Movie Recommender Systems

Vedant Narendra Joshi
ITCS-5156 Fall 2022 – Lee

Primary Paper [1]

Research Paper Title: Movie Recommendation System Using Collaborative Filtering

Authors: Meenu Gupta, Aditya Thakkar, Aashish, Vishal Gupta, Dhruv Pratap Singh Rathore

Year Published: January 2021

Conference Name: Proceedings of the International Conference on Electronics and Sustainable Communication Systems (ICESC 2020)

1 Introduction

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books, and videos, to friends and stories on social media platforms, to products on e-commerce websites, to search results returned on Google. Sometimes, the recommender systems can make improvements based on the activities of many people. For example, if Amazon observes that many customers who buy the latest Apple MacBook also buy a USB-C-to USB Adapter, they can recommend the Adapter to a new user who has just added a MacBook to his cart. Due to the advances in recommender systems, users constantly expect good recommendations.

Three main approaches are used for our recommender systems. One is Demographic Filtering i.e. They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. Since each user is different, this approach is too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. Second is content-based filtering, where we try to profile the user's interests using information collected, and recommend items based on that profile. The other is collaborative filtering, where we try to group similar users together and use information about the group to make recommendations to the user.

1.1 Problem Statement

Due to the overwhelming number of options available on the Internet, it is necessary to filter, organize, and effectively distribute relevant information in order to address the issue of excess information that many internet users have been experiencing.

Recommender systems try to mitigate this problem of having huge options by filtering out the right content for them.

Like this, when we talk of entertainment, we think about movies. The Internet has tens of thousands of films from various eras. Users sometimes struggle to decide what to watch, thus it is advantageous to develop a movie recommendation system based on user ratings, genres, preferences, and other factors.

In this assignment, I will test several different algorithms on the MovieLens Dataset and analyze which one provides the most accurate recommendation.

1.2 Motivation & Challenges

The Motivation:

- We frequently favor items that are comparable to other things we enjoy. We also prefer to like items that people who are similar to us like. These patterns can be utilized to create predictions about future developments. These advances can be managed with the use of **recommendation systems.**

The Challenges:

- **Scalability** : In many of the environments in which these systems make recommendations, there are millions of users and products. Thus, large number of computation power is often necessary to calculate the recommendations.

- **Sparsity** : The number of movies watched on major OTT platforms is extremely large. The most active users will have rated a small subset of the overall database. Thus, even the most popular

- **Cold Start** : These systems often require a large amount of existing data on a user in order to make accurate recommendations

2 Related Works

2.1 Related Work [2]

- **Research Paper Title:** Movie Recommendation System Using Collaborative Filtering
- **Authors:** Ching-Seh (Mike) Wu, Deepti Garg, Unnathi Bhandary
- **Year Published:** November 2018
- **Conference Name:** 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)

Summary of the Approach:

Recommendation system is of 2 types namely, Collaborative Filtering and Content Based Filtering. In this research paper [2], the authors have used a Collaborative filtering Technique for their Recommendation system. Similar to everything, Data pre-processing and classification needs to be performed by any system. For this purpose, the authors have designed their data preparation and data analysis using Apache Mahout.
The following is how this document is structured: First, a quick overview of some recent important research in the field of recommender systems will be presented. Second, authors present their knowledge of the collaborative filtering process. Third, the technique to data preparation and analysis using Mahout will be discussed. Finally, a qualitative assessment of the techniques employed will be offered.

Initially, the authors perform a thorough Data Pre-processing and Data Cleaning steps. Since they are using the Apache Mahout Platform, no other Machine Learning Algorithm was defined as such. After the Data Cleaning and pre-processing steps, with the help of Pearson coefficient formula, the authors match the ratings of the users with one another and store the resulting output in the Hadoops Distributed file system. The following is the Image attached showing the similarity ratings calculated.

Fig. 1 Similarity Ratings

The project is closely related to mine since it uses the Collaborative Filtering Technique. Apart from that, it uses the Pearson coefficient formula for similarity ratings. My project uses the Cosine Similarity rating.
The only con which I find in this project is that it uses Apache Mahout. The system has not been tested much and hence the results should be verified again.

2.2 Related Work [3]

- **Research Paper Title:** Movie Recommender System using Single Value Decomposition and K-means Clustering
- **Authors:** Mayur Rahul, Vinod Kumar, Vikash Yadav, Rishabh Jain
- **Year Published:** January 2021
- **Conference Name:** IOP Conference Series: Materials Science and Engineering

Summary of the Approach:

In this study, the authors used two publicly available movie datasets, MovieLens and Flixter, to predict movie ratings using single value decomposition (SVD) for dimension reduction and k-means clustering. The following are the key outcomes of this work:

(1) The authors provide a novel movie recommender system based on SVD and k-means clustering.

