Section 1

In [1]:
```python
# import libraries

import pandas as pd
import numpy as np
from surprise import Dataset, Reader, SVD
from surprise.model_selection import train_test_split
```

In [2]:
```python
# load datasets

movies_df=pd.read_csv("movies.csv")
ratings_df=pd.read_csv("ratings.csv")
```

In [3]:
```python
movies_df.head()
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [4]:
```python
ratings_df.head()
```

Out[4]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [5]:
```python
# merging two datasets based on movie id to create a final dataset

df=pd.merge(movies_df,ratings_df, on='movieId', how='inner')
df.head()
```

Out[5]:

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1	4.0	964982703
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	5	4.0	847434962
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	7	4.5	1106635946
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	15	2.5	1510577970
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	17	4.5	1305696483

In [6]:
```python
# there might be more than one rating by a user to a particular movie, so lets find that out
groupby_movie_userid = df.groupby(['userId', 'movieId']).size().reset_index(name='count')
multiple_rating = groupby_movie_userid[groupby_movie_userid['count'] > 1]
multiple_rating
```

Out[6]:

userId	movieId	count

There aren't any scenario where one user has multiple review for the same movie.

In [7]:
```python
# find the minimum and maximum ratings
print(ratings_df['rating'].min())
print(ratings_df['rating'].max())
```

```
0.5
5.0
```

The rating scale is set from 0.5 to 5, indicating that movie ratings are expected to fall within this range.

In [8]:
```python
# now, let's reate a reader object to parse the dataset
reader = Reader(rating_scale=(0.5, 5))
```

In [9]:
```python
# loading the dataset into Surprise's data format
data = Dataset.load_from_df(df[['userId', 'movieId', 'rating']], reader=reader)
```

In [10]:
```python
# Split the data into training and testing sets
trainset, testset = train_test_split(data, test_size=0.2)
```

In [11]:
```python
#using the SVD algorithm for collaborative filtering
algorithm = SVD()

# training the algorithm on the training set
algorithm.fit(trainset)
```

Out[11]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7fc719c7e130>
```

In [16]:
```python
def get_movie_recommendations(movie_title_user):
    num_recommendations = 10
    # Convert movie title to lowercase for case insensitivity
    # so the user can enter in either lower or upper case
    movie_title = movie_title_user.lower()

    # Check if the movie exists in the dataset
    if movie_title not in df['title'].str.lower().unique():
        print(f"{movie_title_user} not found.")
        raise ValueError("Movie not found.")

    # Find movie ID for the given movie title
    movie_id = df.loc[df['title'].str.lower() == movie_title, 'movieId'].values[0]

    # Predict ratings for all movies for the given user
    movie_ratings = []
    for movie_id in trainset.all_items():
        predicted_rating = algorithm.predict(uid=trainset.to_raw_uid(0), iid=trainset.to_raw_iid(movie_id)).est
        movie_ratings.append((movie_id, predicted_rating))

    # Sort the movies based on predicted ratings
    movie_ratings.sort(key=lambda x: x[1], reverse=True)

    # Get the top 10 or num_recommendations movies with the highest predicted ratings
    top_movies = movie_ratings[:num_recommendations]
    print("=================================================")
    print(f"Top {num_recommendations} Movies recommendation based on '{movie_title_user}' are:")
    print("=================================================")
    for movie in top_movies:
        movie_id = movie[0]
        movie_title = df[df['movieId'] == movie_id]['title'].values
        if len(movie_title) > 0:
            print(movie_title[0])

    return None
```

In [17]:
```python
# Get user entry and recommend movies
movie_title = input("Enter a movie name: ")

recommendations = get_movie_recommendations(movie_title)
```

```
Enter a movie name: Pollock (2000)
=================================================
Top 10 Movies recommendation based on 'Pollock (2000)' are:
=================================================
Amazing Panda Adventure, The (1995)
Drop Zone (1994)
Little Nikita (1988)
Smoke Signals (1998)
Houseguest (1994)
Priest (1994)
Ilsa, She Wolf of the SS (1974)
Juror, The (1996)
Lord of Illusions (1995)
Three Musketeers, The (1993)
```

Section 2

Introduction

The movie recommendation system project utilizes collaborative filtering techniques to provide personalized movie recommendations to users. The project involves various stages, starting from data acquisition and preprocessing to model training and recommendation generation. This write-up outlines the key stages of the project.

Data Acquisition

The first stage of the project is acquiring the necessary data. In this case, the data includes movie information and user ratings acquired from https://grouplens.org/datasets/movielens/. The "movies.csv" and "ratings.csv" datasets are loaded using the pandas library, and the two datasets are merged based on the movie ID to create the final dataset for analysis.

Data Preprocessing:

Once the data is acquired, preprocessing steps are performed to prepare it for analysis. Data preprocessing includes handling missing values, cleaning the data, and ensuring data consistency. In this project, the merging of the movies and ratings datasets ensures that only movies with valid ratings are considered.

Exploratory Data Analysis (EDA)

EDA is an essential stage that involves analyzing the dataset to gain insights and understanding.In this project, the EDA stage includes examining multiple ratings given by users to a particular movie, finding the minimum and maximum ratings, and exploring the dataset's structure.

Model Training

The next stage involves training a recommendation model using collaborative filtering techniques. Surprise, a Python library for recommender systems, is utilized in this project. The Surprise library provides various algorithms, and in this case, the SVD (Singular Value Decomposition) algorithm is employed for collaborative filtering. The training data is split into training and testing sets using the train_test_split function.

Recommendation Generation

Once the model is trained, the recommendation generation stage begins. Users are prompted to enter a movie title, and the system generates personalized recommendations based on the input. The entered movie title is matched with the dataset, and if found, the model predicts ratings for all movies using the trained algorithm. The movies are then sorted based on the predicted ratings, and the top movies with the highest predicted ratings are recommended to the user.

Result Presentation

The final stage involves presenting the recommendations to the user. In this project, the recommended movies are printed on the console, along with a message indicating the number of recommendations and the user's input.

Conclusion

T he movie recommendation system project follows a structured workflow, starting from data acquisition and preprocessing to model training and recommendation generation. Each stage plays a crucial role in developing an effective and personalized recommendation system. By following this project workflow, data scientists can build robust recommendation systems to provide tailored suggestions to users based on their preferences.

In [ ]: