Movie Recommendation System

Devanshi Nigam

Aim: The goal of the project was to help users find movies that they are likely to enjoy, making the movie-watching experience more personalized and enjoyable.

ML Model: To achieve this, I used Natural Language Processing (NLP).

Dataset: https://www.kaggle.com/datasets/akshaypawar7/millions-of-movies

These files contain metadata for more than 700,000 movies listed in the TMDB Dataset. The dataset Update daily to ensure updated movies dataset. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages, reviews, recommendations.

[3]:
```
data = pd.read_csv("/kaggle/input/millions-of-movies/movies.csv")
data
```

[3]:

	id	title	genres	original language	overview	popularity	production companies	release date
0	615656	Meg 2: The Trench	Action-Science Fiction-Horror	en	An exploratory dive into the deepest depths of...	8763.998	Apelles Entertainment-Warner Bros. Pictures-di...	2023-08-02
1	758323	The Pope's Exorcist	Horror-Mystery-Thriller	en	Father Gabriele Amorth Chief Exorcist of the V...	5953.227	Screen Gems-2.0 Entertainment-Jesus & Mary-Wor...	2023-04-05
2	667538	Transformers: Rise of the Beasts	Action-Adventure-Science Fiction	en	When a new threat capable of destroying the en...	5409.104	Skydance-Paramount-di Bonaventura Pictures-Bay...	2023-06-06
3	640146	Ant-Man and the Wasp: Quantumania	Action-Adventure-Science Fiction	en	Super-Hero partners Scott Lang and Hope van Dy...	4425.387	Marvel Studios-Kevin Feige Productions	2023-02-15
4	677179	Creed III	Drama-Action	en	After dominating the boxing world Adonis Creed...	3994.342	Metro-Goldwyn-Mayer-Proximity Media-Balboa Pro...	2023-03-01
...	...	...	...	...	...	...	...	...
722691	740445	Pandemia en Argentina	Documentary	es	NaN	0.600	Sucesos Argentinos	2020-09-06
722692	282567	Shriasthu Shubhamasthu	Romance	en	Shriasthu Shubhamasthu is a 2000 Kannada film ...	0.600	NaN	2000-08-10
722693	302134	Banda Ramudu	Drama	te	A thief saves a beautiful princess from the cl...	0.600	NaN	1959-11-06
722694	892477	The Drag Phenomenon	NaN	es	With more than 20 years of celebration the Dra...	0.600	NaN	2021-11-06
722695	968161	Gising Sining	NaN	tl	In a country where fascism foolishness of peop...	0.600	Studio RD	NaN

722696 rows × 20 columns

[4]:
```
print(data.shape)
```

```
(722696, 20)
```

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 722696 entries, 0 to 722695
Data columns (total 20 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   id                    722696 non-null  int64
 1   title                 722690 non-null  object
 2   genres                511964 non-null  object
 3   original_language     722696 non-null  object
 4   overview              604177 non-null  object
 5   popularity            722696 non-null  float64
 6   production_companies  337209 non-null  object
 7   release_date          670504 non-null  object
 8   budget                722696 non-null  float64
 9   revenue               722696 non-null  float64
 10  runtime               688266 non-null  float64
 11  status                722696 non-null  object
 12  tagline               108190 non-null  object
 13  vote_average          722696 non-null  float64
 14  vote_count            722696 non-null  float64
 15  credits               497685 non-null  object
 16  keywords              210361 non-null  object
 17  poster_path           537657 non-null  object
 18  backdrop_path         222705 non-null  object
 19  recommendations       34782 non-null   object
dtypes: float64(6), int64(1), object(13)
memory usage: 110.3+ MB
```

Data Cleaning:

[6]:
```
data.isnull().sum()
```

[6]:
```
id                        0
title                     6
genres                210732
original_language          0
overview              118519
popularity                 0
production_companies  385487
release_date           52192
budget                     0
revenue                    0
runtime                34430
status                     0
tagline               614506
vote_average               0
vote_count                 0
credits               225011
keywords              512335
poster_path           185039
backdrop_path         499991
recommendations       687914
dtype: int64
```

[7]:
```
data.duplicated().sum()
```

[7]:
```
0
```

Data Analysis:

For data analysis, I used Natural Language Processing (NLP) Machine Learning (ML) Technique. Steps involved are as follows:

i) Stemming

Stremming

Stemming is the process of reducing words to their base or root form.

[22]:
```
import nltk
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

[23]:
```
def stem(text):
    y = []
    for i in text.split():
        y.append(ps.stem(i))

    return " ".join(y)
```

[24]:
```
new_df["tags"] = new_df["tags"].apply(stem)
```

ii) Text Vectorization

Text Vectorization

Text vectorization is the process of converting text data into numerical vectors so that they can be used as input for machine learning models.

In [25]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(stop_words="english",max_features=2000)
```

In [26]:
```
vectors = cv.fit_transform(new_df["tags"]).toarray()
```

In [27]:
```
cv.get_feature_names()[80:85]
```

Out[27]:
```
['alter', 'altern', 'alway', 'amazon', 'ambit']
```

iii) ML Modelling

ML Modeling

Cosine Similarity is a process of creating a machine learning model that utilizes the cosine similarity metric to determine the similarity between two pieces of text.

In [28]:
```
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity(vectors)
```

In [29]:
```
similarity.shape
```

Out[29]:
```
(7659, 7659)
```

iv) Testing

Testing

In [30]:
```
def recommend(movies):
    movie_index = new_df[new_df.title == movies].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)), reverse=True, key= lambda x:x[1])[1:6]

    for i in movies_list:
        print(new_df.iloc[i[0]].title)
```

In [31]:
```
recommend("Toy Story")
```

```
Toy Story 4
Toy Story 2
Small Soldiers
The Indian in the Cupboard
Toy Story 3
```

In [32]:
```
recommend("The Conjuring")
```

```
The Conjuring: The Devil Made Me Do It
Grave Encounters
Terrified
A Haunted House 2
The Conjuring 2
```

Observation : The function is returning 5 similar movies.