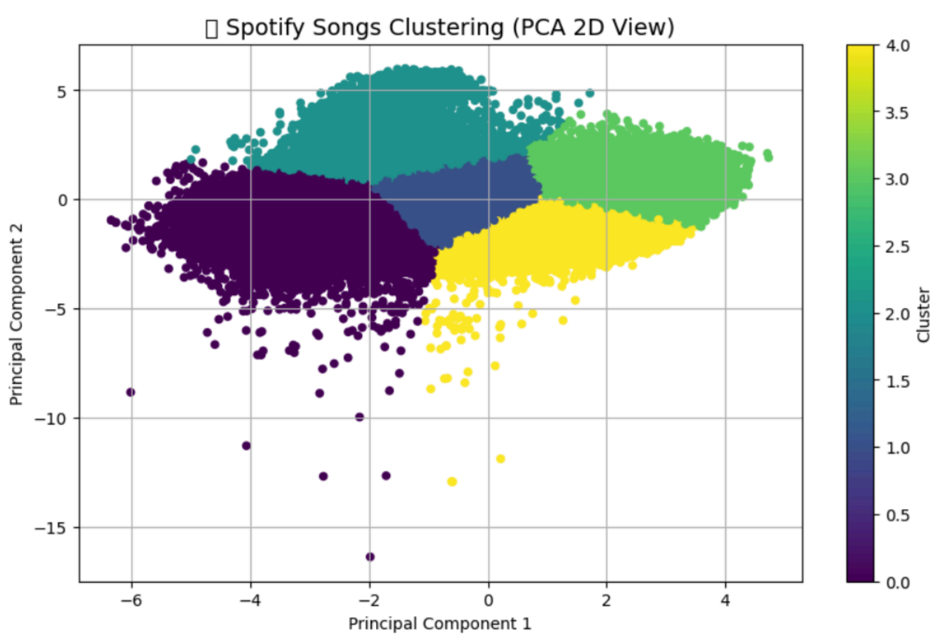
- KMeans from `sklearn.cluster` was used with `n_clusters=6`.
- The model was trained on standardized features.
- Each song was labeled with a cluster number from 0 to 5.

- PCA components were used to plot the clustered data.

```
from sklearn.cluster import KMeans

# Let’s say we choose 5 clusters
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(pca_data)

# Add cluster info back to original dataframe
df['Cluster'] = clusters
```

```
import matplotlib.pyplot as plt

# Basic scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(pca_data[:, 0], pca_data[:, 1], c=clusters, cmap='viridis', s=20)
plt.title(" Spotify Songs Clustering (PCA 2D View)", fontsize=14)
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.colorbar(label='Cluster')
plt.grid(True)
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 127912 (\N{ARTIST PALETTE}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



Songs plotted with cluster labels using PCA components.
Each color represents a different cluster of songs that share similar characteristics. The X and Y axes correspond to the first and second principal components.

Recommendation Logic:

Once the songs are clustered, we can recommend songs from the same cluster as a chosen song:

```
song_name = "Blinding Lights"
if song_name in original_df['name'].values:
    liked_song = original_df[original_df['name'] == song_name].iloc[0]
    liked_cluster = liked_song['Cluster']
    print(f"\n You liked: {liked_song['name']} by {liked_song['artists']} (Cluster {liked_cluster})\n"
    recommendations = original_df[
        (original_df['Cluster'] == liked_cluster) &
        (original_df['name'] != song_name)
    ][['name', 'artists']].head(5)
    print("Recommended Songs:")
    print(recommendations)
else:
    print(" Song not found in the dataset.")
```

```
 You liked: Blinding Lights by ['The Weeknd'] (Cluster 4)

Recommended Songs:
                                        name                artists
5667                               Gandagana      ['Georgian People']
7186  Woke Up This Morning (My Baby She Was Gone)          ['B.B. King']
7232                  Blue Train - Remastered 2003     ['John Coltrane']
7236                                  Milestones        ['Miles Davis']
7252                 One For Daddy-O - Remastered  ['Cannonball Adderley']
```

Conclusion:

In this experiment, a recommendation system was implemented using unsupervised learning. Dimensionality reduction via PCA allowed effective visualization and simplification of the data. K-Means clustering grouped songs with similar features, allowing content-based recommendations.

This approach is scalable, efficient, and interpretable, making it suitable for music-based recommendation systems.