SILICON INSTITUTE OF TECHNOLOGY

BHUBANESWAR, ODISHA-751024

Silicon

...beyond teaching

Machine Learning Project

On

MOVIE RECOMMENDATION SYSTEM

I Prasanti

Roll-16,CEN-A1

20BCEA56

CONTENT

DECLARATION

I, I PRASANTI, declare that the work presented in this machine learning report titled MOVIE RECOMMENDATION, is entirely my own effort and has not been submitted, in part or in whole, for any other degree, diploma, or qualification. All sources of information used in this report have been appropriately acknowledgedand cited.

I confirm that the data used in this report is authentic and was obtained through ethical means. The data cleaning and pre-processing steps were performed with utmost care, and the model development process was carried out with due diligence. The machine learning algorithms and techniques used in this report are appropriately selected and implemented, and the results presented in this report are accurate and unbiased.

Finally, I acknowledge and accept that any external resources used in the creation of this report have been appropriately cited, and I have not plagiarized or misrepresented any work.

I PRASANTI-20BCEA56

05/05/2023

CERTIFICATE OF APPROVAL

This is to certify that I Prasanti, a student at Silicon Institute of Technology, has completed the machine learning report titled Movie Recommendation under the supervision of Prof. Nayan Ranjan Paul.

The report is a comprehensive analysis of movie dataset and applying it to a Machine Learning Model for recommendation, and it demonstrates the student's proficiency in the field of machine learning.

The report was completed to the satisfaction of the examining committee and is worthy of consideration for any future academic orprofessional pursuits.

We congratulate I Prasanti on the successful completion of this project and wish them all the best in their future endeavours.

SILICON INSTITUTE OF
TECHNOLOGY

05/05/2023

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who have helped me throughout the completion of this machine learning report.

First and foremost, I would like to thank my supervisor Nayan Sir for his guidance, support, and valuable insights throughout the project. His encouragement and feedback were crucial in shaping the direction of this report.

I would also like to thank SILICON INSTITUTE OF TECHNOLOGY for providing me with the resources and facilities necessary to carry out this project.

I am grateful to my team members: Rojalin Parida, Anwesha Rath for their support and assistance in collecting and processing the data used in this report. Their contributions were vital in ensuring the accuracy and reliabilityof the results.

Finally, I would like to thank my family and friends for their unwavering support and encouragement throughout this project. Their constant motivation and understanding helped me stay focused and determined.

Thank you all for your invaluable contributions.

I PRASANTI

INTRODUCTION

Recommendation systems are becoming increasingly important in today's hectic world. People are always in the lookout for products/services that are best suited for them. Therefore, the recommendation systems are important as they help them make the right choices, without having to expend their cognitive resources.

Entertainment is very crucial in today's world for almost everybody. Everyone wants to watch a good movie without getting bored and without wasting time and money. So this is when a recommendation system comes to work.

In this project, we will understand the basics of Recommendation Systems and learn how to build a Movie Recommendation System by implementing the K-Nearest Neighbors algorithm. We will also predict the rating of the given movie based on its neighbors and compare it with the actual rating.

We are committed to developing a transparent and accessible project, and we will share our progress and results with the wider community. We believe that our project can make a contribution to the field of movie recommendation and machine learning.

DATASETS

The dataset used in this context is downloaded from MovieLens datasets and saved to the working directory.

To read the contents of the file into a Python program, used the Pandas library's "read_csv()" function. It has information about 9000 unique movies. It has two csv files:

- movies.csv : this file contain movieId,title of movies and genre of movies.
- rating.csv : this file contain ratings given by userId to MovieId and rating given to movieId.

With the data now loaded into their program, can analyse and process it using Python's various data analysis libraries and tools. It is important to note that the quality and reliability of the results obtained from the analysis will depend heavily on the quality and accuracy of the data contained in the dataset. Therefore, it is crucial to carefully select and pre-process datasets before using them for analysis.

