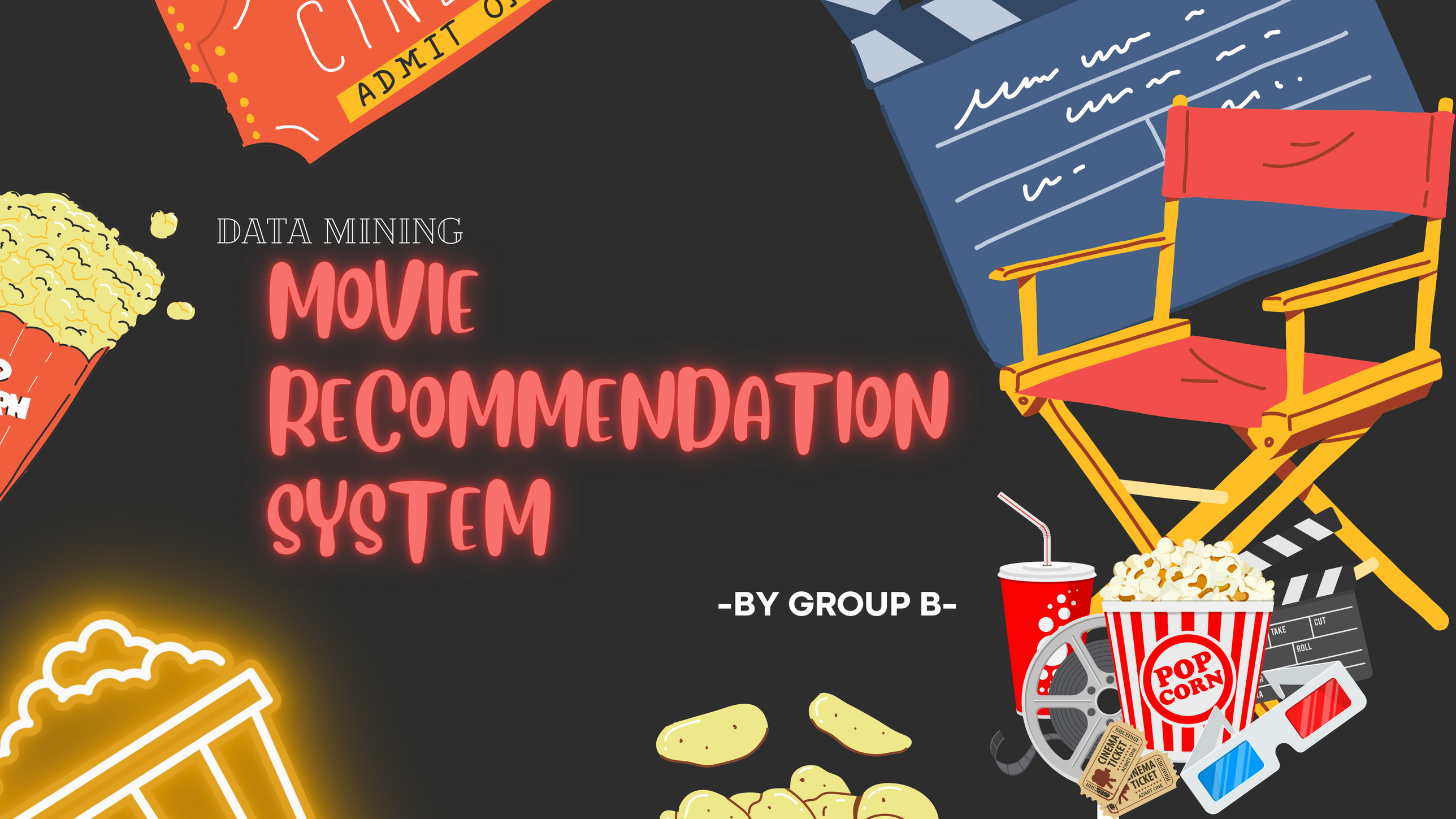


DATA MINING

MOVIE RECOMMENDATION SYSTEM

-BY GROUP B-

DESCRIPTION OF PROJECT

TYPE OF INFORMATION FILTERING SYSTEM BASED ON USER'S INTEREST.

