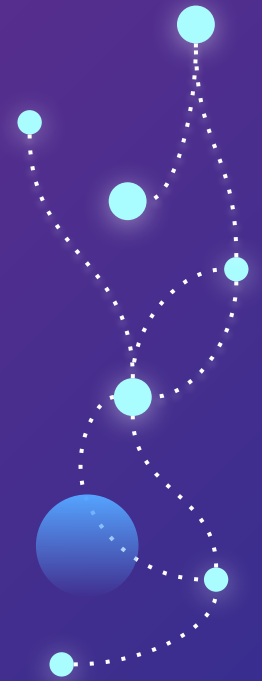
SC1015 Mini Project: Movie Recommendation Interface

Lab A132 Team 4
Aw Yu Ling Jewel (U2120808B)
Celeste Ang Jianing (U2222319H)
Kek Shi Min Emmelyn (U2121570F)

TABLE OF CONTENTS



01

MOTIVATION

Convenience?

Too many movies leads to choice paralysis for movie watchers who are just looking for a way to relax

How can we use different movie characteristics to make suitable recommendations based on a past movie watched?

The Movie Database API

02

EXTRACTION + EXPLORATORY DATA ANALYSIS

```
API_KEY = '96c7131158ec2d199c338e83be8f0fd5'

# Create empty DF to store the movie data
movie = pd.DataFrame()

resp = r.get("https://api.themoviedb.org/3/discover/movie?" \
                f"api_key={API_KEY}&language=en-US&sort_by=revenue.desc&" \
                "include_video=false&page=1")
```

```
rows = []
for i in tqdm(range(len(movie))):
        row = []
        movie_id = movie['id'][i]


        # Get the movie's other data that is not found in all movies
        resp = r.get(f'https://api.themoviedb.org/3/movie/{movie_id}?api_key={API_KEY}&language=en-US')


        try:
            budget = resp.json()['budget']
            if budget == 0:
                row.append(np.nan)
            else:
                row.append(budget)
        except KeyError:
            row.append(np.nan)
```

10,000 rows extracted

#	Column	Non-Null Count	Dtype
0	UNNAMED: 0	10000 non-null	int64
1	TITLE	10000 non-null	object
2	ID	10000 non-null	int64
3	POPULARITY	10000 non-null	float64
4	VOTE_COUNT	10000 non-null	int64
5	VOTE_AVERAGE	10000 non-null	float64
6	RELEASE_DATE	9953 non-null	object
7	ORIGINAL_LANGUAGE	10000 non-null	object
8	ADULT	10000 non-null	bool
9	BUDGET	7495 non-null	float64
10	GENRES	10000 non-null	object
11	REVENUE	10000 non-null	int64
12	YEAR_RELEASED	9953 non-null	float64
13	DECADE_RELEASED	9953 non-null	object
14	KEYWORDS	8991 non-null	object

#	Column	Non-Null Count	Dtype
0	TITLE	6971 non-null	object
1	ID	6971 non-null	int64
2	POPULARITY	6971 non-null	float64
3	VOTE_COUNT	6971 non-null	int64
4	VOTE_AVERAGE	6971 non-null	float64
5	RELEASE_DATE	6971 non-null	datetime64[ns]
6	ORIGINAL_LANGUAGE	6971 non-null	object
7	ADULT	6971 non-null	bool
8	BUDGET	6971 non-null	int64
9	GENRES	6971 non-null	object
10	REVENUE	6971 non-null	int64
11	YEAR_RELEASED	6971 non-null	int64
12	DECADE_RELEASED	6971 non-null	object
13	KEYWORDS	6971 non-null	object

```
movie_cleaning.isnull().sum()
```

```
UNNAMED: 0              0
TITLE                   0
ID                      0
POPULARITY              0
VOTE_COUNT              0
VOTE_AVERAGE            0
RELEASE_DATE           47
ORIGINAL_LANGUAGE       0
ADULT                   0
BUDGET               2505
GENRES                  0
REVENUE                 0
YEAR_RELEASED          47
DECADE_RELEASED        47
KEYWORDS             1009
```

Remove null values

Data left: 6971 rows

Data columns to be covered

- Genres
- Decade released
- Vote average
- Original Language

Genres

	GENRES	MAIN_GENRE
1	[Action, Adventure, Fantasy, Science Fiction]	Action
2	[Adventure, Science Fiction, Action]	Adventure
3	[Science Fiction, Adventure, Action]	Science Fiction
4	[Drama, Romance]	Drama
5	[Adventure, Action, Science Fiction, Fantasy]	Adventure
...	...	...
9993	[Comedy, Music, Romance]	Comedy
9994	[Crime, Drama, History, Romance, Thriller]	Crime
9995	[Comedy, Drama, Romance]	Comedy
9996	[Comedy, War]	Comedy
9998	[Comedy, Fantasy, Horror]	Comedy