Importing the dataset:

- movies.csv:

```
movies = pd.read_csv('movies.csv')
movies
```

	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
...	...	...	...
9737	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy
9738	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy

9742 rows × 3 columns

- ratings.csv:

```
ratings = pd.read_csv('ratings.csv')
ratings
```

	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 × 4 columns

METHODOLOGY

A recommendation system collect data about the user's preferences either implicitly or explicitly on different items like movies. An implicit acquisition in the development of movie recommendation system uses the user's behaviour while watching the movies. On the other hand, a explicit acquisition in the development of movie recommendation system uses the user's previous ratings or history. The other supporting technique that are used in the development of recommendation system is clustering. Clustering is a process to group a set of objects in such a way that objects in the same clusters are more similar to each other than to those in other clusters.

KMeans Clustering along with K-Nearest Neighbour is implemented on the movie lens dataset to obtain the best-optimized result. In existing technique, the data is scattered which results in a high number of clusters while in the proposed technique data is gathered and results in a low number of clusters. The process of recommendation of a movie is optimized in the proposed scheme.

The proposed recommender system predicts the user's preference of a movie based on different parameters. The recommender system works on the concept that people are having common preference or choice. These users will influence on each other's opinions. This process optimizes the process and having lower RMSE.

Collaborative filtering systems analyse the user's behaviour and preferences and predict what they would like based on similarity with other users. There are two kinds.
of collaborative filtering systems; user-based recommender and item-based recommender.

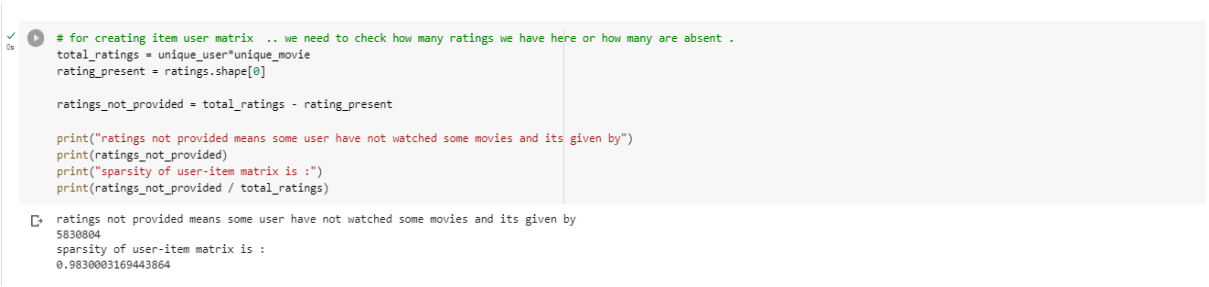
- Use-based filtering: User-based preferences are very common in the field of designing personalized systems. This approach is based on the user's likings. The process starts with users giving ratings (1-5) to some movies. These ratings can be implicit or explicit. Explicit ratings are when the user explicitly rates the item on some scale or indicates a thumbs-up/thumbs-down to the item. Often explicit ratings are hard to gather. In these scenarios, we gather implicit ratings based on their behaviour. Note that there are no clear rules in determining implicit ratings. Next, for each user,

we first find some defined number of nearest neighbours. We calculate correlation between users' ratings using Pearson Correlation algorithm.

- Item-based filtering: Unlike the user-based filtering method, itembased focuses on the similarity between the item's users like instead of the users themselves. The most similar items are computed ahead of time. Then for recommendation, the items that are most like the target item are recommended to the user. Here we have taken anime genre and type as the items.

One of the advantages of KNN is that it is a non-parametric algorithm, meaning that it does not make any assumptions about the distribution of the data. This makes it useful for handling complex, non-linear relationships between variables. However, KNN can be computationally expensive, particularly as the size of the dataset increases.

- <u>Sparsity of user-item matrix:</u>



```
# for creating item user matrix  .. we need to check how many ratings we have here or how many are absent .
total_ratings = unique_user*unique_movie
rating_present = ratings.shape[0]

ratings_not_provided = total_ratings - rating_present

print("ratings not provided means some user have not watched some movies and its given by")
print(ratings_not_provided)
print("sparsity of user-item matrix is :")
print(ratings_not_provided / total_ratings)
```

```
ratings not provided means some user have not watched some movies and its given by
5830804
sparsity of user-item matrix is :
0.9830003169443864
```

