project-320210207

December 16, 2023

#

AID311 Math for Data Science Project Phase 1

0.1 Movie Production Analysis & Prediction

0.2 Student Credentials:

- Name: Yousef Ibrahim Gomaa Mahmoud
- ID: 320210207
- Section: AID 3
- Group: 1
- E-mail: yousef.gomaa@ejust.edu.eg

0.3 Problem Introduction:

In this notebook, we explore various aspects of movie production and promotion, aiming to uncover patterns and insights that can contribute to the success of a film. Each column provides valuable information that could hold the key to understanding the dynamics of a movie's performance.

The task is to analyze this dataset and build a predictive model that can help stakeholders make informed decisions about movie production and marketing strategies.

The models used being: - Naive Bayesian - Bayesian Belief Network - Decision Tree (Entropy, and error estimation) - LDA - PCA - K-NN (Different distances)

But, before doing so, some preprocessing operations may be required in order to prepare the data so that it could be used in the machine learning process.

Finally, each model is evaluated in terms of accuracy and other metrics to find the best suited one for said task.

[ ]:
```
# Libraries needed:
# Pre-processing:
# Pandas
import pandas as pd

# Data Visualization:
# Standard Visualization Packages
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='whitegrid')
```

0.4 Data Description:

```
[ ]: df = pd.read_csv("Movie.xls")
```

```
[ ]: df.head()
```

```
[ ]:    Marketing expense  Production expense  Multiplex coverage     Budget  \
     0            20.1264               59.62               0.462  36524.125
     1            20.5462               69.14               0.531  35668.655
     2            20.5458               69.14               0.531  39912.675
     3            20.6474               59.36               0.542  38873.890
     4            21.3810               59.36               0.542  39701.585

        Movie_length  Lead_ Actor_Rating  Lead_Actress_rating  Director_rating  \
     0         138.7               7.825                8.095            7.910
     1         152.4               7.505                7.650            7.440
     2         134.6               7.485                7.570            7.495
     3         119.3               6.895                7.035            6.920
     4         127.7               6.920                7.070            6.815

        Producer_rating  Critic_rating  Trailer_views 3D_available  Time_taken  \
     0            7.995           7.94         527367          YES      109.60
     1            7.470           7.44         494055           NO      146.64
     2            7.515           7.44         547051           NO      147.88
     3            7.020           8.26         516279          YES      185.36
     4            7.070           8.26         531448           NO      176.48

        Twitter_hastags     Genre  Avg_age_actors  Num_multiplex  Collection
     0          223.840  Thriller              23            494       48000
     1          243.456     Drama              42            462       43200
     2         2022.400    Comedy              38            458       69400
     3          225.344     Drama              45            472       66800
     4          225.792     Drama              55            395       72400
```

The dataset comprises key parameters such as: - Marketing Expense, - Production Expense, - Multiplex Coverage, - Budget, - Movie Length, - Lead Actor Rating, - Lead Actress Rating, - Director Rating, - Producer Rating, - Critic Rating, - Trailer Views, - 3D Availability, - Time Taken, - Twitter Hashtags, - Genre, - Average Age of Actors, - Number of Multiplexes, - and Collection.

0.4.1 Data Preprocessing:

```
[ ]: df.shape
```

```
[ ]: (506, 18)
```

```
[ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Marketing expense    506 non-null    float64
 1   Production expense   506 non-null    float64
 2   Multiplex coverage   506 non-null    float64
 3   Budget               506 non-null    float64
 4   Movie_length         506 non-null    float64
 5   Lead_ Actor_Rating   506 non-null    float64
 6   Lead_Actress_rating  506 non-null    float64
 7   Director_rating      506 non-null    float64
 8   Producer_rating      506 non-null    float64
 9   Critic_rating        506 non-null    float64
 10  Trailer_views        506 non-null    int64
 11  3D_available         506 non-null    object
 12  Time_taken           494 non-null    float64
 13  Twitter_hastags      506 non-null    float64
 14  Genre                506 non-null    object
 15  Avg_age_actors       506 non-null    int64
 16  Num_multiplex        506 non-null    int64
 17  Collection           506 non-null    int64
dtypes: float64(12), int64(4), object(2)
memory usage: 71.3+ KB
```

[ ]:
```
df.describe().T
```

[ ]:

	count	mean	std	min
Marketing expense	506.0	92.270471	172.030902	20.1264
Production expense	506.0	77.273557	13.720706	55.9200
Multiplex coverage	506.0	0.445305	0.115878	0.1290
Budget	506.0	34911.144022	3903.038232	19781.3550
Movie_length	506.0	142.074901	28.148861	76.4000
Lead_ Actor_Rating	506.0	8.014002	1.054266	3.8400
Lead_Actress_rating	506.0	8.185613	1.054290	4.0350
Director_rating	506.0	8.019664	1.059899	3.8400
Producer_rating	506.0	8.190514	1.049601	4.0300
Critic_rating	506.0	7.810870	0.659699	6.6000
Trailer_views	506.0	449860.715415	68917.763145	212912.0000
Time_taken	494.0	157.391498	31.295161	0.0000
Twitter_hastags	506.0	260.832095	104.779133	201.1520
Avg_age_actors	506.0	39.181818	12.513697	3.0000
Num_multiplex	506.0	545.043478	106.332889	333.0000
Collection	506.0	45057.707510	18364.351764	10000.0000

\

	25%	50%	75%	max

```
Marketing expense          21.64090      25.1302      93.54165     1799.524
Production expense         65.38000      74.3800      91.20000      110.480
Multiplex coverage          0.37600       0.4620       0.55100        0.615
Budget                  32693.95250   34488.2175   36793.54250    48772.900
Movie_length              118.52500     151.0000     167.57500      173.500
Lead_ Actor_Rating          7.31625       8.3075       8.86500        9.435
Lead_Actress_rating         7.50375       8.4950       9.03000        9.540
Director_rating             7.29625       8.3125       8.88375        9.425
Producer_rating             7.50750       8.4650       9.03000        9.635
Critic_rating               7.20000       7.9600       8.26000        9.400
Trailer_views          409128.00000  462460.0000  500247.50000   567784.000
Time_taken                132.30000     160.0000     181.89000      217.520
Twitter_hastags           223.79600     254.4000     283.41600     2022.400
Avg_age_actors             28.00000      39.0000      50.00000       60.000
Num_multiplex             465.00000     535.5000     614.75000      868.000
Collection              34050.00000   42400.0000   50000.00000   100000.000
```

[ ]:
```
df.skew(numeric_only=True)
```

[ ]:
```
Marketing expense        5.223149
Production expense       0.295022
Multiplex coverage      -0.729308
Budget                   0.403612
Movie_length            -0.598963
Lead_ Actor_Rating      -1.010577
Lead_Actress_rating     -1.007492
Director_rating         -1.003848
Producer_rating         -1.004680
Critic_rating            0.176139
Trailer_views           -0.843831
Time_taken              -0.473481
Twitter_hastags         13.790552
Avg_age_actors           0.012971
Num_multiplex            0.534221
Collection               1.110912
dtype: float64
```

[ ]:
```
df.kurtosis(numeric_only=True)
```

[ ]:
```
Marketing expense        37.130509
Production expense       -1.233540
Multiplex coverage       -0.064667
Budget                    1.891500
Movie_length             -0.967716
Lead_ Actor_Rating        0.497728
Lead_Actress_rating       0.472927
Director_rating           0.458464
```

```
Producer_rating         0.503435
Critic_rating          -0.751736
Trailer_views           0.489241
Time_taken              1.113759
Twitter_hastags       214.232263
Avg_age_actors         -1.199662
Num_multiplex          -0.121132
Collection              1.516783
dtype: float64
```

[ ]:
```
df.isna().sum()
```

[ ]:
```
Marketing expense       0
Production expense      0
Multiplex coverage      0
Budget                  0
Movie_length            0
Lead_ Actor_Rating      0
Lead_Actress_rating     0
Director_rating         0
Producer_rating         0
Critic_rating           0
Trailer_views           0
3D_available            0
Time_taken             12
Twitter_hastags         0
Genre                   0
Avg_age_actors          0
Num_multiplex           0
Collection              0
dtype: int64
```

[ ]:
```
df.duplicated().sum()
```

[ ]:
```
0
```

[ ]:
```
df[df['Time_taken'].isna()]
```

[ ]:
```
     Marketing expense  Production expense  Multiplex coverage     Budget  \
20             45.0358               71.28               0.462  30941.350
58             23.0890               65.26               0.547  34135.475
60             22.9864               65.26               0.547  31891.255
104            22.7920               72.12               0.480  34257.685
105            22.6524               72.12               0.480  32502.305
215            23.9604               76.18               0.511  34341.010
260            30.8022               62.94               0.353  40012.665
359           105.2262               91.20               0.230  33952.160
```

403	516.0340	91.20	0.307	29713.695
416	236.6840	91.20	0.321	37674.010
440	461.0220	91.20	0.260	32318.990
496	25.7920	74.38	0.415	29941.450