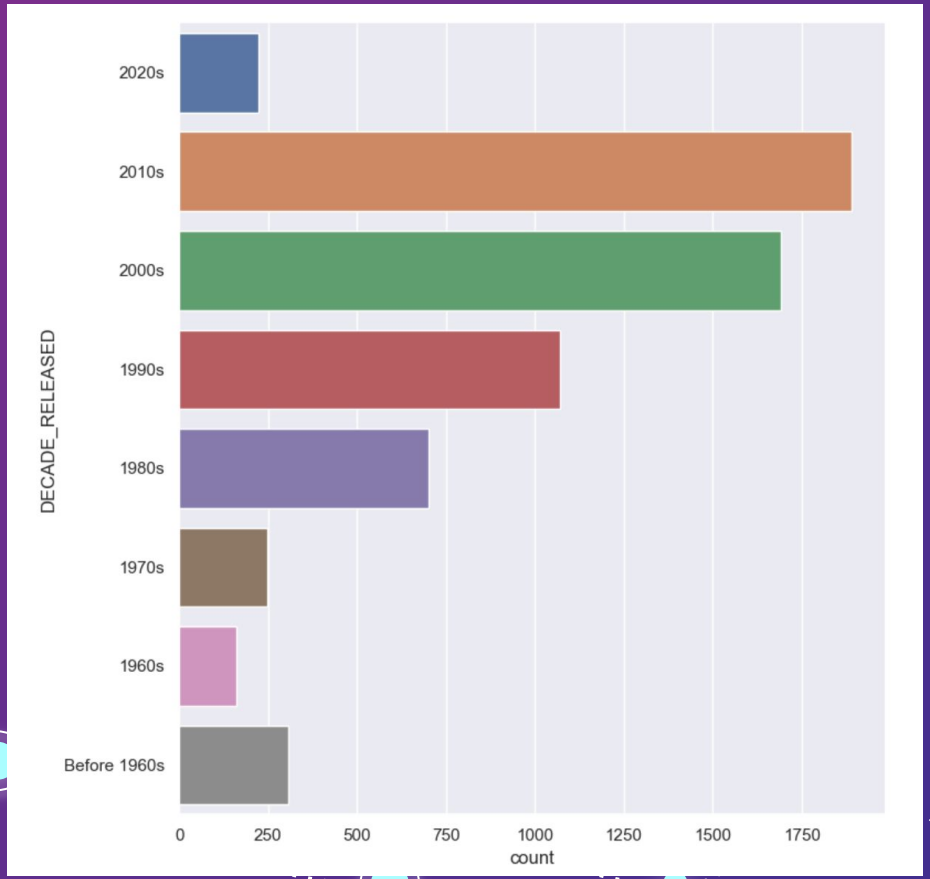
Many overlaps in genres

Extract first (main) genre to reduce overlaps

Main genres

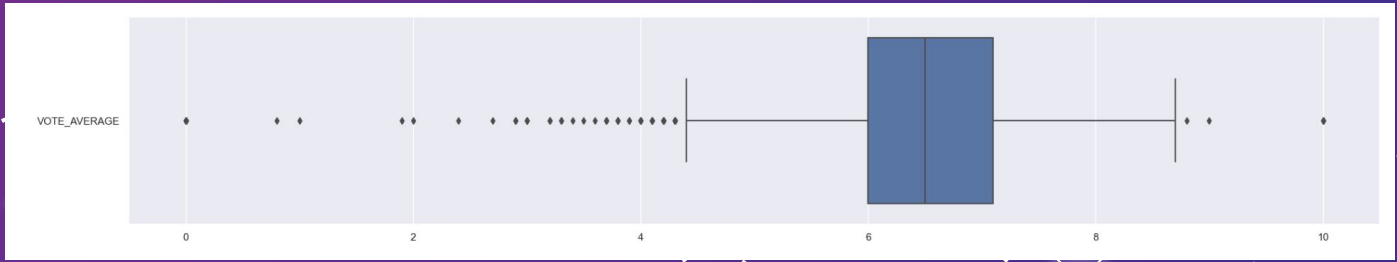
- Drama
- Comedy
- Action

Decade released



Vote average

	VOTE_AVERAGE
count	6399.000000
mean	6.474324
std	0.906762
min	0.000000
25%	6.000000
50%	6.500000
75%	7.100000
max	10.000000