- <u>Rating count which tells which rating is more frequent:</u>

```
# 1)plot ratings count which gives information about which rating(on scale of 0 to 5) is more frequent

rating_cnt = pd.DataFrame(ratings.groupby('rating').size(),columns=['count'])
rating_cnt
# this rating cnt doesnt contain count of rating 0
# append rating_cnt

rating_cnt = rating_cnt.append(pd.DataFrame({'count':ratings_not_provided},index = [0])).sort_index()
rating_cnt

#since the count of rating 0 is too large in comparison to others rating ...use log value
rating_cnt['log_count'] = np.log(rating_cnt['count'])
rating_cnt
```

```
<ipython-input-8-9fb35d41f813>:8: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  rating_cnt = rating_cnt.append(pd.DataFrame({'count':ratings_not_provided},index = [0])).sort_index()
```

	count	log_count
0.0	5830804	15.578665
0.5	1370	7.222566
1.0	2811	7.941296
1.5	1791	7.490529
2.0	7551	8.929435
2.5	5550	8.621553
3.0	20047	9.905835
3.5	13136	9.483112
4.0	26818	10.196829
4.5	8551	9.053804
5.0	13211	9.488805

- Frequency of each movie

```
# 2) plot rating frequency of each movie(how many time a movie has been rated)

movie_freq = pd.DataFrame(ratings.groupby('movieId').size(),columns=['count'])
movie_freq.head()
```

movieId	count
1	215
2	110
3	52
4	7
5	49

- Implementing fuzzywuzzy: Now when a movie name is given as input we need to find that any such movie present in our dataset or not.If it is not present then we cant recommend anything . so for string matching we are going to use fuzzy matching, based on result of fuzzy matching , a list of recommedation will be generated. lets create a function which take parameters (input_string , mapper=movie_to_index) . This fucntion will return movie_id of movie title which is best match with input string . It also prints the all matches.

```
# fuzzy_movie_name_matching
from fuzzywuzzy import fuzz

def fuzzy_movie_name_matching (input_str,mapper,print_matches):
    # match_movie is list of tuple of 3 values(movie_name,index,fuzz_ratio)
    match_movie = []
    for movie,ind in mapper.items():
        current_ratio = fuzz.ratio(movie.lower(),input_str.lower())
        if(current_ratio>=50):
            match_movie.append((movie,ind,current_ratio))

    # sort the match_movie with respect to ratio

    match_movie = sorted(match_movie,key =lambda x:x[2])[::-1]

    if len(match_movie)==0:
        print("Oops..! no such movie is present here\n")
        return -1
    if print_matches == True:
        print("some matching of input_str are\n")
        for title,ind,ratio in match_movie:
            print(title,ind,'\n')


    return match_movie[0][1]
```

