A Project Report

On

Machine Learning based recommendation system for Movie and TV Shows.

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of Mr. Arvindhan M.
Assistant Professor
Department of Computer Science and Engineering

Submitted By

19SCSE1180061 – SHIVAM SINGH

19SCSE1010240 – TANMAY SINGH

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA

DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"Machine Learning based recommendation system for Movie and TV Shows"** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Mr. Arvindhan M., Assistant Professor**, **Department of Computer Scienceand Engineering** of School of Computing Science and Engineering, Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any otherdegree of this or any other places.

19SCSE1180061 - SHIVAM SINGH

19SCSE1010240 – TANMAY SINGH

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Mr. Arvindhan M.)

CERTIFICATE

The Final Viva-Voce examination of **19SCSE1180061 – SHIVAM SINGH, 19SCSE1010240 – TANMAY SINGH** has been held on ____________________and his/herwork is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

ABSTRACT

The purpose of the project is to research about Content and Collaborative based movie recommendation engines. Nowadays recommender systems are used in our day-to-day life. We try to understand the distinct types of reference engines/systems and compare their work on the movies datasets. We start to produce a versatile model to complete this study and start by developing and

relating the different kinds of prototypes on a minor dataset of 100,000 evaluations.

The growth of e-commerce has given rise to recommendation engines. Several recommendation engines exist within the market to recommend a wide variety of goods to users. These recommendations support

various aspects such as users' interests, users' history, users' locations, and more. Away from all the above aspects, one thing is common which is individuality. Content and collaborative-based movie recommendation engines recommend users based on the user's viewpoint, whereas many things are there within the marketplace that are related to which a user is uninformed of. This stuff should also be suggested by the engine to clients; But due to the range of "individuality", these machines do not suggest things that are out of the crate. The Hybrid System of Movie Recommendation Engine has crossed this variety of individuality. The Movie Recommendation Engine will suggest movies to clients according to their interest and be evaluated by other clients who are almost user-like. Additionally, for this, there are web services that are capable of acting as a tool adornment.

Table of Contents

List of Figures

Acronyms

SVM	Support Vector Machine
ML	Machine Learning
DL	Deep Learning
CNN	Convolution Neural Networks

CHAPTER-1

Introduction

A movie recommendation engine / system can be an information sorting system that works to estimate ratings or preferences and will give the user an item and set up a simple or similar language "recommendation engine / system to attract the user something. Suggests important supported". Recommendatory systems can also enhance the experience for:

- News websites
- Computer games
- Knowledge base
- Social media platform
- Stock trading support system

A content and collaborative-based recommendation engine / system can also be a method of information sorting system that works to predict user preferences and provide suggestions that support them. The content on some platforms extends from movies, music, books and videos to friends. And to produce stories on social platform and on ecommerce websites, for persons on professional and dating websites, returned to see search results. Two critical approaches are mainly used for recommendation engines. First, content-based filtering, where we attempt to profile client interests utilizing gathered information and recommend items that support that profile, and second, collaborative filtering [8] continues where we try and identify together and use information about identities to create recommendations systems. Every user has a different mindset to decide their likes and dislikes. Additionally, even a customer’s taste can look at different aspects, such as mood, seasons, or different activities performed by the user. As an example, the type of music you want to focus on during exercise is severely different from that in which he listens to music while making dinner. They have to find new areas to see more about the customer, while still determining the majority of what is already known about the customer.

Introduction to Simple recommenders:

The simple recommendation system process provides generalized recommendations to each user, supported movie popularity and / or genre. The basic approach behind this technique is that more popular and critically acclaimed movies will be better likely to be liked by the general audience. For example, IMDB Top 250 is an example of this technique.

There are basically three types of recommender systems:

- **Demographic Filtering-** 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 considered to be 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.

- **Content Based Filtering-** They suggest similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations. The general idea behind these recommender systems is that if a person liked a particular item, he or she will also like an item that is similar to it.

- **Collaborative Filtering-** This system matches persons with similar interests and provides recommendations based on this matching. Collaborative filters do not require item metadata like its content-based counterparts.