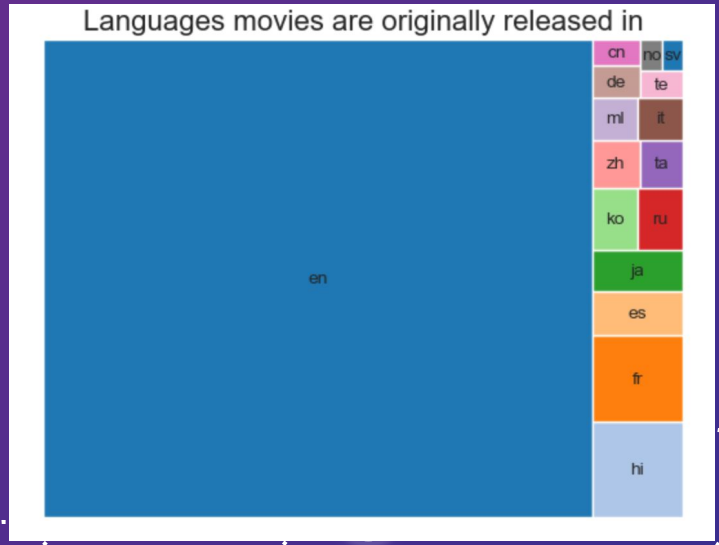


Original Language

```
en    5896
hi     194
fr     176
es      91
ja      83
ko      63
ru      63
zh      53
ta      47
ml      43
it      42
de      35
cn      28
te      26
no      15
sv      14
pt       9
da       9
th       8
fi       8
kn       7
fa       7
pl       6
nl       6
ur       6
tr       5
tl       5
ar       3
id       3
cs       3
el       2
pa       2
he       2
bn       2
af       1
nb       1
iu       1
ca       1
uk       1
et       1
ro       1
vi       1
hu       1
Name: ORIGINAL_LANGUAGE, dtype: int64
```

```python
import squarify

squarify.plot(sizes=OL_items['counts'], label=OL_items['Languages'],
              color = sb.color_palette("tab20", len(OL_items['counts'].value_counts())),
              text_kwargs={'fontsize': 10})
plt.title('Languages movies are originally released in', fontsize=16)
plt.axis('off')
```



Final dataframe

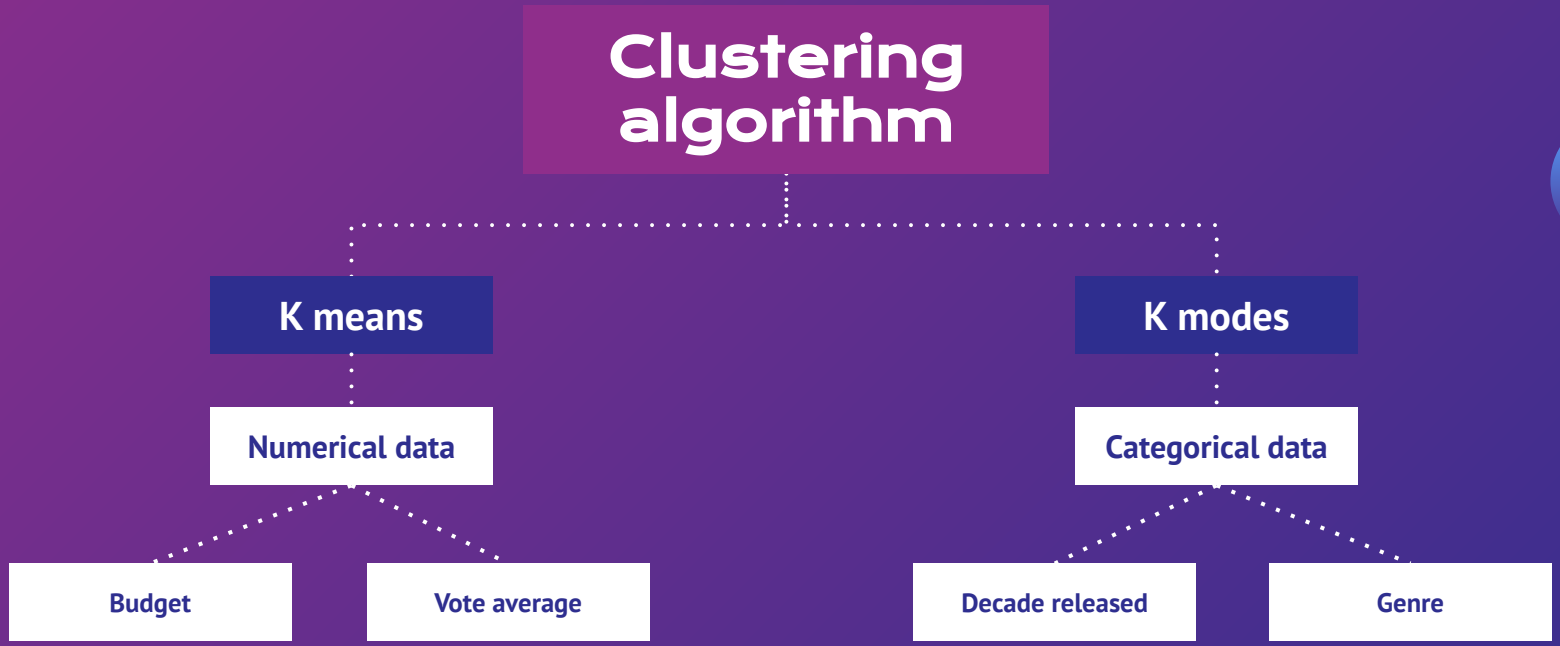
```
#Remove unwanted columns which will not benefit the model
df_progress = df_cleaned.drop(columns = ["POPULARITY", "GENRES", "ADULT", "VOTE_COUNT", "ORIGINAL_LANGUAGE",
                                         "YEAR_RELEASED", "RELEASE_DATE", "KEYWORDS"])
df_progress2 = df_progress.reset_index(drop=True)
df_progress2
```

