MOVIE RECOMMENDATION SYSTEM

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

(Deemed to be University), Patiala – 147004

Machine Learning Project

Submitted By:

Devansh Gupta

102103028

Submitted To:

Ms. Kudratdeep Aulakh

Index

1. Introduction

1.1 TMDB 5000 Movie Dataset

https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

The TMDB 5000 Movie Dataset is a comprehensive collection of metadata for 5,000 movies, providing valuable insights into the world of filmmaking. This dataset can be utilized to explore trends, analyze ratings, study release patterns, and perform various research tasks related to movies. The dataset is readily available on Kaggle, making it accessible to data enthusiasts and researchers alike.

1.2 Description

The TMDB 5000 Movie Dataset encompasses a wide range of movies, spanning various genres, release dates, and popularity levels. It comprises 4803 rows and 20 columns, offering a rich source of information for data analysis and exploration. The key attributes of the dataset include:

- movieId: Unique identifier for each movie
- title: Title of the movie
- genres: List of genres associated with the movie
- overview : Description of the movie
- Keywords : Major words that can describe a movie
- Cast : lead cast of movie
- Crew : All the crew of movie

Applications and Use Cases

The TMDB 5000 Movie Dataset serves as a valuable resource for a variety of applications and use cases:

- **Movie Recommendation Systems:** Develop recommendation systems that suggest movies to users based on their preferences and the dataset.
- **Movie Trend Analysis:** Track and analyze trends in movie genres, popularity, and ratings to inform marketing and production decisions.
- **Movie Success Factors:** Identify factors that contribute to movie success, such as genre, cast, production budget, and marketing strategies.

- **Academic Research:** Conduct research on various aspects of filmmaking, such as genre evolution, audience preferences, and the impact of digital distribution.

Conclusion

The TMDB 5000 Movie Dataset is a treasure trove of information for anyone interested in the world of movies. Its diverse collection of metadata and accessible format make it an ideal resource for data analysis, exploration, and research. By delving into the dataset, researchers and enthusiasts can uncover valuable insights into movie trends, audience preferences, and the factors that drive success in the film industry.

2. Libraries Used

Following libraries are used in movie recommendation system Project

- **pandas:** Pandas is a library for data analysis and manipulation. It is used in this code to read the CSV files, merge the two datasets, clean the data, and create a new dataset with the desired features.
- **ast:** The ast library is used to evaluate string literals, which are used to store lists in the CSV files.
- **sklearn.feature_extraction.text:** The CountVectorizer class from this library is used to convert the text data in the 'tags' column into a numerical representation that can be used for machine learning.
- **sklearn.metrics.pairwise:** The cosine_similarity function from this library is used to calculate the cosine similarity between the vectors of movie tags. This is used to recommend movies that are similar to a given movie.
- **pickle**: The pickle library is used to save the movie dataset and cosine similarity matrix to disk, so that they can be reused later.

Usage of Libraries:

- pandas:
  - **pd.read_csv('tmdb_5000_movies.csv'):** Reads the CSV file containing movie metadata into a Pandas DataFrame.
  - **pd.read_csv('tmdb_5000_credits.csv'):** Reads the CSV file containing movie credits into a Pandas DataFrame.
  - **movies.merge(credits,on='title'):** Merges the two DataFrames based on the 'title' column.
  - **movies[['movie_id','title','overview','genres','keywords','cast','crew']]:** Selects only the desired columns from the DataFrame.
  - **movies.dropna(inplace=True):** Removes rows with missing values from the DataFrame.
  - **movies.sample(5):** Randomly samples 5 rows from the DataFrame.
- **ast:**
  - **ast.literal_eval(text):** Evaluates the string literal 'text' and returns the resulting Python object, which is usually a list.
- **sklearn.feature_extraction.text:**
  - **cv=CountVectorizer(max_features=5000,stop_words='english'):** Creates a CountVectorizer object with a maximum of 5000 features and removes English stop words.
  - **vector = cv.fit_transform(new['tags']).toarray():** Fits the CountVectorizer to the 'tags' column and transforms the text data into a numerical representation.

- **sklearn.metrics.pairwise:**
  - **similarity = cosine_similarity(vector):** Calculates the cosine similarity between the vectors of movie tags.
- **pickle:**
  - **pickle.dump(new,open('movie_list.pkl','wb')):** Saves the movie dataset to a file named 'movie_list.pkl'.
  - **pickle.dump(similarity,open('similarity.pkl','wb')):** Saves the cosine similarity matrix to a file named 'similarity.pkl'.

3. Algorithm(s) Used

Description of the algorithms used in the code, including a detailed explanation of each algorithm:

Algorithms Used:

- **Data Cleaning and Preprocessing:**

The code employs various data cleaning and preprocessing techniques to prepare the movie dataset for further analysis. These techniques include:

- **Handling Missing Values:** The code removes rows with missing values using the **pandas.DataFrame.dropna()** function. This ensures that the analysis is based on complete data.
- **Converting String Literals to Lists:** The code utilizes the **ast.literal_eval()** function to evaluate string literals, which are used to store lists in the CSV files. This converts the string representations of lists into actual Python lists.
- **Removing Stop Words:** The code utilizes the CountVectorizer from **sklearn.feature_extraction.text** to remove English stop words from the text data. Stop words are common words that do not add much meaning to the text and can be safely removed without affecting the analysis.
- **Feature Extraction:** The code employs the CountVectorizer algorithm to convert the text data in the 'tags' column into a numerical representation that can be used for machine learning. The CountVectorizer creates a vocabulary of unique words from the text data and then counts the occurrences of each word in each document (movie). This results in a matrix where each row represents a movie and each column represents a word in the vocabulary.

- **Cosine Similarity:**

  Cosine similarity is a widely used technique in recommender systems, particularly for content-based filtering. It measures the similarity between two vectors, in this case, the vectors of movie tags. The cosine of the angle between two vectors represents the degree to which they are pointing in the same direction. A higher cosine similarity indicates that the two vectors are more similar, meaning that the corresponding movies share more common tags and are likely to have similar content.

In the context of movie recommendation, cosine similarity is used to identify movies that are similar to a given movie based on their tags. The tags are extracted

from various sources, such as movie descriptions, keywords, cast, and crew information. These tags are then converted into a numerical representation using a technique called feature extraction.

The feature extraction process transforms the text data into a matrix where each row represents a movie and each column represents a unique tag. This matrix is then used to calculate the cosine similarity between each pair of movies. The resulting cosine similarity matrix provides a measure of similarity between all movies in the dataset.

Cosine similarity is a versatile and effective technique for movie recommendation because it captures the semantic similarity between movies based on their content. It is particularly useful when dealing with text data, such as movie descriptions, keywords, and cast information. By identifying movies with high cosine similarity to a user's preferred movies, cosine similarity-based recommender systems can provide personalized and relevant recommendations.

4. Code and Screenshots

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```
✓ 0.0s

```
movies = pd.read_csv('tmdb_5000_movies.csv')
credits = pd.read_csv('tmdb_5000_credits.csv')
```
✓ 0.4s

```
movies.info()
```
[86] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4803 non-null   int64  
 1   genres                4803 non-null   object 
 2   homepage              1712 non-null   object 
 3   id                    4803 non-null   int64  
 4   keywords              4803 non-null   object 
 5   original_language     4803 non-null   object 
 6   original_title        4803 non-null   object 
 7   overview              4800 non-null   object 
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object 
 10  production_countries  4803 non-null   object 
 11  release_date          4802 non-null   object 
 12  revenue               4803 non-null   int64  
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object 
 15  status                4803 non-null   object 
 16  tagline               3959 non-null   object 
 17  title                 4803 non-null   object 
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64  
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

```
movies.head(1)
```

```
movies.shape
```

✓ 0.0s

```
(4803, 20)
```

```
credits.head(1)
```

✓ 0.0s

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

```
# Merging the two database on their tittle
movies = movies.merge(credits,on='title')
```

✓ 0.0s

Removing some of the columunn which wouuld not be needed in further process

```
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
```

✓ 0.0s Python

```
movies.head(1)
```

✓ 0.0s Python

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

```python
import ast
```

✓ 0.0s Python

```python
def convert(text):
    L = []
    for i in ast.literal_eval(text):
        L.append(i['name'])
    return L
```

✓ 0.0s Python

```python
movies.dropna(inplace=True)
```

✓ 0.0s Python

```python
movies['genres'] = movies['genres'].apply(convert)
movies.head()
```

✓ 0.1s Python

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

```python
movies['keywords'] = movies['keywords'].apply(convert)
movies.head()
```

✓ 0.3s Python

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

```python
def convert3(text):
    L = []
    counter = 0
    for i in ast.literal_eval(text):
        if counter < 3:
            L.append(i['name'])
        counter+=1
    return L
```

✓ 0.0s Python

```python
movies['cast'] = movies['cast'].apply(convert)
movies.head(1)
```

✓ 3.1s Python

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weave...	[{"credit_id": "52fe48009251416c750aca23", "de...

```python
movies['cast'] = movies['cast'].apply(lambda x:x[0:3])
```

✓ 0.0s Python

```python
def fetch_director(text):
    L = []
    for i in ast.literal_eval(text):
        if i['job'] == 'Director':
            L.append(i['name'])
    return L
```

✓ 0.0s Python

```python
movies['crew'] = movies['crew'].apply(fetch_director)
```

✓ 3.0s Python

```python
#movies['overview'] = movies['overview'].apply(lambda x:x.split())
movies.sample(5)
```

✓ 0.0s Python

```
#movies['overview'] = movies['overview'].apply(lambda x:x.split())
movies.sample(5)
```

✓ 0.0s Python

	movie_id	title	overview	genres	keywords	cast	crew
2469	318850	The Young Messiah	Tells the story of Jesus Christ at age seven a...	[Drama]	[egypt, jesus christ, gospel, christian, journ...	[Adam Greaves-Neal, Sara Lazzaro, Vincent Walsh]	[Cyrus Nowrasteh]
1151	1885	The Karate Kid	Hassled by the school bullies, Daniel LaRusso ...	[Drama]	[flat, taskmaster, karate, egg, kids and famil...	[Ralph Macchio, Pat Morita, William Zabka]	[John G. Avildsen]
3382	104896	A Dog Of Flanders	Poor but happy, young Nello and his grandfathe...	[Drama, Family]	[]	[Jack Warden, Jeremy James Kissner, Jesse James]	[Kevin Brodie]
872	1717	All the King's Men	The story of an idealist's rise to power in th...	[Drama, Thriller]	[corruption, journalist, based on novel, black...	[Sean Penn, Jude Law, Kate Winslet]	[Steven Zaillian]
2960	13994	Romance & Cigarettes	Down-and-dirty musical love story set in the w...	[Comedy, Music, Romance]	[infidelity, lovers, working class, new york c...	[James Gandolfini, Susan Sarandon, Kate Winslet]	[John Turturro]

```
movies['cast'] = movies['cast'].apply(collapse)
movies['crew'] = movies['crew'].apply(collapse)
movies['genres'] = movies['genres'].apply(collapse)
movies['keywords'] = movies['keywords'].apply(collapse)
```

✓ 0.0s Python

```
movies.head()
```

✓ 0.0s Python

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes]
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dccomics, crimefighter, terrorist, secretiden...	[ChristianBale, MichaelCaine, GaryOldman]	[ChristopherNolan]

```
movies['overview'] = movies['overview'].apply(lambda x:x.split())
```
✓ 0.0s Python

```
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']
```
✓ 0.0s Python

```
new = movies.drop(columns=['overview','genres','keywords','cast','crew'])
new.head()
```
✓ 0.0s Python

	movie_id	title	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...

```
new['tags'] = new['tags'].apply(lambda x: " ".join(x))
new.head()
```
✓ 0.0s Python

	movie_id	title	tags
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...
2	206647	Spectre	A cryptic message from Bond's past sends him o...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...
4	49529	John Carter	John Carter is a war-weary, former military ca...

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
```
✓ 0.0s Python

```
vector = cv.fit_transform(new['tags']).toarray()
```
✓ 0.4s Python

```
vector.shape
```
✓ 0.0s

```
(4806, 5000)
```

```
from sklearn.metrics.pairwise import cosine_similarity
```
✓ 0.0s

```
similarity = cosine_similarity(vector)
```
✓ 0.8s

```
similarity
```
✓ 0.0s

```
array([[1.        , 0.08964215, 0.06071767, ..., 0.02519763, 0.0277885 ,
        0.        ],
       [0.08964215, 1.        , 0.06350006, ..., 0.02635231, 0.        ,
        0.        ],
       [0.06071767, 0.06350006, 1.        , ..., 0.02677398, 0.        ,
        0.        ],
       ...,
       [0.02519763, 0.02635231, 0.02677398, ..., 1.        , 0.07352146,
        0.04774099],
```

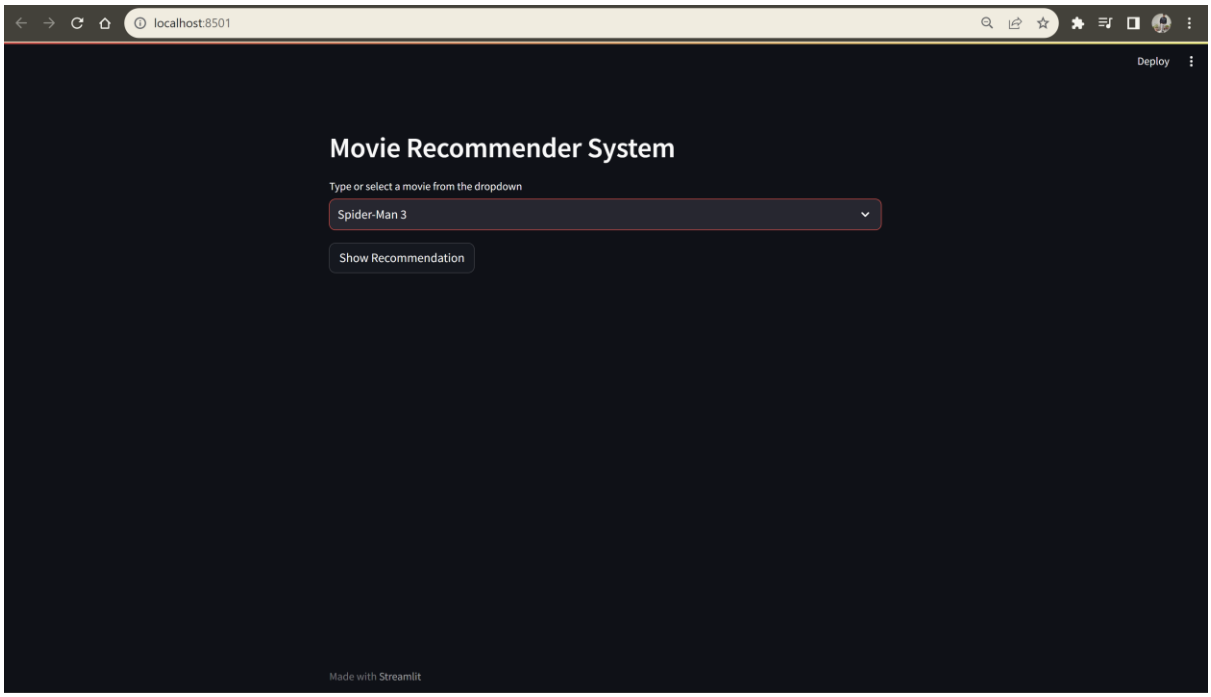
```
new[new['title'] == 'The Lego Movie'].index[0]
```
✓ 0.0s Python

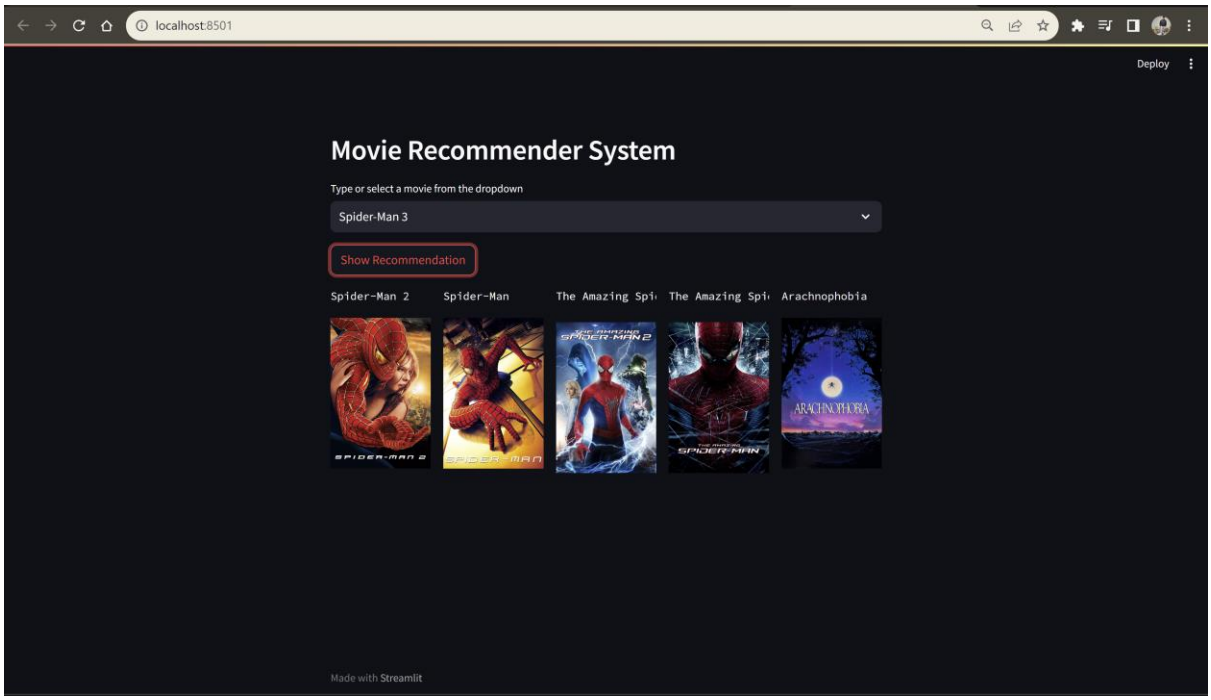
```
744
```

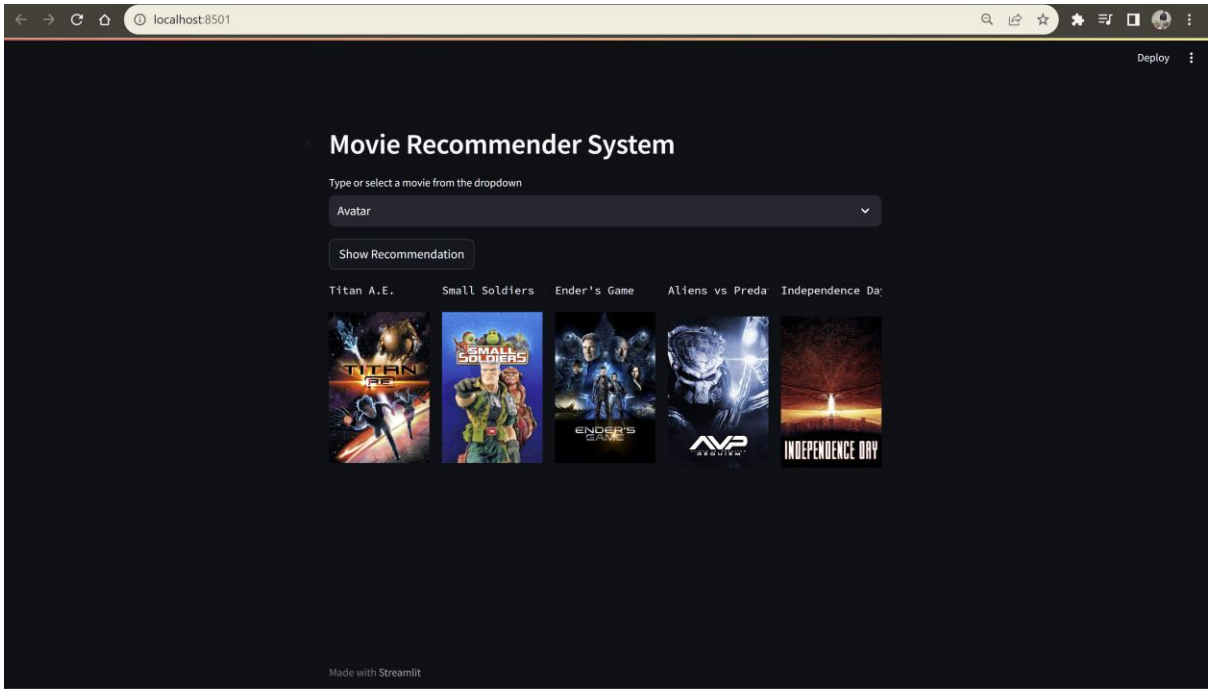
```
def recommend(movie):
    index = new[new['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])
    for i in distances[1:6]:
        print(new.iloc[i[0]].title)
```
✓ 0.0s Python

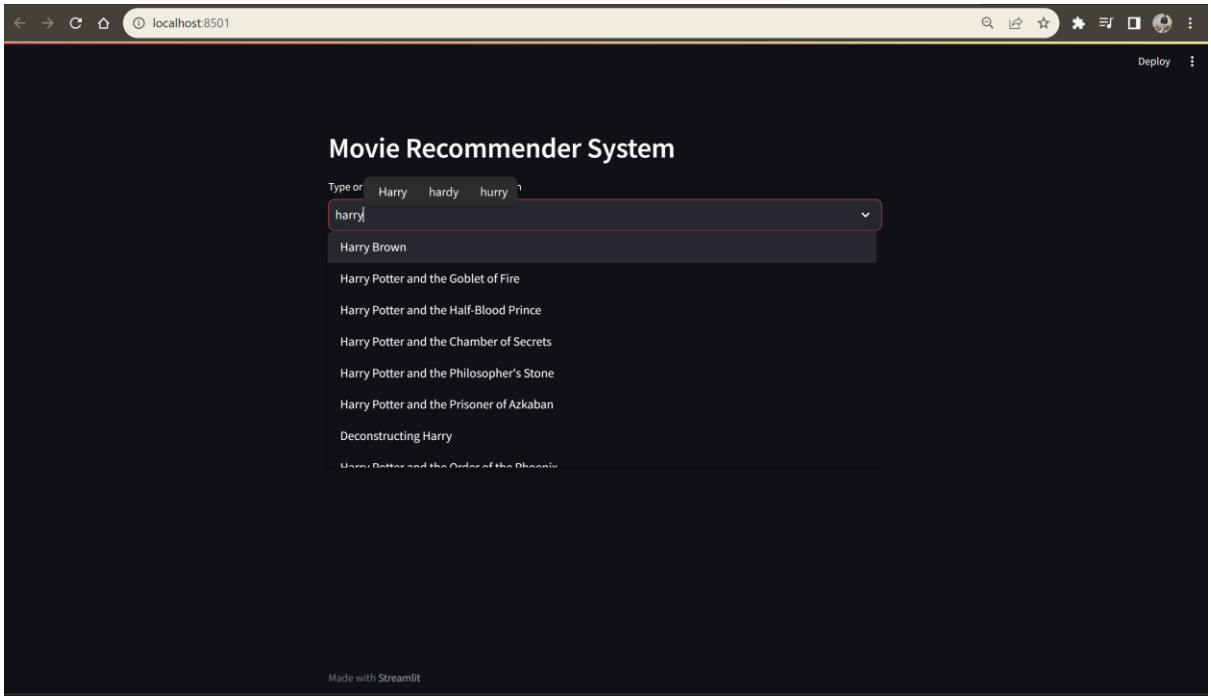
```
recommend('Independence Day')
```
✓ 0.0s Python

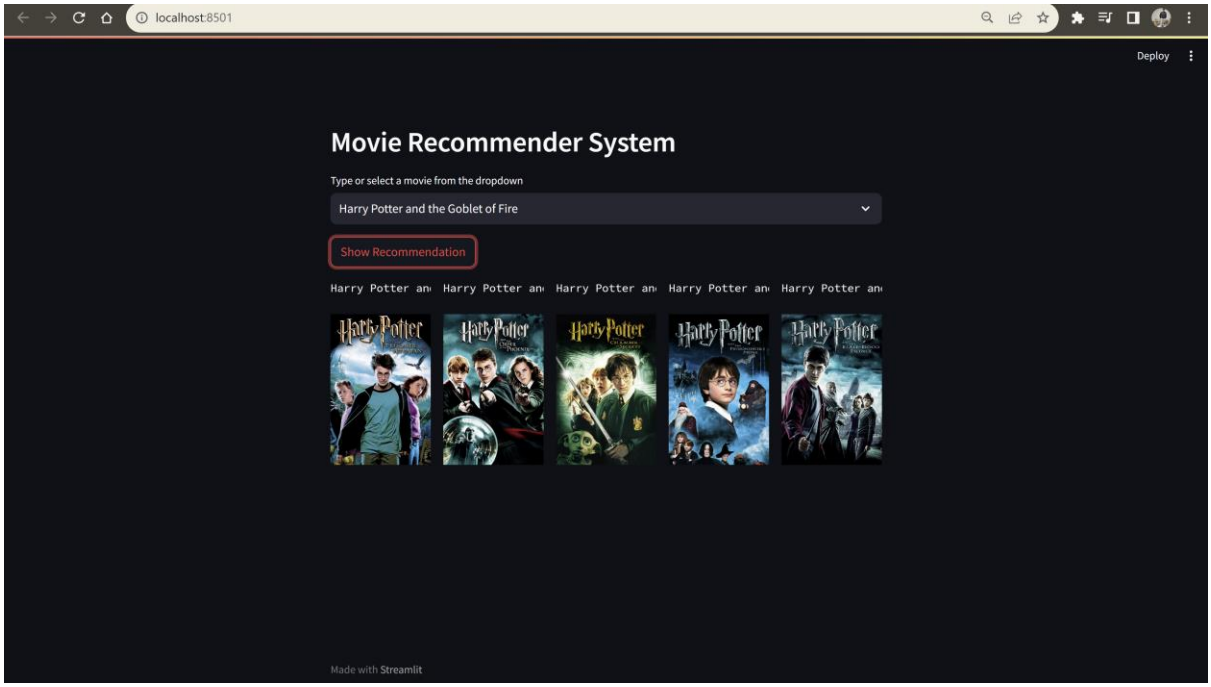
```
Independence Daysaster
Escape from Planet Earth
Independence Day: Resurgence
The Day the Earth Stood Still
Alien
```


localhost:8501
Deploy
Movie Recommender System
Type or select a movie from the dropdown
Spider-Man 3
Show Recommendation
Made with Streamlit


localhost:8501
Deploy
Movie Recommender System
Type or select a movie from the dropdown
Spider-Man 3
Show Recommendation
Spider-Man 2
Spider-Man
The Amazing Spi
The Amazing Spi
Arachnophobia
Made with Streamlit


localhost:8501
Deploy
Movie Recommender System
Type or select a movie from the dropdown
Avatar
Show Recommendation
Titan A.E.
Small Soldiers
Ender's Game
Aliens vs Preda
Independence Da
Made with Streamlit


localhost:8501
Deploy
Movie Recommender System
Type or
Harry
hardy
hurry
harry
Harry Brown
Harry Potter and the Goblet of Fire
Harry Potter and the Half-Blood Prince
Harry Potter and the Chamber of Secrets
Harry Potter and the Philosopher's Stone
Harry Potter and the Prisoner of Azkaban
Deconstructing Harry
Made with Streamlit


localhost:8501
Deploy
Movie Recommender System
Type or select a movie from the dropdown
Harry Potter and the Goblet of Fire
Show Recommendation
Harry Potter an
Harry Potter an
Harry Potter an
Harry Potter an
Harry Potter an
Made with Streamlit