MOVIE RECOMMENDATION ENGINE

A Project Report

In the partial fulfillment for the award of the degree of

B.Tech

under

Academy of Skill Development

ASD
Skills For Employment

Submitted by

ATHARV MASLEKAR, SHREYASH DESHMUKH and SHIVPRASAD RATHOD

Shri Guru Gobind Singhji Institute of Engineering and Technology

Certificate from the Mentor

This is to certify that **Atharv Maslekar, Shreyash Deshmukh and Shivprasad Rathod** has successfully completed the project titled **Movie Recommendation Engine** under my supervision during the period from February to May which is in partial fulfillment of requirements for the award of the B.Tech and submitted to Department **Electronics and Telecommunication** of **Shri Guru Gobind Singhji Institute of Engineering and Technology.**

_____

Signature of the Mentor

Date:

Acknowledgement

I take this opportunity to express my deep gratitude and sincerest thanks to my project mentor, **Mr. Souvik Ganguly** for giving the most valuable suggestion, helpful guidance and encouragement in the execution of this project work.

I would like to give a special mention to my colleagues. Last but not the least I am grateful to all the faculty members of **Academy of Skill Development** for their support.

Academy Of Skill Development (ASD) an MSME is a non-profit organization registered with Govt. of West Bengal. ASD has affiliations to conduct global certification of Microsoft Technology Associate (MTA), AutoDESK Certified User (ACU), Adobe Certified Associate (ACA), EC-Council Associate Certifications. Mission of ASD is to develop skills of the candidates to meet the dynamics of demand in the job market. We look forward to bridge the gap between the industry and academia. Recent survey has revealed the huge gap between demand and supply of candidates. Today industry is looking for ready-made candidates with right set of skills. Hence employability enhancement training and internship is very important. At ASD we provide the latest and best Skill Development Internships.

Vision and Mission

Vision: 65% of India's population are the driving force of the economy and ASD here has a small role in conducting skill enhancement programs for the youth. This skill development programs will help the youth of the country to get success in their careers and in different national level programs like Digital India, Make in India, etc. Our country needs a huge workforce that can be produced by organisations like us. To be part of a massive scale training target of 500 million skilful candidates in India by 2025, ASD is very much committed to be a part of it. We facilitate the skill development initiatives for the society.

Mission: Mission of ASD is to develop skills of the candidates to meet the dyamics of demand in the job market.

Partners

ACADEMY OF SKILL DEVELOPMENT is partnered with big organisations such as Microsoft Technology Associate, Microsoft Office specialist, Autodesk, Adobe certified credentials, Hewlett Packard Enterprise. It is also associated with Udyog Aadhar: Govt. MSME initiative.

2.ABSTRACT:

The basic concept behind a movie recommendation system is quite simple. In particular, there are two main elements in every recommender system: users and items. The system generates movie predictions for its users, while items are the movies themselves.

The primary goal of movie recommendation systems is to filter and predict only those movies that a corresponding user is most likely to want to watch. The ML algorithms for these recommendation systems use the data about this user from the system's database. This data is used to predict the future behaviour of the user concerned based on the information from the past.

The recommendation system analyses the past preferences of the user concerned, and then it uses this information to try to find similar movies. This information is available in the database (e.g. title, genre, etc.). After that, the system provides movie recommendations for the user. That said, the core element in content-based filtering is only the data of only one user that is used to make predictions.

3.INTRODUCTION:

The largest movie libraries in the world are all digitized and transferred to online streaming services, like Netflix, HBO, or YouTube. Enhanced with AI-powered tools, these platforms can now assist us with probably the most difficult chore of all — picking a movie.

Well, you don't have to worry about that anymore. It's officially showtime for machine learning to demonstrate its capabilities in the world of cinema as known today. Data scientists are all set to explore our behavioural patterns and the ones of the movies to build sophisticated predictive systems for true movie fans.