	TITLE	ID	VOTE_AVERAGE	BUDGET	REVENUE	DECADE_RELEASED	MAIN_GENRE
0	Avatar	19995	7.6	237000000	2920357254	2000s	Action
1	Avengers: Endgame	299534	8.3	356000000	2799439100	2010s	Adventure
2	Avatar: The Way of Water	76600	7.7	460000000	2293000000	2020s	Science Fiction
3	Titanic	597	7.9	200000000	2187463944	1990s	Drama
4	Star Wars: The Force Awakens	140607	7.3	245000000	2068223624	2010s	Adventure
...	...	...	...	...	...	...	...
6966	At Long Last Love	4704	5.5	6000000	1500000	1970s	Comedy
6967	Under Capricorn	4175	6.0	2500000	1500000	Before 1960s	Crime
6968	Stolen Kisses	255	7.3	350000	1500000	1960s	Comedy
6969	To Be or Not to Be	198	7.9	1200000	1500000	Before 1960s	Comedy
6970	The Raven	29056	6.4	350000	1499275	1960s	Comedy

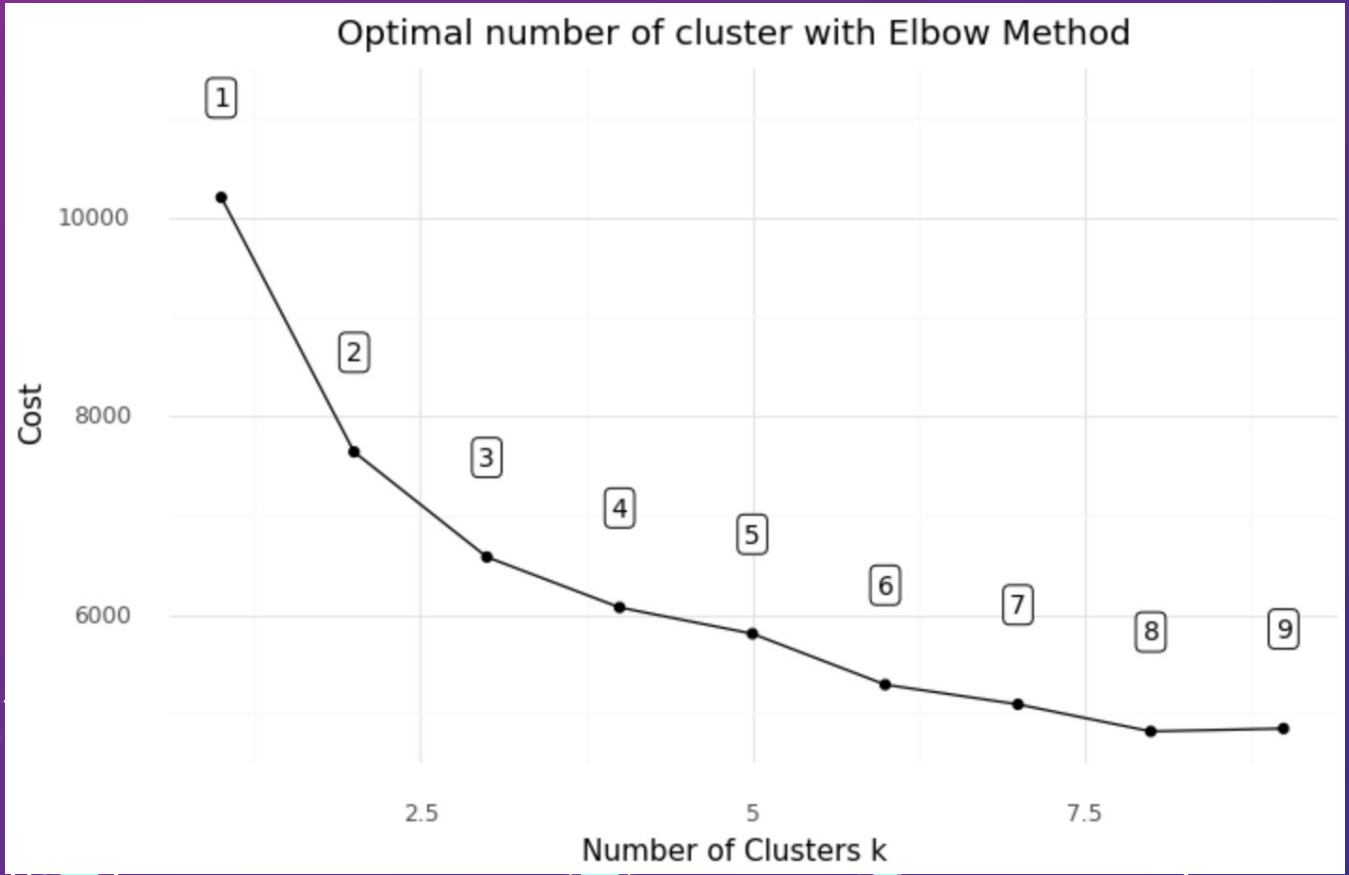
6971 rows × 7 columns

03

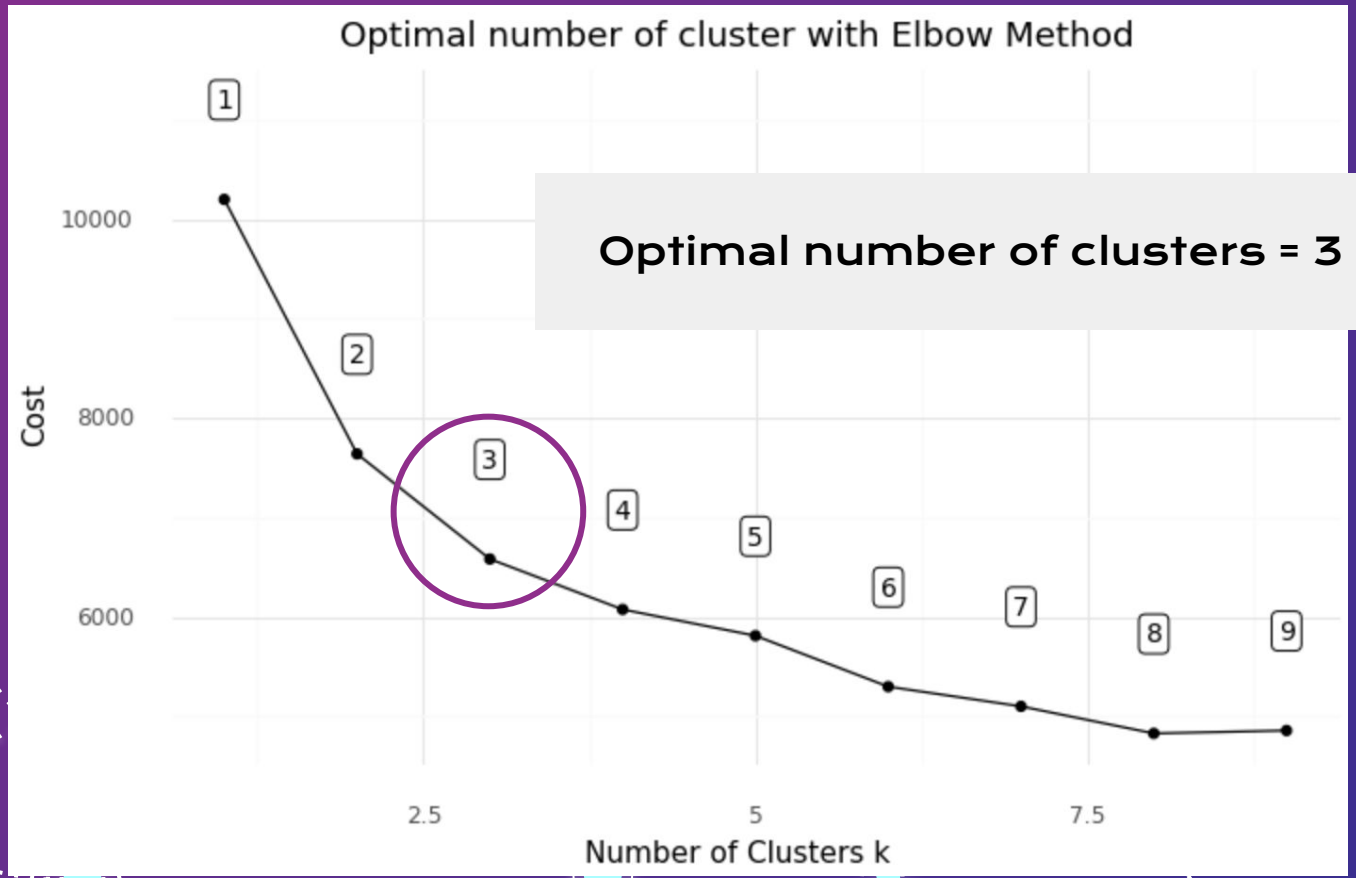
K MEANS + K MODES


Clustering algorithm
K means
K modes
Numerical data
Categorical data
Budget
Vote average
Decade released
Genre

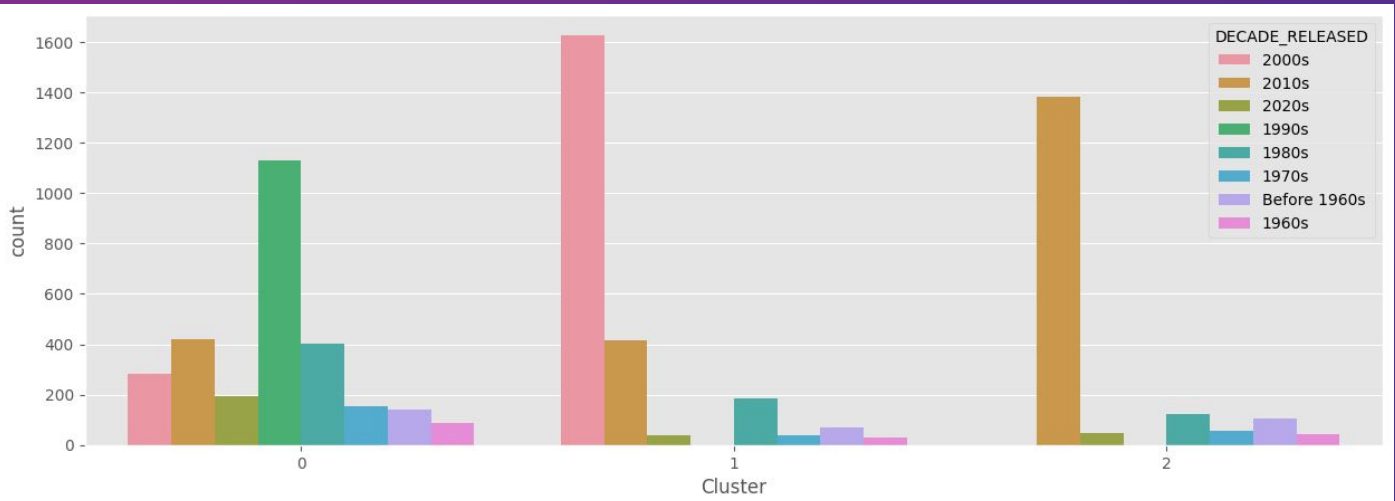
KModes - Elbow Method



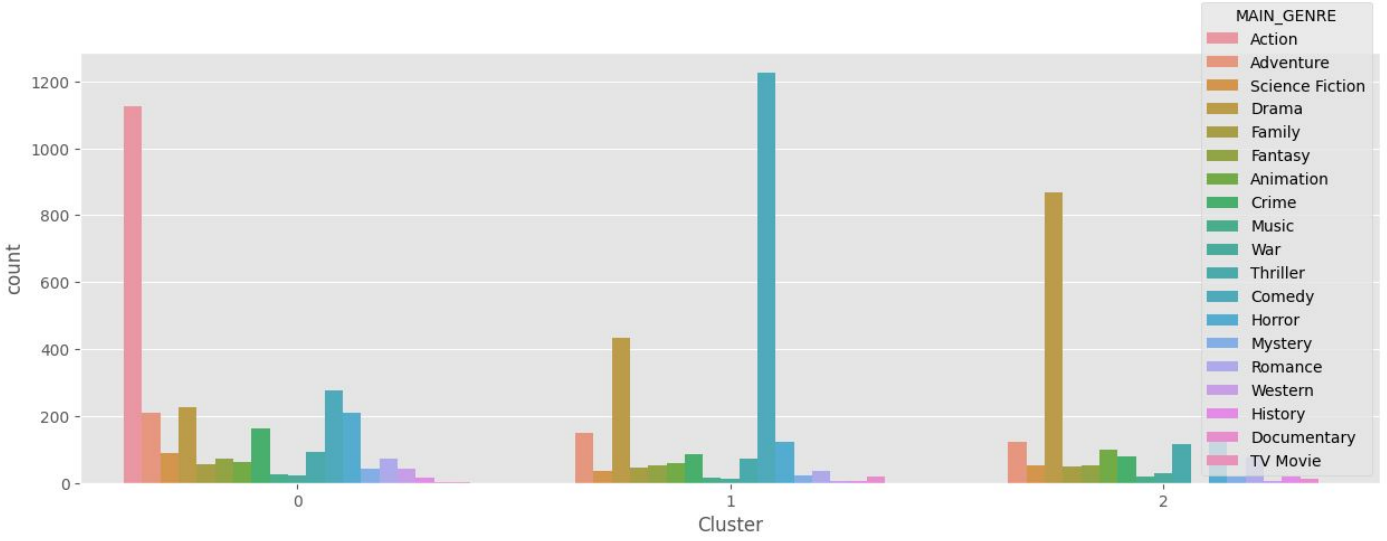
KModes - Elbow Method



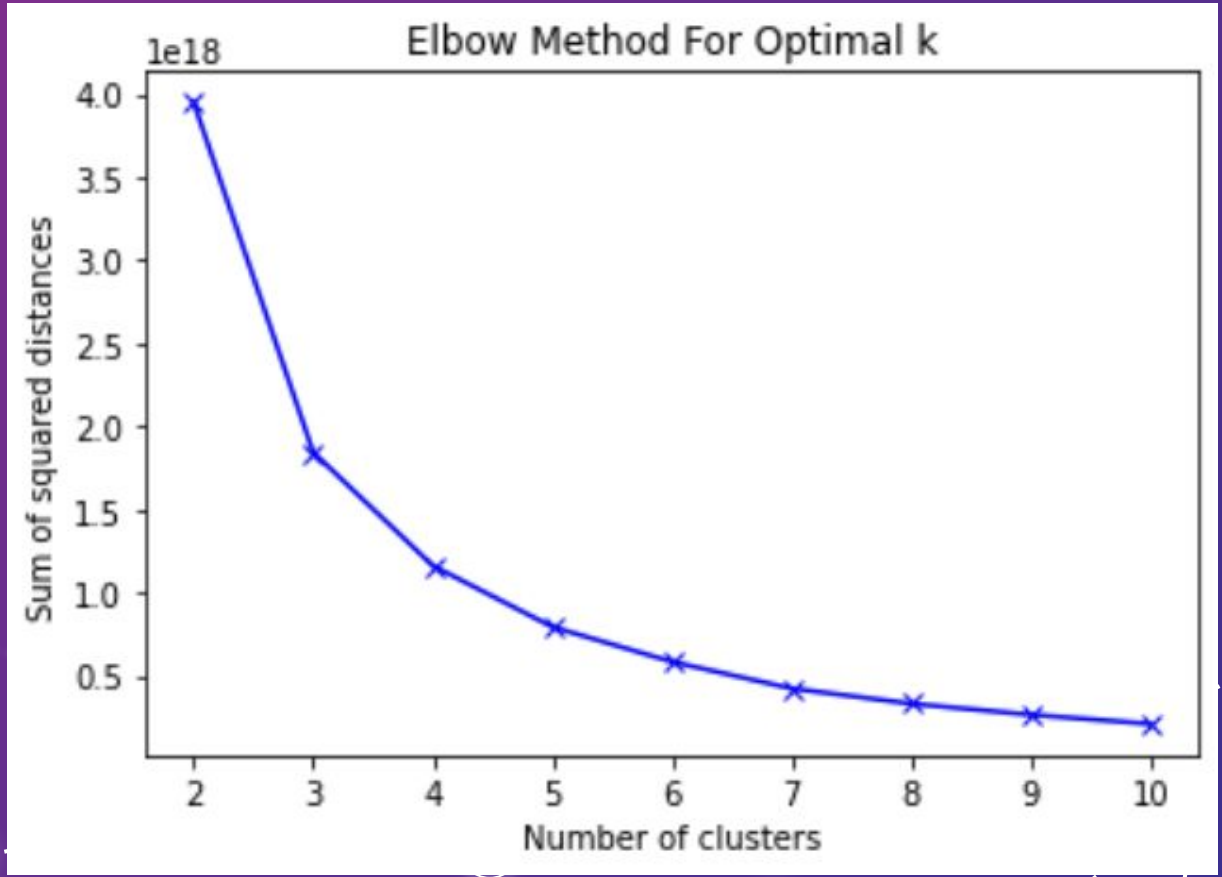
Decade Released



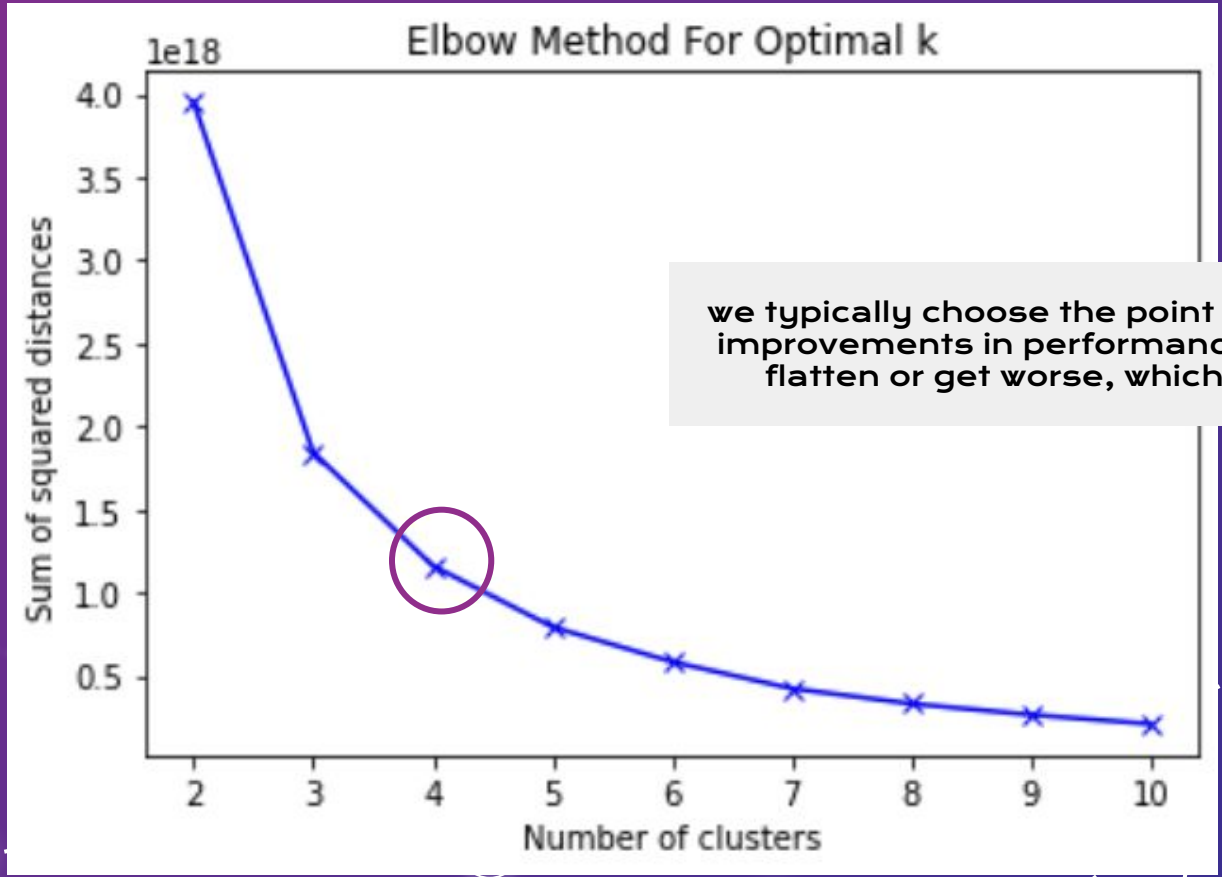
Genre



Kmeans- Elbow Method

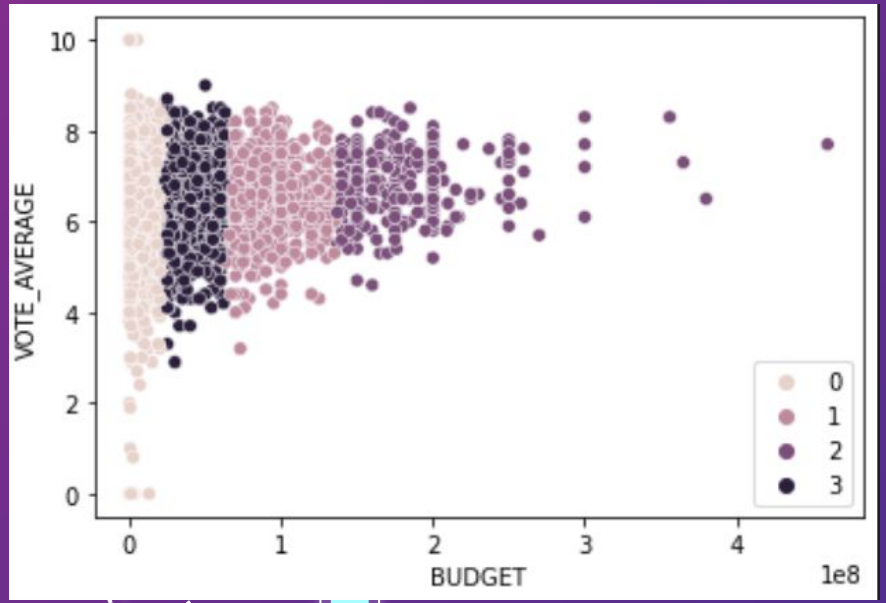


Kmeans- Elbow Method



we typically choose the point where the improvements in performance start to flatten or get worse, which is k =4

We see that k =4 is the best we can do without overfitting



For lower budget, vote average has a large range from 2 to 9. As the budget increases, we can see the range of vote_average decreasing and it converge to a value of 7.

04

Results

Functions

```python
#Function for getting all possible recommendations
def find_equals(userinput):
    try:
        index = final.index[final['TITLE'] == userinput].tolist()
        value1 = index[0]
        kmodes_cluster_no = (final.loc[value1]["kmodes_cluster"])
        kmeans_cluster_no = (final.loc[value1]["kmeans_cluster"])
        recommend = final[(final["kmodes_cluster"] == kmodes_cluster_no) & (final["kmeans_cluster"] == kmeans_cluster_no)]
        return recommend
    except:
        print("Movie not in database. Try another movie.")

#Function for processing recommendations
def recommendations(table):
    actual_list = table['TITLE'].values.tolist()
    return actual_list
```

Get dataframe with recommended movies

Turn dataframe into list

Final interface

```
Welcome to movie recommendation database

1: Input movie
2: Exit

Please choose an option 1

Please input movie you want recommendations to be made from (Follow exact wording from TMDB website)
Hugo
```

```
Please input movie you want recommendations to be made from (Follow exact wording from TMDB website) Hugo
Here are some suggestions

Avengers: Endgame
Star Wars: The Force Awakens
Avengers: Infinity War
The Lion King
The Avengers
Frozen II
Star Wars: The Last Jedi
Frozen
Beauty and the Beast
Captain America: Civil War
Would you like any more recommendations? (Y/N) Y
Transformers: Age of Extinction
Star Wars: The Rise of Skywalker
Toy Story 4
Toy Story 3
Aladdin
Pirates of the Caribbean: On Stranger Tides
Finding Dory
Alice in Wonderland
Zootopia
The Hobbit: An Unexpected Journey

Would you like any more recommendations? (Y/N) Y
The Jungle Book
The Hobbit: The Desolation of Smaug
Harry Potter and the Deathly Hallows: Part 1
The Battle at Lake Changjin
Guardians of the Galaxy Vol. 2
Inside Out
Coco
Pirates of the Caribbean: Dead Men Tell No Tales
Maleficent
Madagascar 3: Europe's Most Wanted


Would you like any more recommendations? (Y/N)
```

THANKS!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution