In [1]:
```
import pandas as pd
import numpy as np
import random as rnd
```

1. Data Preprocessing

In [2]:
```
df=pd.read_csv(r"C:\Users\Shobhit Khaiwal\Downloads\movie.csv")
```

In [3]:
```
df
```

Out[3]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	a
0	Color	James Cameron	723.0	178.0	0.0	
1	Color	Gore Verbinski	302.0	169.0	563.0	
2	Color	Sam Mendes	602.0	148.0	0.0	
3	Color	Christopher Nolan	813.0	164.0	22000.0	
4	NaN	Doug Walker	NaN	NaN	131.0	
...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	
5039	Color	NaN	43.0	43.0	NaN	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	
5041	Color	Daniel Hsia	14.0	100.0	0.0	
5042	Color	Jon Gunn	43.0	90.0	16.0	

5043 rows × 28 columns

In [4]:
```
print(df.columns.values)
```

```
['color' 'director_name' 'num_critic_for_reviews' 'duration'
 'director_facebook_likes' 'actor_3_facebook_likes' 'actor_2_name'
 'actor_1_facebook_likes' 'gross' 'genres' 'actor_1_name' 'movie_title'
 'num_voted_users' 'cast_total_facebook_likes' 'actor_3_name'
 'facenumber_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']
```

In [5]:
```
df.head()
```

Out[5]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor
0	Color	James Cameron	723.0	178.0	0.0	
1	Color	Gore Verbinski	302.0	169.0	563.0	
2	Color	Sam Mendes	602.0	148.0	0.0	
3	Color	Christopher Nolan	813.0	164.0	22000.0	
4	NaN	Doug Walker	NaN	NaN	131.0	

5 rows × 28 columns

In [6]:
```
df.tail(
)
```

Out[6]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	a
5038	Color	Scott Smith	1.0	87.0	2.0	
5039	Color	NaN	43.0	43.0	NaN	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	
5041	Color	Daniel Hsia	14.0	100.0	0.0	
5042	Color	Jon Gunn	43.0	90.0	16.0	

5 rows × 28 columns

In [7]:
```
df.info(
)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   color                      5024 non-null   object 
 1   director_name              4939 non-null   object 
 2   num_critic_for_reviews     4993 non-null   float64
 3   duration                   5028 non-null   float64
 4   director_facebook_likes    4939 non-null   float64
 5   actor_3_facebook_likes     5020 non-null   float64
 6   actor_2_name               5030 non-null   object 
 7   actor_1_facebook_likes     5036 non-null   float64
 8   gross                      4159 non-null   float64
 9   genres                     5043 non-null   object 
 10  actor_1_name               5036 non-null   object 
 11  movie_title                5043 non-null   object 
 12  num_voted_users            5043 non-null   int64  
 13  cast_total_facebook_likes  5043 non-null   int64  
 14  actor_3_name               5020 non-null   object 
 15  facenumber_in_poster       5030 non-null   float64
 16  plot_keywords              4890 non-null   object 
 17  movie_imdb_link            5043 non-null   object 
 18  num_user_for_reviews       5022 non-null   float64
 19  language                   5029 non-null   object 
 20  country                    5038 non-null   object 
 21  content_rating             4740 non-null   object 
 22  budget                     4551 non-null   float64
 23  title_year                 4935 non-null   float64
 24  actor_2_facebook_likes     5030 non-null   float64
 25  imdb_score                 5043 non-null   float64
 26  aspect_ratio               4714 non-null   float64
 27  movie_facebook_likes       5043 non-null   int64  
dtypes: float64(13), int64(3), object(12)
memory usage: 1.1+ MB
```