(2) Metrics like as standard deviation (SD), root mean square error (RMSE), mean absolute error (MAE), t-value, dunn matrix, average similarity, and processing time are used to evaluate the model's outcomes.

(3) The findings are based on two publicly available datasets, MovieLens and Flixter.

Using two publicly available datasets, the system employs the SVD for dimension reduction and classification based on the k-means clustering algorithm. The most prevalent method among researchers is k-means clustering, and SVD is consistent in removing unnecessary data from the population. The combination of these strategies yielded positive outcomes.

The following is the comparison attached after testing the dataset on the above metrics.

	Flexiter	MovieLens
SD	0.13743	0.11453
MAE	0.73372	0.62896
RMSE	0.94876	0.92934
T-value	2.66674	3.74562
Dunn Index (4 Clusters)	0.34873	0.31945
Average Similarity (4 Clusters)	0.96	0.95
Computational Time in sec	71.89	57.43

Following Graph shows the results in terms of a line graph.

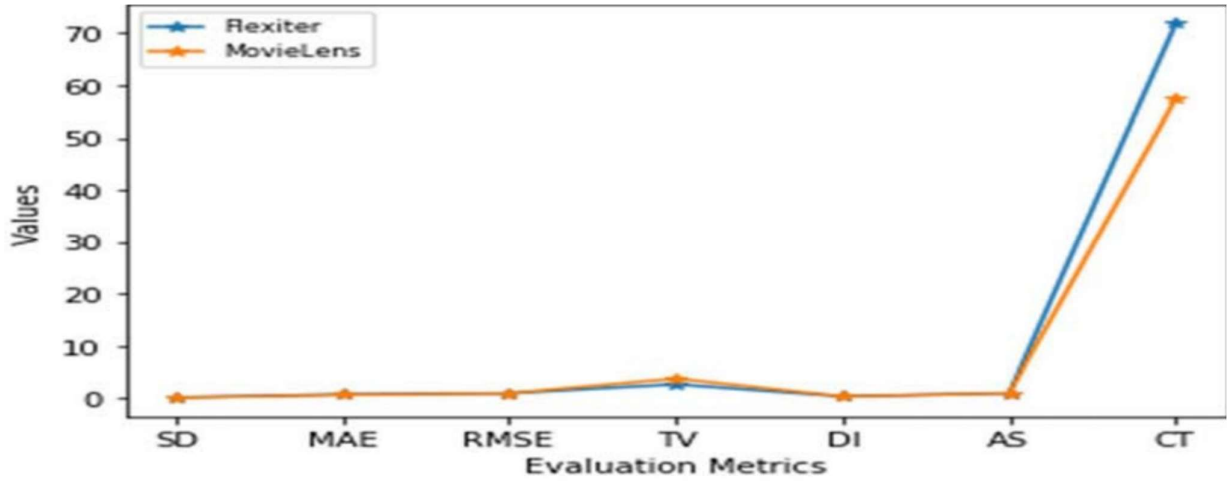

Fig.2 Shows the Line graphs comparing metrics on two Datasets

This document was encouraging as it helps me conclude to choose the MovieLens Data rather than the Flexiter Dataset. The only difference is that it uses KMeans Clustering Algorithms to make groups whereas the project which I worked on used the KNN Algorithm.

3 Method

Following are the Methods used the Project:

1. KNN (K Nearest Neighbors)

- A K-Nearest-Neighbor algorithm is a data categorization technique that looks at the data points surrounding it to decide which group a data point belongs to.
- When an algorithm examines one point on a grid to determine whether it belongs to group A or B, it examines the states of the points nearby. The range is arbitrary, but the goal is to obtain a sample of the data. If the bulk of the points are in group A, the data point in question is likely to be in group A rather than B, and vice versa.
- Because it does not create a model of the data set beforehand, the K-Nearest-Neighbor algorithm is an example of a "lazy learner." It only performs calculations when prompted to poll the data point's neighbors. This makes K-NN for data mining relatively simple to implement.

  Following is the diagrammatic representation of the KNN algorithm.

