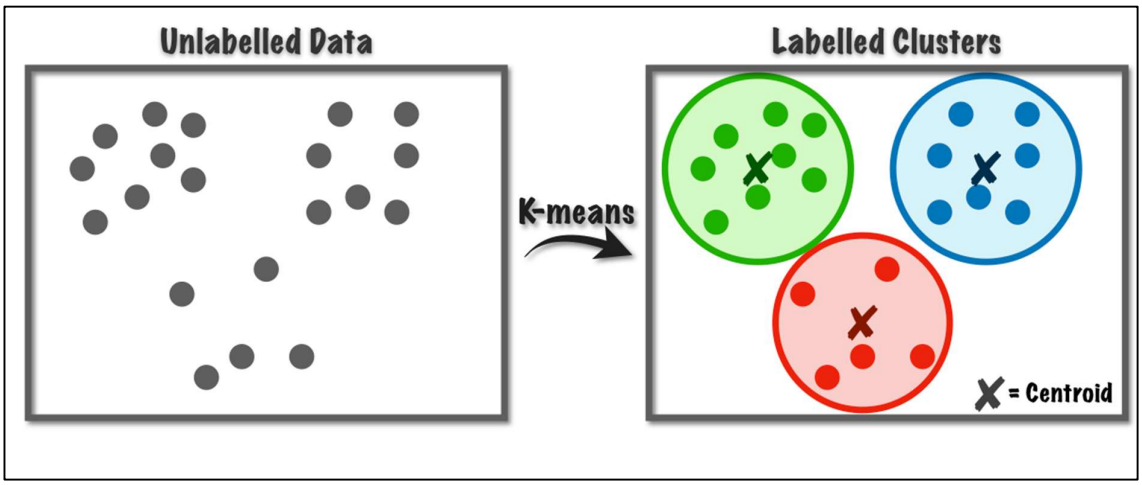
Name: Ritesh Maurya Exp No.8 ID:VU4F2223112

Aim: Develop a recommendation system by Applying any machine learning techniques and using available dataset.

Theory:

We have used the **K-Means Clustering** algorithm, which is an unsupervised machine learning technique, to create a movie recommendation system. K-Means works by grouping movies into clusters based on their features, such as genres and descriptions. We convert these features into numerical data using **TF-IDF Vectorization**. Once movies are clustered, when a user selects a movie, the system recommends other movies from the same cluster. This is a **content-based filtering** approach, which suggests similar items based on their inherent attributes. The steps involve data cleaning, vectorization, clustering, and recommending from the same cluster.\



Program:

Step 1: Installing Required Libraries

You will need the following Python libraries. Make sure to install them using pip.

```
[ ]: pip install pandas scikit-learn
```

Step 2: Loading the Dataset

Loading dataset "tmdb_5000_movies.csv" dataset using pandas.

Name: Ritesh Maurya Exp No.8 ID:VU4F2223112

```
[10]: import pandas as pd

# Load the dataset (replace the path with your actual file path)
movies_df = pd.read_csv("tmdb_5000_movies.csv")

# Check the first few rows of the dataset
print(movies_df.head())
```

```
      budget                                             genres  \
0  237000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...
1  300000000  [{"id": 12, "name": "Adventure"}, {"id": 14, "...
2  245000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...
3  250000000  [{"id": 28, "name": "Action"}, {"id": 80, "nam...
4  260000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...

                                       homepage      id  \
0                   http://www.avatarmovie.com/   19995
1  http://disney.go.com/disneypictures/pirates/     285
2   http://www.sonypictures.com/movies/spectre/  206647
3            http://www.thedarkknightrises.com/   49026
4          http://movies.disney.com/john-carter   49529

                                            keywords original_language  \
0  [{"id": 1463, "name": "culture clash"}, {"id":...                en
1  [{"id": 270, "name": "ocean"}, {"id": 726, "na...                en
2  [{"id": 470, "name": "spy"}, {"id": 818, "name...                en
3  [{"id": 849, "name": "dc comics"}, {"id": 853,...                en
4  [{"id": 818, "name": "based on novel"}, {"id":...                en

                             original_title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                    The Dark Knight Rises
4                                 John Carter

                                            overview  popularity  \
0  In the 22nd century, a paraplegic Marine is di...  150.437577
1  Captain Barbossa, long believed to be dead, ha...  139.082615
```

Step 3: Cleaning the Genres Column

The genres column is in JSON-like format, so cleaning and extracting the genre names.

```
[12]: import ast

# Function to clean the genres column
def clean_genres(genres):
    genres = ast.literal_eval(genres)
    return [genre['name'] for genre in genres]

# Apply the cleaning function to the 'genres' column
movies_df['clean_genres'] = movies_df['genres'].apply(clean_genres)

# Check if it's cleaned
print(movies_df[['title', 'clean_genres']].head())
```

```
                                      title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                    The Dark Knight Rises
4                                 John Carter

                                    clean_genres
0  [Action, Adventure, Fantasy, Science Fiction]
1                  [Adventure, Fantasy, Action]
2                    [Action, Adventure, Crime]
3             [Action, Crime, Drama, Thriller]
4          [Action, Adventure, Science Fiction]
```

Step 4: Combining Genres and Overview

Combining the cleaned genres with the movie overview to create a new feature for clustering.