	Movie_length	Lead_ Actor_Rating	Lead_Actress_rating	Director_rating	\
20	171.6	8.035	8.205	7.955	
58	102.7	6.010	6.115	5.965	
60	139.7	6.335	6.420	6.235	
104	163.5	8.685	8.875	8.660	
105	170.2	8.905	9.025	8.935	
215	115.9	7.925	8.095	8.020	
260	155.3	8.940	9.025	8.815	
359	154.8	8.610	8.810	8.720	
403	169.5	9.125	9.310	9.060	
416	164.3	9.050	9.230	8.980	
440	165.9	8.985	9.170	9.020	
496	146.4	8.570	8.695	8.510	

	Producer_rating	Critic_rating	Trailer_views	3D_available	Time_taken	\
20	8.210	7.80	371051	YES	NaN	
58	6.280	7.06	480067	NO	NaN	
60	6.560	7.06	465689	NO	NaN	
104	8.935	6.82	432081	YES	NaN	
105	8.925	6.82	430817	YES	NaN	
215	8.065	7.28	456943	YES	NaN	
260	8.995	9.40	483080	YES	NaN	
359	8.845	6.96	437945	NO	NaN	
403	9.100	6.96	384237	YES	NaN	
416	9.100	7.96	335532	YES	NaN	
440	9.095	7.96	360183	NO	NaN	
496	8.630	7.16	380129	NO	NaN	

	Twitter_hastags	Genre	Avg_age_actors	Num_multiplex	Collection
20	302.176	Action	44	484	27200
58	283.728	Comedy	22	438	46600
60	222.992	Thriller	30	439	37400
104	203.216	Comedy	20	458	40200
105	263.120	Comedy	57	515	39000
215	244.000	Drama	30	480	50000
260	225.408	Drama	21	681	67600
359	283.616	Drama	26	743	45200
403	301.328	Thriller	40	677	16600
416	201.200	Thriller	35	647	15000
440	241.680	Comedy	38	753	21000
496	243.152	Thriller	44	611	39400

[ ]:
```
df.fillna(df.mean(numeric_only=True), inplace=True) # Fill numerical data with␣
↪average. (i.e: 'Time_taken')
df.isna().sum()
# No more missing values.
```

[ ]:
```
Marketing expense      0
Production expense     0
Multiplex coverage     0
Budget                 0
Movie_length           0
Lead_ Actor_Rating     0
Lead_Actress_rating    0
Director_rating        0
Producer_rating        0
Critic_rating          0
Trailer_views          0
3D_available           0
Time_taken             0
Twitter_hastags        0
Genre                  0
Avg_age_actors         0
Num_multiplex          0
Collection             0
dtype: int64
```

[ ]:
```
df.cov(numeric_only=True)
```

[ ]:
```
                     Marketing expense  Production expense  \
Marketing expense         2.959463e+04          959.693552
Production expense        9.596936e+02          188.257770
Multiplex coverage       -8.391878e+00           -1.214147
Budget                   -1.472117e+05       -20975.197451
Movie_length              1.708106e+03          249.027806
Lead_ Actor_Rating        6.892832e+01           10.219447
Lead_Actress_rating       6.888692e+01           10.241019
Director_rating           6.929999e+01           10.289825
Producer_rating           6.797539e+01           10.164691
Critic_rating            -2.099370e+01           -2.277047
Trailer_views            -5.257621e+06      -559471.159699
Time_taken                1.366738e+02            6.691969
Twitter_hastags           2.436630e+02           -1.205652
Avg_age_actors            1.274508e+02            9.582362
Num_multiplex             7.011489e+03         1032.301192
Collection               -1.230783e+06      -122144.467076

                     Multiplex coverage        Budget  Movie_length  \
Marketing expense             -8.391878 -1.472117e+05  1.708106e+03
```

```
Production expense          -1.214147 -2.097520e+04  2.490278e+02
Multiplex coverage           0.013428  1.366722e+02 -2.385927e+00
Budget                     136.672162  1.523371e+07 -2.639697e+04
Movie_length                -2.385927 -2.639697e+04  7.923584e+02
Lead_ Actor_Rating          -0.093895 -8.577976e+02  2.216542e+01
Lead_Actress_rating         -0.094036 -8.393672e+02  2.215373e+01
Director_rating             -0.094467 -8.352540e+02  2.228732e+01
Producer_rating             -0.093028 -8.414358e+02  2.206151e+01
Critic_rating                0.011127  5.982895e+02 -4.045057e+00
Trailer_views             4642.967137  1.620754e+08 -1.143251e+06
Time_taken                   0.127252  4.880402e+03 -1.725097e+01
Twitter_hastags              0.059270  1.254414e+04  2.766426e+01
Avg_age_actors              -0.133556 -3.159743e+03  2.648814e+01
Num_multiplex              -11.280366 -1.173664e+05  2.017071e+03
Collection                 913.558764  4.990880e+07 -1.954011e+05

                     Lead_ Actor_Rating  Lead_Actress_rating  Director_rating  \
Marketing expense             68.928324            68.886921        69.299991
Production expense            10.219447            10.241019        10.289825
Multiplex coverage            -0.093895            -0.094036        -0.094467
Budget                      -857.797596          -839.367228      -835.254004
Movie_length                  22.165423            22.153726        22.287321
Lead_ Actor_Rating             1.111477             1.109173         1.114884
Lead_Actress_rating            1.109173             1.111528         1.115314
Director_rating                1.114884             1.115314         1.123385
Producer_rating                1.100001             1.099948         1.105936
Critic_rating                 -0.118219            -0.115450        -0.116516
Trailer_views             -35621.646928        -35424.037429    -35533.287779
Time_taken                     1.240404             1.237966         1.175922
Twitter_hastags                1.597638             1.131100         1.119079
Avg_age_actors                 0.485419             0.501403         0.550022
Num_multiplex                 79.181895            79.399617        79.946757
Collection                 -4866.461104         -4829.873048     -4800.883544

                     Producer_rating  Critic_rating  Trailer_views  \
Marketing expense          67.975385     -20.993696  -5.257621e+06
Production expense         10.164691      -2.277047  -5.594712e+05
Multiplex coverage         -0.093028       0.011127   4.642967e+03
Budget                   -841.435814     598.289460   1.620754e+08
Movie_length               22.061512      -4.045057  -1.143251e+06
Lead_ Actor_Rating          1.100001      -0.118219  -3.562165e+04
Lead_Actress_rating         1.099948      -0.115450  -3.542404e+04
Director_rating             1.105936      -0.116516  -3.553329e+04
Producer_rating             1.101663      -0.115637  -3.529360e+04
Critic_rating              -0.115637       0.435203   1.039518e+04
Trailer_views          -35293.602883   10395.175892   4.749658e+09
Time_taken                  0.931302      -0.301125   1.587972e+05
```

```
Twitter_hastags              0.643354      -1.635117  -4.841236e+04
Avg_age_actors               0.427421      -0.411089  -4.288429e+04
Num_multiplex               78.517631      -9.032870  -3.987284e+06
Collection               -4784.120800    4134.690314   9.114048e+08

                       Time_taken  Twitter_hastags  Avg_age_actors  \
Marketing expense      136.673773       243.662995      127.450800
Production expense       6.691969        -1.205652        9.582362
Multiplex coverage       0.127252         0.059270       -0.133556
Budget                4880.401553     12544.144993    -3159.743000
Movie_length           -17.250975        27.664256       26.488137
Lead_ Actor_Rating       1.240404         1.597638        0.485419
Lead_Actress_rating      1.237966         1.131100        0.501403
Director_rating          1.175922         1.119079        0.550022
Producer_rating          0.931302         0.643354        0.427421
Critic_rating           -0.301125        -1.635117       -0.411089
Trailer_views       158797.151008    -48412.361256   -42884.286769
Time_taken             956.114511       -20.676163       27.878378
Twitter_hastags        -20.676163     10978.666640       -6.345839
Avg_age_actors          27.878378        -6.345839      156.592619
Num_multiplex         -186.437732        69.691782      104.867327
Collection           62466.087626     44491.615901   -10898.829883

                    Num_multiplex    Collection
Marketing expense    7.011489e+03 -1.230783e+06
Production expense   1.032301e+03 -1.221445e+05
Multiplex coverage  -1.128037e+01  9.135588e+02
Budget              -1.173664e+05  4.990880e+07
Movie_length         2.017071e+03 -1.954011e+05
Lead_ Actor_Rating   7.918189e+01 -4.866461e+03
Lead_Actress_rating  7.939962e+01 -4.829873e+03
Director_rating      7.994676e+01 -4.800884e+03
Producer_rating      7.851763e+01 -4.784121e+03
Critic_rating       -9.032870e+00  4.134690e+03
Trailer_views       -3.987284e+06  9.114048e+08
Time_taken          -1.864377e+02  6.246609e+04
Twitter_hastags      6.969178e+01  4.449162e+04
Avg_age_actors       1.048673e+02 -1.089883e+04
Num_multiplex        1.130668e+04 -7.649419e+05
Collection          -7.649419e+05  3.372494e+08
```

Conclusion:

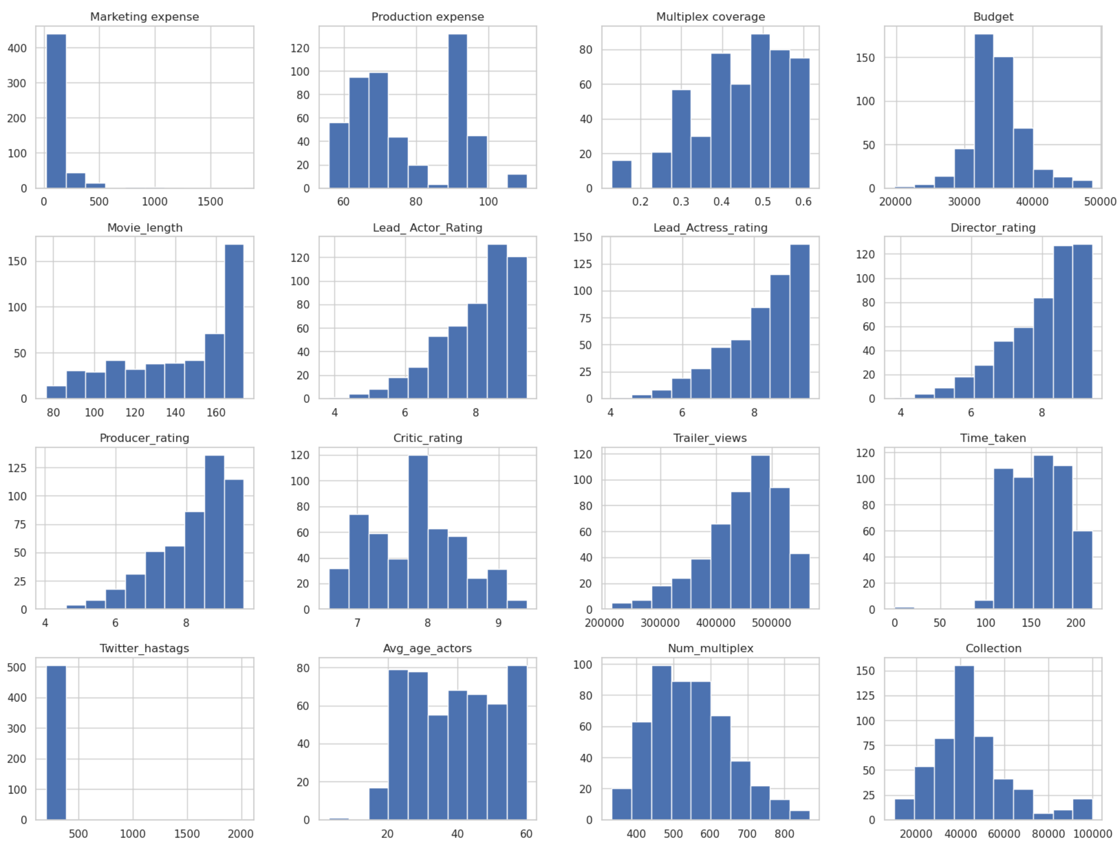
- Data types are correctly casted.
- Missing values found in the data given, which are in turn replaced with mean values since it is float64 type.
- No duplicated records.

0.4.2 Data Visualization:

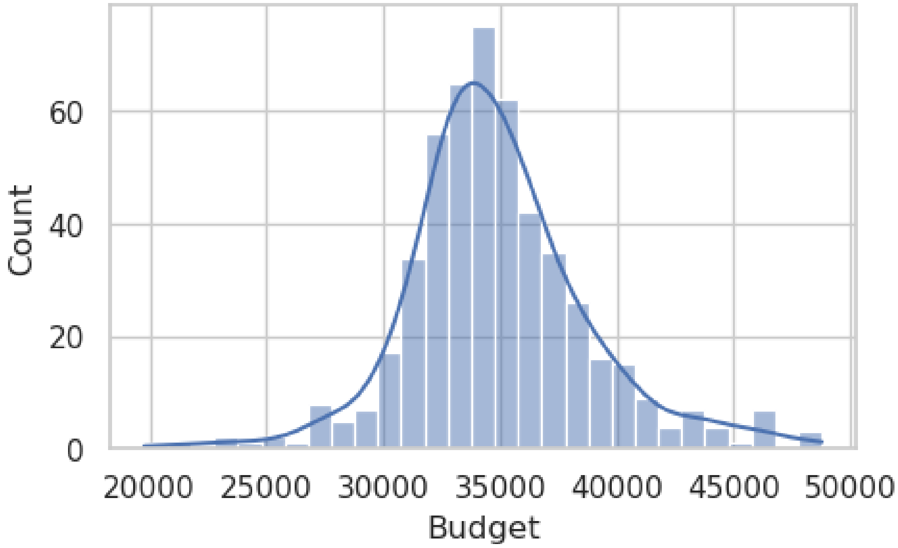
0.4.3 Numerical Data Visualization:

[ ]:
```
df.hist(figsize=(20,15))
```

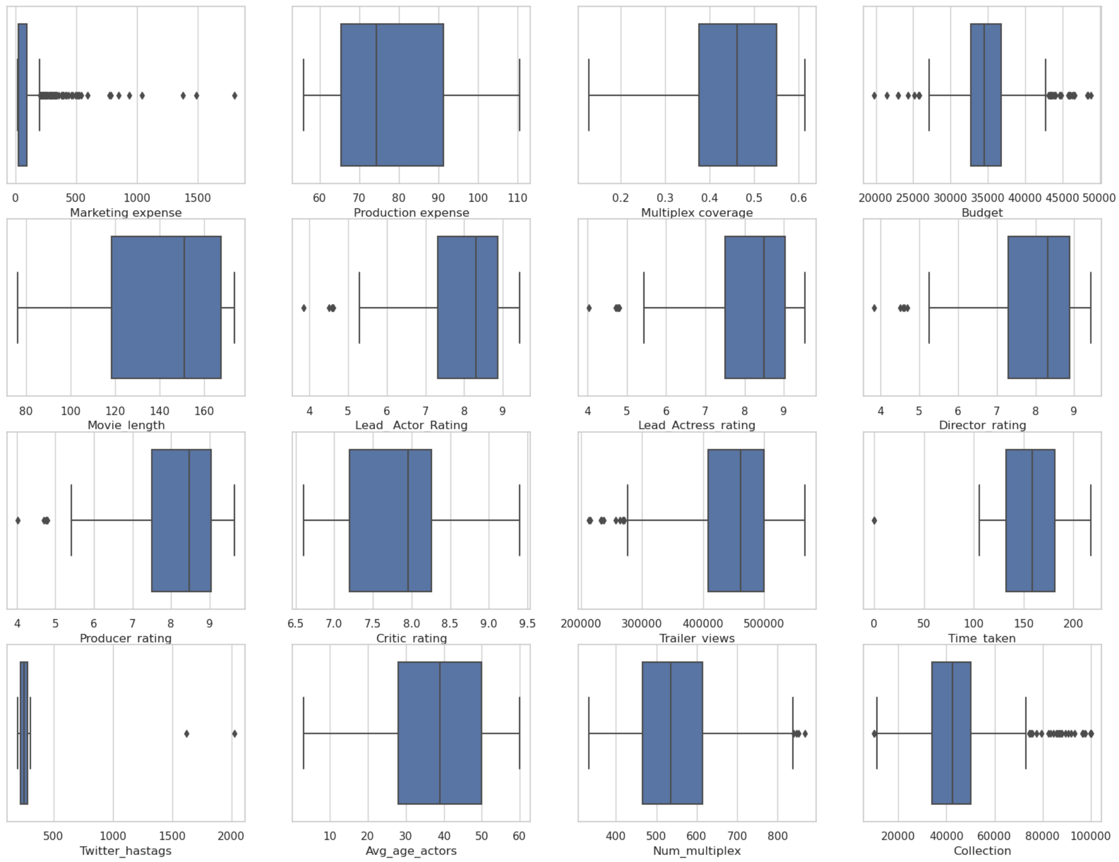
[ ]:
```
array([[<Axes: title={'center': 'Marketing expense'}>,
        <Axes: title={'center': 'Production expense'}>,
        <Axes: title={'center': 'Multiplex coverage'}>,
        <Axes: title={'center': 'Budget'}>],
       [<Axes: title={'center': 'Movie_length'}>,
        <Axes: title={'center': 'Lead_ Actor_Rating'}>,
        <Axes: title={'center': 'Lead_Actress_rating'}>,
        <Axes: title={'center': 'Director_rating'}>],
       [<Axes: title={'center': 'Producer_rating'}>,
        <Axes: title={'center': 'Critic_rating'}>,
        <Axes: title={'center': 'Trailer_views'}>,
        <Axes: title={'center': 'Time_taken'}>],
       [<Axes: title={'center': 'Twitter_hastags'}>,
        <Axes: title={'center': 'Avg_age_actors'}>,
        <Axes: title={'center': 'Num_multiplex'}>,
        <Axes: title={'center': 'Collection'}>]], dtype=object)
```



[ ]:
```
plt.figure(figsize=(5,3))
sns.histplot(data=df, x='Budget', kde = True)
plt.show()
```

