In [1]:

**import** pandas **as** pd

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

In [2]:

df **=** pd**.**read\_csv(r"D:\Imarticus\Project\TMDb\_Movie\_Popularity\_Analysis-main\tmdb-movies.c

In [3]:

df

Out[3]:

**id imdb\_id popularity budget revenue original\_title cast**

**0** 135397 tt0369610 32.985763 150000000 1513528810

Jurassic World

Chris Pratt|Bryce Dallas Howard|Irrfan

Khan|Vi...

http

**1** 76341 tt1392190 28.419936 150000000 378436354

Mad Max: Fury Road

Tom Hardy|Charlize Theron|Hugh Keays-

Byrne|Nic...

http:/

**2** 262500 tt2908446 13.112507 110000000 295238201 Insurgent

Shailene Woodley|Theo James|Kate Winslet|Ansel...

http://www.thediverg

**3** 140607 tt2488496 11.173104 200000000 2068178225

Star Wars: The Force Awakens

Harrison Ford|Mark

Hamill|Carrie Fisher|Adam D...

http://www.sta

Furious 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **4** | 168259 | tt2820852 | 9.335014 | 190000000 | 1506249360 |
| **...** | ... | ... | ... | ... | ... |
| **10861** | 21 | tt0060371 | 0.080598 | 0 | 0 |

Vin Diesel|Paul Walker|Jason Statham|Michelle ...

... ...

The Endless Summer

Michael Hynson|Robert August|Lord 'Tally

Ho' B...

**10862** 20379 tt0060472 0.065543 0 0 Grand Prix

James Garner|Eva Marie Saint|Yves Montand|Tosh...

**id imdb\_id popularity budget revenue original\_title cast**

**10863** 39768 tt0060161 0.065141 0 0

Beregis Avtomobilya

Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...

**10864** 21449 tt0061177 0.064317 0 0 What's Up,

Tiger Lily?

Tatsuya Mihashi|Akiko Wakabayashi|Mie

Hama|Joh...

**10865** 22293 tt0060666 0.035919 19000 0

Manos: The Hands of

Fate

Harold P. Warren|Tom Neyman|John Reynolds|Dian...

# 10866 rows × 21 columns

In [4]:

Out[4]:

In [5]:

df**.**isnull()**.**sum()

id 0

imdb\_id 10

popularity 0

budget 0

revenue 0

original\_title 0

cast 76

homepage 7930

director 44

tagline 2824

keywords 1493

overview 4

runtime 0

genres 23

production\_companies 1030

release\_date 0

vote\_count 0

vote\_average 0

release\_year 0

budget\_adj 0

revenue\_adj 0

dtype: int64

df **=** df[df**.**genres**.**isnull()**==False**]

In [6]:

df **=** df[df**.**production\_companies**.**isnull()**==False**]

In [7]:

df **=** df**.**drop(['id','imdb\_id','original\_title','homepage','tagline','keywords','overview',

In [8]:

df **=** df**.**drop(['cast','production\_companies'],axis **=** 1)

In [42]:

plt**.**boxplot(df**.**popularity)

Out[42]:

In [10]:

con **=** []

**for** i **in** range(0,len(df)): p**=**df**.**genres**.**iloc[i]

p **=** str(p)

t **=** p**.**split('|')

**for** j **in** t:

**if** j **in** con:

**continue else**:

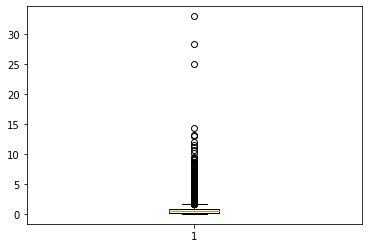
con**.**append(j)

{'whiskers': [<matplotlib.lines.Line2D at 0x172ec1c4c70>,

<matplotlib.lines.Line2D at 0x172ec1da100>], 'caps': [<matplotlib.lines.Line2D at 0x172ec1da370>,

<matplotlib.lines.Line2D at 0x172ec1da7c0>],

'boxes': [<matplotlib.lines.Line2D at 0x172ec1c49a0>], 'medians': [<matplotlib.lines.Line2D at 0x172ec1daac0>], 'fliers': [<matplotlib.lines.Line2D at 0x172ec1daf10>], 'means': []}



In [11]:

df[con] **=** 0

In [12]:

df**.**info()

<class 'pandas.core.frame.DataFrame'> Int64Index: 9827 entries, 0 to 10865 Data columns (total 31 columns):