USED IN VARIETY OF FIELD ; YOUTUBE, FACEBOOK & NETFLIX.

TIME CONSUMING PROBLEM TO FIND THE MOVIES.

THE SYSTEM WILL HELP ITS USERS TO FIND THE BEST MOVIE THAT SUITS THEIR TASTE.

K-MEANS CLUSTERING AND ASSOCIATION RULES HAS BEEN IMPLEMENTED.

THE LIST OF MOVIES THAT HAS BEEN WATCH BY USER WILL BE PUT INTO A CLUSTER.

PROBLEM TO BE SOLVED

USER DIFFICULT TO CHOOSE THE MOVIES THEY WANT TO WATCH

HARD TO NARROW DOWN THE OPTIONS

TOUGH TO FIND MOVIES THAT FITS PEOPLE TASTES & PREFERENCES

HELP TO SUGGEST MOVIES BASED ON THE USER VIEWING HISTORY AND PREFERENCES.

HELP TO DISCOVER NEW MOVIES THAT USER MAY NOT HAVE CONSIDERED

USER MAY CHOOSE TO USE FILTERS TO NARROW DOWN OPTIONS OF MOVIES BASED ON THE POPULARITY AND GENRE.

DATA SELECTED

- CSV DATA IS FROM KAGGLE
- 3 DIFFERENT DATASET HAS LINK WITH EACH OTHER ABOUT MOVIES INFORMATION HAS BEEN CHOOSE

movies_metadata

- adult
- belongs_to_collection
- budget
- genres
- homepage
- id
- imdb_id
- original_language
- original_title
- overview
- popularity
- poster_path
- production_companies
- production_countries
- release_date
- revenue
- runtime
- spoken_language
- status
- tagline
- title
- video
- vote_average
- vote_count

movies

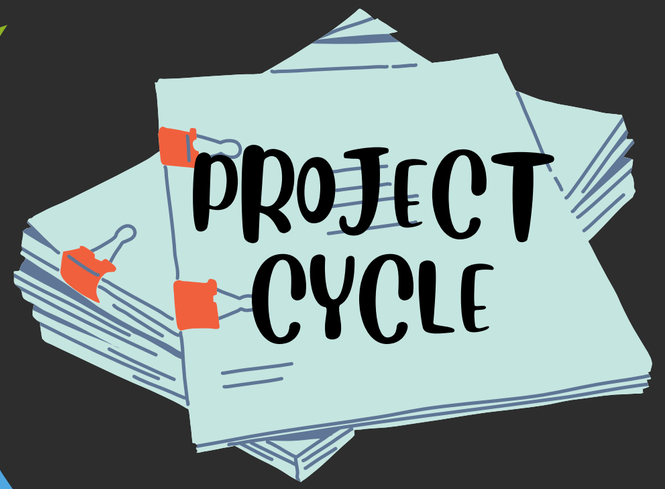
- movieId
- movie_title
- genres

rating

- userId
- movieId
- rating
- timestamp

SUMMARY

PROJECT



START/END

SYSTEM DEPLOYMENT

MODEL EVALUATION

MODEL DEVELOPMENT

MOVIES ANALYSIS

DATA PREPROCESSING

DATA PREPARATION

OBJECTIVES

01

TO SEE THE TOP MOVIES BASED ON THE RATING AND POPULARITY OF THE MOVIES

02

TO SEE THE TOP MOVIES IN EACH CATEGORY OF THE GENRES.

