Movie Recommendation System

Project Title: Movie Recommendation System

Submitted By: A. Angelin Selvi

Course Name: Data Science with Python

Institution: FIIT Training institute

Date: 15-1-2025

Abstract

This project focuses on building a movie recommendation system to suggest movies based on user preferences. Utilizing collaborative filtering techniques, the system employs user ratings and cosine similarity to find similar movies. The results demonstrate the potential of machine learning in personalized content recommendations.

Table of Contents

Introduction

Recommendation systems are pivotal in modern digital platforms for enhancing user experiences. This project implements a collaborative filtering-based movie recommendation system using Python. The system suggests movies similar to a user's input, showcasing the effectiveness of machine learning in real-world applications.

Data Description

1. **Movies Dataset**:
    - **File Name:** movies.csv

- **Attributes:**
  - movieId: Unique identifier for each movie.
  - title: Movie title.
  - genres: Genre categories.
- **Size:** 9742 rows x 3 columns.

2. **Ratings Dataset**:
   - **File Name:** ratings.csv
   - **Attributes:**
     - userId: Unique identifier for users.
     - movieId: Identifier linking to the movies dataset.
     - rating: User-assigned rating.
     - timestamp: Time of the rating.
   - **Size:** 100836 rows x 4 columns.

Methodology

1. **Data Preprocessing:**
   - Pivot the ratings dataset to create a user-movie matrix.
   - Handle missing values by filling them with zeroes.
2. **Filtering Data:**
   - Retain movies with more than 10 ratings.
   - Retain users who have rated more than 50 movies.
3. **Model:**
   - Used k-Nearest Neighbors (k-NN) with cosine similarity to identify similar movies.

Implementation

- **Libraries Used:** Pandas, NumPy, Matplotlib, Scikit-learn.
- **Key Functions:**
  - get_movie_recommendation(movie_name): Returns a list of recommended movies based on the input title.
- **Example Query:**
  - Input: "Iron Man"

- Output: Suggestions such as *Up (2009)*, *Guardians of the Galaxy (2014)*, *The Dark Knight (2008)*, etc

---

Project Work Flow: (explained each line)

1. **Import Libraries which we'll use in this project. I've used Pandas, Matplotlib, Numpy, Scipy, Scikit-learn.**
2. **Import the files we have our raw data on. I'm using 2 datasets as raw file.**

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        from scipy.sparse import csr_matrix
        from sklearn.neighbors import NearestNeighbors
```

```
In [2]: movies = pd.read_csv('C:\projects\movies1.csv')
```

```
In [3]: ratings = pd.read_csv('c:\projects\score1.csv')
```

3. **Lets use .head(), .tail() to see the top and bottom 5 in list from movies dataset**

```
In [4]: movies.head

Out[4]: <bound method NDFrame.head of       movieId
        0          1                              Toy Story (1995)
        1          2                                Jumanji (1995)
        2          3                       Grumpier Old Men (1995)
        3          4                      Waiting to Exhale (1995)
        4          5            Father of the Bride Part II (1995)
        ...      ...                                           ...
        9737  193581  Black Butler: Book of the Atlantic (2017)
        9738  193583               No Game No Life: Zero (2017)
        9739  193585                               Flint (2017)
        9740  193587        Bungo Stray Dogs: Dead Apple (2018)
        9741  193609        Andrew Dice Clay: Dice Rules (1991)

                                                 genres
        0     Adventure|Animation|Children|Comedy|Fantasy
        1                      Adventure|Children|Fantasy
        2                                  Comedy|Romance
        3                            Comedy|Drama|Romance
        4                                          Comedy
        ...                                           ...
        9737             Action|Animation|Comedy|Fantasy
        9738                    Animation|Comedy|Fantasy
        9739                                       Drama
        9740                            Action|Animation
        9741                                      Comedy

        [9742 rows x 3 columns]>
```

In [5]:
```
movies.tail
```

Out[5]:
```
<bound method NDFrame.tail of       movieId                                      title  \
0           1                           Toy Story (1995)
1           2                             Jumanji (1995)
2           3                    Grumpier Old Men (1995)
3           4                   Waiting to Exhale (1995)
4           5         Father of the Bride Part II (1995)
...       ...                                        ...
9737   193581  Black Butler: Book of the Atlantic (2017)
9738   193583               No Game No Life: Zero (2017)
9739   193585                               Flint (2017)
9740   193587        Bungo Stray Dogs: Dead Apple (2018)
9741   193609        Andrew Dice Clay: Dice Rules (1991)

                                           genres
0     Adventure|Animation|Children|Comedy|Fantasy
1                      Adventure|Children|Fantasy
2                                  Comedy|Romance
3                            Comedy|Drama|Romance
4                                          Comedy
...                                           ...
9737              Action|Animation|Comedy|Fantasy
9738                     Animation|Comedy|Fantasy
9739                                        Drama
9740                             Action|Animation
9741                                       Comedy

[9742 rows x 3 columns]>
```

4. **Lets use .head(), .tail() to see the top and bottom 5 in list from ratings dataset**

In [6]:
```
ratings.head
```

Out[6]:
```
<bound method NDFrame.head of         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
...        ...      ...     ...         ...
100831     610   166534     4.0  1493848402
100832     610   168248     5.0  1493850091
100833     610   168250     5.0  1494273047
100834     610   168252     5.0  1493846352
100835     610   170875     3.0  1493846415

[100836 rows x 4 columns]>
```

```
In [7]: ratings.tail
```

```
Out[7]: <bound method NDFrame.tail of         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
...        ...      ...     ...         ...
100831     610   166534     4.0  1493848402
100832     610   168248     5.0  1493850091
100833     610   168250     5.0  1494273047
100834     610   168252     5.0  1493846352
100835     610   170875     3.0  1493846415

[100836 rows x 4 columns]>
```

5. **Checking at Info to know more about the dataset.**

```
In [8]: print(movies.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  9742 non-null   int64
 1   title    9742 non-null   object
 2   genres   9742 non-null   object
dtypes: int64(1), object(2)
memory usage: 228.5+ KB
None
```

6. **Movies dataset!**

```
In [9]: print('datatypes')
        print(movies.dtypes)
        print('size')
        print(movies.size)
        print('shape')
        print(movies.shape)
        print('columns')
        print(movies.columns)
```

```
datatypes
movieId     int64
title      object
genres     object
dtype: object
size
29226
shape
(9742, 3)
columns
Index(['movieId', 'title', 'genres'], dtype='object')
```

7. **Ratings dataset!**

In [10]:
```
print('datatypes')
print(ratings.dtypes)
print('size')
print(ratings.size)
print('shape')
print(ratings.shape)
print('columns')
print(ratings.columns)
```

```
datatypes
userId         int64
movieId        int64
rating       float64
timestamp      int64
dtype: object
size
403344
shape
(100836, 4)
columns
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

8. **Now Let's add 2 dataset to a single dataset and name it Final_dataset**
9. **View the first 5 of this final_dataset**

In [14]:
```
final_dataset = ratings.pivot(index='movieId',columns='userId',values='rating')
final_dataset.head()
```

Out[14]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	NaN	NaN	NaN	4.0	NaN	4.5	NaN	NaN	NaN	...	4.0	NaN	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	NaN	NaN	NaN	NaN	NaN	4.0	NaN	4.0	NaN	NaN	...	NaN	4.0	NaN	5.0	3.5	NaN	NaN	2.0	NaN	NaN
3	4.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 610 columns

10. **Let's clean the dataset by doing noise removal process (here noise refers to incomplete data)**

In [15]:
```
final_dataset.fillna(0,inplace=True)
final_dataset.head()
```

Out[15]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	0.0	...	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0	0.0
3	4.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0

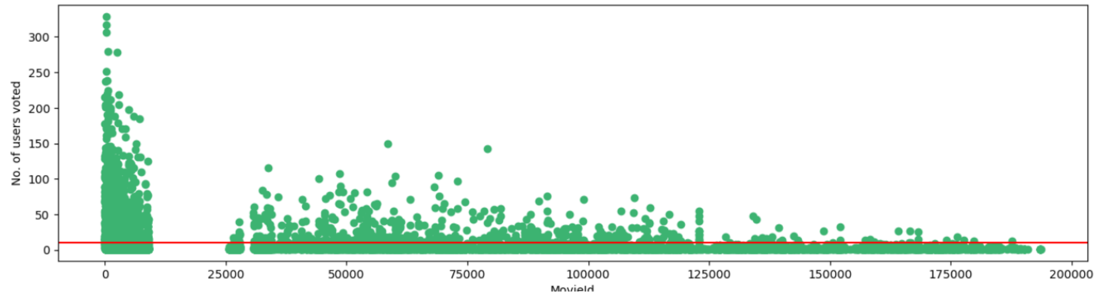
5 rows × 610 columns

11. Let's look at the lowest rated movies through this which can be used for filtering if needed.

In [16]:
```
no_user_voted = ratings.groupby('movieId')['rating'].agg('count')
no_movies_voted = ratings.groupby('userId')['rating'].agg('count')
```

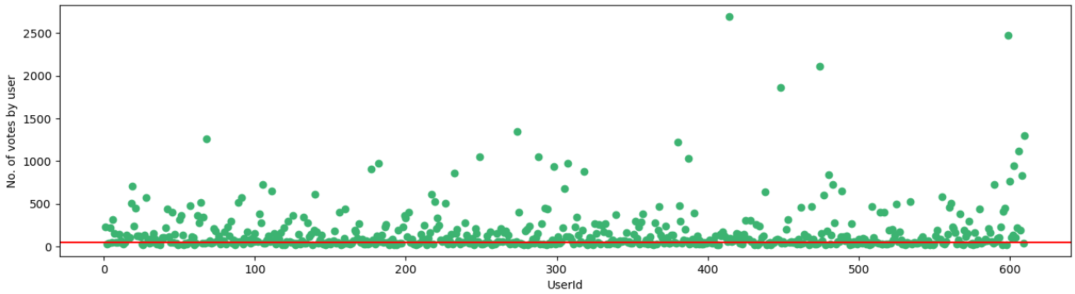
12. Let's visualize using plot diagram

In [17]:
```
f,ax = plt.subplots(1,1,figsize=(16,4))
# ratings['rating'].plot(kind='hist')
plt.scatter(no_user_voted.index,no_user_voted,color='mediumseagreen')
plt.axhline(y=10,color='r')
plt.xlabel('MovieId')
plt.ylabel('No. of users voted')
plt.show()
```



In [18]:
```
final_dataset = final_dataset.loc[no_user_voted[no_user_voted > 10].index,:]
```

In [19]:
```
f,ax = plt.subplots(1,1,figsize=(16,4))
plt.scatter(no_movies_voted.index,no_movies_voted,color='mediumseagreen')
plt.axhline(y=50,color='r')
plt.xlabel('UserId')
plt.ylabel('No. of votes by user')
plt.show()
```



13. Check the Index of final Dataset

In [20]:
```
final_dataset=final_dataset.loc[:,no_movies_voted[no_movies_voted > 50].index]
final_dataset
```

Out[20]:

userId	1	4	6	7	10	11	15	16	17	18	...	600	601	602	603	604	605	606	607	608	610
movieId																					
1	4.0	0.0	0.0	4.5	0.0	0.0	2.5	0.0	4.5	3.5	...	2.5	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	5.0
2	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	...	4.0	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0
3	4.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
5	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.5	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0
6	4.0	0.0	4.0	0.0	0.0	5.0	0.0	0.0	0.0	4.0	...	0.0	0.0	3.0	4.0	3.0	0.0	0.0	0.0	0.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
174055	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
176371	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
177765	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
179819	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
187593	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2121 rows × 378 columns

14. **Check Sparsity**

```
In [21]: sample = np.array([[0,0,3,0,0],[4,0,0,0,2],[0,0,0,0,1]])
         sparsity = 1.0 - ( np.count_nonzero(sample) / float(sample.size) )
         print(sparsity)

0.7333333333333334
```

15. **Compressed Sparse row: used to save dataset as matrices**

```
In [22]: csr_sample = csr_matrix(sample)
         print(csr_sample)

  (0, 2)	3
  (1, 0)	4
  (1, 4)	2
  (2, 4)	1
```

```
In [23]: csr_data = csr_matrix(final_dataset.values)
         final_dataset.reset_index(inplace=True)
```

16. **Using KNN algorithm**

```
In [24]: knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20, n_jobs=-1)
         knn.fit(csr_data)

Out[24]: ▾ NearestNeighbors
         NearestNeighbors(algorithm='brute', metric='cosine', n_jobs=-1, n_neighbors=20)
```

17. **Code that works as recommendation system.**

```
In [25]: def get_movie_recommendation(movie_name):
             n_movies_to_reccomend = 10
             movie_list = movies[movies['title'].str.contains(movie_name)]
             if len(movie_list):
                 movie_idx= movie_list.iloc[0]['movieId']
                 movie_idx = final_dataset[final_dataset['movieId'] == movie_idx].index[0]
                 distances , indices = knn.kneighbors(csr_data[movie_idx],n_neighbors=n_movies_to_reccomend+1)
                 rec_movie_indices = sorted(list(zip(indices.squeeze().tolist(),distances.squeeze().tolist())),key=lambda x: x[1])[:0:-1]
                 recommend_frame = []
                 for val in rec_movie_indices:
                     movie_idx = final_dataset.iloc[val[0]]['movieId']
                     idx = movies[movies['movieId'] == movie_idx].index
                     recommend_frame.append({'Title':movies.iloc[idx]['title'].values[0],'Distance':val[1]})
                 df = pd.DataFrame(recommend_frame,index=range(1,n_movies_to_reccomend+1))
                 return df
             else:
                 return "No movies found. Please check your input"
```

18. **Now let’s get our movie recommendation!**

In [26]:
```
get_movie_recommendation('Iron Man')
```

Out[26]:

	Title	Distance
1	Up (2009)	0.368857
2	Guardians of the Galaxy (2014)	0.368758
3	Watchmen (2009)	0.368558
4	Star Trek (2009)	0.366029
5	Batman Begins (2005)	0.362759
6	Avatar (2009)	0.310893
7	Iron Man 2 (2010)	0.307492
8	WALL·E (2008)	0.298138
9	Dark Knight, The (2008)	0.285835
10	Avengers, The (2012)	0.285319

19. Let’s try giving movies not from dataset.

In [27]:
```
get_movie_recommendation('frozen')
```

Out[27]:
```
'No movies found. Please check your input'
```

Results and Analysis

The recommendation system accurately suggests movies that align with the input query's theme. Example results demonstrate the system's ability to leverage user preferences to identify relevant recommendations. The sparsity of the data (73%) highlights the effectiveness of k-NN in handling such datasets.

Conclusion

This project successfully implemented a collaborative filtering-based movie recommendation system. The methodology and results underscore the practical applications of machine learning in improving user experience. Future enhancements could include hybrid approaches combining content-based and collaborative filtering.

References

1. Python Libraries: Pandas, NumPy, Matplotlib, Scikit-learn.
2. Dataset Sources: Kaggle.
3. Algorithm: k-Nearest Neighbors with Cosine Similarity.