# Column Non-Null Count Dtype

1. popularity 9827 non-null float64
2. budget 9827 non-null int64
3. revenue 9827 non-null int64
4. director 9807 non-null object
5. runtime 9827 non-null int64
6. genres 9827 non-null object
7. vote\_count 9827 non-null int64
8. vote\_average 9827 non-null float64
9. release\_year 9827 non-null int64
10. budget\_adj 9827 non-null float64
11. revenue\_adj 9827 non-null float64
12. Action 9827 non-null int64
13. Adventure 9827 non-null int64
14. Science Fiction 9827 non-null int64
15. Thriller 9827 non-null int64
16. Fantasy 9827 non-null int64

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 16 | Crime | 9827 | non-null | int64 |
| 17 | Western | 9827 | non-null | int64 |
| 18 | Drama | 9827 | non-null | int64 |
| 19 | Family | 9827 | non-null | int64 |
| 20 | Animation | 9827 | non-null | int64 |
| 21 | Comedy | 9827 | non-null | int64 |
| 22 | Mystery | 9827 | non-null | int64 |
| 23 | Romance | 9827 | non-null | int64 |
| 24 | War | 9827 | non-null | int64 |
| 25 | History | 9827 | non-null | int64 |
| 26 | Music | 9827 | non-null | int64 |
| 27 | Horror | 9827 | non-null | int64 |
| 28 | Documentary | 9827 | non-null | int64 |
| 29 | TV Movie | 9827 | non-null | int64 |
| 30 | Foreign | 9827 | non-null | int64 |

dtypes: float64(4), int64(25), object(2) memory usage: 2.4+ MB

In [13]:

**for** i **in** range(0,len(df)): p**=**df**.**genres**.**iloc[i]

p **=** str(p)

t **=** p**.**split('|')

**for** k **in** t:

**if** k **!=** '':

df[k]**.**iloc[i] **=** 1

C:\Users\kevpa\anaconda3\lib\site-packages\pandas\core\indexing.py:1732: SettingWithCopyWa rning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_gu ide/indexing.html#returning-a-view-versus-a-copy

self.\_setitem\_single\_block(indexer, value, name)

In [14]:

df

Out[14]:

**popularity budget revenue director runtime genres vote\_count vote\_averag**

**0** 32.985763 150000000 1513528810 Colin

Trevorrow

124 Action|Adventure|Science

Fiction|Thriller

|  |  |
| --- | --- |
| 5562 | 6. |
| 6185 | 7. |
| 2480 | 6. |

**1** 28.419936 150000000 378436354 George Miller 120 Action|Adventure|Science

Fiction|Thriller

**2** 13.112507 110000000 295238201 Robert

Schwentke

119 Adventure|Science Fiction|Thriller

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3** | 11.173104 | 200000000 | 2068178225 | J.J. Abrams | 136 | Action|Adventure|Science  Fiction|Fantasy | 5292 | 7. |
| **4** | 9.335014 | 190000000 | 1506249360 | James Wan | 137 | Action|Crime|Thriller | 2947 | 7. |
| **...** | ... | ... | ... | ... | ... | ... | ... |  |
| **10861** | 0.080598 | 0 | 0 | Bruce Brown | 95 | Documentary | 11 | 7. |
| **10862** | 0.065543 | 0 | 0 | John Frankenheimer | 176 | Action|Adventure|Drama | 20 | 5. |
| **10863** | 0.065141 | 0 | 0 | Eldar Ryazanov | 94 | Mystery|Comedy | 11 | 6. |
| **10864** | 0.064317 | 0 | 0 | Woody Allen | 80 | Action|Comedy | 22 | 5. |

**popularity budget revenue director runtime genres vote\_count vote\_averag**

**10865** 0.035919 19000 0 Harold P.

Warren

74 Horror 15 1.

# 9827 rows × 31 columns

In [16]:

Out[16]:

df['popularity']**.**index

Int64Index([ 0, 1, 2, 3, 4, 5, 6, 7, 8,

9,

...

10856, 10857, 10858, 10859, 10860, 10861, 10862, 10863, 10864,

10865],

dtype='int64', length=9827)

In [17]:

sns**.**scatterplot(df["popularity"]**.**index,df["popularity"])

Out[17]:

In [18]:

Out[18]:

In [19]:

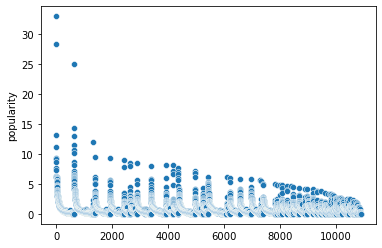
Out[19]:

In [20]:

C:\Users\kevpa\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positiona l argument will be `data`, and passing other arguments without an explicit keyword will re sult in an error or misinterpretation.