0s completed at 11:33PM

- KNN-Implementation

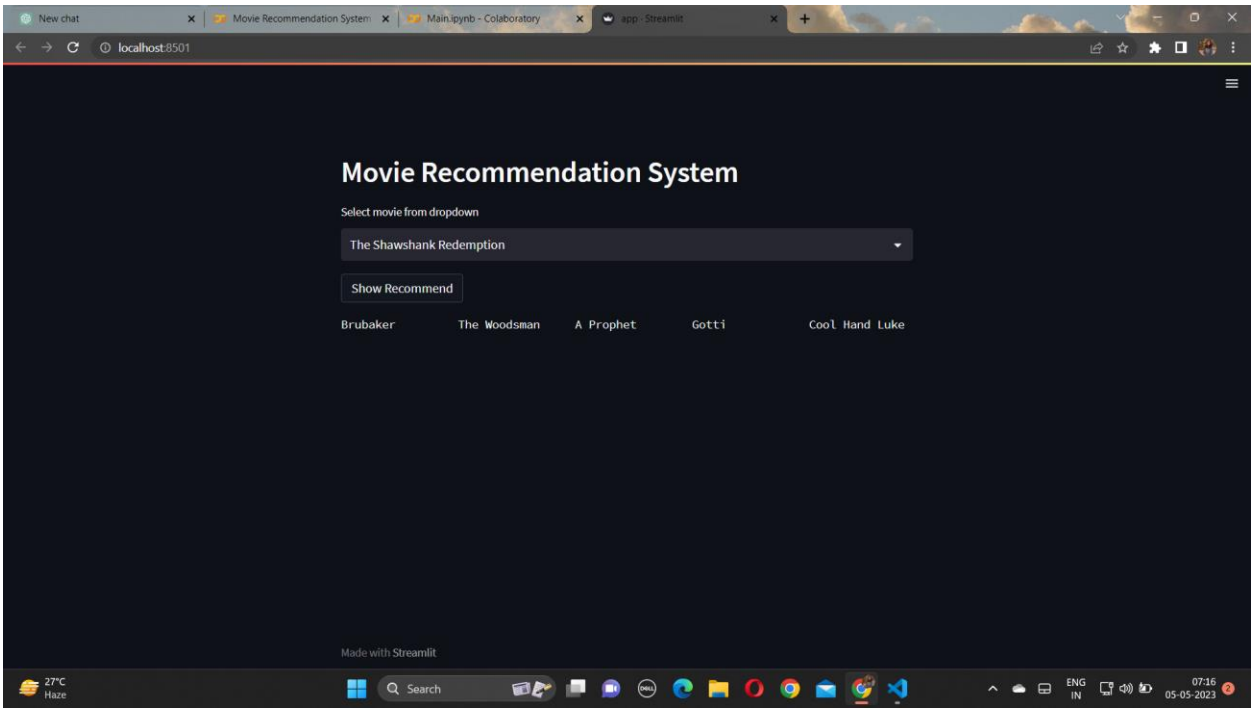
```
# create a function which takes a movie name and make recommedation for it
def make_recommendation(input_str,data,model,mapper,n_recommendation):
    print("system is working....\n")
    model.fit(data)

    index = fuzzy_movie_name_matching (input_str,mapper,print_matches = False)

    if index==-1 :
        print("pls enter a valid movie name\n")
        return

    index_list = model.kneighbors(data[index],n_neighbors=n_recommendation+1,return_distance=False)
    # now we ind of all recommendation
    # build mapper index->title
    index_to_movie={
        ind:movie for movie,ind in mapper.items()
    }

    print("Viewer who watches this movie ",input_str,"also watches following movies.")
    #print(index_list[0][2])
    for i in range(1,index_list.shape[1]):
        print(index_to_movie[index_list[0][i]])



    return
```

- Result analysis:

```
make_recommendation('Father of the Bride Part II ',item_user_mat_sparse,recommendation_model,movie_to_index,10)
```

```
Viewer who watches this movie  Father of the Bride Part II  also watches following movies.
Sabrina (1995)
Miracle on 34th Street (1994)
Striptease (1996)
Juror, The (1996)
Mr. Holland's Opus (1995)
Sgt. Bilko (1996)
Twister (1996)
Grumpier Old Men (1995)
Tin Cup (1996)
Willy Wonka & the Chocolate Factory (1971)
```

- Deploying the model: Here streamlit is used to deploy the model.

