About Dataset

A data set of 1,000 most popular movies on IMDB in the last 10 years. The data points included are: Title, Genre, Description, Director, Actors, Year, Runtime, Rating, Votes, Revenue, Metascrore Feel free to tinker with it and derive interesting insights

Download

https://www.kaggle.com/datasets/PromptCloudHQ/imdb-dataa (https://www.kaggle.com/datasets/PromptCloudHQ/imdb-dataa)

In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data=pd.read_csv("IMDB-Movie-Data.csv")#Read csv file
data.head() #Display Top 5 rows of the dataset
```

Out[2]:

	Rank	Title	Genre	Description	Director	Actors	Yea
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	201
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	201
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	201
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	201
4	5	Suicide Squad	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	201

In [3]:

```
data.tail() #Check last 5 rows of the dataset
```

Out[3]:

	Rank	Title	Genre	Description	Director	Actors	Year	Run (Minu
995	996	Secret in Their Eyes	Crime,Drama,Mystery	A tight-knit team of rising investigators, alo...	Billy Ray	Chiwetel Ejiofor, Nicole Kidman, Julia Roberts...	2015	
996	997	Hostel: Part II	Horror	Three American college students studying abroa...	Eli Roth	Lauren German, Heather Matarazzo, Bijou Philli...	2007	
997	998	Step Up 2: The Streets	Drama,Music,Romance	Romantic sparks occur between two dance studen...	Jon M. Chu	Robert Hoffman, Briana Evigan, Cassie Ventura,...	2008	
998	999	Search Party	Adventure,Comedy	A pair of friends embark on a mission to reuni...	Scot Armstrong	Adam Pally, T.J. Miller, Thomas Middleditch,Sh...	2014	
999	1000	Nine Lives	Comedy,Family,Fantasy	A stuffy businessman finds himself trapped ins...	Barry Sonnenfeld	Kevin Spacey, Jennifer Garner, Robbie Amell,Ch...	2016	

In [4]:

```
data.shape #Shape of our dataset
```

Out[4]:

```
(1000, 12)
```

In [5]:

```
data.isnull().sum() #Any null value in our dataset
```

Out[5]:

```
Rank                    0
Title                   0
Genre                   0
Description             0
Director                0
Actors                  0
Year                    0
Runtime (Minutes)       0
Rating                  0
Votes                   0
Revenue (Millions)    128
Metascore              64
dtype: int64
```

In [6]:

```
data2=data.fillna(value=1)
data2
```

Out[6]:

	Rank	Title	Genre	Description	Director	Actors	
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2
4	5	Suicide Squad	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	2
...	...	...	...	...	...	...	
995	996	Secret in Their Eyes	Crime,Drama,Mystery	A tight-knit team of rising investigators, alo...	Billy Ray	Chiwetel Ejiofor, Nicole Kidman, Julia Roberts...	2
996	997	Hostel: Part II	Horror	Three American college students studying abroa...	Eli Roth	Lauren German, Heather Matarazzo, Bijou Philli...	2
997	998	Step Up 2: The Streets	Drama,Music,Romance	Romantic sparks occur between two dance studen...	Jon M. Chu	Robert Hoffman, Briana Evigan, Cassie Ventura,...	2
998	999	Search Party	Adventure,Comedy	A pair of friends embark on a mission to reuni...	Scot Armstrong	Adam Pally, T.J. Miller, Thomas Middleditch,Sh...	2
999	1000	Nine Lives	Comedy,Family,Fantasy	A stuffy businessman finds himself trapped ins...	Barry Sonnenfeld	Kevin Spacey, Jennifer Garner, Robbie Amell,Ch...	2

1000 rows × 12 columns

In [7]:

```
data3=data.fillna(method="bfill",inplace=True)
data3
```

In [8]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Rank                1000 non-null   int64  
 1   Title               1000 non-null   object 
 2   Genre               1000 non-null   object 
 3   Description         1000 non-null   object 
 4   Director            1000 non-null   object 
 5   Actors              1000 non-null   object 
 6   Year                1000 non-null   int64  
 7   Runtime (Minutes)   1000 non-null   int64  
 8   Rating              1000 non-null   float64
 9   Votes               1000 non-null   int64  
 10  Revenue (Millions)  1000 non-null   float64
 11  Metascore           1000 non-null   float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
```

In [9]:

```
data.describe()
```

Out[9]:

	Rank	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metas
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.0(
mean	500.500000	2012.783000	113.172000	6.723200	1.698083e+05	81.295100	58.97
std	288.819436	3.205962	18.810908	0.945429	1.887626e+05	104.038645	17.0!
min	1.000000	2006.000000	66.000000	1.900000	6.100000e+01	0.000000	11.0(
25%	250.750000	2010.000000	100.000000	6.200000	3.630900e+04	12.467500	47.0(
50%	500.500000	2014.000000	111.000000	6.800000	1.107990e+05	47.240000	60.0(
75%	750.250000	2016.000000	123.000000	7.400000	2.399098e+05	110.520000	72.0(
max	1000.000000	2016.000000	191.000000	9.000000	1.791916e+06	936.630000	100.0(

In [10]:

```
data.corr()
```

Out[10]:

	Rank	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
Rank	1.000000	-0.261605	-0.221739	-0.219555	-0.283876	-0.281174	-0.197986
Year	-0.261605	1.000000	-0.164900	-0.211219	-0.411904	-0.116304	-0.061384
Runtime (Minutes)	-0.221739	-0.164900	1.000000	0.392214	0.407062	0.246862	0.185508
Rating	-0.219555	-0.211219	0.392214	1.000000	0.511537	0.175988	0.576603
Votes	-0.283876	-0.411904	0.407062	0.511537	1.000000	0.573607	0.307505
Revenue (Millions)	-0.281174	-0.116304	0.246862	0.175988	0.573607	1.000000	0.129089
Metascore	-0.197986	-0.061384	0.185508	0.576603	0.307505	0.129089	1.000000

In [11]:

```
from sklearn.preprocessing import LabelEncoder
```

In [12]:

```
lb=LabelEncoder()
lb
```

Out[12]:

```
LabelEncoder()
```

In [13]:

```
data["Title"]=lb.fit_transform(data['Title'])
data["Genre"]=lb.fit_transform(data['Genre'])
data["Title"]=lb.fit_transform(data['Title'])
data["Director"]=lb.fit_transform(data['Director'])
data["Actors"]=lb.fit_transform(data['Actors'])
data["Rating"]=lb.fit_transform(data['Rating'])
```

In [14]:

```
data.columns
```

Out[14]:

```
Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors', 'Yea
r',
       'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
       'Metascore'],
      dtype='object')
```

In [15]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Rank                1000 non-null   int64  
 1   Title               1000 non-null   int64  
 2   Genre               1000 non-null   int32  
 3   Description         1000 non-null   object 
 4   Director            1000 non-null   int32  
 5   Actors              1000 non-null   int32  
 6   Year                1000 non-null   int64  
 7   Runtime (Minutes)   1000 non-null   int64  
 8   Rating              1000 non-null   int64  
 9   Votes               1000 non-null   int64  
 10  Revenue (Millions)  1000 non-null   float64
 11  Metascore           1000 non-null   float64
dtypes: float64(2), int32(3), int64(6), object(1)
memory usage: 82.2+ KB
```

In [16]:

```
data.head()
```

Out[16]:

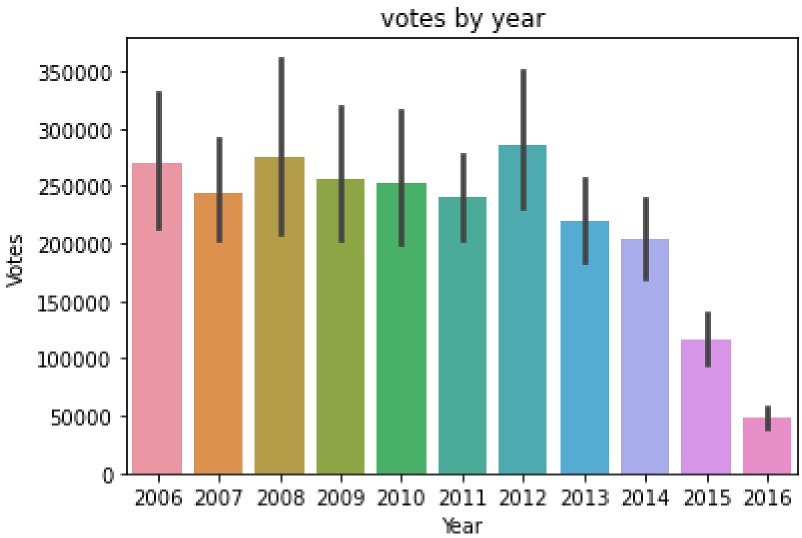
	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions
0	1	287	11	A group of intergalactic criminals are forced ...	265	184	2014	121	47	757074	333.1
1	2	568	85	Following clues to the origin of mankind, a te...	518	736	2012	124	36	485820	126.4
2	3	655	195	Three girls are kidnapped by a man with a diag...	391	418	2016	117	39	157606	138.1
3	4	635	92	In a city of humanoid animals, a hustling thea...	105	658	2016	108	38	60545	270.3
4	5	673	7	A secret government agency recruits some of th...	136	971	2016	123	28	393727	325.0

In [17]:

