INTRODUCTION

- NAME - SHUVRANSHU SENGUPTA
- SCHOOL OF ELECTRONICS ENGIEERING
- KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT) Deemed to be University

In this project I took an open source dataset based on movies, the languages they are made in, director names etc. I performed various Data Visualization and plotted a lot of bar graphs, pie charts which depicted significant relations between various segments. Further more I performed the train test split and used some regression models like Linear Regression, Decision Tree Regression and a few more to check the accuracy of the model.

IMPORTING LIBRARIES AND UNDERSTANDING THE DATASET

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
import xgboost as xgb
from sklearn.metrics import mean_squared_error, accuracy_score, r2_score, mean_absolute_error
from sklearn import metrics
```

```
# Opening/reading the dataset

df=pd.read_csv('movie_metadata.csv')
pd.set_option('display.max_column', None)
df
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	a
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	
...	...	...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	
5039	Color	NaN	43.0	43.0	NaN	319.0	Valorie Curry	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	

```
# Finding out the total rows and columns in this dataset

df.shape
```

```
(5043, 28)
```

```
# Finding out info about the dataset

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                   5031 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
```

```
# finding out the total number of null values present in the dataframe.

df.isnull().sum()
```

```
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                      12
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
```

```
# Displaying all the column names in the dataframe
```

```
df.columns
```

```
Index(['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'],
      dtype='object')
```

```
df['color'].value_counts()
```

```
Color               4815
 Black and White     209
Name: color, dtype: int64
```

```
df['duration'].value_counts()
```

```
90.0     161
100.0    141
101.0    139
98.0     135
97.0     131
        ...
200.0      1
58.0       1
14.0       1
34.0       1
293.0      1
Name: duration, Length: 191, dtype: int64
```

```
# Using the describe function to see the min and max values as well as mean and standard deviation.

df.describe()
```

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	grc
count	4993.000000	5028.000000	4939.000000	5020.000000	5036.000000	4.159000e+
mean	140.194272	107.201074	686.509212	645.009761	6560.047061	4.846841e+
std	121.601675	25.197441	2813.328607	1665.041728	15020.759120	6.845299e+
min	1.000000	7.000000	0.000000	0.000000	0.000000	1.620000e+
25%	50.000000	93.000000	7.000000	133.000000	614.000000	5.340988e+
50%	110.000000	103.000000	49.000000	371.500000	988.000000	2.551750e+
75%	195.000000	118.000000	194.500000	636.000000	11000.000000	6.230944e+
max	813.000000	511.000000	23000.000000	23000.000000	640000.000000	7.605058e+

```
# Printing the total number of variables present in the dataframe columns

df.nunique()
```

```
color                         2
director_name              2398
num_critic_for_reviews      528
duration                    191
director_facebook_likes     435
actor_3_facebook_likes      906
actor_2_name               3032
actor_1_facebook_likes      878
gross                      4035
genres                      914
actor_1_name               2097
movie_title                4917
num_voted_users            4826
cast_total_facebook_likes  3978
actor_3_name               3521
facenumber_in_poster         19
plot_keywords              4760
movie_imdb_link            4919
num_user_for_reviews        954
language                     47
country                      65
content_rating               18
budget                      439
title_year                   91
actor_2_facebook_likes      917
imdb_score                   78
aspect_ratio                 22
movie_facebook_likes        876
dtype: int64
```

```
# Finding in which languages are the movies made

df['language'].value_counts()
```

```
English       4704
French          73
Spanish         40
Hindi           28
Mandarin        26
German          19
Japanese        18
Russian         11
Cantonese       11
Italian         11
Korean           8
Portuguese       8
Hebrew           5
Arabic           5
Swedish          5
Danish           5
Dutch            4
Norwegian        4
Persian          4
```

```
Polish          4
Thai            3
Chinese         3
None            2
Dari            2
Romanian        2
Zulu            2
Aboriginal      2
Icelandic       2
Indonesian      2
Greek           1
Slovenian       1
Czech           1
Dzongkha        1
Vietnamese      1
Mongolian       1
Swahili         1
Telugu          1
Filipino        1
Kazakh          1
Panjabi         1
Kannada         1
Urdu            1
Aramaic         1
Maya            1
Bosnian         1
Hungarian       1
Tamil           1
Name: language, dtype: int64
```

```
# Printing the movie titles

df['movie_title']
```

```
0                                              Avatar
1            Pirates of the Caribbean: At World's End
2                                             Spectre
3                              The Dark Knight Rises
4      Star Wars: Episode VII - The Force Awakens    ...
                               ...
5038                           Signed Sealed Delivered
5039                          The Following
5040                               A Plague So Pleasant
5041                                   Shanghai Calling
5042                                 My Date with Drew
Name: movie_title, Length: 5043, dtype: object
```

```
# Printing hose movie titles which are based on English from our dataframe.

df.movie_title[df.language == 'English']
```

```
0                                       Avatar
1      Pirates of the Caribbean: At World's End
2                                      Spectre
3                        The Dark Knight Rises
5                                  John Carter
                          ...
```

```
5038                    Signed Sealed Delivered
5039                  The Following
5040                       A Plague So Pleasant
5041                           Shanghai Calling
5042                          My Date with Drew
Name: movie_title, Length: 4704, dtype: object
```

```
# Creating a different dataframe to save all the english movies (subset of the main dataframe)
# We'll use this subset dataframe for our further analysis

Hollywood=df[df.language == 'English']
Hollywood
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	a
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	
...	...	...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	
5039	Color	NaN	43.0	43.0	NaN	319.0	Valorie Curry	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	
5041	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	
5042	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	

4704 rows × 28 columns

```
# Storing movies with IMDB score of 7 or greater than 7

scores=Hollywood.imdb_score[Hollywood.imdb_score>=7]
```

```
# Storing movie titles with IMDB rating of 7 or greater than 7

name=Hollywood.movie_title[Hollywood.imdb_score>=7]
```

```
# Storing movie genre with IMDB rating of 7 or greater than 7

genre=Hollywood.genres[Hollywood.imdb_score>=7]
```

```
# Creating a dataframe where in we concatenate all the above parameters into a single dataframe.

hollywoodmovieratings=pd.concat([name,genre,scores],axis=1)
hollywoodmovieratings
```

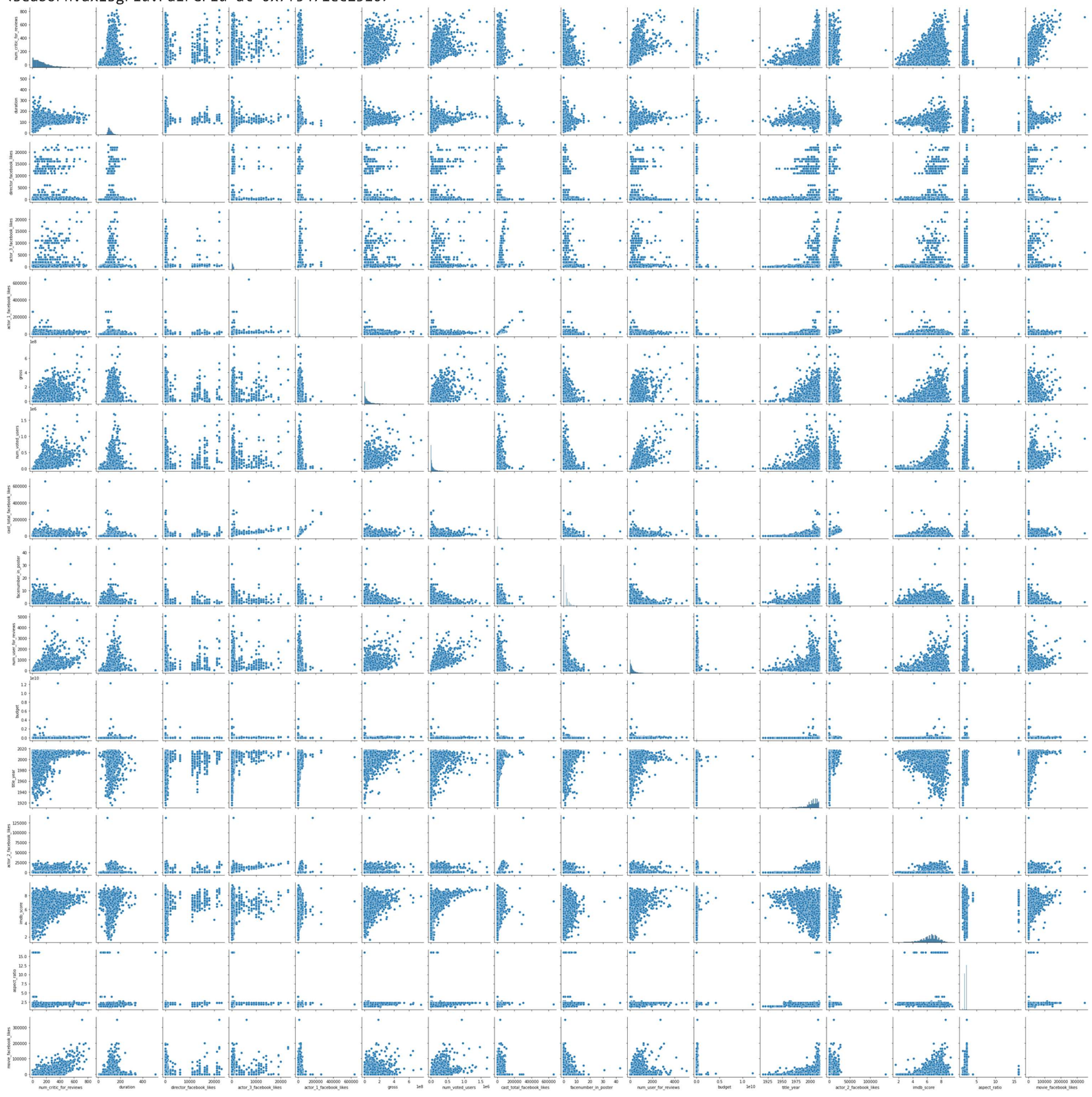
	movie_title	genres	imdb_score
0	Avatar	Action|Adventure|Fantasy|Sci-Fi	7.9
1	Pirates of the Caribbean: At World's End	Action|Adventure|Fantasy	7.1
3	The Dark Knight Rises	Action|Thriller	8.5
7	Tangled	Adventure|Animation|Comedy|Family|Fantasy|Musi...	7.8
8	Avengers: Age of Ultron	Action|Adventure|Sci-Fi	7.5
...	...	...	...
5015	Slacker	Comedy|Drama	7.1
5033	Primer	Drama|Sci-Fi|Thriller	7.0
5036	The Mongol King	Crime|Drama	7.8
5038	Signed Sealed Delivered	Comedy|Drama	7.7
5039	The Following	Crime|Drama|Mystery|Thriller	7.5

1569 rows × 3 columns

VISUALIZING THE DATA

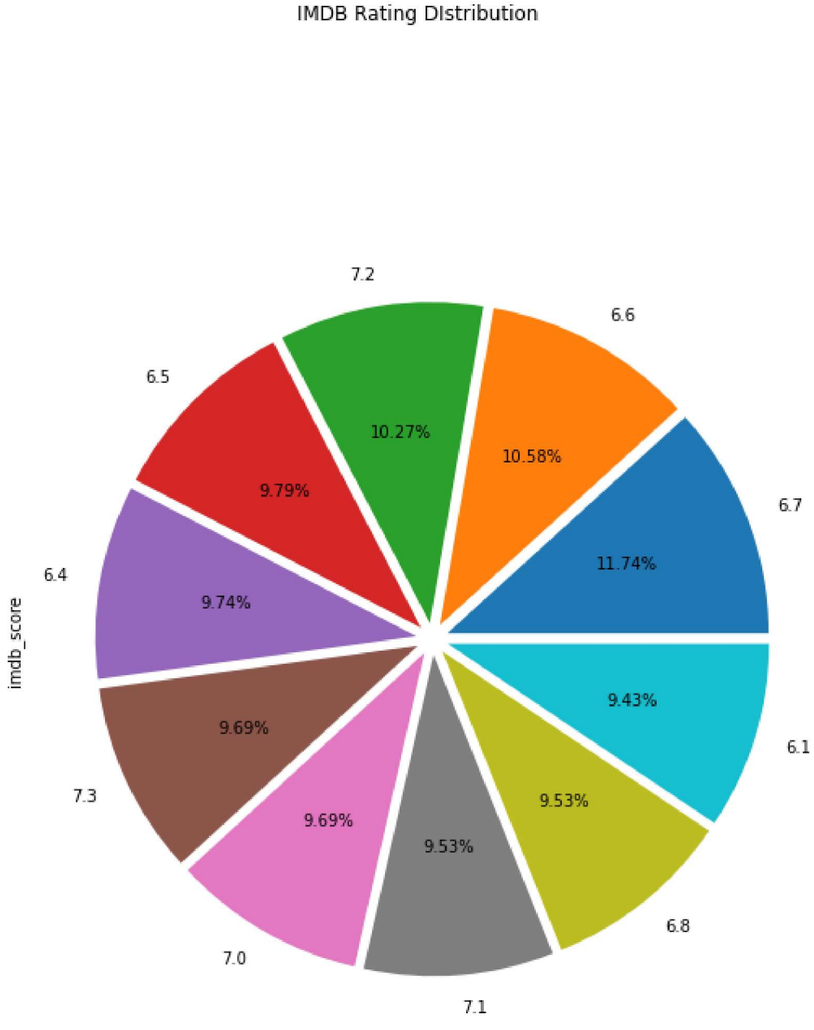
```
sns.pairplot(data=df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f5471ec2510>
```



```
# Pie plot showing top 10 IMDB Ratings
# For example you can see from the plot that around 9.69% of the data has IMDB rating of 7.3

f,ax=plt.subplots(figsize=(10,20))
ax1=plt.subplot(211)
f.suptitle(("IMDB Rating DIstribution"))
explode=(0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05)
df['imdb_score'].value_counts(ascending=False).head(10).plot(kind='pie',autopct="%0.2f%%",explode=explode,ax=ax1)
```

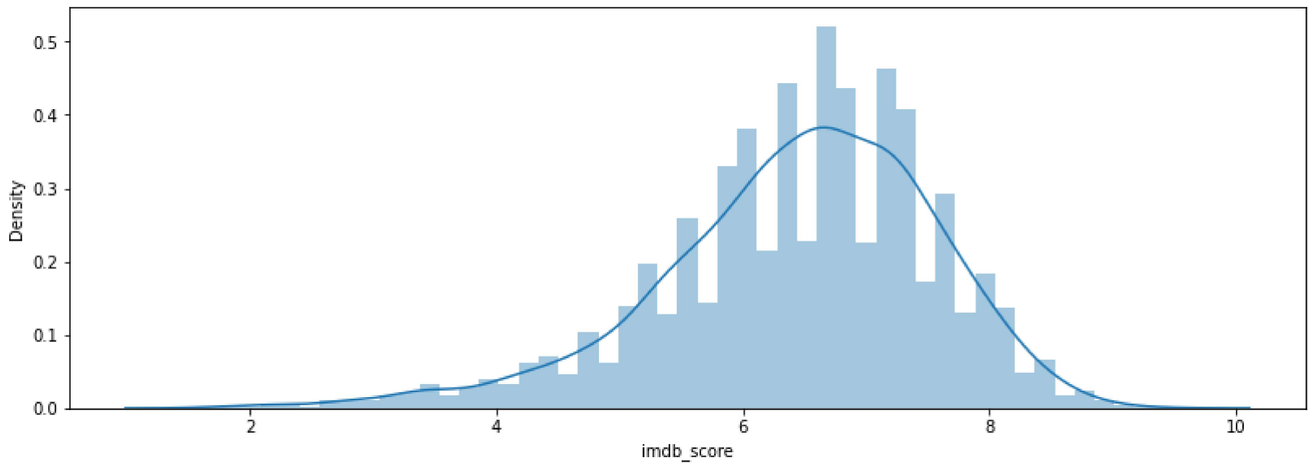
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5462383e90>
```




```
# Now we plot a distribution plot for all prices
# From this distribution plot we can see that the ratings are highest between 6 to 8
```

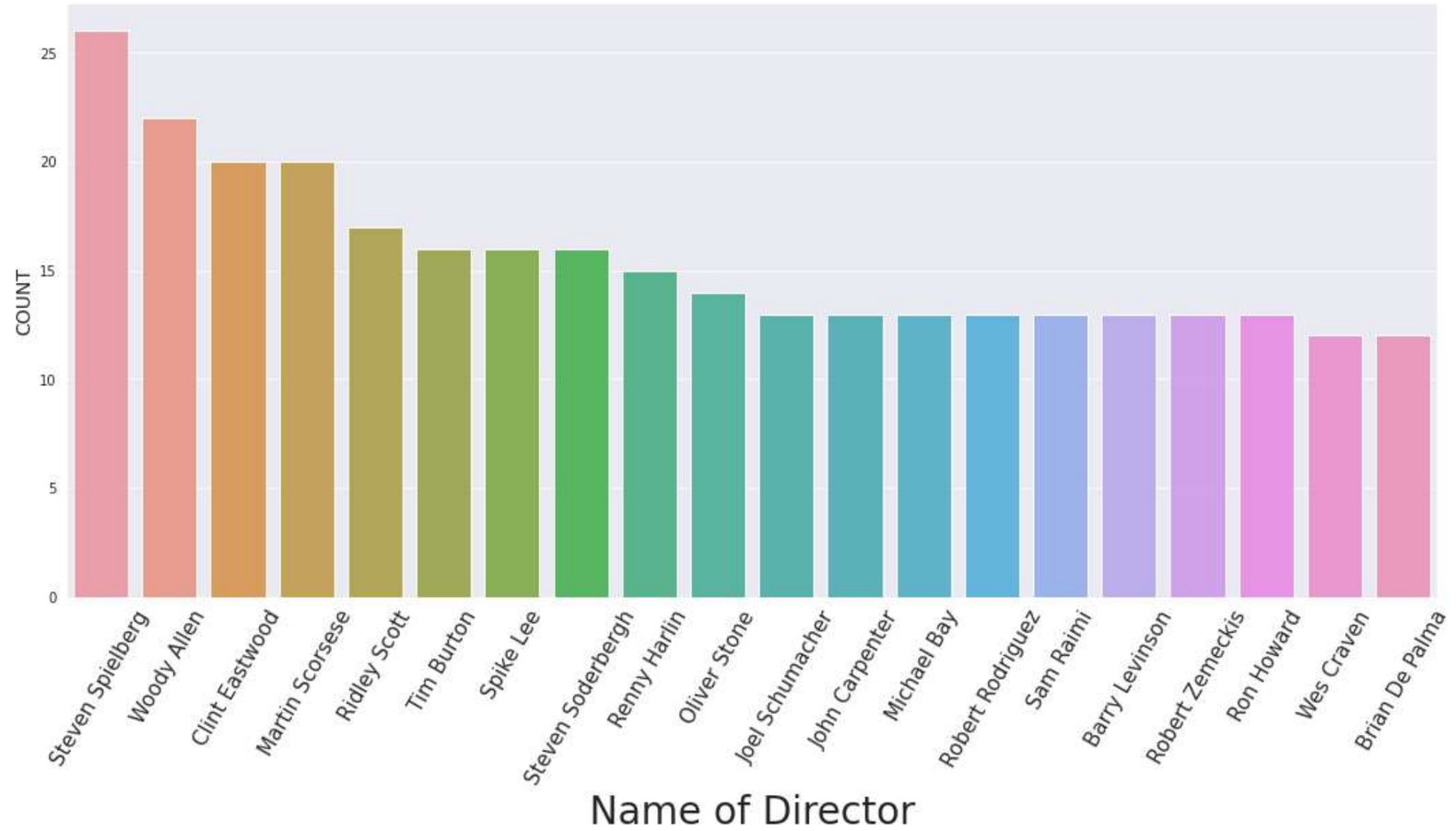
```
f,ax=plt.subplots(figsize=(30,10))
ax3=plt.subplot(224)
sns.distplot(df['imdb_score'],ax=ax3)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f545cad31d0>
```



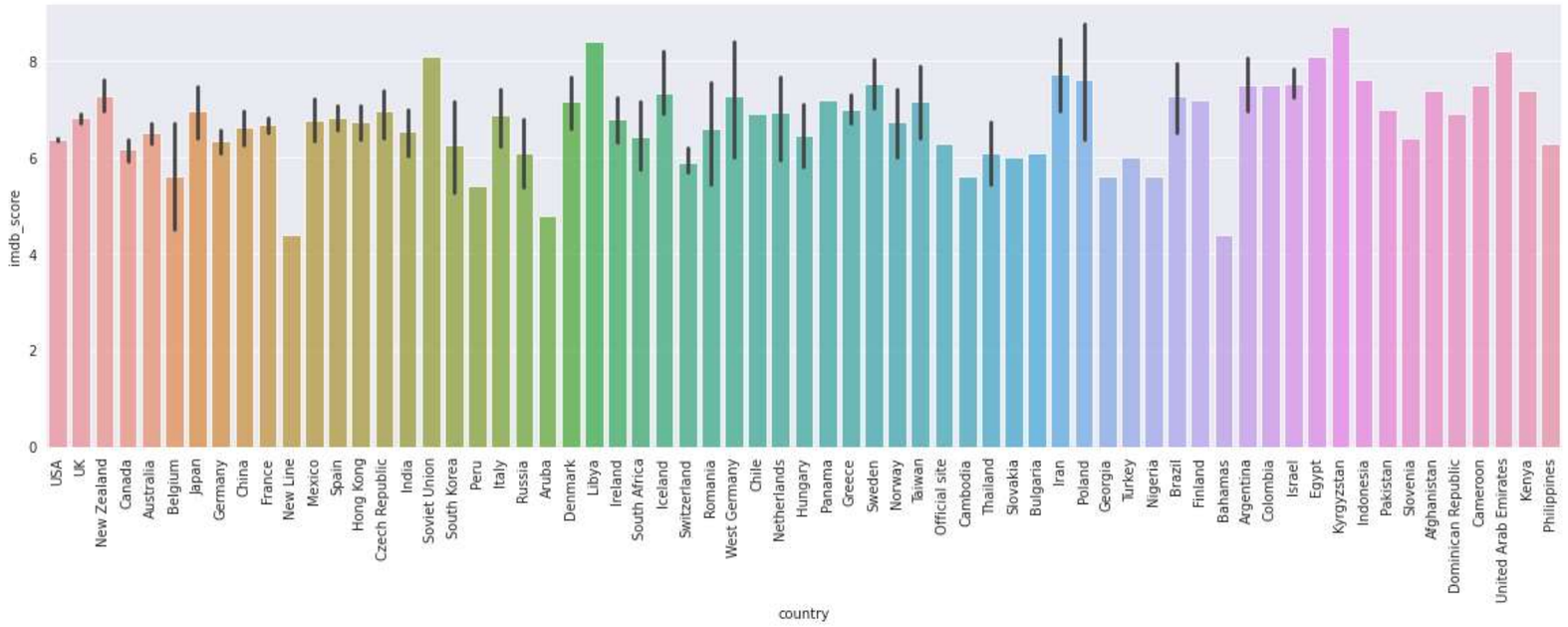

```
# Plotting the count of languages in the dataset.

sns.set_style("darkgrid")
ls=df['language'].value_counts().head(15).sort_values(ascending=False)
plt.figure(figsize=(20,8))
temp =sns.barplot(ls.index, ls.values, alpha=0.8)
plt.ylabel('COUNT', fontsize=14)
plt.xlabel('Type of Languages', fontsize=24)
temp.set_xticklabels(rotation=45,labels=ls.index,fontsize=15)
plt.show()
```


```
# Plotting the Directors names present in the dataset.

sns.set_style("darkgrid")
ls=df['director_name'].value_counts().head(20).sort_values(ascending=False)
plt.figure(figsize=(18,8))
temp =sns.barplot(ls.index, ls.values,alpha=0.9)
plt.ylabel('COUNT', fontsize=14)
plt.xlabel('Name of Director', fontsize=28)
temp.set_xticklabels(rotation=60,labels=ls.index,fontsize=15)
plt.show()
```



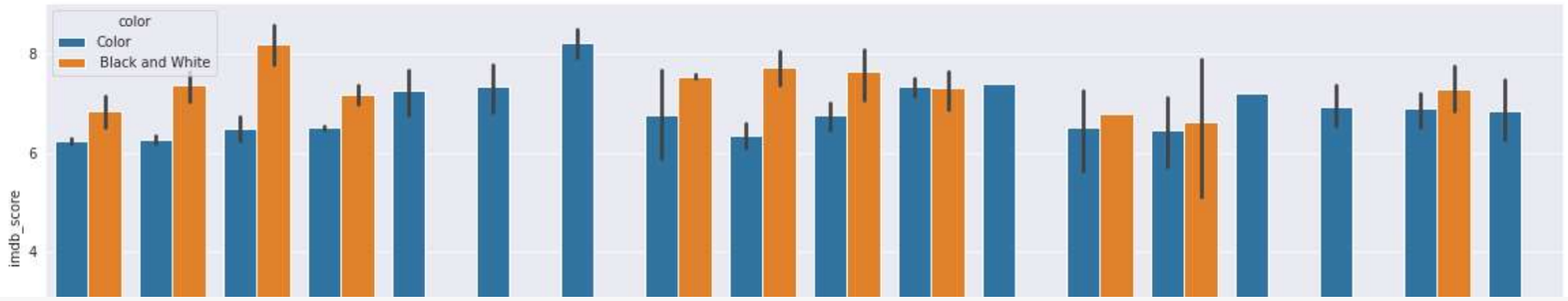
```
# Visualizing the barplot of countries and the imdb scores.

plt.figure(figsize=(20, 6))
sns.barplot(x='country',y='imdb_score',data=df,alpha=0.8);
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]),
 <a list of 65 Text major ticklabel objects>)
```



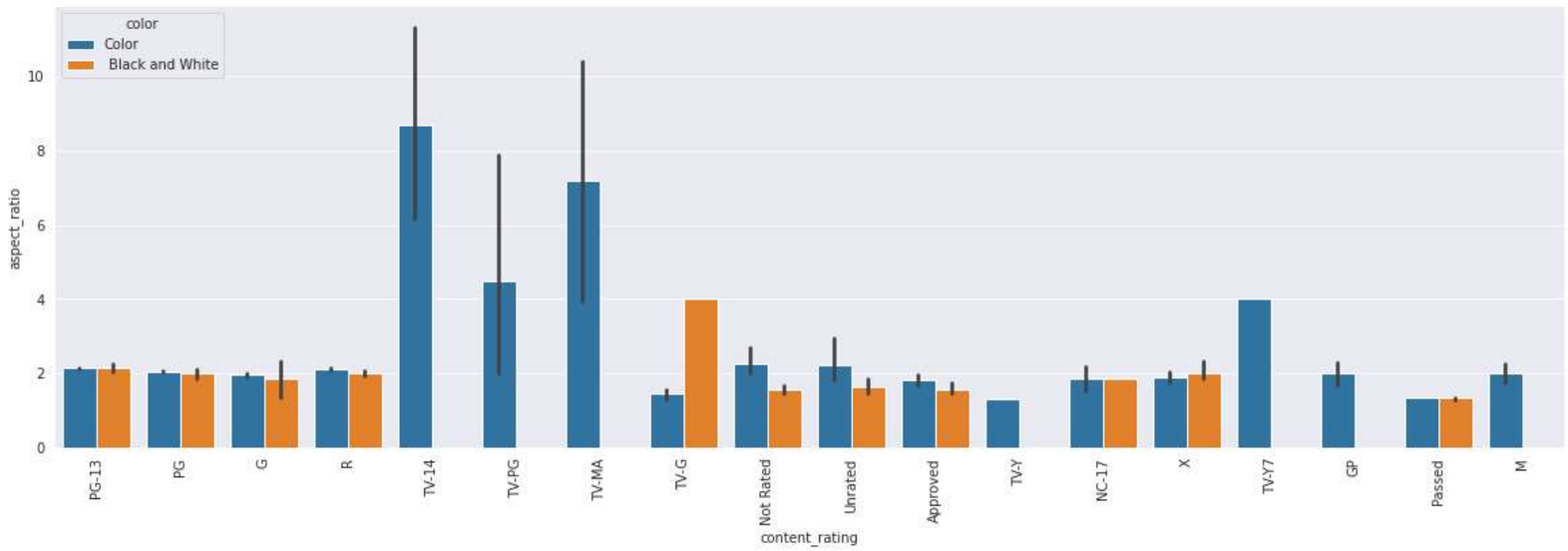

```
# This visualization shows the type of content having higher IMDB Rating and shows us the type of movie color present in data

plt.figure(figsize=(20, 6))
sns.barplot(x='content_rating',y='imdb_score',hue='color',data=df);
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17]), <a list of 18 Text major ticklabel objects>)
```



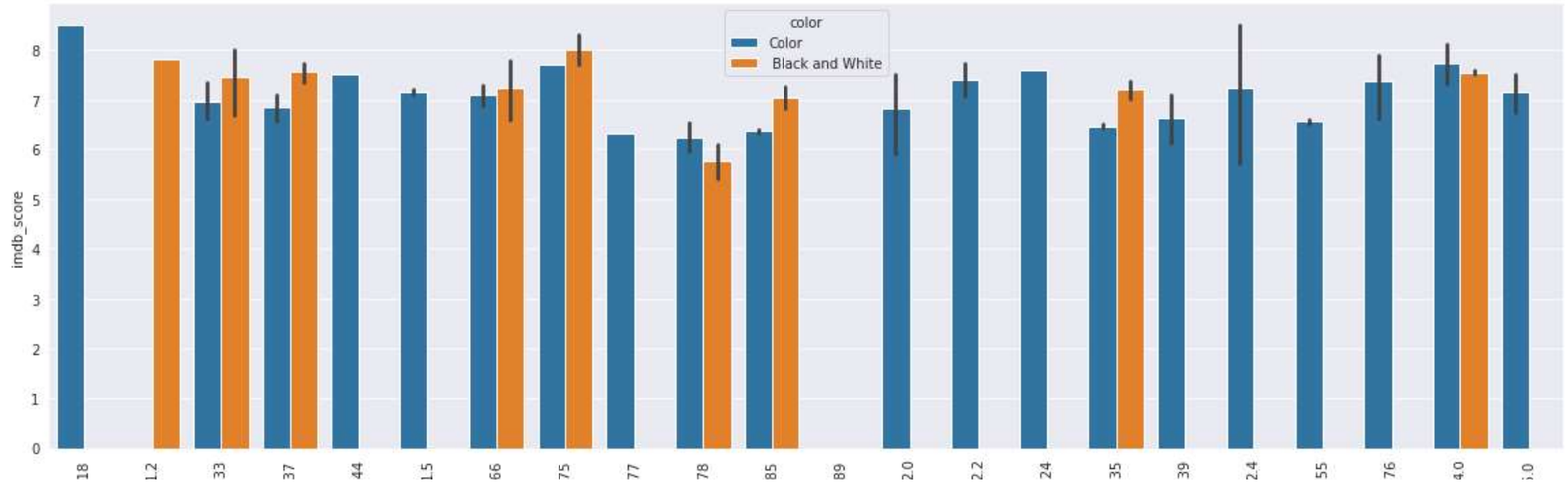

```
# This visualization shows the type of content rating on X-axis having aspect ratio on Y-axis and hue with type of movie color.

plt.figure(figsize=(20, 6))
sns.barplot(x='content_rating',y='aspect_ratio',hue='color',data=df);
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17]), <a list of 18 Text major ticklabel objects>)
```




```
# This visualization shows the aspect ratio and its IMDB rating with hue as the color column.

plt.figure(figsize=(20, 6))
sns.barplot(x='aspect_ratio',y='imdb_score',hue='color',data=df);
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21]), <a list of 22 Text major ticklabel objects>)
```



DATA PREPROCESSING

```
# removing all null values

df=df.dropna()
```

```
# Checking whether all null values are gone or not

df.isnull().sum()
```

```
color                       0
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
movie_imdb_link             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
```

```
df.columns
```

```
Index(['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'],
      dtype='object')
```

```
# Removing some unnecessary columns

df=df.drop(columns=['movie_imdb_link','color','movie_title','facenumber_in_poster', 'plot_keywords',
                    'actor_3_name','movie_imdb_link','aspect_ratio','language'])
```

```
df
```

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_
0	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	

```
df.shape
```

```
(3756, 20)
```

2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	

LABEL ENCODING

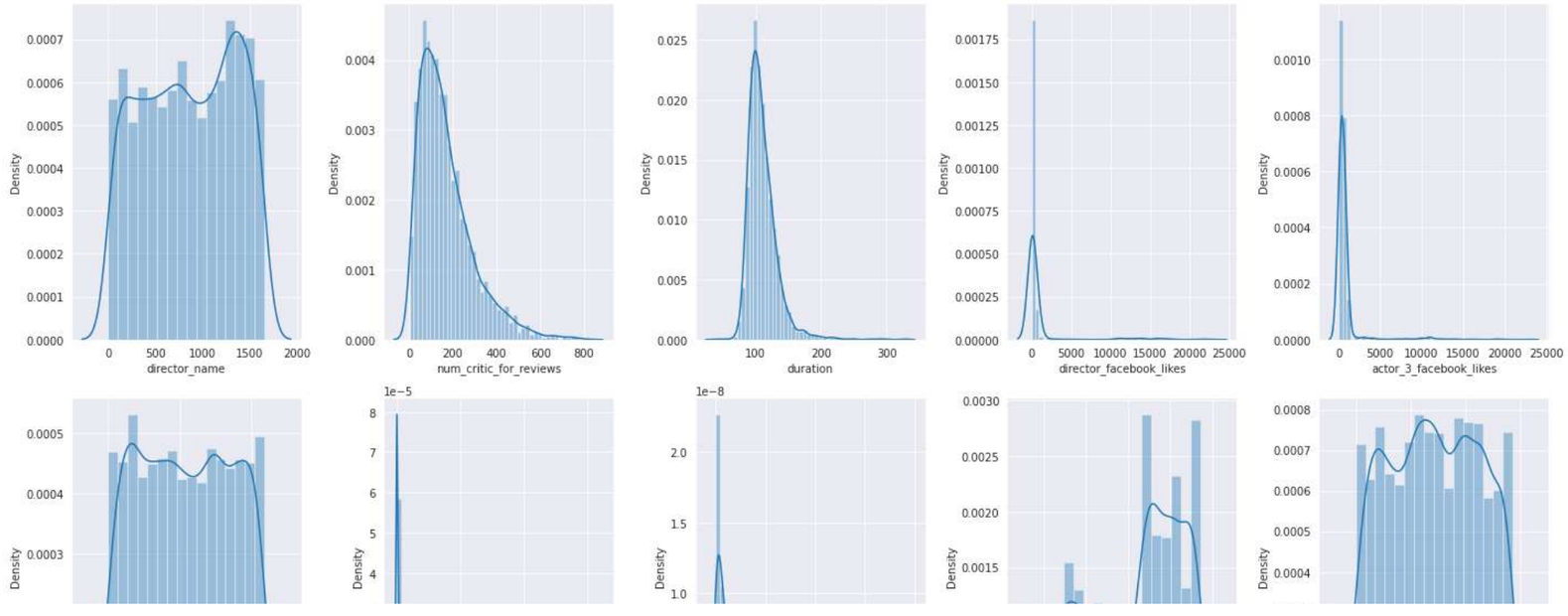
```
# Now we label encoded the categorical columns in the dataset and transformed them to numeric values.

cat_cols=['content_rating','director_name','genres','actor_1_name','actor_2_name','country']
le=LabelEncoder()
for i in cat_cols:
    df[i]=le.fit_transform(df[i])
df.dtypes
```

```
director_name                int64
num_critic_for_reviews     float64
duration                   float64
director_facebook_likes    float64
actor_3_facebook_likes     float64
actor_2_name                 int64
actor_1_facebook_likes     float64
gross                      float64
genres                       int64
actor_1_name                 int64
num_voted_users              int64
cast_total_facebook_likes    int64
num_user_for_reviews       float64
country                      int64
content_rating               int64
budget                     float64
title_year                 float64
actor_2_facebook_likes     float64
imdb_score                 float64
movie_facebook_likes         int64
dtype: object
```

DISTRIBUTION PLOT

```
# The distribution plot shows us the overall distribution of the data.

rows=4
cols=5
fig, ax=plt.subplots(nrows=rows,ncols=cols,figsize=(20,20))
col=df.columns
index=0
for i in range(rows):
```

```
    for j in range(cols):
        sns.distplot(df[col[index]],ax=ax[i][j])
        index=index+1

plt.tight_layout()
```



Log Transformation

```
# Displaying all column names,copypaste this in the next cell.

df.columns
```

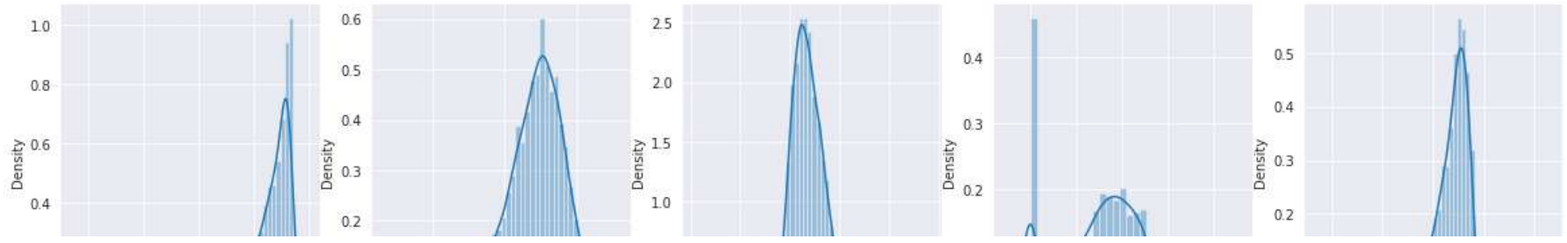
```
Index(['director_name', 'num_critic_for_reviews', 'duration',
       'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',
       'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',
       'num_voted_users', 'cast_total_facebook_likes', 'num_user_for_reviews',
       'country', 'content_rating', 'budget', 'title_year',
       'actor_2_facebook_likes', 'imdb_score', 'movie_facebook_likes'],
      dtype='object')
```

```
# Selecting all features which are skewed and storing them i n the skewed_features

skewed_features=['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',
       'num_voted_users', 'cast_total_facebook_likes', 'num_user_for_reviews',
       'country', 'content_rating', 'budget', 'title_year',
       'actor_2_facebook_likes', 'imdb_score', 'movie_facebook_likes']
```

```
# Applying log transformation on the skewed features

for i in skewed_features:
    df[i]=np.log(df[i]+1)
```

```
# Checking the changes in the distribution of data after applying log transformation.

rows=4
cols=5
```

```
fig, ax=plt.subplots(nrows=rows,ncols=cols,figsize=(20,20))
col=df.columns
index=0
for i in range(rows):
    for j in range(cols):
        sns.distplot(df[col[index]],ax=ax[i][j])
        index=index+1

plt.show()
```



TRAIN TEST SPLIT

director_name num_critic_for_reviews duration director_facebook_likes actor_3_facebook_likes

```
# splitting data into dependent and independent variables

X=df.drop(labels=['imdb_score'],axis=1)
Y=df['imdb_score']
X.head()
```

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_fac
0	6.431331	6.584791	5.187386	0.000000	6.752270	6.910751	
1	6.289716	5.713733	5.135798	6.335054	6.908755	7.373374	
2	7.241366	6.401917	5.003946	0.000000	5.087596	7.493317	
3	5.529429	6.701960	5.105945	9.998843	10.043293	5.945421	
5	4.143135	6.137727	4.890349	6.165418	6.274762	7.516433	

```
# target column

Y.head()
```

```
0    2.186051
1    2.091864
2    2.054124
3    2.251292
5    2.028148
Name: imdb_score, dtype: float64
```

```
# Train_test_split
# Splitting data set into training and testing.

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=40)
print(X_train.shape,X_test.shape,Y_train.shape,Y_test.shape)
```

```
(3004, 19) (752, 19) (3004,) (752,)
```

MACHINE LEARNING

LINEAR REGRESSION

```
lm=LinearRegression()
lm = lm.fit(X_train,Y_train)

#Traindata Predictions
train_pred = lm.predict(X_train)

#testdata predictions
test_pred = lm.predict(X_test)


RMSE_test = np.sqrt(mean_squared_error(Y_test, test_pred))
RMSE_train= np.sqrt(mean_squared_error(Y_train,train_pred))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('-'*50)
print('RSquared value on train:',lm.score(X_train, Y_train))
print('RSquared value on test:',lm.score(X_test, Y_test))
```

```
RMSE TrainingData =  0.12041595568987472
RMSE TestData =  0.11861105307571114
--------------------------------------------------
RSquared value on train: 0.40800950550852266
RSquared value on test: 0.4021001552231892
```

```
# Calculating errors for using error values in mean absolute percentage error

errors = abs(test_pred - Y_test)
```

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / Y_test)

# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 95.5 %.
```

DECISION TREE REGRESSOR

```
DT=DecisionTreeRegressor(max_depth=9)
DT.fit(X_train,Y_train)

#predicting train
train_preds=DT.predict(X_train)
#predicting on test
test_preds=DT.predict(X_test)

RMSE_train=(np.sqrt(metrics.mean_squared_error(Y_train,train_preds)))
```

```
RMSE_test=(np.sqrt(metrics.mean_squared_error(Y_test,test_preds)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('-'*50)
print('RSquared value on train:',DT.score(X_train, Y_train))
print('RSquared value on test:',DT.score(X_test, Y_test))
```

```
RMSE TrainingData =  0.08302711763715252
RMSE TestData =  0.12749462918197255
--------------------------------------------------
RSquared value on train: 0.7185595074222233
RSquared value on test: 0.30918477988711746
```

```
# Calculating errors for using error values in mean absolute percentage error

errors = abs(test_preds - Y_test)
```

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / Y_test)

# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 95.4 %.
```

RANDOM FOREST REGRESSOR

```
RF=RandomForestRegressor().fit(X_train,Y_train)

#predicting train
train_preds1=RF.predict(X_train)
#predicting on test
test_preds1=RF.predict(X_test)

RMSE_train=(np.sqrt(metrics.mean_squared_error(Y_train,train_preds1)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(Y_test,test_preds1)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('-'*50)
print('RSquared value on train:',RF.score(X_train, Y_train))
print('RSquared value on test:',RF.score(X_test, Y_test))
```

```
RMSE TrainingData =  0.041534403213748156
RMSE TestData =  0.10108255162010776
--------------------------------------------------
RSquared value on train: 0.9295692019987555
RSquared value on test: 0.5657593164101153
```

```
# Calculating errors for using error values in mean absolute percentage error

errors = abs(test_preds1 - Y_test)
```

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / Y_test)

# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 96.29 %.
```

K-NEAREST NEIGHBOURS

```
knn=KNeighborsRegressor()
knn.fit(X_train,Y_train)

#predicting train
train_preds2=knn.predict(X_train)
#predicting on test
test_preds2=knn.predict(X_test)

RMSE_train=(np.sqrt(metrics.mean_squared_error(Y_train,train_preds2)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(Y_test,test_preds2)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('-'*50)
print('RSquared value on train:',knn.score(X_train, Y_train))
print('RSquared value on test:',knn.score(X_test, Y_test))
```

```
RMSE TrainingData =  0.10856954037953365
RMSE TestData =  0.13173731630991498
--------------------------------------------------
RSquared value on train: 0.518758958219276
RSquared value on test: 0.2624427420400328
```

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / Y_test)

# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 96.32 %.
```

XG BOOST REGRESSOR

```
xgbr =xgb.XGBRegressor().fit(X_train, Y_train)
#predicting train
train_preds6=xgbr.predict(X_train)
#predicting on test
test_preds6=xgbr.predict(X_test)
```

```
RMSE_train=(np.sqrt(metrics.mean_squared_error(Y_train,train_preds6)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(Y_test,test_preds6)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('-'*50)
print('RSquared value on train:',xgbr.score(X_train, Y_train))
print('RSquared value on test:',xgbr.score(X_test, Y_test))
```

```
[08:13:10] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
RMSE TrainingData =  0.09278927696480212
RMSE TestData =  0.09860319542019205
--------------------------------------------------
RSquared value on train: 0.6484863121948481
RSquared value on test: 0.5868002065711571
```

```
# Calculating errors for using error values in mean absolute percentage error

errors = abs(test_preds6 - Y_test)
```

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / Y_test)

# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 96.32 %.
```