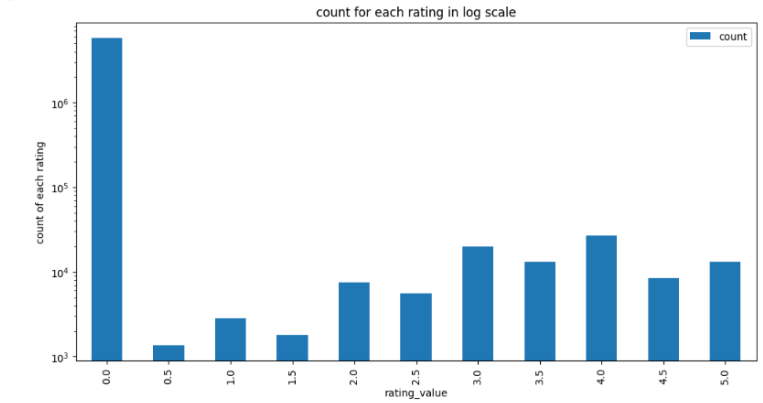


VISUALIZATION

- Bar plot to visualize the ratings

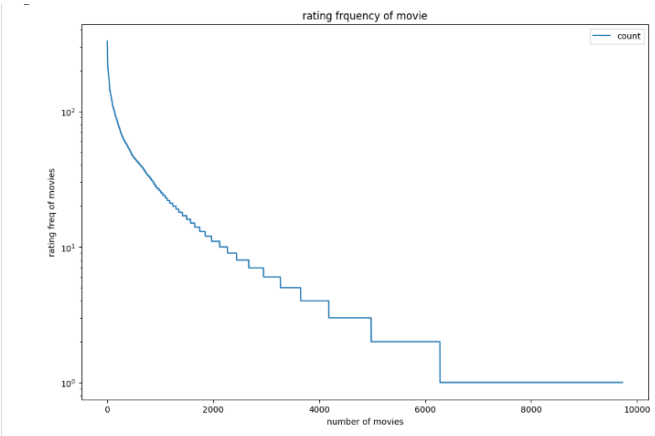
```
# plot a bar plot to visualise the ratings
rating_cnt_for_vis = rating_cnt
ax = rating_cnt_for_vis.reset_index().rename(columns = {'index':'rating_value'}).plot(
    x='rating_value',
    y='count',
    logy = True,
    kind='bar',
    title='count for each rating in log scale',
    figsize=(12,6)
)
ax.set_xlabel('rating_value')
ax.set_ylabel('count of each rating')
print("frequency of rating like 3 and 4 are more in compare to other ratings")
```

```
frequency of rating like 3 and 4 are more in compare to other ratings
```



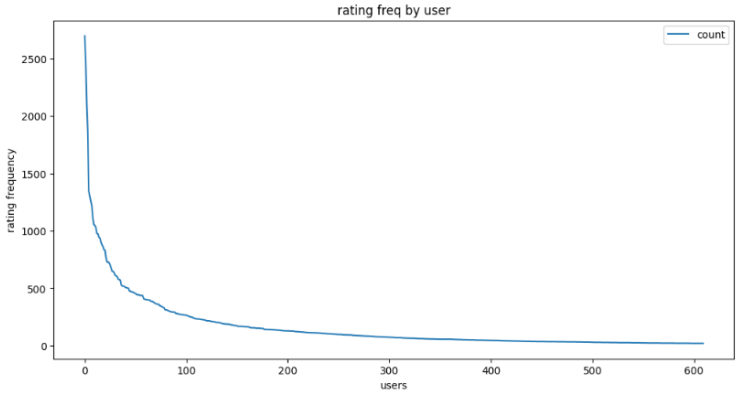
- Graph for movie rating frequency:

```
# plot movie rating freq
movie_freq_copy = movie_freq.sort_values(by='count',ascending=False)
movie_freq_copy=movie_freq_copy.reset_index(drop=True)

ax1  = movie_freq_copy.plot(
    title='rating frquency of movie',
    logy=True,
    figsize=(12,8)
)
ax1.set_xlabel('number of movies')
ax1.set_ylabel('rating freq of movies')
```



- Graph for rating frequency

```
# plot rating freq
ax  = user_cnt_copy.sort_values('count',ascending=False).reset_index(drop=True).plot(
      title='rating freq by user',
      figsize=(12,6),

)
ax.set_xlabel("users")
ax.set_ylabel("rating frequency")
```



CONCLUSION

In conclusion, a movie recommendation system based on KNN algorithm can be a useful tool for users who are looking for personalized movie recommendations based on their past preferences. Through the implementation of the system, it is possible to accurately predict which movie titles a user is likely to enjoy based on their ratings of previously watched movie titles.

However, there are some limitations to the KNN-based recommendation system, such as the need for a large dataset, a large number of users, and the selection of appropriate similarity metrics. Additionally, it may not perform well when the dataset contains users with sparse ratings, meaning they have only rated a few titles.

Future scope for movie recommendation systems includes the incorporation of more advanced techniques such as deep learning and natural language processing, which can consider other factors such as user demographics, textual data, and temporal trends. Additionally, the use of hybrid recommendation systems that combine content-based filtering, collaborative filtering, and other approaches can also improve the performance of the system.

Overall, the potential for movie recommendation systems is vast, and it can greatly enhance the user experience by providing personalized recommendations, which can help users to discover new movie titles.