Name: Ritesh Maurya Exp No.8 ID:VU4F2223112

```
[13]: # Combine genres and overview as text features for clustering
movies_df['text_features'] = movies_df['clean_genres'].astype(str) + ' ' + movies_df['overview'].astype(str)

# Check the combined text features
print(movies_df[['title', 'text_features']].head())
```

```
                                   title  \
0                                 Avatar
1  Pirates of the Caribbean: At World's End
2                                Spectre
3                  The Dark Knight Rises
4                            John Carter

                                      text_features
0  ['Action', 'Adventure', 'Fantasy', 'Science Fi...
1  ['Adventure', 'Fantasy', 'Action'] Captain Bar...
2  ['Action', 'Adventure', 'Crime'] A cryptic mes...
3  ['Action', 'Crime', 'Drama', 'Thriller'] Follo...
4  ['Action', 'Adventure', 'Science Fiction'] Joh...
```

Step 5: Vectorize Text Features

Using TF-IDF to convert text features into vectors.

```
[14]: from sklearn.feature_extraction.text import TfidfVectorizer

# Vectorize the text features
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
tfidf_matrix = tfidf_vectorizer.fit_transform(movies_df['text_features'])

print(tfidf_matrix.shape)  # Check the shape of the matrix
```

```
(4803, 5000)
```

Step 6: Appling K-Means Clustering

Now, clustering the movies into groups using K-Means.

```
[15]: from sklearn.cluster import KMeans

# Apply K-Means clustering
kmeans = KMeans(n_clusters=10, random_state=42)
movies_df['cluster'] = kmeans.fit_predict(tfidf_matrix)

# Check the cluster assigned to each movie
print(movies_df[['title', 'cluster']].head())
```

```
                                   title  cluster
0                                 Avatar        2
1  Pirates of the Caribbean: At World's End        6
2                                Spectre        9
3                  The Dark Knight Rises        9
4                            John Carter        2
```

Step 7: Creating a Recommendation Function

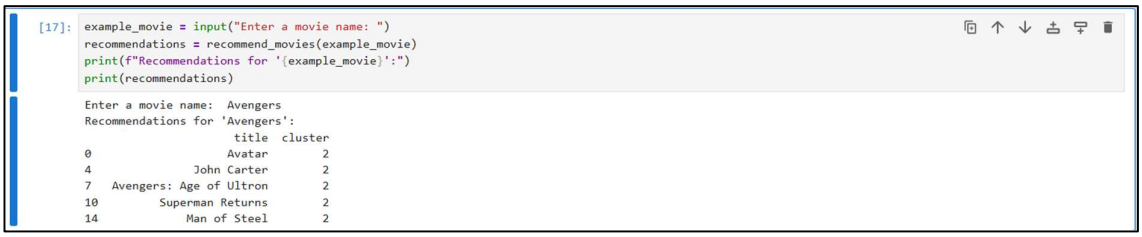
Creating a function that recommends movies based on the cluster of a given movie.

```
[16]: def recommend_movies(movie_title, num_recommendations=5):
    # Find the cluster of the input movie
    movie_cluster = movies_df[movies_df['title'].str.contains(movie_title, case=False, na=False)]['cluster'].values[0]

    # Get all movies from the same cluster
    similar_movies = movies_df[movies_df['cluster'] == movie_cluster][['title', 'cluster']]

    # Exclude the input movie from recommendations
    similar_movies = similar_movies[similar_movies['title'].str.lower() != movie_title.lower()]

    # Return the top 'num_recommendations' similar movies
    return similar_movies.head(num_recommendations)
```

Name: Ritesh Maurya Exp No.8 ID:VU4F2223112

Step 8: Testing the Recommendation System

Testing the recommendation system with a sample movie.



```
[17]: example_movie = input("Enter a movie name: ")
      recommendations = recommend_movies(example_movie)
      print(f"Recommendations for '{example_movie}':")
      print(recommendations)

      Enter a movie name:  Avengers
      Recommendations for 'Avengers':
                          title  cluster
      0                  Avatar        2
      4             John Carter        2
      7   Avengers: Age of Ultron        2
      10        Superman Returns        2
      14            Man of Steel        2
```

```
[30]: example_movie = input("Enter a movie name: ")
      recommendations = recommend_movies(example_movie)
      print(f"Recommendations for '{example_movie}':")
      print(recommendations)

      Enter a movie name:  Toy Story 3
      Recommendations for 'Toy Story 3':
                        title  cluster
      6               Tangled        3
      34  Monsters University        3
      54    The Good Dinosaur        3
      57              WALL·E        3
      66                  Up        3
```

Conclusion: Hence, we have successfully Developed a recommendation system by Applying any machine learning techniques and using available dataset.