CHAPTER-2

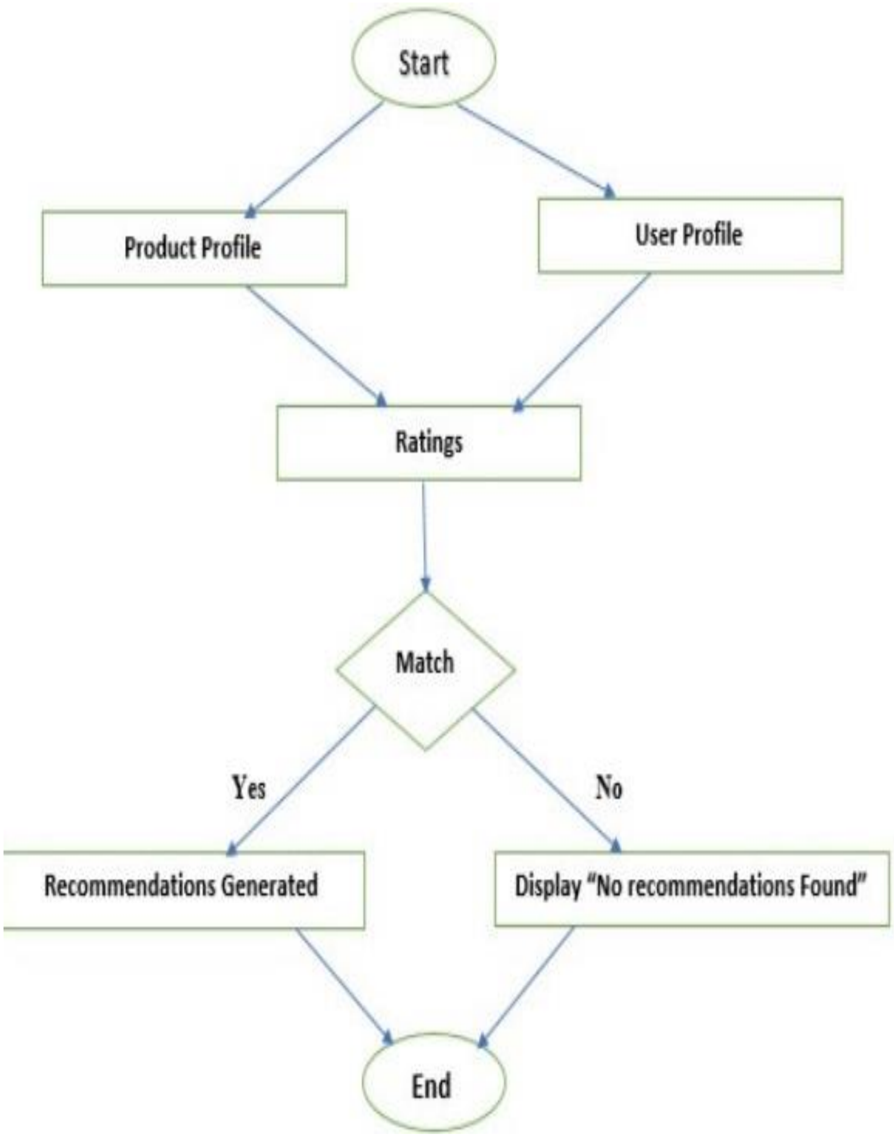
Literature Survey

In the context of a review of the literature, a recommendation system using a content-based collaborative and hybrid approach by a previous researcher is a different approach to the development of recommendation-based engines. In 2007 a web-based and knowledge-based intelligence movie recommendation system has been offered using the hybrid filtering method. In 2017, a movie recommendation system supported style and rating coefficient of correlation purpose by the authors. In 2013 a Bayesian network and trust model-based movie recommendation engine have been recommended to predict ratings for users and items, primarily from datasets to recommend users their choice and vice versa. In 2018, the authors built a recommendation engine by analyzing the ratings dataset collected from Kaggle to recommend movies for a user selected from Python. In 2018 movie recommendation engines provide a process to help users categorize users with similar k mean cuckoo values and reinforcement learning based recommender systems, which are using bicycling techniques. Initial research mainly concentrated on the content of the recommendation system that examined the features of the object to complete the recommendation task. Experiments verified that their approaches were more elastic and precise. Bayesian networks are employed for model-based preferences based on their context. In 2007 Salakhuddinov and Minh proposed a collaborative filtering method, the probabilistic matrix factor, which can handle large-scale datasets. The collaborative filtering algorithm was distributed into portions for deeper study in the movie recommendation by Hurlkartal. When clients adopt new behavior, it is difficult for collaborative filtering to react instantly. Therefore, both researchers and practitioners have a desire to align collaborative filtering method and content-based methodology to solve the issue. Ternary implemented Unplugged Learning of Machine Learning to examine the polarity of machine reflectivity.