A movie recommendation system, or a movie recommender system, is an ML-based approach to filtering or predicting the users' film preferences based on their past choices and behaviour. It's an advanced filtration mechanism that predicts the possible movie choices of the concerned user and their preferences towards a domain-specific item, aka movie.

Now we will see the practical implementation of the Apriori Algorithm in this project. The Apriori algorithm uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rule, it determines how strongly or how weakly two objects are connected. It can also be used in the healthcare field to find drug reactions for patients.

4.ALGORITHMS:

Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

Prerequisite – Frequent Item set in Data set (Association Rule Mining)

To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called Apriori property which helps by reducing the search space.

Apriori Property –

All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure.

Apriori assumes that - All subsets of a frequent itemset must be frequent (Apriori property). If an itemset is infrequent, all its supersets will be infrequent.

Confidence – A confidence of 60% means that 60% of the customers, who purchased milk and bread also bought butter.

Confidence(A->B)=Support_count(A∪B)/Support_count(A)

So here, by taking an example of any frequent itemset, we will show the rule generation.

Itemset {I1, I2, I3} //from L3

So rules can be

[I1^I2]=>[I3] //confidence = sup(I1^I2^I3)/sup(I1^I2) = 2/4*100=50%

[I1^I3]=>[I2] //confidence = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%

[I2^I3]=>[I1] //confidence = sup(I1^I2^I3)/sup(I2^I3) = 2/4*100=50%

[I1]=>[I2^I3] //confidence = sup(I1^I2^I3)/sup(I1) = 2/6*100=33%

[I2]=>[I1^I3] //confidence = sup(I1^I2^I3)/sup(I2) = 2/7*100=28%

[I3]=>[I1^I2] //confidence = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%

So if minimum confidence is 50%, then first 3 rules can be considered as strong association rules.

Steps for Apriori Algorithm -
Below are the steps for the apriori algorithm:

Step-1: Determine the support of itemsets in the transactional database, and select the minimum support and confidence.

Step-2: Take all supports in the transaction with higher support value than the minimum or selected support value.

Step-3: Find all the rules of these subsets that have higher confidence value than the threshold or minimum confidence.

Step-4: Sort the rules as the decreasing order of lift.

Limitations of Apriori Algorithm -

Apriori Algorithm can be slow. The main limitation is time required to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets i.e. it is not an efficient approach for large number of datasets. For example, if there are 10^4 from frequent 1- itemsets, it need to generate more than 10^7 candidates into 2-length which in turn they will be tested and accumulate. Furthermore, to detect frequent pattern in size 100 i.e. v1, v2… v100, it have to generate 2^100 candidate itemsets that yield on costly and wasting of time of candidate generation. So, it will check for many sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions. [Source : Source].

5.PROBLEM STATEMENT:

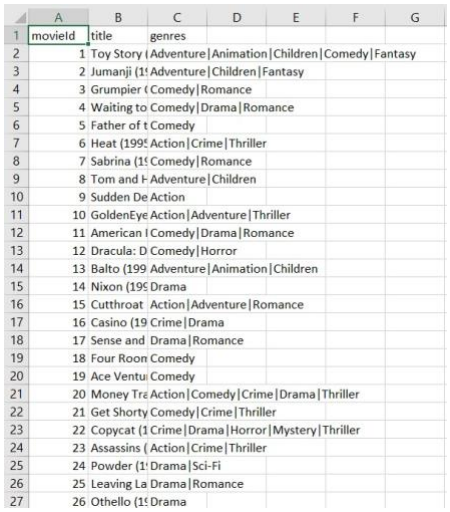
The problem is to find similar kind of movies according to the viewers rating and recommend to the user.

6. DATASET DETAILS:

Two datasets in ".CSV" file format named "movies.csv" and "ratings.csv" which later was read by the Pandas library of Python. The various columns present in the dataset are:

movies.csv:

movies.csv



	A	B	C	D	E	F	G
1	movieId	title	genres				
2	1	Toy Story (	Adventure|Animation|Children|Comedy|Fantasy				
3	2	Jumanji (1!	Adventure|Children|Fantasy				
4	3	Grumpier (	Comedy|Romance				
5	4	Waiting to	Comedy|Drama|Romance				
6	5	Father of t	Comedy				
7	6	Heat (199!	Action|Crime|Thriller				
8	7	Sabrina (1!	Comedy|Romance				
9	8	Tom and H	Adventure|Children				
10	9	Sudden De	Action				
11	10	GoldenEye	Action|Adventure|Thriller				
12	11	American I	Comedy|Drama|Romance				
13	12	Dracula: D	Comedy|Horror				
14	13	Balto (199	Adventure|Animation|Children				
15	14	Nixon (19!	Drama				
16	15	Cutthroat	Action|Adventure|Romance				
17	16	Casino (19	Crime|Drama				
18	17	Sense and	Drama|Romance				
19	18	Four Roon	Comedy				
20	19	Ace Ventu	Comedy				
21	20	Money Tra	Action|Comedy|Crime|Drama|Thriller				
22	21	Get Shorty	Comedy|Crime|Thriller				
23	22	Copycat (1	Crime|Drama|Horror|Mystery|Thriller				
24	23	Assassins (	Action|Crime|Thriller				
25	24	Powder (1!	Drama|Sci-Fi				
26	25	Leaving La	Drama|Romance				
27	26	Othello (1!	Drama				

ratings.csv:

ratings.csv

	A	B	C	D	E
1	userId	movieId	rating	timestamp	
2	1	1	4	9.65E+08	
3	1	3	4	9.65E+08	
4	1	6	4	9.65E+08	
5	1	47	5	9.65E+08	
6	1	50	5	9.65E+08	
7	1	70	3	9.65E+08	
8	1	101	5	9.65E+08	
9	1	110	4	9.65E+08	
10	1	151	5	9.65E+08	
11	1	157	5	9.65E+08	
12	1	163	5	9.65E+08	
13	1	216	5	9.65E+08	
14	1	223	3	9.65E+08	
15	1	231	5	9.65E+08	
16	1	235	4	9.65E+08	
17	1	260	5	9.65E+08	
18	1	296	3	9.65E+08	
19	1	316	3	9.65E+08	
20	1	333	5	9.65E+08	
21	1	349	4	9.65E+08	
22	1	356	4	9.65E+08	
23	1	362	5	9.65E+08	
24	1	367	4	9.65E+08	
25	1	423	3	9.65E+08	
26	1	441	4	9.65E+08	
27	1	457	5	9.65E+08	

7.WHAT IS ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING?

Artificial Intelligence: - Artificial intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks—as, for example, discovering proofs for mathematical theorems or playing chess—with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human flexibility over wider domains or in tasks requiring much everyday knowledge. On the other hand, some programs have attained the performance levels of human experts and professionals in performing certain specific tasks, so that artificial intelligence in this limited sense is found in applications as diverse as medical diagnosis, computer search engines, and voice or handwriting recognition.

MACHINE LEARNING: - Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves. With machine learning algorithms, AI was able to develop beyond just performing the tasks it was programmed to do. Before ML entered the mainstream, AI programs were only used to automate low-level tasks in business and enterprise settings.

This included tasks like intelligent automation or simple rule-based classification. This meant that AI algorithms were restricted to only the domain of what they were processed for. However, with machine learning, computers were able to move past doing what they were programmed and began evolving with each iteration.

Machine learning is fundamentally set apart from artificial intelligence, as it has the capability to evolve. Using various programming techniques, machine learning algorithms are able to process large amounts of data and extract useful information. In this way, they can improve upon their previous iterations by learning from the data they are provided.

We cannot talk about machine learning without speaking about big data, one of the most important aspects of machine learning algorithms. Any type of AI is usually dependent on the quality of its dataset for good results, as the field makes use of statistical methods heavily.

Machine learning is no exception, and a good flow of organized, varied data is required for a robust ML solution. In today's online-first world, companies have access to a large amount of data about their customers, usually in the millions. This data, which is both large in the number of data points and the number of fields, is known as big data due to the sheer amount of information it holds.

Big data is time-consuming and difficult to process by human standards, but good quality data is the best fodder to train a machine learning algorithm. The more clean, usable, and machine-readable data there is in a big dataset, the more effective the training of the machine learning algorithm will be.

As explained, machine learning algorithms have the ability to improve themselves through training. Today, ML algorithms are trained using three prominent methods. These are three types of machine learning: supervised learning, unsupervised learning, and reinforcement learning.

8. HARDWARE AND SOFTWARE REQUIREMENTS: -

Hardware Requirements: -

- Processor: Intel i5 or above
- RAM: Minimum 8GB or more.
- Hard Disk: Minimum 4 GB of space
- Input Device: Keyboard
- Output Device: Screens of Monitor or a Laptop

Software Requirement: -

- Operating system: Windows & Linux
- Python: 3.9
- Python Modules: streamlit, tensorflow
- Anaconda:3.5
- IDE: Jupiter Notebook
- Visualization: matplotlib, pandas.
- Server: Web Server with HTTP process.

9. SNAPSHOTS:

MOVIE RECOMMENDATION ENGINE

MOVIE RECOMMENDATION ENGINE : A Engine which will recommend best movie to the user

Importing library files

1. Pandas
2. Numpy
3. Matplotlib
4. Seaborn

```
[ ] import warnings
    warnings.filterwarnings('ignore')
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
```

Load the dataset through pandas library

```
[ ] movies=pd.read_csv('movies.csv')  # movies data
    ratings= pd.read_csv('ratings.csv')  # ratings data
    movies.head(2) # movies dataset visualization
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy

```
[ ] ratings.head(2)  # ratings dataset visualization
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247

Find all types of unique movie genres

```
[ ] movies['genres'].unique()
```

```
array(['Adventure|Animation|Children|Comedy|Fantasy',
       'Adventure|Children|Fantasy', 'Comedy|Romance',
       'Comedy|Drama|Romance', 'Comedy', 'Action|Crime|Thriller',
       'Adventure|Children', 'Action', 'Action|Adventure|Thriller',
       'Comedy|Horror', 'Adventure|Animation|Children', 'Drama',
       'Action|Adventure|Romance', 'Crime|Drama', 'Drama|Romance',
       'Action|Comedy|Crime|Drama|Thriller', 'Comedy|Crime|Thriller',
       'Crime|Drama|Horror|Mystery|Thriller', 'Drama|Sci-Fi',
       'Children|Drama', 'Adventure|Drama|Fantasy|Mystery|Sci-Fi',
       'Mystery|Sci-Fi|Thriller', 'Children|Comedy', 'Drama|War',
       'Action|Crime|Drama', 'Action|Adventure|Fantasy',
       'Comedy|Drama|Thriller', 'Mystery|Thriller',
       'Animation|Children|Drama|Musical|Romance',
       'Crime|Mystery|Thriller', 'Adventure|Drama', 'Drama|Thriller',
       'Comedy|Crime', 'Action|Sci-Fi|Thriller',
       'Action|Comedy|Horror|Thriller', 'Comedy|Drama', 'Documentary',
       'Action|Crime|Drama|Thriller', 'Crime|Drama|Romance',
       'Action|Adventure|Drama', 'Action|Thriller',
       'Drama|Horror|Thriller', 'Comedy|Horror|Romance',
       'Adventure|Comedy|Crime|Romance',
       'Adventure|Children|Comedy|Musical', 'Action|Drama|War',
       'Crime|Drama|Thriller', 'Action|Adventure|Comedy|Crime',
       'Drama|Mystery', 'Drama|Mystery|Romance', 'Thriller',
       'Adventure|Drama|IMAX', 'Action|Drama|Romance|War', 'Drama|Horror',
       'Adventure|Drama|War', 'Comedy|War', 'Crime|Drama|Mystery',
       'Action|Adventure|Mystery|Sci-Fi', 'Drama|Thriller|War',
       'Action|Romance|Western', 'Crime|Film-Noir|Mystery|Thriller',
```

Find all types of unique ratings of movies

```
[ ] ratings['rating'].unique()
```

```
array([4. , 5. , 3. , 2. , 1. , 4.5, 3.5, 2.5, 0.5, 1.5])
```

Find all types of unique movie titles

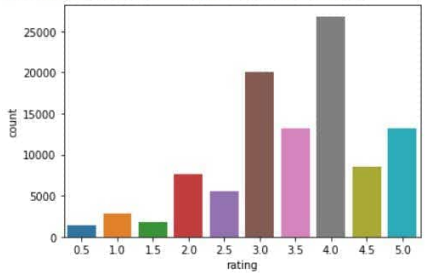
```
[ ] movies['title'].unique()
```

```
array(['Toy Story (1995)', 'Jumanji (1995)', 'Grumpier Old Men (1995)',
       ..., 'Flint (2017)', 'Bungo Stray Dogs: Dead Apple (2018)',
       'Andrew Dice Clay: Dice Rules (1991)'], dtype=object)
```

Countplot of user ratings

```
[ ] sns.countplot(ratings['rating'])
```

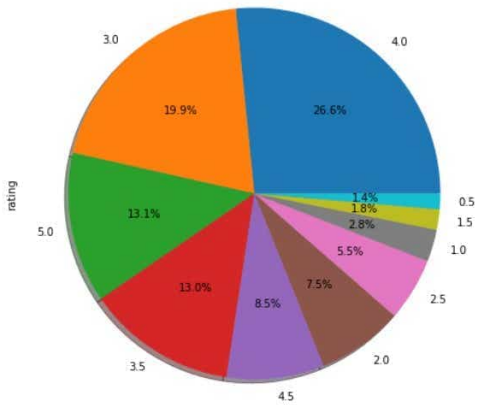
```
<AxesSubplot:xlabel='rating', ylabel='count'>
```



Pie Plot of Movies ratings

```
plt.figure(figsize=(10,8))
ratings['rating'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

[ ]




```
ratings=ratings[['userId','movieId','rating']] #separate columns from timestamp column
ratings_df=ratings.groupby(['userId','movieId']).agg(np.max) #Groupby data
ratings_df.head()
```

```
[ ] ratings=ratings[['userId','movieId','rating']] #separate columns from timestamp column
    ratings_df=ratings.groupby(['userId','movieId']).agg(np.max) #Groupby data
    ratings_df.head()
```

		rating
userId	**movieId**	
1	**1**	4.0
	3	4.0
	6	4.0
	47	5.0
	50	5.0

On the above, groupby data based on : Each user viewed which movies and how much rated for those movies. userid 1 viewed movie ids 1,3,6,47,50 ans so on and gave individual rating for each movie.

```
count_ratings=ratings.groupby('rating').count() #count all ratings
count_ratings
```

[ ]

rating	userId	movieId
0.5	1370	1370
1.0	2811	2811
1.5	1791	1791
2.0	7551	7551
2.5	5550	5550
3.0	20047	20047
3.5	13136	13136
4.0	26818	26818
4.5	8551	8551
5.0	13211	13211

```
count_ratings['perc_total']=round(count_ratings['userId']*100/count_ratings['userId'].sum(),
                                  1)
count_ratings
```

[ ]

rating	userId	movieId	perc_total
0.5	1370	1370	1.4
1.0	2811	2811	2.8
1.5	1791	1791	1.8
2.0	7551	7551	7.5
2.5	5550	5550	5.5
3.0	20047	20047	19.9
3.5	13136	13136	13.0
4.0	26818	26818	26.6
4.5	8551	8551	8.5
5.0	13211	13211	13.1

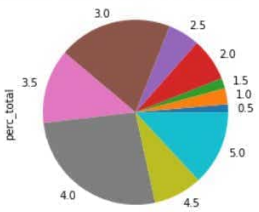
**On the above predict the percentage for each ratings.

1. ratings 5.0 got 13.1 %
2. ratings 4.5 got 8.5 %
3. ratings 4.0 got 26.6 %
4. ratings 3.5 got 13.0 %
5. ratings 3.0 got 19.9 %

3. ratings 4.0 got 26.6 %
4. ratings 3.5 got 13.0 %
5. ratings 3.0 got 19.9 %
6. ratings 2.5 got 5.5 %
7. ratings 2.0 got 7.5 %
8. ratings 1.5 got 1.8 %
9. ratings 1.0 got 2.8 %
10. ratings 0.5 got 1.4 % Highest rating is 4.0 and Lowest rating is 1.4 &**

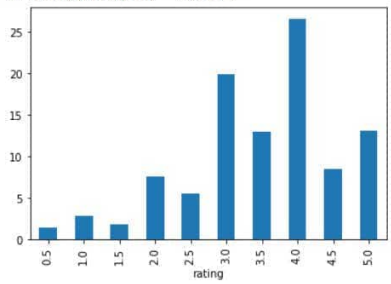
```
# PLotting of each ratings
count_ratings['perc_total'].plot.pie()  #pie plot
```

```
<AxesSubplot:ylabel='perc_total'>
```



```
[ ] count_ratings['perc_total'].plot.bar()  # bar plot
```

```
<AxesSubplot:xlabel='rating'>
```




```
genres=movies['genres']
genres.head()  # visualization of genres column from movie dataset
```

```
0    Adventure|Animation|Children|Comedy|Fantasy
1              Adventure|Children|Fantasy
2                        Comedy|Romance
3                  Comedy|Drama|Romance
4                               Comedy
Name: genres, dtype: object
```

```
genre_list=" "
for index,row in movies.iterrows():
    genre_list+=row.genres+"|"
genre_list_split=genre_list.split("|")
new_list=list(set(genre_list_split))
new_list.remove('')
new_list
```

```
['War',
 'Mystery',
 ' Adventure',
 'Film-Noir',
 'Western',
 'Crime',
 'Animation',
 'Adventure',
 'Children',
 'Comedy',
 'Documentary',
 'Romance',
 'Action',
 '(no genres listed)',
 'Sci-Fi',
 'IMAX',
 'Musical',
 'Fantasy',
 'Drama',
 'Thriller',
 'Horror']
```

On the above split the all movie genres from the dataset and used "set" data structure for non duplication of genres in the same. If in the

```
for genre in new_list:
    m[genre]=m.apply(lambda _:int(genre in _.genres),axis=1)
m.head()
```

	movieId	title	genres	War	Mystery	Adventure	Film-Noir	Western	Crime	Animation	...	Romance	Action	(no genres listed)	Sci-Fi	IMAX	Musical	Fantasy	Drama
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	0	0	0	0	0	0	1	...	0	0	0	0	0	0	1	0
1	2	Jumanji (1995)	Adventure|Children|Fantasy	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0
2	3	Grumpier Old Men (1995)	Comedy|Romance	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0
4	5	Father of the Bride Part II (1995)	Comedy	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 24 columns

Make a Data with title of the movie and genre of the data. If the genre of the movie matches the entire column then it shows 1. Like movie id 3 whose title is Grumpier Old Men(1995) and genre is Comedy|Romance. This movies genre matches the Romance column very much. '...' indicates there are few columns.

```
avg=pd.DataFrame(ratings.groupby('movieId')['rating'].agg(['mean','count']))
avg # make a dataframe for each movie id with its corresponding ratings mean and count
```

movieId	mean	count
1	3.920930	215
2	3.431818	110
3	3.259615	52
4	2.357143	7
5	3.071429	49
...	...	...
193581	4.000000	1
193583	3.500000	1
193585	3.500000	1
193587	3.500000	1
193609	4.000000	1

9724 rows × 2 columns

```
[ ] avg['movieId']=avg.index
    avg # add the movieid column
```

movieId	mean	count	movieId
1	3.920930	215	1
2	3.431818	110	2
3	3.259615	52	3
4	2.357143	7	4
5	3.071429	49	5
...	...	...	...
193581	4.000000	1	193581
193583	3.500000	1	193583
193585	3.500000	1	193585
193587	3.500000	1	193587
193609	4.000000	1	193609

9724 rows × 3 columns

```
[ ] np.percentile(avg['count'],70)
```

```
7.0
```

On the above 70 % movie has average count 7

```
[ ] np.percentile(avg['count'],50)
```

```
3.0
```

On the above 50 % movie has average count 3

```
idx2title={int(row['movieId']):row['title']
           for _,row in movies.iterrows()}
idx2title
```

```
{1: 'Toy Story (1995)',
 2: 'Jumanji (1995)',
 3: 'Grumpier Old Men (1995)',
 4: 'Waiting to Exhale (1995)',
 5: 'Father of the Bride Part II (1995)',
 6: 'Heat (1995)',
 7: 'Sabrina (1995)',
 8: 'Tom and Huck (1995)',
 9: 'Sudden Death (1995)',
 10: 'GoldenEye (1995)',
 11: 'American President, The (1995)',
```

```
1300: 'My Life as a Dog (Mitt liv som hund) (1985)',
1301: 'Forbidden Planet (1956)',
...}
```

On the above display the all movie titles arranged one after the another

```
[ ] title2idx={j:i for i,j in idx2title.items()}
    title2idx
```

```
{'Toy Story (1995)': 1,
 'Jumanji (1995)': 2,
 'Grumpier Old Men (1995)': 3,
 'Waiting to Exhale (1995)': 4,
 'Father of the Bride Part II (1995)': 5,
 'Heat (1995)': 6,
 'Sabrina (1995)': 7,
 'Tom and Huck (1995)': 8,
 'Sudden Death (1995)': 9,
 'GoldenEye (1995)': 10,
 'American President, The (1995)': 11,
 'Dracula: Dead and Loving It (1995)': 12,
 'Balto (1995)': 13,
 'Nixon (1995)': 14,
 'Cutthroat Island (1995)': 15,
 'Casino (1995)': 16,
 'Sense and Sensibility (1995)': 17,
 'Four Rooms (1995)': 18,
 'Ace Ventura: When Nature Calls (1995)': 19,
 'Money Train (1995)': 20,
 'Get Shorty (1995)': 21,
```

```
 'Killing Fields, The (1984)': 1299,
 'My Life as a Dog (Mitt liv som hund) (1985)': 1300,
 'Forbidden Planet (1956)': 1301,
 ...}
```

On the above display the all movie titles arranged one after the another in reverse order

```
highratings=ratings[ratings.rating>=4]
highratings
```

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0
...	...	...	...
100830	610	166528	4.0
100831	610	166534	4.0
100832	610	168248	5.0
100833	610	168250	5.0

Display the movie data whose ratings is greater than or equal to 4

```
[ ]  itemsets=[[idx2title[mov] for mov in highratings[highratings.userId==user].movieId]
              for user in highratings.userId]
     itemsets
```

```
[['Toy Story (1995)',
  'Grumpier Old Men (1995)',
  'Heat (1995)',
  'Seven (a.k.a. Se7en) (1995)',
  'Usual Suspects, The (1995)',
  'Bottle Rocket (1996)',
  'Braveheart (1995)',
  'Rob Roy (1995)',
  'Canadian Bacon (1995)',
  'Desperado (1995)',
  'Billy Madison (1995)',
  'Dumb & Dumber (Dumb and Dumber) (1994)',
  'Ed Wood (1994)',
  'Star Wars: Episode IV - A New Hope (1977)',
  'Tommy Boy (1995)',
  'Clear and Present Danger (1994)',
  'Forrest Gump (1994)',
  'Jungle Book, The (1994)',
  'Mask, The (1994)',
  'Dazed and Confused (1993)',
  'Fugitive, The (1993)',
  'Jurassic Park (1993)',
  "Schindler's List (1993)",
  'So I Married an Axe Murderer (1993)',
```

On the above display all movie names whose rating is greater than or equal to 4.0

Recommendation Engine making starts here

Recommendation engine will be made by using Apriori algorithm which is very much popular algorithm for Association Rule Mining.

```
[ ] from mlxtend.preprocessing import TransactionEncoder
    te=TransactionEncoder()
    tr_ary=te.fit(itemsets).transform(itemsets)
    DF=pd.DataFrame(tr_ary,columns=te.columns_)
    DF.head()
```

	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Salem's Lot (2004)	'Til There Was You (1997)	'burbs, The (1989)	(500) Days of Summer (2009)	*batteries not included (1987)	...And Justice for All (1979)	00 Schneider - Jagd auf Nihil Baxter (1994)	1-900 (06) (1994)	...	Zombieland (2009)	Zookeeper (2011)	Zoolander (2001)	Zootopia (2016)	Zulu (1964)	[REC] (2007)	[REC]² (2009)	eXistenZ (1999)	xX
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	Fals

On the above rows has been converted into columns and columns has been converted into rows. Movie does not rated is False and rated is True.

```
[ ] from mlxtend.frequent_patterns import apriori, association_rules
    f=apriori(DF, min_support=0.2, use_colnames=True, max_len=2)
    rules=association_rules(f,metric='lift',min_threshold=2)
    rules.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Blade Runner (1982))	(2001: A Space Odyssey (1968))	0.336414	0.333059	0.240243	0.714128	2.144151	0.128197	2.333009
1	(2001: A Space Odyssey (1968))	(Blade Runner (1982))	0.333059	0.336414	0.240243	0.721323	2.144151	0.128197	2.381197
2	(Dr. Strangelove or: How I Learned to Stop Wor...	(2001: A Space Odyssey (1968))	0.319926	0.333059	0.218979	0.684468	2.055096	0.112425	2.113703
3	(2001: A Space Odyssey (1968))	(Dr. Strangelove or: How I Learned to Stop Wor...	0.333059	0.319926	0.218979	0.657478	2.055096	0.112425	1.985492
4	(Aladdin (1992))	(Lion King, The (1994))	0.332565	0.331083	0.232462	0.698997	2.111246	0.122355	2.222296

On the above, the movie has been recommended to use user where 'antecedents' is the movie column and 'consequents' is the recommended movie column. In the first row if the see the '(2001:A Space Odyssey(1968))' movie then '(Blade Runner(1982))' will be recommended to you for further view and whose confience of the view is 72 % approximately. Lift column is used for to tell us likelihood of watching both movies together is 2.144151 times more than the likelihood of just watching one movie. Support is the default popularity of an item. In mathematical terms the support of one item is nothing but the ratio of transactions involving one item to the total number of transactions. support(movie_1)=(All transactions involving movie_1)/(total transactions) Conviction compares the probability that X appears without Y if they were dependent with the actual frequency of the appearance of X without Y. Threshold in the Apriori algorithm identifies the item sets which are subsets of at least as each transaction is seen as a set of items.

10. FUTURE SCOPE:

Future advances will boost the value of recommender systems, which can be a very effective tool in a business's toolbox. One use scenario is the ability to predict seasonal purchases based on suggestions, identify significant purchases, and provide customers with better recommendations that can improve retention and brand loyalty.

11. CONLUSION:

In this project we build a Movie Recommendation Engine with basic machine learning algorithm called Apriori algorithm, which analyses the past preferences of the user concerned, and then it uses this information to try to find similar movies. We provided the model with an understanding of the algorithm, methods and discussed the challenges of implementing them at scale. We introduced the core concepts behind the programming of model and provided implementations of the algorithm at the user level. With that this project met all the criteria on the check list and should be useful to many.

12. REFERENCES & BIBLIOGRAPHY:

i) **Movie Recommendation Engine by towardsdatascience**