warnings.warn(

<AxesSubplot:ylabel='popularity'>



df["vote\_count"]**.**max()

9767

df**.**columns

Index(['popularity', 'budget', 'revenue', 'director', 'runtime', 'genres', 'vote\_count', 'vote\_average', 'release\_year', 'budget\_adj', 'revenue\_adj', 'Action', 'Adventure', 'Science Fiction', 'Thriller', 'Fantasy', 'Crime', 'Western', 'Drama', 'Family', 'Animation', 'Comedy',

'Mystery', 'Romance', 'War', 'History', 'Music', 'Horror', 'Documentary', 'TV Movie', 'Foreign'],

dtype='object')

col **=** df**.**columns

In [21]:

Out[21]:

|  |  |  |
| --- | --- | --- |
| 1 | 1 |  |
| 2 | 1 |  |
| 3 | 0 |  |
| 4 | 1 |  |
|  | .. |  |
| 10861 | 0 |  |
| 10862 | 0 |  |
| 10863 | 0 |  |
| 10864 | 0 |  |
| 10865 | 0 |  |
| Name: | Thriller, Length: 9827, dtype: | int64 |

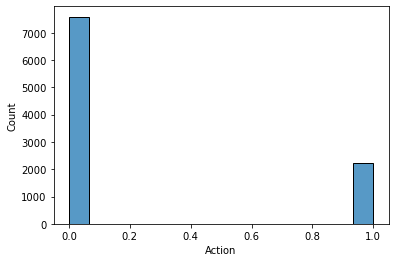
In [22]:

df**.**iloc[0:,14] 0 1

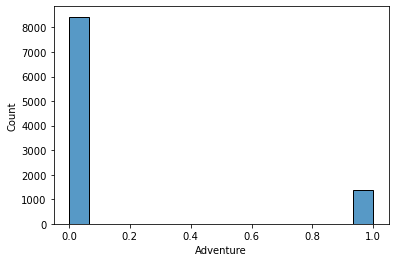
Action

**for** i **in** range(11,29): sns**.**histplot(df**.**iloc[0:,i]) print(col[i])

plt**.**show()

