In [1]:
```
# William Barker
# DSC630
# Week 10

import pandas as pd

# Load the ratings data
ratings = pd.read_csv('ratings.csv')

# Load the movies data
movies = pd.read_csv('movies.csv')

# Merge ratings and movies data
data = pd.merge(ratings, movies, on='movieId')
```

In [2]:
```
data
```

Out[2]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	5	1	4.0	847434962	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	7	1	4.5	1106635946	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	15	1	2.5	1510577970	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
4	17	1	4.5	1305696483	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
...	...	...	...	...	...	...
100831	610	160341	2.5	1479545749	Bloodmoon (1997)	Action|Thriller
100832	610	160527	4.5	1479544998	Sympathy for the Underdog (1971)	Action|Crime|Drama
100833	610	160836	3.0	1493844794	Hazard (2005)	Action|Drama|Thriller
100834	610	163937	3.5	1493848789	Blair Witch (2016)	Horror|Thriller
100835	610	163981	3.5	1493850155	31 (2016)	Horror

100836 rows × 6 columns

In [3]:
```
from sklearn.model_selection import train_test_split

# Split data into training and test sets (80% training, 20% test)
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)
```

In [4]:
```
# Apparenlty collaborative filtering is a popular technique for building recommender systems that use
# past behavior to make recommendations. The surprise library provides a collection of algorithms for
# recommendation systems
```

localhost
Content | VM instances – Compute Engine – My Fir... | Content | ChatGPT | Home Page - Select or create a notebook | BarkerWilliamDSC630Week10 - Jupyter...
jupyter BarkerWilliamDSC630Week10 Last Checkpoint: a few seconds ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
Run Code

In [4]:
```
# Apparenlty collaborative filtering is a popular technique for building recommender systems that use
# past behavior to make recommendations. The surprise library provides a collection of algorithms for
# recommendation systems

# pip install scikit-surprise
```

```
Requirement already satisfied: scikit-surprise in ./opt/anaconda3/lib/python3.9/site-packages (1.1.3)
Requirement already satisfied: joblib>=1.0.0 in ./opt/anaconda3/lib/python3.9/site-packages (from scikit-surprise)
(1.3.2)
Requirement already satisfied: numpy>=1.17.3 in ./opt/anaconda3/lib/python3.9/site-packages (from scikit-surprise)
(1.22.4)
Requirement already satisfied: scipy>=1.3.2 in ./opt/anaconda3/lib/python3.9/site-packages (from scikit-surprise) (
1.7.1)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
# conda install "numpy>=1.16.5,<1.23.0"
```

```
zsh:1: 1.16.5, not found

Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
# Building our model and training it on our data

from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
import numpy as np

# Create a Reader object to parse the ratings data
reader = Reader(rating_scale=(0.5, 5))

# Load the training data into Surprise's Dataset format
train_dataset = Dataset.load_from_df(train_data[['userId', 'movieId', 'rating']], reader)

# Build the collaborative filtering model (SVD algorithm)
model = SVD()

# Evaluate the model using cross-validation
cross_validate(model, train_dataset, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8742  0.8767  0.8923  0.8867  0.8804  0.8820  0.0066
MAE (testset)     0.6712  0.6772  0.6895  0.6802  0.6766  0.6789  0.0060
Fit time          0.54    0.54    0.57    0.59    0.56    0.56    0.02
Test time         0.08    0.08    0.12    0.08    0.11    0.10    0.02
```

Out[6]:
```
{'test_rmse': array([0.87419585, 0.87665649, 0.8922696 , 0.88674616, 0.88035873]),
 'test_mae': array([0.671155  , 0.67715068, 0.68946982, 0.68024742, 0.67658834]),
 'fit_time': (0.5411787033081055,
  0.5426898002624512,
  0.5660197734832764,
```

```
0.08462905883789062,
0.11308908462524414)}
```

In [7]:
```python
# Defining a function to recommend movies based on a movie liked by the user

def get_movie_recommendations(movie_title, model, movies, data, n=10):
    # Get the movieId of the input movie title
    movie_id = movies[movies['title'] == movie_title]['movieId'].iloc[0]

    # Get all ratings of the input movie
    movie_ratings = data[data['movieId'] == movie_id]

    # Predict ratings for all movies for the target user (userId=0 for simplicity)
    user_id = 0
    predictions = []
    for movie_id in data['movieId'].unique():
        prediction = model.predict(user_id, movie_id)
        predictions.append((movie_id, prediction.est))

    # Sort predictions in descending order of predicted ratings
    predictions.sort(key=lambda x: x[1], reverse=True)

    # Get the top n recommended movie titles
    recommended_movies = []
    for movie_id, _ in predictions[:n]:
        recommended_movie = movies[movies['movieId'] == movie_id]['title'].iloc[0]
        recommended_movies.append(recommended_movie)

    return recommended_movies
```

In [14]:
```python
# Get user input and then apply the function we defined previously

liked_movie = input("Please input a movie you like with the year it released in parenthesis: ")
recommended_movies = get_movie_recommendations(liked_movie, model, movies, data)

print(f"Top 10 recommended movies based on '{liked_movie}':")
for movie in recommended_movies:
    print(movie)
```

```
Please input a movie you like: Toy Story (1995)
Top 10 recommended movies based on 'Toy Story (1995)':
Shawshank Redemption, The (1994)
Godfather: Part II, The (1974)
Fight Club (1999)
Spirited Away (Sen to Chihiro no kamikakushi) (2001)
Departed, The (2006)
Godfather, The (1972)
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
Rear Window (1954)
Great Escape, The (1963)
Boondock Saints, The (2000)
```