```
data.dropna(axis=0,inplace=True)
```

In [18]:

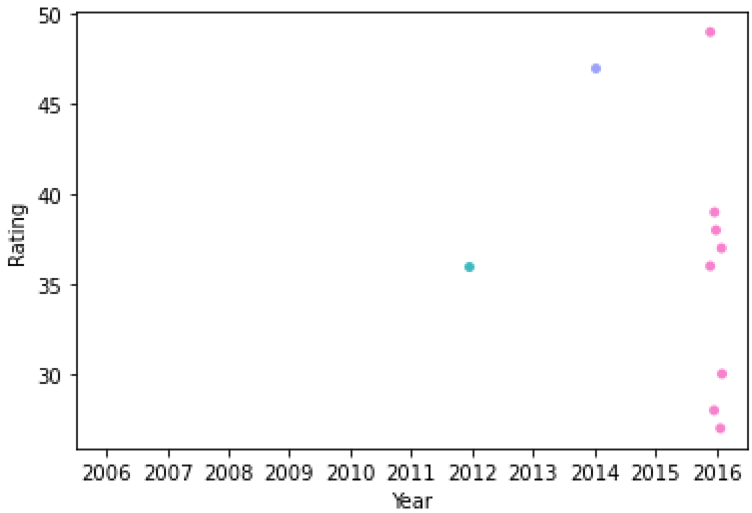
```
sns.barplot(x='Year',y='Votes',data=data)
plt.title("votes by year")
plt.show()
```



A Movie analysis is reviewing the types of variables it offers to compare to the year and votes. You can use a bar chart to visualize the total votes for each year.When the chart is sorted in ascending order, you can see the highest and lowest values. In 2012 has highest votes and in 2016 has lowest votes.

In [19]:

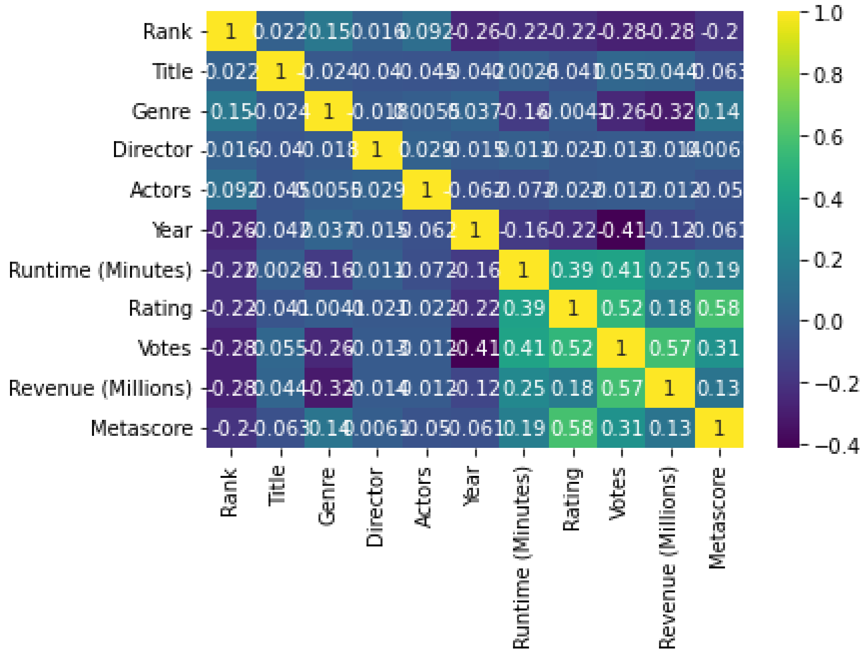
```
sns.stripplot(y = data['Rating'].head(10), x = data['Year'],data=data)
plt.show()
```



Stripplot is used to show all observations along with some representation of the underlying distribution.

In [20]:

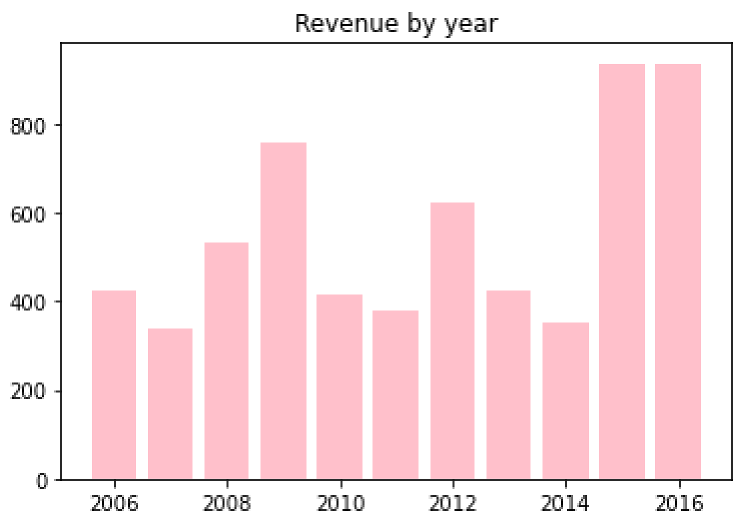
```
sns.heatmap(data.corr(), annot = True, cmap = 'viridis')
plt.show()
```



A heatmap is a graphical representation of numerical data.Create a heat map to visualize areas with the most point features as the hottest.Here we can see that many of variables has 100% accuracy.

In [21]:

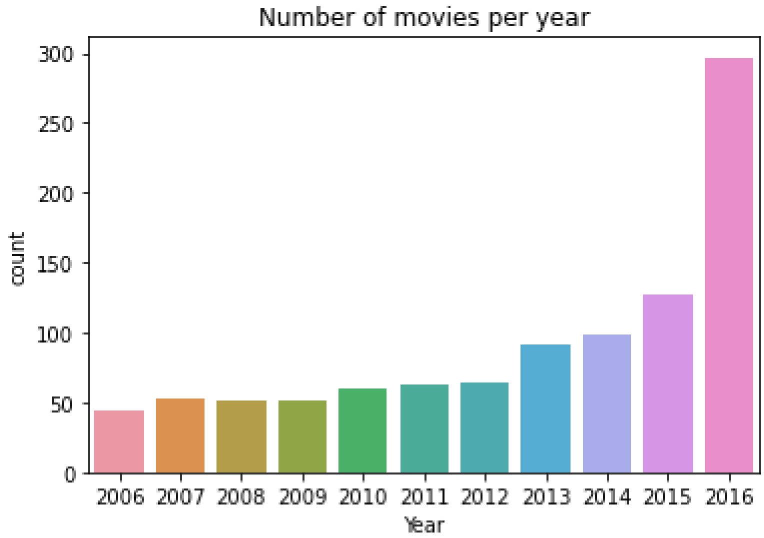
```
data_year=data["Year"]
data_revenue=data["Revenue (Millions)"]
plt.bar(data_year,data_revenue,color="pink")
plt.title("Revenue by year")
plt.show()
```



On above bar chart we can see that on year 2016 and 2015 has highest revenue other hand 2007 has lowest revenue.

In [22]:

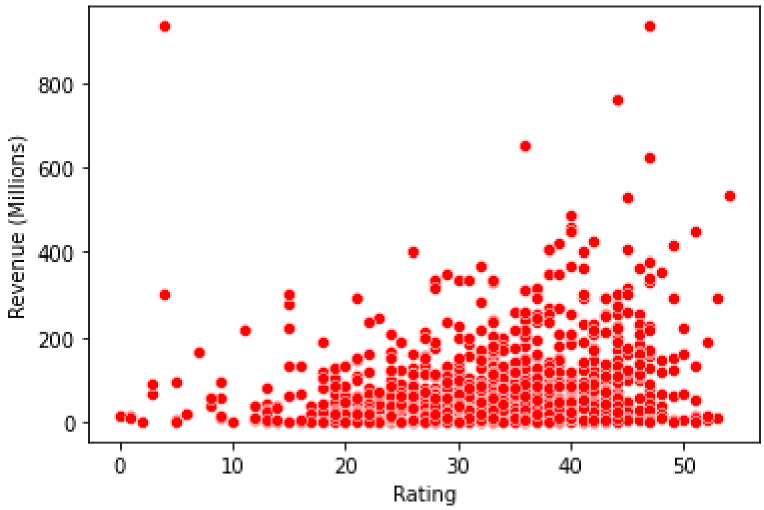
```
sns.countplot(x="Year",data=data)
plt.title("Number of movies per year")
plt.show()
```



The above count plot helps us to understand that the category which has the most number of movies in 2016.

In [23]:

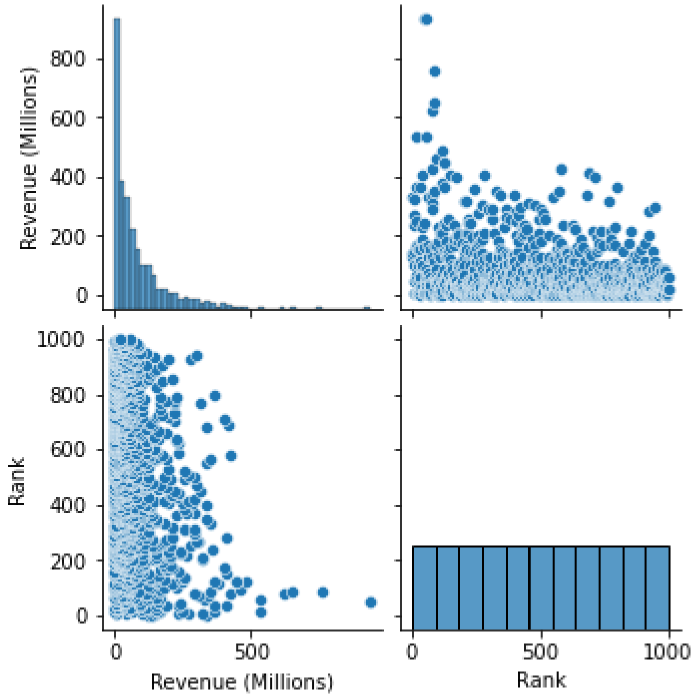
```
sns.scatterplot(x="Rating",y="Revenue (Millions)",data=data,color="red")# Does Rating affec
plt.show()
```



Scatter plots are used to determine the strength of a relationship between two numeric variables.A public rating has noticed an increase in Revenue. We can see that if rating is increase than Revenue also simultaneously rised.

In [24]:

```
sns.pairplot(data[["Revenue (Millions)","Rank",]],diag_kind="auto")
plt.show()
```



The default pairs plot by itself often gives us valuable insights.We see that Rank and Revenue are positively correlated showing that Rank has increased directly Revenue also rised.

In [25]:

```
from sklearn import linear_model
```

In [26]:

```
from sklearn.linear_model import LinearRegression
```

In [27]:

```
lr=LinearRegression()
```

In [28]:

```
x=data[["Rank"]]
y=data[["Revenue (Millions)"]]
```

In [29]:

```
from sklearn.model_selection import train_test_split
```

In [30]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [31]:

```
x_train.head()
```

Out[31]:

	Rank
105	106
68	69
479	480
399	400
434	435

In [32]:

```
lr.fit(x_train,y_train)
```

Out[32]:

```
LinearRegression()
```

In [33]:

```
y_pred=lr.predict(x_test)
```

In [34]:

```
lr.score(x_train,y_train)
```

Out[34]:

```
0.08015876065570848
```

In [35]:

```
print("Coefficient: \n",lr.coef_)
```

```
Coefficient:
 [[-0.09956305]]
```

In [36]:

```
print("Intercept :\n",lr.intercept_)
```

```
Intercept :
 [129.9446301]
```

In [37]:

```
lr.score(y_test, y_pred)
```

Out[37]:

```
-2.6978713157382135
```

In [38]:

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [39]:

```
mean_absolute_error(y_test, y_pred)
```

Out[39]:

```
69.2409777608766
```

In [40]:

```
mean_squared_error(y_test, y_pred)
```

Out[40]:

```
11175.122946057541
```

In [41]:

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[41]:

```
105.71245407262828
```

In [42]:

```
r2_score(y_test, y_pred)
```

Out[42]:

```
0.0764065782166029
```

In [43]:

```
a={"Actual value":x_train,"predicted value":y_pred}
a
```

Out[43]:

```
{'Actual value':      Rank
 105   106
 68     69
 479   480
 399   400
 434   435
 ..    ...
 835   836
 192   193
 629   630
 559   560
 684   685

 [700 rows x 1 columns],
 'predicted value': array([[ 30.97896277],
        [ 44.32041088],
        [100.17527947],
        [ 74.78670284],
```

Multi linear regression

In [44]:

```
x=data[["Rank","Votes","Genre","Director","Rating"]]
y=data[["Revenue (Millions)"]]
```

In [45]:

```
from sklearn.model_selection import train_test_split
```

In [46]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [47]:

```
x_train.head()
```

Out[47]:

	Rank	Votes	Genre	Director	Rating
105	106	115546	145	147	43
68	69	291	149	529	41
479	480	41642	185	351	33
399	400	113686	112	589	27
434	435	223	26	116	30

In [48]:

```
lr.fit(x_train,y_train)
```

Out[48]:

```
LinearRegression()
```

In [49]:

```
y_pred=lr.predict(x_test)
```

In [50]:

```
lr.score(x_train,y_train)
```

Out[50]:

```
0.3392838717599991
```

In [51]:

```
print("Coefficient: \n",lr.coef_)
```

```
Coefficient: 
 [[-5.43594907e-02  2.96269065e-04 -2.18711585e-01  4.71542953e-03
  -1.82147912e+00]]
```

In [52]:

```
print("Intercept :\n",lr.intercept_)
```

```
Intercept :
 [138.35418115]
```

In [53]:

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [54]:

```
mean_absolute_error(y_test, y_pred)
```

Out[54]:

```
49.22025475683878
```

In [55]:

```
mean_squared_error(y_test, y_pred)
```

Out[55]:

```
6386.541757634827
```

In [56]:

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[56]:

```
79.9158417188659
```

In [57]:

```
r2_score(y_test, y_pred)
```

Out[57]:

```
0.4721697484878724
```

In [ ]:

```

```

Decision Tree

In [58]:

```
x=data[["Rank","Votes"]]
y=data[["Revenue (Millions)"]]
```

In [59]:

```
from sklearn.model_selection import train_test_split
```

In [60]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [61]:

```
x_train.head()
```

Out[61]:

	Rank	Votes
105	106	115546
68	69	291
479	480	41642
399	400	113686
434	435	223

In [62]:

```
from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor()
reg.fit(x_train, y_train)
```

Out[62]:

```
DecisionTreeRegressor()
```

In [63]:

```
y_train_pred = reg.predict(x_train)
y_train_pred
```

Out[63]:

```
array([2.6860e+01, 1.0000e-02, 6.2880e+01, 1.1371e+02, 3.0000e-02,
       6.7240e+01, 2.5030e+01, 3.0000e-02, 3.1580e+01, 7.1590e+01,
       8.3810e+01, 2.0007e+02, 7.1900e+00, 7.2200e+00, 1.4440e+01,
       3.3300e+00, 4.0250e+01, 3.6380e+01, 4.0070e+01, 9.3663e+02,
       1.8019e+02, 4.7310e+01, 1.6380e+01, 1.3300e+00, 2.9102e+02,
       6.2328e+02, 1.5880e+02, 2.7000e+00, 7.9000e-01, 3.9000e+00,
       1.3600e+00, 3.4900e+01, 5.3080e+01, 2.5980e+01, 3.2460e+01,
       2.3490e+02, 4.4813e+02, 5.2880e+01, 5.8880e+01, 7.7600e+00,
       9.5330e+01, 6.2880e+01, 7.1900e+01, 5.3217e+02, 8.4240e+01,
       1.2487e+02, 3.8560e+01, 3.4400e+00, 1.7740e+01, 1.5520e+01,
       3.5645e+02, 1.0100e+01, 1.0147e+02, 2.2490e+01, 9.5000e+01,
       3.1450e+01, 2.7360e+01, 5.8720e+01, 5.0000e-02, 5.6440e+01,
       7.5590e+01, 1.6216e+02, 1.7200e+00, 1.0000e-01, 1.2487e+02,
       5.1000e-01, 9.0000e-02, 2.0320e+01, 2.5511e+02, 9.4000e+00,
       7.0000e-02, 3.2280e+01, 3.1060e+01, 2.2000e-01, 1.3090e+01,
       1.2507e+02, 3.7370e+01, 1.2100e+00, 6.5070e+01, 4.2303e+02,
       1.6970e+01, 3.8350e+01, 9.1120e+01, 1.6000e-01, 1.1000e-01,
       7.9000e-01, 3.6800e+00, 5.2000e+00, 4.7700e+01, 1.7659e+02,
```

In [64]:

```
y_pred=reg.predict(x_test)
```

In [65]:

```
reg.score(x_train,y_train)
```

Out[65]:

```
1.0
```

In [66]:

```
print("Coefficient: \n",lr.coef_)
```

```
Coefficient: 
 [[-5.43594907e-02  2.96269065e-04 -2.18711585e-01  4.71542953e-03
  -1.82147912e+00]]
```

In [67]:

```
print("Intercept :\n",lr.intercept_)
```

```
Intercept :
 [138.35418115]
```

In [68]:

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [69]:

```
mean_absolute_error(y_test, y_pred)
```

Out[69]:

```
73.6444
```

In [70]:

```
mean_squared_error(y_test, y_pred)
```

Out[70]:

```
14304.773884666665
```

In [71]:

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[71]:

```
119.60256637993461
```

In [72]:

```
r2_score(y_test, y_pred)
```

Out[72]:

```
-0.18225053305906003
```

In [73]:

```
a={"Actual value":x_train,"predicted value":y_pred}
a
```

Out[73]:

```
{'Actual value':      Rank   Votes
 105   106  115546
 68     69     291
 479   480   41642
 399   400  113686
 434   435     223
 ..    ...     ...
 835   836   38804
 192   193  268282
 629   630   74886
 559   560  115355
 684   685  245144

 [700 rows x 2 columns],
 'predicted value': array([1.4821e+02, 2.3210e+01, 7.1000e+00, 7.7000e+00,
5.7370e+01,
        3.0052e+02, 6.5000e+00, 3.0436e+02, 8.0020e+01, 2.0300e+01,
        1.1280e+01, 3.8320e+01, 3.4900e+01, 6.7240e+01, 1.3880e+02,
```

conclusion

The simple linear regression has best r2 score i.e.76% We've seen a nice range of different movies from different groups in this notebook, with Pulp Fiction and Schindler's List coming out on top overall. Of course there is so much more exploring that could be done - exploring more genres, specific countries etc. but this is where I'll end this notebook.

Thank you so much for reading. Hope you enjoyed!