6131881421

Yoschanin Pulsirivong

*PLEASE ADD A movies.csv FILE AND TO SAME DIRECTORY, FILE WAS TOO LARGE

Exploration

[5]:
```
full = pd.read_csv("movies.csv")
full.dropna(inplace=True)
full.head()
```

[5]:

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name
0	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore
1	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom
2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear
3	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale
5	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton

5 rows × 27 columns

[6]:
```
full = pd.read_csv("movies.csv")
full.dropna(inplace=True)
full.tail()
```

[6]:

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_nar
5026	Olivier Assayas	81.0	110.0	107.0	45.0	Béatrice Da
5027	Jafar Panahi	64.0	90.0	397.0	0.0	Narge Mamizad
5033	Shane Carruth	143.0	77.0	291.0	8.0	David Sulliv
5035	Robert Rodriguez	56.0	81.0	0.0	6.0	Pe Marqua
5042	Jon Gunn	43.0	90.0	16.0	16.0	Bri Herzling

5 rows × 27 columns

[7]:
```
full = pd.read_csv("movies.csv")
full.dropna(inplace=True)
full.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3757 entries, 0 to 5042
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   director_name              3757 non-null   object
 1   num_critic_for_reviews     3757 non-null   float64
 2   duration                   3757 non-null   float64
 3   director_facebook_likes    3757 non-null   float64
 4   actor_3_facebook_likes     3757 non-null   float64
 5   actor_2_name               3757 non-null   object
 6   actor_1_facebook_likes     3757 non-null   float64
 7   gross                      3757 non-null   float64
 8   genres                     3757 non-null   object
 9   actor_1_name               3757 non-null   object
 10  movie_title                3757 non-null   object
 11  num_voted_users            3757 non-null   int64
 12  cast_total_facebook_likes  3757 non-null   int64
 13  actor_3_name               3757 non-null   object
 14  facenumber_in_poster       3757 non-null   float64
 15  plot_keywords              3757 non-null   object
 16  movie_imdb_link            3757 non-null   object
 17  num_user_for_reviews       3757 non-null   float64
 18  language                   3757 non-null   object
 19  country                    3757 non-null   object
 20  content_rating             3757 non-null   object
 21  budget                     3757 non-null   float64
 22  title_year                 3757 non-null   float64
 23  actor_2_facebook_likes     3757 non-null   float64
 24  imdb_score                 3757 non-null   float64
 25  aspect_ratio               3757 non-null   float64
 26  movie_facebook_likes       3757 non-null   int64
dtypes: float64(13), int64(3), object(11)
memory usage: 821.8+ KB
```

*__NaN__ value cells were dropped before running these commands

*__i__mports were run in a previous cell

Exploration(2)

[8]:
```
full = pd.read_csv("movies.csv")
full.dropna(inplace=True)
full.describe()
```

[8]:

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes
count	3757.000000	3757.000000	3757.000000	3757.000000	3757.000000
mean	167.348682	110.256322	807.137876	771.194304	7749.534735
std	123.448979	22.643932	3067.787384	1894.004897	15517.667480
min	2.000000	37.000000	0.000000	0.000000	0.000000
25%	77.000000	96.000000	11.000000	194.000000	745.000000
50%	138.000000	106.000000	64.000000	436.000000	1000.000000
75%	224.000000	120.000000	235.000000	691.000000	13000.000000
max	813.000000	330.000000	23000.000000	23000.000000	640000.000000

[9]:
```
full = pd.read_csv("movies.csv")
full.isna()
```

[9]:

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_nar
0	False	False	False	False	False	Fal
1	False	False	False	False	False	Fal
2	False	False	False	False	False	Fal
3	False	False	False	False	False	Fal
4	False	True	True	False	True	Fal
...	...	...	...	...	...	
5038	False	False	False	False	False	Fal
5039	True	False	False	True	False	Fal
5040	False	False	False	False	False	Fal
5041	False	False	False	False	False	Fal
5042	False	False	False	False	False	Fal

5043 rows × 27 columns

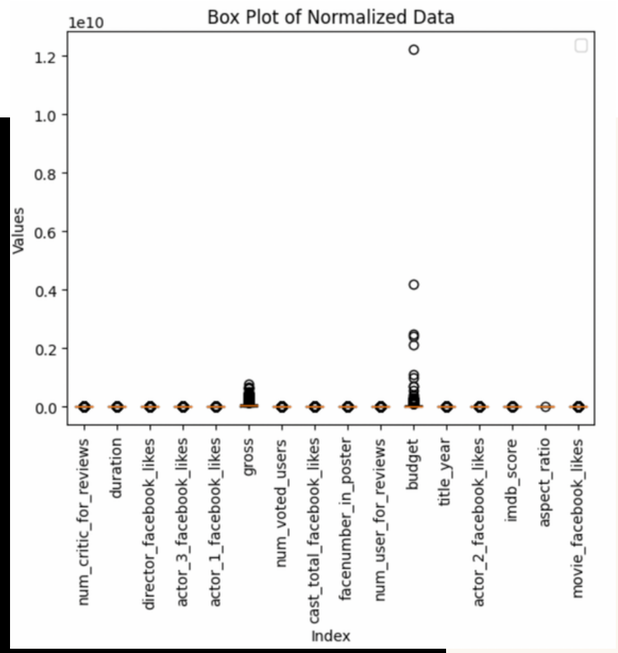
[10]:
```
full = pd.read_csv("movies.csv")
full.value_counts()
```

[10]:
```
director_name     num_critic_for_reviews  duration  director_facebook_likes  actor_3_facebook_likes  a
ctor_2_name         actor_1_facebook_likes  gross       genres                                  actor_1_
name          movie_title           num_voted_users  cast_total_facebook_likes  actor_3_name     facenum
ber_in_poster  plot_keywords                                                    movie_imdb_link
num_user_for_reviews  language  country      content_rating  budget      title_year  actor_2_facebook_
likes  imdb_score  aspect_ratio  movie_facebook_likes
Frank Oz          168.0                   87.0      0.0                     548.0                   E
wen Bremner         22000.0               8579684.0   Comedy                                  Peter Di
nklage       Death at a Funeral   89547            24324                     Kris Marshall    0.0
end credits roll call|four word title|funeral|secret|uncle        http://www.imdb.com/title/tt079536
8/?ref_=fn_tt_tt_1  199.0               English   USA         R               9000000.0   2007.0
557.0                   7.4         1.85          0                       2
Wes Craven        160.0                   107.0     0.0                     255.0                   E
mily Meade          798.0                 14637490.0  Horror|Mystery|Thriller               Frank Gr
illo         My Soul to Take      16411            2537                      Zena Grey        2.0
evil|killer|murder|psychopath|serial killer                    http://www.imdb.com/title/tt087223
0/?ref_=fn_tt_tt_1  136.0               English   USA         R               25000000.0  2010.0
374.0                   4.8         2.35          0                       2
Shawn Levy        69.0                    88.0      189.0                   799.0                   D
onald Faison        934.0                 47811275.0  Adventure|Comedy|Family               Frankie
Muniz        Big Fat Liar         29008            3707                      Lee Majors       1.0
brunette|film producer|liar|prank|stunt                        http://www.imdb.com/title/tt026529
8/?ref_=fn_tt_tt_1  99.0                English   USA         PG              15000000.0  2002.0
927.0                   5.4         1.85          896                     2
Albert Hughes     208.0                   122.0     117.0                   140.0                   J
ason Flemyng        40000.0               31598308.0  Horror|Mystery|Thriller               Johnny D
epp          From Hell            124765           41636                     Ian Richardson   1.0
freemason|jack the ripper|opium|prostitute|victorian era       http://www.imdb.com/title/tt012068
1/?ref_=fn_tt_tt_1  541.0               English   USA         R               35000000.0  2001.0
1000.0                  6.8         2.35          0                       2
Tamra Davis       111.0                   93.0      33.0                    135.0                   K
atherine Boecher    1000.0                37188667.0  Comedy|Drama                          Britney
Spears       Crossroads           34219            1531                      Dave Allen       1.0
audition|friendship|graduation|high school graduation|love     http://www.imdb.com/title/tt027502
2/?ref_=fn_tt_tt_1  578.0               English   USA         PG-13           12000000.0  2002.0
188.0                   3.3         1.85          0                       2

..
Hyung-rae Shim    4.0                     100.0     26.0                    385.0                   S
tephanie Danielson  898.0                 163591.0    Comedy                                  Jason Me
```

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from kneed import KneeLocator
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
```

Box plots of .describe

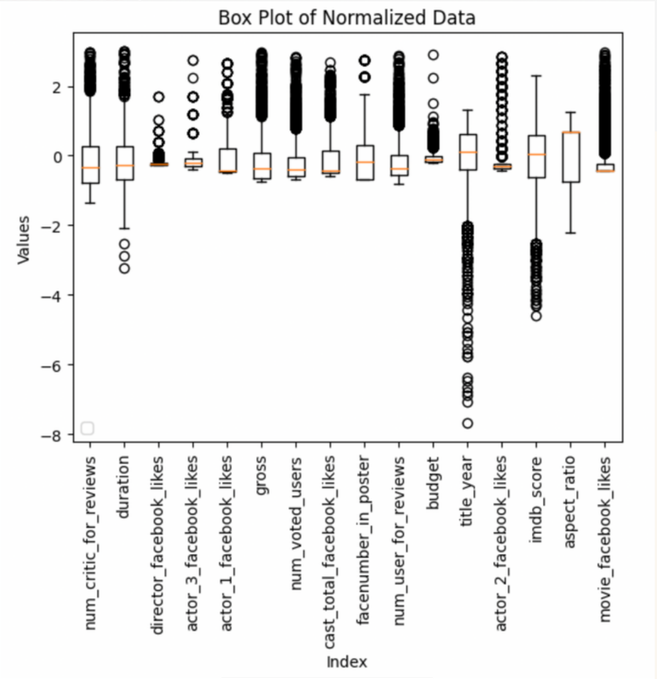
```
ful = pd.read_csv("movies.csv")
full = ful.dropna()
full.to_numpy()
full = full.select_dtypes(include=[np.number])


# full_subset = full.iloc[:5000]
# normalized_subset = normalized_df.iloc[:5000]

# z_scores = (full - full.mean()) / full.std()
# threshold = 3
# outliers = (z_scores > threshold).any(axis=1)
# cleaned_data = full[~outliers]
# model = KMeans(n_clusters=3)
# model = model.fit(scale(cleaned_data))

plt.boxplot(full)
plt.xticks(range(1, len(full.columns) + 1), full.columns, rotation=90)
plt.ylabel('Normalized Values')
plt.title('Box Plot of Normalized Data')
plt.xlabel('Index')
plt.ylabel('Values')
plt.legend()
plt.show()
```



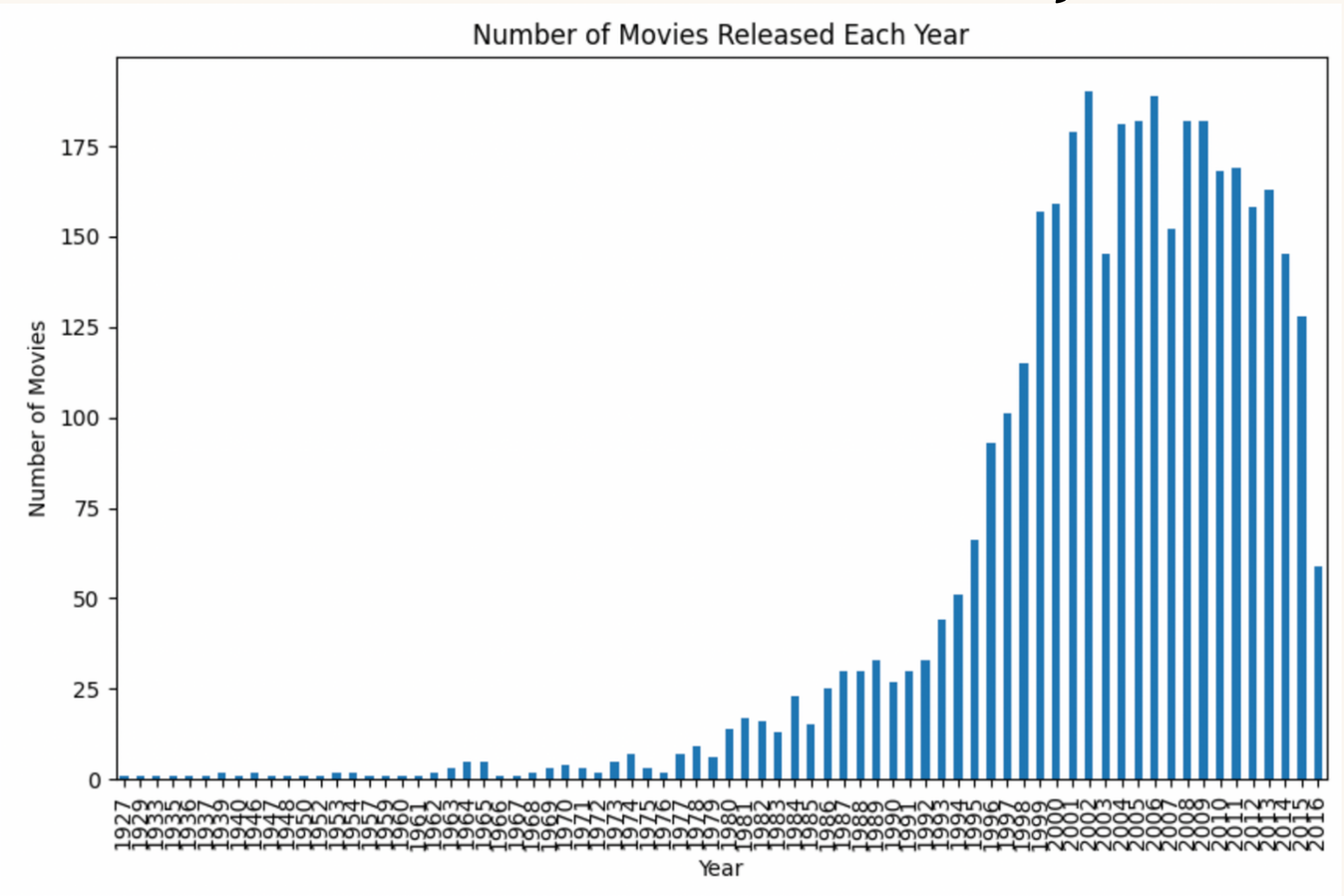
< Before removal of anomalies and normalization for better visualization.

After anomaly removal and normalization >

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import scale
from sklearn.metrics.cluster import contingency_matrix


full = pd.read_csv("movies.csv")
full = full.dropna()
numeric_columns = full.select_dtypes(include=[np.number])
normalized_columns = scale(numeric_columns)
normalized_df = pd.DataFrame(normalized_columns, columns=numeric_columns.columns)

z_scores = (normalized_df - normalized_df.mean()) / normalized_df.std()
threshold = 3
outliers = (z_scores > threshold).any(axis=1)
cleaned_data = normalized_df[~outliers]
model = KMeans(n_clusters=3)
model = model.fit(scale(cleaned_data))

plt.boxplot(cleaned_data)
plt.xticks(range(1, len(cleaned_data.columns) + 1), cleaned_data.columns, rotation=90)
plt.ylabel('Normalized Values')
plt.title('Box Plot of Normalized Data')
plt.xlabel('Index')
plt.ylabel('Values')
plt.legend()
plt.show()
```



```python
import pandas as pd
import matplotlib.pyplot as plt

full = pd.read_csv("movies.csv")
full = full.dropna()
full['title_year'] = full['title_year'].astype(int)

title_years = full['title_year']
year_counts = title_years.value_counts()
year_counts = year_counts.sort_index()

plt.figure(figsize=(10, 6))
year_counts.plot(kind='bar')
plt.xlabel('Year')
plt.ylabel('Number of Movies')
plt.title('Number of Movies Released Each Year')
plt.show()
```



Value_counts() plot to show the number of movies released each year

Integer ----> Float

fillna(0)

[18]:
```
import pandas as pd
import numpy as np
full = pd.read_csv("movies.csv")
full.value_counts()
numeric_columns = full.select_dtypes(include=[np.number]).columns
full[numeric_columns] = full[numeric_columns].astype(float)
full
```

[18]:

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_nar
0	James Cameron	723.0	178.0	0.0	855.0	Joel Dav Moc
1	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloc
2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinne
3	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Ba
4	Doug Walker	NaN	NaN	131.0	NaN	Rob Walk
...	...	...	...	...	...	
5038	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuni
5039	NaN	43.0	43.0	NaN	319.0	Valorie Cu
5040	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxw Moo
5041	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henn
5042	Jon Gunn	43.0	90.0	16.0	16.0	Bri Herzling

5043 rows × 27 columns

[23]:
```
import pandas as pd
import numpy as np
full = pd.read_csv("movies.csv")
full.value_counts()
numeric_columns = full.select_dtypes(include=[np.number]).columns
full[numeric_columns] = full[numeric_columns].astype(float)
full = full.fillna(0)
full
```

[23]:

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_nar
0	James Cameron	723.0	178.0	0.0	855.0	Joel Dav Moc
1	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloc
2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinne
3	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Ba
4	Doug Walker	0.0	0.0	131.0	0.0	Rob Walk
...	...	...	...	...	...	
5038	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuni
5039	0	43.0	43.0	0.0	319.0	Valorie Cu
5040	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxw Moo
5041	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henn
5042	Jon Gunn	43.0	90.0	16.0	16.0	Bri Herzling

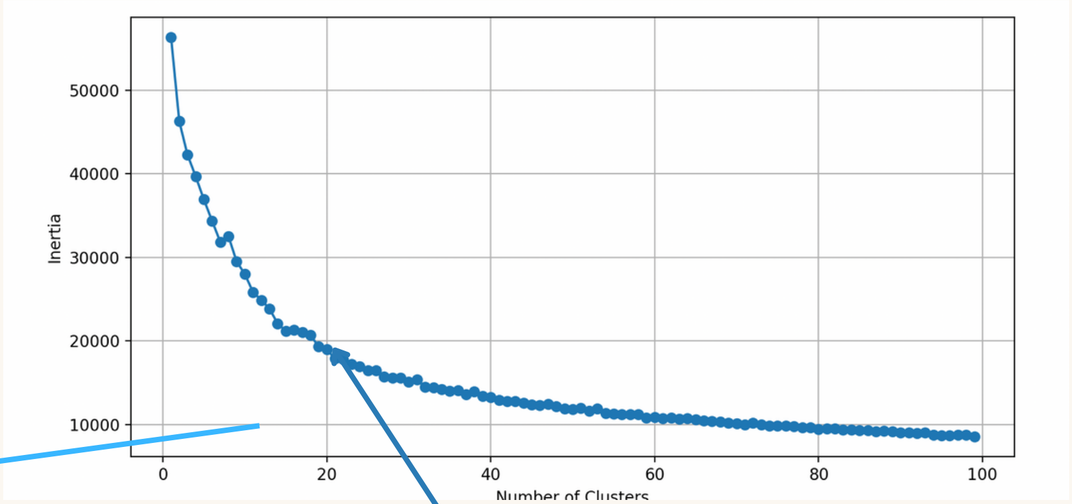
5043 rows × 27 columns

Normalization in slide 4

```
full = pd.read_csv("movies.csv")
full.dropna(inplace=True)
scaler = StandardScaler()
full[['num_critic_for_reviews_T', 'duration_T', 'director_facebook_likes_T', 'actor_3_facebook_likes_T', 'actor_1_facebook_likes_T', 'gross_T', 'num_voted_users_T',
'cast_total_facebook_likes_T', 'num_user_for_reviews_T', 'budget_T', 'title_year_T', 'actor_2_facebook_likes_T', 'imdb_score_T', 'aspect_ratio_T', 'movie_facebook_likes_T' ]] =
scaler.fit_transform(full[['num_critic_for_reviews', 'duration', 'director_facebook_likes', 'actor_3_facebook_likes', 'actor_1_facebook_likes', 'gross', 'num_voted_users',
'cast_total_facebook_likes', 'num_user_for_reviews', 'budget', 'title_year', 'actor_2_facebook_likes', 'imdb_score', 'aspect_ratio', 'movie_facebook_likes' ]])
```

select only numeric columns, exclude strings, after removing empty cells then standardization with StandardScaler

```
def optimise_k_means(data, max_k):
  means = []
  inertias = []

  for k in range(1, max_k):
    kmeans=KMeans(n_clusters=k)
    kmeans.fit(data)

    means.append(k)
    inertias.append(kmeans.inertia_)

  fig, ax = plt.subplots(figsize=(10,5))
  ax.plot(means, inertias, 'o-')
  ax.set_xlabel('Number of Clusters')
  ax.set_ylabel('Inertia')
  ax.grid(True)
  plt.show()
```




```
optimise_k_means(full[['num_critic_for_reviews_T', 'duration_T', 'director_facebook_likes_T', 'actor_3_facebook_likes_T', 'actor_1_facebook_likes_T', 'gross_T',
'num_voted_users_T', 'cast_total_facebook_likes_T', 'num_user_for_reviews_T', 'budget_T', 'title_year_T', 'actor_2_facebook_likes_T', 'imdb_score_T', 'aspect_ratio_T',
'movie_facebook_likes_T']], 100)
```

Finding the elbow point

k = ~20

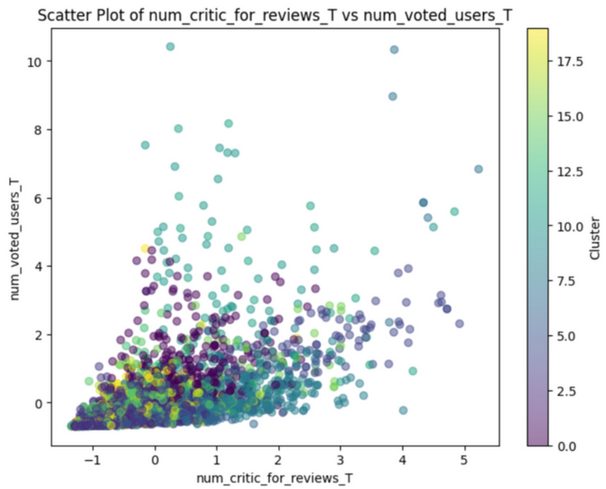
```
kmeans = KMeans(n_clusters=20)
kmeans.fit(full[['num_critic_for_reviews_T', 'duration_T', 'director_facebook_likes_T', 'actor_3_facebook_likes_T', 'actor_1_facebook_likes_T', 'gross_T', 'num_voted_users_T',
'cast_total_facebook_likes_T', 'num_user_for_reviews_T', 'budget_T', 'title_year_T', 'actor_2_facebook_likes_T', 'imdb_score_T', 'aspect_ratio_T', 'movie_facebook_likes_T']])
full['kmeans_20']=kmeans.labels_
t_columns = [col for col in full.columns if '_T' in col]

for i in range(len(t_columns)):
  for j in range(i+1, len(t_columns)):
    plt.figure(figsize=(8, 6))
    plt.scatter(full[t_columns[i]], full[t_columns[j]], c=full['kmeans_20'], cmap='viridis', alpha=0.5)
    plt.xlabel(t_columns[i])
    plt.ylabel(t_columns[j])
    plt.title(f'Scatter Plot of {t_columns[i]} vs {t_columns[j]}')
    plt.colorbar(label='Cluster')
    plt.show()
```

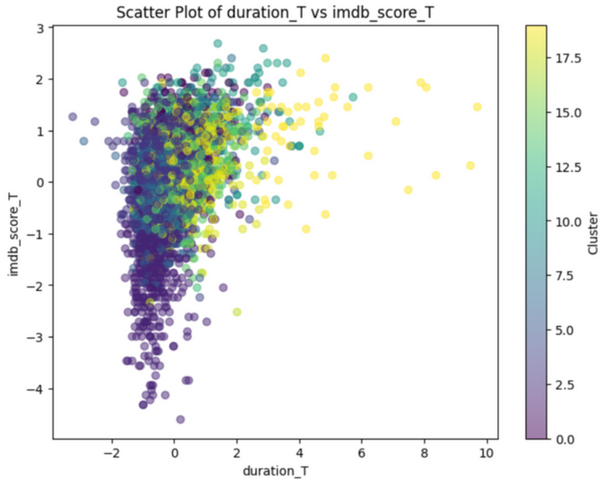
```
full = pd.read_csv("movies.csv")
full.dropna(inplace=True)
scaler = StandardScaler()
full[['num_critic_for_reviews_T', 'duration_T', 'director_facebook_likes_T', 'actor_3_facebook_likes_T', 'actor_1_facebook_likes_T', 'gross_T', 'num_voted_users_T',
'cast_total_facebook_likes_T', 'num_user_for_reviews_T', 'budget_T', 'title_year_T', 'actor_2_facebook_likes_T', 'imdb_score_T', 'aspect_ratio_T', 'movie_facebook_likes_T' ]] =
scaler.fit_transform(full[['num_critic_for_reviews', 'duration', 'director_facebook_likes', 'actor_3_facebook_likes', 'actor_1_facebook_likes', 'gross', 'num_voted_users',
'cast_total_facebook_likes', 'num_user_for_reviews', 'budget', 'title_year', 'actor_2_facebook_likes', 'imdb_score', 'aspect_ratio', 'movie_facebook_likes' ]])

def optimise_k_means(data, max_k):
    means = []
    inertias = []

    for k in range(1, max_k):
        kmeans=KMeans(n_clusters=k)
        kmeans.fit(data)

        means.append(k)
        inertias.append(kmeans.inertia_)

    fig, ax = plt.subplots(figsize=(10,5))
    ax.plot(means, inertias, 'o-')
    ax.set_xlabel('Number of Clusters')
    ax.set_ylabel('Inertia')
    ax.grid(True)
    plt.show()

optimise_k_means(full[['num_critic_for_reviews_T', 'duration_T', 'director_facebook_likes_T', 'actor_3_facebook_likes_T', 'actor_1_facebook_likes_T', 'gross_T',
'num_voted_users_T', 'cast_total_facebook_likes_T', 'num_user_for_reviews_T', 'budget_T', 'title_year_T', 'actor_2_facebook_likes_T', 'imdb_score_T', 'aspect_ratio_T',
'movie_facebook_likes_T']], 100)

kmeans = KMeans(n_clusters=20)
kmeans.fit(full[['num_critic_for_reviews_T', 'duration_T', 'director_facebook_likes_T', 'actor_3_facebook_likes_T', 'actor_1_facebook_likes_T', 'gross_T', 'num_voted_users_T',
'cast_total_facebook_likes_T', 'num_user_for_reviews_T', 'budget_T', 'title_year_T', 'actor_2_facebook_likes_T', 'imdb_score_T', 'aspect_ratio_T', 'movie_facebook_likes_T']])
full['kmeans_20']=kmeans.labels_
t_columns = [col for col in full.columns if '_T' in col]

for i in range(len(t_columns)):
    for j in range(i+1, len(t_columns)):
        plt.figure(figsize=(8, 6))
        plt.scatter(full[t_columns[i]], full[t_columns[j]], c=full['kmeans_20'], cmap='viridis', alpha=0.5)
        plt.xlabel(t_columns[i])
        plt.ylabel(t_columns[j])
        plt.title(f'Scatter Plot of {t_columns[i]} vs {t_columns[j]}')
        plt.colorbar(label='Cluster')
        plt.show()
```

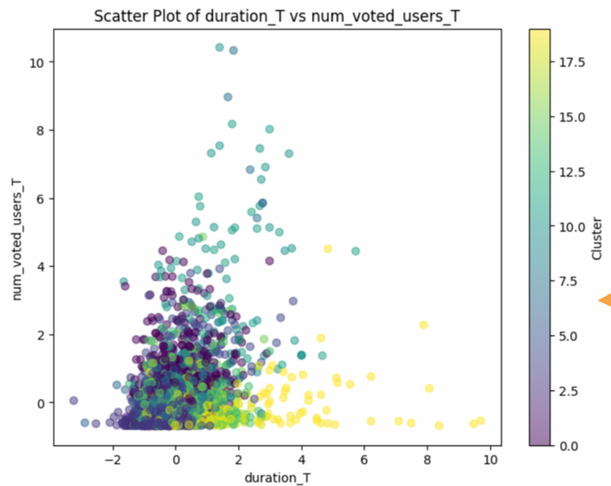
New column, information on which cluster each row belongs to

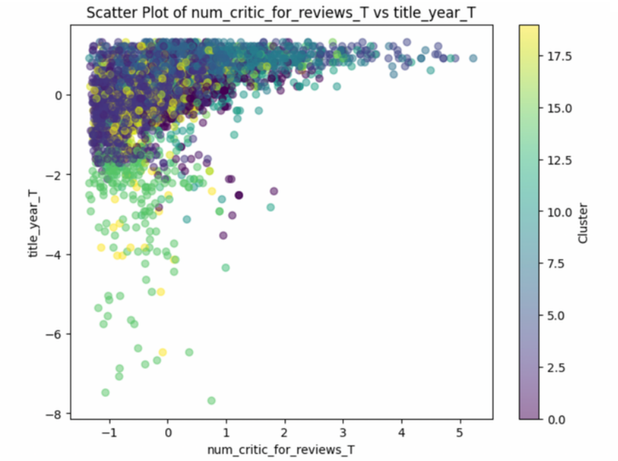
Finding the elbow point

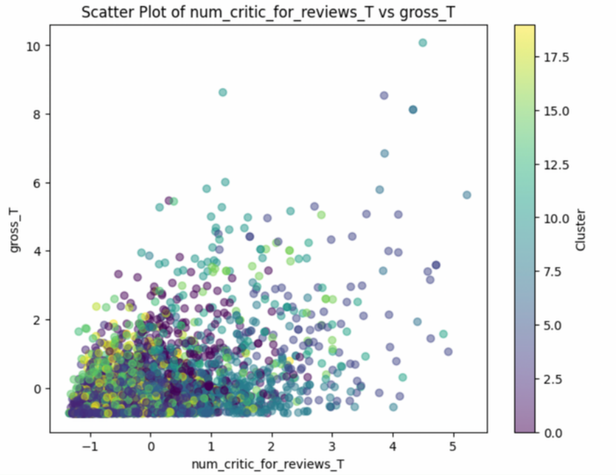
k = ~20

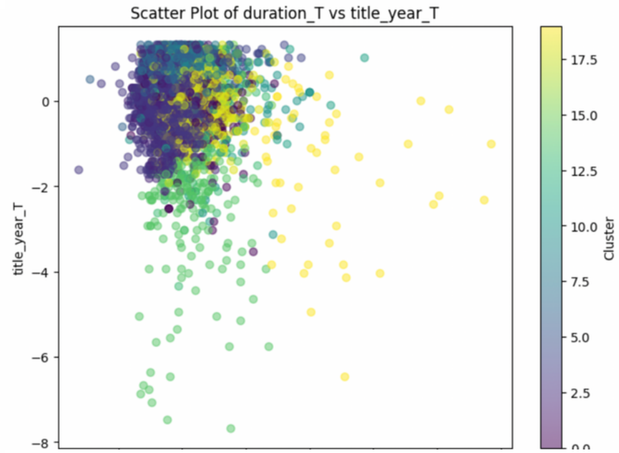


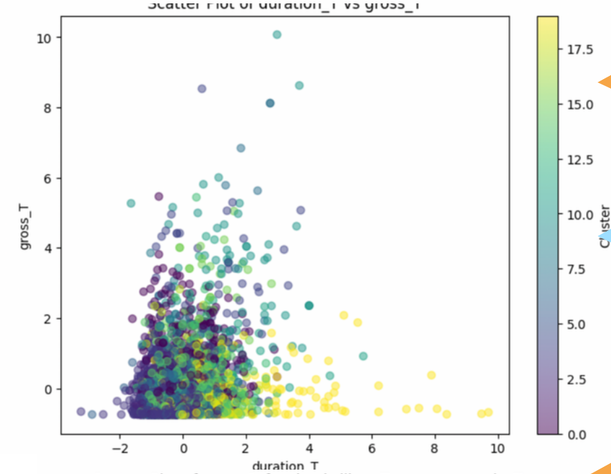


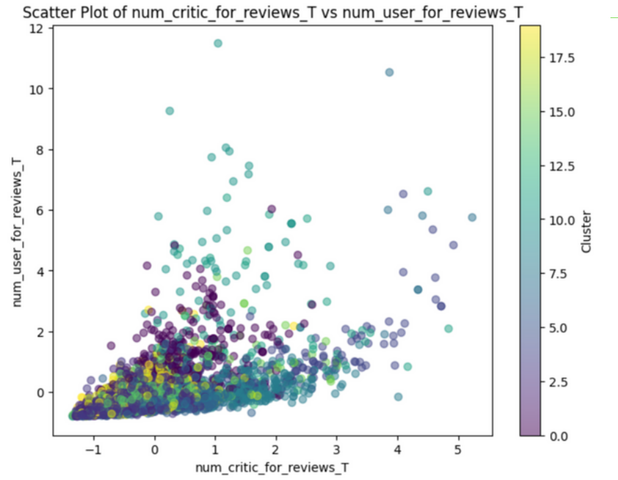


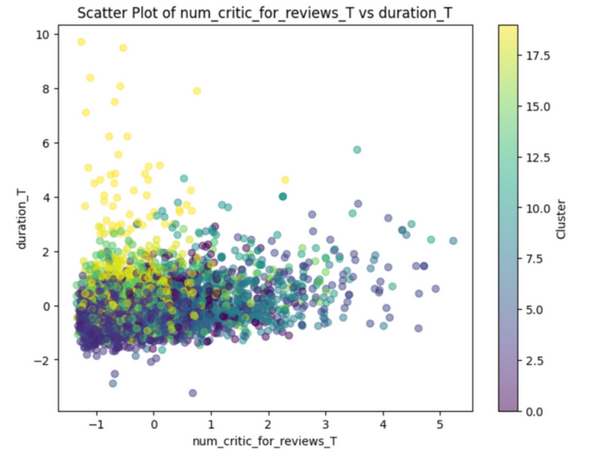


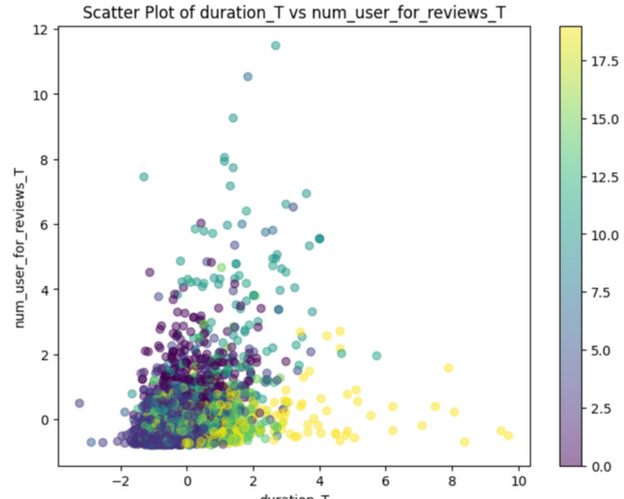


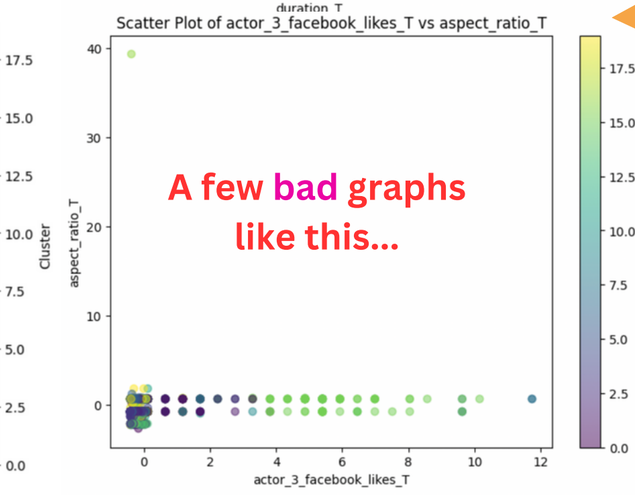






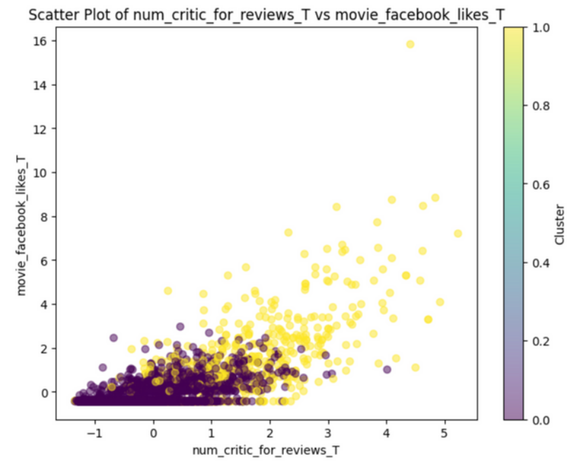


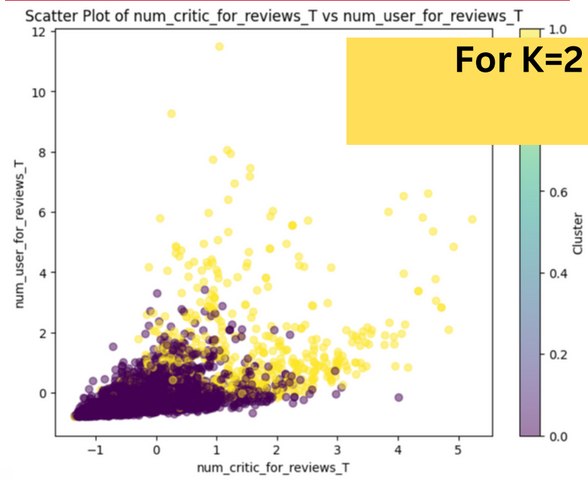


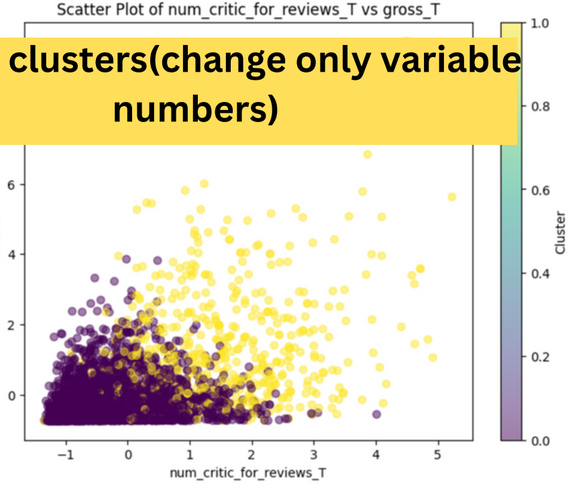


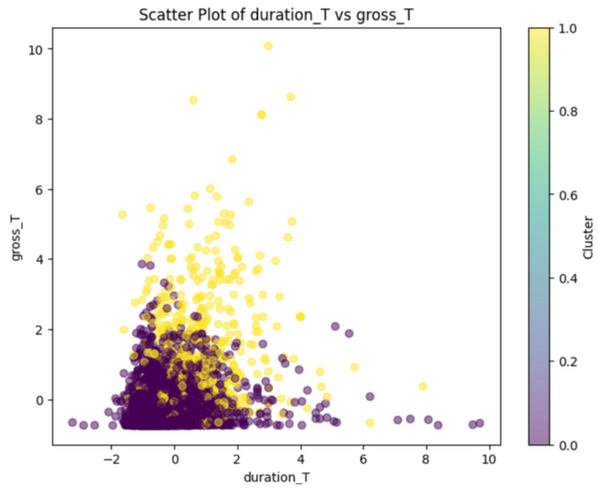
A few bad graphs like this...

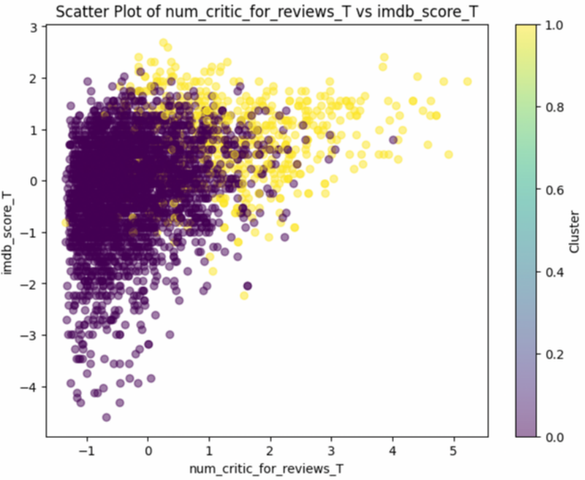
For K=2 clusters(change only variable numbers)

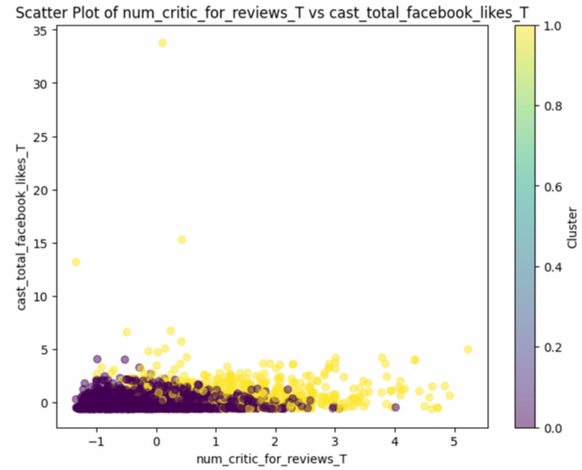


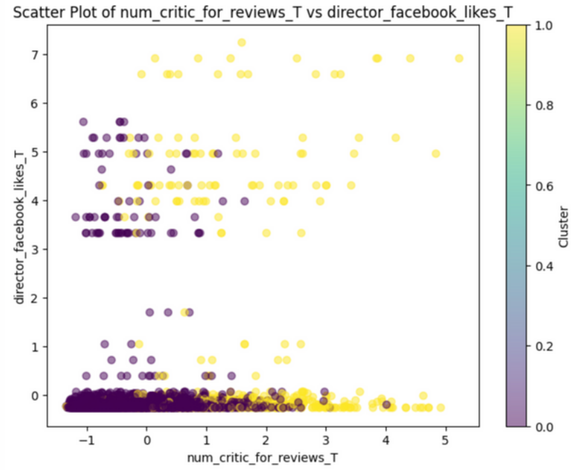


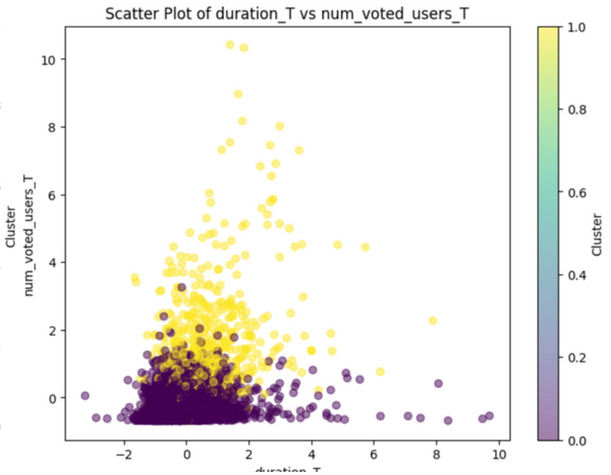


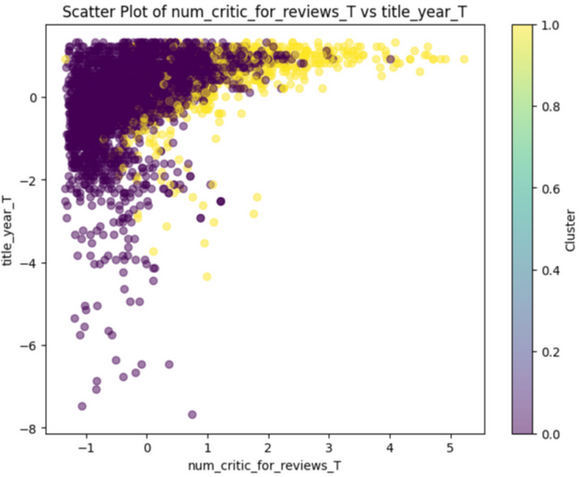


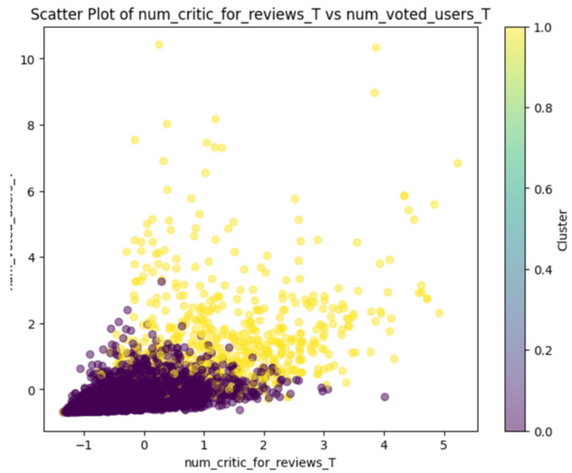


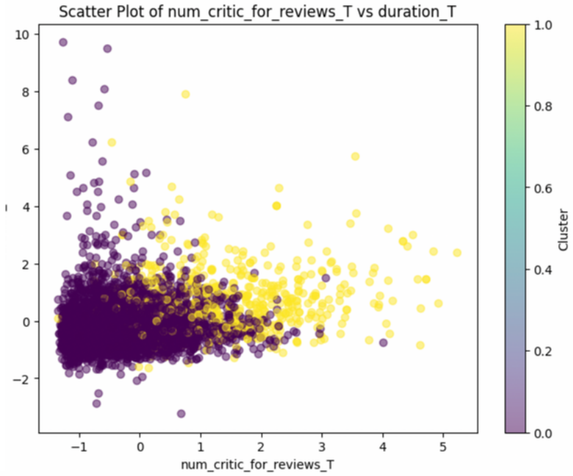


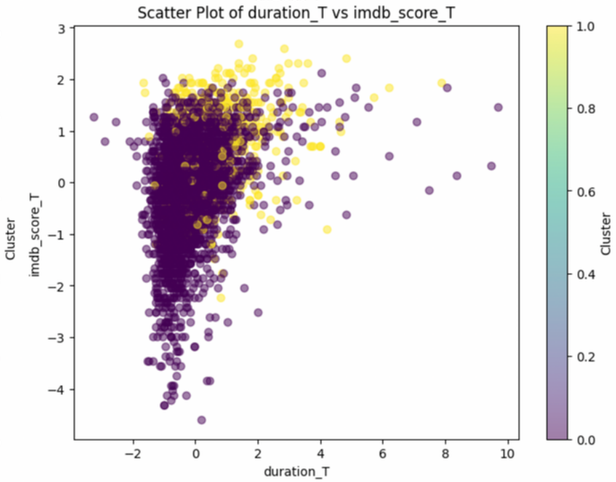












CONTINGENCY MATRIX

```python
ful = pd.read_csv("movies.csv")
full = ful.fillna(0)
full.to_numpy()
numeric_columns = full.select_dtypes(include=[np.number])
normalized_columns = scale(numeric_columns)
normalized_df = pd.DataFrame(normalized_columns, columns=numeric_columns.columns)

full_subset = full.iloc[:500]
normalized_df.to_numpy()
normalized_subset = normalized_df.iloc[:500]
model = KMeans(n_clusters=3)
model = model.fit(scale(normalized_subset))
print(model.labels_)

labels_pred = model.fit_predict(scale(normalized_subset))
labels_true = labels_pred

contingency_matrix = contingency_matrix(labels_true, labels_pred)

print("Contingency Matrix:")
print(contingency_matrix)
```

```
[2 2 2 2 1 0 2 0 2 2 2 2 0 2 2 2 0 2 2 0 2 2 0 2 0 2 2 2 0 2 2 2 2 2 0 0 0
 0 2 2 0 0 0 2 0 2 2 2 2 0 2 0 2 0 2 0 0 0 0 0 0 2 2 0 0 2 2 0 0 2 2 0 0 0
 2 0 0 0 2 2 2 2 0 0 0 0 2 0 0 0 0 0 0 0 0 2 2 2 0 2 0 2 2 2 0 2 2 0 0 2 0
 0 0 0 2 2 0 0 2 0 2 0 0 2 0 0 0 2 2 0 2 0 0 0 2 0 0 0 0 0 0 2 0 0 0 0 0 2
 0 0 0 0 0 0 0 2 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 1 0 2 0 0 0 0 0
 0 2 2 0 2 0 0 0 2 0 2 0 0 0 0 0 2 0 0 0 2 0 2 0 0 0 2 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 2 0 0 0 0 0 0 0 2 2 2 0 0 2 2 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 2 0
 0 1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 2 1 0 0 0 2 0 2 0 0 0 2 0 0 0 0 0 0
 2 0 0 2 0 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 2 0 0 0 0 0 0 0 0 2 0 0 0 0 0 2
 0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0
 0 0 0 0 0 0 0 0 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 1 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 0 0 0 2 0 0 2 0 2 0 0 0 0
 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 1 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Contingency Matrix:
[[304   0   0]
 [  0 130   0]
 [  0   0  66]]
```

*Similar to many of the K cluster graphs, but I could not find the exact one.

Files Included