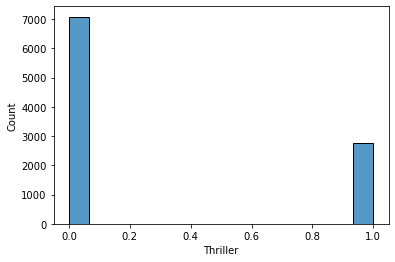


Adventure

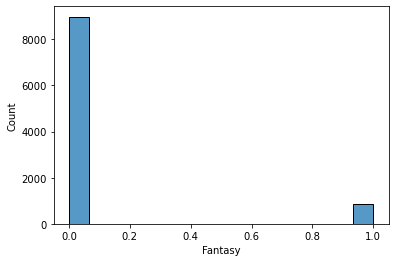


Science Fiction

Thriller

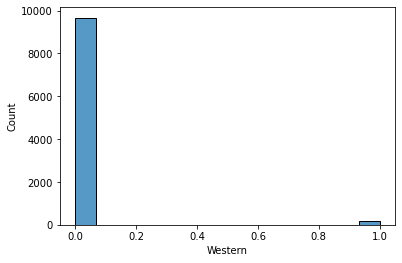


Fantasy

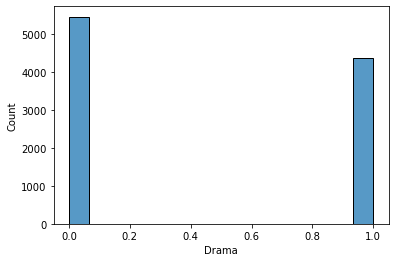


Crime

Western

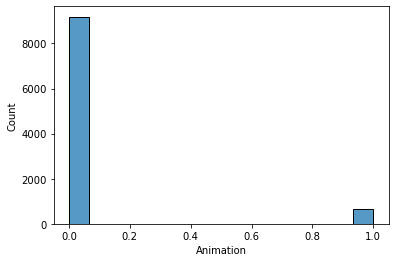


Drama

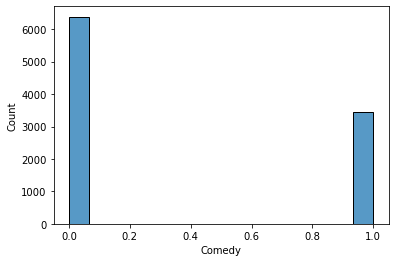


Family

Animation

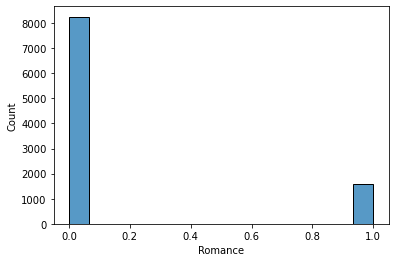


Comedy

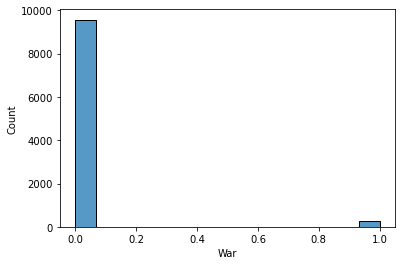


Mystery

Romance

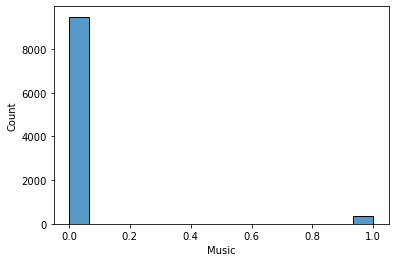


War

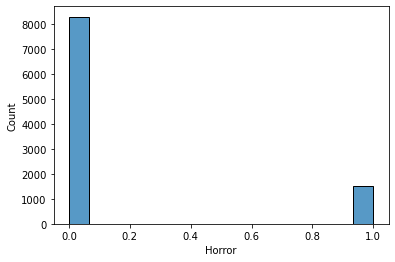


History

Music



Horror



Documentary

In [23]:

len(df**.**columns)

Out[23]: 31

In [24]:

df **=** df**.**drop(['genres'],axis **=** 1)

In [25]:

**from** sklearn.preprocessing **import** LabelEncoder

In [26]:

df['director'] **=** LabelEncoder**.**fit\_transform(df,df['director'])

C:\Users\kevpa\anaconda3\lib\site-packages\sklearn\preprocessing\\_label.py:117: UserWarnin g: Pandas doesn't allow columns to be created via a new attribute name - see https://panda s.pydata.org/pandas-docs/stable/indexing.html#attribute-access

self.classes\_, y = \_unique(y, return\_inverse=True)

In [27]:

df **=** df**.**drop('release\_year',axis**=**1)

In [28]:

**from** sklearn.model\_selection **import** train\_test\_split, cross\_val\_score

X **=** df**.**drop('popularity',axis**=**1) y **=** df['popularity']

In [29]:

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2, random\_state**=**0)

In [30]:

**from** sklearn.ensemble **import** RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegr

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.svm **import** SVR

**from** xgboost **import** XGBRegressor

**from** sklearn.tree **import** DecisionTreeRegressor

In [31]:

linreg **=** LinearRegression() linreg**.**fit(X\_train,y\_train)

Out[31]:

In [32]:

LinearRegression()

Out[32]:

In [33]:

xgb **=** XGBRegressor(n\_estimators**=**50) xgb**.**fit(X\_train,y\_train)

linreg**.**score(X\_test,y\_test) 0.7348247436783566

Out[33]:

In [34]:

Out[34]:

In [35]:

ada **=** AdaBoostRegressor(random\_state**=**0, n\_estimators**=**100)**.**fit(X\_train,y\_train)

XGBRegressor(base\_score=0.5, booster='gbtree', callbacks=None, colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=1, early\_stopping\_rounds=None, enable\_categorical=False, eval\_metric=None, gamma=0, gpu\_id=-1, grow\_policy='depthwise', importance\_type=None, interaction\_constraints='', learning\_rate=0.300000012, max\_bin=256, max\_cat\_to\_onehot=4, max\_delta\_step=0, max\_depth=6, max\_leaves=0, min\_child\_weight=1, missing=nan, monotone\_constraints='()', n\_estimators=50, n\_jobs=0, num\_parallel\_tree=1, predictor='auto', random\_state=0, reg\_alpha=0, reg\_lambda=1, ...)

xgb**.**score(X\_test,y\_test)

0.5836697781105052

In [36]:

Out[36]:

In [37]:

rf **=** RandomForestRegressor(n\_estimators**=**1000, random\_state**=**0) rf**.**fit(X\_train,y\_train)

0.6363811855675683

ada**.**score(X\_test,y\_test)

Out[37]:

In [38]:

Out[38]:

In [ ]:

In [ ]:

In [ ]:

RandomForestRegressor(n\_estimators=1000, random\_state=0)

rf**.**score(X\_test,y\_test)

0.7598590839467816