In [8]:
```
df.shape
```

Out[8]:
```
(5043, 28)
```

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	
num_critic_for_reviews	4993.0	1.401943e+02	1.216017e+02	1.00	50.00	
duration	5028.0	1.072011e+02	2.519744e+01	7.00	93.00	
director_facebook_likes	4939.0	6.865092e+02	2.813329e+03	0.00	7.00	
actor_3_facebook_likes	5020.0	6.450098e+02	1.665042e+03	0.00	133.00	
actor_1_facebook_likes	5036.0	6.560047e+03	1.502076e+04	0.00	614.00	
gross	4159.0	4.846841e+07	6.845299e+07	162.00	5340987.50	2551
num_voted_users	5043.0	8.366816e+04	1.384853e+05	5.00	8593.50	3
cast_total_facebook_likes	5043.0	9.699064e+03	1.816380e+04	0.00	1411.00	
facenumber_in_poster	5030.0	1.371173e+00	2.013576e+00	0.00	0.00	
num_user_for_reviews	5022.0	2.727708e+02	3.779829e+02	1.00	65.00	
budget	4551.0	3.975262e+07	2.061149e+08	218.00	6000000.00	2000
title_year	4935.0	2.002471e+03	1.247460e+01	1916.00	1999.00	
actor_2_facebook_likes	5030.0	1.651754e+03	4.042439e+03	0.00	281.00	
imdb_score	5043.0	6.442138e+00	1.125116e+00	1.60	5.80	
aspect_ratio	4714.0	2.220403e+00	1.385113e+00	1.18	1.85	
movie_facebook_likes	5043.0	7.525965e+03	1.932045e+04	0.00	0.00	

