



BSD3533 DATA MINING

2022/2023 SEMESTER I

GROUP: GROUP B

TITLE:

MOVIE RECOMMENDATION SYSTEM

PREPARED FOR

DR KU MUHAMMAD NA'IM BIN KU KHALIF

PREPARED BY

MATRIC ID	NAME	SECTION
SD20022	NIK NUR AIN BINTI NIK JID	01G
SD20034	NIK NURUL SYUHADA BINTI MOHD ALI	01G
SD20043	SITI NOR BALQIS BINTI JAFAR	01G
SD20035	YANG NUR AZIZAH BINTI MOHD. KAMARULZAMAN	02G
SD20050	PUTRI ARINA BINTI SLAMET	02G

GROUP MEMBERS PHOTOS

NIK NUR AIN BINTI NIK JID	 A portrait of a young woman with dark hair, wearing a black hijab and a red long-sleeved shirt, smiling at the camera. She is standing indoors in front of a window with a view of a modern building.
NIK NURUL SYUHADA BINTI MOHD ALI	 A portrait of a young woman wearing a pink hijab and a dark blue long-sleeved shirt, standing on a wooden bridge. The bridge has white railings and is decorated with red lanterns. There are green trees and a small pavilion in the background.
SITI NOR BALQIS BINTI JAFAR	 A portrait of a young woman wearing a pink hijab and a purple patterned long-sleeved shirt, standing indoors. She is holding her hands in front of her and looking at the camera.
YANG NUR AZIZAH BINTI MOHD. KAMARULZAMAN	 A portrait of a young woman wearing a white hijab with a floral pattern and a white long-sleeved shirt, smiling at the camera. She is standing indoors.
PUTRI ARINA BINTI SLAMET	 A portrait of a young woman wearing a black hijab and a blue long-sleeved shirt, standing outdoors. She is holding a red bag and looking at the camera. There are green plants and a stone wall in the background.

TABLE OF CONTENTS

1.0 EXECUTIVE SUMMARY	3
1.1 Description of the Selected Project	3
1.2 Problem to be Solved	5
1.3 Basic Description of the Data Selected	5
2.0 SUMMARY OF THE PROJECT CONTEXT AND OBJECTIVES	7
2.1 Summary of the Project Context	7
2.2 Objectives	8
3.0 METHODOLOGY	9
3.1 Data Pipeline (ETL)	9
3.2 Data Collection	9
3.3 Data Preparation	10
3.4 Movies Analysis	15
3.5 Modelling	20
3.5.1 K-Means Clustering	21
3.5.2 Association Rules	22
3.6 Model Evaluation	25
3.6.1 K-Means Clustering	25
3.6.2 Association Rules	26
4.0 RESULT AND DISCUSSION	27
4.1 Results	27
4.2 Visualization	32
4.2 GUI of Movie Recommendation	33
5.0 CONCLUSION	38
REFERENCES	39

1.0 EXECUTIVE SUMMARY

1.1 Description of the Selected Project

A recommendation system is a type of information filtering system that sorts items based on the user's interests. Due to problem overload in recent decades, the recommendation system has become an inalienable part of ecommerce and social websites. In an age of information overload, recommendation systems have evolved to help users find interesting items based on their preferences or choices. It was used in a variety of fields, including online learning, e-commerce, and so on. YouTube, Amazon.com, Movie Lens, Netflix, Facebook, and other sites are examples of such applications.

Besides, making a full commitment to completing daily tasks in life must be a lot of stress for some people. Movies have played a significant role in people's lives for the past century as a platform to relieve stress. Sometimes people just need some easy entertainment to take themselves away from the reality of life for a while. Hence, watching a movie is a key for it as it is the easiest activity to do. However, not all people like to watch all types of genres for a movie. Searching for a movie that we want to watch that is preferable to our taste sometimes will take a lot of time.

To overcome the time-consuming problem of finding movies, a movie recommendation system has been employed for this project. The system will help its users to find the best movie that suits their taste. The model that has been implement in the movie recommendation system are K-Means Clustering and Association Rules. Both of this model will find a set of movies to be recommend to their users that is similar to the movie that they has watch. The algorithm to create the output is based on the movies that has been watch by the other users. The list of movies that has been watch by one user will be put into a cluster to be link with the input movie which is the searched movie and to see the probability of a movie to be watch after a certain movie.

1.2 Problem to be Solved

Nowadays, people prefer to watch movies through various applications offered such as Youtube, Netflix, and others. However, there are some problems that can be seen such as people find it difficult to choose the movies they want to watch. There are so many movies to choose from and it is hard to narrow down the options and find something that fits people tastes and preferences. In addition, with so many platforms that can watch movies as state above make it harder to find one movies that want to watch. A movie recommendation system is one of the great way to overcome this problem because it will help to suggest movies based on the user viewing history and preferences. The system will help to discover new movies that user may not have considered because the recommended movies will be based on other users history that has similar tastes and preferences. Other than that, the user may choose to use filters to marrow down options of movies to be watch based on the popularity and genre. This will help the user to look for the reviews, ratings, and trailers of the movie to help them with the decision of whether to watch the movie or not.

1.3 Basic Description of the Data Selected

Data that has been chosen to be used in this project are from Kaggle which is an open source data platform. Three different datasets that has link with each other about movie's information has been choose to be used in the recommendation system. Table 1.1 shows list of attributes in each dataset.

Table 1.1 List of Attributes in Datasets

CSV Name	movies_metadata	movies	ratings
List of Attributes	adult belongs_to_collection budget genres homepage id imdb_id original language original_title	movieId movie_title genres	userId movieId rating timestamp

	overview popularity poster_path production_companies production_countries release_date revenue runtime spoken_languages status tagline title video vote_average vote_count		
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	--

All this three dataset will be combined into one dataframe to make the analysis and recommendation of movies easier. The movies and ratings datasets will be merged under movieID. The attribute 'title' in movies_metadata dataset will be rename to movie_title. After that movies_metadata and the merged dataset will be merge under movie_title attribute.

2.0 SUMMARY OF THE PROJECT CONTEXT AND OBJECTIVES

2.1 Summary of the Project Context

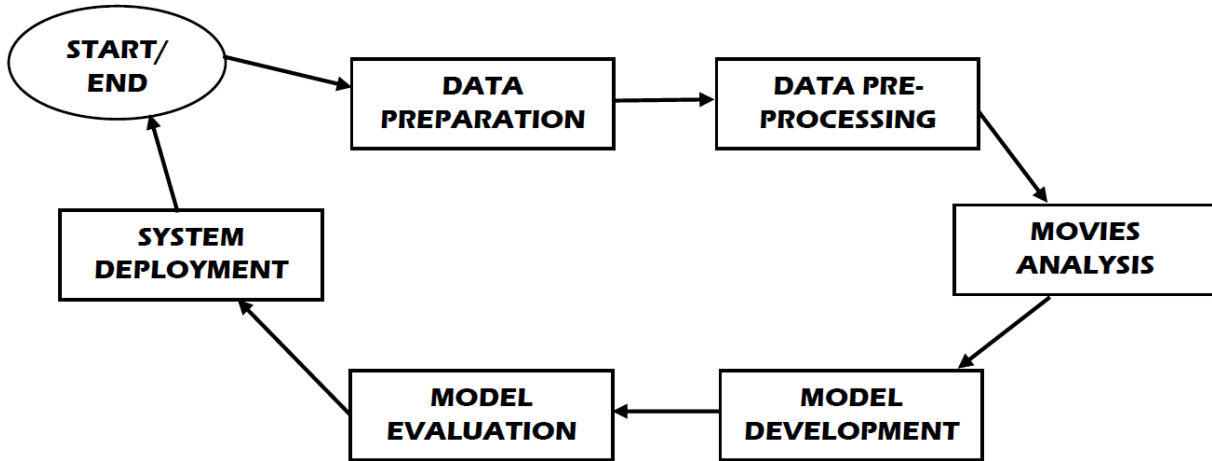


Figure 2.1 Project Cycle

After the data has been prepared, the data will be clean in the pre-processing step. This process is to remove the noise and messy data. The step that needs to be done in the pre-processing stage is to remove the null values, remove the duplicated data, and drop all the unnecessary attributes. After that, the cleaned data will be analyzed to see the top movies in the dataset based on the rating, popularity, and genres of the movies. To proceed with the development of a recommendation system for the movies, the model that suited the process will be developed which are K-Means Clustering and Association Rules algorithms. Next, the model that has been developed will go through the evaluation process to see the accuracy of the output given. The last step to complete the project cycle is to develop the recommendation system based on the results given by the model with interactive interface to be used for the end user.

The main point of this project is to focus on development of movie recommendations according to user interests. To accurately recommend the movies to users, K-Means clustering algorithm along with a pre filter has been applied. Each cluster of movies has been identified in order to determine the best movies for users. The process of determining movie recommendations for users involves the use of movie information such as genre and popularity.

This project can assist users in finding the best movies based on their interests, which can indirectly save users time from searching for movies that do not suit their tastes.

2.2 Objectives

There are two (4) objectives that must be met to achieve the aim of this project which is to determine the best movies to recommend to the users. Below is the objectives:

1. To see the top movies based on the rating and popularity of the movie.
2. To see the top movies in each category of the genres.
3. To develop a movie recommendation system based on user preferences.
4. To see the probability of a movie to be watched after a specific movie.

3.0 METHODOLOGY

3.1 Data Pipeline (ETL)

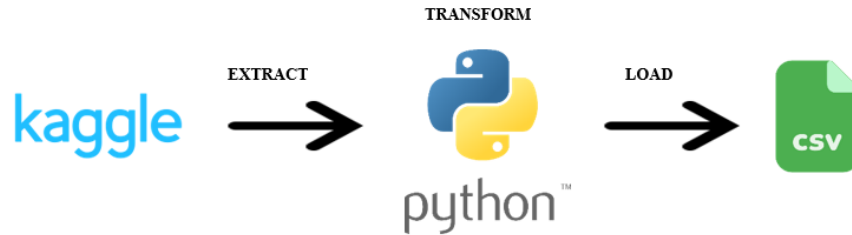


Figure 3.1 ETL Pipeline

Figure 3.1 shows the pipeline of Extract, Transform, Load (ETL) for the dataset of movie recommendation. In the ETL process, it can extract data from various data sources, transform the data, and then load the data into a data warehouse system. In this project, we take a few datasets from Kaggle as the data source for this project. After that all the dataset will be extracted to Jupyter Notebook to go through the pre-processing transformation process by using python programming language. In the transform step, the datasets will be merged into one single data frame, and then will be cleaned to remove the messy data. The data that has been cleaned will be saved into a new csv file to proceed with the analysis of the dataset.

3.2 Data Collection

First and foremost, datasets should be collected before proceed with the analysis and movie recommendation. Therefore, three different dataset has been collected from the link below:

- Movie Metadata:
https://www.kaggle.com/code/rounakbanik/movie-recommender-systems/data?select=movies_metadata.csv
- Movies List:
<https://www.kaggle.com/datasets/bandikarthik/movie-recommendation-system?select=movies.csv>

- Rating by user:

https://www.kaggle.com/code/rounakbanik/movie-recommender-systems/data?select=ratings_small.csv

The movie metadata dataset consists of the information about the movie such as movie name, genres, overview, vote_average, vote count etc. The movie list consists of movie id and title. The rating list consists of user id, movie id, and rating by user. The important of this three dataset is to merge the rating by user with movie metadata because the movie metadata is the main dataset that we want to analyse. However, these two dataset cannot be merged directly with each other because there are no attributes that can be linked together. To solve this problem, the movies list has been used as the middle platform to merge that two dataset.

3.3 Data Preparation

The first step in data preparation is to collect the dataset from various data sources. After that all the dataset that has been collected will extract into Jupyter Notebook to start the pre-processing process. Figure below shows that the datasets have been loaded into three different data frames.

```
#extract data
movieMetadata = pd.read_csv("movies_metadata.csv")
movies = pd.read_csv("movies.csv")
ratings = pd.read_csv("ratings.csv")
```

From the overview of movies and ratings dataset, we can see that both data has movies id attributes.

```
movies.head()
```

	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


```
[ ] ratings.head()
```

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

Merged both dataframe using movieId as shown in the figure below.

```
#merged the movies and ratings by movieId
merged = movies.merge(ratings, on="movieId")

#display
merged.head()
```

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	7	3.0	851866703
1	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	9	4.0	938629179
2	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	13	5.0	1331380058
3	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	15	2.0	997938310
4	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	19	3.0	855190091

However, from the merged dataset above, we can see that the movie title is including the year released of that movie. Hence, we need to remove the year to be match the movie title from merged data with the movie title in the movie metadata. After that rename the title from movie metadata same as the merged data which is movie_title to combine both dataset based on the movie_title.

```
[ ] #clean the movie name and create in the new column:

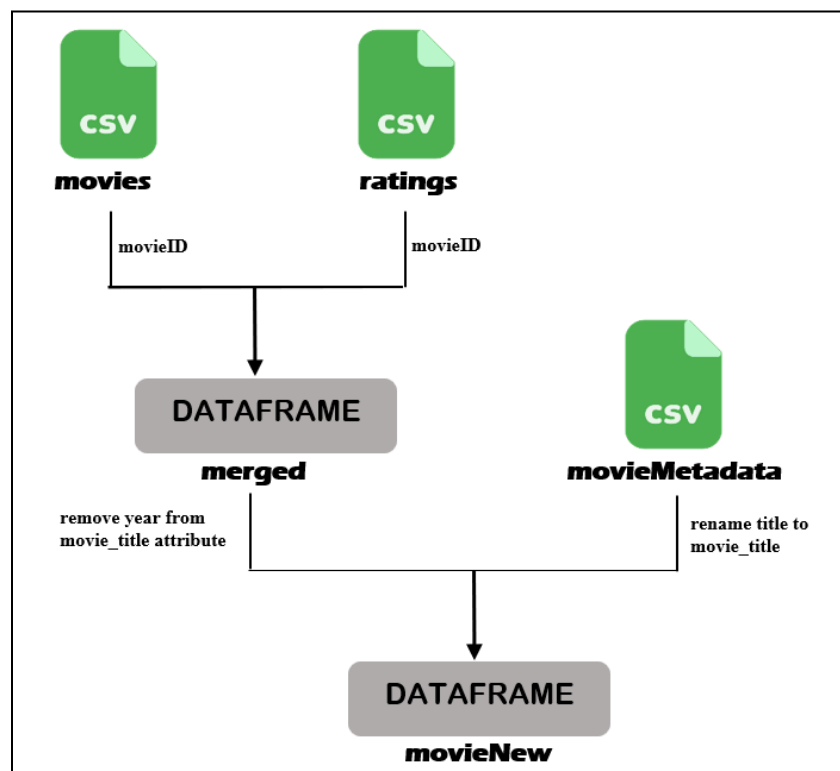
#remove (year) at the title in merged data
merged["movie_title"] = [movie.split(" ")[0] for movie in merged["title"]]

#rename column title in movieMetadata
movieMetadata.rename(columns = {'title':'movie_title'}, inplace = True)

#merged both dataset
movieNew = merged.merge(movieMetadata, how="right",on="movie_title")

#display new data frame
movieNew.head()
```

Summary of the merging process is shown in the figure below.



```
print('No of attributes: ', len(movieNew.columns))
print('No of rows: ', len(movieNew))
```

```
No of attributes: 30
No of rows: 127283
```

Based on the figure above, we can see that the new data frame consists of 30 attributes and 127,283 rows of data.

```
[ ] movieNew.drop(["title", "genres_x", "timestamp", "adult", "homepage", "id", "original_title",
                  "overview", "poster_path", "tagline", "video", "belongs_to_collection"], axis=1, inplace=True)
```

After all three datasets have been merged, the data need to be clean. Before starting the cleaning process, we drop all the unnecessary attributes from the dataset that doesn't have important information.

```
[ ] #checking null
movieNew.isnull().sum()
```

movieId	37997
userId	37997
rating	37997
movie_title	6
budget	0
genres_y	0
imdb_id	17
original_language	11
popularity	5
production_companies	3
production_countries	3
release_date	96
revenue	6
runtime	263
spoken_languages	6
status	94
vote_average	6
vote_count	6
dtype: int64	

The most important thing in the data cleaning step is to deal with the null values. As we can see from the figure above, this dataset contains null values in most of all the attributes and some of the attributes have a very high number of null values.

```
[ ] #recheck null
movieNew.isnull().sum()
```

```
movieId      0
userId       0
rating       0
movie_title   0
budget       0
genres_y     0
imdb_id      0
original_language  0
popularity   0
production_companies  0
production_countries  0
release_date  0
revenue      0
runtime      0
spoken_languages  0
status       0
vote_average  0
vote_count   0
dtype: int64
```

With the large number of data in this dataset which is 127,283, it is safe to drop all rows that contain null values because even after dropping the null values we will still have a majority number of dataset.

```
#checking duplicate
movieNew[movieNew.duplicated(keep=False)]
```

```
movieId  userId  rating  movie_title  budget  genres_y  imdb_id  original_language  popularity  production_companies  production_countries  release_da
```

Next, check the duplicated data in the dataset, and as we can see from the figure above, there is no data duplicated data in this dataset.

```
#write to csv
from csv import writer
movieNew.to_csv('Movies Clean.csv', index=False)
```

The data that has been cleaned will then be saved into a new csv file to process with the analysis of the data.

3.4 Movies Analysis

```
[ ] print('New no of attributes: ', len(movieNew.columns))
    print('New no of rows: ', len(movieNew))
```

```
New no of attributes: 18
New no of rows: 89264
```

The new dataset after the cleaning process consists of 18 attributes and 89,264 rows of data.

```
movies.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89264 entries, 0 to 89263
Data columns (total 18 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   movieId                               89264 non-null  float64
1   userId                               89264 non-null  float64
2   rating                               89264 non-null  float64
3   movie_title                           89264 non-null  object
4   budget                               89264 non-null  int64
5   genres_y                             89264 non-null  object
6   imdb_id                              89264 non-null  object
7   original_language                    89264 non-null  object
8   popularity                           89264 non-null  float64
9   production_companies                 89264 non-null  object
10  production_countries                 89264 non-null  object
11  release_date                         89264 non-null  object
12  revenue                              89264 non-null  float64
13  runtime                             89264 non-null  float64
14  spoken_languages                     89264 non-null  object
15  status                               89264 non-null  object
16  vote_average                         89264 non-null  float64
17  vote_count                           89264 non-null  float64
dtypes: float64(8), int64(1), object(9)
memory usage: 12.3+ MB
```

Based on the above output, there are 18 attributes in the dataset. There are 8 attributes that come from the float data type which are 'movieId', 'userId', 'rating', 'popularity', 'revenue', 'runtime', 'vote_average' and 'vote_count'. 9 attributes comes from the object data type which are 'movie_title', 'genres_y', 'imdb_id', 'original_language', 'production_companies', 'production_countries', 'release_date', 'spoken_languages' and 'status' while another 1 attribute is 'budget'.

Before moving to the modelling of the movie recommendation system, analysis of the movies should be done to see the top movies based on the rating, popularity, and genres of the movies. Figure below shows the libraries that are needed in order to make the analysis and the dataset will also be imported into Python.

```
#Libraries
import numpy as np
import pandas as pd
from ast import literal_eval
```

The use of import literal_eval library in the figure above is to separated genres of the movie into different rows. The process of separating the genres are shown below:

```
#separated the genres
movies['genres'] = movies['genres_y'].fillna('').apply(literal_eval).apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else x)
```

Result:

Before

```
movies['genres_y']
0      [{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 28, 'name': 'Drama'}]
1      [{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 28, 'name': 'Drama'}]
2      [{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 28, 'name': 'Drama'}]
3      [{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 28, 'name': 'Drama'}]
4      [{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 28, 'name': 'Drama'}]
...
89259  [{'id': 18, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}]
89260  [{'id': 18, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}]
89261  [{'id': 18, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}]
89262  [{'id': 18, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}]
89263  [{'id': 18, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}, {'id': 28, 'name': 'Drama'}]
Name: genres_y, Length: 89264, dtype: object
```



After

```
movies['genres']
0      [Animation, Comedy, Family]
1      [Animation, Comedy, Family]
2      [Animation, Comedy, Family]
3      [Animation, Comedy, Family]
4      [Animation, Comedy, Family]
...
89259  [Drama, Action, Romance]
89260  [Drama, Action, Romance]
89261  [Drama, Action, Romance]
89262  [Drama, Action, Romance]
89263  [Drama, Action, Romance]
Name: genres, Length: 89264, dtype: object
```

Identify the quality of movies to be on top chart:

In order to make sure that a movie is eligible to be on the top chart, it needs to have certain votes and also the average ratings.

```
#find vote count and average
vote_counts = movies[movies['vote_count'].notnull()]['vote_count'].astype('int')
vote_averages = movies[movies['vote_average'].notnull()]['vote_average'].astype('int')
C = vote_averages.mean()
C
6.175871571966302
```



```
#find 95th percentile
m = vote_counts.quantile(0.95)
m
```

5091.0

```
#find qualified data to be on the cart
qualified = movies[(movies['vote_count'] >= m) & (movies['vote_count'].notnull()) & (movies['vote_average'].notnull())][['movie_title', 'year', 'vote_count', 'vote_average', 'popularity', 'genres', 'rating']]
qualified['vote_count'] = qualified['vote_count'].astype('int')
qualified['vote_average'] = qualified['vote_average'].astype('int')
qualified.shape
```

