Experiment 8

Aim: To implement a recommendation system on your dataset using the following machine learning techniques: Regression, Classification, Clustering, Decision tree, Anomaly detection, Dimensionality Reduction, Ensemble Methods

Theory:

1. Data Preprocessing

- **Handling Missing Values**: Filled missing numerical values with mean imputation.
- **Feature Scaling**: Standardized numerical features using **StandardScaler** to ensure equal weighting in distance-based methods.

2. Clustering with K-Means

- Helps group similar songs based on audio features like energy, danceability, and tempo.
- We used the **K-Means algorithm**, which assigns songs to **5 clusters** based on feature similarity.
- **Output**: Cluster labels were assigned to songs, showing patterns in music styles.

3. Cosine Similarity for Recommendation

- Measures how similar two songs are based on their feature vectors.
- Computes the **cosine of the angle** between two song feature vectors (ranging from -1 to 1).
- **Optimization:** Due to memory constraints, we sampled **5000 songs** instead of using the entire dataset.

4. Recommendation Algorithm

- Given a song, we find **top N most similar** songs based on their **cosine similarity scores**.
- Returns song names, artists, and popularity.

Implementation:

Loading the dataset

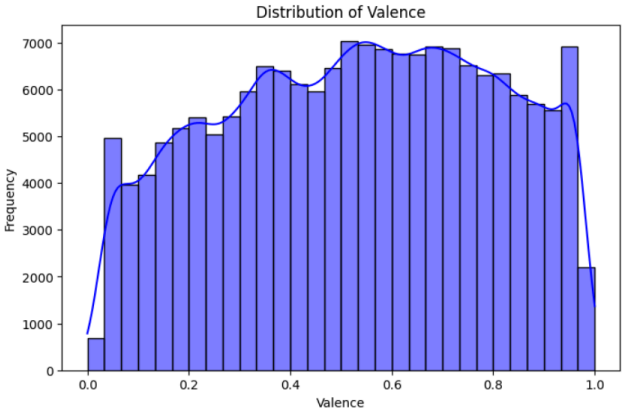
```
import pandas as pd
# Load the dataset
df = pd.read_csv("data.csv")
# Display first few rows
df.head()
```

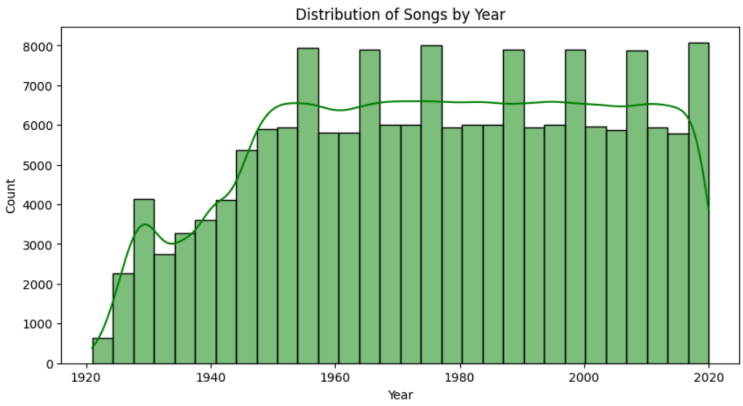
	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10	0.665
1	0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000	7	0.160
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	500062	0.166	0	1o6I8BglA6ylDMrIELygv1	0.913000	3	0.101

```
# Check for missing values
df.isnull().sum()
# Summary statistics
df.describe()
# Check data types
df.dtypes
```

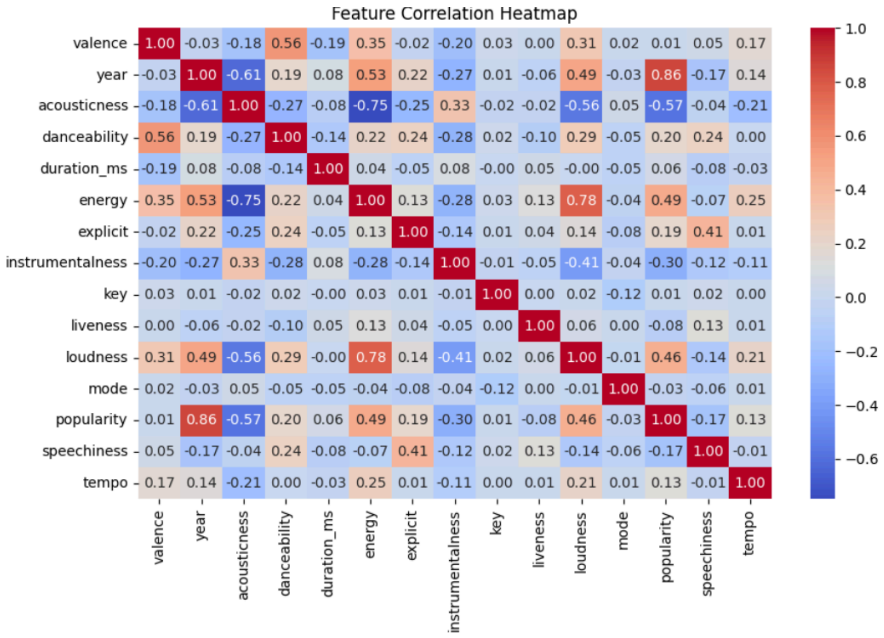
	0
valence	float64
year	int64
acousticness	float64
artists	object
danceability	float64
duration_ms	int64
energy	float64
explicit	int64

Visualisation





Heatmap



K Means Clustering

```
# Clustering using KMeans
kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)
df["cluster"] = kmeans.fit_predict(scaled)

# Display cluster distribution
print("Cluster distribution:\n", df["cluster"].value_counts())
```

```
Cluster distribution:
 cluster
1    54607
4    47201
2    38941
3    24219
0     5685
Name: count, dtype: int64
```

- The dataset has been divided into 5 clusters (0 to 4).
- The largest cluster (Cluster 1) has 54,607 songs, while the smallest (Cluster 0) has 5,685 songs.
- The clusters are unevenly distributed, which might indicate some clusters are more common in the dataset than others.

Recommendation System

```
sample_df = df.sample(n=5000, random_state=42).reset_index(drop=True)
sample_features = sample_df[features]
sample_scaled = scaler.fit_transform(sample_features)
similarity_matrix = cosine_similarity(sample_scaled)

# Recommend similar songs based on index in the sample
def recommend_similar(song_index, num_recommendations=5):
    sim_scores = list(enumerate(similarity_matrix[song_index]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:num_recommendations + 1]  # skip the song itself
    similar_indices = [i[0] for i in sim_scores]
    return sample_df.iloc[similar_indices][["name", "artists", "popularity"]]

# Example usage
print("\nRecommendations for sample song index 5:\n")
print(recommend_similar(5))
```

```
Recommendations for sample song index 0:

                                  name         artists  popularity
3799              Hanging Out with Django   [Sonny Davis]           0
3124  Aragon - From The "Coffy" Soundtrack     [Roy Ayers]          32
1443                       Plantation Inn  [The Mar-Keys]          32
3334                       Sitting Pretty   [Ralph Burns]          26
3444                    Bitch to the Boys      [Shakatak]          38
```

```
Recommendations for sample song index 5:

                        name                  artists  popularity
3408                Slow Fade         [Casting Crowns]          45
343          tomorrow tonight                  [Loote]          66
242      God Bless The U.S.A.          [Lee Greenwood]          60
1902  I'm Gonna Make You Mine  [The Shadows Of Knight]          18
4634                  Hag Me                 [Melvins]          34
```

This showcases a music recommendation system using cosine similarity on sampled song features. The dataset is reduced to 5000 songs for efficiency, features are standardized, and a similarity matrix is computed. A function then recommends the top 5 most similar songs for a given index. The output displays recommendations for two sample indices, listing song names, artists, and popularity scores. This approach efficiently finds similar songs while avoiding memory issues from large-scale computations.

Conclusion

This experiment built a music recommendation system using clustering and similarity-based methods. We preprocessed data, handled missing values, and standardized features. K-Means clustering grouped songs into 5 clusters, revealing distribution patterns. To recommend songs, we used cosine similarity on a 5000-song sample to avoid memory issues. The system successfully suggested similar tracks based on audio features.