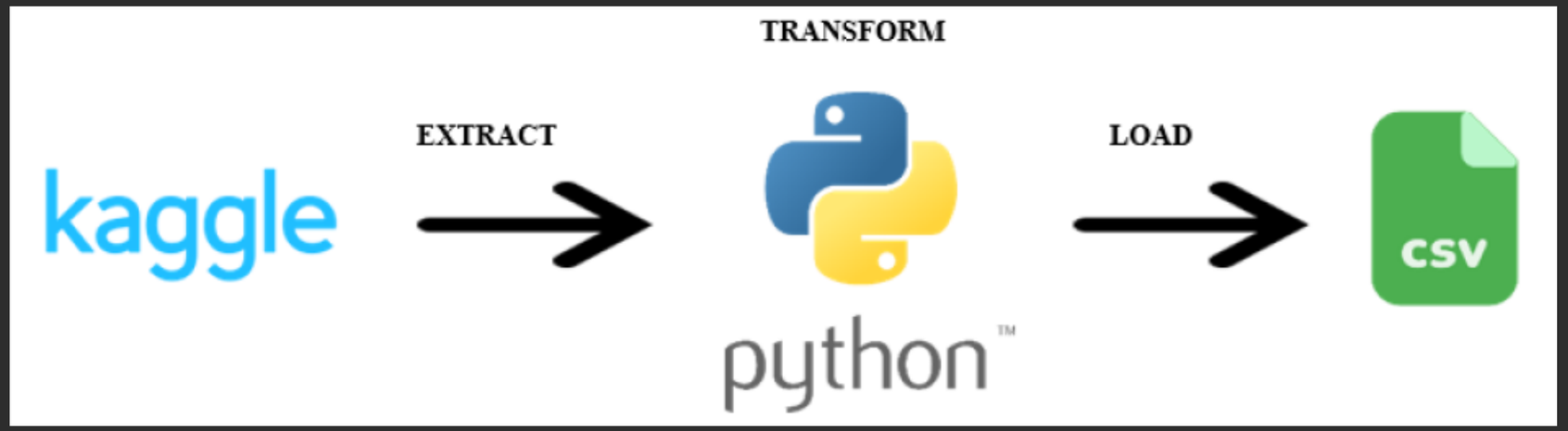
03

TO DEVELOP A MOVIES RECOMMENDATION SYSTEM BASED ON USER PREFERENCES

04

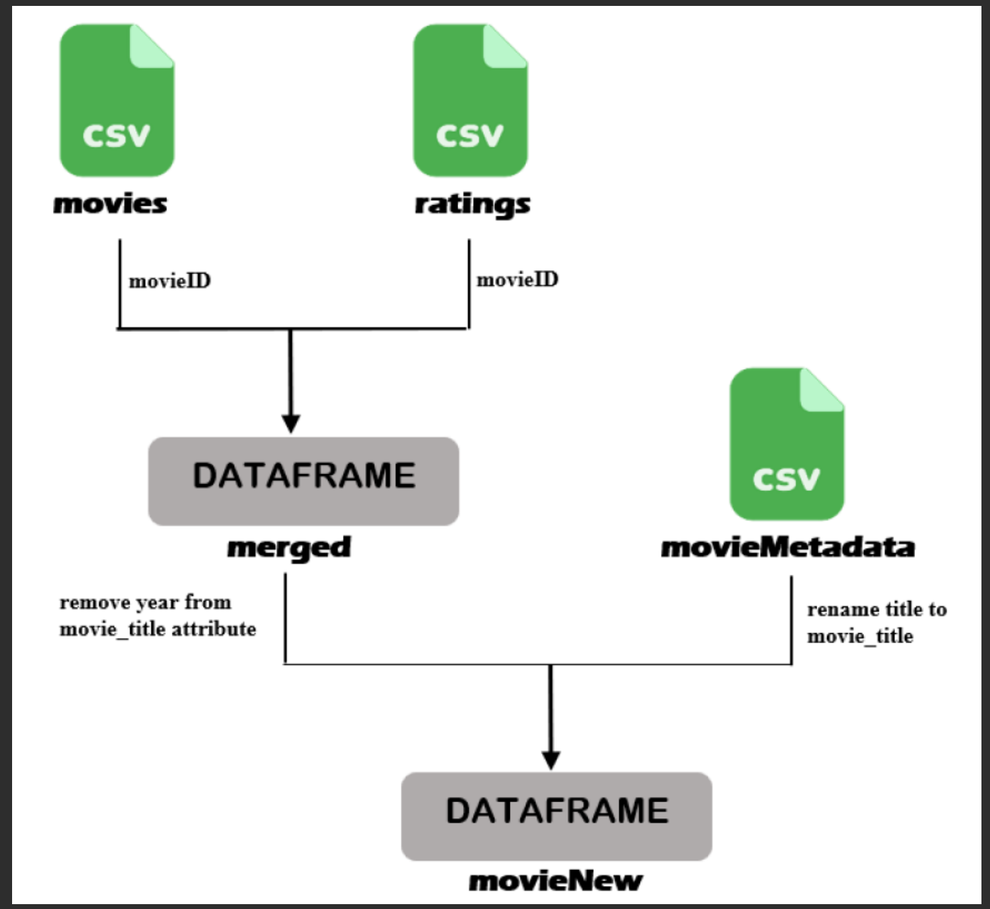
TO SEE THE PROBABILITY OF A MOVIE TO BE WATCHED AFTER A SPECIFIC MOVIE.

DATA PIPELINE



The data pipeline that has been used is ETL concept.

PROCESS OF MERGING DATASETS



CLEANING DATASET

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

```
#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
```

```
#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

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

MOVIE ANALYSIS

```
#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
```

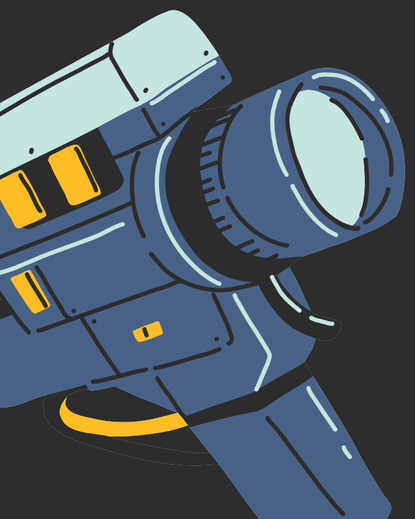
```
#find qualified data to be on the cart
qualified = movies[(movies['vote_count'] >= m) & (movies['vote_count'].notnull()) & (movies['vote_average'].notnull())][['movie_t
qualified['vote_count'] = qualified['vote_count'].astype('int')
qualified['vote_average'] = qualified['vote_average'].astype('int')
qualified.shape
```

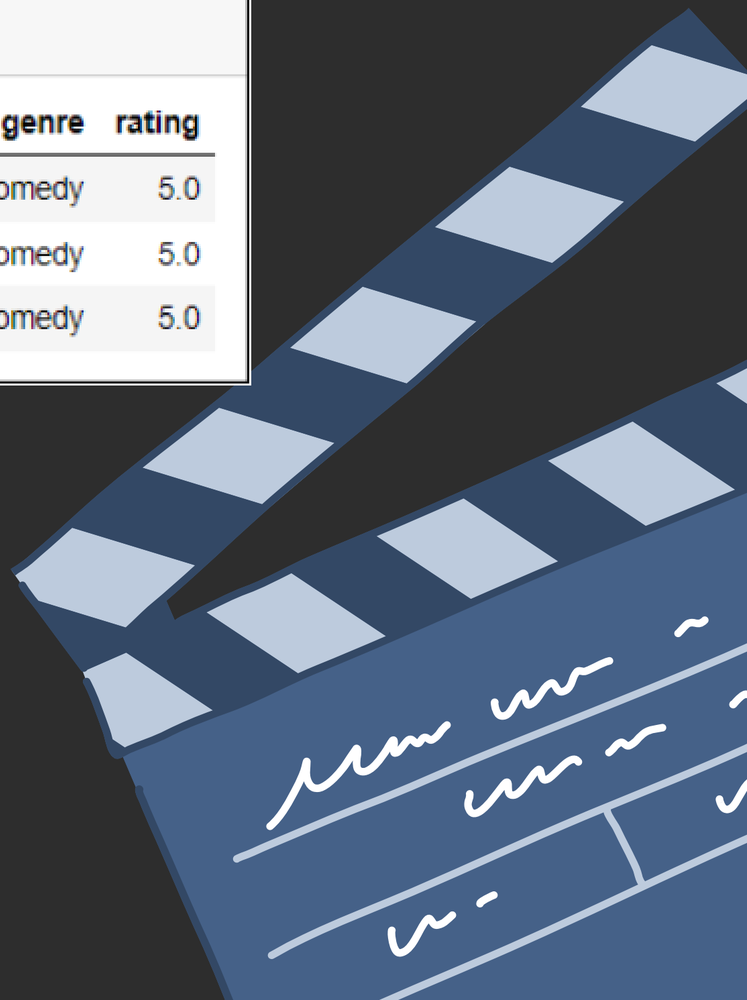
```
(4483, 7)
```

MOVIE ANALYSIS

```
#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





MODELLING

K-Means Clustering

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

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

```
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"]
```

MODELLING

Association Rules

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

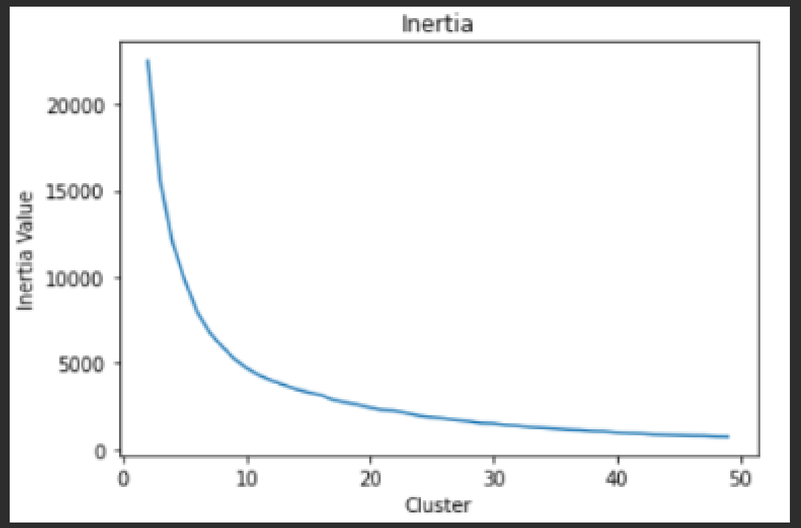
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Ace Ventura: Pet Detective)	(Forrest Gump)	0.260805	0.508197	0.213115	0.817143	1.607926	0.080575	2.68954
1	(Jurassic Park)	(Ace Ventura: Pet Detective)	0.408346	0.260805	0.205663	0.503650	1.931137	0.099165	1.48926
2	(Ace Ventura: Pet Detective)	(Jurassic Park)	0.260805	0.408346	0.205663	0.788571	1.931137	0.099165	2.79836
3	(Ace Ventura: Pet Detective)	(Pulp Fiction)	0.260805	0.482861	0.208644	0.800000	1.656790	0.082711	2.58569
4	(AladdinAladdin)	(Beauty and the BeastBeauty and the BeastBeaut...	0.320417	0.269747	0.213115	0.665116	2.465707	0.126683	2.18061
...	...	...	...	...	...	...	...	...	...
139	(Seven)	(Pulp Fiction, Forrest Gump)	0.299553	0.344262	0.210134	0.701493	2.037669	0.107009	2.19672
140	(Pulp Fiction, Terminator 2: Judgment Day)	(Forrest Gump)	0.241431	0.508197	0.208644	0.864198	1.700518	0.085950	3.62146
141	(Pulp Fiction, Forrest Gump)	(Terminator 2: Judgment Day)	0.344262	0.353204	0.208644	0.606061	1.715893	0.087049	1.64186
142	(Terminator 2: Judgment Day, Forrest Gump)	(Pulp Fiction)	0.274218	0.482861	0.208644	0.760870	1.575751	0.076235	2.16258
143	(Terminator 2: Judgment Day)	(Pulp Fiction, Forrest Gump)	0.353204	0.344262	0.208644	0.590717	1.715893	0.087049	1.60216

```
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
```

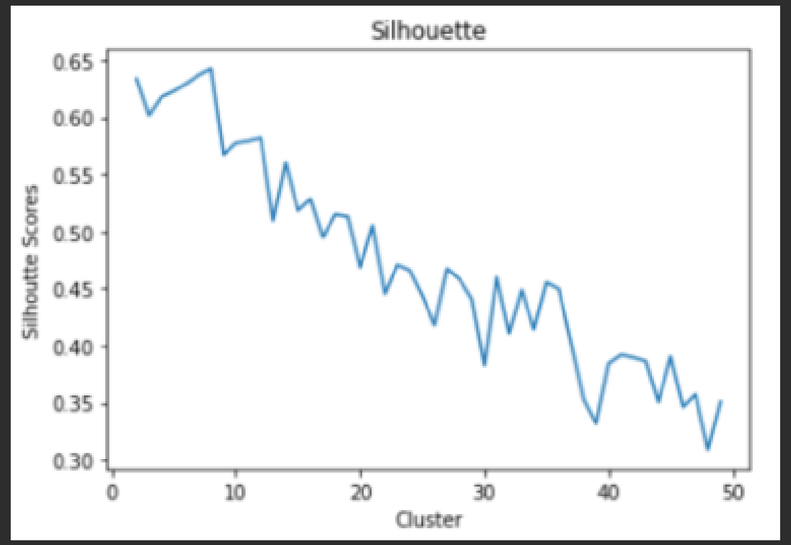
MODEL EVALUATION

K-Means Clustering



Elbow point of inertia graph which is (9,5000) - optimal number of clusters.

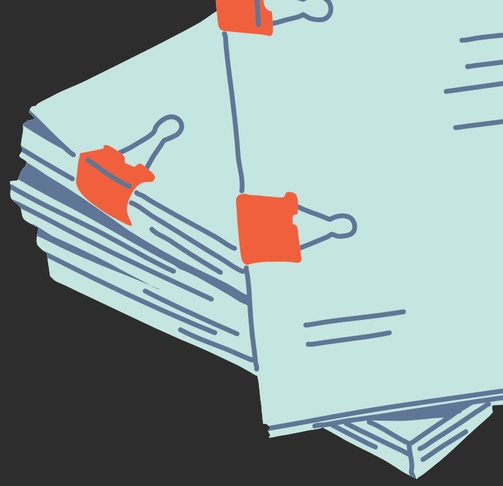
As the number of cluster increase, the value of inertia will decrease as the clusters



Silhouette score in this project are between 0.3 to 0.65.

K-Means clustering model is well defined.

MODEL EVALUATION



Association Rules

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: Evaluating the relevance and strength of the rule for movie recommendations.**
4. **Coverage: Evaluate how many users are covered by the rule.**

RESULT

```
#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

Family Category

```
#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

Animation Category

```
#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

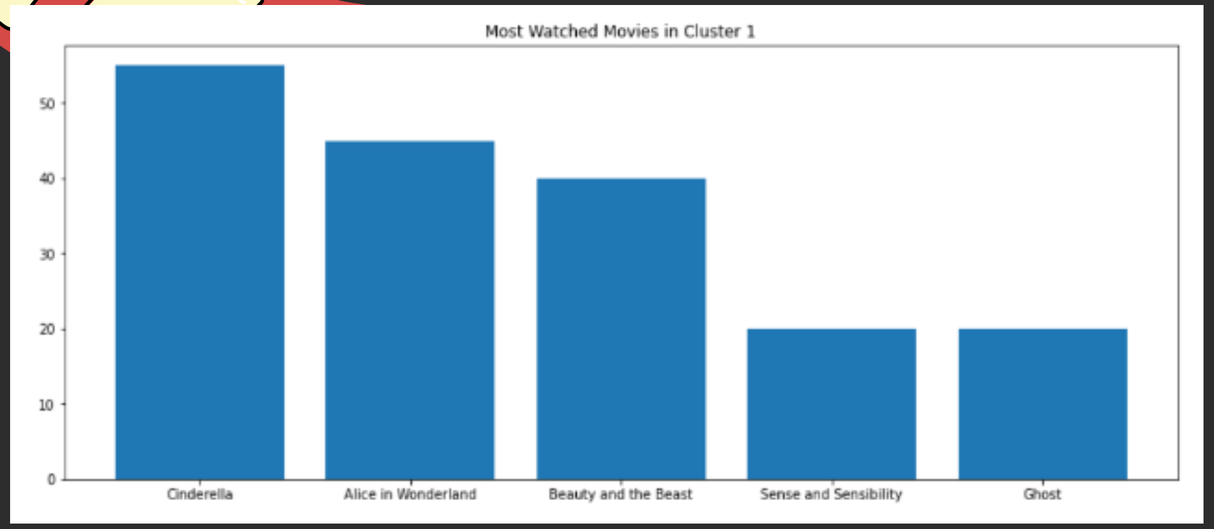
Adventure Category

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

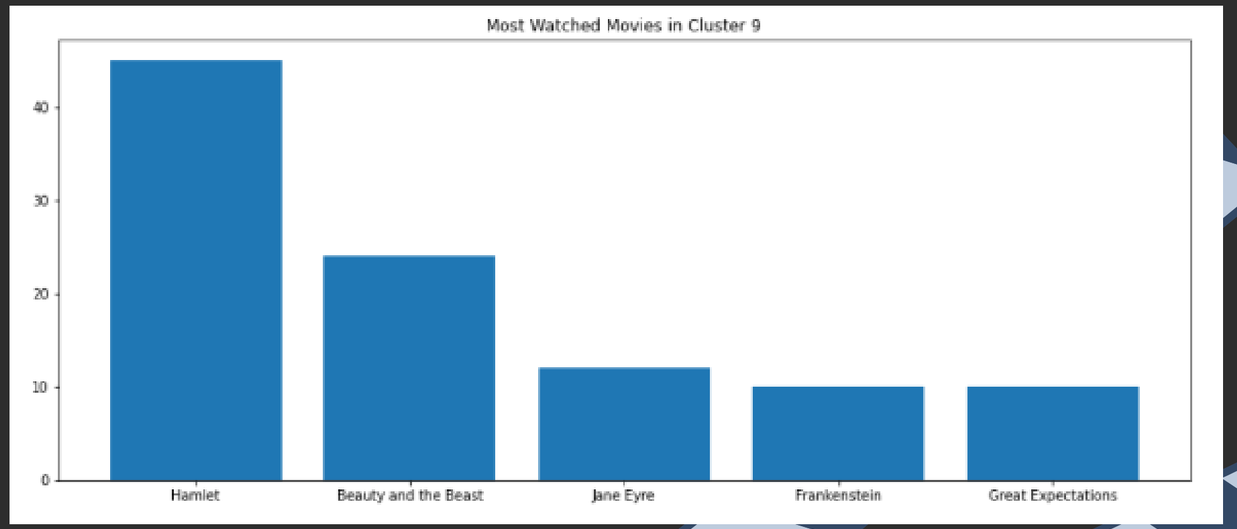
	movie_title	year	vote_count	vote_average	popularity	genre	rating
22054	Psycho	1960	2405.0	8.3	36.826309	Horror	5.0
65072	Saw	2004	2255.0	7.2	23.508433	Horror	5.0
21849	Alien	1979	4564.0	7.9	23.377420	Horror	5.0

Horror Category

VISUALIZATION



Cluster 1



Cluster 9

GUI PART : SIMULATION IN CANVA

CONCLUSION

ACHIEVE THE OBJECTIVES:

1. Can clearly see that the top movie in the dataset based on rating and popularity of the movie

2. Can see the top movies by genre

3. A movie recommendation system using K-means and Association Rule can be useful tool for providing personalized movies recommendations to users.

Limitations of this approach is that it may not be able to handle very sparse data.

RESOURCE PAGE