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
color                       19
director_name              104
num_critic_for_reviews      50
duration                    15
director_facebook_likes    104
actor_3_facebook_likes      23
actor_2_name                13
actor_1_facebook_likes       7
gross                      884
genres                       0
actor_1_name                 7
movie_title                  0
num_voted_users              0
cast_total_facebook_likes    0
actor_3_name                23
facenumber_in_poster        13
plot_keywords              153
movie_imdb_link              0
num_user_for_reviews        21
language                    14
country                      5
content_rating             303
budget                     492
title_year                 108
actor_2_facebook_likes      13
imdb_score                   0
aspect_ratio               329
movie_facebook_likes         0
dtype: int64
```

In [11]:
```
df.dropna(axis=0,subset=['director_name' ,'num_critic_for_reviews' ,'duration',
 'director_facebook_likes' ,'actor_3_facebook_likes' ,'actor_2_name'
 ,'actor_1_facebook_likes' ,'actor_1_name','plot_keywords' ,'actor_3_name',
 'facenumber_in_poster','num_user_for_reviews','language',
                'country','actor_2_facebook_likes'],inplace=True)
```

In [12]:
```
df.drop('color',axis=1,inplace=True)
```

In [13]:
```
df.drop('movie_imdb_link',axis=1,inplace=True)
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
director_name                 0
num_critic_for_reviews        0
duration                      0
director_facebook_likes       0
actor_3_facebook_likes        0
actor_2_name                  0
actor_1_facebook_likes        0
gross                       643
genres                        0
actor_1_name                  0
movie_title                   0
num_voted_users               0
cast_total_facebook_likes     0
actor_3_name                  0
facenumber_in_poster          0
plot_keywords                 0
num_user_for_reviews          0
language                      0
country                       0
content_rating              169
budget                      352
title_year                    0
actor_2_facebook_likes        0
imdb_score                    0
aspect_ratio                209
movie_facebook_likes          0
dtype: int64
```

In [15]:
```
df['gross'].fillna(df['gross'].median(),inplace=True)
```

In [16]:
```
df['content_rating'].fillna('R',inplace=True)
```

In [17]:
```
df['budget'].fillna(df['budget'].median(),inplace=True)
```

In [18]:
```
df['aspect_ratio'].fillna(df['aspect_ratio'].median(),inplace=True)
```

In [19]:
```
df.isna().sum()
```

Out[19]:
```
director_name               0
num_critic_for_reviews      0
duration                    0
director_facebook_likes     0
actor_3_facebook_likes      0
actor_2_name                0
actor_1_facebook_likes      0
gross                       0
genres                      0
actor_1_name                0
movie_title                 0
num_voted_users             0
cast_total_facebook_likes   0
actor_3_name                0
facenumber_in_poster        0
plot_keywords               0
num_user_for_reviews        0
language                    0
country                     0
content_rating              0
budget                      0
title_year                  0
actor_2_facebook_likes      0
imdb_score                  0
aspect_ratio                0
movie_facebook_likes        0
dtype: int64
```

In [20]:
```
df.drop_duplicates(inplace=True)
```

In [21]:
```
df.shape
```

Out[21]:
```
(4693, 26)
```

In [22]:
```
df[['language','country']].groupby(['country'],as_index=False).value_counts()
```

Out[22]:

	country	language	count
0	Afghanistan	Dari	1
1	Argentina	Spanish	4
2	Aruba	English	1
3	Australia	English	51
4	Australia	Dzongkha	1
...	...	...	...
106	USA	Dari	1
107	USA	Bosnian	1
108	USA	Aramaic	1
109	West Germany	German	2
110	West Germany	English	1

111 rows × 3 columns

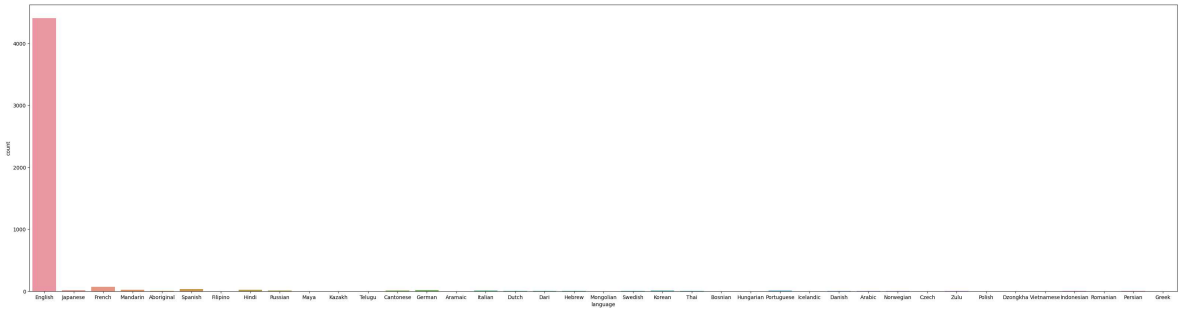
In [23]:
```
df['language'].value_counts()
```

Out[23]:
```
language
English       4405
French          69
Spanish         35
Hindi           25
Mandarin        24
German          18
Japanese        16
Russian         11
Italian         10
Cantonese       10
Korean           8
Portuguese       8
Danish           5
Persian          4
Norwegian        4
Dutch            4
Swedish          4
Hebrew           4
Thai             3
Arabic           3
Dari             2
Indonesian       2
Zulu             2
Aboriginal       2
Vietnamese       1
Polish           1
Romanian         1
Czech            1
Dzongkha         1
Mongolian        1
Icelandic        1
Hungarian        1
Bosnian          1
Aramaic          1
Telugu           1
Kazakh           1
Maya             1
Filipino         1
Greek            1
Name: count, dtype: int64
```

In [24]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [25]:
```
plt.figure(figsize=(40,10))
sns.countplot(df,x='language')
```

Out[25]: <Axes: xlabel='language', ylabel='count'>



In [26]:
```
df.drop('language',axis=1,inplace=True)
```

In [27]:
```
df['Profit']=df['budget'].sub(df['gross'],axis=0)
```

In [28]:
```
df['Profit'].head(10)
```

Out[28]:
```
0    -523505847.0
1      -9404152.0
2      44925825.0
3    -198130642.0
5     190641321.0
6     -78530303.0
7      59192738.0
8    -208991599.0
9     -51956980.0
10    -80249062.0
Name: Profit, dtype: float64
```

In [29]:
```
df['Profit_Percentage']=(df['Profit']/df['gross'])*100
```

In [30]:
```
df['Profit_Percentage'].head(10)
```

Out[30]:
```
0     -68.836532
1      -3.039439
2      22.454585
3     -44.212697
5     260.942743
6     -23.335284
7      29.477389
8     -45.532772
9     -17.206749
10    -24.299558
Name: Profit_Percentage, dtype: float64
```

In [31]:
```
plt.figure(figsize=(40,10))
sns.countplot(df,x=
              'country')
```

Out[31]:
```
<Axes: xlabel='country', ylabel='count'>
```

In [32]:
```
val=df['country'].value_counts()[:2].index
```

In [33]:
```
print(val
     )
```