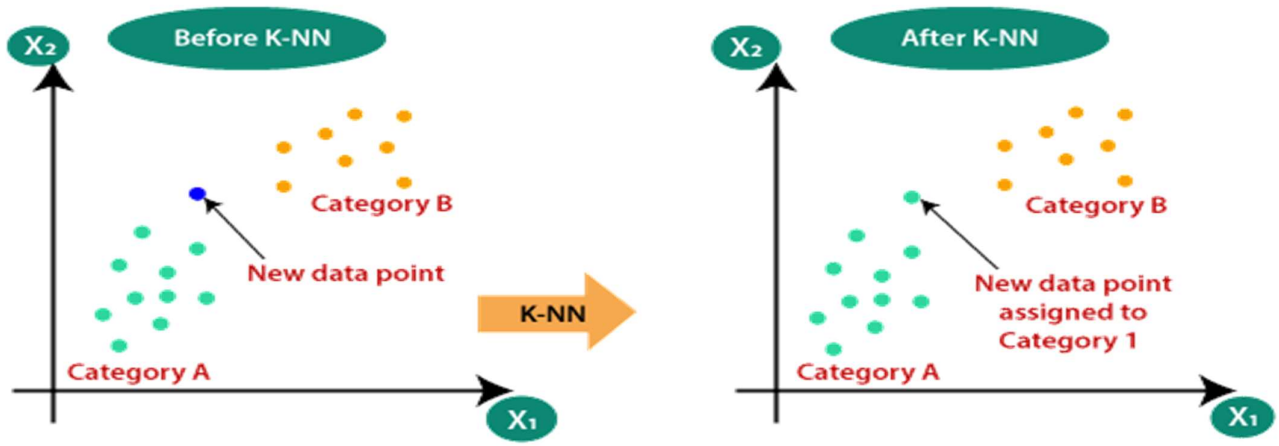

Fig. 3 Diagrammatic Representation of KNN.

2. Cosine Similarity

- It computes the distance between the target movie and the movies in the dataset. It calculates the cosine angle between two vectors in multi-dimensional space and analyzes the similarity between two texts regardless of how varied they are in size.
- A. B = product (dot) of the vector's 'A' and 'B'.
  ||A|| and ||B|| = length of the two vectors 'A' and 'B.'
  ||A|| * ||B|| = cross product of the two vectors 'A' and 'B.'

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Fig 4. Formula of Cosine Similarity

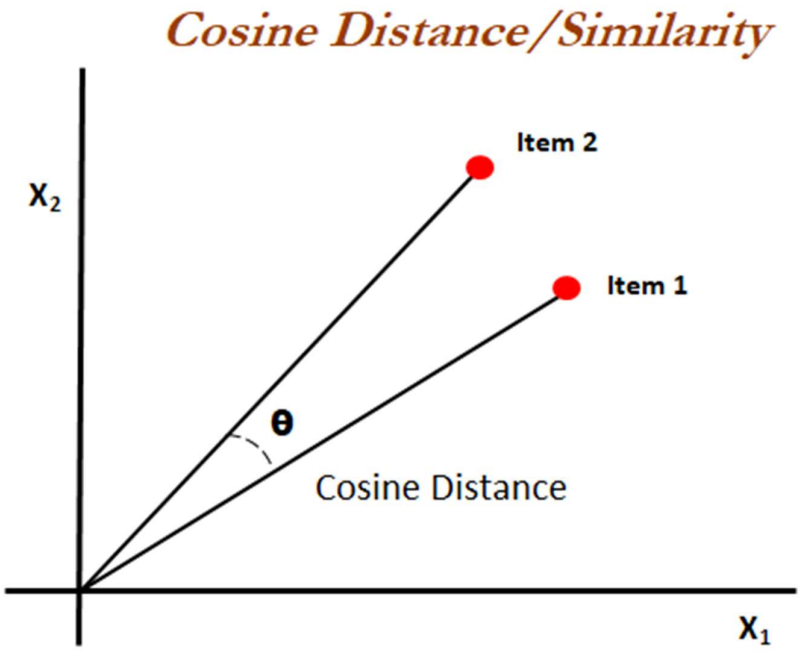

Fig 5. Diagrammatic Representation of Cosine Similarity/Distance

3.1 Architecture

The Following Flowchart shows the Proposed Workflow of our project.

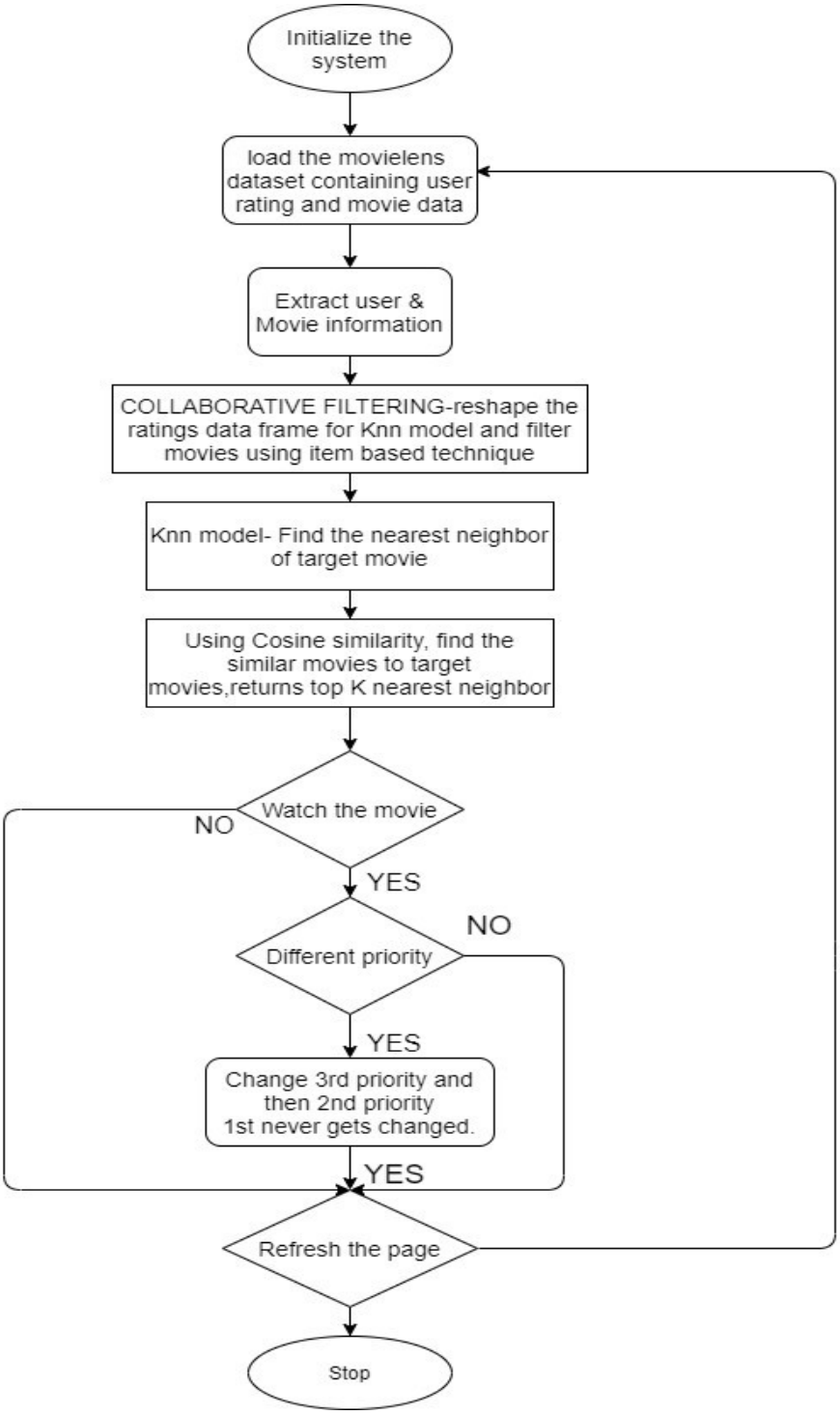

Fig 6. Proposed Workflow of the Project

4 Experimental Setup

1. **Dataset [4]**
   Dataset Name: MovieLens 100K Dataset
   Dataset Description:
   The dataset Consists of following features:-
   1) 100,000 ratings (1-5) from 943 users on 1682 movies.
   2) Each user has rated at least 20 movies.
   3) Simple demographic info for the users (age, gender, occupation, zip)

   Dataset Link: https://grouplens.org/datasets/movielens/100k/

2. **Data Pre-processing**
   Since the Dataset is not readily merged and it is not properly ordered as well, we must do so by following some data pre-processing techniques. With the help of pandas Library, we merge the dataset so that it can be properly used for the further process. Following Image shows how the data was separated using split function and pandas library.

```
In [119…
d = 'movie id | movie title | release date | video release date | IMDb URL | unknown | Action | Adventure | Animation | Children | Comedy | Crime |
column_names2 = d.split(' | ')
column_names2
```

```
Out[119…
['movie id',
 'movie title',
 'release date',
 'video release date',
 'IMDb URL',
 'unknown',
 'Action',
 'Adventure',
 'Animation',
 'Children',
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Fantasy',
 'Film-Noir',
 'Horror',
 'Musical',
 'Mystery',
 'Romance',
 'Sci-Fi',
 'Thriller',
 'War',
 'Western']
```

```
In [120…
items_dataset = pd.read_csv('ml-100k/u.item', sep='|',header=None,names=column_names2,encoding='latin-1')
items_dataset
```

Out[120…

	movie id	movie title	release date	video release date	IMDb URL	unknown	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	1	0	0	0	0	0
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	0	0	0	0	0	0
		Four	01-													

Fig 7. Data Separation Process

3. Exploratory Data Analysis

As we know, Exploratory data analysis is an important step in any deep or Machine Learning algorithm to know more about our data. For this project, we need to perform EDA thoroughly since we have plenty of data and we need to filter and organize the right one.