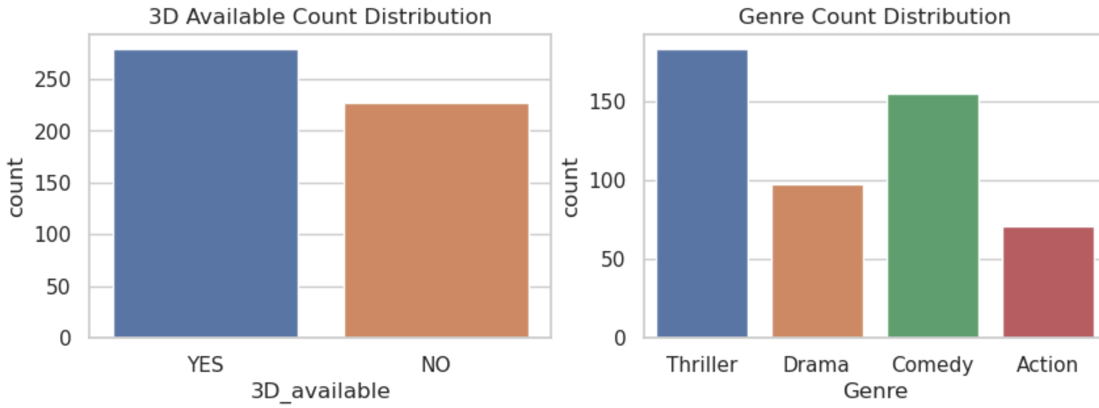


[ ]:
```
x = df.drop(['3D_available','Genre'],axis = 1)
# 16 columns without the categorical ones ^^
plt.figure(figsize=(20,15))
for idx,i in enumerate(x.columns):
    plt.subplot(4, 4, idx+1)
    sns.boxplot(x = i, data = x)
    plt.xlabel(i)
plt.show()
```



0.4.4 Categorical Data Visualization:

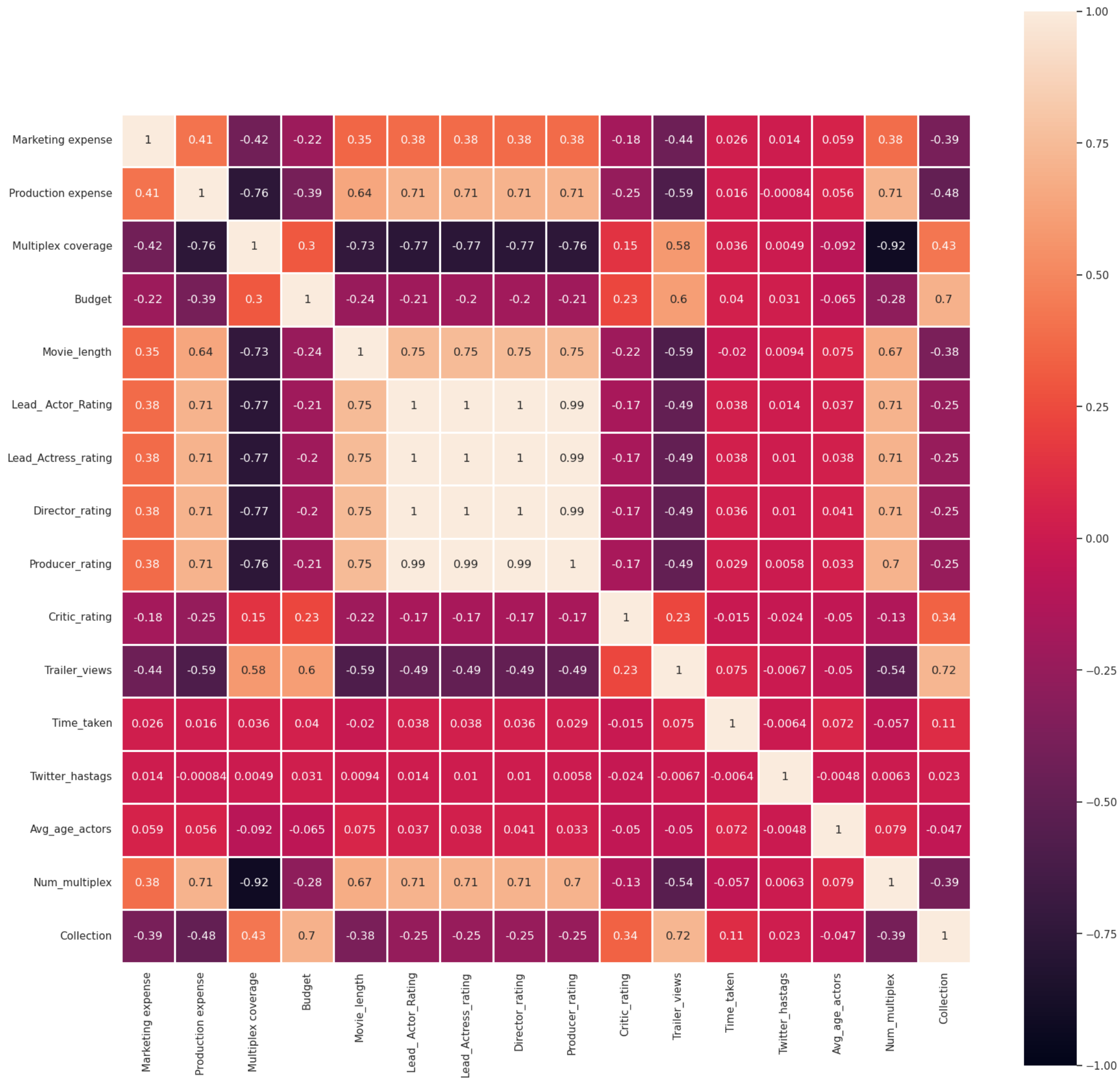
[ ]:
```
plt.figure(figsize=(10, 3))
plt.subplot(1,2,1)
sns.countplot(x = '3D_available',data=df)
plt.title("3D Available Count Distribution")
plt.subplot(1,2,2)
sns.countplot(x = 'Genre',data=df)
plt.title("Genre Count Distribution")
plt.show()
```



0.4.5 Correlation Heatmap:

[ ]:
```
plt.figure(figsize=(20, 20))
# Numeric correlation
sns.heatmap(df.corr(numeric_only=True), linewidths=1, vmin=-1, vmax=1,␣
↪annot=True, square=True)
```

[ ]: <Axes: >



[ ]:
```
df_num = df.copy()
```

[ ]:
```
# Converting/encoding data to numerical values.
# (e.g: 3D_Available, Genre)
from sklearn.preprocessing import LabelEncoder

laben = LabelEncoder()
obj = df_num.select_dtypes(include='object')
non_obj = df_num.select_dtypes(exclude='object')
for i in range (0, obj.shape[1]) :
    obj[obj.columns[i]] = laben.fit_transform(obj[obj.columns[i]])
df_num = pd.concat([non_obj, obj], axis = 1)
df_num.head(5)
```

```
[ ]:    Marketing expense  Production expense  Multiplex coverage     Budget  \
0            20.1264               59.62               0.462  36524.125
1            20.5462               69.14               0.531  35668.655
2            20.5458               69.14               0.531  39912.675
3            20.6474               59.36               0.542  38873.890
4            21.3810               59.36               0.542  39701.585

   Movie_length  Lead_ Actor_Rating  Lead_Actress_rating  Director_rating  \
0         138.7               7.825                8.095            7.910
1         152.4               7.505                7.650            7.440
2         134.6               7.485                7.570            7.495
3         119.3               6.895                7.035            6.920
4         127.7               6.920                7.070            6.815

   Producer_rating  Critic_rating  Trailer_views  Time_taken  Twitter_hastags  \
0            7.995           7.94         527367      109.60          223.840
1            7.470           7.44         494055      146.64          243.456
2            7.515           7.44         547051      147.88         2022.400
3            7.020           8.26         516279      185.36          225.344
4            7.070           8.26         531448      176.48          225.792

   Avg_age_actors  Num_multiplex  Collection  3D_available  Genre
0              23            494       48000             1      3
1              42            462       43200             0      2
2              38            458       69400             0      1
3              45            472       66800             1      2
4              55            395       72400             0      2
```

0.4.6 Machine Learning Models:

0.4.7 Preparation:

```
[ ]: # Machine Learning:
# Machine Learning Tools
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_predict,␣
↪StratifiedKFold
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, mean_absolute_error,␣
↪mean_squared_error,\
    recall_score, precision_score, f1_score, roc_curve, auc, confusion_matrix
```

```
[ ]: X = df_num.drop(['3D_available','Genre'],axis=1)
     Y = df_num['3D_available']
```

```
[ ]: x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,␣
      ↪random_state=43)
```

```
[ ]: sc = StandardScaler()
     x_train = sc.fit_transform(x_train)
     x_test = sc.transform(x_test)
```

0.4.8 PCA

```
[ ]: pca = PCA(n_components=3)
     x_train_pca = pca.fit_transform(x_train)
     x_test_pca = pca.transform(x_test)
```