(4483, 7)

Based on the above outputs, for a movie to pass the qualifier to be considered for the chart, a movie has to have at least 5091 votes and average rating of 6.1758. Hence, from this dataset, 4483 movies qualify to be on the top chart.

```
#remove the duplicated data and keep first
qualifiedMovies.drop_duplicates(keep='first', inplace=True)
qualifiedMovies[qualifiedMovies.duplicated(keep=False)]
```

	movie_title	year	vote_count	vote_average	popularity	genres	rating
--	-------------	------	------------	--------------	------------	--------	--------

After the movies that are eligible to be on the top chart have been decided, duplicated rows should also be checked in order to make an accurate analysis. Based on the figure below, there exist duplicated rows in the dataset and the duplicated rows has been removed.

```
qualifiedMovies = qualifiedMovies.sort_values(['rating', 'popularity', 'vote_average'], ascending=False)
qualifiedMovies.head(10)
```

	movie_title	year	vote_count	vote_average	popularity	genres	rating
4303	Beauty and the Beast	2017	5530	6	287.253654	['Family', 'Fantasy', 'Romance']	5.0
4119	Big Hero 6	2014	6289	7	213.849907	['Adventure', 'Family', 'Animation', 'Action', ...]	5.0
3308	Avatar	2009	12114	7	185.070892	['Action', 'Adventure', 'Fantasy', 'Science Fi...]	5.0
4050	Gone Girl	2014	6023	7	154.801009	['Mystery', 'Thriller', 'Drama']	5.0
4175	The Hunger Games: Mockingjay - Part 1	2014	5767	6	147.098006	['Science Fiction', 'Adventure', 'Thriller']	5.0
249	Pulp Fiction	1994	8670	8	140.950236	['Thriller', 'Crime']	5.0
1505	Fight Club	1999	9678	8	63.869599	['Drama']	5.0
4075	Guardians of the Galaxy	2014	10014	7	53.291601	['Action', 'Science Fiction', 'Adventure']	5.0
572	Forrest Gump	1994	8147	8	48.307194	['Comedy', 'Drama', 'Romance']	5.0
2364	Pirates of the Caribbean: The Curse of the Bla...	2003	7191	7	47.326665	['Adventure', 'Fantasy', 'Action']	5.0

Thus, top movies based on rating, popularity and vote could be decided based on the dataset. Based on the figure above, the top movie based on rating, popularity, and vote is Beauty and the Beast that released in the year of 2017 with the highest popularity which is 287.2535 followed by Big Hero 6 and Avatar.

Aside from deciding the top movies based on rating, popularity and vote, the top movies could also be decided based on the genres. In order to do that, the genres should be separated into different rows as shown below.

```
#separated genres
s = movies.apply(lambda x: pd.Series(x['genres']),axis=1).stack().reset_index(level=1, drop=True)
s.name = 'genre'
gen_md = movies.drop('genres', axis=1).join(s)
```

Result:

<i>Before</i>		<i>After</i>	
movies['genres']		gen_md['genre']	
0	[Animation, Comedy, Family]	0	Animation
1	[Animation, Comedy, Family]	0	Comedy
2	[Animation, Comedy, Family]	0	Family
3	[Animation, Comedy, Family]	1	Animation
4	[Animation, Comedy, Family]	1	Comedy
...		...	
89259	[Drama, Action, Romance]	89262	Action
89260	[Drama, Action, Romance]	89262	Romance
89261	[Drama, Action, Romance]	89263	Drama
89262	[Drama, Action, Romance]	89263	Action
89263	[Drama, Action, Romance]	89263	Romance
Name: genres, Length: 89264, dtype: object		Name: genre, Length:	

As shown in the figure below, the build genre chart will be defined based on the vote count and average vote to see each genre.

```
def build_chart(genre, percentile=0.85):
    qualified = gen_md[(gen_md['vote_count'] >= m) & (gen_md['vote_count'].notnull()) & (gen_md['vote_average'].notnull())[['movie_id', 'vote_count', 'vote_average']]
    qualified['vote_count'] = qualified['vote_count'].astype('int')
    qualified['vote_average'] = qualified['vote_average'].astype('int')

    return qualified

#write to csv
from csv import writer
qualified.to_csv('Build Genre Chart.csv', index=False)
```

```
#Top movies based on genre comedy
```

```
genreChart[genreChart['genre'] == 'Comedy'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
11887	Big Hero 6	2014	6289	7	213.849907	Comedy	5.0
1392	Forrest Gump	1994	8147	8	48.307194	Comedy	5.0
12185	Kingsman: The Secret Service	2015	6069	7	28.224212	Comedy	5.0

As shown in the figure above, the genre chart will then be sorted according to their genre based on the rating, popularity and votes. According to the output, the top comedy movie according to our rating and popularity is Big Hero 6 with 5 rate and 213.85 popularity followed by Forrest Gump and Kingsman: The Secret Service.

3.5 Modelling

Figure below shows the library that is needed for modelling.

```
#Libraries
import numpy as np
import pandas as pd
from tqdm import tqdm
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, precision_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
```

For the data that need to be used in modelling, we only choose the user id and the movie title that they have watch to create the recommendation system algorithm for a movie based on users watch history.

```
model_data = movies[['userId', 'movie_title']]
```

The dataset will then be encoded in order to prepare the data into numerical form to make the analysis of clustering become easier.

```
encoded_df = pd.crosstab(model_data["userId"], model_data["movie_title"])

#display
encoded_df.head()
```

movie_title	\$9.99	'Neath the Arizona Skies	(500) Days of Summer	...And Justice for All	1-900	10	10 Attitudes	10 Items or Less	10 Things I Hate About You	10 Years	...	Zoot Suit	Zorba the Greek	Zulu	[REC]	eXistenZ	loudQUIETloud: A Film About the Pixies	xXx	xXx: State of the Union
userId																			
1.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3.0	0	0	0	0	0	0	0	0	0	0	...	0	0	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
5.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 6350 columns

3.5.1 K-Means Clustering

K-Means Clustering recommendation will be utilized in this movie recommendation system. It helps to list out the movies in each of the clusters. The use of principal Component Analysis (PCA) as a preprocessing step before applying the k-mean clustering algorithm. PCA helps to simplify the complexity in high-dimensional data while retaining trends and patterns. This will help to improve the performance and efficiency of k-means, as well as to make it easier to visualize the cluster in the data.

```
pca = PCA(n_components=3)
pca.fit(encoded_df)
pca_samples = pca.transform(encoded_df)
```

The clusters of the movies need to be created in order to decide which movies are the most watched movies based on each cluster.

```
clusterer = KMeans(n_clusters=20, random_state=30, n_init=30).fit(pca_samples)
centers = clusterer.cluster_centers_
c_preds = clusterer.predict(pca_samples)
```

Based on the figure below, if the users watch Titanic movie then the system will recommend to the users to watch Ghost, Armageddon, Ghostbusters, Forrest Gump, and Ocean's Eleven because users who watch Titanic movie, most of them watch the movies in order.