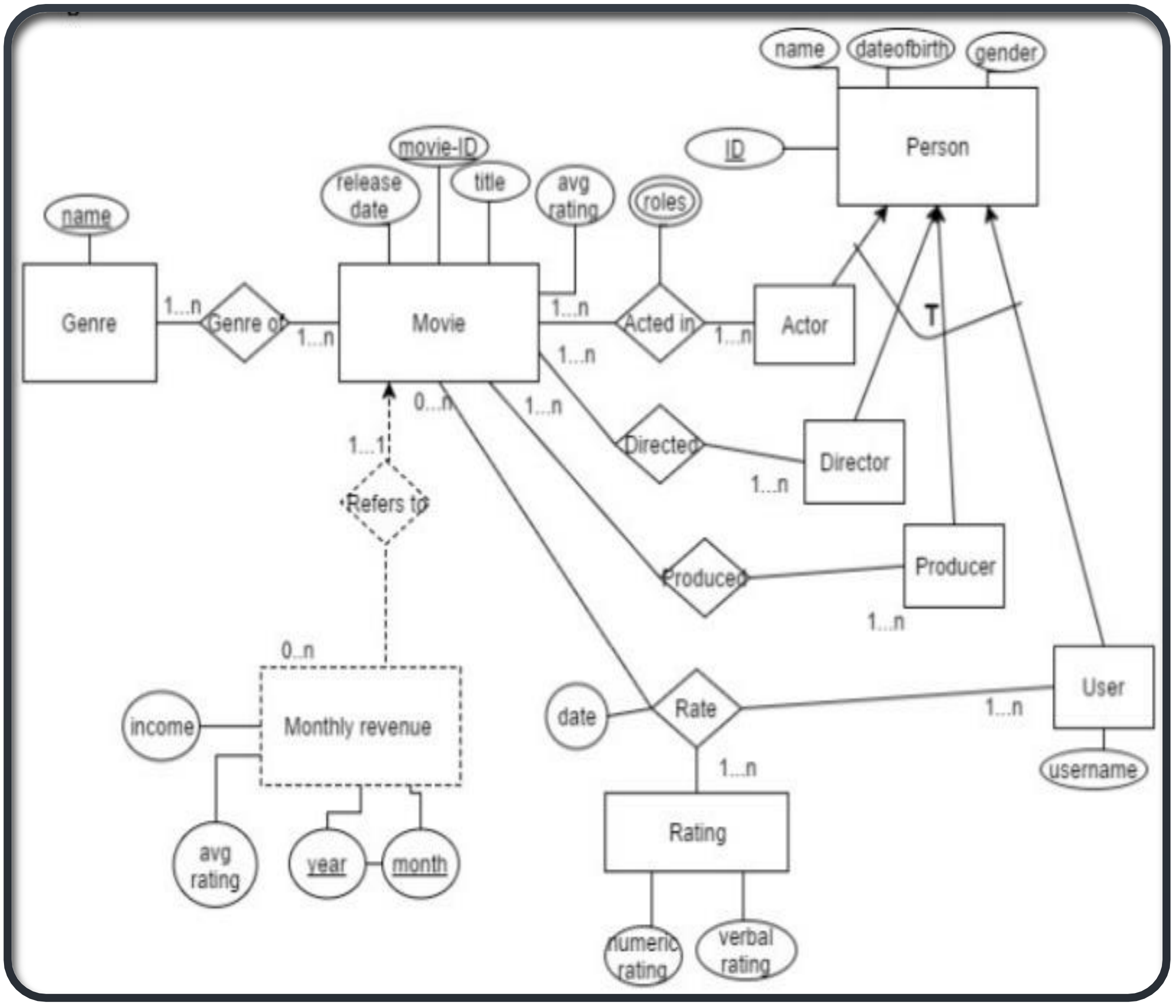
CHAPTER 3

SYSTEM DESIGN

DATA FLOW DIAGRAM:

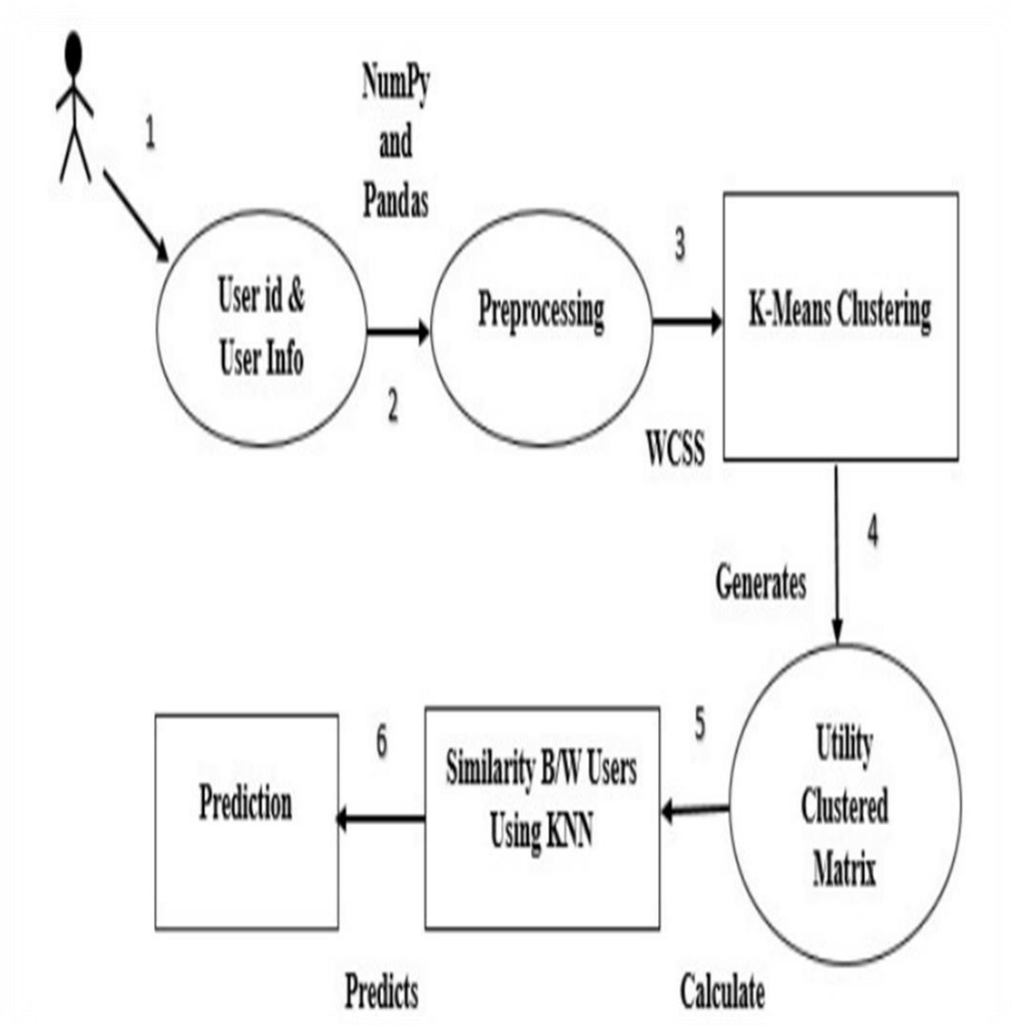


ER DIAGRAM:

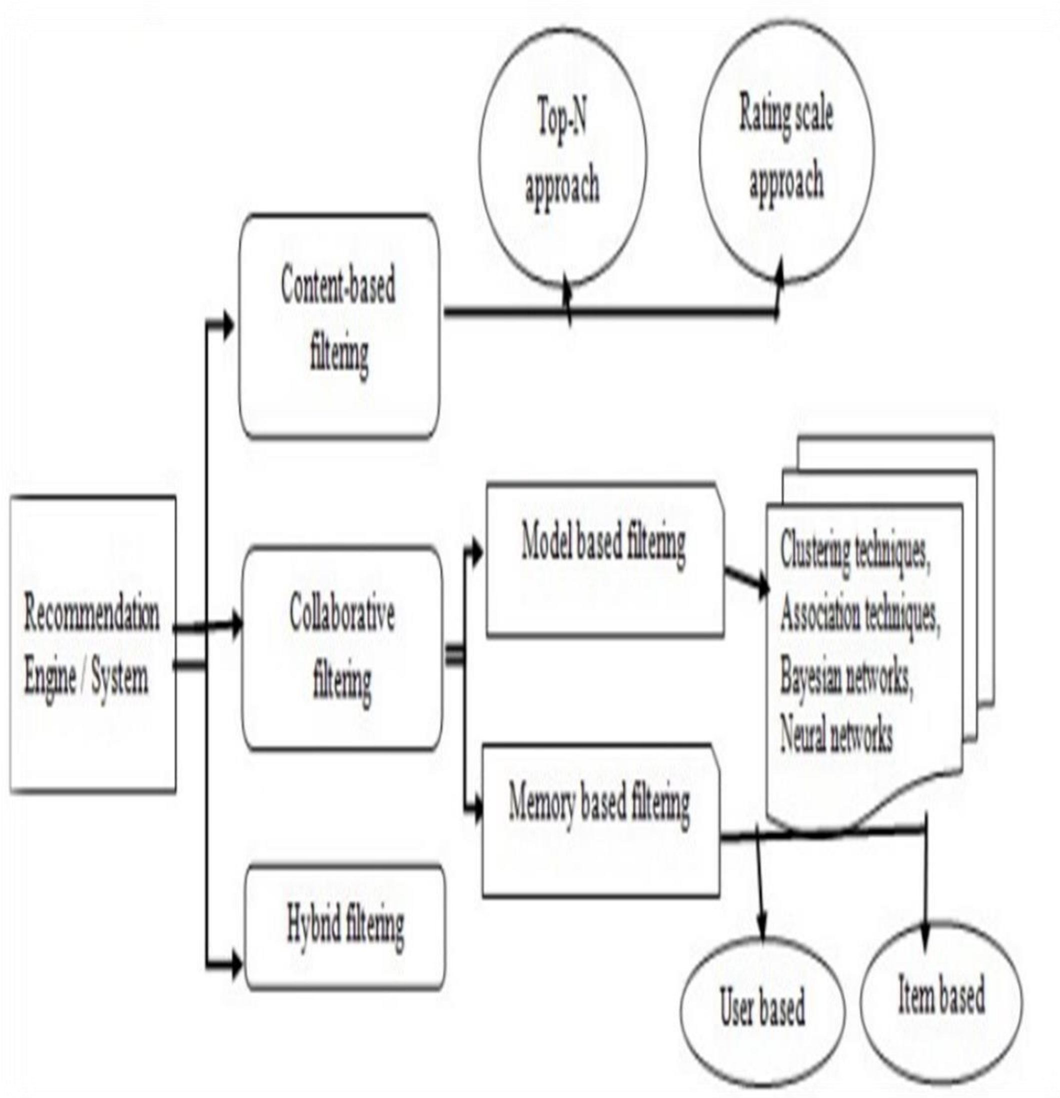


name
dateofbirth
gender
ID
Person
movie-ID
release date
title
avg rating
roles
name
Genre
1...n
Genre of
1...n
Movie
1...n
Acted in
1...n
Actor
1...n
T
0...n
1...n
1...1
Directed
Director
1...n
Refers to
Produced
Producer
1...n
0...n
User
income
Monthly revenue
date
Rate
1...n
username
1...n
avg rating
year
month
Rating
numeric rating
verbal rating

FLOW DIAGRAM:



ARCHITECTURE DIAGRAM:



CHAPTER 4

Dataset

The movie dataset is hired in our research paper and collected from the Kaggle database. The Kaggle database provides datasets in the form of several varieties of movie content. The user rating data consists of records and has a user ID, movie ID, rating, and timestamp. The Characterization of the movie's content information includes over 54058 records and includes movie ID, title, genre, director, actor, and more.

CHAPTER 5

Requirements of Project

- Linux operating System
- Python 2.7
- Flask Framework
- Flask wtForms
- Flask Mysqldb
- Numpy
- Flask Mail
- SciPy
- Scikit-learn

Chapter 6
Implementation

```
import pandas as pd
import numpy as np
df1=pd.read_csv('../input/tmdb-movie-metadata/tmdb_5000_credits.csv')
df2=pd.read_csv('../input/tmdb-movie-metadata/tmdb_5000_movies.csv')
```

The first dataset contains the following features:-

- movie_id - A unique identifier for each movie.
- cast - The name of lead and supporting actors.
- crew - The name of Director, Editor, Composer, Writer etc.

The second dataset has the following features:-

- budget - The budget in which the movie was made.
- genre - The genre of the movie, Action, Comedy ,Thriller etc.
- homepage - A link to the homepage of the movie.
- id - This is infact the movie_id as in the first dataset.
- keywords - The keywords or tags related to the movie.
- original_language - The language in which the movie was made.
- original_title - The title of the movie before translation or adaptation.
- overview - A brief description of the movie.
- popularity - A numeric quantity specifying the movie popularity.
- production_companies - The production house of the movie.
- production_countries - The country in which it was produced.
- release_date - The date on which it was released.
- revenue - The worldwide revenue generated by the movie.

```
df1.columns = ['id','tittle','cast','crew']
df2= df2.merge(df1,on='id')
```

Just a peak at our data.

```
df2.head(5)
```

budget	genres	homepage	id	keywords	original_language	original_titl
237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar
300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End
245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre
250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises
260000000	[{"id": 28, "name": "Action"}, {"id": 12	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on	en	John Carte

Demographic Filtering

```
C= df2['vote_average'].mean()
C
```

OUTPUT:6.092171559442011

```
m= df2['vote_count'].quantile(0.9)
m
```

OUTPUT: 1838.4000000000015

Now, we can filter out the movies that qualify for the chart

```
q_movies = df2.copy().loc[df2['vote_count'] >= m]
q_movies.shape
```

OUTPUT: (481, 23)

```
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)

# Define a new feature 'score' and calculate its value with `weighted_rating()`
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
```

```
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)

#Print the top 15 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

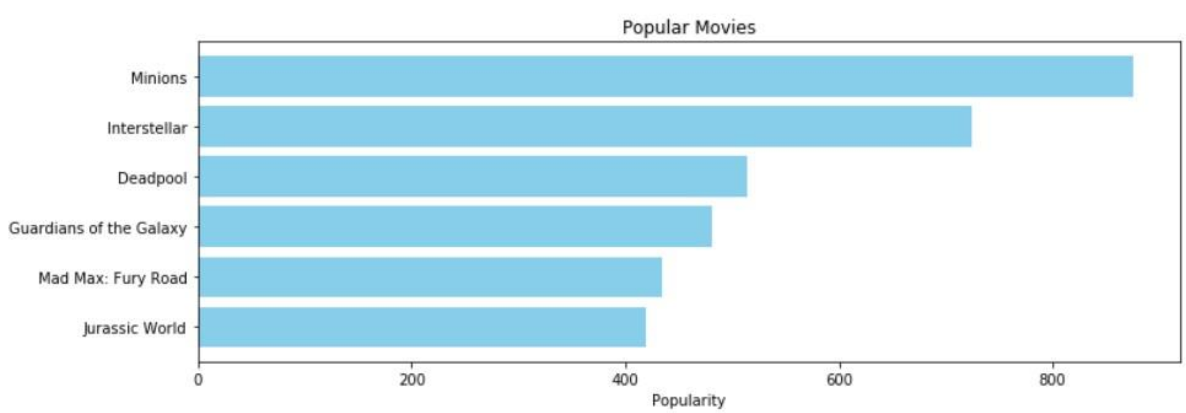
OUTPUT:

	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