0.4.9 LDA

```
[ ]: lda = LinearDiscriminantAnalysis(n_components=1)
     x_train_lda = lda.fit_transform(x_train, y_train)
     x_test_lda = lda.transform(x_test)
```

0.4.10 KNN with PCA

```
[ ]: knn_pca = KNeighborsClassifier(n_neighbors=2)
     knn_pca.fit(x_train_pca, y_train)
     y_pred1 = knn_pca.predict(x_test_pca)
     accuracy_pca =  accuracy_score(y_test, y_pred1)
     print('Accuracy PCA: '+str(accuracy_pca*100)+'%')
```

```
Accuracy PCA: 50.98039215686274%
```

```
[ ]: y_pred_pca = cross_val_predict(knn_pca, x_train_pca, y_train,␣
      ↪cv=StratifiedKFold(n_splits=5,
                                                                        ␣
      ↪shuffle=True,
                                                                        ␣
      ↪random_state=43))
     fpr_pca, tpr_pca, _ = roc_curve(y_train, y_pred_pca, pos_label=1)
     roc_auc_pca = auc(fpr_pca, tpr_pca)
```

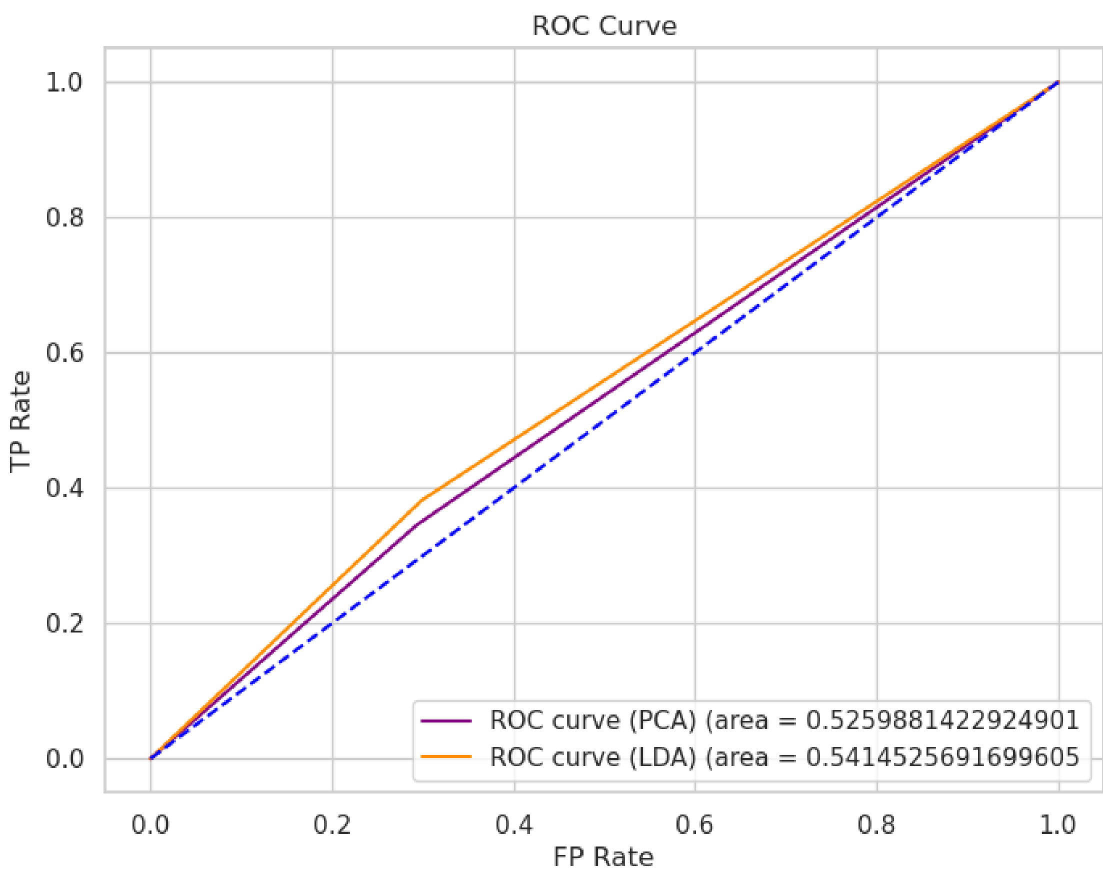
0.4.11 KNN with LDA

```
[ ]: knn_lda = KNeighborsClassifier(n_neighbors=2)
     knn_lda.fit(x_train_lda, y_train)
     y_pred2 = knn_lda.predict(x_test_lda)
     accuracy_lda =  accuracy_score(y_test, y_pred2)
     print('Accuracy LDA: '+str(accuracy_lda*100)+'%')
```

```
Accuracy LDA: 49.01960784313725%
```

[ ]:
```
y_pred_lda = cross_val_predict(knn_lda, x_train_lda, y_train,␣
 ↪cv=StratifiedKFold(n_splits=5,                                                       ␣

 ↪shuffle=True,                                                                         ␣

 ↪random_state=43))
fpr_lda, tpr_lda, threshold = roc_curve(y_train, y_pred_lda, pos_label=1)
roc_auc_lda = auc(fpr_lda, tpr_lda)
```

[ ]:
```
plt.figure(figsize=(8, 6))
plt.plot(fpr_pca, tpr_pca, color='purple', label='ROC curve (PCA) (area =␣
 ↪'+str(roc_auc_pca))
plt.plot(fpr_lda, tpr_lda, color='darkorange', label='ROC curve (LDA) (area =␣
 ↪'+str(roc_auc_lda))
plt.plot([0, 1], [0, 1], color='blue', linestyle='--')
plt.xlabel('FP Rate')
plt.ylabel('TP Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

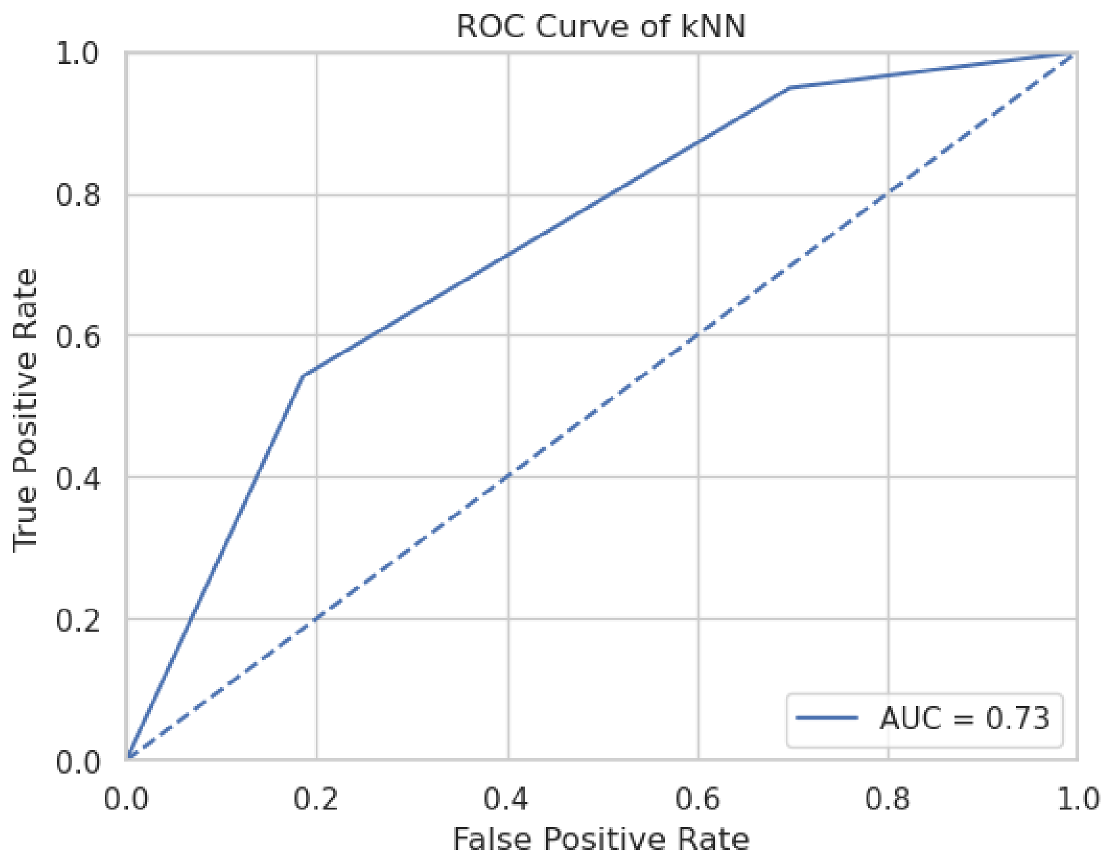


0.4.12 Standard KNN (Classifier)

[ ]:
```
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(x_train, y_train)
y_pred3 = knn.predict(x_test)
print(y_pred3[:10])
print('Accuracy:', accuracy_score(y_test, y_pred3))
```

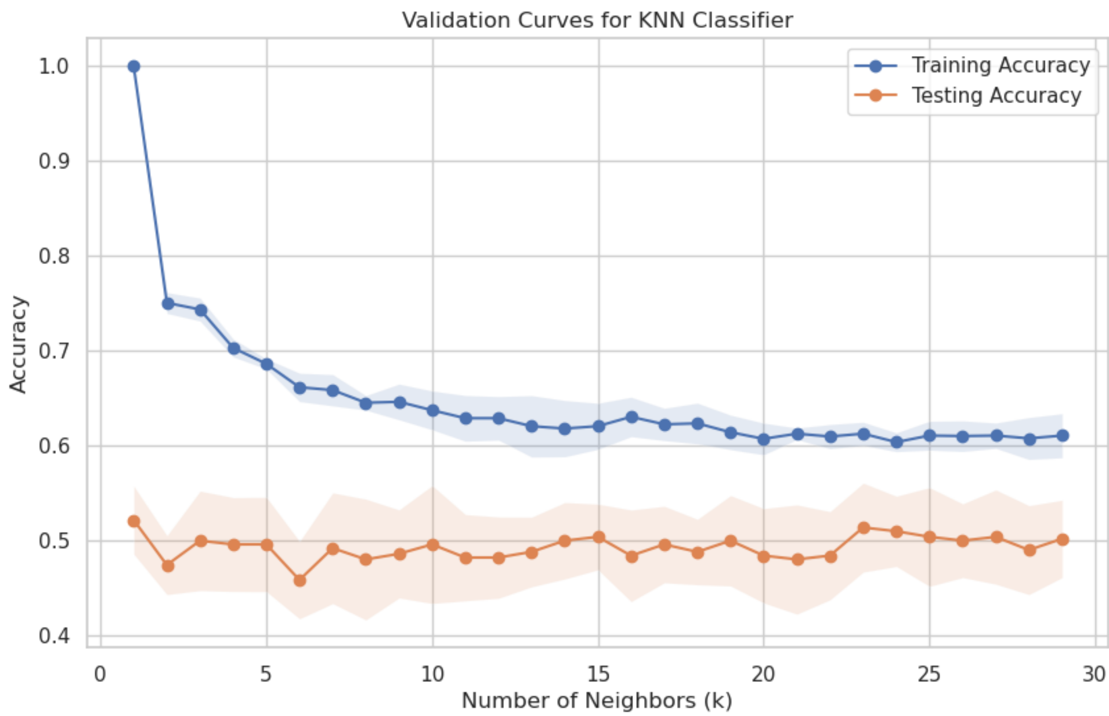
```
[0 0 1 1 0 0 1 0 0 1]
Accuracy: 0.6568627450980392
```

[ ]:
```
y_scores = knn.predict_proba(x_test)
fpr, tpr, threshold = roc_curve(y_test, y_scores[:, 1])
roc_auc = auc(fpr, tpr)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of kNN')
plt.show()
```



[ ]:
```
from sklearn.model_selection import validation_curve
param_range = np.arange(1, 30)