Following are the images which show the EDA performed in the project.

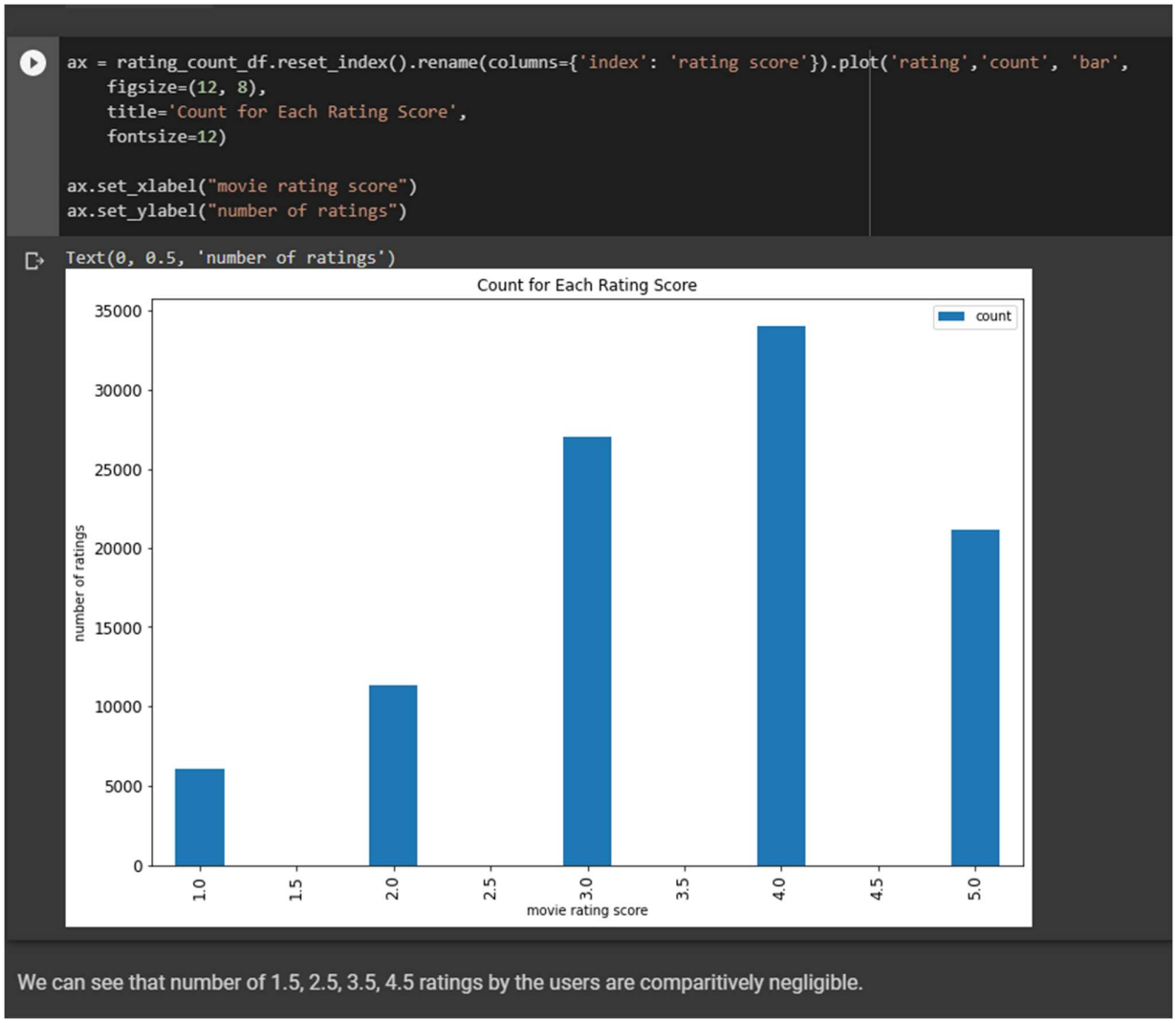

Fig 8. Initial Data Plotting

From the above data plotted, it was observed the number of 1.5,2.5,3.5 and 4.5 ratings were negligible as compared to other ratings. Proceeding with these ratings will make the model work abnormally and not recommend accurate results. For this purpose, we append the ratings into the dataset and create a new dataset.

Following is code used for it.

```
[17] total_count = num_items * num_users
     zero_count = total_count-refined_dataset.shape[0]
     zero_count
```

```
1469459
```

```
# append counts of zero rating to df_ratings_cnt
rating_count_df = rating_count_df.append(
    pd.DataFrame({'count': zero_count}, index=[0.0]),
    verify_integrity=True,
).sort_index()
rating_count_df
```

	count
0.0	1469459
1.0	6083
1.5	3
2.0	11334
2.5	6
3.0	27060
3.5	19
4.0	34042
4.5	16
5.0	21130

Fig 9. Appended Dataset

Again, after this we can observe that the number of ratings for 0 are way more than the other ratings. This creates a sparseness in our dataset which is not very good. To mitigate this problem, we log transform the data.

Following image shows the bar graph after log transforming the dataset.

```
ax = rating_count_df.plot('rating score', 'log_count', 'bar', figsize=(12, 8),
    title='Count for Each Rating Score (in Log Scale)',
    logy=True,
    fontsize=12,)

ax.set_xlabel("movie rating score")
ax.set_ylabel("number of ratings")
```

```
Text(0, 0.5, 'number of ratings')
```

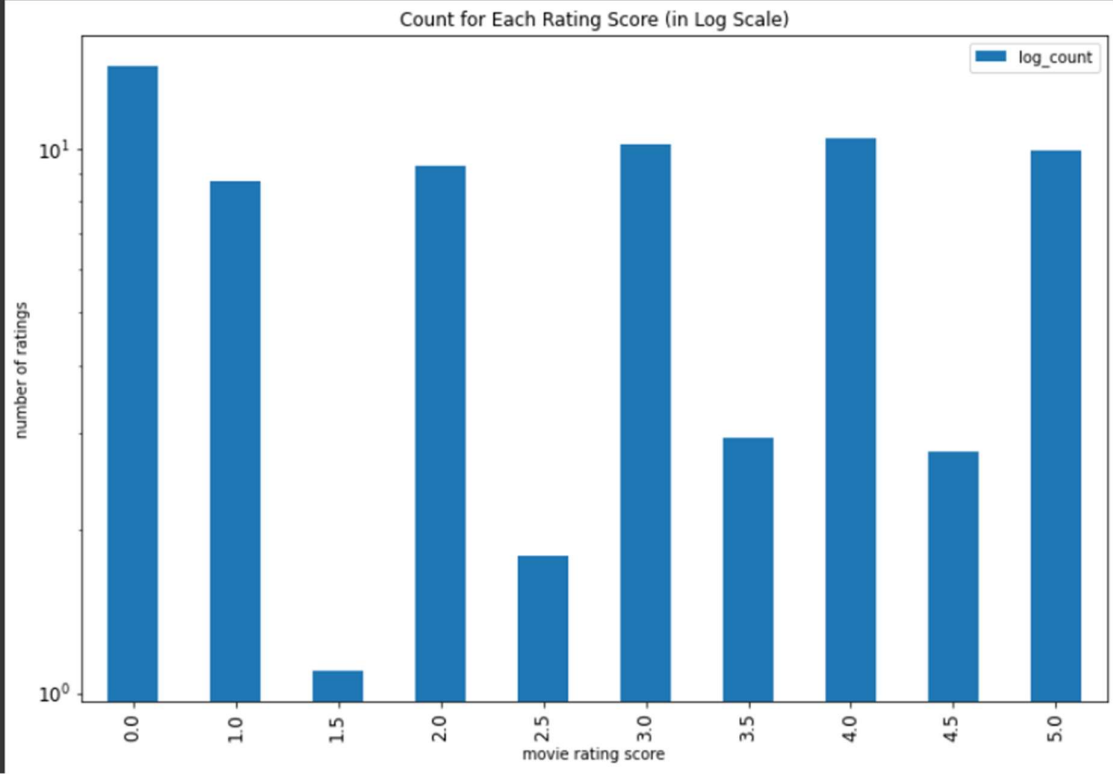

Fig 10. Graph after Log Transforming Data

After all the changes have been made, we plot a final line graph to see the number or ratings against the movie id.

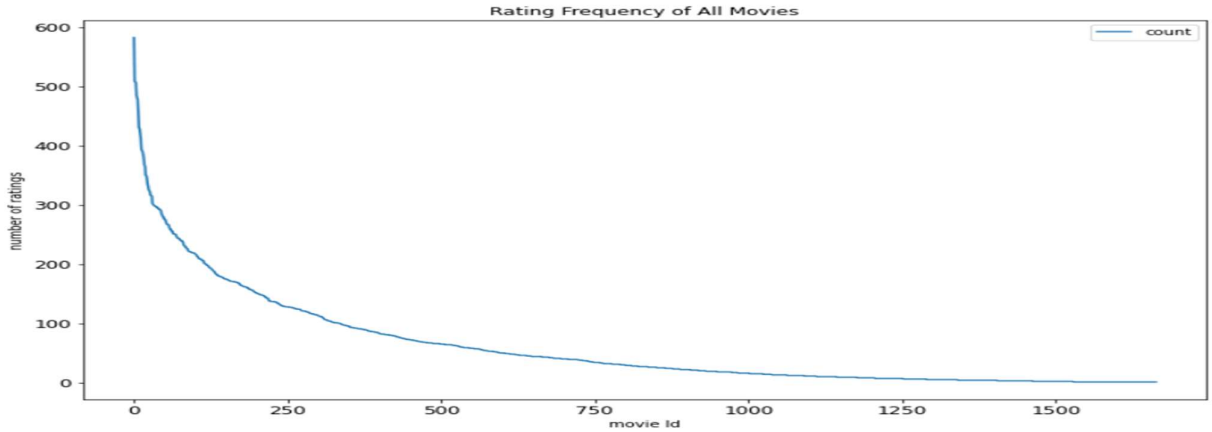

Fig 11. Movie ratings vs Movie ID

4. Model Creation

Reshaping the Dataframe

We must convert (reshape) the data so that each row of the data frame represents a distinct movie and each column represents a different user. So, we want the data to be [movies, users] array if the subject is a movie and related movies must be located, and [users, movies] array otherwise.

We will pivot the data frame to the wide format, with movies as rows and users as columns, to reshape it. We can expect many missing values because we know that not all consumers view all the movies. Because we will be performing linear algebra procedures, we will need to fill those missing observations with 0s (calculating distances between vectors).
Finally, for the most efficient calculations, we turn the data frame values into a SciPy sparse matrix.

After that, the data frame is loaded into a KNN model. Changing the model so that each user has an n-dimensional rating space, where n is the total number of movies.

We will train the KNN model to locate closely matching comparable users to the user we provide as input, and then we will recommend the top movies that would be of interest to the input user.

```
# pivot and create movie-user matrix
user_to_movie_df = refined_dataset.pivot(
    index='user id',
     columns='movie title',
      values='rating').fillna(0)

user_to_movie_df.head()
```

movie title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	8 1/2 (1963)	8 Heads in a Duffel Bag (1997)	8 Seconds (1994)	A Chef in Love (1996)	Above the Rim (1994)	Absolute Power (1997)	Abyss, The (1989)	Ace Ventura: Pet Detective (1994)	A Ventur Wh Natu Ca (199
user id																			
1	0.0	0.0	2.0	5.0	0.0	0.0	3.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	3.0	0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0
3	0.0	0.0	0.0	0.0	2.0	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
4	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
5	0.0	0.0	2.0	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	0.0	0.0	1

5 rows × 1664 columns

Fig 10. Reshaping

```
# transform matrix to scipy sparse matrix
user_to_movie_sparse_df = csr_matrix(user_to_movie_df.values)
user_to_movie_sparse_df
```

```
<943x1664 sparse matrix of type '<class 'numpy.float64'>'
        with 99693 stored elements in Compressed Sparse Row format>
```

Fitting K-Nearest Neighbours model to the scipy sparse matrix:

```
knn_model = NearestNeighbors(metric='cosine', algorithm='brute')
knn_model.fit(user_to_movie_sparse_df)
```

```
NearestNeighbors(algorithm='brute', leaf_size=30, metric='cosine',
                 metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                 radius=1.0)
```

```
## function to find top n similar users of the given input user
def get_similar_users(user, n = 5):
  ## input to this function is the user and number of top similar users you want.

  knn_input = np.asarray([user_to_movie_df.values[user-1]])  #.reshape(1,-1)
  # knn_input = user_to_movie_df.iloc[0,:].values.reshape(1,-1)
  distances, indices = knn_model.kneighbors(knn_input, n_neighbors=n+1)

  print("Top",n,"users who are very much similar to the User-",user, "are: ")
  print(" ")
  for i in range(1,len(distances[0])):
    print(i,". User:", indices[0][i]+1, "separated by distance of",distances[0][i])
  return indices.flatten()[1:] + 1, distances.flatten()[1:]
```

Fig 11. SciPy Matrix and KNN model Creation

5. Results

After completing all the necessary modifications, it was time to make the suggestions by invoking the function. The recommend movies function accepts the number of movies the user wants to be recommended. Following Image shows the output after running the function.

```
print("Movies recommended based on similar users are: ")
recommend_movies(10)
```

```
Movies recommended based on similar users are:
['Star Wars (1977)',
 'Terminator, The (1984)',
 "Schindler's List (1993)",
 'Fugitive, The (1993)',
 'Forrest Gump (1994)',
 'Princess Bride, The (1987)',
 'Empire Strikes Back, The (1980)',
 'Pulp Fiction (1994)',
 'Die Hard (1988)',
 'Monty Python and the Holy Grail (1974)']
```

Fig 11. Results

5 Conclusion

From the above activity, the movie Recommender system was systematically implemented.

To a large extent, the system's results were correct. An intriguing finding is that the above KNN model for movies recommends movies created in years quite comparable to the input movie. However, the cosine distance between all of those recommendations is shown to be fairly minimal. This could be because our movie-user matrix contains too many zero values. When we have too many zero values in our data, data sparsity becomes a serious problem for the KNN model, and the distance in the KNN model begins to fall apart.

Overall, it was a fantastic learning experience, from selecting the Research Papers to contributing to the project.

Recommender systems based on Deep Learning, as well as a hybrid technique combining Collaborative Filtering and Content-Based Filtering, may be employed in the future.

6 My Contributions

1. The project was run without clearing the multiple entries in the dataset.
   I cleaned the data by removing the multiple entries and created a refined Dataset.
   This gave a clearer picture in the analysis of the data.

Following is the Image Attached of the above change.

```
merged_dataset[(merged_dataset['movie title'] == 'Chasing Amy (1997)') & (merged_dataset['user id'] == 894)]
```

	user id	movie id	rating	timestamp	movie title
4800	894	246	4	882404137	Chasing Amy (1997)
22340	894	268	3	879896041	Chasing Amy (1997)

```
refined_dataset = merged_dataset.groupby(by=['user id','movie title'], as_index=False).agg({"rating":"mean"})

refined_dataset.head()
```

	user id	movie title	rating
0	1	101 Dalmatians (1996)	2.0
1	1	12 Angry Men (1957)	5.0
2	1	20,000 Leagues Under the Sea (1954)	3.0
3	1	2001: A Space Odyssey (1968)	4.0
4	1	Abyss, The (1989)	3.0

Fig 12. Refined Dataset

2. It has been discovered that the developed recommendation system can be made more efficient because it has few downsides.

   Drawbacks:

   1. However, this recommendation method has a downside in that it also recommends movies that the given input User has already viewed.

   2. There is also the option of recommending movies that have not been seen by any of the similar users.

   The disadvantages listed above are addressed, and a new recommender system with modifications is created.
   The function below is written to remove movies that have already been seen by the current user but have not been seen by any of the comparable users.

```
def filtered_movie_recommendations(n):

  first_zero_index = np.where(mean_rating_list == 0)[0][-1]
  sortd_index = np.argsort(mean_rating_list)[::-1]
  sortd_index = sortd_index[:list(sortd_index).index(first_zero_index)]
  n = min(len(sortd_index),n)
  movies_watched = list(refined_dataset[refined_dataset['user id'] == user_id]['movie title'])
  filtered_movie_list = list(movies_list[sortd_index])
  count = 0
  final_movie_list = []
  for i in filtered_movie_list:
    if i not in movies_watched:
      count+=1
      final_movie_list.append(i)
    if count == n:
      break
  if count == 0:
    print("There are no movies left which are not seen by the input users and seen by similar users. May be increasing the number of similar users
  else:
    pprint(final_movie_list)
```

```
filtered_movie_recommendations(10)
```

```
['Star Wars (1977)',
 "Schindler's List (1993)",
 'Princess Bride, The (1987)',
 'Empire Strikes Back, The (1980)',
 'Return of the Jedi (1983)',
 'Fargo (1996)',
 'Dances with Wolves (1990)',
 'Toy Story (1995)',
 'Braveheart (1995)',
 'Star Trek: First Contact (1996)']
```

Coding up all of the above individual cells into a function.

Giving Input as **User id, Number of similar Users to be considered, Number of top movie we want to recommend**

Fig 13. Modified Function

7 References

[1] M. Gupta, A. Thakkar, Aashish, V. Gupta and D. P. S. Rathore, "Movie Recommender System Using Collaborative Filtering," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 415-420, doi: 10.1109/ICESC48915.2020.9155879.

[2] C. S. M. Wu, D. Garg, and U. Bhandary, “Movie Recommendation System Using Collaborative Filtering,” In 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), pp. 11-15, IEEE, 2018 Nov.

[3] Mayur Rahul et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1022 012100

[4] https://grouplens.org/datasets/movielens/100k/