```
pop= df2.sort_values('popularity', ascending=False)
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))

plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center',
        color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("Popular Movies")
```

OUTPUT: Text (0.5,1,'Popular Movies')



Content Based Filtering

```
df2['overview'].head(5)
```

OUTPUT:

```
0    In the 22nd century, a paraplegic Marine is di...
1    Captain Barbossa, long believed to be dead, ha...
2    A cryptic message from Bond’s past sends him o...
3    Following the death of District Attorney Harve...
4    John Carter is a war-weary, former military ca...
Name: overview, dtype: object
```

```
#Import TfIdfVectorizer from scikit-learn
from sklearn.feature_extraction.text import TfidfVectorizer

#Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')

#Replace NaN with an empty string
df2['overview'] = df2['overview'].fillna('')

#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(df2['overview'])

#Output the shape of tfidf_matrix
tfidf_matrix.shape
```

OUTPUT: (4803, 20978)

```
# Import linear_kernel
from sklearn.metrics.pairwise import linear_kernel

# Compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

```
#Construct a reverse map of indices and movie titles
indices = pd.Series(df2.index, index=df2['title']).drop_duplicates()

# Function that takes in movie title as input and outputs most similar movies
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df2['title'].iloc[movie_indices]

get_recommendations('The Dark Knight Rises')
```

OUTPUT:

```
65                              The Dark Knight
299                              Batman Forever
428                              Batman Returns
1359                                     Batman
3854    Batman: The Dark Knight Returns, Part 2
119                               Batman Begins
2507                                  Slow Burn
```

```
9           Batman v Superman: Dawn of Justice
1181                                  JFK
210                         Batman & Robin
Name: title, dtype: object
```

get_recommendations('The Avengers')

OUTPUT:

```
7              Avengers: Age of Ultron
3144                         Plastic
1715                         Timecop
4124                 This Thing of Ours
3311              Thank You for Smoking
3033                   The Corruptor
588     Wall Street: Money Never Sleeps
2136         Team America: World Police
1468                    The Fountain
1286                     Snowpiercer
Name: title, dtype: object
```

Credits, Genres and Keywords Based Recommender

```
# Parse the stringified features into their corresponding python objects
from ast import literal_eval

features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(literal_eval)
# Get the director's name from the crew feature. If director is not listed, return NaN
def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
# Returns the list top 3 elements or entire list; whichever is more.
def get_list(x):
```

```
    if isinstance(x, list):
        names = [i['name'] for i in x]
        #Check if more than 3 elements exist. If yes, return only first three. If no, return en
tire list.
        if len(names) > 3:
            names = names[:3]
        return names

    #Return empty list in case of missing/malformed data
    return []
# Define new director, cast, genres and keywords features that are in a suitable form.
df2['director'] = df2['crew'].apply(get_director)

features = ['cast', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(get_list)

# Print the new features of the first 3 films
df2[['title', 'cast', 'director', 'keywords', 'genres']].head(3)
```

OUTPUT:

	title	cast	director	keywords	genres
0	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	James Cameron	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley]	Gore Verbinski	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux]	Sam Mendes	[spy, based on novel, secret agent]	[Action, Adventure, Crime]

```
# Function to convert all strings to lower case and strip names of spaces
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        #Check if director exists. If not, return empty string
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
```

```
        else:
            return ''

# Apply clean_data function to your features.
features = ['cast', 'keywords', 'director', 'genres']

for feature in features:
    df2[feature] = df2[feature].apply(clean_data)
def create_soup(x):
    return ' '.join(x['keywords']) + ' ' + ' '.join(x['cast']) + ' ' + x['director'] + ' ' + ' '.join(x['
genres'])
df2['soup'] = df2.apply(create_soup, axis=1)

# Import CountVectorizer and create the count matrix
from sklearn.feature_extraction.text import CountVectorizer

count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df2['soup'])

# Compute the Cosine Similarity matrix based on the count_matrix
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)

# Reset index of our main DataFrame and construct reverse mapping as before
df2 = df2.reset_index()
indices = pd.Series(df2.index, index=df2['title'])
```

```
get_recommendations('The Dark Knight Rises', cosine_sim2)
```

OUTPUT:

```
65                 The Dark Knight
119                  Batman Begins
4638    Amidst the Devil's Wings
1196                  The Prestige
3073             Romeo Is Bleeding
3326                Black November
1503                        Takers
1986                        Faster
303                       Catwoman
747                  Gangster Squad
Name: title, dtype: object
```

```
get_recommendations('The Godfather', cosine_sim2)
```

OUTPUT:

```
867      The Godfather: Part III
2731      The Godfather: Part II
4638    Amidst the Devil's Wings
2649          The Son of No One
1525             Apocalypse Now
1018             The Cotton Club
1170     The Talented Mr. Ripley
1209              The Rainmaker
1394              Donnie Brasco
1850                  Scarface
Name: title, dtype: object
```

Chapter 7

Conclusion and Future Work

We have implemented a movie recommendation engine / system using simple recommendations, content-based filtering, collaborative filtering, and hybrid systems. In addition, a movie recommendation engine has been developed using different method prediction methods. This model is implemented in the python programming language. We have observed that the RMSE value of the proposed technique is healthier than the current technology after implementing the system with the help of python programming language. In future, we can try and test the system using more data and improve the accuracy of the system. In addition, we can try users better to increase the accuracy of the recommendation system.

Chapter 8

References

[1] Lin, Y., Chen,T., Yu, L. Using Machine Learning to assist crime prevention. In: 2017 sixth IIAI-AAI

[2] Kerr, J.: Vancouver police go high tech to predict and prevent crime before it happens. Vancouver Courier, July 23, 2017.

[3] Marchant, R., Haan, S., clancey, G., Cripps, S.: Applying machine learning to criminology: semi parametric spatial demographic Bayesian regression. Security informs. (2018).

[4] M. J. H. B. T. A. M. K. T. Baig, M.Q., "Artificial intelligence, modelling and simulation (aims), 2014 2nd international conference on," pp. 109–114, November 2014.

[5] Anitha A, Paul G and Kumari S2016 A cyber defence using Artificial Intelligence International Journal of Pharmacy and Technology 8 2532-57

[6] Zeroday. "a lua based firmware for wifi-soc esp8266",Github. Retrieved APR 2015

[7] A. Bogomoloy, B. Lepri, J. Staiano, N. Oliver, F.Pianesi and A. Pentland, 'once upon a crime: towards Crime Prediction from Demographics and Mobile Data', CoRR, vol. 14092983, 2014.

[8] R. Arulanandam, B. Savarimuthu and M. Purvis, 'Extracting Crime Information from Online Newspaper Articles', in Proceedings of the Second Australasian Web Conference - Volume 155, Auckland, New Zealand, 2014, pp. 31-38.

[9] A. Buczak and C. Gifford, 'Fuzzy association rule mining for community crime pattern discovery', in ACM SIGKDD Workshop on Intelligence and Security Informatics, Washington, D.C., 2010, pp. 1-10.

[10] M. Tayebi, F. Richard and G. Uwe, 'Understanding the Link Between Social and Spatial Distance in the Crime World', in Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12), Redondo Beach, California, 2012, pp. 550-

553.

[11] S. Nath, 'Crime Pattern Detection Using Data Mining', in Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference on, 2006, pp. 41,44.

[12] Crimereports.com, 2015.

[13] S. Chainey, L. Tompson and S. Uhlig, 'The Utility of Hotspot Mapping for Predicting Spatial Patterns of Crime', Security Journal, vol. 21, no. 1-2, pp. 4-28, 2008.

[14] Data.denvergov.org, 'Denver Open Data Catalog: Crime', 2015.

[15] Imgh.us, 2015. [Accessed: 20- May-2015].

[16] O. Knowledge, 'Crime — Datasets - US City Open Data Census', Us-city.census.okfn.org, 2015