In [1]:

Mission-

Implementation of Machine Learning Regression Models for prediction of Imdb rating with all feautres as trainig dataset

Steps-

```
1) A CLOSER LOOK AT DATA
    1.1) Import libraries & Load training data into pandas dataframe
    1.2) Visualizations
    1.3) Findings

2) Data Preprocessing
    2.1) Feature Scaling
    2.2) Create a reserved(test) dataset

3) Machine Learning Models
    3.1) Linear Regression
    3.2) TensorFlow Keras Regressor
```

1) A CLOSER LOOK AT DATA

1.1) Import libraries & Load training data into pandas dataframe

importing Libraries

In [98]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas import DataFrame,Series
import seaborn as sns
from IPython.core.interactiveshell import InteractiveShell
%matplotlib inline
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode,iplot,plot
init_notebook_mode(connected=True)
import warnings # current version of seaborn generates a bunch of warnings tha
t we'll ignore
warnings.filterwarnings("ignore")
from sklearn.preprocessing import StandardScaler
```

In [107]:
```
data=pd.read_csv("C:/Users/subra/Desktop/Temp/movie_metadata.csv")
```

In [100]:
```
data.head()
```

Out[100]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor
0	Color	James Cameron	723.0	178.0	0.0	855.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000
2	Color	Sam Mendes	602.0	148.0	0.0	161.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	2300
4	NaN	Doug Walker	NaN	NaN	131.0	NaN

5 rows × 28 columns

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
color                      5024 non-null object
director_name              4939 non-null object
num_critic_for_reviews     4993 non-null float64
duration                   5028 non-null float64
director_facebook_likes    4939 non-null float64
actor_3_facebook_likes     5020 non-null float64
actor_2_name               5030 non-null object
actor_1_facebook_likes     5036 non-null float64
gross                      4159 non-null float64
genres                     5043 non-null object
actor_1_name               5036 non-null object
movie_title                5043 non-null object
num_voted_users            5043 non-null int64
cast_total_facebook_likes  5043 non-null int64
actor_3_name               5020 non-null object
facenumber_in_poster       5030 non-null float64
plot_keywords              4890 non-null object
movie_imdb_link            5043 non-null object
num_user_for_reviews       5022 non-null float64
language                   5031 non-null object
country                    5038 non-null object
content_rating             4740 non-null object
budget                     4551 non-null float64
title_year                 4935 non-null float64
actor_2_facebook_likes     5030 non-null float64
imdb_score                 5043 non-null float64
aspect_ratio               4714 non-null float64
movie_facebook_likes       5043 non-null int64
dtypes: float64(13), int64(3), object(12)
memory usage: 1.1+ MB
```

2.Data Preprocessing

Seperating the datas as numerical and catogerical data for further Data Analysis

In [6]:
```
numerical_features = data.select_dtypes(exclude=['object']).columns
categorical_features = data.select_dtypes(include=['object']).columns
```

In [7]:
```
numdata=data[numerical_features]
```

In [8]:
```
numdata
```

Out[8]:

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes
0	723.0	178.0	0.0	855.0
1	302.0	169.0	563.0	1000.0
2	602.0	148.0	0.0	161.0
3	813.0	164.0	22000.0	23000.0
4	NaN	NaN	131.0	NaN
5	462.0	132.0	475.0	530.0
6	392.0	156.0	0.0	4000.0
7	324.0	100.0	15.0	284.0
8	635.0	141.0	0.0	19000.0
9	375.0	153.0	282.0	10000.0
10	673.0	183.0	0.0	2000.0
11	434.0	169.0	0.0	903.0
12	403.0	106.0	395.0	393.0
13	313.0	151.0	563.0	1000.0
14	450.0	150.0	563.0	1000.0
15	733.0	143.0	0.0	748.0
16	258.0	150.0	80.0	201.0
17	703.0	173.0	0.0	19000.0
18	448.0	136.0	252.0	1000.0
19	451.0	106.0	188.0	718.0
20	422.0	164.0	0.0	773.0
21	599.0	153.0	464.0	963.0
22	343.0	156.0	0.0	738.0
23	509.0	186.0	0.0	773.0
24	251.0	113.0	129.0	1000.0
25	446.0	201.0	0.0	84.0
26	315.0	194.0	0.0	794.0
27	516.0	147.0	94.0	11000.0
28	377.0	131.0	532.0	627.0
29	644.0	124.0	365.0	1000.0
...	...	...	...	...
5013	28.0	79.0	3.0	42.0

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes
5014	58.0	80.0	892.0	492.0
5015	61.0	100.0	0.0	0.0
5016	NaN	90.0	0.0	9.0
5017	1.0	90.0	138.0	138.0
5018	5.0	120.0	589.0	4.0
5019	43.0	91.0	158.0	265.0
5020	NaN	143.0	8.0	8.0
5021	51.0	85.0	157.0	10.0
5022	6.0	60.0	0.0	4.0
5023	22.0	88.0	38.0	211.0
5024	42.0	78.0	91.0	86.0
5025	73.0	108.0	0.0	105.0
5026	81.0	110.0	107.0	45.0
5027	64.0	90.0	397.0	0.0
5028	12.0	83.0	18.0	0.0
5029	78.0	111.0	62.0	6.0
5030	NaN	84.0	5.0	12.0
5031	13.0	82.0	120.0	84.0
5032	10.0	98.0	3.0	152.0
5033	143.0	77.0	291.0	8.0
5034	35.0	80.0	0.0	0.0
5035	56.0	81.0	0.0	6.0
5036	NaN	84.0	2.0	2.0
5037	14.0	95.0	0.0	133.0
5038	1.0	87.0	2.0	318.0
5039	43.0	43.0	NaN	319.0
5040	13.0	76.0	0.0	0.0
5041	14.0	100.0	0.0	489.0
5042	43.0	90.0	16.0	16.0

5043 rows × 16 columns

```
In [9]: data.isnull().sum().sort_values(ascending = False)
```

```
Out[9]: gross                       884
        budget                      492
        aspect_ratio                329
        content_rating              303
        plot_keywords               153
        title_year                  108
        director_name               104
        director_facebook_likes     104
        num_critic_for_reviews       50
        actor_3_name                 23
        actor_3_facebook_likes       23
        num_user_for_reviews         21
        color                        19
        duration                     15
        facenumber_in_poster         13
        actor_2_name                 13
        actor_2_facebook_likes       13
        language                     12
        actor_1_name                  7
        actor_1_facebook_likes        7
        country                       5
        movie_facebook_likes          0
        genres                        0
        movie_title                   0
        num_voted_users               0
        movie_imdb_link               0
        imdb_score                    0
        cast_total_facebook_likes     0
        dtype: int64
```

we have training dataset with 284807 rows & 30 features & 1 target variable 'Class' (0 or 1 , zero means normal transaction & 1 means fradulent). due to data privacy 28 of the features are anonymous so they will not be useful during data visualization but we have 2 name features 'Time' (It contains the seconds elapsed between each transaction and the first transaction in the dataset) and 'Amount'(amount of transaction). Lets plot some graphs-

Missing Values

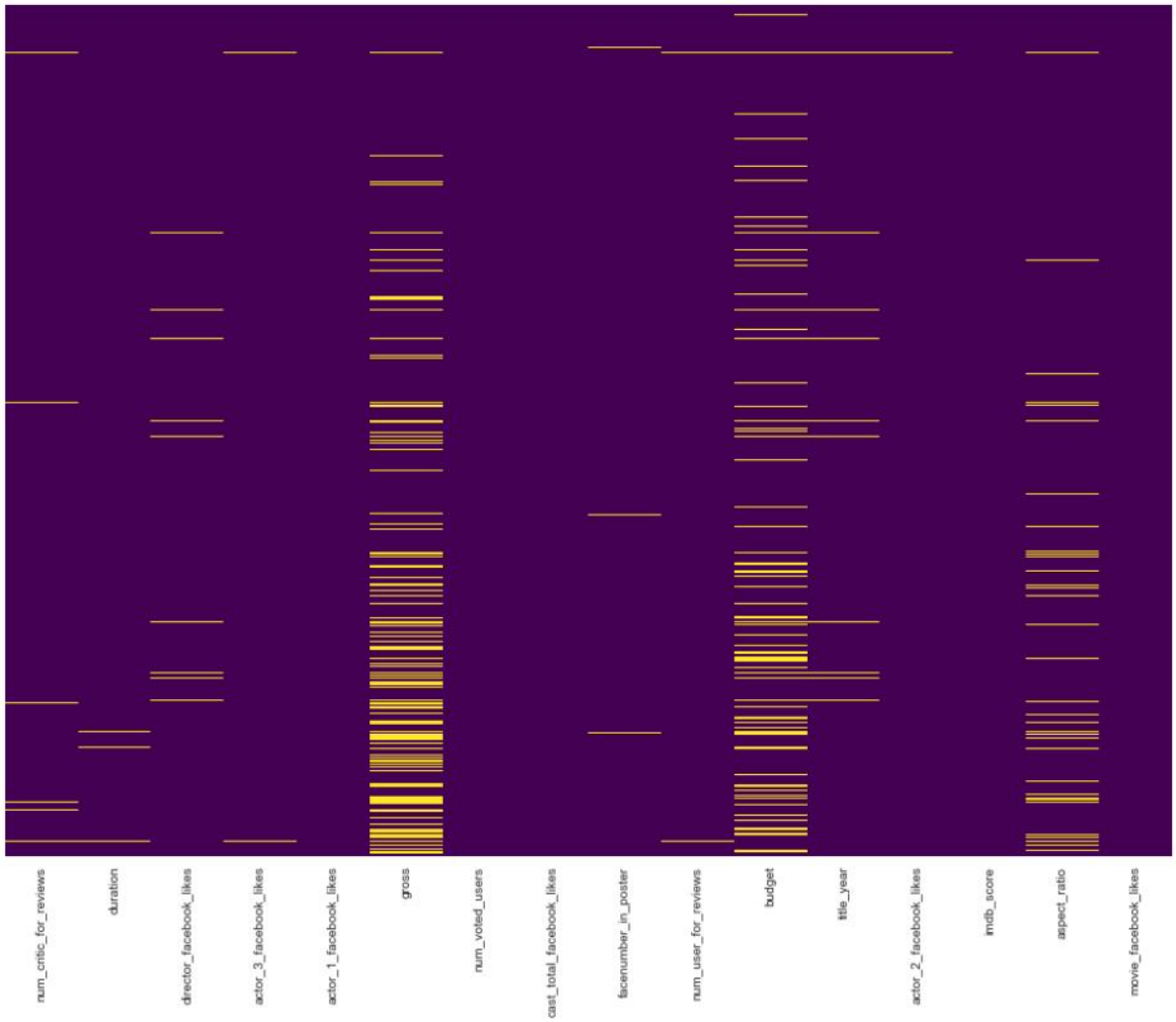
In [10]:
```
total = data.isnull().sum().sort_values(ascending=False)
percent = (data.isnull().sum()/data.isnull().count()).sort_values(ascending=False)
missingdata = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missingdata.head(20)
```

Out[10]:

	Total	Percent
gross	884	0.175292
budget	492	0.097561
aspect_ratio	329	0.065239
content_rating	303	0.060083
plot_keywords	153	0.030339
title_year	108	0.021416
director_name	104	0.020623
director_facebook_likes	104	0.020623
num_critic_for_reviews	50	0.009915
actor_3_name	23	0.004561
actor_3_facebook_likes	23	0.004561
num_user_for_reviews	21	0.004164
color	19	0.003768
duration	15	0.002974
facenumber_in_poster	13	0.002578
actor_2_name	13	0.002578
actor_2_facebook_likes	13	0.002578
language	12	0.002380
actor_1_name	7	0.001388
actor_1_facebook_likes	7	0.001388

In [11]:
```
fig=plt.figure(figsize=(16,12))
sns.heatmap(numdata.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x2623a44aba8>



In [12]:
```
numdata.fillna(numdata.median(),inplace = True)
```

Out[12]:

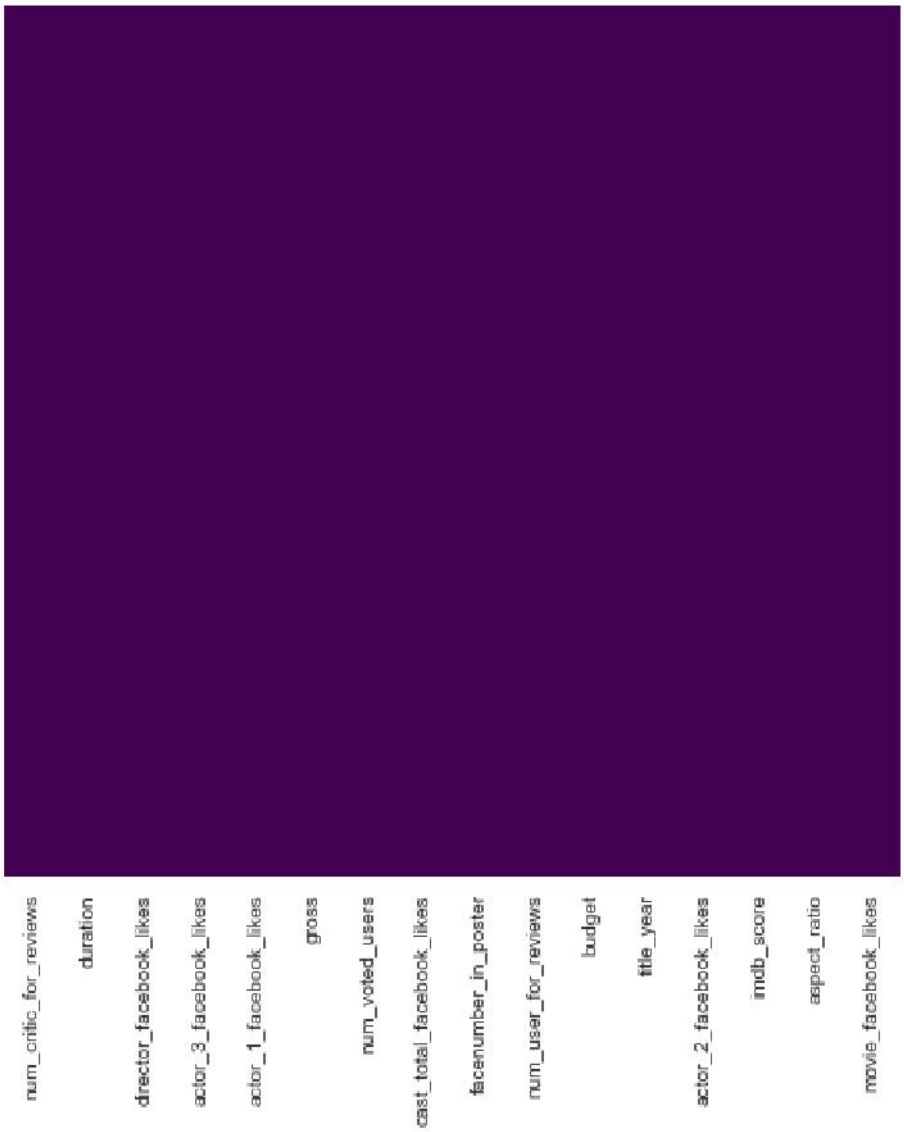
	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes
0	723.0	178.0	0.0	855.0
1	302.0	169.0	563.0	1000.0
2	602.0	148.0	0.0	161.0
3	813.0	164.0	22000.0	23000.0
4	110.0	103.0	131.0	371.5
5	462.0	132.0	475.0	530.0
6	392.0	156.0	0.0	4000.0
7	324.0	100.0	15.0	284.0
8	635.0	141.0	0.0	19000.0
9	375.0	153.0	282.0	10000.0
10	673.0	183.0	0.0	2000.0
11	434.0	169.0	0.0	903.0
12	403.0	106.0	395.0	393.0
13	313.0	151.0	563.0	1000.0
14	450.0	150.0	563.0	1000.0
15	733.0	143.0	0.0	748.0
16	258.0	150.0	80.0	201.0
17	703.0	173.0	0.0	19000.0
18	448.0	136.0	252.0	1000.0
19	451.0	106.0	188.0	718.0
20	422.0	164.0	0.0	773.0
21	599.0	153.0	464.0	963.0
22	343.0	156.0	0.0	738.0
23	509.0	186.0	0.0	773.0
24	251.0	113.0	129.0	1000.0
25	446.0	201.0	0.0	84.0
26	315.0	194.0	0.0	794.0
27	516.0	147.0	94.0	11000.0
28	377.0	131.0	532.0	627.0
29	644.0	124.0	365.0	1000.0
...	...	...	...	...
5013	28.0	79.0	3.0	42.0

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes
5014	58.0	80.0	892.0	492.0
5015	61.0	100.0	0.0	0.0
5016	110.0	90.0	0.0	9.0
5017	1.0	90.0	138.0	138.0
5018	5.0	120.0	589.0	4.0
5019	43.0	91.0	158.0	265.0
5020	110.0	143.0	8.0	8.0
5021	51.0	85.0	157.0	10.0
5022	6.0	60.0	0.0	4.0
5023	22.0	88.0	38.0	211.0
5024	42.0	78.0	91.0	86.0
5025	73.0	108.0	0.0	105.0
5026	81.0	110.0	107.0	45.0
5027	64.0	90.0	397.0	0.0
5028	12.0	83.0	18.0	0.0
5029	78.0	111.0	62.0	6.0
5030	110.0	84.0	5.0	12.0
5031	13.0	82.0	120.0	84.0
5032	10.0	98.0	3.0	152.0
5033	143.0	77.0	291.0	8.0
5034	35.0	80.0	0.0	0.0
5035	56.0	81.0	0.0	6.0
5036	110.0	84.0	2.0	2.0
5037	14.0	95.0	0.0	133.0
5038	1.0	87.0	2.0	318.0
5039	43.0	43.0	49.0	319.0
5040	13.0	76.0	0.0	0.0
5041	14.0	100.0	0.0	489.0
5042	43.0	90.0	16.0	16.0

5043 rows × 16 columns

In [13]:
```
fig=plt.figure(figsize=(8,8))
sns.heatmap(numdata.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x2623a8bb3c8>



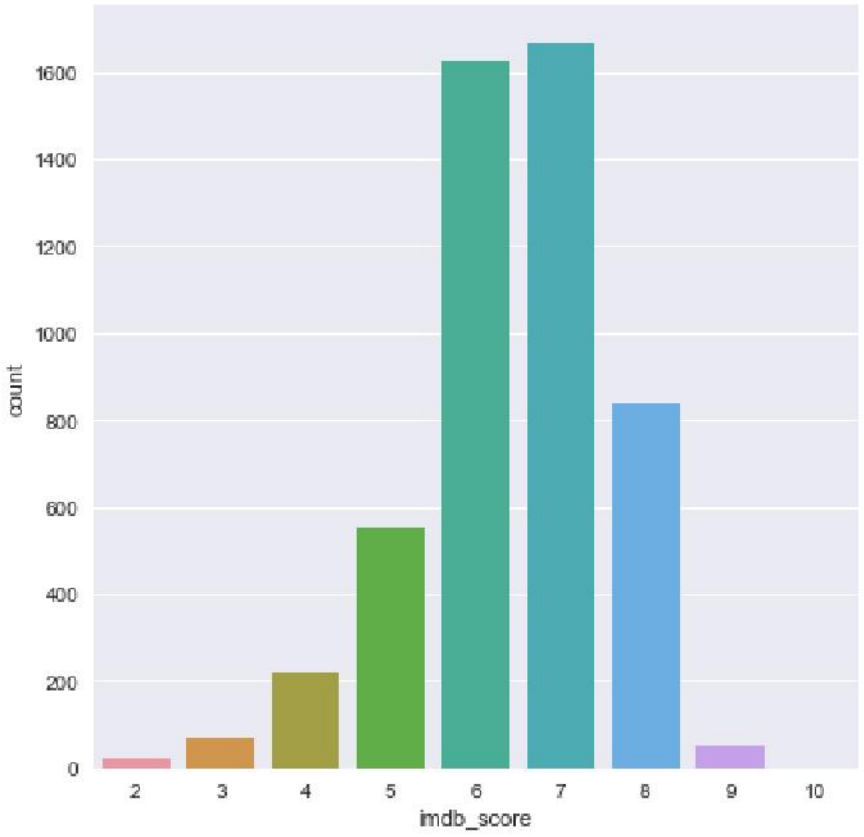
Data Analysis & Visualizations

In [14]:
```
data['imdb_score'] = data['imdb_score'].apply(lambda x:int(round(x)))
```

In [15]:
```
fig=plt.figure(figsize=(40,30))
sns.factorplot('imdb_score',kind='count',data=data,size=6)
```

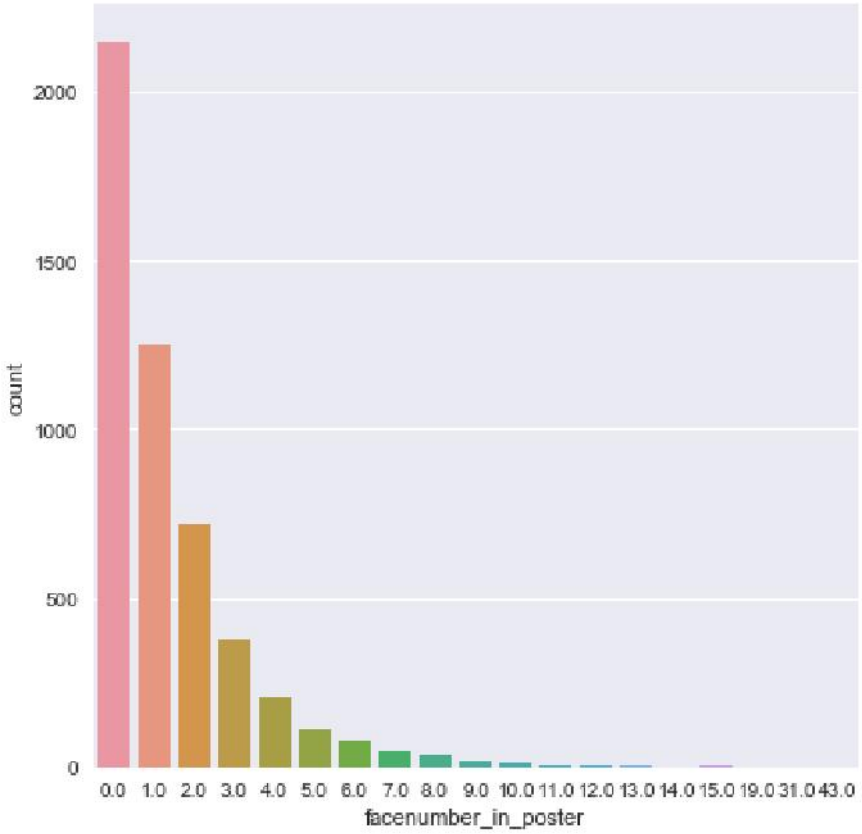
Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x2623a481278>

<matplotlib.figure.Figure at 0x2623a903710>
```



In [16]:
```
sns.factorplot('facenumber_in_poster',kind='count',data=data, size=6)
```

Out[16]: <seaborn.axisgrid.FacetGrid at 0x2623ade1a90>

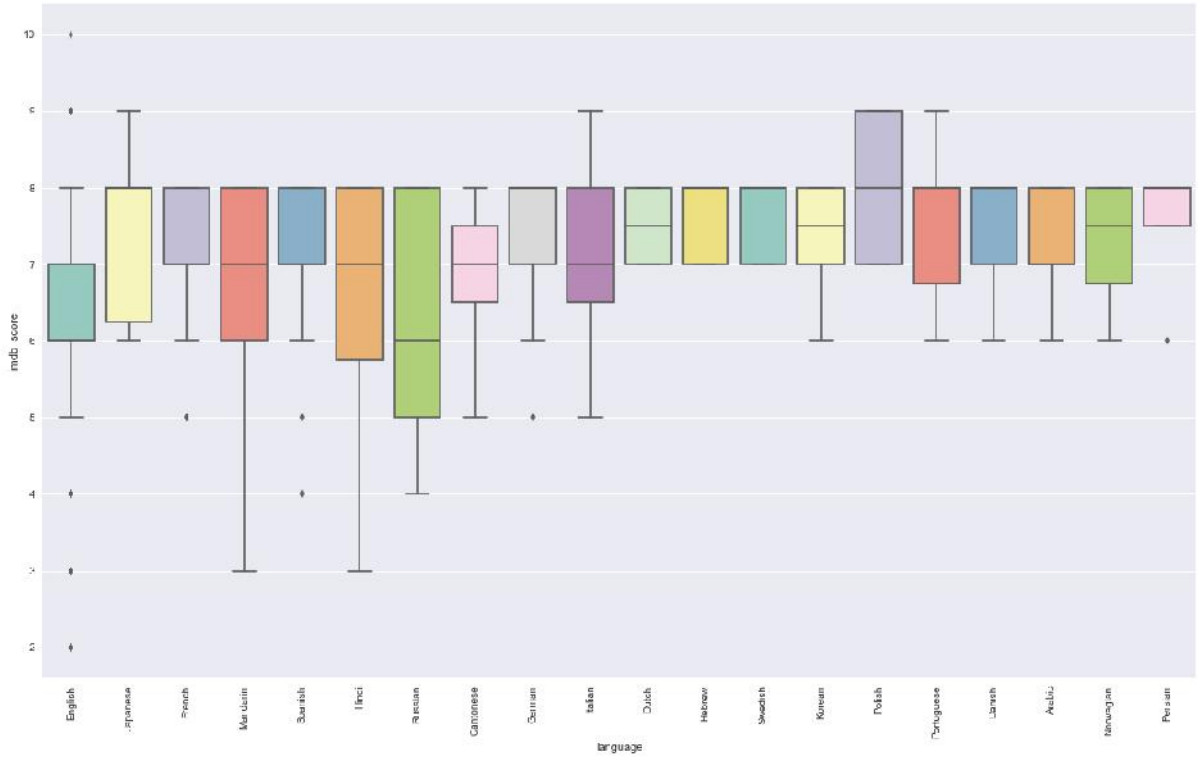


In [17]:
```
tmp = data['language'].value_counts()
lang_list=tmp[tmp>3].index.tolist()
lang_list
```

Out[17]:
```
['English',
 'French',
 'Spanish',
 'Hindi',
 'Mandarin',
 'German',
 'Japanese',
 'Italian',
 'Cantonese',
 'Russian',
 'Portuguese',
 'Korean',
 'Danish',
 'Arabic',
 'Swedish',
 'Hebrew',
 'Polish',
 'Norwegian',
 'Persian',
 'Dutch']
```

IMDB Score vs Language

In [20]:
```
plt.figure(figsize = (20, 12))
sns.boxplot(x = 'language',y='imdb_score', data = data[data['language'].isin(l
ang_list)],palette="Set3")
xt = plt.xticks(rotation=90)
```



In [19]:
```
data[data['language'].isin(lang_list)].head()
```

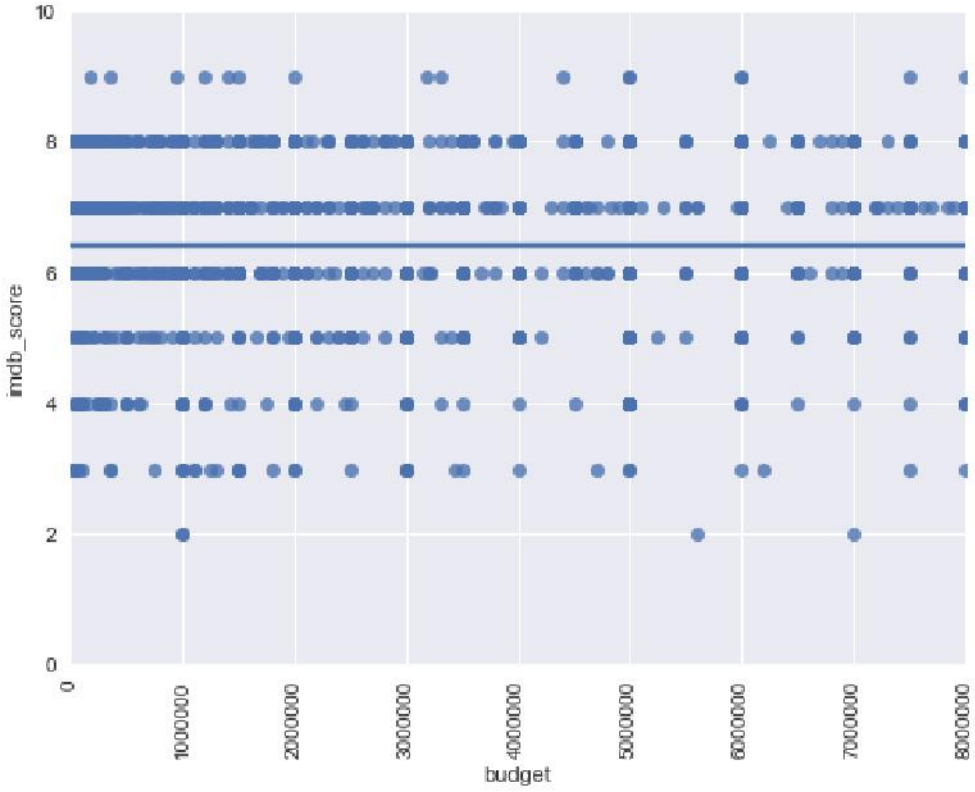
Out[19]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	acto
0	Color	James Cameron	723.0	178.0	0.0	855.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000
2	Color	Sam Mendes	602.0	148.0	0.0	161.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	2300
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0

5 rows × 28 columns

IMDB Score vs Budget

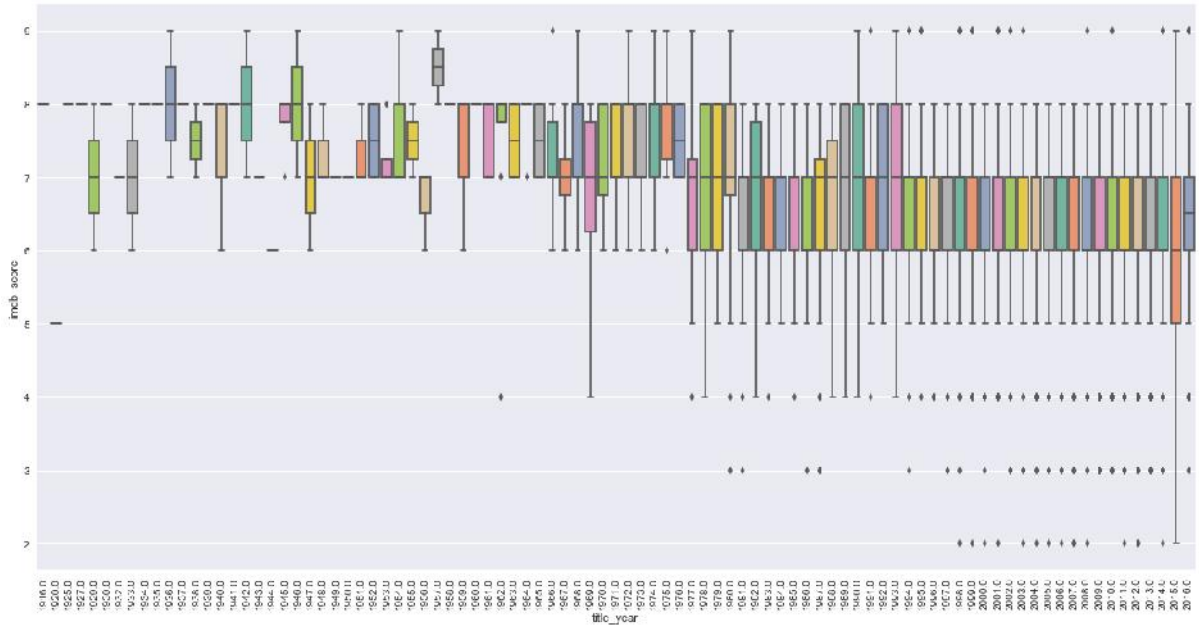
In [359]:
```
f, axis = plt.subplots(figsize=(8, 6))
#fig=plt.figure(figsize=(20,10))
fig=sns.regplot(x='budget',y='imdb_score',data=data)
xt = plt.xticks(rotation=90)
fig.axis(ymin=0, ymax=10);
fig.axis(xmin=0, xmax=8000000);
```



Buget is not correlated with imdb rating.

IMDB Score vs Title_year

In [21]:
```
fig=plt.figure(figsize=(20,10))
sns.boxplot(x='title_year',y='imdb_score',data=data,width=0.8,palette="Set2")
xt = plt.xticks(rotation=90)
```

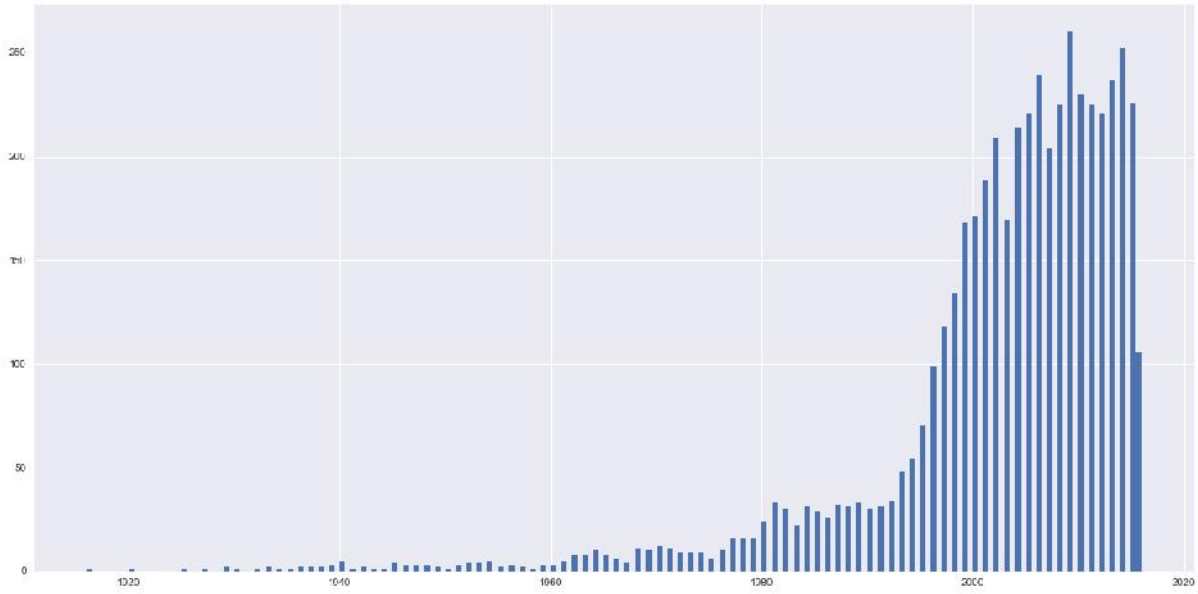


In [20]:
```
allyears=data['title_year']
decade=[1910,1920,1930,1940,1950,1960,1970,1980,1990,2000,2010,2016]
decadecat=pd.cut(allyears,decade)
yearcount=pd.value_counts(decadecat)
yearcount
```

IMDB Score Count vs Movie Year

In [22]:
```
fig=plt.figure(figsize=(20,10))
data['title_year'].hist(bins=200)
```

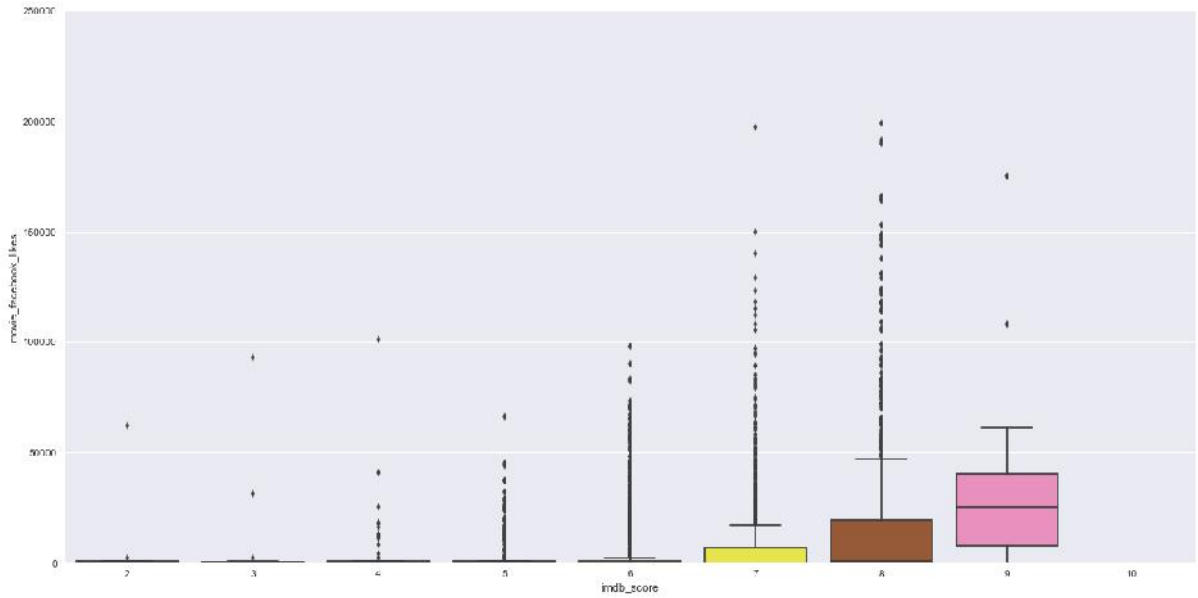
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x262efc9abe0>



IMDB Score vs Facebook Popularity

In [22]:
```
fig=plt.figure(figsize=(20,10))
sns.boxplot(x='imdb_score',y='movie_facebook_likes',data=data,palette="Set1")
plt.ylim(0,250000)
```

Out[22]: (0, 250000)

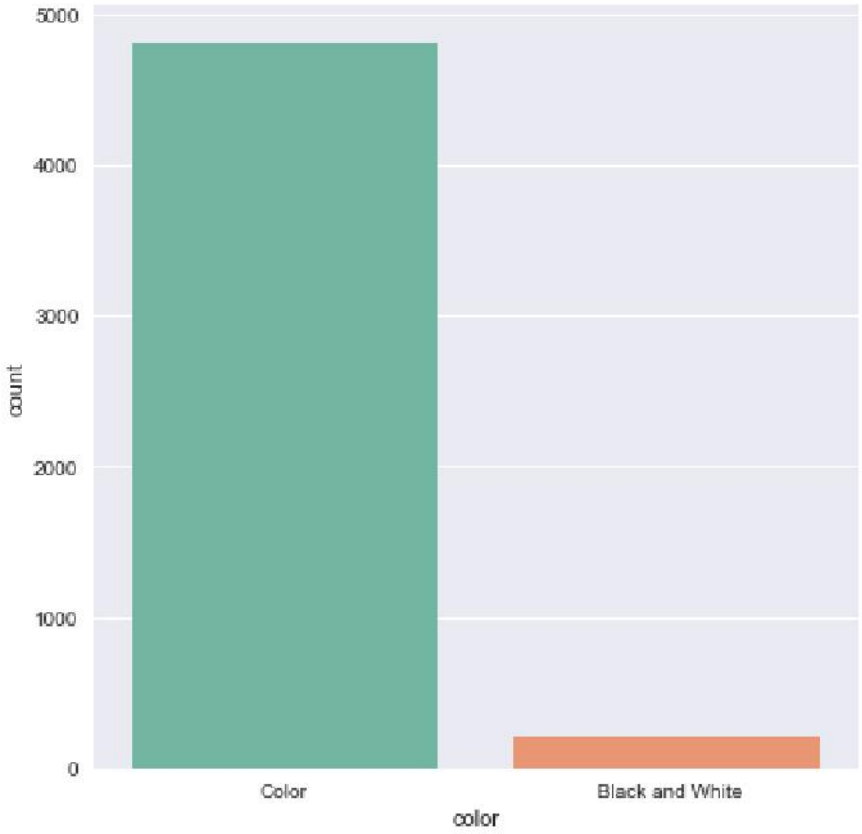


We could see that movies with high amount of facebook like have good Imdb rating

Distibution of Count feature

```
In [32]: sns.factorplot('color',kind='count',data=data, size=6, palette="Set2")
```

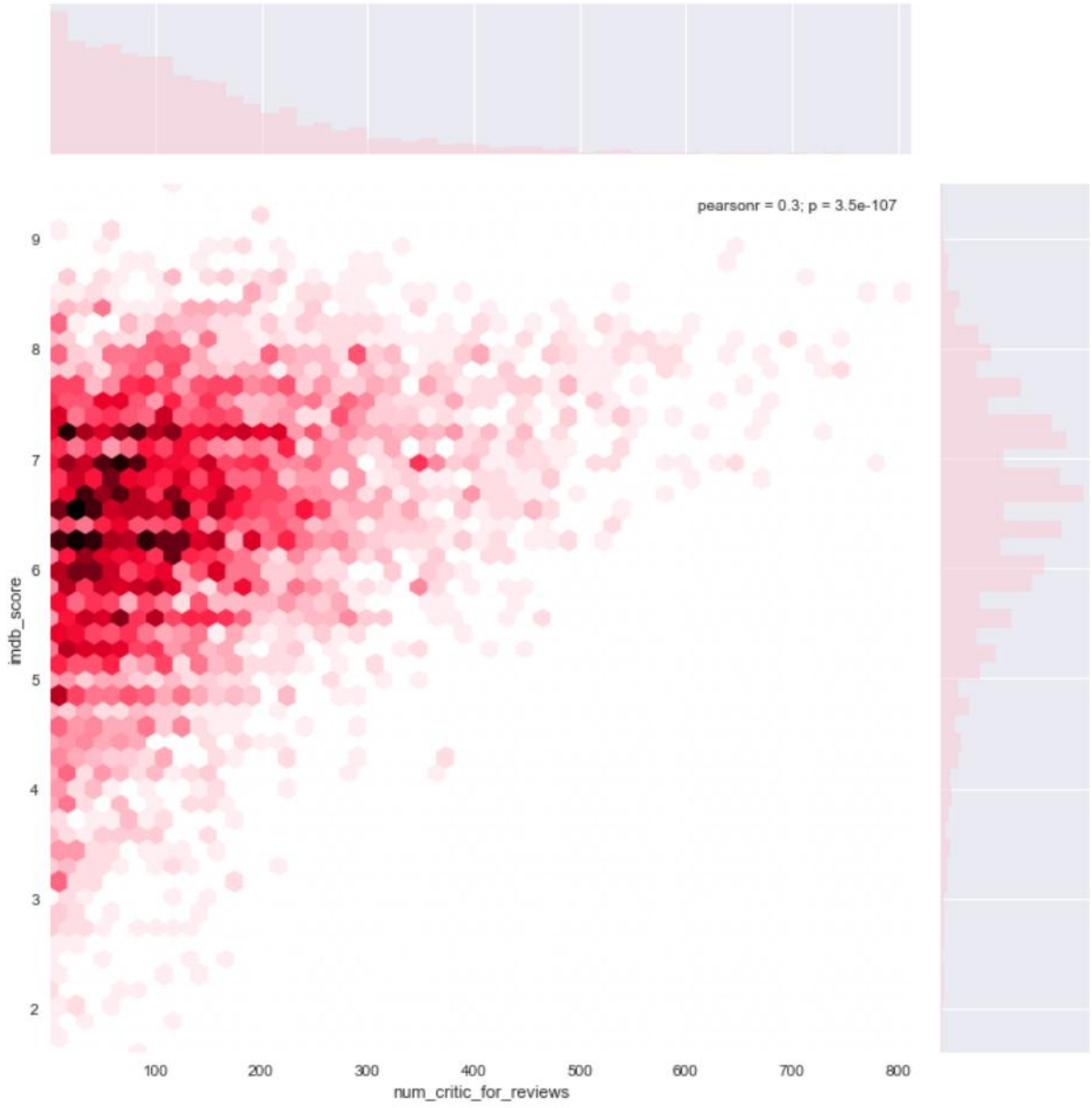
```
Out[32]: <seaborn.axisgrid.FacetGrid at 0x2623bfda9e8>
```



IMDB Score vs num_critic_for_reviews

In [34]:
```
sns.jointplot(x='num_critic_for_reviews',y='imdb_score',data=numdata,kind='hex',size=10,dropna=True,color='pink')
```

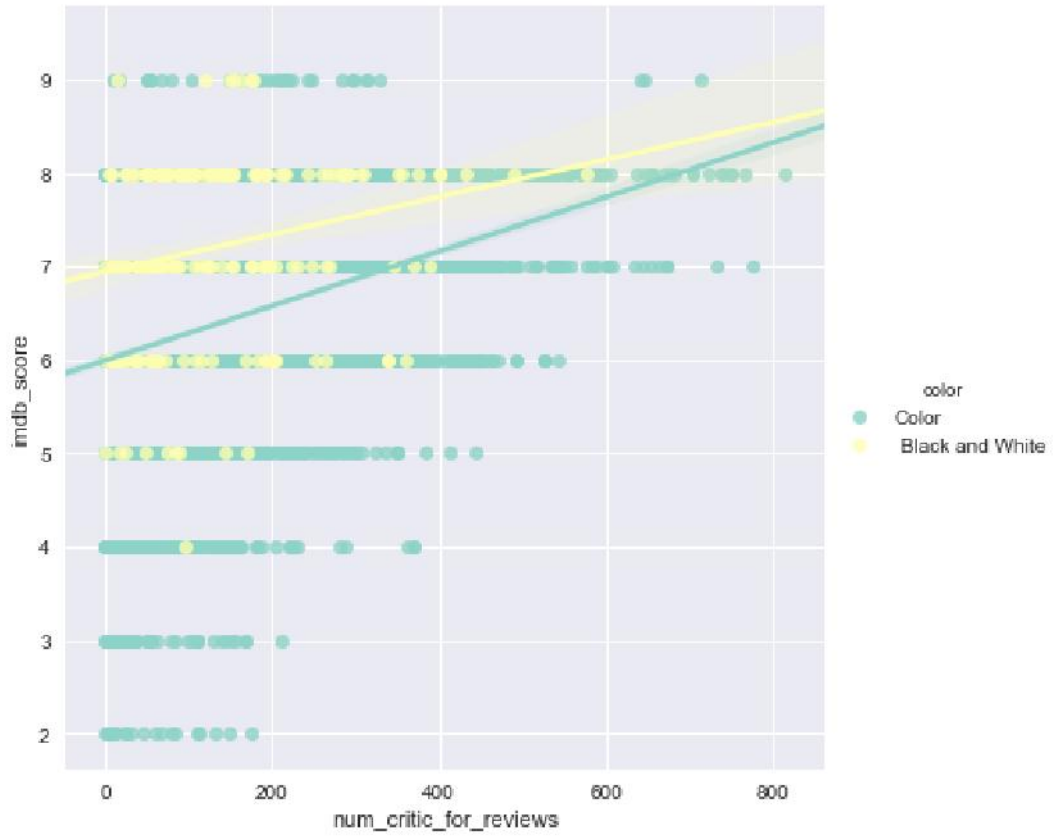
Out[34]: <seaborn.axisgrid.JointGrid at 0x2623d7370f0>



IMDB Score vs num_critic_for_reviews with hue of Color

In [40]:
```
sns.lmplot(x='num_critic_for_reviews',y='imdb_score',data=data,hue='color',size=6,palette="Set3")
```

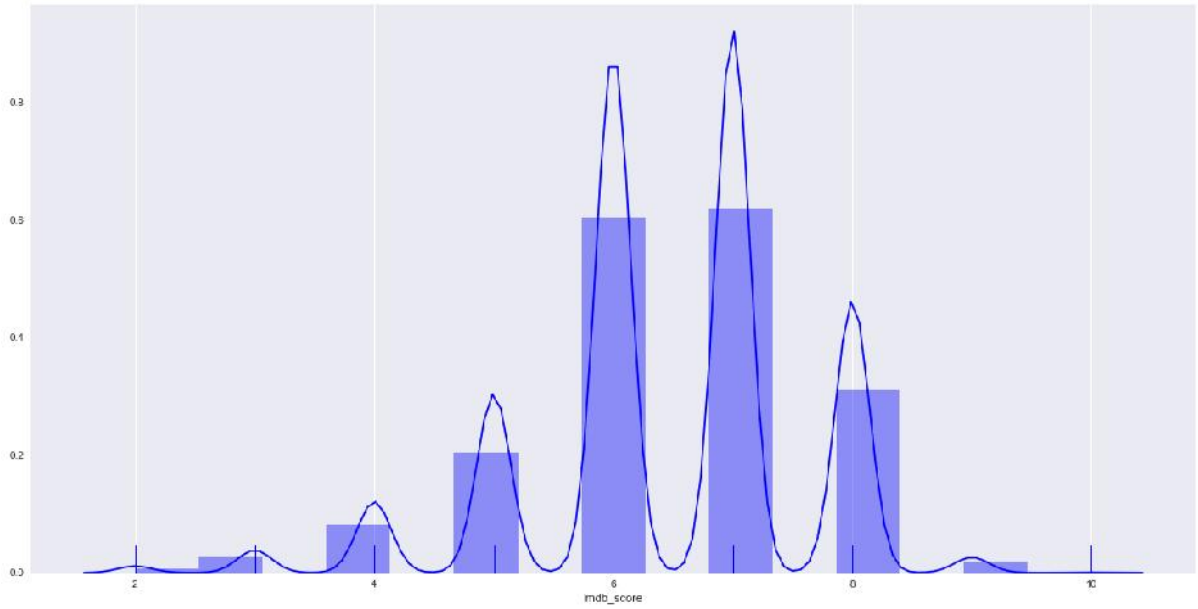
Out[40]: <seaborn.axisgrid.FacetGrid at 0x2624018c940>



###IMDB rating Distribution

In [41]:
```
fig=plt.figure(figsize=(20,10))
sns.distplot(data['imdb_score'],bins=15,rug=True,color='b')
```

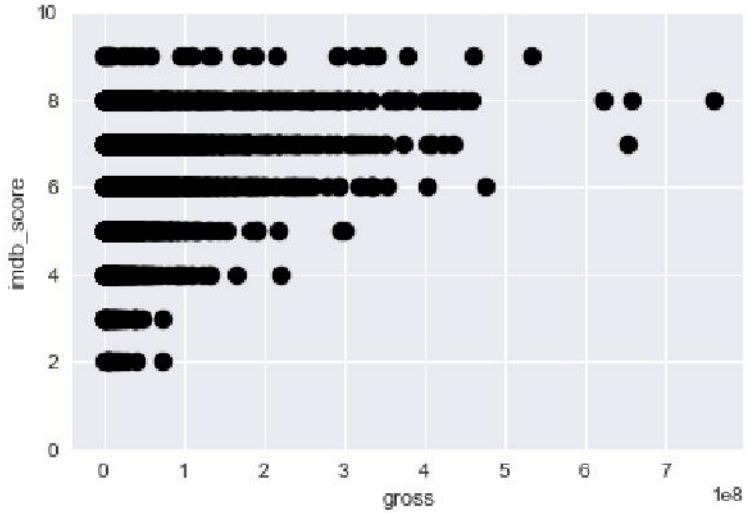
Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x26240245908>



IMDB Score vs gross

In [46]:
```
fig=plt.figure(figsize=(20,20))
data.plot.scatter(x='gross', y='imdb_score', ylim=(0,10), linewidths=5,edgecol
ors='black');
```

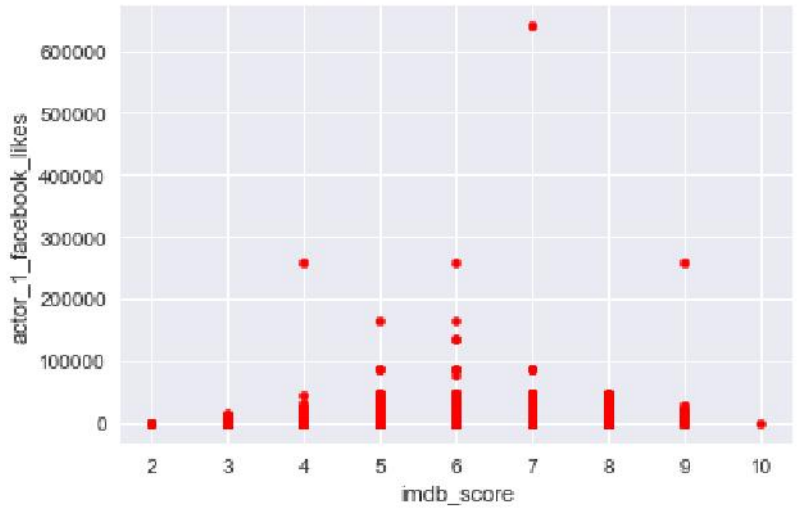
<matplotlib.figure.Figure at 0x262478594a8>

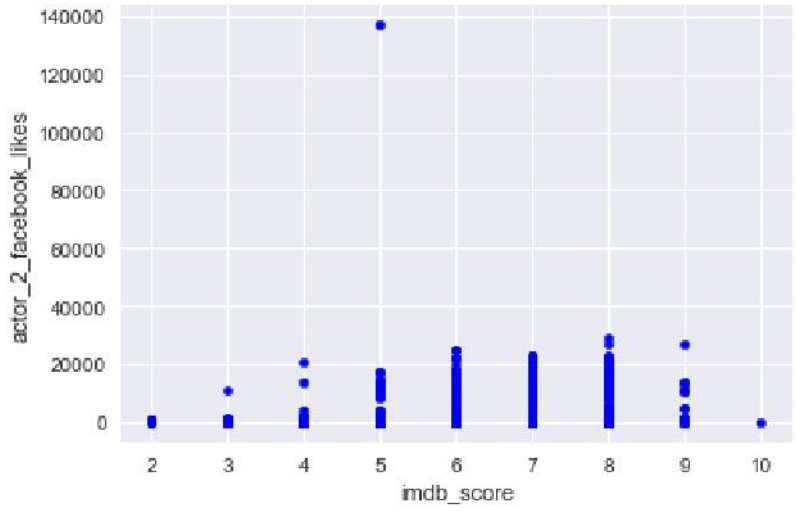


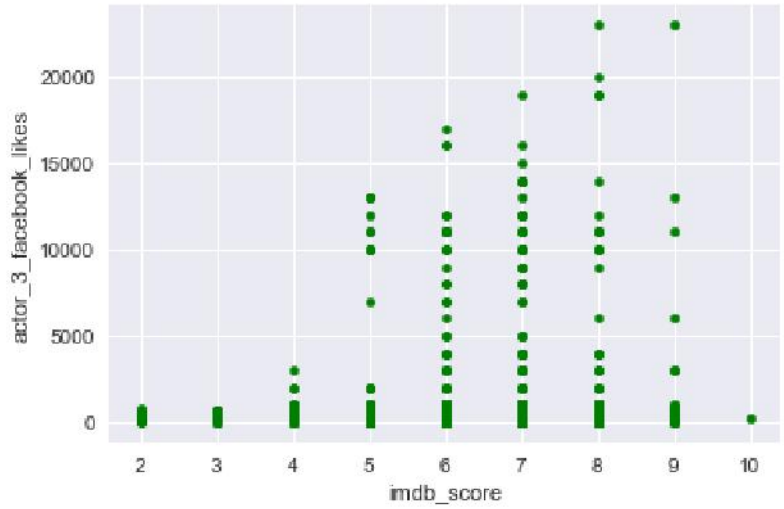
IMDB Score vs Actors Facebook Likes

In [55]:
```
y=data['imdb_score']
data.plot.scatter('imdb_score','actor_1_facebook_likes',color='red')
data.plot.scatter('imdb_score','actor_2_facebook_likes',color='blue')
data.plot.scatter('imdb_score','actor_3_facebook_likes',color='green')
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x26248dcd3c8>







We can see some outliers in actor_1_facebook_likes and actor_2_facebook_likes

Correlation between Features

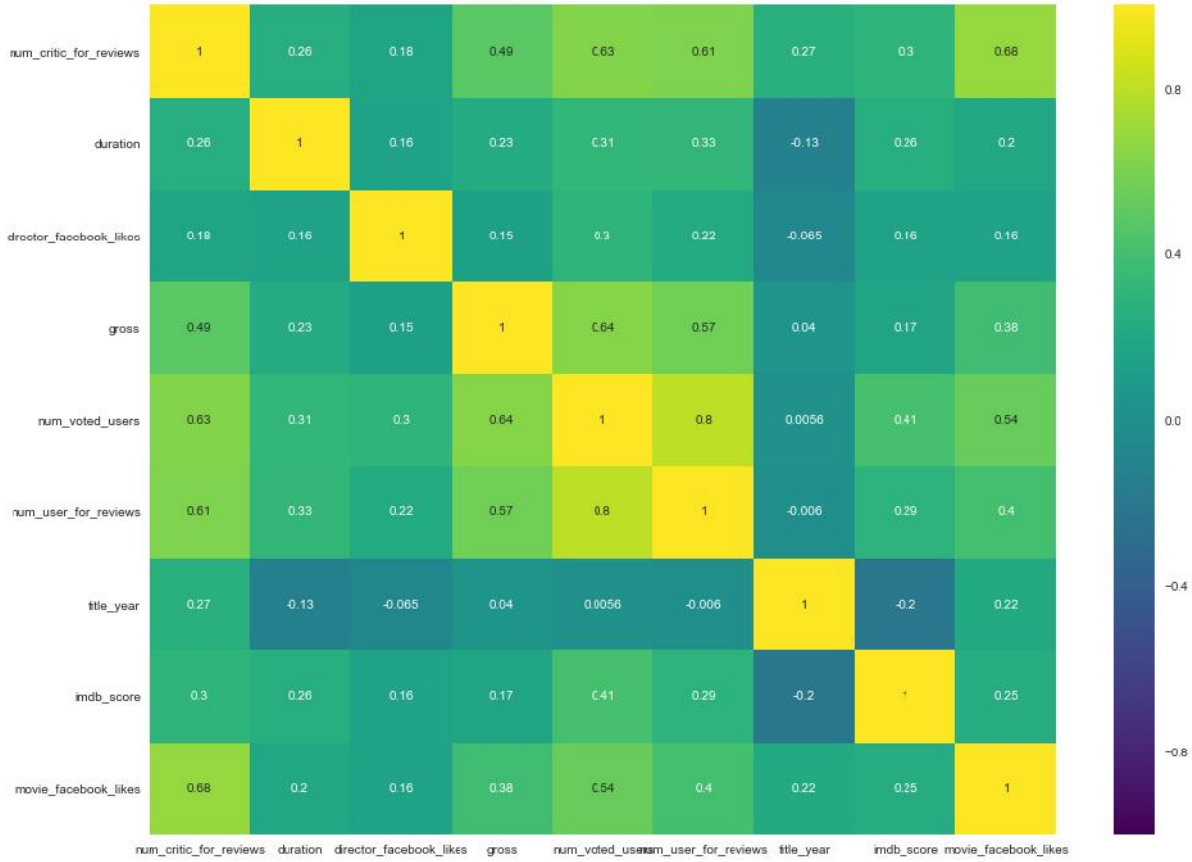
In [24]:
```
corrm=numdata.corr()
corrm
```

Out[24]:

	num_critic_for_reviews	duration	director_facebook_likes	a
num_critic_for_reviews	1.000000	0.257531	0.183657	0
duration	0.257531	1.000000	0.162048	0
director_facebook_likes	0.183657	0.162048	1.000000	0
actor_3_facebook_likes	0.271857	0.123717	0.121046	1
actor_1_facebook_likes	0.191033	0.088779	0.092232	0
gross	0.486001	0.231116	0.145844	0
num_voted_users	0.625106	0.314881	0.297940	0
cast_total_facebook_likes	0.263886	0.123160	0.120986	0
facenumber_in_poster	-0.033749	0.013068	-0.041472	0
num_user_for_reviews	0.609531	0.328020	0.223538	0
budget	0.121572	0.069681	0.022747	0
title_year	0.266797	-0.127405	-0.064740	0
actor_2_facebook_likes	0.282912	0.132069	0.120900	0
imdb_score	0.302465	0.260965	0.162838	0
aspect_ratio	-0.053660	-0.089716	-0.015580	-(
movie_facebook_likes	0.683412	0.196726	0.162060	0

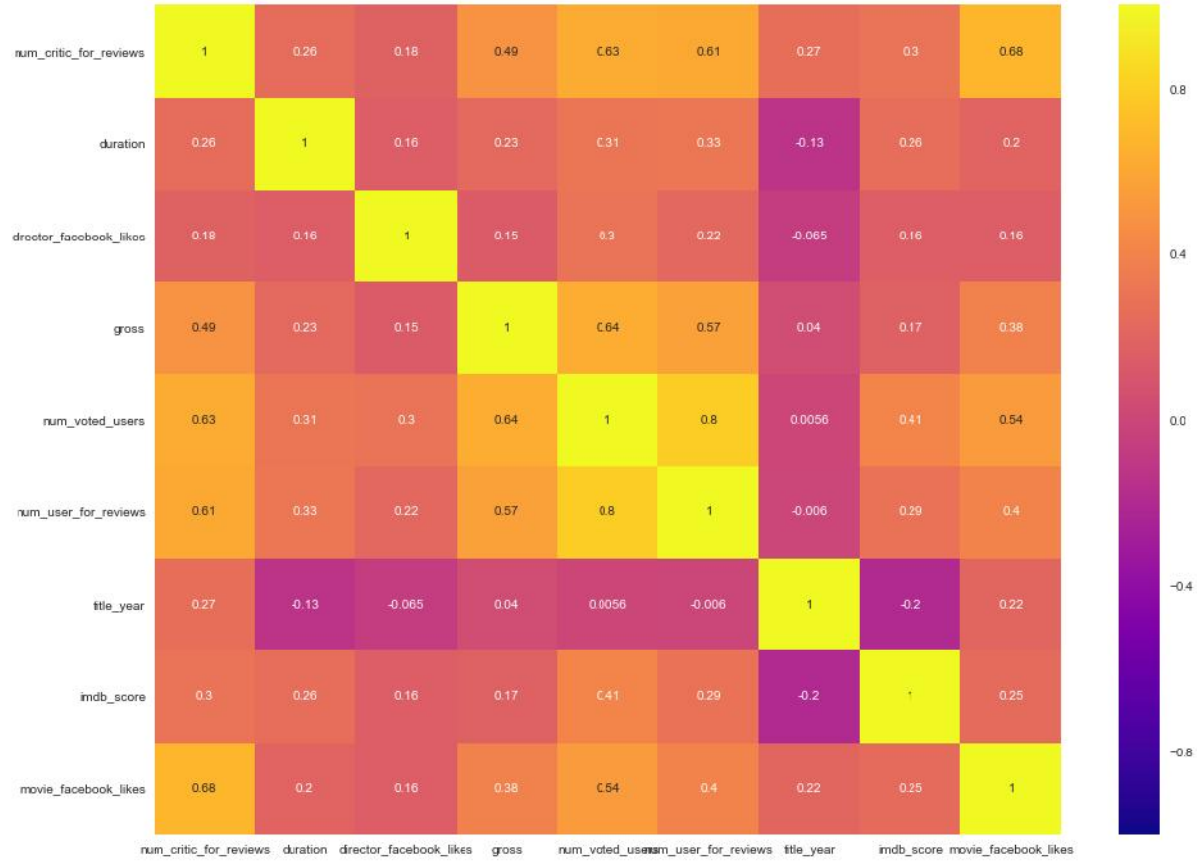
In [57]:
```
fig=plt.figure(figsize=(16,12))
sns.heatmap(numdata.corr(),annot=True,cmap="viridis")
```

Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x26248f47160>

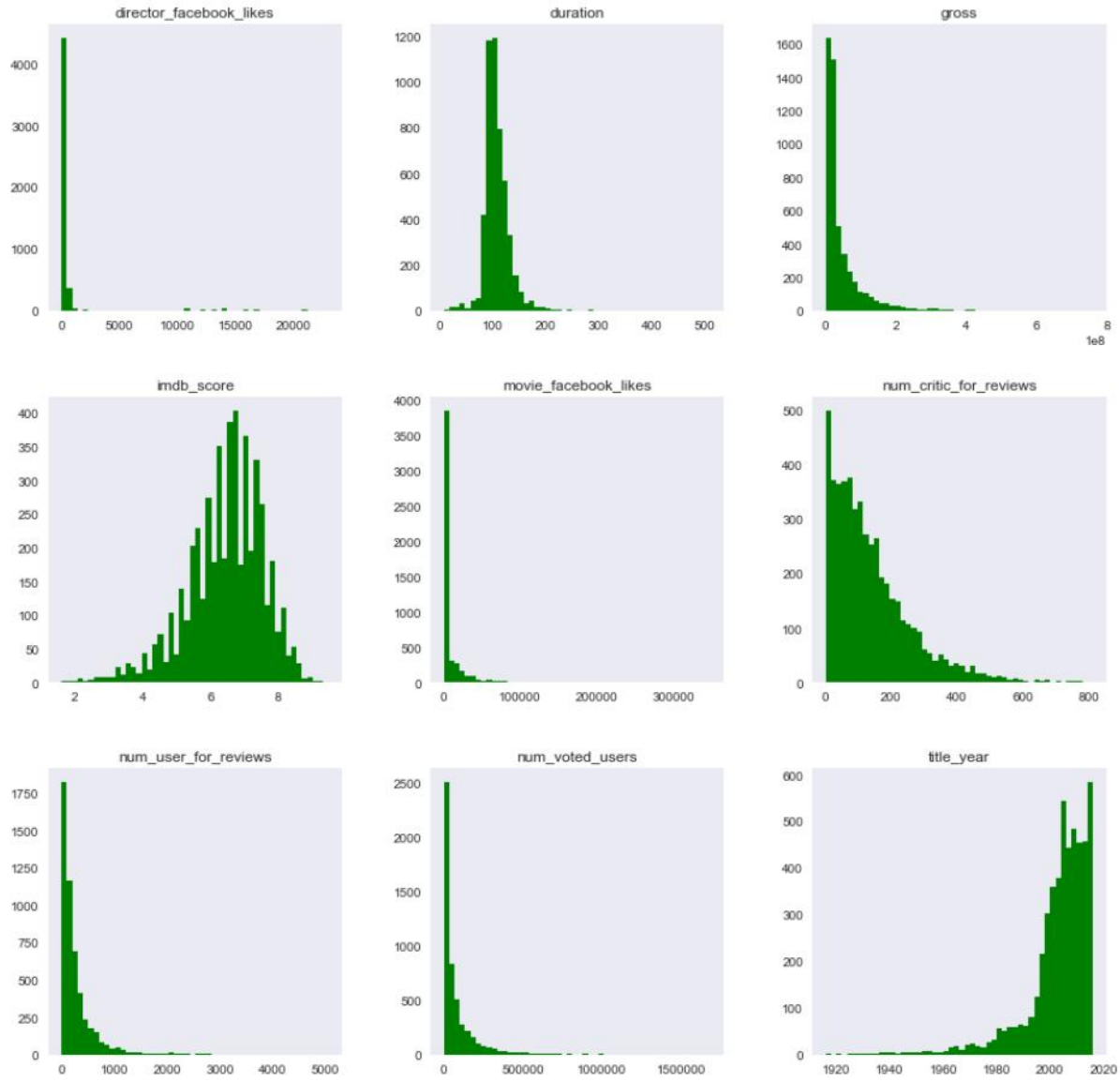


In [26]:
```
topcor=corrm.index[abs(corrm['imdb_score'])>0.1]
numdata=numdata[topcor]
fig=plt.figure(figsize=(16,12))
sns.heatmap(numdata.corr(),annot=True,cmap="plasma")
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x2623aef6a90>



In [61]:
```
pd.DataFrame.hist(numdata, figsize = [15,15],grid=False, xlabelsize=10, ylabelsize=10,bins=50,color="green");
```



In [372]:
```
numdata.columns
```

Out[372]:
```
Index(['num_critic_for_reviews', 'duration', 'director_facebook_likes',
       'gross', 'num_voted_users', 'num_user_for_reviews', 'title_year',
       'imdb_score', 'movie_facebook_likes'],
      dtype='object')
```

In [373]:
```
numdata = numdata[['num_critic_for_reviews', 'duration', 'director_facebook_likes',
       'gross', 'num_voted_users', 'num_user_for_reviews', 'title_year',
       'movie_facebook_likes']]
```

In [112]:
```
numdata
```

Out[112]:

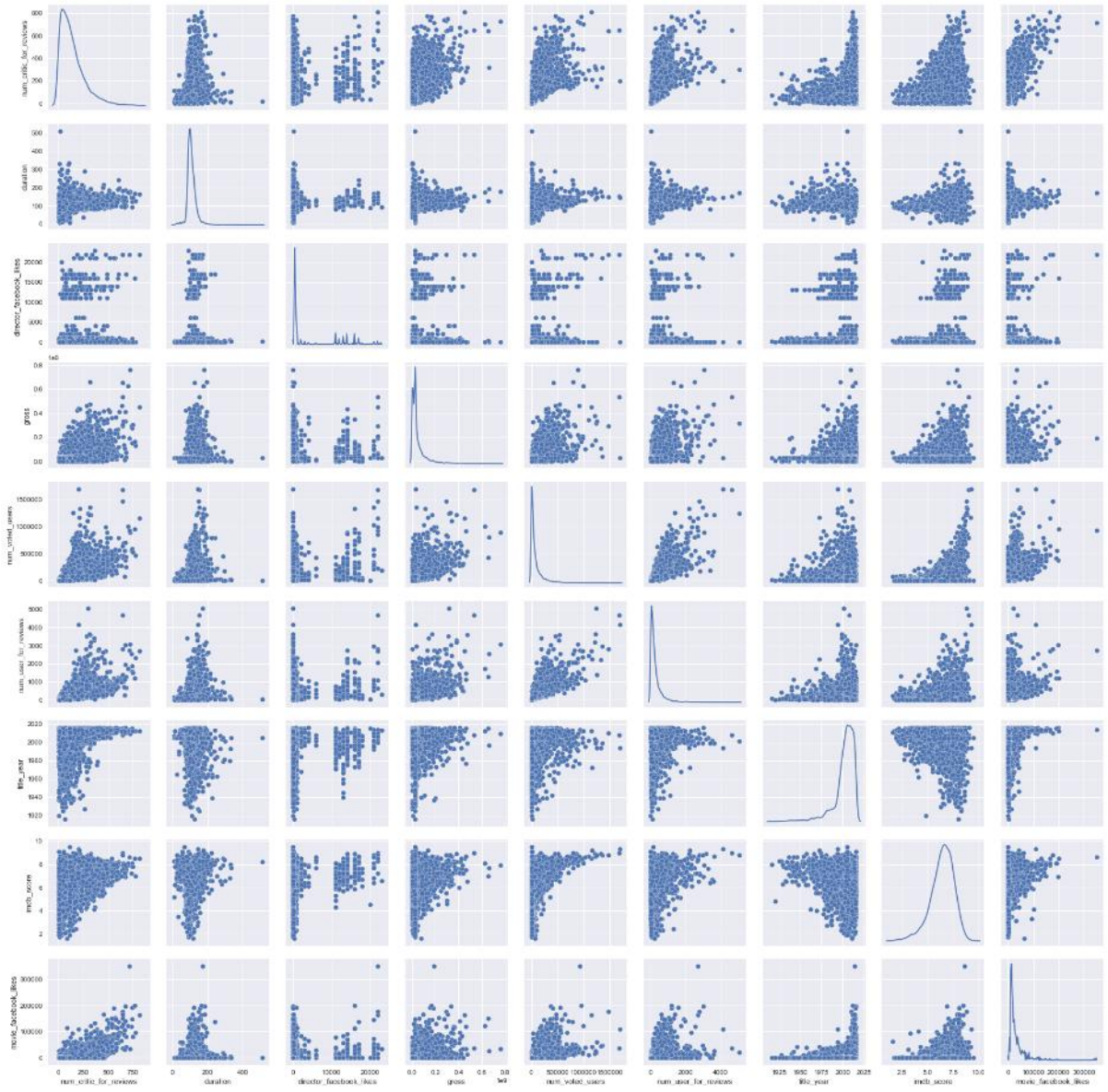
	num_critic_for_reviews	duration	director_facebook_likes	gross	num_vote
0	723.0	178.0	0.0	760505847.0	886204
1	302.0	169.0	563.0	309404152.0	471220
2	602.0	148.0	0.0	200074175.0	275868
3	813.0	164.0	22000.0	448130642.0	1144337
4	110.0	103.0	131.0	25517500.0	8
5	462.0	132.0	475.0	73058679.0	212204
6	392.0	156.0	0.0	336530303.0	383056
7	324.0	100.0	15.0	200807262.0	294810
8	635.0	141.0	0.0	458991599.0	462669
9	375.0	153.0	282.0	301956980.0	321795
10	673.0	183.0	0.0	330249062.0	371639
11	434.0	169.0	0.0	200069408.0	240396
12	403.0	106.0	395.0	168368427.0	330784
13	313.0	151.0	563.0	423032628.0	522040
14	450.0	150.0	563.0	89289910.0	181792
15	733.0	143.0	0.0	291021565.0	548573
16	258.0	150.0	80.0	141614023.0	149922
17	703.0	173.0	0.0	623279547.0	995415
18	448.0	136.0	252.0	241063875.0	370704
19	451.0	106.0	188.0	179020854.0	268154
20	422.0	164.0	0.0	255108370.0	354228
21	599.0	153.0	464.0	262030663.0	451803
22	343.0	156.0	0.0	105219735.0	211765
23	509.0	186.0	0.0	258355354.0	483540
24	251.0	113.0	129.0	70083519.0	149019
25	446.0	201.0	0.0	218051260.0	316018
26	315.0	194.0	0.0	658672302.0	793059
27	516.0	147.0	94.0	407197282.0	272670
28	377.0	131.0	532.0	65173160.0	202382
29	644.0	124.0	365.0	652177271.0	418214
...	...	...	...	...	...
5013	28.0	79.0	3.0	25517500.0	493

	num_critic_for_reviews	duration	director_facebook_likes	gross	num_vote
5014	58.0	80.0	892.0	25517500.0	15091
5015	61.0	100.0	0.0	1227508.0	15103
5016	110.0	90.0	0.0	25517500.0	57
5017	1.0	90.0	138.0	25517500.0	114
5018	5.0	120.0	589.0	25517500.0	2986
5019	43.0	91.0	158.0	25517500.0	3836
5020	110.0	143.0	8.0	25517500.0	125
5021	51.0	85.0	157.0	192467.0	4067
5022	6.0	60.0	0.0	25517500.0	70
5023	22.0	88.0	38.0	76382.0	1194
5024	42.0	78.0	91.0	25517500.0	1771
5025	73.0	108.0	0.0	180483.0	16792
5026	81.0	110.0	107.0	136007.0	3924
5027	64.0	90.0	397.0	673780.0	4555
5028	12.0	83.0	18.0	25517500.0	57
5029	78.0	111.0	62.0	94596.0	6318
5030	110.0	84.0	5.0	25517500.0	156
5031	13.0	82.0	120.0	25517500.0	133
5032	10.0	98.0	3.0	25517500.0	438
5033	143.0	77.0	291.0	424760.0	72639
5034	35.0	80.0	0.0	70071.0	589
5035	56.0	81.0	0.0	2040920.0	52055
5036	110.0	84.0	2.0	25517500.0	36
5037	14.0	95.0	0.0	4584.0	1338
5038	1.0	87.0	2.0	25517500.0	629
5039	43.0	43.0	49.0	25517500.0	73839
5040	13.0	76.0	0.0	25517500.0	38
5041	14.0	100.0	0.0	10443.0	1255
5042	43.0	90.0	16.0	85222.0	4285

5043 rows × 8 columns

In [65]:
```
sns.pairplot(numdata,diag_kind='kde',palette='pink')
```

Out[65]:
```
<seaborn.axisgrid.PairGrid at 0x26257893080>
```



Working on Categorical features

In [67]:
```
categorical_features
```

Out[67]:
```
Index(['color', 'director_name', 'actor_2_name', 'genres', 'actor_1_name',
       'movie_title', 'actor_3_name', 'plot_keywords', 'movie_imdb_link',
       'language', 'country', 'content_rating'],
      dtype='object')
```

In [68]:
```
catdata = data[categorical_features]
```

In [69]:
```
catdata.head()
```

Out[69]:

	color	director_name	actor_2_name	genres	actor_1_name	mov
0	Color	James Cameron	Joel David Moore	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avat
1	Color	Gore Verbinski	Orlando Bloom	Action|Adventure|Fantasy	Johnny Depp	Pirat the Caril At W End
2	Color	Sam Mendes	Rory Kinnear	Action|Adventure|Thriller	Christoph Waltz	Spec
3	Color	Christopher Nolan	Christian Bale	Action|Thriller	Tom Hardy	The Knig Rise
4	NaN	Doug Walker	Rob Walker	Documentary	Doug Walker	Star Epis - The Awa ...

In [70]:
```
#missing data
total = catdata.isnull().sum().sort_values(ascending=False)
percent = (catdata.isnull().sum()/catdata.isnull().count()).sort_values(ascend
ing=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

Out[70]:

	Total	Percent
content_rating	303	0.060083
plot_keywords	153	0.030339
director_name	104	0.020623
actor_3_name	23	0.004561
color	19	0.003768
actor_2_name	13	0.002578
language	12	0.002380
actor_1_name	7	0.001388
country	5	0.000991
movie_imdb_link	0	0.000000
movie_title	0	0.000000
genres	0	0.000000

In [71]:
```
catdata.fillna(value = 'NULL',inplace = True)
```

Segment all the categories in Genres to individual Features

In [72]:
```
df_genres = pd.DataFrame(catdata['genres'])
df_genres = pd.DataFrame(df_genres.genres.str.split('|').tolist(),columns = [
"Genre_"+str(i) for i in  range(0,8)] )
catdata.drop('genres',inplace = True, axis = 1)
catdata = catdata.merge(df_genres,left_index = True,right_index = True)
```

In [73]:
```
catdata.head()
```

Out[73]:

	color	director_name	actor_2_name	actor_1_name	movie_title	actor_3_name	plot_k
0	Color	James Cameron	Joel David Moore	CCH Pounder	Avatar	Wes Studi	avatar
1	Color	Gore Verbinski	Orlando Bloom	Johnny Depp	Pirates of the Caribbean: At World's End	Jack Davenport	godde propo
2	Color	Sam Mendes	Rory Kinnear	Christoph Waltz	Spectre	Stephanie Sigman	bomb
3	Color	Christopher Nolan	Christian Bale	Tom Hardy	The Dark Knight Rises	Joseph Gordon-Levitt	decep offi...
4	NULL	Doug Walker	Rob Walker	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	NULL	NULL

In [74]:
```
catdata.drop(['movie_imdb_link','Genre_6','Genre_7'],inplace = True, axis = 1)
```

In [75]:
```
catdata
```

Out[75]:

	color	director_name	actor_2_name	actor_1_name	movie_title	actor_3_name	p
0	Color	James Cameron	Joel David Moore	CCH Pounder	Avatar	Wes Studi	a
1	Color	Gore Verbinski	Orlando Bloom	Johnny Depp	Pirates of the Caribbean: At World's End	Jack Davenport	g p
2	Color	Sam Mendes	Rory Kinnear	Christoph Waltz	Spectre	Stephanie Sigman	b
3	Color	Christopher Nolan	Christian Bale	Tom Hardy	The Dark Knight Rises	Joseph Gordon-Levitt	d d
4	NULL	Doug Walker	Rob Walker	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	NULL	N
5	Color	Andrew Stanton	Samantha Morton	Daryl Sabara	John Carter	Polly Walker	a
6	Color	Sam Raimi	James Franco	J.K. Simmons	Spider-Man 3	Kirsten Dunst	s
7	Color	Nathan Greno	Donna Murphy	Brad Garrett	Tangled	M.C. Gainey	1
8	Color	Joss Whedon	Robert Downey Jr.	Chris Hemsworth	Avengers: Age of Ultron	Scarlett Johansson	a
9	Color	David Yates	Daniel Radcliffe	Alan Rickman	Harry Potter and the Half-Blood Prince	Rupert Grint	b
10	Color	Zack Snyder	Lauren Cohan	Henry Cavill	Batman v Superman: Dawn of Justice	Alan D. Purwin	b r
11	Color	Bryan Singer	Marlon Brando	Kevin Spacey	Superman Returns	Frank Langella	d
12	Color	Marc Forster	Mathieu Amalric	Giancarlo Giannini	Quantum of Solace	Rory Kinnear	a
13	Color	Gore Verbinski	Orlando Bloom	Johnny Depp	Pirates of the Caribbean: Dead Man's Chest	Jack Davenport	b

	color	director_name	actor_2_name	actor_1_name	movie_title	actor_3_name	p
14	Color	Gore Verbinski	Ruth Wilson	Johnny Depp	The Lone Ranger	Tom Wilkinson	h
15	Color	Zack Snyder	Christopher Meloni	Henry Cavill	Man of Steel	Harry Lennix	b a
16	Color	Andrew Adamson	Pierfrancesco Favino	Peter Dinklage	The Chronicles of Narnia: Prince Caspian	Damián Alcázar	b
17	Color	Joss Whedon	Robert Downey Jr.	Chris Hemsworth	The Avengers	Scarlett Johansson	a
18	Color	Rob Marshall	Sam Claflin	Johnny Depp	Pirates of the Caribbean: On Stranger Tides	Stephen Graham	b
19	Color	Barry Sonnenfeld	Michael Stuhlbarg	Will Smith	Men in Black 3	Nicole Scherzinger	a
20	Color	Peter Jackson	Adam Brown	Aidan Turner	The Hobbit: The Battle of the Five Armies	James Nesbitt	a
21	Color	Marc Webb	Andrew Garfield	Emma Stone	The Amazing Spider-Man	Chris Zylka	li
22	Color	Ridley Scott	William Hurt	Mark Addy	Robin Hood	Scott Grimes	1 h
23	Color	Peter Jackson	Adam Brown	Aidan Turner	The Hobbit: The Desolation of Smaug	James Nesbitt	d s
24	Color	Chris Weitz	Eva Green	Christopher Lee	The Golden Compass	Kristin Scott Thomas	d
25	Color	Peter Jackson	Thomas Kretschmann	Naomi Watts	King Kong	Evan Parke	a w
26	Color	James Cameron	Kate Winslet	Leonardo DiCaprio	Titanic	Gloria Stuart	a
27	Color	Anthony Russo	Scarlett Johansson	Robert Downey Jr.	Captain America: Civil War	Chris Evans	b u

	color	director_name	actor_2_name	actor_1_name	movie_title	actor_3_name	p
28	Color	Peter Berg	Alexander Skarsgård	Liam Neeson	Battleship	Tadanobu Asano	b
29	Color	Colin Trevorrow	Judy Greer	Bryce Dallas Howard	Jurassic World	Omar Sy	d w
...	...	...	...	...	...	...	..
5013	Color	Eric Eason	Panchito Gómez	Franky G	Manito	Casper Martinez	e d
5014	Color	Uwe Boll	Katharine Isabelle	Matt Frewer	Rampage	Michael Paré	d
5015	Black and White	Richard Linklater	Richard Linklater	Tommy Pallotta	Slacker	Jean Caffeine	a s
5016	Color	Joseph Mazzella	Mikaal Bates	Tjasa Ferme	Dutch Kills	Damon Owlia	N
5017	Color	Travis Legge	Suzi Lorraine	Kristen Seavey	Dry Spell	Travis Legge	a d
5018	Color	Alex Kendrick	Lisa Arnold	Shannen Fields	Flywheel	Janet Lee Dapper	b fi
5019	Color	Marcus Nispel	Brittany Curran	Ashley Tramonte	Exeter	Lindsay MacDonald	a
5020	NULL	Brandon Landers	Alana Kaniewski	Robbie Barnes	The Ridges	Brandon Landers	a
5021	Color	Jay Duplass	Katie Aselton	Mark Duplass	The Puffy Chair	Bari Hyman	b
5022	Black and White	Jim Chuchu	Olwenya Maina	Paul Ogola	Stories of Our Lives	Mugambi Nthiga	N
5023	Color	Daryl Wein	Heather Burns	Zoe Lister-Jones	Breaking Upwards	Ebon Moss-Bachrach	N
5024	Color	Jason Trost	Jason Trost	Sean Whalen	All Superheroes Must Die	Nick Principe	a d
5025	Color	John Waters	Mink Stole	Divine	Pink Flamingos	Edith Massey	a
5026	Color	Olivier Assayas	Béatrice Dalle	Maggie Cheung	Clean	Don McKellar	ja
5027	Color	Jafar Panahi	Nargess Mamizadeh	Fereshteh Sadre Orafaiy	The Circle	Mojgan Faramarzi	a

	color	director_name	actor_2_name	actor_1_name	movie_title	actor_3_name	p
5028	Black and White	Ivan Kavanagh	Michael Parle	Patrick O'Donnell	Tin Can Man	Emma Eliza Regan	N
5029	Color	Kiyoshi Kurosawa	Anna Nakagawa	Kôji Yakusho	The Cure	Denden	b
5030	Color	Tadeo Garcia	Michael Cortez	Tatiana Suarez-Pico	On the Downlow	Eric Ambriz	g i
5031	Color	Thomas L. Phillips	Joe Coffey	Julianna Pitt	Sanctuary; Quite a Conundrum	John Lucas	n
5032	Color	Ash Baron-Cohen	Stanley B. Herman	Peter Greene	Bang	James Noble	c n
5033	Color	Shane Carruth	David Sullivan	Shane Carruth	Primer	Casey Gooden	c fi
5034	Color	Neill Dela Llana	Edgar Tancangco	Ian Gamazon	Cavite	Quynn Ton	ji g
5035	Color	Robert Rodriguez	Peter Marquardt	Carlos Gallardo	El Mariachi	Consuelo Gómez	a
5036	Color	Anthony Vallone	John Considine	Richard Jewell	The Mongol King	Sara Stepnicka	je
5037	Color	Edward Burns	Caitlin FitzGerald	Kerry Bishé	Newlyweds	Daniella Pineda	w
5038	Color	Scott Smith	Daphne Zuniga	Eric Mabius	Signed Sealed Delivered	Crystal Lowe	fi
5039	Color	NULL	Valorie Curry	Natalie Zea	The Following	Sam Underwood	c
5040	Color	Benjamin Roberds	Maxwell Moody	Eva Boehnke	A Plague So Pleasant	David Chandler	N
5041	Color	Daniel Hsia	Daniel Henney	Alan Ruck	Shanghai Calling	Eliza Coupe	N
5042	Color	Jon Gunn	Brian Herzlinger	John August	My Date with Drew	Jon Gunn	a

5043 rows × 16 columns

In [76]:
```
from pandas import DataFrame,Series
catdata.columns
```

Out[76]:
```
Index(['color', 'director_name', 'actor_2_name', 'actor_1_name', 'movie_titl
e',
       'actor_3_name', 'plot_keywords', 'language', 'country',
       'content_rating', 'Genre_0', 'Genre_1', 'Genre_2', 'Genre_3', 'Genre_
4',
       'Genre_5'],
      dtype='object')
```

In [77]:
```
catdata['director_name'].value_counts().head()
```

Out[77]:
```
NULL                104
Steven Spielberg     26
Woody Allen          22
Martin Scorsese      20
Clint Eastwood       20
Name: director_name, dtype: int64
```

In [78]:
```
#missing data
total = catdata.isnull().sum().sort_values(ascending=False)
percent = (catdata.isnull().sum()/catdata.isnull().count()).sort_values(ascend
ing=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

Out[78]:

	Total	Percent
Genre_5	4946	0.980765
Genre_4	4597	0.911561
Genre_3	3616	0.717034
Genre_2	1988	0.394210
Genre_1	633	0.125521
Genre_0	0	0.000000
content_rating	0	0.000000
country	0	0.000000
language	0	0.000000
plot_keywords	0	0.000000
actor_3_name	0	0.000000
movie_title	0	0.000000
actor_1_name	0	0.000000
actor_2_name	0	0.000000
director_name	0	0.000000
color	0	0.000000

Country vs IMDB Score

In [79]:
```
tmp = data['country'].value_counts()
count_list=tmp[tmp>50].index.tolist()
x=data[data['country'].isin(count_list)]
x
```

Out[79]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	a
0	Color	James Cameron	723.0	178.0	0.0	8
1	Color	Gore Verbinski	302.0	169.0	563.0	1
2	Color	Sam Mendes	602.0	148.0	0.0	1
3	Color	Christopher Nolan	813.0	164.0	22000.0	2
5	Color	Andrew Stanton	462.0	132.0	475.0	5
6	Color	Sam Raimi	392.0	156.0	0.0	4
7	Color	Nathan Greno	324.0	100.0	15.0	2
8	Color	Joss Whedon	635.0	141.0	0.0	1
9	Color	David Yates	375.0	153.0	282.0	1
10	Color	Zack Snyder	673.0	183.0	0.0	2
11	Color	Bryan Singer	434.0	169.0	0.0	9
12	Color	Marc Forster	403.0	106.0	395.0	3
13	Color	Gore Verbinski	313.0	151.0	563.0	1
14	Color	Gore Verbinski	450.0	150.0	563.0	1
15	Color	Zack Snyder	733.0	143.0	0.0	7
16	Color	Andrew Adamson	258.0	150.0	80.0	2
17	Color	Joss Whedon	703.0	173.0	0.0	1
18	Color	Rob Marshall	448.0	136.0	252.0	1
19	Color	Barry Sonnenfeld	451.0	106.0	188.0	7
21	Color	Marc Webb	599.0	153.0	464.0	9
22	Color	Ridley Scott	343.0	156.0	0.0	7
23	Color	Peter Jackson	509.0	186.0	0.0	7
24	Color	Chris Weitz	251.0	113.0	129.0	1
26	Color	James Cameron	315.0	194.0	0.0	7

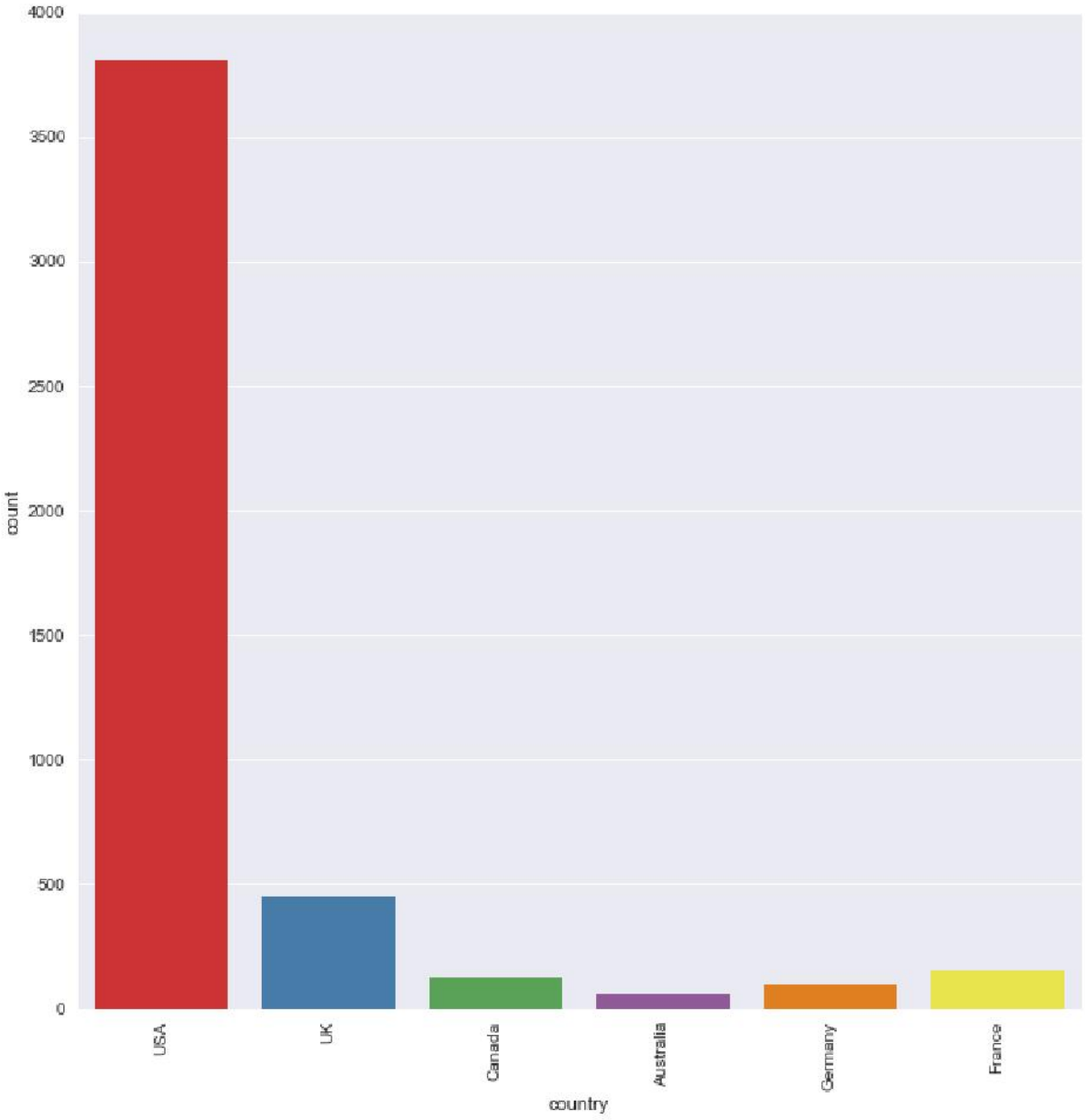
	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	a
27	Color	Anthony Russo	516.0	147.0	94.0	1
28	Color	Peter Berg	377.0	131.0	532.0	6
29	Color	Colin Trevorrow	644.0	124.0	365.0	1
30	Color	Sam Mendes	750.0	143.0	0.0	3
31	Color	Sam Raimi	300.0	135.0	0.0	4
32	Color	Shane Black	608.0	195.0	1000.0	3
...	...	...	...	...	...	..
5008	Black and White	Kevin Smith	136.0	102.0	0.0	2
5009	Color	James Bidgood	8.0	65.0	0.0	N
5010	Color	Andrew Bujalski	43.0	85.0	26.0	3
5011	Color	Neil LaBute	80.0	97.0	119.0	7
5012	Color	David Ayer	233.0	109.0	453.0	1
5013	Color	Eric Eason	28.0	79.0	3.0	4
5014	Color	Uwe Boll	58.0	80.0	892.0	4
5015	Black and White	Richard Linklater	61.0	100.0	0.0	0
5016	Color	Joseph Mazzella	NaN	90.0	0.0	9
5017	Color	Travis Legge	1.0	90.0	138.0	1
5018	Color	Alex Kendrick	5.0	120.0	589.0	4
5019	Color	Marcus Nispel	43.0	91.0	158.0	2
5020	NaN	Brandon Landers	NaN	143.0	8.0	8
5021	Color	Jay Duplass	51.0	85.0	157.0	1
5023	Color	Daryl Wein	22.0	88.0	38.0	2
5024	Color	Jason Trost	42.0	78.0	91.0	8

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	a
5025	Color	John Waters	73.0	108.0	0.0	1
5026	Color	Olivier Assayas	81.0	110.0	107.0	4
5030	Color	Tadeo Garcia	NaN	84.0	5.0	1
5031	Color	Thomas L. Phillips	13.0	82.0	120.0	8
5032	Color	Ash Baron-Cohen	10.0	98.0	3.0	1
5033	Color	Shane Carruth	143.0	77.0	291.0	8
5035	Color	Robert Rodriguez	56.0	81.0	0.0	6
5036	Color	Anthony Vallone	NaN	84.0	2.0	2
5037	Color	Edward Burns	14.0	95.0	0.0	1
5038	Color	Scott Smith	1.0	87.0	2.0	3
5039	Color	NaN	43.0	43.0	NaN	3
5040	Color	Benjamin Roberds	13.0	76.0	0.0	0
5041	Color	Daniel Hsia	14.0	100.0	0.0	4
5042	Color	Jon Gunn	43.0	90.0	16.0	1

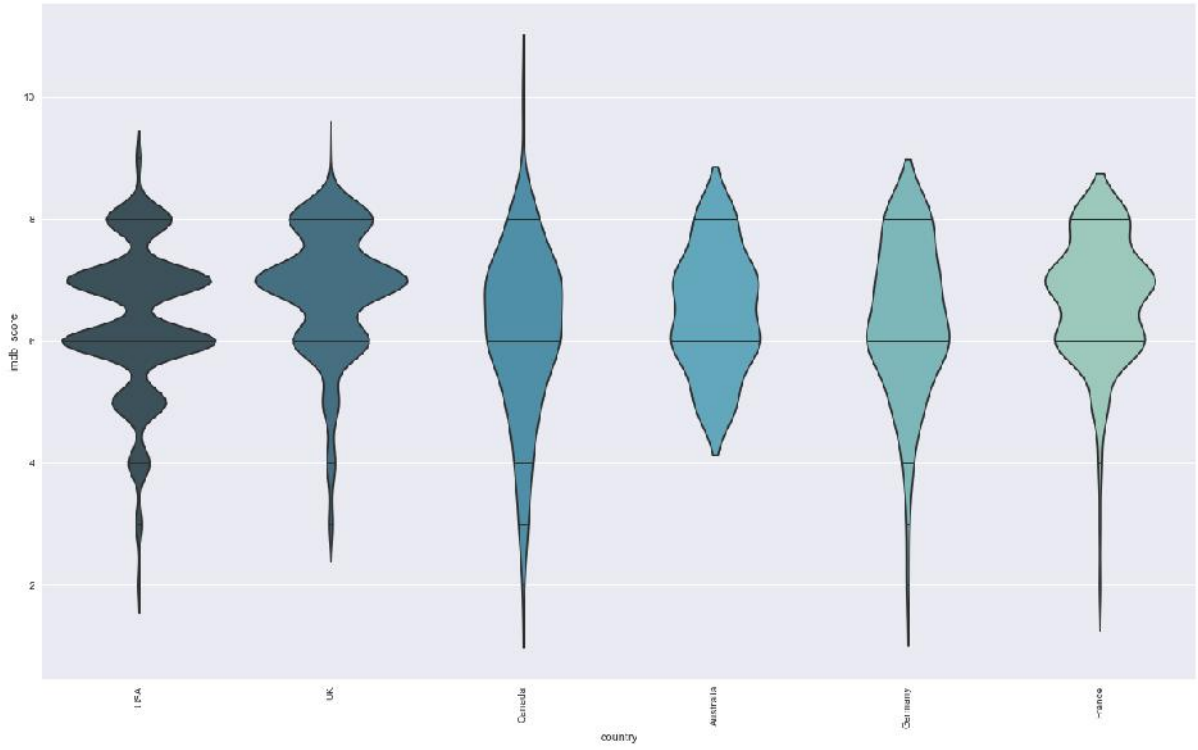
4687 rows × 28 columns

In [82]:
```
plt.figure(figsize = (30, 30))
sns.factorplot('country',kind='count',data=x,size=9,palette='Set1')
xt = plt.xticks(rotation=90)
```

```
<matplotlib.figure.Figure at 0x26247339208>
```

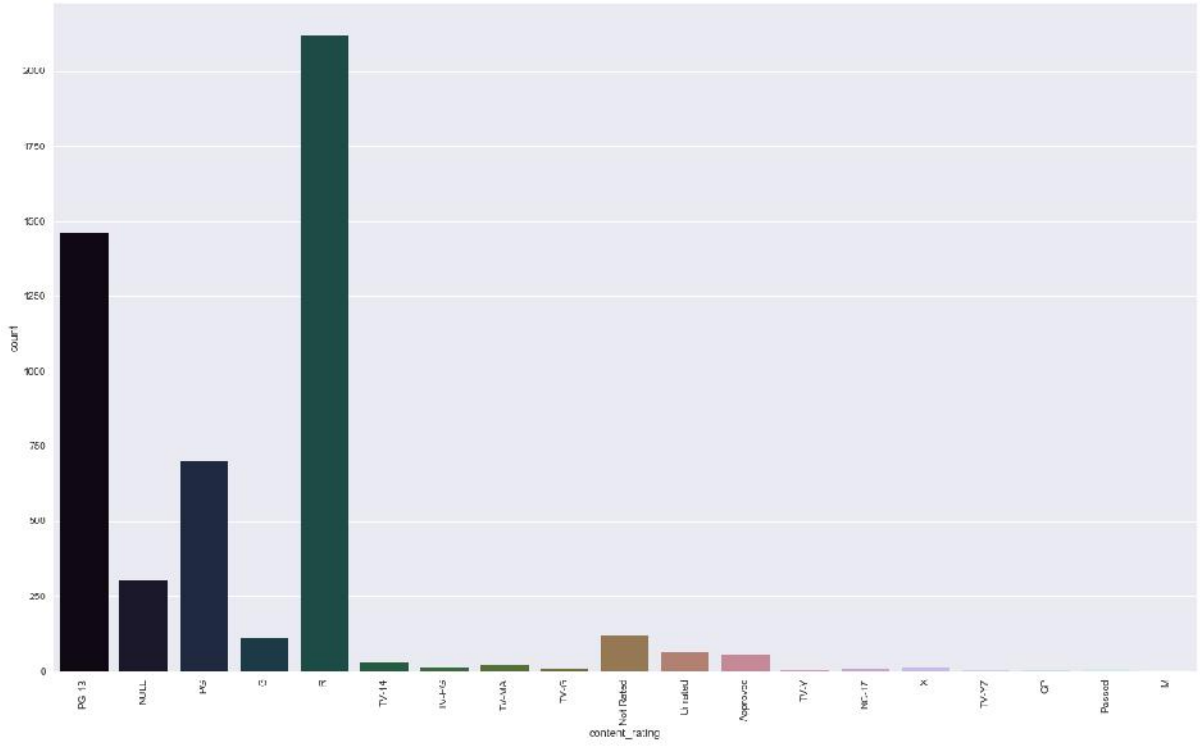


In [87]:
```
plt.figure(figsize = (20, 12))
sns.violinplot(x = 'country',y='imdb_score', data = data[data['country'].isin(
count_list)],inner="stick",palette='GnBu_d')
xt = plt.xticks(rotation=90)
```



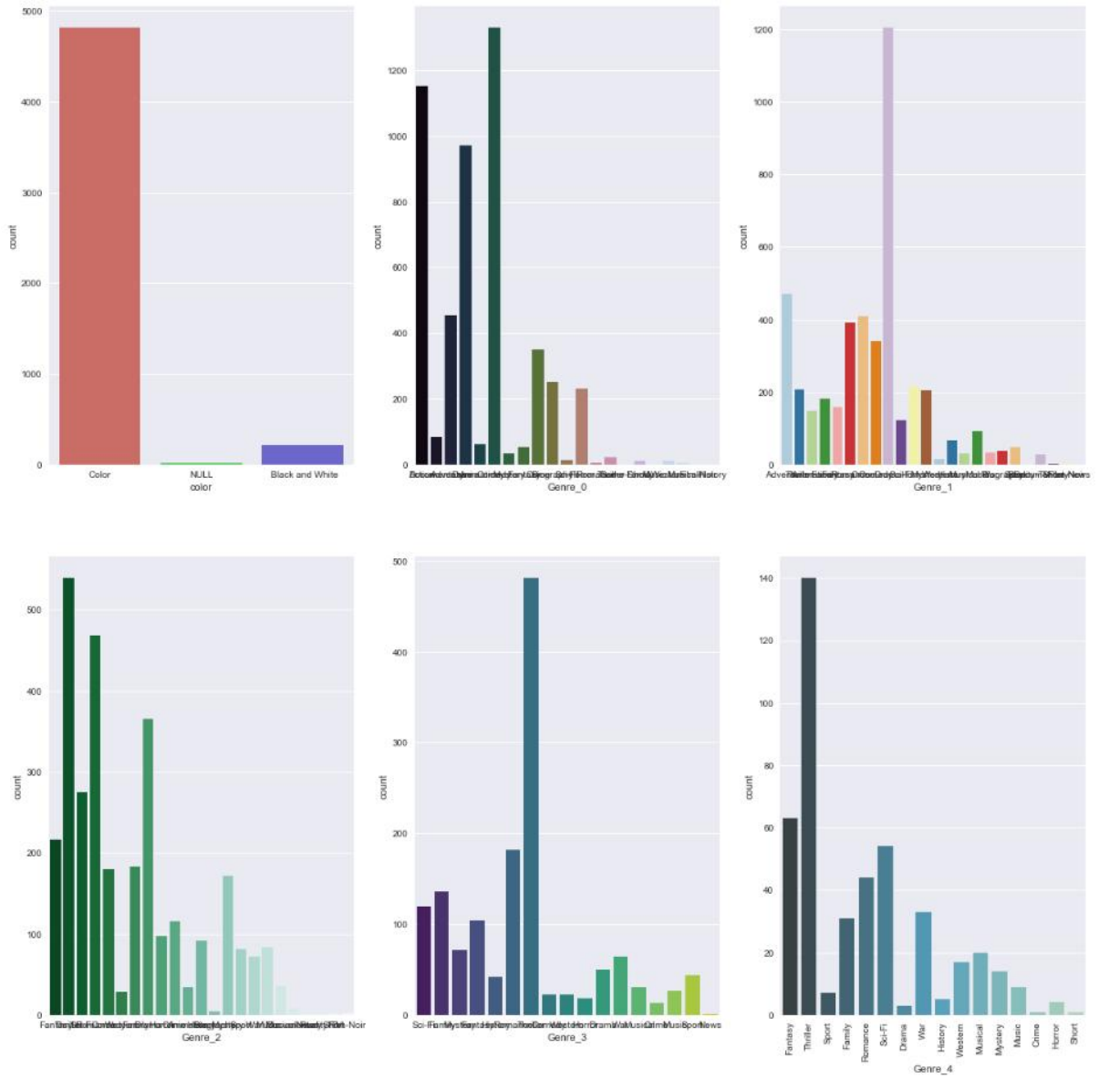
Content Rating Distribution

In [86]:
```
plt.figure(figsize = (20, 12))
sns.countplot(x = catdata['content_rating'], data = catdata,palette='cubehelix')
xt = plt.xticks(rotation=90)
```



Distribution of Categorical Features

In [90]:
```
fig, ax = plt.subplots(figsize=(20,20), ncols=3, nrows=2)
sns.countplot(data = catdata, x= 'color', ax = ax[0][0],palette='hls')
sns.countplot(data = catdata, x= 'Genre_0', ax = ax[0][1],palette='cubehelix')
sns.countplot(data = catdata, x= 'Genre_1', ax = ax[0][2],palette='Paired')
sns.countplot(data = catdata, x= 'Genre_2', ax = ax[1][0],palette='BuGn_r')
sns.countplot(data = catdata, x= 'Genre_3', ax = ax[1][1],palette='viridis')
sns.countplot(data = catdata, x= 'Genre_4', ax = ax[1][2],palette='GnBu_d')
xt=plt.xticks(rotation=90);
```

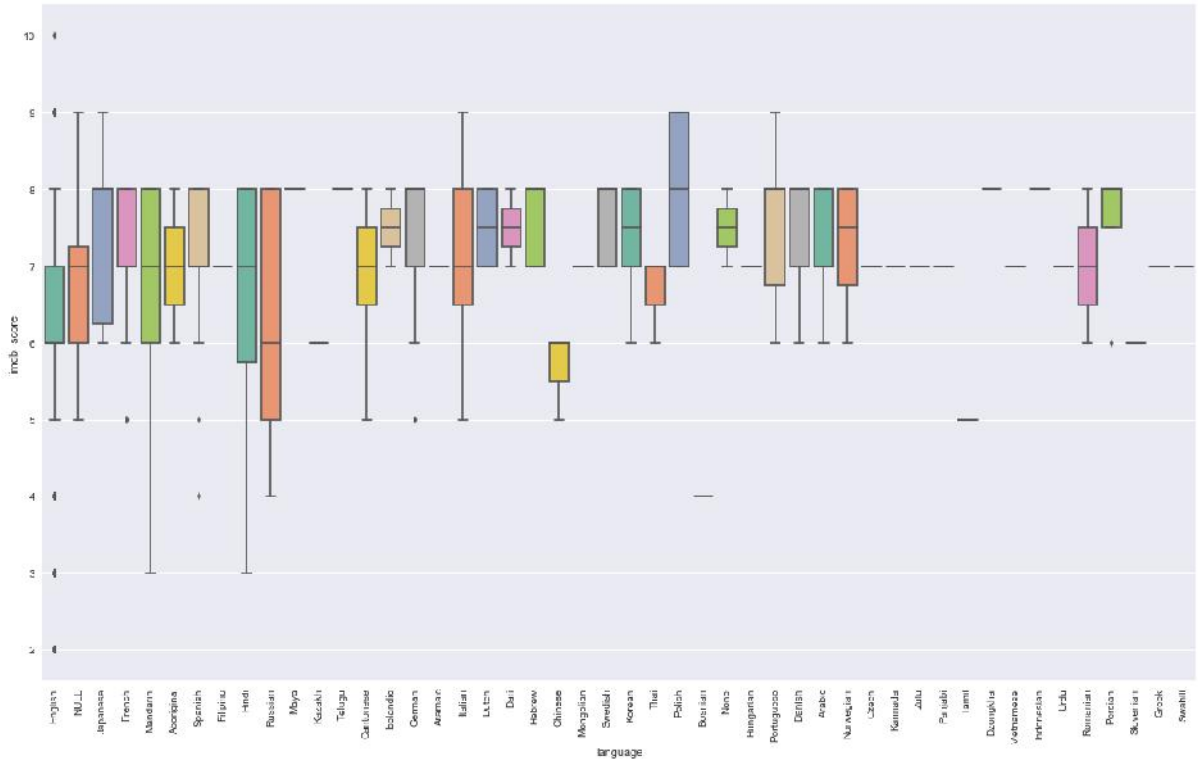


In [134]:
```
catdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 16 columns):
color              5043 non-null object
director_name      5043 non-null object
actor_2_name       5043 non-null object
actor_1_name       5043 non-null object
movie_title        5043 non-null object
actor_3_name       5043 non-null object
plot_keywords      5043 non-null object
language           5043 non-null object
country            5043 non-null object
content_rating     5043 non-null object
Genre_0            5043 non-null object
Genre_1            4410 non-null object
Genre_2            3055 non-null object
Genre_3            1427 non-null object
Genre_4            446 non-null object
Genre_5            97 non-null object
dtypes: object(16)
memory usage: 630.5+ KB
```

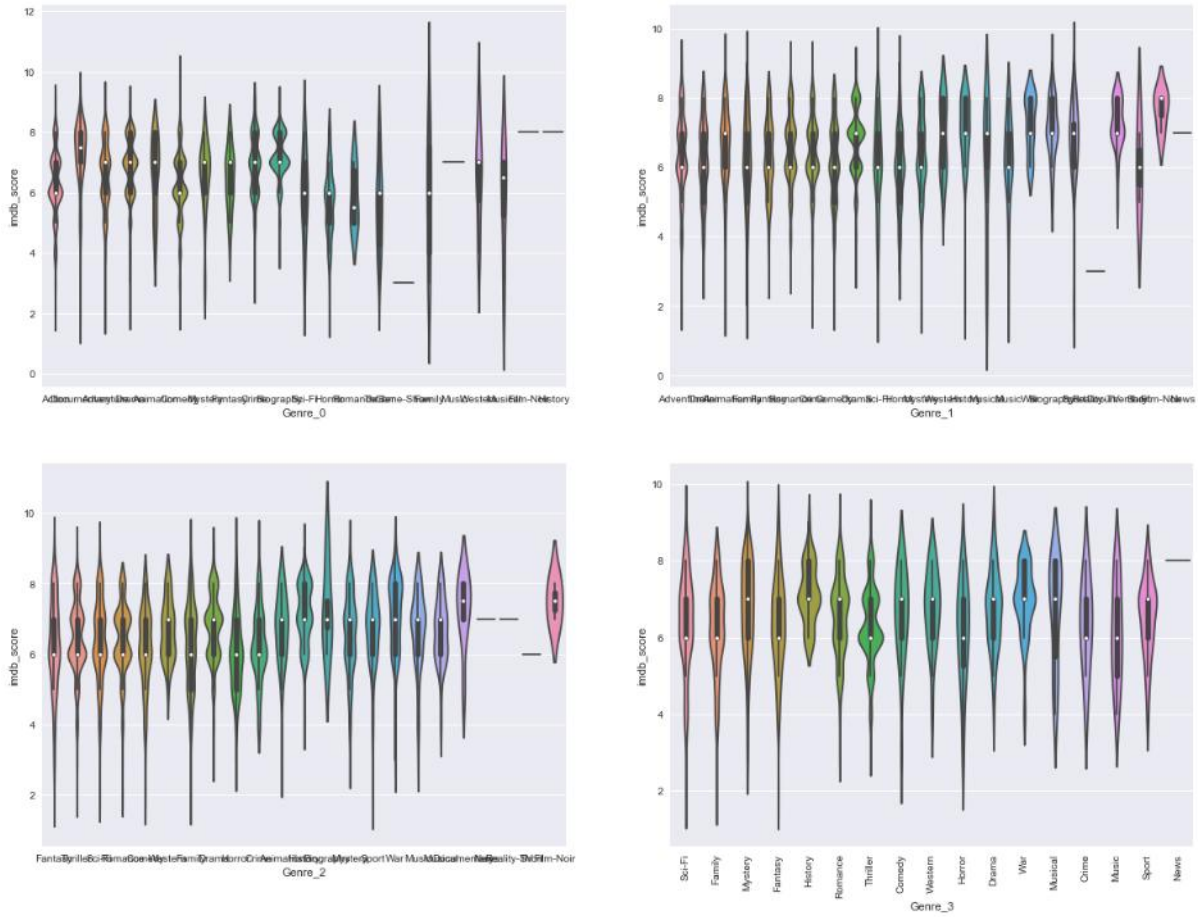
Language Vs IMDB Score

In [93]:
```
plt.figure(figsize = (20, 12))
sns.boxplot(x = catdata['language'], y = data['imdb_score'],palette='Set2')
xt=plt.xticks(rotation=90);
```



Genres Vs IMDB Score

In [136]:
```
fig, ax = plt.subplots(2, 2, figsize = (20, 15))
sns.violinplot(x = catdata['Genre_0'], y = data['imdb_score'],  ax = ax[0][0])
xt=plt.xticks(rotation=90);
sns.violinplot(x = catdata['Genre_1'], y = data['imdb_score'], ax = ax[0][1])
xt=plt.xticks(rotation=90);
sns.violinplot(x = catdata['Genre_2'], y = data['imdb_score'],  ax = ax[1][0])
xt=plt.xticks(rotation=90);
sns.violinplot(x = catdata['Genre_3'], y = data['imdb_score'], ax = ax[1][1])
xt=plt.xticks(rotation=90);
```



In [137]:
```
catdata = pd.get_dummies(catdata,columns=['color','language','country','conten
t_rating','Genre_0','Genre_1','Genre_2','Genre_3',
'Genre_4'])
```

In [138]:
```
(catdata.columns)
```

Out[138]:
```
Index(['director_name', 'actor_2_name', 'actor_1_name', 'movie_title',
       'actor_3_name', 'plot_keywords', 'Genre_5', 'color_ Black and White',
       'color_Color', 'color_NULL',
       ...
       'Genre_4_Music', 'Genre_4_Musical', 'Genre_4_Mystery',
       'Genre_4_Romance', 'Genre_4_Sci-Fi', 'Genre_4_Short', 'Genre_4_Sport',
       'Genre_4_Thriller', 'Genre_4_War', 'Genre_4_Western'],
      dtype='object', length=244)
```

In [139]:
```
catdata=catdata.drop(['director_name', 'actor_2_name', 'actor_1_name', 'movie_
title','actor_3_name', 'plot_keywords', 'Genre_5'],axis=1)
```

Concatenating Numerical and Categorical data

In [140]:
```
merged = pd.concat([numdata,catdata], axis =1)
merged.head()
```

Out[140]:

	num_critic_for_reviews	duration	director_facebook_likes	gross	num_voted_u
0	723.0	178.0	0.0	760505847.0	886204
1	302.0	169.0	563.0	309404152.0	471220
2	602.0	148.0	0.0	200074175.0	275868
3	813.0	164.0	22000.0	448130642.0	1144337
4	110.0	103.0	131.0	25517500.0	8

5 rows × 245 columns

In [228]:
```
merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 7 columns):
num_critic_for_reviews     5043 non-null float64
duration                   5043 non-null float64
director_facebook_likes    5043 non-null float64
movie_facebook_likes       5043 non-null int64
title_year                 5043 non-null float64
num_voted_users            5043 non-null int64
gross                      5043 non-null float64
dtypes: float64(5), int64(2)
memory usage: 275.9 KB
```

Splitting Dataset into Train & Test Set

In [ ]:
```
from sklearn.model_selection import train_test_split # to split the data into
 two parts
X_train,X_test,y_train,y_test = train_test_split(merged,y, random_state = 0,te
st_size = 0.15)
```

Feature Scaling

In [230]:
```
from sklearn.preprocessing import StandardScaler
sc_x=StandardScaler()
X_train=sc_x.fit_transform(X_train)
X_test=sc_x.transform(X_test)
```

Basic Regression Model

In [387]:
```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
```

Out[387]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [388]:
```
y_pred=regressor.predict(X_test)
predictions=list(y_pred)
predictions
```

```
Out[388]: [6.5083334816359351,
           6.9185696989955998,
           6.0523157815606066,
           6.2147726701801158,
           6.2676074196570362,
           8.7132109888899123,
           6.2340086997478368,
           5.7514093817125911,
           6.1940516092167899,
           6.4914469167490925,
           6.6513565595054969,
           7.273215922988177,
           6.3203564036997291,
           6.5682506786803794,
           6.2862282288371052,
           6.3602381669764139,
           6.2744319730252229,
           6.3558484144408416,
           6.5233918562632791,
           6.1001929370878054,
           6.2112813847495012,
           5.8641915720372566,
           6.3748415981163733,
           6.7474057551437223,
           7.6120989239099792,
           6.7638955795412672,
           6.1532877366477674,
           6.2170618659775387,
           7.3000064397222078,
           7.06430731762861,
           5.913149687742143,
           6.7060114570397902,
           5.9877055654240694,
           6.1868143380341571,
           7.3715624255243277,
           7.2257444972059632,
           7.4261993031194118,
           6.951653000195896,
           6.1633824889353646,
           5.9456277756909532,
           7.0608733894926221,
           6.8419942153048217,
           6.4677047458238519,
           6.1625193191379966,
           5.8944236268864154,
           6.4287110593700785,
           6.610945611487848,
           5.9441456940366448,
           6.3425422844754085,
           6.3856589938954498,
           6.2815687439533221,
           6.1666539947412504,
           7.3890477304312245,
           6.1671786942320335,
           7.4877070805383559,
           6.2372181582368773,
           6.3210975171355148,
```

6.8113843717396412,
6.2647807967233158,
6.1136572183486555,
6.2893898984406356,
6.0375502215237908,
6.1953535536164948,
6.3494994615176159,
6.1832470543024778,
7.2882106963109541,
6.5850997184703051,
6.1340444770683185,
10.801920429673396,
6.1708154801353539,
5.8091362964306912,
6.317232079587642,
5.675432243308995,
6.4286842634705179,
6.2409665896927695,
8.1139883437754143,
5.697786100838254,
6.8019508270999829,
7.0267262398473829,
6.3668470070596914,
5.978498886660212,
5.9070186821036366,
6.8458965442181476,
6.5374494336787734,
6.2579364815511935,
6.4015179378918372,
7.5820386809533913,
6.4611904909978071,
6.7071316312254474,
7.1815702119309828,
6.108935323905424,
6.1989358543471909,
8.1771509204880104,
6.09194090168263,
6.3638138868170779,
6.6695258351631965,
6.2124741718164351,
6.5852660798238816,
6.4958405641004484,
6.3348749674647138,
5.7258015978144901,
6.5049883125528991,
6.0269023966799056,
6.5448374684817274,
6.3602381669764139,
5.747447645328938,
6.1965794113615109,
8.9268386729791072,
6.5261900886598649,
6.6409361305505001,
6.2920206077361085,
5.886499339189629,
6.5626783364586947,
6.4300255581442167,

8.1057488358389129,
5.884900714833031,
8.4206208965295062,
6.6527585455257601,
7.6979311092525977,
6.5954467898694658,
6.5143365545079455,
6.2013407840833699,
7.0619870403208225,
6.9466974221548057,
6.2292918624706637,
6.4602714623041928,
6.2061770289225784,
6.2020862676117536,
6.3958077417244743,
5.7676734215413621,
5.7880366207684748,
6.1920742392732659,
5.9880897269250264,
5.776968325966731,
6.2639839943778037,
6.0911689120478201,
6.4574944900851854,
6.2092432743274086,
6.2951320612975801,
6.5238097862036124,
6.2454896691521933,
6.2850742173621317,
5.8154338924190041,
5.6411050582557838,
6.3009076112661946,
6.1709083777571108,
5.8256707905168179,
6.1937071560994976,
6.4015329985763199,
5.7037058947895591,
5.682587275036151,
7.1565129275586363,
6.7005208338332167,
6.2810868616809525,
6.1684473850943631,
6.0794538137411154,
7.4541975766307331,
6.164506419481671,
6.3005750119713859,
6.4542454835264715,
6.4226923166312435,
6.2551435073631731,
6.0568577851820704,
5.6872881115393739,
7.8379569928172028,
6.3323033712304886,
6.363710517221203,
6.6787017598897434,
6.4601402628058358,
6.3577176771687007,
5.7904514489998258,

5.958653427322016,
6.8821303004838397,
5.6870707284176705,
6.9041408689031059,
6.7755658396107865,
6.4814006998699591,
6.518530712390783,
6.6900103115496936,
7.9119046013802201,
6.1360679935200748,
6.1230474879508332,
6.0868417112433439,
6.8991007757916272,
6.116772734243785,
7.2338293123079369,
6.5095235490448351,
5.9304873092360024,
6.7616308974821573,
6.3365675025668091,
6.4979667371821339,
6.6609813545900201,
7.0502430867259598,
5.8283624582416778,
6.5587323077806925,
5.8099853420017666,
6.054762517568391,
6.0974145463313318,
6.2560323195788463,
6.0395747245757736,
5.9224388278447151,
6.8230494723768746,
6.2130636742139602,
6.0363829474753565,
5.8707601583259628,
6.0277491243345924,
5.9247922403712323,
6.3027621853072295,
6.6580137363201777,
6.7115959666084066,
6.7069026169108028,
6.227049910733367,
6.2451245987608877,
6.2988550484068728,
5.8919740973485943,
6.6008791092295693,
6.7972122376462654,
6.0090737080258458,
6.3318990320896198,
6.774450230493569,
6.7030010036566674,
6.4086746760947477,
6.4825323451273933,
7.1113049875365082,
5.7413809755303333,
7.0424473930110185,
6.6375108836282681,
6.7598683416502796,

7.8774691438993676,
6.367379630896874,
8.0258090795000943,
6.7959968061495637,
6.0149083198561097,
6.3189017457463192,
7.0330657422851512,
6.0920905725540582,
6.0563911485217341,
6.551742591261366,
7.2848085068559643,
7.1766800118970409,
6.7761656349399964,
6.0562166804824953,
7.4519761226608408,
6.2633644078316459,
5.9029549032908459,
6.8371227663037564,
7.8503936181094476,
7.4747926064799968,
6.2623625009228538,
6.211812667530185,
5.8215677843836202,
6.4931470694859339,
5.8321213089474666,
6.9221403240150785,
6.7084088393617094,
6.2842980009217086,
5.8944595080357525,
5.6025091885512932,
6.6078015602593574,
6.5032613079109876,
5.8665652206613474,
6.3113546179636701,
6.3878198064986149,
7.9869020612510875,
6.7731866577993287,
6.5816320842256779,
5.7675930452026325,
6.1331939785937504,
5.8225711290744497,
6.0094532627861481,
6.3461587733923102,
7.2009693067095739,
7.1101301226809488,
6.9486119023023427,
7.6392089797935379,
6.1986367740221802,
6.3263738983711626,
5.8105230261449679,
6.476241920586129,
6.3373905865428268,
6.0645492086993436,
6.4588485105067797,
6.1072892218799613,
6.4227598150115188,
7.1399597662281664,

7.1095543196084154,
6.7471364557704261,
6.4105745059083015,
6.6998077046184239,
6.1382168006138311,
6.3916254885484864,
6.2929598765328318,
5.6286633907738599,
6.7120286833826741,
6.2363174936231447,
6.067730287672501,
5.8940293729109792,
6.140512972786178,
6.6074099977050826,
6.2732915793369486,
5.5707138885068872,
6.5742852687809359,
6.2889182146875457,
6.9135358905832014,
6.3746643734043502,
6.2062323647982698,
5.8748278030347141,
6.5005232150045202,
5.719882342203606,
6.0647169434056103,
7.1427484511161268,
9.6569145060458368,
6.1709881187559867,
7.2143980632593836,
8.7170985670398817,
6.0840835348479532,
7.2368459923216797,
8.7944175822386743,
6.312800873910791,
7.1316378450772291,
6.3099430822360345,
6.2378669438593697,
6.3687274559553479,
6.0058178395911188,
6.9471344531778607,
6.4565260985732325,
5.7623465841834225,
6.2561612734546177,
6.5571006680054031,
6.1233100798774478,
6.1825194810123385,
6.2482379305389681,
6.1169313303754267,
5.7208682504351893,
5.8630242761052109,
6.0808375033381541,
6.5916136919989068,
6.9470820135502365,
6.3994096076344276,
6.5305202878772635,
6.183443423245599,
6.0511022539133279,

6.4650159100814468,
6.1556628625357357,
6.5799589591372554,
5.8324155430265634,
6.1523191888777733,
5.9393603362731531,
5.8161141079794731,
6.8565381148185587,
7.2748309082062157,
6.2395464948779651,
6.3837370570921896,
7.9217610653271944,
6.1796358591533904,
5.8112235516622368,
5.6715601440450962,
6.8296351341185488,
7.5758837840793412,
6.2414593425581142,
7.5963209564798682,
7.0768035474720747,
6.1713682019880221,
5.6706896965602986,
6.2550420584699813,
6.2078303840437163,
6.7239767837858748,
6.1450044258716723,
6.4217517023511652,
5.6461116573494383,
6.3619040974190959,
5.6095931750312644,
6.2506508787052919,
6.2449800588304667,
5.7473726343521321,
6.0200043328823343,
6.5500591224420051,
6.2575214851715728,
5.87494218360189,
6.2673921233757861,
6.5266832189188424,
6.8978075459097434,
6.0997767815458488,
6.3606009837895492,
6.1599781206333919,
6.4053851128888919,
7.1428624280900621,
6.6062229271700517,
6.3797060147028635,
6.0742769681788014,
6.6057160635404166,
6.1882045405639072,
7.0240774196058524,
6.8228073383777632,
6.1848075519647923,
6.9201199904453903,
7.0873162687880633,
6.0059340738680422,
6.8580688486504435,

6.3206691725700921,
6.3793673790168786,
6.1275249806360552,
5.5353801784994481,
6.9453592994026376,
6.1095943086558346,
7.2423240150157699,
6.7928120961885821,
5.5993555791023377,
7.2583314339678351,
6.794459638242218,
7.6699522730997174,
6.5110029128574665,
6.1583421346547187,
6.1451558998707849,
5.6344587961435133,
7.0427966571411762,
6.3061741539141396,
6.3074504542796035,
6.3767243039612644,
6.2790965869131741,
6.5186129265969193,
6.1057954552756115,
6.2519842437315702,
6.6338299630958266,
6.0432238513213585,
6.2349368512025451,
6.313708698239628,
6.5028866758677975,
5.9983565450147864,
5.77957863632437,
6.627367099830682,
6.3531749150987356,
6.7626823961968547,
6.6141608273748806,
6.178819711158293,
6.0849839436855593,
6.1254909561629516,
6.1506957071312511,
5.6060573426999376,
5.9725346830920412,
7.1855165470323161,
5.9048562570200671,
6.5124338821633048,
6.3968210477617324,
6.0095395520718142,
6.6097984395243126,
6.5822576669628665,
6.875232134162836,
6.0861014790100354,
6.1718378019162774,
5.7005756139191641,
6.5190483693781029,
6.1670402018579287,
6.4364144810391615,
6.2146704752012027,
6.9086752744929045,

6.417160640600982,
7.1344805957310005,
8.4304754034355263,
5.9614536797337045,
5.9951076367838612,
6.3264449751033576,
5.9477807929689668,
6.3767246733681899,
8.5502067872705751,
6.2187828684817879,
6.7668679742436542,
6.0374646822634581,
6.1467135743430523,
6.2444432226445246,
6.2420965549170786,
6.6039878543185493,
5.7276793804179142,
6.0642730776700642,
6.6492289498546429,
6.1878189108595905,
6.1413850677928403,
6.7270589313969014,
5.8571964927804867,
6.1696101398615406,
6.3545404905323641,
6.1958540664560671,
6.2739055349223509,
6.3510604269982291,
6.4758708307578159,
7.842998330088844,
6.5777974484881279,
6.4104602799488539,
6.5940065652462909,
6.6971570608053907,
6.1539690762541293,
6.194831656620627,
6.8944533840086271,
7.784558451909847,
6.5143033851671204,
5.7815426952864089,
5.9915330156818909,
6.2318930919081055,
6.1420155624177353,
6.5456606682708163,
6.5869185096025209,
6.2173603233104728,
6.2024543917208312,
6.4279254421574032,
5.6505507161750286,
6.228898891467133,
6.5099211131028865,
6.8250963710248493,
6.0282190793806913,
6.083208138345773,
7.5143426484507163,
6.6173372017720471,
7.8409124103374577,

6.4798267455202323,
6.3113390935934355,
6.3867939811879486,
6.6616041245027935,
6.185054437817965,
6.282717361211926,
6.1269442426921712,
6.6509470422998271,
5.8401098164643397,
6.3235980740962505,
7.1901024039634969,
6.2124950267434595,
6.5474620380730748,
6.8615525028412687,
6.3012553264174391,
6.1116594690325767,
6.1472946057079367,
6.1936433218248981,
6.1593399086905505,
5.8087824466558295,
6.1896549728081718,
7.2200207729974002,
6.1733919945865461,
6.0801366571907858,
6.8865752661942947,
5.9580457394449837,
7.5780425236382429,
6.4786400894784819,
6.5218578000622571,
6.1844701097216461,
6.264273393486067,
6.4112054192353032,
5.8845885382788472,
6.1824510027152249,
7.0248895329041376,
6.3679901173750073,
6.1201726414955449,
6.4274122174917556,
6.2229411517571869,
6.3277204528201931,
5.8720510521427318,
7.6912124052257553,
5.7265775375002077,
6.035103045935144,
6.4588450555690571,
6.3838362021558854,
6.5963370726951265,
5.8720193178343489,
5.9384781673169416,
6.4874511003748063,
6.0922939102209126,
6.2478589151912542,
5.859755510283545,
6.7934107596531064,
6.370380611521961,
5.9103539227362045,
7.4860313864289054,

6.876610980527011,
6.3842215958621447,
6.3514051118656543,
7.8747633585847385,
6.4445708028459379,
6.1454829292405018,
6.5801392283134827,
5.8967094864441076,
5.8498643386691302,
6.8387665075293773,
5.9769096539081881,
7.3423767347655433,
6.4576661635401083,
6.9584781793797195,
6.3229217774972657,
6.3443671145372642,
6.1235740168132899,
6.9812833513426371,
6.2133979049722923,
6.1387397737392257,
5.7442051494288311,
5.7496283677793789,
5.75972994863168,
5.8483172666325736,
6.6662687140862324,
5.6929970229749278,
6.7534982033690412,
6.3512167685490315,
6.7428930207568403,
6.18838426048148,
6.5092537434762363,
5.9937992849584347,
6.1762017749670015,
5.6797529250840419,
6.4958724507339332,
6.4447701254100966,
6.8588531817653511,
6.9355494083679332,
5.5840346569653905,
6.7163581821119296,
6.2511578340997707,
6.3776273003996522,
6.6699360822556084,
6.4485895475944242,
8.083488785311868,
6.1868502837080355,
6.2354151439246737,
6.5543509495746708,
6.0669649800657508,
6.2097238947064213,
6.6950820156001711,
6.3496325509026379,
6.266798539541095,
6.4106577966359986,
6.1744660042071748,
6.6410093411014026,
7.7167600655957393,

6.7247826033404143,
6.9340367478460534,
6.191999020097998,
5.9458120974889237,
5.8018655236180301,
7.0379953716595693,
6.3041882147295079,
6.4498343868403021,
6.5557418784696226,
5.9876817145474304,
6.6809652364578778,
5.7314126193901895,
7.0993274541337357,
5.7009119688829353,
7.2757575393906837,
6.5856379372246225,
7.0654296981201448,
6.3409365057590383,
6.4188391149466844,
6.3577163890554047,
5.6352318528405085,
6.4360229039722148,
6.1095622620329566,
6.2908420053196643,
7.3024244001615166,
6.2718405973864515,
5.9241297082123197,
7.4530188366831842,
6.0702064090053947,
6.503266279196624,
6.4756750934450551,
6.1903216323699866,
6.3495312126678733,
6.4479276772238849,
6.2999629970766993,
6.1962336881238818,
6.8299039528001391,
7.5475944963778359,
6.2323962240649866,
5.6888909312836482,
6.3942496334011416,
6.4404750356036198,
5.8434008807720597,
5.8606471728705465,
5.9712267865097699,
5.9405871243221515,
6.0716711048440164,
6.6180546598997863,
7.6159658090412918,
6.2085640995056712,
6.5566604996553952,
6.9459743964278839,
5.7006173198050041,
6.0657786326310736,
6.7725629332043917,
6.4724574367212231,
6.5012513419905646,

6.7759339159572232,
6.1035262808917174,
7.9081750492753047,
6.1964230410975834,
5.6868171712073465,
6.8205186781727631,
6.1444059866619716,
6.0939187797371677,
5.9742035841830745,
5.9830694868495229,
6.5630451545940343,
6.1659545318228863,
6.4454634914892743,
7.1402592405627416,
7.1240759380679783,
6.8570557554710136,
6.2712135246668241,
6.0739128287177779,
6.2430490211150369,
6.0939720473220476,
6.6481078651265619,
7.3479344974823411,
7.2165774157964817,
6.066966150903526,
8.2648063606597866,
6.3116346632468172,
5.8124409152269108,
6.4239282247074803,
6.6102568765444794,
6.0185833877074524,
6.840548575938719,
6.1188999735763181,
6.1972801078318005,
6.3014950875629241,
6.0595130389840577,
5.7148210802704789,
6.0363216938411872,
6.4248249774555397,
6.1578403530301511,
6.0630288928439571,
6.189290388508625,
5.8024315324878293,
6.5132696343691743,
5.884038569231886,
6.4762475990978485,
6.4103253495711625,
6.3560226088625802,
5.6886940554451346,
6.7404075089414777,
6.191721457656377,
5.782241785080692,
5.8239953302953209,
6.2187108618648246,
5.7348218506928728,
7.3856399136563935,
7.0521397881659,
6.6926562950633741,

```
6.5997169261366944,
7.3866127682659979,
6.2300355197427093,
6.3881482540014192,
6.0162332856391485,
6.2965768432475695,
7.2523703296750961,
6.1423557071477131,
6.4869131348943405,
6.3443432693856376,
6.4063777912948883,
6.5537803884583443,
6.2343278940552667,
6.1525150137742308,
6.5834467431401009,
6.3991721261679269]
```

Evaluation

In [389]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,predictions)
```

Out[389]: 0.98934304878858437

Model in Tensorflow & Keras

In [390]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn import metrics
from scipy.stats import zscore
import tensorflow as tf
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.callbacks import EarlyStopping
```

In [393]:
```
merged=merged[['num_critic_for_reviews','duration','director_facebook_likes',
'movie_facebook_likes','title_year','num_voted_users','gross']]
merged
```

Out[393]:

	num_critic_for_reviews	duration	director_facebook_likes	movie_facebook_likes
0	723.0	178.0	0.0	33000
1	302.0	169.0	563.0	0
2	602.0	148.0	0.0	85000
3	813.0	164.0	22000.0	164000
4	110.0	103.0	131.0	0
5	462.0	132.0	475.0	24000
6	392.0	156.0	0.0	0
7	324.0	100.0	15.0	29000
8	635.0	141.0	0.0	118000
9	375.0	153.0	282.0	10000
10	673.0	183.0	0.0	197000
11	434.0	169.0	0.0	0
12	403.0	106.0	395.0	0
13	313.0	151.0	563.0	5000
14	450.0	150.0	563.0	48000
15	733.0	143.0	0.0	118000
16	258.0	150.0	80.0	0
17	703.0	173.0	0.0	123000
18	448.0	136.0	252.0	58000
19	451.0	106.0	188.0	40000
20	422.0	164.0	0.0	65000
21	599.0	153.0	464.0	56000
22	343.0	156.0	0.0	17000
23	509.0	186.0	0.0	83000
24	251.0	113.0	129.0	0
25	446.0	201.0	0.0	0
26	315.0	194.0	0.0	26000
27	516.0	147.0	94.0	72000
28	377.0	131.0	532.0	44000
29	644.0	124.0	365.0	150000
...	...	...	...	...
5013	28.0	79.0	3.0	61

	num_critic_for_reviews	duration	director_facebook_likes	movie_facebook_likes
5014	58.0	80.0	892.0	0
5015	61.0	100.0	0.0	2000
5016	110.0	90.0	0.0	33
5017	1.0	90.0	138.0	200
5018	5.0	120.0	589.0	725
5019	43.0	91.0	158.0	0
5020	110.0	143.0	8.0	33
5021	51.0	85.0	157.0	297
5022	6.0	60.0	0.0	45
5023	22.0	88.0	38.0	324
5024	42.0	78.0	91.0	835
5025	73.0	108.0	0.0	0
5026	81.0	110.0	107.0	171
5027	64.0	90.0	397.0	697
5028	12.0	83.0	18.0	105
5029	78.0	111.0	62.0	817
5030	110.0	84.0	5.0	22
5031	13.0	82.0	120.0	424
5032	10.0	98.0	3.0	20
5033	143.0	77.0	291.0	19000
5034	35.0	80.0	0.0	74
5035	56.0	81.0	0.0	0
5036	110.0	84.0	2.0	4
5037	14.0	95.0	0.0	413
5038	1.0	87.0	2.0	84
5039	43.0	43.0	49.0	32000
5040	13.0	76.0	0.0	16
5041	14.0	100.0	0.0	660
5042	43.0	90.0	16.0	456

5043 rows × 7 columns

To split the data into two parts

In [394]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(merged,y, random_state = 0,te
st_size = 0.15)
```

Scale the Data by MinMaxScaler

In [396]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(x_train)
```

Out[396]:
```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

In [397]:
```
x_train=pd.DataFrame(data=scaler.transform(x_train),columns=x_train.columns,in
dex=x_train.index)
```

In [398]:
```
x_test=pd.DataFrame(data=scaler.transform(x_test),columns=x_test.columns,index
=x_test.index)
```

In [244]:
```
num_critic_for_reviews=tf.feature_column.numeric_column('num_critic_for_review
s')
duration=tf.feature_column.numeric_column('duration')
director_facebook_likes=tf.feature_column.numeric_column('director_facebook_li
kes')
movie_facebook_likes=tf.feature_column.numeric_column('movie_facebook_likes')
title_year=tf.feature_column.numeric_column('title_year')
num_voted_users=tf.feature_column.numeric_column('num_voted_users')
gross=tf.feature_column.numeric_column('gross')
```

In [245]:
```
features=[num_critic_for_reviews,duration,director_facebook_likes,movie_facebo
ok_likes,title_year,num_voted_users,gross]
```

In [399]:
```
x_train.shape
```

Out[399]:
```
(4286, 7)
```

In [402]:
```
merged = merged.dropna(how='any')
merged.shape
```

Out[402]:
```
(5043, 7)
```

In [403]:
```
X = merged.iloc[:, 0:6].values
```

Model For Prediction

In [404]:
```
def baseline_model(learn_rate=0.01, momentum=0):
    # create model
    model = Sequential()
    model.add(Dense(6, input_dim=6, kernel_initializer='normal', activation='r
elu'))
    model.add(Dense(5, input_dim=6, kernel_initializer='normal', activation='r
elu'))
    model.add(Dense(4, input_dim=6, kernel_initializer='normal', activation='r
elu'))
    model.add(Dense(2, input_dim=6, kernel_initializer='normal', activation='r
elu'))
    model.add(Dense(1, kernel_initializer='normal', activation='relu'))
    # compile model
    optimizer = SGD(lr=learn_rate, momentum=momentum)
    model.compile(loss='mean_absolute_error', optimizer='adam')
    return model
```

In [405]:
```
from keras.wrappers.scikit_learn import KerasClassifier
```

In [406]:
```
np.random.seed(42)
#evaluate model with standardized dataset
estimator = KerasRegressor(build_fn=baseline_model, nb_epoch=10, batch_size=50
, verbose=1)
#using 10-fold cross validation to evaluate the model
kfold = KFold(n_splits=10, random_state=42)
results = cross_val_score(estimator, X, y, cv=kfold)
print("Results: %.2f (%.2f) MAE" % (results.mean(), results.std()))
```

```
Epoch 1/10
4538/4538 [==============================] - 2s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 2/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 4/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 5/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 6/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 7/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 8/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 10/10
4538/4538 [==============================] - 0s - loss: 6.4167
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/505 [=>............................] - ETA: 8sEpoch 1/10
4538/4538 [==============================] - 1s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 2/10
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 4/10
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 5/10
```

```
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 6/10
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 7/10
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 8/10
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 10/10
4538/4538 [==============================] - 0s - loss: 6.4480
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/505 [=>............................] - ETA: 8sEpoch 1/10
4538/4538 [==============================] - 1s - loss: 5.5222
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 2/10
4538/4538 [==============================] - 0s - loss: 3.1568
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4538/4538 [==============================] - 0s - loss: 1.0215
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 4/10
4538/4538 [==============================] - 0s - loss: 0.9382
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 5/10
4538/4538 [==============================] - 0s - loss: 0.9178
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 6/10
4538/4538 [==============================] - 0s - loss: 0.9399
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 7/10
4538/4538 [==============================] - 0s - loss: 0.8952
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 8/10
4538/4538 [==============================] - 0s - loss: 0.8918
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4538/4538 [==============================] - 0s - loss: 0.8988
```

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 10/10
4538/4538 [==============================] - 0s - loss: 0.9161
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/505 [=>............................] - ETA: 8sEpoch 1/10
4539/4539 [==============================] - 1s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 2/10
4539/4539 [==============================] - 0s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4539/4539 [==============================] - 0s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 4/10
4539/4539 [==============================] - 0s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 5/10
4539/4539 [==============================] - 0s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 6/10
4539/4539 [==============================] - 0s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 7/10
4539/4539 [==============================] - 0s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 8/10
4539/4539 [==============================] - 0s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4539/4539 [==============================] - ETA: 0s - loss: 6.425
9□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
 - 0s - loss: 6.4391
Epoch 10/10
4539/4539 [==============================] - 0s - loss: 6.4391
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/504 [=>............................] - ETA: 8sEpoch 1/10
4539/4539 [==============================] - 1s - loss: 5.1912
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 2/10
4539/4539 [==============================] - 0s - loss: 4.1528
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4539/4539 [==============================] - 0s - loss: 1.2026
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
Epoch 4/10
4539/4539 [==============================] - 0s - loss: 0.9616

Epoch 5/10
4539/4539 [==============================] - 1s - loss: 0.9504

Epoch 6/10
4539/4539 [==============================] - 0s - loss: 0.9010

Epoch 7/10
4539/4539 [==============================] - 0s - loss: 0.8988

Epoch 8/10
4539/4539 [==============================] - 0s - loss: 0.8816

Epoch 9/10
4539/4539 [==============================] - 0s - loss: 0.8620

Epoch 10/10
4539/4539 [==============================] - 0s - loss: 0.8872

 50/504 [=>............................] - ETA: 9sEpoch 1/10
4539/4539 [==============================] - 1s - loss: 6.4360

Epoch 2/10
4539/4539 [==============================] - 0s - loss: 6.4360

Epoch 3/10
4539/4539 [==============================] - 0s - loss: 6.4360

Epoch 4/10
4539/4539 [==============================] - 0s - loss: 6.4360

Epoch 5/10
4539/4539 [==============================] - 0s - loss: 6.4360

Epoch 6/10
4539/4539 [==============================] - 0s - loss: 6.4360

Epoch 7/10
4539/4539 [==============================] - 0s - loss: 6.4360
```

```
Epoch 8/10
4539/4539 [==============================] - 0s - loss: 6.4360
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4539/4539 [==============================] - 0s - loss: 6.4360
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 10/10
4539/4539 [==============================] - 0s - loss: 6.4360
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/504 [=>............................] - ETA: 8sEpoch 1/10
4539/4539 [==============================] - 1s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 2/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
 - ETA: 0s - loss: 6.405
3□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 4/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 5/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 6/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 7/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 8/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 10/10
4539/4539 [==============================] - 0s - loss: 6.4404
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/504 [=>............................] - ETA: 8sEpoch 1/10
4539/4539 [==============================] - 1s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
Epoch 2/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 4/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 5/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 6/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 7/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 8/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 10/10
4539/4539 [==============================] - 0s - loss: 6.4281
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/504 [=>............................] - ETA: 8sEpoch 1/10
4539/4539 [==============================] - 1s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 2/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 4/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 5/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
Epoch 6/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 7/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 8/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 10/10
4539/4539 [==============================] - 0s - loss: 6.4367
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/504 [=>............................] - ETA: 8sEpoch 1/10
4539/4539 [==============================] - 1s - loss: 5.1441
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 2/10
4539/4539 [==============================] - 0s - loss: 4.5588
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 3/10
4539/4539 [==============================] - 0s - loss: 4.3143
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 4/10
4539/4539 [==============================] - 0s - loss: 2.8770
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 5/10
4539/4539 [==============================] - 0s - loss: 0.8968
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 6/10
4539/4539 [==============================] - 0s - loss: 0.8515
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 7/10
4539/4539 [==============================] - 0s - loss: 0.8657
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 8/10
4539/4539 [==============================] - 0s - loss: 0.8655
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 9/10
4539/4539 [==============================] - 0s - loss: 0.8202
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Epoch 10/10
```

```
4539/4539 [==============================] - 0s - loss: 0.8350
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

 50/504 [=>............................] - ETA: 8sResults: 4.79 (2.58) MAE
```

Finally we achieved a very good result of 98%. That seems to be a very good result.