# Use validation_curve to generate validation curves for a KNN classifier
train_scores, test_scores = validation_curve(
    KNeighborsClassifier(), X, Y, param_name='n_neighbors',␣
 ↪param_range=param_range,
    cv=5, scoring='accuracy'
)

# Calculate mean and standard deviation
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot validation curves
plt.figure(figsize=(10, 6))
plt.plot(param_range, train_mean, label='Training Accuracy', marker='o')
```

```
plt.fill_between(param_range, train_mean - train_std, train_mean + train_std,␣
↪alpha=0.15)
plt.plot(param_range, test_mean, label='Testing Accuracy', marker='o')
plt.fill_between(param_range, test_mean - test_std, test_mean + test_std,␣
↪alpha=0.15)
plt.xlabel('Number of Neighbors (k)')
plt.ylabel('Accuracy')
plt.title('Validation Curves for KNN Classifier')
plt.legend()
plt.show()
```

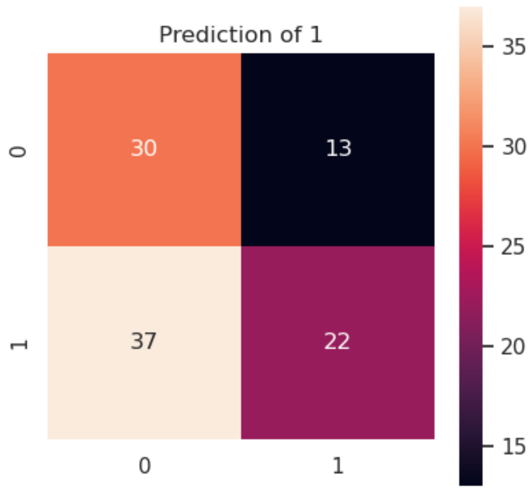


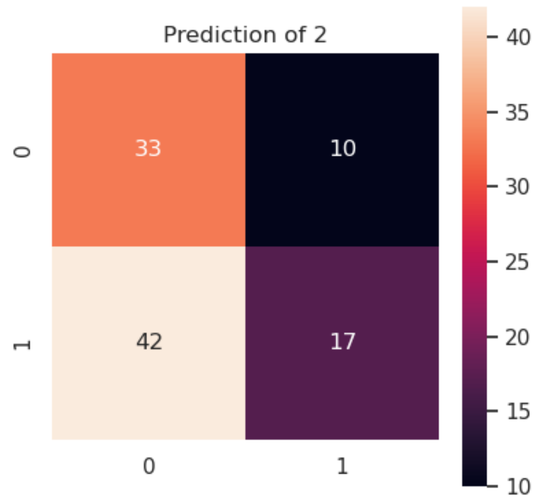
0.4.13 Confusion Matrix Heatmap

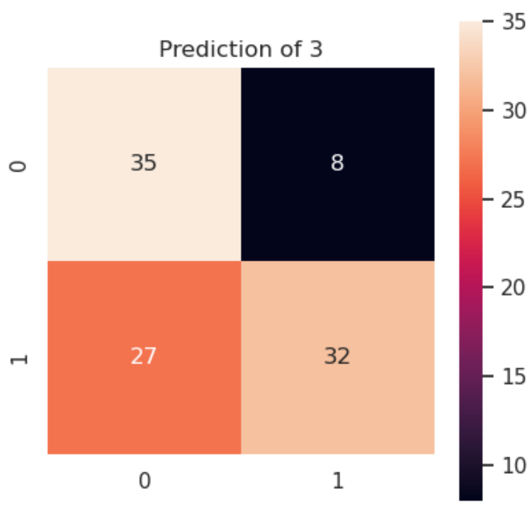
[ ]:
```
plt.figure(figsize=(10,10))
for i, pred in enumerate([y_pred1,y_pred2,y_pred3]):
    cf = confusion_matrix(y_test, pred)
    print("Confusion Matrix for Prediction: "+str(i+1))
    print(cf)
    plt.subplot(2, 2, i + 1)
    sns.heatmap(cf, annot=True, square=True)
    plt.title("Prediction of "+str(i+1))
plt.show()
```

```
Confusion Matrix for Prediction: 1
```

```
[[30 13]
 [37 22]]
Confusion Matrix for Prediction: 2
[[33 10]
 [42 17]]
Confusion Matrix for Prediction: 3
[[35  8]
 [27 32]]
```







0.5 Regression:

0.5.1 Standard KNN (Regressor)

[ ]:
```
X2 = df_num.drop(['3D_available','Genre'],axis=1)
Y2 = df_num['Collection']
x_train, x_test, y_train, y_test = train_test_split(X2, Y2, test_size=0.2,␣
 ↪random_state=43)
```

[ ]:
```
from sklearn.neighbors import KNeighborsRegressor
# Euclidean distance metric
knn = KNeighborsRegressor(n_neighbors=7, metric='euclidean')
knn.fit(x_train, y_train)
y_pred3 = knn.predict(x_test)

print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred3))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred3))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred3)))
```