```
def recommend_movies(movie_name):
    cluster_number = encoded_df.groupby("cluster").sum()[movie_name].argmax()
    columns = ["cluster"]+[movie_name]
    recommended_movies = encoded_df[encoded_df.cluster==cluster_number]
    recommended_movies = list(recommended_movies.drop(columns,axis=1).sum().sort_values(ascending=False)[:5].keys())
    print(recommended_movies)

movieName = input()
recommend_movies(movieName)

Titanic
['Ghost', 'Armageddon', 'Ghostbusters', 'Forrest Gump', 'Ocean's Eleven']
```

3.5.2 Association Rules

Association rules are also utilized in this movie recommendation system. It helps to predict what are the probability for other movies user will watch if they watch certain movies.

```
trans_encoder = TransactionEncoder()
trans_data = trans_encoder.fit(records).transform(records)
trans_df = pd.DataFrame(trans_data, columns=trans_encoder.columns_)
trans_df = trans_df.replace({False:0})
trans_df.head()
```

	\$9.99	'Neath the Arizona Skies	(500) Days of Summer	...And Justice for All	1-900	10	10 Attitudes	10 Items or Less	10 Things I Hate About You	10 Years	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...

5 rows × 6452 columns

The figure above shows that the data has transfrom to make the movie title as columns and each of the user id as a row and count the watched movies as 1 and 0 for represent the movie that didn't watch yet.

```
trans_df = apriori(trans_df, min_support = 0.2, use_colnames = True)
trans_df
```

	support	itemsets
0	0.260805	(Ace Ventura: Pet Detective)
1	0.320417	(AladdinAladdin)
2	0.327869	(American Beauty)
3	0.298063	(Apollo 13)
4	0.220566	(Babe)
...
118	0.225037	(Jurassic Park, Forrest Gump, Terminator 2: Ju...
119	0.208644	(True Lies, Jurassic Park, Forrest Gump)
120	0.213115	(Pulp Fiction, Schindler's List, Forrest Gump)
121	0.210134	(Pulp Fiction, Forrest Gump, Seven)
122	0.208644	(Pulp Fiction, Forrest Gump, Terminator 2: Jud...

123 rows × 2 columns

The minimum support of 0.2 has been used to determine which movie will be considered as ‘frequent’ to be included in the final set of association rules.

```
df_ar = association_rules(trans_df, metric = "confidence", min_threshold = 0.5)
df_ar
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Ace Ventura: Pet Detective)	(Forrest Gump)	0.260805	0.508197	0.213115	0.817143	1.607926	0.080575	2.689549
1	(Jurassic Park)	(Ace Ventura: Pet Detective)	0.408346	0.260805	0.205663	0.503650	1.931137	0.099165	1.489261
2	(Ace Ventura: Pet Detective)	(Jurassic Park)	0.260805	0.408346	0.205663	0.788571	1.931137	0.099165	2.798365
3	(Ace Ventura: Pet Detective)	(Pulp Fiction)	0.260805	0.482861	0.208644	0.800000	1.656790	0.082711	2.585693
4	(AladdinAladdin)	(Beauty and the BeastBeauty and the BeastBeaut...	0.320417	0.269747	0.213115	0.665116	2.465707	0.126683	2.180618
...
139	(Seven)	(Pulp Fiction, Forrest Gump)	0.299553	0.344262	0.210134	0.701493	2.037669	0.107009	2.196721
140	(Pulp Fiction, Terminator 2: Judgment Day)	(Forrest Gump)	0.241431	0.508197	0.208644	0.864198	1.700518	0.085950	3.621461
141	(Pulp Fiction, Forrest Gump)	(Terminator 2: Judgment Day)	0.344262	0.353204	0.208644	0.606061	1.715893	0.087049	1.641866
142	(Terminator 2: Judgment Day, Forrest Gump)	(Pulp Fiction)	0.274218	0.482861	0.208644	0.760870	1.575751	0.076235	2.162580
143	(Terminator 2: Judgment Day)	(Pulp Fiction, Forrest Gump)	0.353204	0.344262	0.208644	0.590717	1.715893	0.087049	1.602163

The minimum threshold of association rules has been set to 0.5. In other words, this mean when the movie X is watched, we can say that the probability of the user to watch movie Y is 50% or more.

df_ar.loc[13]	
antecedents	(Apollo 13)
consequents	(Jurassic Park)
antecedent support	0.298063
consequent support	0.408346
support	0.213115
confidence	0.715
lift	1.750967
leverage	0.091402
conviction	2.07598
Name: 13, dtype: object	

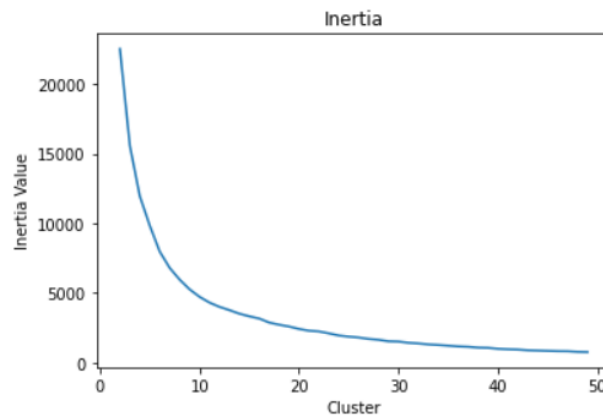
Based on the association rules above, 30% of people watched Apollo 13 (antecedent support) while 41% of people watched Jurassic Park (consequent support). 21% of people watched both of them (support). 72% of people who watched Apollo 13 also watched Jurassic Park (confidence). The people who watched them both is 9% more than who watched them separately (leverage). The rate of the movies related each other is 2.08 (conviction).

3.6 Model Evaluation

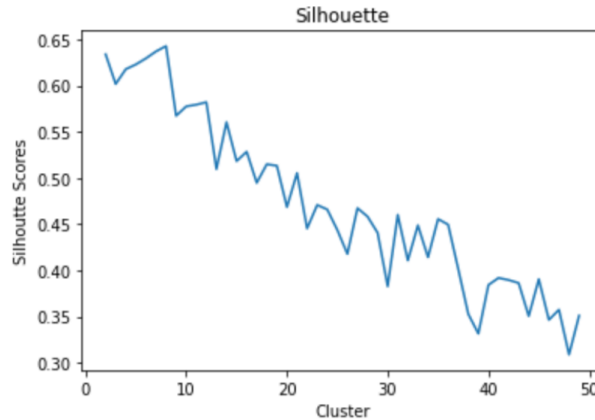
The evaluation of model is an important step to see the performance of the model that has been develop. The model evaluation of this project has been done as below:

3.6.1 K-Means Clustering

The performance of the k-means clustering model have been done using the inertia and silhouette score of k-means. Inertia is the sum of squared distances of samples to their closest cluster center. It help to measures the compactness of the clusters. A lower value of inertia indicates that the clusters are more tightly packed together. Meanwhile, the silhouette score is to measure how similar an object is to its own cluster compared to other clusters. A higher silhouette score indicates that the clusters are more well-defined.



As we can see from the inertia graph above, the elbow point of inertia graph which is (9,5000) is consider as the the optimal number of clusters. As the number of cluster increase, the value of inertia will decrease as the clusters become more compact and this will shows that the clustering model is well-defined.



The range of the silhouette score need to be in the range of -1 to 1 to say that the model is accurate. As we can see from the silhouette graph above, range of silhouette score in this project are between 0.3 to 0.65. Hence, the k-means clustering model is well defined.

3.6.2 Association Rules

There are several methods for evaluating the strength of association rules for movie recommendations, it is important to consider the following evaluation metrics:

1. Support: To see the proportion of users who have watched the movie in the dataset.
2. Confidence: To see proportion of users who watched the antecedent movie also watched the consequent movie.
3. Lift: The ratio of the observed support to the expected support if the antecedent and consequent movies were independent.
4. Coverage: The proportion of users covered by the rule.

Support and Confidence are commonly used evaluation metrics for association rules in general. However, Lift is an important metric for evaluating the relevance and strength of the rule for movie recommendations. The Coverage helps to evaluate how many users are covered by the rule, it can be used to measure the scalability of the recommendation system.

4.0 RESULT AND DISCUSSION

4.1 Results

Top movies based on genre category:

```
#Top movies based on genre romance  
genreChart[genreChart['genre'] == 'Family'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
84646	Minions	2015	4729.0	6.4	547.488298	Family	5.0
88253	Beauty and the Beast	2017	5530.0	6.8	287.253654	Family	5.0
81711	Big Hero 6	2014	6289.0	7.8	213.849907	Family	5.0

For the category 'Family', the top three movies are Minions, Beauty and the Beast, and Big Hero 6.

```
#Top movies based on genre romance  
genreChart[genreChart['genre'] == 'Animation'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
84646	Minions	2015	4729.0	6.4	547.488298	Animation	5.0
81711	Big Hero 6	2014	6289.0	7.8	213.849907	Animation	5.0
57594	Spirited Away	2001	3968.0	8.3	41.048867	Animation	5.0

For the category 'Animation', the top three movies are Minions, Big Hero 6, and Spirited Away.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'Adventure'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
84646	Minions	2015	4729.0	6.4	547.488298	Adventure	5.0
81711	Big Hero 6	2014	6289.0	7.8	213.849907	Adventure	5.0
74309	Avatar	2009	12114.0	7.2	185.070892	Adventure	5.0

For the category 'Adventure', the top three movies are Minions, Big Hero 6, and Avatar.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'Fantasy'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
88253	Beauty and the Beast	2017	5530.0	6.8	287.253654	Fantasy	5.0
74309	Avatar	2009	12114.0	7.2	185.070892	Fantasy	5.0
59966	Pirates of the Caribbean: The Curse of the Bla...	2003	7191.0	7.5	47.326665	Fantasy	5.0

For the category 'Fantasy', the top three movies are Beauty and the Beast, Avatar, and Pirates of the Caribbean.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'Romance'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
88253	Beauty and the Beast	2017	5530.0	6.8	287.253654	Romance	5.0
8372	Forrest Gump	1994	8147.0	8.2	48.307194	Romance	5.0
29796	Titanic	1997	7770.0	7.5	26.889070	Romance	5.0

For the category 'Romance', the top three movies are Beauty and the Beast, Forrest Gump, and Titanic.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'Science Fiction'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
74309	Avatar	2009	12114.0	7.2	185.070892	Science Fiction	5.0
81880	The Hunger Games: Mockingjay - Part 1	2014	5767.0	6.6	147.098006	Science Fiction	5.0
12203	Blade Runner	1982	3833.0	7.9	96.272374	Science Fiction	5.0

For the category 'Science Fiction', the top three movies are Avatar, The Hunger Games, and Blade Runner.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'Mystery'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
81349	Gone Girl	2014	6023.0	7.9	154.801009	Mystery	5.0
75315	Inception	2010	14075.0	8.1	29.108149	Mystery	5.0
65072	Saw	2004	2255.0	7.2	23.508433	Mystery	5.0

For the category 'Mystery', the top three movies are Gone Girl, Inception, and Saw.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'Thriller'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
81349	Gone Girl	2014	6023.0	7.9	154.801009	Thriller	5.0
81880	The Hunger Games: Mockingjay - Part 1	2014	5767.0	6.6	147.098006	Thriller	5.0
6758	Pulp Fiction	1994	8670.0	8.3	140.950236	Thriller	5.0

For the category 'Thriller', the top three movies are Gone Girl, The Hunger Games, and Pulp Fiction.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'War'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
11669	Schindler's List	1993	4436.0	8.3	41.725123	War	5.0
81648	Fury	2014	4028.0	7.4	36.713807	War	5.0
81847	The Imitation Game	2014	5895.0	8.0	31.595940	War	5.0

For the category 'War', the top three movies are Schindler's List, Fury, and The Imitation Game.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'Horror'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
22054	Psycho	1960	2405.0	8.3	36.826309	Horror	5.0
65072	Saw	2004	2255.0	7.2	23.508433	Horror	5.0
21849	Alien	1979	4564.0	7.9	23.377420	Horror	5.0

For the category 'Horror', the top three movies are Psycho, Saw, and Alien.

```
#Top movies based on genre romance
genreChart[genreChart['genre'] == 'Music'].head(3)
```

	movie_title	year	vote_count	vote_average	popularity	genre	rating
13844	Beauty and the Beast	1991	3029.0	7.5	23.433511	Music	5.0
60770	School of Rock	2003	1454.0	6.8	17.951420	Music	5.0
33717	Peter Pan	1953	1380.0	7.0	16.861533	Music	5.0

For the category 'Music', the top three movies are Beauty and the Beast, School of Rock, and Peter Pan.

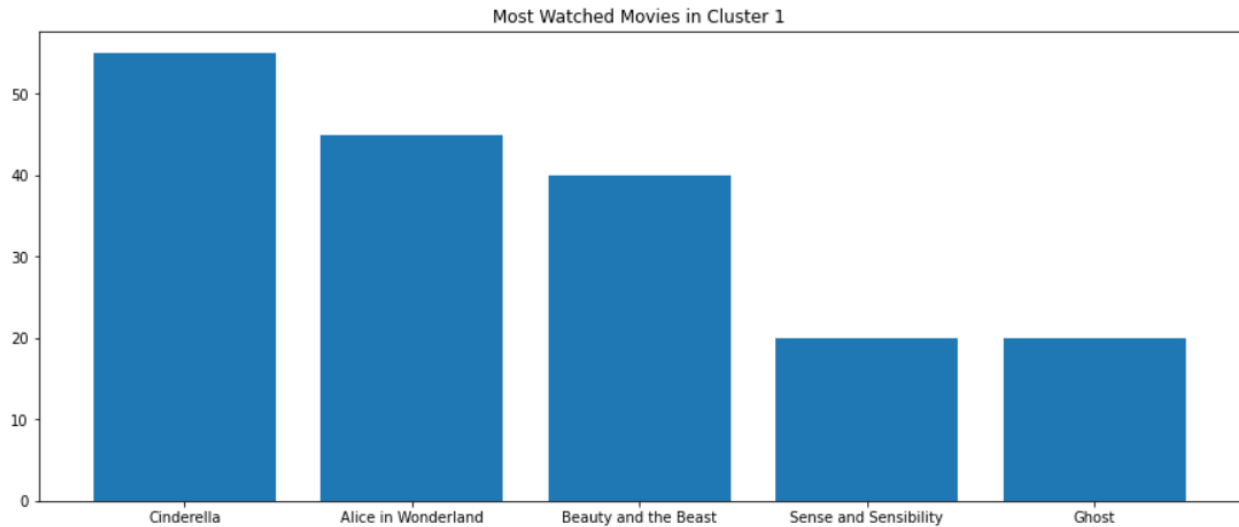
```
movieName = input()  
recommend_movies(movieName)
```

```
Beauty and the Beast  
['Aladdin', 'Ghost', 'Batman', 'Dances with Wolves', 'Pulp Fiction']
```

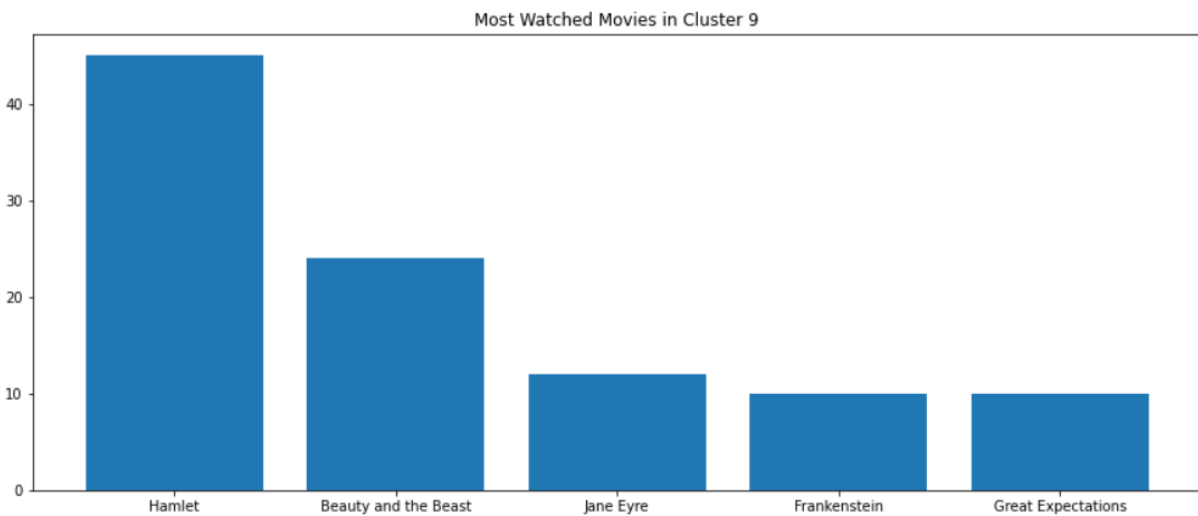
We also see the other set of recommendation based on Beauty and the Beast movie. The result are Aladdin, Ghost, Batman, Dances with Wolves, and Pulp Fiction to be watch for the user that watched Beauty and the Beast.

4.2 Visualization

For this study, we create a few visualization to see which movies are most watched in cluster. Based on the figure below, the most watched movies in Cluster 1 are Cinderella, followed by Alice in Wonderland, Beauty and the Beast, Sense and Sensibility and also Ghost.

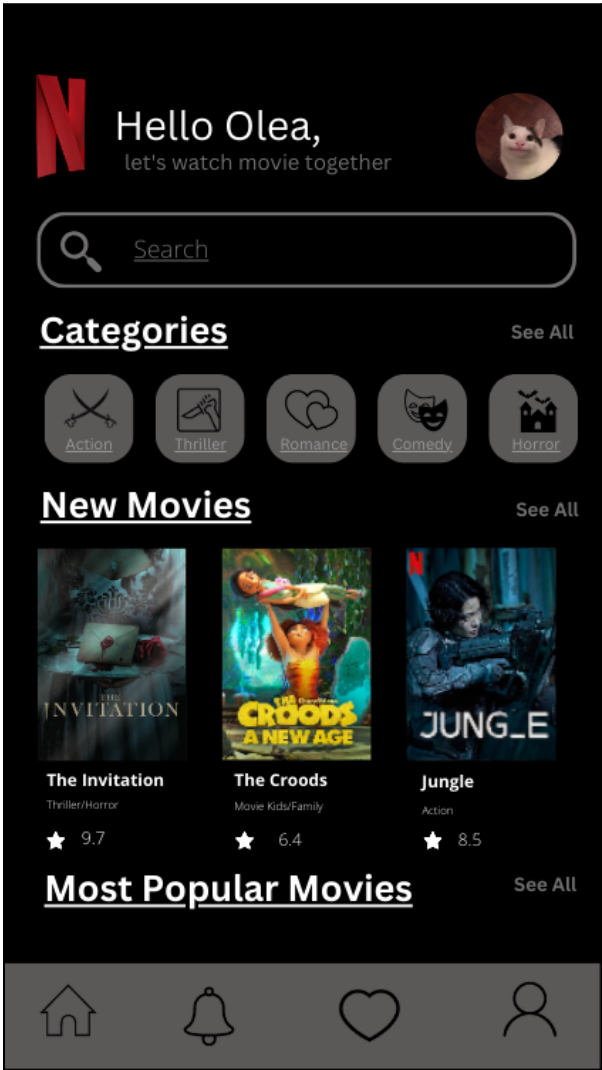



Based on the figure below, the most watched movies in Cluster 9 are Hamlet, Beauty and the Beast, Jane Eyre, Frankenstein, and Great Expectations.




4.2 GUI of Movie Recommendation










Table 4.1 shows an example of a GUI of movie recommendation. It can be used to analyze which movie is the user interested more based on their options and recommend the best movie from the outcome.

GUI	EXPLANATION
	<p>First, after login to this movie recommendation system, the user will see this home screen which has a search button and categories button. This home screen also shows the new movies and most popular movies for the first 3 movies with the Title of the movie, genre and the star recommendation.</p> <p>Users can click the “See All” button to see all the movies listed for the selected category.</p>

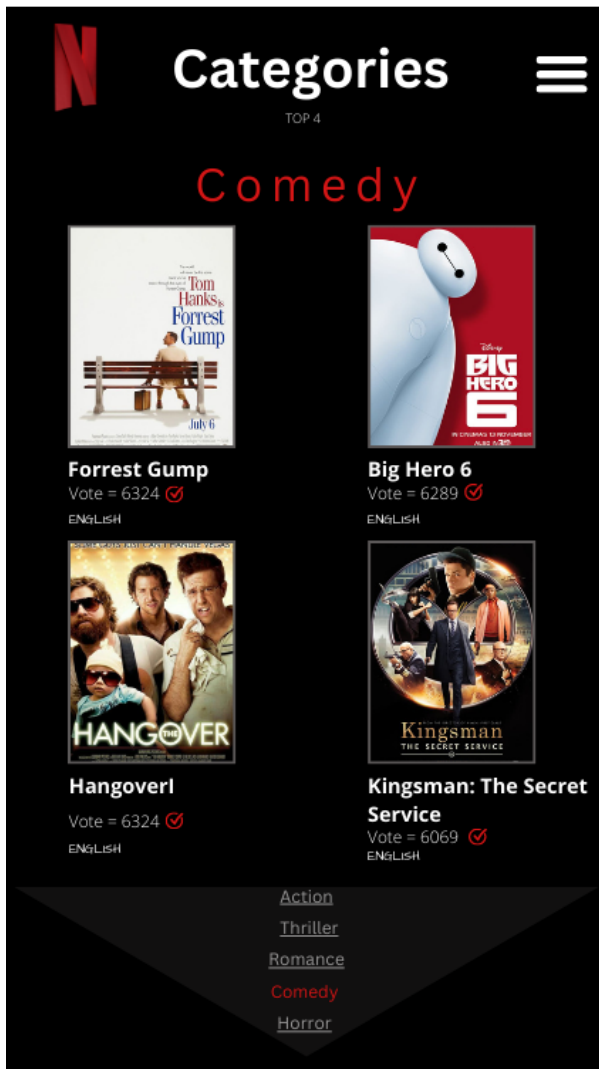


Most Popular Movies



 <p>Beauty and the Beast Family/Fantasy/Romance 🔥 287.2537</p>	 <p>Big Hero 6 Adventure/Family/Animation/Action 🔥 213.8499</p>	 <p>Avatar Adventure/Fantasy/Science Fiction/Action 🔥 185.0709</p>
 <p>Gone Girl Mystery/Thriller/Drama 🔥 154.8010</p>	 <p>The Hunger Games: Mockingjay Part I Science Fiction/Adventure/Thriller 🔥 147.0980</p>	 <p>Pulp Fiction Thriller/Crime 🔥 140.9502</p>
 <p>Fight Club Drama 🔥 63.8696</p>	 <p>Guardians Of The Galaxy Mystery/Thriller/Drama 🔥 53.2916</p>	 <p>Forrest Gump Comedy/Drama/Romance 🔥 48.3072</p>

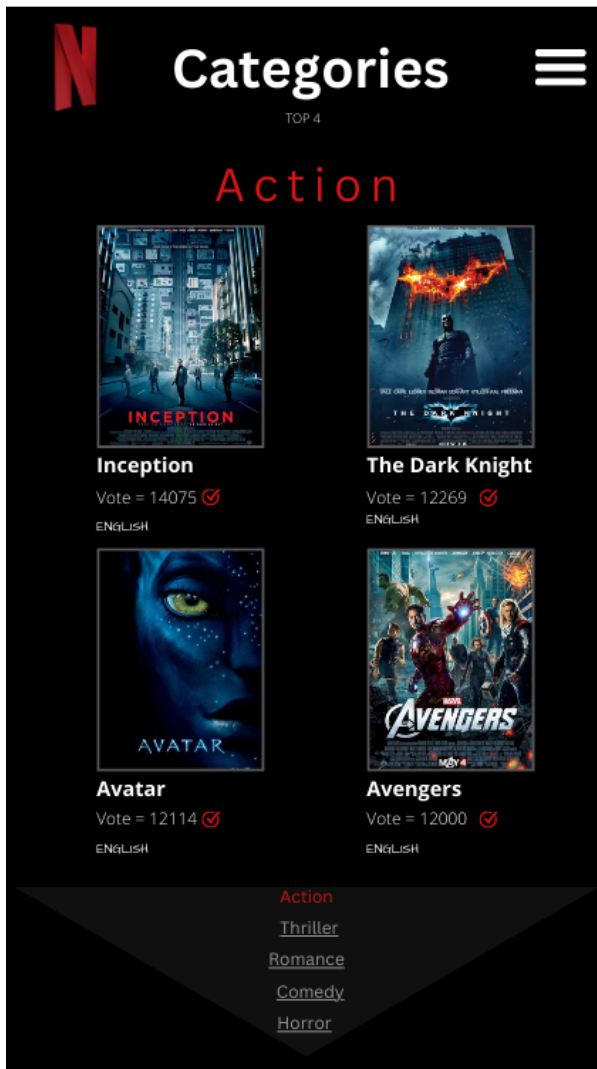
After clicking the “See All” button from “Most Popular Movies”, the screen will appear with all the listed movies for the chosen category.



If the user clicks the “Category” button from the home screen it will appear the listed movies from the selected category.

For this screen , the user clicks the “Comedy” button.

The bottom of the screen there is a button to see others' categories.



After clicking the “Action” button, the screen will now appear the listed movies for the action category.



Search Result



Titanic



Romance/Drama/Historical Drama
ENGLISH

James Cameron's "Titanic" is an epic, action-packed romance set against the ill-fated maiden voyage of the R.M.S. Titanic; the pride and joy of the White Star Line and, at the time, the largest moving object ever built. She was the most luxurious liner of her era -- the "ship of dreams" -- which ultimately carried over 1,500 people to their death in the ice cold waters of the North Atlantic in the early hours of April 15, 1912.

★ 4.8

Recommendation



Armageddon



Ghostbusters



Forrest Gump



Ocean's Eleven

For the movie recommendation, the user needs to click the search button and type any movies that the user wants to watch. After that, some movies will appear that are related to what word that the user types.

For example, users search "Titanic" and the search result is the movie itself and also appear a recommendation movie that has similar genre or vote rating at the bottom of the screen.

5.0 CONCLUSION

Based on the analysis that had been made to achieve the objectives, we can clearly see that the top movie in the dataset based on rating and popularity of the movie is Beauty and the Beast that released in the year of 2017 with 287.2535 popularity. To answer the objectives number two of this project which is top movies based on the genre, we can see from the comedy category Forrest Gump is the highest and for action, Inception is the highest.

A movie recommendation system using K-Means clustering and Association Rules can be a powerful tool for suggesting relevant and personalized movie recommendations to users. The K-Means algorithm is used to group similar movies together based on the users watched history. The Association Rules algorithm is then used to identify patterns and relationships between the movies, allowing the system to make recommendations based on the movies that users have watched or rated in the past. However, one of the limitations of this approach is that it may not be able to handle very sparse data, if the number of movies watched by users is too low the algorithm might not work properly. Additionally, it requires a large amount of data and computational resources to cluster and analyze the movies, which can be challenging for large-scale systems.

In conclusion, a movie recommendation system using K-Means and Association Rules can be a useful tool for providing personalized movie recommendations to users, but it requires a significant amount of data and computational resources to be effective. It is important to consider the trade-off between the benefits and limitations of this approach, as well as the specific requirements and constraints of the project, when deciding whether to implement a movie recommendation system using K-Means and Association Rules.

REFERENCES

1. Asad, S. M. (2020). AI Movies Recommendation Sytem Based on K-Means Clustering Algorithm. Retrieved from <https://asdkazmi.medium.com/ai-movies-recommendation-system-with-clustering-based-k-means-algorithm-f04467e02fcd#:~:text=learning%20models%20etc.,K%2DMeans%20Clustering%20Algorithm,their%204%2B%20ratings%20on%20movies.>
2. Banik, R. (2018). Movie Recommender Systems. Retrieved from <https://www.kaggle.com/code/rounakbanik/movie-recommender-systems>
3. Bury, K. (n.d.). Association Rule for movie ratings. RPubs. Retrieved from <https://rpubs.com/Bury/AssociationRuleForMovieRatings>
4. Code for Cause. (2021). ML Bootcamp - K-Means Clustering & Recommendations system Project. You Tube. Retrieved from <https://www.youtube.com/watch?v=34BBEVf94Rw>
5. Kniazieva, Y. (2022). What Is a Movie Recommendation System in ML? Retrieved from <https://labeleyourdata.com/articles/movie-recommendation-with-machine-learning#:~:text=A%20movie%20recommendation%20system%2C%20or,their%20past%20choices%20and%20behavior.>
6. Vidiyala, R. (2020). How to Build a Movie Recommendation System. Retrieve from. <https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109>