In [39]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
%matplotlib inline
```

In [221…
```
#Setting up directory and uploading the dataset
os.chdir("C:\\Users\\tural\\OneDrive\\Desktop\\Study Materials")
df = pd.read_csv('movie_metadata.csv')
df.head(10)
```

Out[221]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_lik
0	Color	James Cameron	723.0	178.0	0.0	85
1	Color	Gore Verbinski	302.0	169.0	563.0	100
2	Color	Sam Mendes	602.0	148.0	0.0	16
3	Color	Christopher Nolan	813.0	164.0	22000.0	2300
4	NaN	Doug Walker	NaN	NaN	131.0	Na
5	Color	Andrew Stanton	462.0	132.0	475.0	53
6	Color	Sam Raimi	392.0	156.0	0.0	400
7	Color	Nathan Greno	324.0	100.0	15.0	28
8	Color	Joss Whedon	635.0	141.0	0.0	1900
9	Color	David Yates	375.0	153.0	282.0	1000

10 rows × 28 columns

In [222…
```
df.drop(['color'], axis=1, inplace = True)
```

Exploring the Dataset

In [223…

```
#Checking the data types and columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   director_name              4939 non-null   object
 1   num_critic_for_reviews     4993 non-null   float64
 2   duration                   5028 non-null   float64
 3   director_facebook_likes    4939 non-null   float64
 4   actor_3_facebook_likes     5020 non-null   float64
 5   actor_2_name               5030 non-null   object
 6   actor_1_facebook_likes     5036 non-null   float64
 7   gross                      4159 non-null   float64
 8   genres                     5043 non-null   object
 9   actor_1_name               5036 non-null   object
 10  movie_title                5043 non-null   object
 11  num_voted_users            5043 non-null   int64
 12  cast_total_facebook_likes  5043 non-null   int64
 13  actor_3_name               5020 non-null   object
 14  facenumber_in_poster       5030 non-null   float64
 15  plot_keywords              4890 non-null   object
 16  movie_imdb_link            5043 non-null   object
 17  num_user_for_reviews       5022 non-null   float64
 18  language                   5031 non-null   object
 19  country                    5038 non-null   object
 20  content_rating             4740 non-null   object
 21  budget                     4551 non-null   float64
 22  title_year                 4935 non-null   float64
 23  actor_2_facebook_likes     5030 non-null   float64
 24  imdb_score                 5043 non-null   float64
 25  aspect_ratio               4714 non-null   float64
 26  movie_facebook_likes       5043 non-null   int64
dtypes: float64(13), int64(3), object(11)
memory usage: 1.0+ MB
```

In [224…

```
# Checking the number of NAs
df.isna().sum().sum()
```

Out[224]:

```
2679
```

In [225…

```
df.imdb_score.isna().sum()
```

Out[225]:

```
0
```

In [226…

```
df.value_counts('country')
```

Out[226]:
```
country
USA            3807
UK              448
France          154
Canada          126
Germany          97
               ...
Nigeria            1
New Line           1
Libya              1
Kyrgyzstan         1
Afghanistan        1
Length: 65, dtype: int64
```

In [8]:
```
df.value_counts('director_name')
```

Out[8]:
```
director_name
Steven Spielberg    26
Woody Allen         22
Clint Eastwood      20
Martin Scorsese     20
Ridley Scott        17
                    ..
James O'Brien        1
James Nunn           1
James Mottern        1
James Melkonian      1
Étienne Faure        1
Length: 2398, dtype: int64
```

In [9]:
```
df[df['imdb_score'] > 8].value_counts('director_name')
```

Out[9]:
```
director_name
Martin Scorsese      8
Christopher Nolan    7
Quentin Tarantino    5
Steven Spielberg     5
Stanley Kubrick      4
                    ..
Irvin Kershner       1
Ingmar Bergman       1
Harold Ramis         1
Gus Van Sant         1
Woody Allen          1
Length: 154, dtype: int64
```

In [10]:
```
df.value_counts('actor_2_name')
```

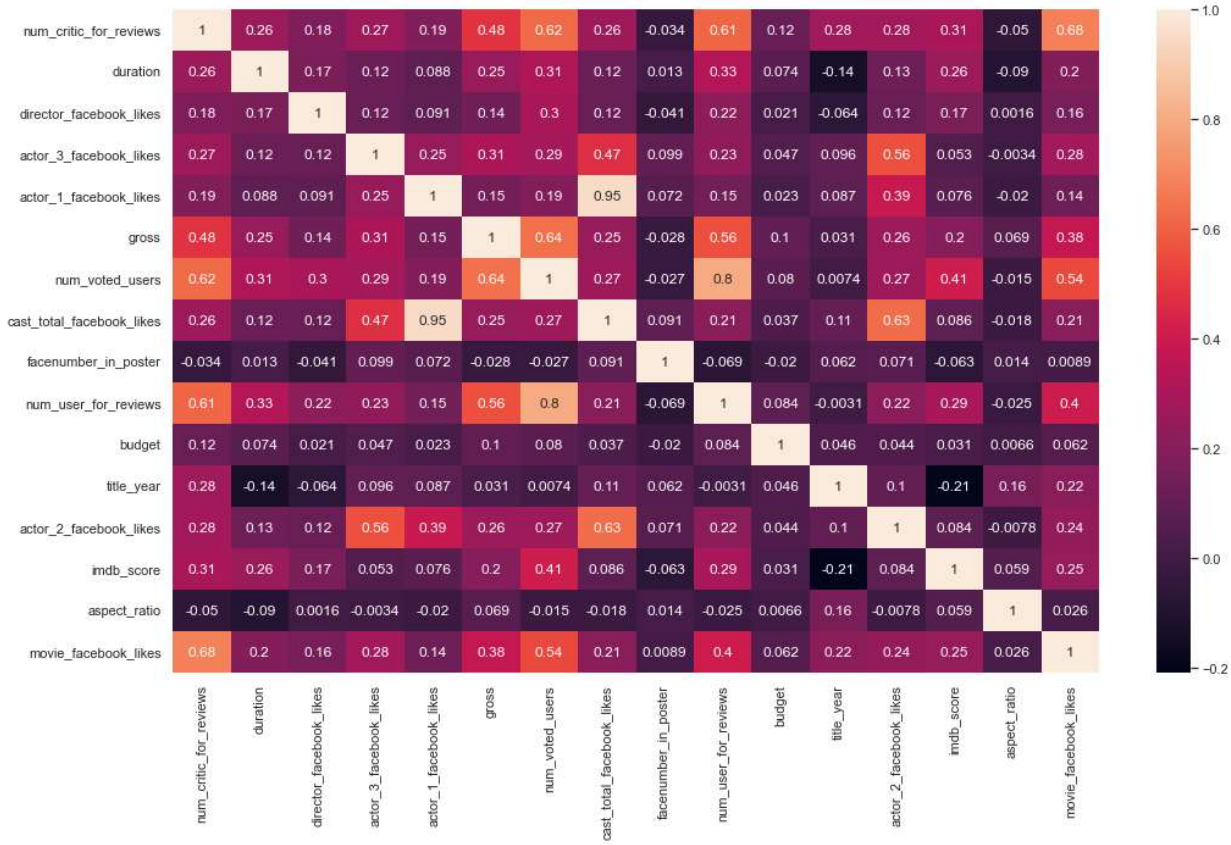
Out[10]:
```
actor_2_name
Morgan Freeman      20
Charlize Theron     15
Brad Pitt           14
James Franco        11
Meryl Streep        11
                    ..
Jodi Lyn O'Keefe     1
Jodie Whittaker      1
Joe Armstrong        1
Joe Bishop           1
Zubaida Sahar        1
Length: 3032, dtype: int64
```

In [11]:
```
df[df['imdb_score'] > 8].value_counts('actor_2_name')
```

Out[11]:
```
actor_2_name
Brad Pitt            4
Orlando Bloom        3
John Ratzenberger    3
Shin'ya Tsukamoto    2
Olaf Lubaszenko      2
                    ..
Gloria Foster        1
Graham McTavish      1
Greg Speirs          1
Gustav Fröhlich      1
Yaphet Kotto         1
Length: 232, dtype: int64
```

In [12]:
```
# Checking the correlation between continous variables to see if linear regression car
sns.set(rc = {'figure.figsize':(17,10)})
sns.heatmap(df.corr(), annot = True)
```

Out[12]:
```
<AxesSubplot:>
```



- As there is no any significantly related variable to imdb score, we will not use linear regression. Instead we can create groups by the imdb scores as high, low, medium and use decision tree model.

In [13]:
```
# Detecting the cut points for creating groups in imdb score
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(y = 'imdb_score', data = df, )
print(df.imdb_score.describe())
```

```
count    5043.000000
mean        6.442138
std         1.125116
min         1.600000
25%         5.800000
50%         6.600000
75%         7.200000
max         9.500000
Name: imdb_score, dtype: float64
```

- We will use upper quartile and above for High , between upper and lower quartile for Medium, beneath than the lower quartile for low

In [255…
```
bins = [0, 5.8, 7.2, np.inf]
names = ['Low', 'Medium', 'High']
df['imdb_cat'] = pd.cut(df['imdb_score'], bins, labels=names)
```

In [256…
```
# Getting the continous variables
df_numeric = df._get_numeric_data()
df_numeric['imdb_cat'] = df['imdb_cat']
df_numeric.drop(["imdb_score", 'title_year'], axis = 1, inplace = True)
```

In [257…
```
df_numeric.value_counts('imdb_cat')
```

Out[257]:
```
imdb_cat
Medium    2515
Low       1310
High      1218
dtype: int64
```

In [258…
```
df_numeric.isna().sum().sum()
```

Out[258]:
```
1951
```

In [259…
```
df_numeric.isna().sum()
```

Out[259]:
```
num_critic_for_reviews        50
duration                      15
director_facebook_likes      104
actor_3_facebook_likes        23
actor_1_facebook_likes         7
gross                        884
num_voted_users                0
cast_total_facebook_likes      0
facenumber_in_poster          13
num_user_for_reviews          21
budget                       492
actor_2_facebook_likes        13
aspect_ratio                 329
movie_facebook_likes           0
imdb_cat                       0
dtype: int64
```

In [260…
```
df_numeric.dropna(inplace=True)
```

In [261…
```
df_numeric.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3801 entries, 0 to 5042
Data columns (total 15 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   num_critic_for_reviews     3801 non-null   float64
 1   duration                   3801 non-null   float64
 2   director_facebook_likes    3801 non-null   float64
 3   actor_3_facebook_likes     3801 non-null   float64
 4   actor_1_facebook_likes     3801 non-null   float64
 5   gross                      3801 non-null   float64
 6   num_voted_users            3801 non-null   int64
 7   cast_total_facebook_likes  3801 non-null   int64
 8   facenumber_in_poster       3801 non-null   float64
 9   num_user_for_reviews       3801 non-null   float64
 10  budget                     3801 non-null   float64
 11  actor_2_facebook_likes     3801 non-null   float64
 12  aspect_ratio               3801 non-null   float64
 13  movie_facebook_likes       3801 non-null   int64
 14  imdb_cat                   3801 non-null   category
dtypes: category(1), float64(11), int64(3)
memory usage: 449.3 KB
```

In [262…
```
df_numeric.iloc[:, 8]
```

Out[262]:
```
0       0.0
1       0.0
2       1.0
3       0.0
5       1.0
       ...
5027    0.0
5029    0.0
5033    0.0
5035    0.0
5042    0.0
Name: facenumber_in_poster, Length: 3801, dtype: float64
```

Decision Trees

In [263…
```
X = df_numeric.drop(['imdb_cat'], axis = 1)
y = df_numeric['imdb_cat']
```

In [264…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, stratify= y)
```