```
Mean Absolute Error: 2515.6862745098047
Mean Squared Error: 14299519.80792317
Root Mean Squared Error: 3781.470588001865
```

[ ]:
```
# Manhattan distance metric
knn = KNeighborsRegressor(n_neighbors=7, metric='manhattan')
knn.fit(x_train, y_train)
y_pred4 = knn.predict(x_test)
print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred4))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred4))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred4)))
```

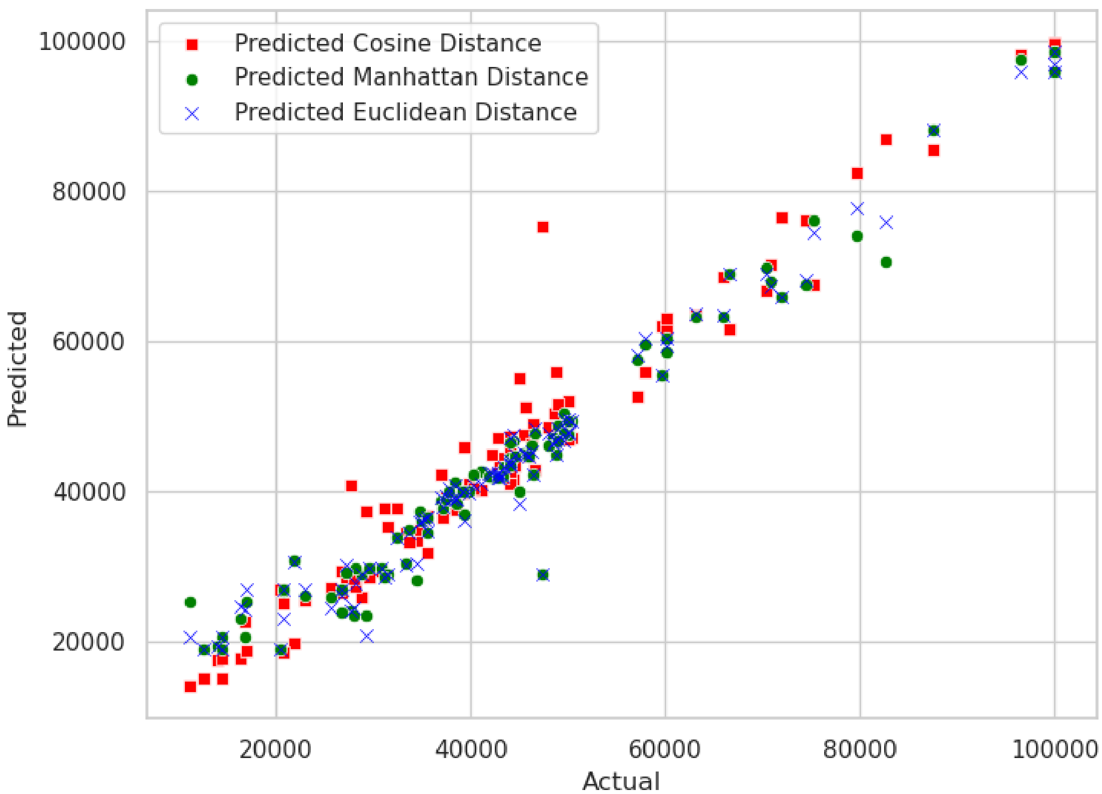
```
Mean Absolute Error: 2605.0420168067226
Mean Squared Error: 15705130.052020807
Root Mean Squared Error: 3962.969852524847
```

[ ]:
```
# Cosine distance metric
knn = KNeighborsRegressor(n_neighbors=7, metric='cosine')
knn.fit(x_train, y_train)
y_pred5 = knn.predict(x_test)
print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred5))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred5))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred5)))
```

```
Mean Absolute Error: 2675.9103641456586
Mean Squared Error: 18408219.28771509
Root Mean Squared Error: 4290.480076601579
```

[ ]:
```
plt.figure(figsize= (8, 6))
```

```
sns.scatterplot(x=y_test, y=y_pred5, color= 'red', label='Predicted Cosine␣
  ↪Distance', marker='s')
sns.scatterplot(x=y_test, y=y_pred4, color= 'green', label='Predicted Manhattan␣
  ↪Distance', marker='o')
sns.scatterplot(x=y_test, y=y_pred3, color= 'blue', label='Predicted Euclidean␣
  ↪Distance', marker='x')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.legend()
plt.show()
```



0.6 Classification:

Binning the numerical data:

[ ]:
```
df.describe()
```

[ ]:

	Marketing expense	Production expense	Multiplex coverage	\
count	506.000000	506.000000	506.000000	
mean	92.270471	77.273557	0.445305	
std	172.030902	13.720706	0.115878	
min	20.126400	55.920000	0.129000	

```
25%           21.640900          65.380000           0.376000
50%           25.130200          74.380000           0.462000
75%           93.541650          91.200000           0.551000
max         1799.524000         110.480000           0.615000

             Budget  Movie_length  Lead_ Actor_Rating  Lead_Actress_rating  \
count    506.000000    506.000000          506.000000           506.000000
mean   34911.144022    142.074901            8.014002             8.185613
std     3903.038232     28.148861            1.054266             1.054290
min    19781.355000     76.400000            3.840000             4.035000
25%    32693.952500    118.525000            7.316250             7.503750
50%    34488.217500    151.000000            8.307500             8.495000
75%    36793.542500    167.575000            8.865000             9.030000
max    48772.900000    173.500000            9.435000             9.540000

       Director_rating  Producer_rating  Critic_rating  Trailer_views  \
count       506.000000       506.000000     506.000000     506.000000
mean          8.019664         8.190514       7.810870  449860.715415
std           1.059899         1.049601       0.659699   68917.763145
min           3.840000         4.030000       6.600000  212912.000000
25%           7.296250         7.507500       7.200000  409128.000000
50%           8.312500         8.465000       7.960000  462460.000000
75%           8.883750         9.030000       8.260000  500247.500000
max           9.425000         9.635000       9.400000  567784.000000

       Time_taken  Twitter_hastags  Avg_age_actors  Num_multiplex  \
count  506.000000       506.000000      506.000000     506.000000
mean   157.391498       260.832095       39.181818     545.043478
std     30.921101       104.779133       12.513697     106.332889
min      0.000000       201.152000        3.000000     333.000000
25%    132.690000       223.796000       28.000000     465.000000
50%    158.980000       254.400000       39.000000     535.500000
75%    181.520000       283.416000       50.000000     614.750000
max    217.520000      2022.400000       60.000000     868.000000

          Collection
count     506.000000
mean    45057.707510
std     18364.351764
min     10000.000000
25%     34050.000000
50%     42400.000000
75%     50000.000000
max    100000.000000
```

[ ]:
```
# Copy to be binned (numerical -> categorical)
df_bin = df.copy()
```

[ ]:
```python
df_bin['Budget_Category'] = pd.qcut(df_bin['Budget'],
        q=[0, .25, .5, .75, 1],
        labels=['A', 'B', 'C', 'D'])
```

[ ]:
```python
df_bin[['Budget', 'Budget_Category']].sample(5)
```

[ ]:
```
        Budget Budget_Category
103  34091.035               B
260  40012.665               D
466  33063.360               B
463  36179.715               C
166  44045.595               D
```

[ ]:
```python
df_bin['Movie_Length_Category'] = pd.qcut(df_bin['Movie_length'],
        q=[0, .33, .66, 1],
        labels=['Short', 'Medium', 'Long'])
df_bin['Actor_Rating_Category'] = pd.qcut(df_bin['Lead_ Actor_Rating'],
        q=[0, .4, .8, 1],
        labels=['Bad', 'Good', 'Excellent'])
df_bin['Actress_Rating_Category'] = pd.qcut(df_bin['Lead_Actress_rating'],
        q=[0, .4, .8, 1],
        labels=['Bad', 'Good', 'Excellent'])
df_bin['Director_Rating_Category'] = pd.qcut(df_bin['Director_rating'],
        q=[0, .4, .8, 1],
        labels=['Bad', 'Good', 'Excellent'])
df_bin['Producer_Rating_Category'] = pd.qcut(df_bin['Producer_rating'],
        q=[0, .4, .8, 1],
        labels=['Bad', 'Good', 'Excellent'])
df_bin['Critic_Rating_Category'] = pd.qcut(df_bin['Critic_rating'],
        q=[0, .4, .8, 1],
        labels=['Bad', 'Good', 'Excellent'])
df_bin['Collection_Category'] = pd.qcut(df_bin['Collection'],
        q=[0, .5, 1],
        labels=['Failure', 'Success'])
```

[ ]:
```python
df_bin = df_bin.iloc[:, 26-8:]
df_bin.sample(5)
```

[ ]:
```
    Budget_Category Movie_Length_Category Actor_Rating_Category  \
17                B                Medium                   Bad
474               A                  Long                  Good
150               B                  Long             Excellent
310               A                 Short                  Good
230               B                Medium                  Good

    Actress_Rating_Category Director_Rating_Category Producer_Rating_Category  \
17                      Bad                      Bad                      Bad
```

```
474                  Good                 Good                 Good
150             Excellent            Excellent            Excellent
310                  Good                 Good                 Good
230                  Good                 Good                 Good

    Critic_Rating_Category Collection_Category
17                     Bad             Failure
474                   Good             Failure
150                   Good             Success
310                    Bad             Failure
230              Excellent             Success
```

[ ]:
```
df_bin = df_bin['Collection_Category']
```

Bayesian Belief Networks

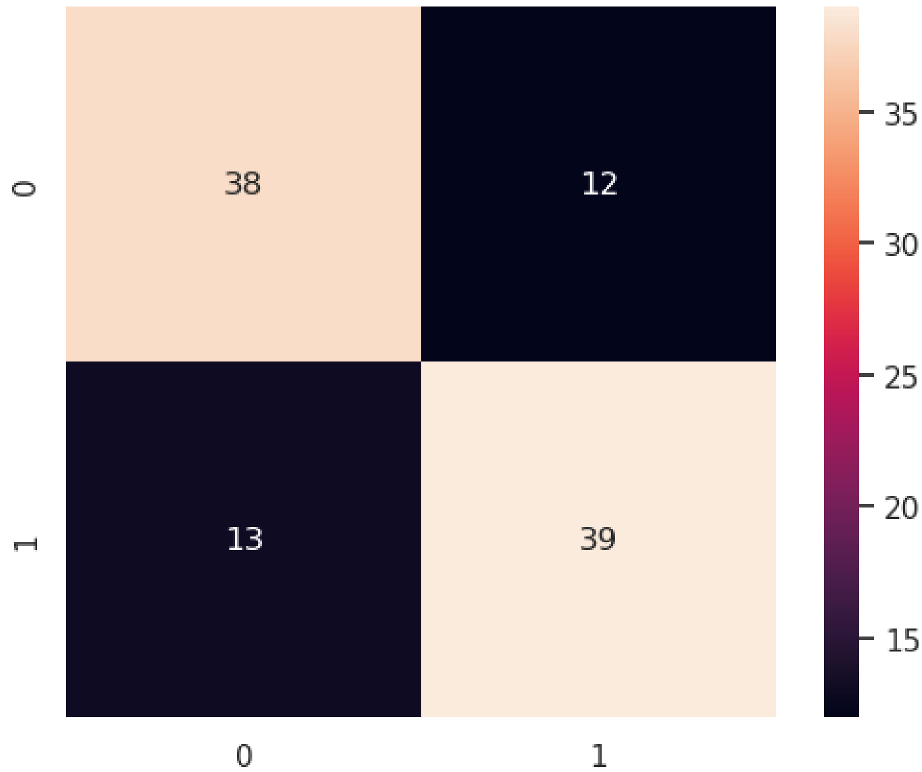
Naive Bayesian

[ ]:
```
X3 = df.drop(columns=['Collection','Genre','3D_available'], axis=1)
Y3 = df_bin

x_train, x_test, y_train, y_test = train_test_split(X3, Y3, test_size=0.2,␣
 ↪random_state=43)

nb = GaussianNB()
nb.fit(x_train, y_train)
nb_pred = nb.predict(x_test)
# Accuracy
accuracy = print('Accuracy Score: ', format(accuracy_score(y_test, nb_pred)))
# Precision
precision = print('Precision Score: ', format(precision_score(y_test, nb_pred,␣
 ↪pos_label='Success')))
# Recall
recall = print('Sensitivity/Recall Score: ', format(recall_score(y_test,␣
 ↪nb_pred, pos_label='Success')))
# F1-score
f1score = print('F1-Measure/F1-Score: ', format(f1_score(y_test, nb_pred,␣
 ↪pos_label='Success')))
```

```
Accuracy Score:  0.7549019607843137
Precision Score:  0.7647058823529411
Sensitivity/Recall Score:  0.75
F1-Measure/F1-Score:  0.7572815533980582
```

[ ]:
```
nb_cf = confusion_matrix(y_test, nb_pred)
nb_cf
sns.heatmap(nb_cf, annot=True, square=True)
```

[ ]: <Axes: >



Decision Tree

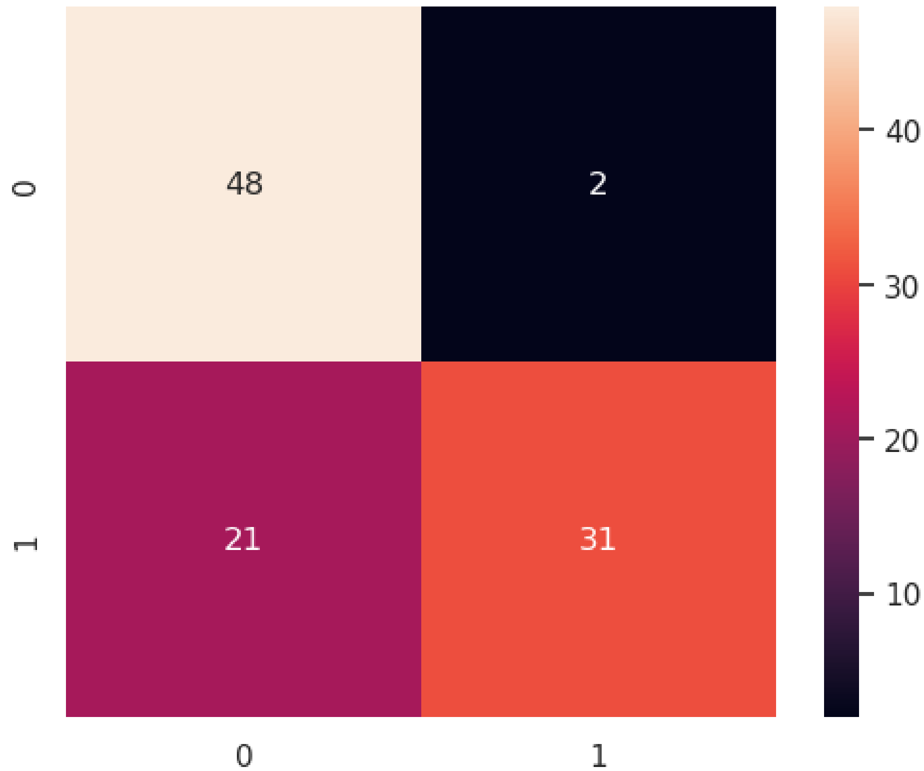
```
[ ]: # Decision Tree
dt = DecisionTreeClassifier(criterion='entropy',max_depth=6)
dt.fit(x_train, y_train)
dt_pred = dt.predict(x_test)
# Accuracy
accuracy = print('Accuracy Score: ', format(accuracy_score(y_test, dt_pred)))
# Precision
precision = print('Precision Score: ', format(precision_score(y_test, dt_pred,␣
    ↪pos_label='Success')))
# Recall
recall = print('Sensitivity/Recall Score: ', format(recall_score(y_test,␣
    ↪dt_pred, pos_label='Success')))
# F1-score
f1score = print('F1-Measure/F1-Score: ', format(f1_score(y_test, dt_pred,␣
    ↪pos_label='Success')))
```

```
Accuracy Score:  0.7745098039215687
Precision Score:  0.9393939393939394
Sensitivity/Recall Score:  0.5961538461538461
```

```
F1-Measure/F1-Score:  0.7294117647058823
```

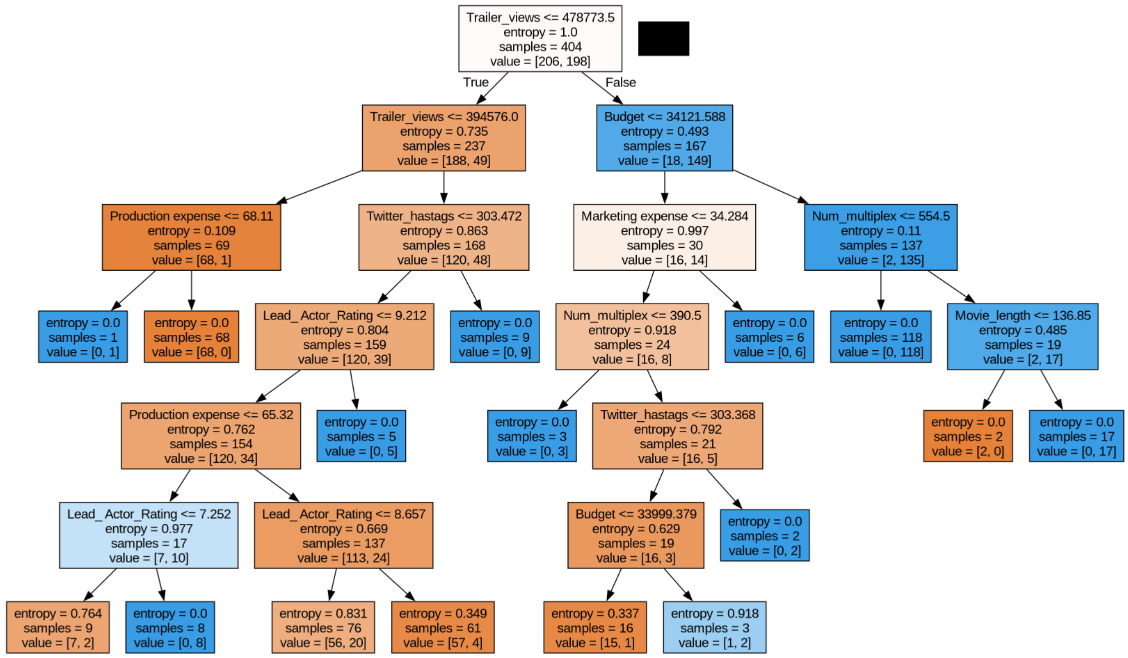
[ ]:
```
dt_cf = confusion_matrix(y_test, dt_pred)
dt_cf

ax= plt.subplot()
sns.heatmap(dt_cf, annot=True, square=True, ax=ax)
```

[ ]:
```
<Axes: >
```



[ ]:
```
from sklearn.tree import export_graphviz
from IPython.display import Image
import pydotplus

dot_data=export_graphviz(dt, out_file=None, feature_names=x_train.columns,␣
 ↪filled=True)
graph=pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

[ ]:



0.7 References:

- https://www.kaggle.com/datasets/balakrishcodes/others
- https://www.kaggle.com/code/auroshisray28/movie-collection-regression-for-beginners