Introduction to the IMDB Top 250 Movies Dataset

The IMDB Top 250 Movies dataset, available on Kaggle, offers a comprehensive list of the highest-rated movies according to user ratings on the Internet Movie Database (IMDB). This dataset is an invaluable resource for movie enthusiasts, data analysts, and machine learning practitioners alike. It provides a rich source of information that can be used for various analytical and predictive tasks, including sentiment analysis, recommendation systems, and trend analysis.

This dataset is having the data of the top 250 Movies as per their IMDB rating listed on the official website of IMDB

Features

- rank - Movie Rank as per IMDB rating
- movie_id - Movie ID
- title - Name of the Movie
- year - Year of Movie release
- link - URL for the Movie
- imdb_votes - Number of people who voted for the IMDB rating
- imdb_rating - Rating of the Movie
- certificate - Movie Certification
- duration - Duration of the Movie
- genre - Genre of the Movie
- cast_id - ID of the cast member who have worked on the Movie
- cast_name - Name of the cast member who have worked on the Movie
- director_id - ID of the director who have directed the Movie
- director_name - Name of the director who have directed the Movie
- writer_id - ID of the writer who have wrote script for the Movie
- writer_name - Name of the writer who have wrote script for the Movie
- storyline - Storyline of the Movie
- user_id - ID of the user who wrote review for the Movie
- user_name - Name of the user who wrote review for the Movie
- review_id - ID of the user review
- review_title - Short review
- review_content - Long review

Source

https://www.kaggle.com/datasets/karkavelrajaj/imdb-top-250-movies

Importing Necessary Libraries

```
import pandas as pd #Pandas is a powerful library for data manipulation and analysis.
```

```
df = pd.read_csv('movies.csv') #Loading the dataset.
```

We will now read the data from a CSV file into a Pandas DataFrame Let us have a look at how our dataset looks like using df.head()

```
df.head() #Displays the first 5 rows of the dataset.
```

	rank	movie_id	title	year	link	imbd_votes	imbd_rating	certificate	duration	genre	..
0	1	tt0111161	The Shawshank Redemption	1994	https://www.imdb.com/title/tt0111161	2,711,075	9.3	R	2h 22m	Drama	
1	2	tt0068646	The Godfather	1972	https://www.imdb.com/title/tt0068646	1,882,829	9.2	R	2h 55m	Crime,Drama	
2	3	tt0468569	The Dark Knight	2008	https://www.imdb.com/title/tt0468569	2,684,051	9.0	PG-13	2h 32m	Action,Crime,Drama	
3	4	tt0071562	The Godfather Part II	1974	https://www.imdb.com/title/tt0071562	1,285,350	9.0	R	3h 22m	Crime,Drama	
4	5	tt0050083	12 Angry Men	1957	https://www.imdb.com/title/tt0050083	800,954	9.0	Approved	1h 36m	Crime,Drama	

5 rows × 22 columns

Exploring the Data:

Understanding the dataset by exploring its structure and contents.

```
df.columns # Displays the names of the columns
```

```
Index(['rank', 'movie_id', 'title', 'year', 'link', 'imbd_votes',
       'imbd_rating', 'certificate', 'duration', 'genre', 'cast_id',
       'cast_name', 'director_id', 'director_name', 'writer_id', 'writer_name',
       'storyline', 'user_id', 'user_name', 'review_id', 'review_title',
       'review_content'],
      dtype='object')
```

```
df.shape # Displays the total count of the Rows and Columns respectively.
```

```
(250, 22)
```

```
df.info() #Displays the total count of values present in the particular column along with the null count and data type.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   rank           250 non-null    int64
 1   movie_id       250 non-null    object
 2   title          250 non-null    object
 3   year           250 non-null    int64
 4   link           250 non-null    object
 5   imbd_votes     250 non-null    object
 6   imbd_rating    250 non-null    float64
 7   certificate    249 non-null    object
 8   duration       250 non-null    object
 9   genre          250 non-null    object
 10  cast_id        250 non-null    object
 11  cast_name      250 non-null    object
 12  director_id    250 non-null    object
 13  director_name  250 non-null    object
 14  writer_id      250 non-null    object
 15  writer_name    250 non-null    object
```

```
 16  storyline       250 non-null    object
 17  user_id         250 non-null    object
 18  user_name       250 non-null    object
 19  review_id       250 non-null    object
 20  review_title    250 non-null    object
 21  review_content  250 non-null    object
dtypes: float64(1), int64(2), object(19)
memory usage: 43.1+ KB
```

Data Cleaning:

Checking for missing values, duplicates, or any inconsistencies and clean the data accordingly.

```
df.isnull().sum()
```

```
rank              0
movie_id          0
title             0
year              0
link              0
imbd_votes        0
imbd_rating       0
certificate       1
duration          0
genre             0
cast_id           0
cast_name         0
director_id       0
director_name     0
writer_id         0
writer_name       0
storyline         0
user_id           0
user_name         0
review_id         0
review_title      0
review_content    0
dtype: int64
```

As we can check there is only 1 null value in the certificate column. As the count of the null value is much less, we can drop the null value as it will not affect the the out come as what we want to predict.

```
df.drop_duplicates(inplace=True) #Dropping the duplicate values in the dataset.
```

```
df = df.dropna() #Dropping the null values in the dataset.
```

```
df.isnull().sum() #Displays the total count of the null values in the particular columns.
```

```
rank              0
movie_id          0
title             0
year              0
link              0
imbd_votes        0
imbd_rating       0
certificate       0
duration          0
genre             0
cast_id           0
cast_name         0
director_id       0
director_name     0
writer_id         0
writer_name       0
storyline         0
user_id           0
user_name         0
review_id         0
review_title      0
review_content    0
dtype: int64
```

Now there is no null value in the dataset.

Feature Selection

Identify the features that will be used for the recommendation system. Common features include:

- Title
- Genre
- Director
- Actors
- Rating
- Year

Here we are creating a data frame df['combined_features'] that will contain the the columns like genre, director name, cast name.

```
df['combined_features'] = df['genre'] + ' ' + df['director_name'] + ' ' + df['cast_name']
```

```
<ipython-input-11-0bd79f9b96ac>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  df['combined_features'] = df['genre'] + ' ' + df['director_name'] + ' ' + df['cast_name']
```

TF-IDF (Term Frequency-Inverse Document Frequency):

Term Frequency (TF): Measures the frequency of a word in a document.

Inverse Document Frequency (IDF): Measures how important a word is. It decreases the weight of commonly occurring words and increases the weight of words that are rare across documents.

The TF-IDF score for a word in a document is the product of its TF and IDF scores. This helps in giving more importance to unique words in a document and less to common words like "the", "and", etc.

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop_words='english')
#Creates an instance of TfidfVectorizer with the stop_words parameter set to 'english'.
#stop_words='english' means that common English words (like "the", "is", "in") will be ignored when computing the TF-IDF scores. These are k

tfidf_matrix = tfidf.fit_transform(df['combined_features']) #df['combined_features'] is a pandas Series containing the text data of combinec
#fit_transform method does two things:
#Fit: Learns the vocabulary and IDF from the combined features.
#Transform: Transforms the combined features into a TF-IDF matrix.
```

Benefits

Dimensionality Reduction: By ignoring common words, it reduces the number of features.

Importance Weighting: TF-IDF gives higher importance to rare and meaningful words, making it easier to compare documents (movies) based on significant terms.

Understanding cosine_similarity

Cosine Similarity:

- Cosine similarity is a measure of similarity between two non-zero vectors.
- It calculates the cosine of the angle between two vectors in a multi-dimensional space.

- The cosine similarity is bounded between -1 and 1, where:
- 1 means the vectors are identical.
- 0 means the vectors are orthogonal (no similarity).
- -1 means the vectors are diametrically opposite.

```
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix) #This function will provide the top 10 movies that are most similar to "The Godfa
```

tfidf_matrix:

tfidf_matrix is a sparse matrix where each row represents a movie and each column represents a word (term) from the combined features. The values in the matrix are the TF-IDF scores.

cosine_similarity(tfidf_matrix, tfidf_matrix):

The cosine_similarity function from sklearn.metrics.pairwise computes the cosine similarity between all pairs of rows in the tfidf_matrix.

By passing tfidf_matrix as both arguments, it calculates the pairwise cosine similarity for all movies with each other.

The Model

Here's the explanation of the get_recommendations function:

1) Get Movie Index:

Find the index of the movie that matches the provided title.

2) Calculate Similarity Scores:

Retrieve the cosine similarity scores for all movies with the selected movie.

enumerate pairs each movie's index with its similarity score.

3) Sort Similarity Scores:

Sort these similarity scores in descending order (most similar first).

4) Select Top Movies:

Select the top 10 most similar movies, excluding the first one (which is the movie itself).

5) Retrieve Movie Titles:

Get the indices of these top similar movies.

Return their titles from the dataframe.

```
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = df[df['title'] == title].index[0]

    # Get the pairwise similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
```

```
    return df['title'].iloc[movie_indices]
```

```
recommendations = get_recommendations('The Godfather') #As we input the name of the movie, we get the reccomendations.
recommendations
```

```
3            The Godfather Part II
52                  Apocalypse Now
16                      Goodfellas
135                         Casino
156                    Raging Bull
210                          Rocky
79     Once Upon a Time in America
128                Some Like It Hot
68           The Dark Knight Rises
108                           Heat
Name: title, dtype: object
```

```
recommendations = get_recommendations('The Dark Knight') #testing with other movie names
recommendations
```

```
68                             The Dark Knight Rises
126                                    Batman Begins
131                          The Wolf of Wall Street
179      Harry Potter and the Deathly Hallows: Part 2
88         Star Wars: Episode VI - Return of the Jedi
208                                   Ford v Ferrari
14     Star Wars: Episode V - The Empire Strikes Back
38                                      The Departed
6       The Lord of the Rings: The Return of the King
142                                 A Beautiful Mind
Name: title, dtype: object
```

```
recommendations = get_recommendations('12 Angry Men') #testing with other movie names
recommendations
```

```
94                   Citizen Kane
181              On the Waterfront
232             The Grapes of Wrath
196    Mr. Smith Goes to Washington
133           Judgment at Nuremberg
218                         Network
58                     Sunset Blvd.
103                Double Indemnity
176                   The Gold Rush
110                        The Sting
Name: title, dtype: object
```