In [265…
```
dtree = DecisionTreeClassifier()
```

In [266…
```
dtree.fit(X_train,y_train)
```

Out[266]:
```
DecisionTreeClassifier()
```

In [267…
```
predictions_tree = dtree.predict(X_test)
```

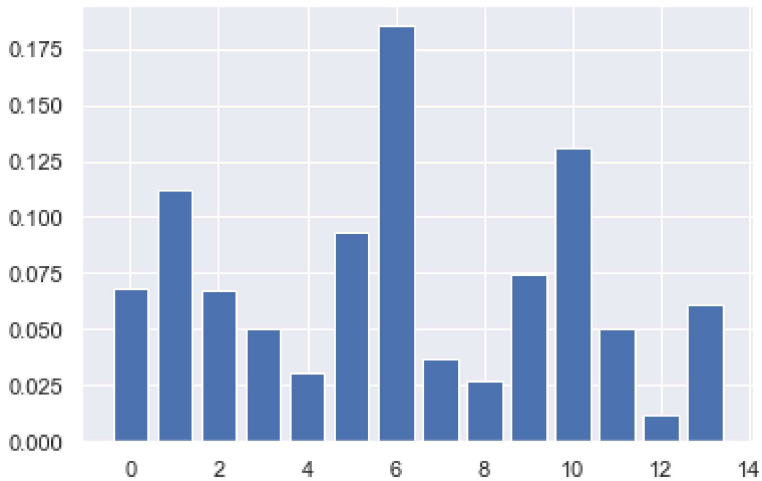
In [268…
```
print(confusion_matrix(y_test,predictions_tree))
print("\n")
print(classification_report(y_test, predictions_tree))
```

```
[[174   9  82]
 [ 22 144 114]
 [118 125 353]]


              precision    recall  f1-score   support

        High       0.55      0.66      0.60       265
         Low       0.52      0.51      0.52       280
      Medium       0.64      0.59      0.62       596

    accuracy                           0.59      1141
   macro avg       0.57      0.59      0.58      1141
weighted avg       0.59      0.59      0.59      1141
```

In [269…
```
importance = dtree.feature_importances_
```

In [270…
```
# summarize feature importance
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

```
Feature: 0, Score: 0.06852
Feature: 1, Score: 0.11197
Feature: 2, Score: 0.06717
Feature: 3, Score: 0.05035
Feature: 4, Score: 0.03068
Feature: 5, Score: 0.09321
Feature: 6, Score: 0.18523
Feature: 7, Score: 0.03671
Feature: 8, Score: 0.02673
Feature: 9, Score: 0.07456
Feature: 10, Score: 0.13130
Feature: 11, Score: 0.05035
Feature: 12, Score: 0.01225
Feature: 13, Score: 0.06098
```



Random Forest

In [271…
```
X = df_numeric.drop(['imdb_cat'], axis = 1)
```

In [272…
```
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

Out[272]:
```
RandomForestClassifier()
```

In [273…
```
rfc_pred = rfc.predict(X_test)
```

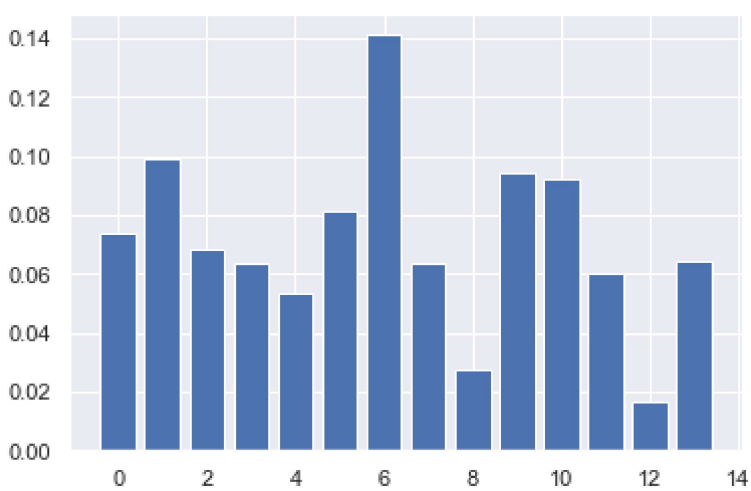
In [274…
```
print(confusion_matrix(y_test,rfc_pred))
print("\n")
print(classification_report(y_test, rfc_pred))
```

```
[[153   1 111]
 [  3 136 141]
 [ 43  50 503]]


              precision    recall  f1-score   support

        High       0.77      0.58      0.66       265
         Low       0.73      0.49      0.58       280
      Medium       0.67      0.84      0.74       596

    accuracy                           0.69      1141
   macro avg       0.72      0.64      0.66      1141
weighted avg       0.71      0.69      0.69      1141
```

In [275…

```
# get importance
importance = rfc.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

```
Feature: 0, Score: 0.07387
Feature: 1, Score: 0.09920
Feature: 2, Score: 0.06868
Feature: 3, Score: 0.06336
Feature: 4, Score: 0.05344
Feature: 5, Score: 0.08157
Feature: 6, Score: 0.14121
Feature: 7, Score: 0.06345
Feature: 8, Score: 0.02772
Feature: 9, Score: 0.09449
Feature: 10, Score: 0.09208
Feature: 11, Score: 0.05998
Feature: 12, Score: 0.01646
Feature: 13, Score: 0.06450
```



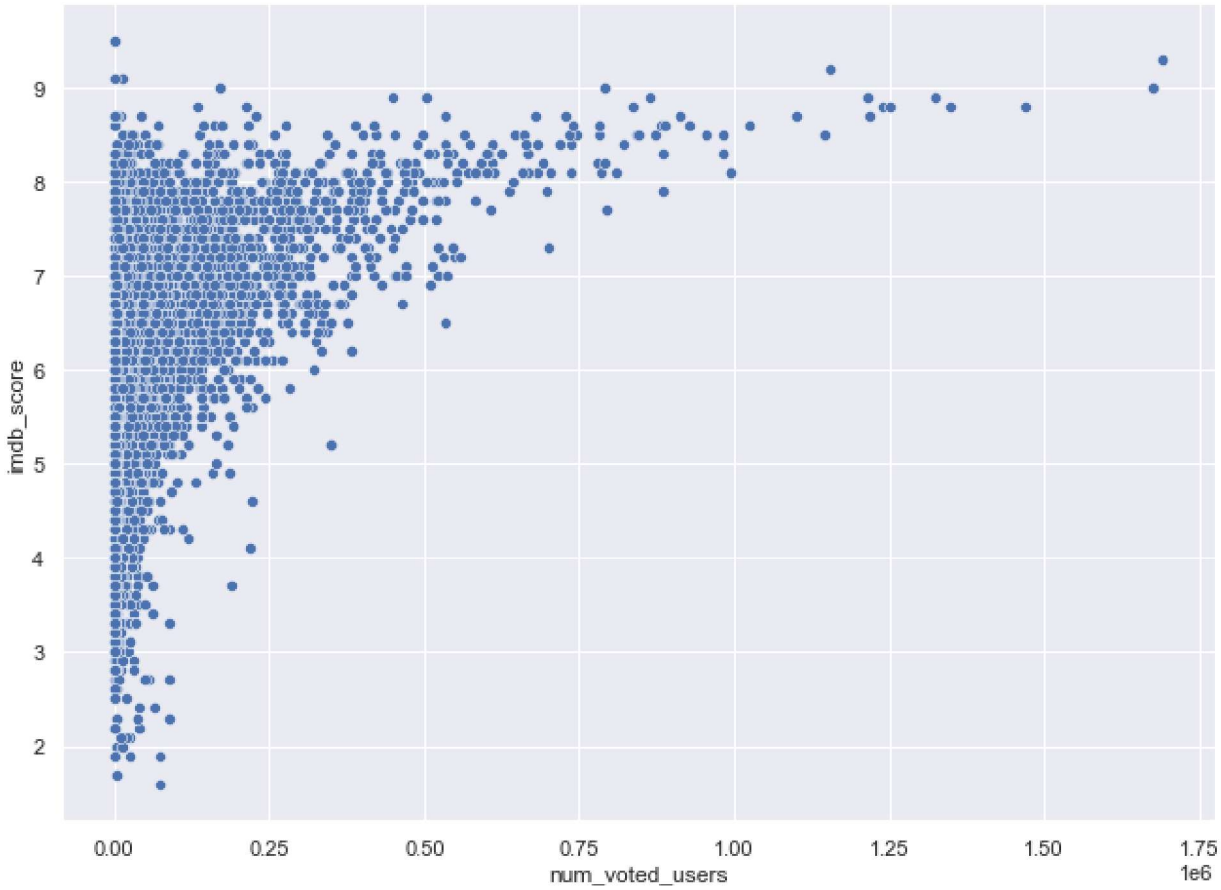
Linear Regression

It was already obvious that Linear regression would not be a good fit from the heatmap. Below is just for the sake of showing what could be done if the data had linear pattern.

In [200…
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
```

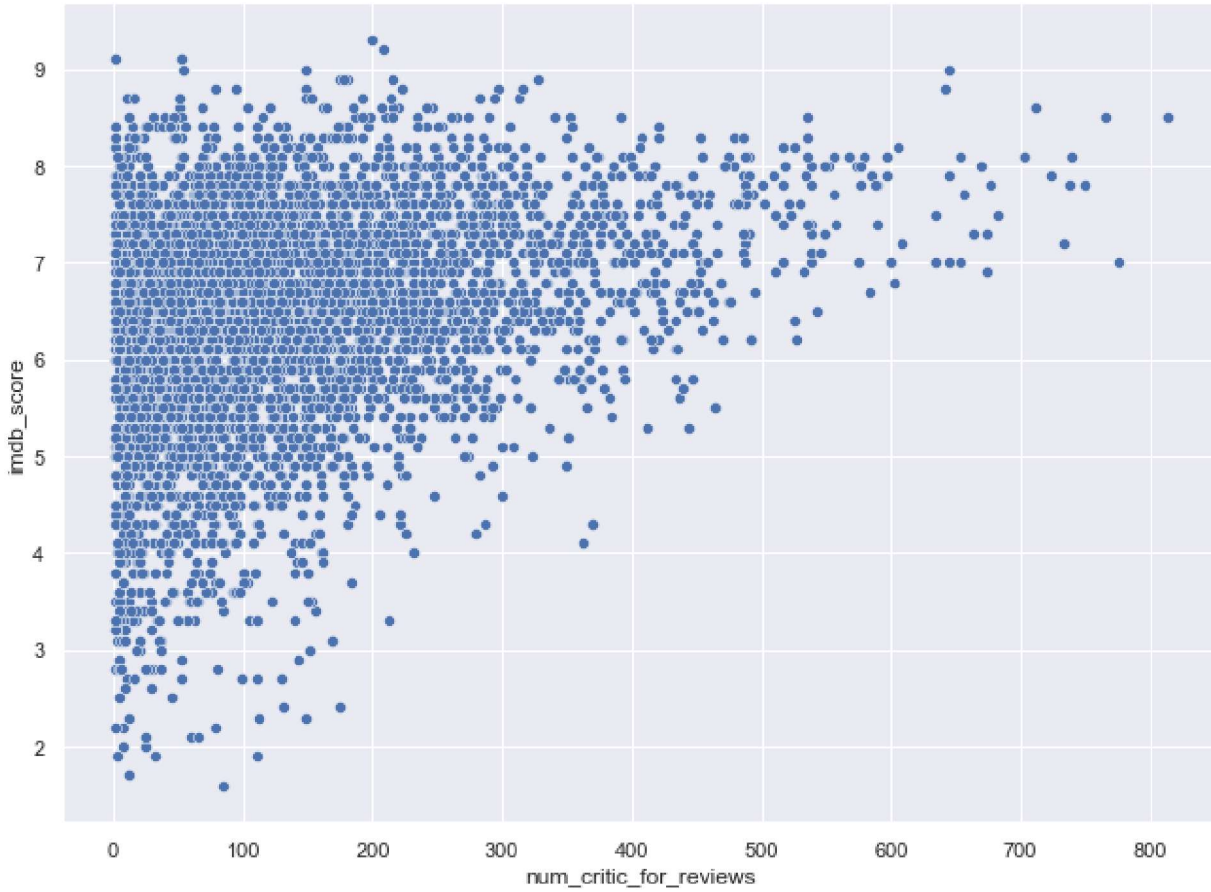
In [195…
```
fig, ax = plt.subplots(figsize=(11, 8))
sns.scatterplot(x = df['num_voted_users'], y = df['imdb_score'])
```

Out[195]:
```
<AxesSubplot:xlabel='num_voted_users', ylabel='imdb_score'>
```



In [196…
```
fig, ax = plt.subplots(figsize=(11, 8))
sns.scatterplot(x = df['num_critic_for_reviews'], y = df['imdb_score'])
```

Out[196]:
```
<AxesSubplot:xlabel='num_critic_for_reviews', ylabel='imdb_score'>
```



In [202…

```
df = df.fillna(df.median())
```

```
C:\Users\tural\AppData\Local\Temp\ipykernel_22796\3493596106.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is depre
cated; in a future version this will raise TypeError.  Select only valid columns befo
re calling the reduction.
  df = df.fillna(df.median())
```

In [197…

```
df.columns
```

Out[197]:

```
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',
       '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')
```

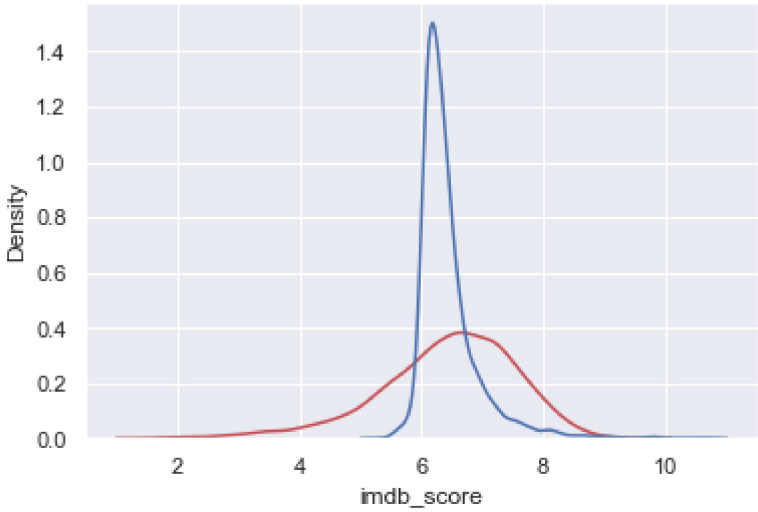
In [203…

```
#Lets develop our model and then show our equation for predicting the ratings
X = df[['num_critic_for_reviews', "duration", "num_voted_users", "num_user_for_reviews
Y = df['imdb_score']
lm.fit(X, Y)
Yhat = lm.predict(X)
```

In [204…

```
# Finally lets draw a distribution plot in order to compare the actual score with our
ax1 = sns.distplot(df['imdb_score'], hist = False, color = 'r', label = "Actual Writin
sns.distplot(Yhat, hist= False, color = 'b', label = "Fitted Value", ax = ax1)
```

```
C:\Users\tural\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\tural\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

Out[204]:

```
<AxesSubplot:xlabel='imdb_score', ylabel='Density'>
```