```
Index(['USA', 'UK'], dtype='object', name='country')
```

In [34]:
```
df['country']=df.country.where(df.country.isin(val),'other')
```

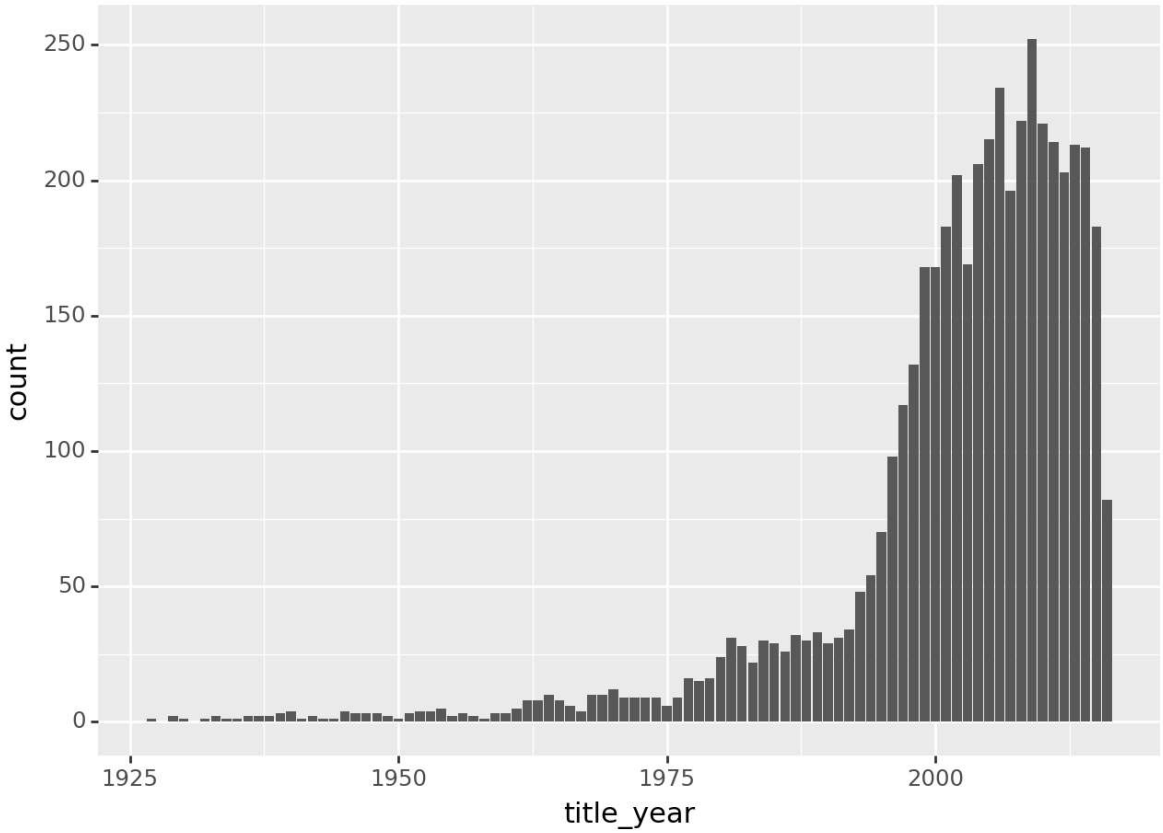
In [35]:
```
df['country'].value_counts()
```

Out[35]:
```
country
USA      3567
other     706
UK        420
Name: count, dtype: int64
```

2.Data Visualization

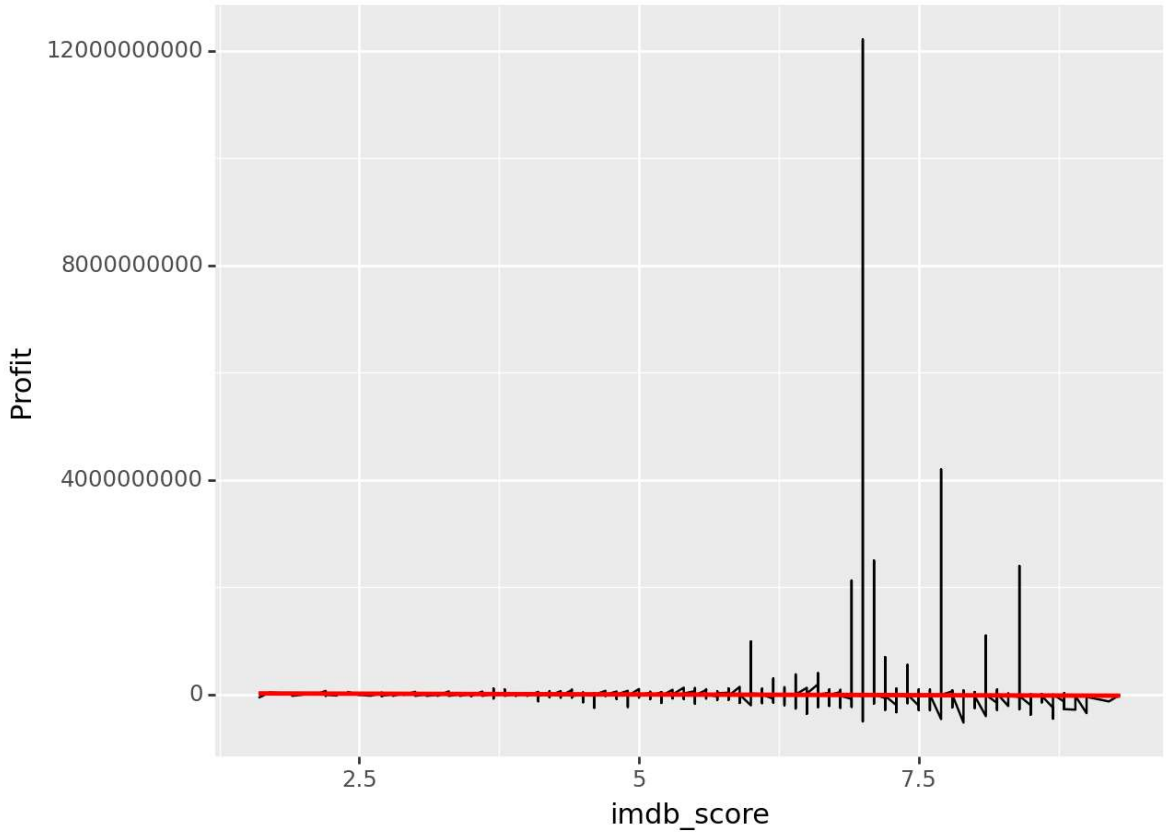
In [36]:
```
from plotnine import*
```

In [37]:
```
(ggplot(df)+aes(x='title_year')+geom_bar(size=20
                                         ))
```



Out[37]:
```
<Figure Size: (640 x 480)>
```

In [38]:
```
ggplot(aes(x='imdb_score',y='Profit'),df)+geom_line()+stat_smooth(colour='red',s
```



Out[38]: <Figure Size: (640 x 480)>

3.Data prepration for model

3.1. Removing the columns with names

In [39]:
```
df.drop('director_name',axis=1,inplace=True)
```

In [40]:
```
df.drop('actor_1_name',axis=1,inplace=True)
```

In [41]:
```
df.drop('actor_2_name',axis=1,inplace=True)
```

In [42]:
```
df.drop('actor_3_name',axis=1,inplace=True)
```

In [43]:
```
df.drop('movie_title',axis=1,inplace=True)
```

In [44]:
```
df.drop('plot_keywords',axis=1,inplace=True)
```

In [45]:
```
df['genres'].value_counts()
```

Out[45]:
```
genres
Drama                                                          209
Comedy                                                         186
Comedy|Drama|Romance                                           182
Comedy|Drama                                                   180
Comedy|Romance                                                 149
                                                              ...
Adventure|Comedy|Family|Sci-Fi                                   1
Action|Adventure|Crime|Drama|Family|Fantasy|Romance|Thriller     1
Adventure|Comedy|History|Romance                                 1
Adventure|Family|Fantasy|Sci-Fi                                  1
Comedy|Crime|Horror                                              1
Name: count, Length: 875, dtype: int64
```

3.2. Removeing linear dependant variable

In [46]:
```
df.drop('genres',axis=1,inplace=True)
```

In [47]:
```
df.drop('country',axis=1,inplace=True)
```

3.3. remove corelated variable

In [48]:
```
df['other_actor_facebook_likes']=df['actor_2_facebook_likes']+df['actor_3_facebc
```

In [49]:
```
df.drop('actor_2_facebook_likes',axis=1,inplace=True)
```

In [50]:
```
df.drop('actor_3_facebook_likes',axis=1,inplace=True)
```

In [51]:
```
df.drop('cast_total_facebook_likes',axis=1,inplace=True)
```

In [52]:
```
df['critic_review_ratio']=df['num_critic_for_reviews']/df['num_user_for_reviews'
```

In [53]:
```
df.drop('num_critic_for_reviews',axis=1,inplace=True)
```

In [54]:
```
df.drop('num_user_for_reviews',axis=1,inplace=True)
```

In [55]:
```
df['imdb_binned_score']=pd.cut(df['imdb_score'],bins=[0,4,6,8,10],right=True,lab
```

In [56]:
```
df.drop('imdb_score',axis=1,inplace=True)
```

In [57]:
```
df.head(10)
```

Out[57]:

	duration	director_facebook_likes	actor_1_facebook_likes	gross	num_voted_u
0	178.0	0.0	1000.0	760505847.0	886
1	169.0	563.0	40000.0	309404152.0	471
2	148.0	0.0	11000.0	200074175.0	275
3	164.0	22000.0	27000.0	448130642.0	1144
5	132.0	475.0	640.0	73058679.0	212
6	156.0	0.0	24000.0	336530303.0	383
7	100.0	15.0	799.0	200807262.0	294
8	141.0	0.0	26000.0	458991599.0	462
9	153.0	282.0	25000.0	301956980.0	321
10	183.0	0.0	15000.0	330249062.0	371

4. Handling the categorical data

In [58]:
```
df=pd.get_dummies(data=df,columns=['content_rating'],prefix=['content_rating'],c
```

In [59]:
```
df.columns
```

Out[59]:
```
Index(['duration', 'director_facebook_likes', 'actor_1_facebook_likes',
       'gross', 'num_voted_users', 'facenumber_in_poster', 'budget',
       'title_year', 'aspect_ratio', 'movie_facebook_likes', 'Profit',
       'Profit_Percentage', 'other_actor_facebook_likes',
       'critic_review_ratio', 'imdb_binned_score', 'content_rating_G',
       'content_rating_GP', 'content_rating_M', 'content_rating_NC-17',
       'content_rating_Not Rated', 'content_rating_PG', 'content_rating_PG-13',
       'content_rating_Passed', 'content_rating_R', 'content_rating_TV-14',
       'content_rating_TV-G', 'content_rating_TV-PG', 'content_rating_Unrated',
       'content_rating_X'],
      dtype='object')
```

5.Splitting data into training and test data

In [60]:
```
x=pd.DataFrame(columns=['duration', 'director_facebook_likes', 'actor_1_facebook
       'gross', 'num_voted_users', 'facenumber_in_poster', 'budget',
       'title_year', 'aspect_ratio', 'movie_facebook_likes',
       'Profit_Percentage', 'other_actor_facebook_likes',
       'critic_review_ratio', 'content_rating_G',
       'content_rating_GP', 'content_rating_M', 'content_rating_NC-17',
       'content_rating_Not Rated', 'content_rating_PG', 'content_rating_PG-13',
       'content_rating_Passed', 'content_rating_R', 'content_rating_TV-14',
       'content_rating_TV-G', 'content_rating_TV-PG', 'content_rating_Unrated',
       'content_rating_X'],data=df)
y=pd.DataFrame(columns=['imdb_binned_score'],data=df)
```

In [61]:
```
from sklearn.model_selection import train_test_split as tts
```

In [62]:
```
X_train,X_test,Y_train,Y_test=tts(x,y,test_size=0.3,random_state=100)
```

6. Feature scaling

In [63]:
```
from sklearn.preprocessing import StandardScaler
```

In [64]:
```
sc=StandardScaler()
```

In [65]:
```
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

7.Logistic Regression

In [66]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [67]:
```
print(X_train,Y_train)
```

```
[[ 0.48347733 -0.21359004 -0.22245118 ...  0.         -0.11242205
  -0.06303257]
 [-0.52625619 -0.23416068  0.63391527 ...  0.         -0.11242205
  -0.06303257]
 [-1.18477804 -0.24531764 -0.4856126  ...  0.          8.89505206
  -0.06303257]
 ...
 [-0.9213693  -0.24287705 -0.39372448 ...  0.         -0.11242205
  -0.06303257]
 [ 0.65908315  1.14930206  1.23337179 ...  0.         -0.11242205
  -0.06303257]
 [-0.9213693  -0.23625261 -0.52329272 ...  0.         -0.11242205
  -0.06303257]]       imdb_binned_score
1410                  2
2709                  3
3828                  2
1476                  2
978                   3
...                 ...
4374                  2
1953                  3
362                   3
80                    3
4122                  2

[3285 rows x 1 columns]
```

In [68]:
```
model.fit(X_train,Y_train)
```

```
C:\Users\Shobhit Khaiwal\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
C:\Users\Shobhit Khaiwal\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

Out[68]: ▾ LogisticRegression

LogisticRegression()

In [69]:
```
Y_predi=model.predict(X_test)
```

8.Accuracy for Logistic Regression

In [70]:
```
from sklearn.metrics import accuracy_score as acs
```

In [71]:
```
Y_predi_accuracy=acs(Y_test,Y_predi)
```

In [72]:
```
print('Accuracy:',Y_predi_accuracy)
```

```
Accuracy: 0.6974431818181818
```

9.Random Forest Classifier

In [73]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [74]:
```
rfc=RandomForestClassifier(n_estimators=250)
```

In [75]:
```
rfc.fit(X_train,np.ravel(Y_train,order='C'))
```

Out[75]: ▾ RandomForestClassifier

RandomForestClassifier(n_estimators=250)

In [76]:
```
rfc_pred=rfc.predict(X_test)
```

10.Accuracy for Random Forest Classifier

In [77]:
```
rfc_predi_accuracy=acs(Y_test,rfc_pred)
```

In [78]:
```
print('Accuracy:',rfc_predi_accuracy)
```

```
Accuracy: 0.7492897727272727
```

In [ ]: