Movies Dataset - IMDB Scores

Introduction:

This dataset is obtained from "Insert Link Here" and contains information regarding movies released between "19xx" to "20xx". The metadata includes the Release Date, Budget, Director, Production House, Genre, Partental Rating, Name of the movie, Writer, IMDB Rating among other features.

The purpose of this analysis is to clean the dataset per the attainable end-goal and perform EDA to facilitate the process of cleaning the dataset. Part of the EDA will include:

1. Identifying any null or missing data points within the interested feature space
2. Identifying and removing outliers from the dataset as appropriate
3. Identifying any significant presence of multi-collinearity between the interested feature space
4. Generating 3 hypothesis of the dataset within the identified feature space and its relationt to the 'response variable' (IMDB score)
5. Understanding the frequency (residual) distribution of the interested feature space
6. Normalizing and scaling the feature space as appropriate, per the EDA performed in step 4
7. Performing analysis on 2 of the hypothesis and providing evidance of whether we accept or reject the null hypothesis made earlier
8. Creating a finalized set of features and providing options for exploring other potential normalization and scaling techniques
9. Providing suggestive next steps to run an ML Model to make predictions on the response variable (IMDB score)

In [44]:

```
#MoviesDatasetChallenge:
import numpy as np
import pandas as pd
import matplotlib as plt
import plotly
import streamlit as st
import plotly.graph_objects as go
```

In [2]:

```
#Import the movies dataset:
import csv
movies_dataset = pd.read_excel(r'/Users/apaintal/Documents/movies_dataset.xlsx', sheet_name=0)
```

In [3]:

```
#Print the dataset heading:
movies_dataset.head()
```

Out[3]:

	budget	company	country	director	genre	gross	name	rating	released	runtime	score	star	votes	wr
0	300000000	Walt Disney Pictures	USA	Gore Verbinski	Action	309420425	Pirates of the Caribbean: At World's End	PG-13	2007-05-25 00:00:00	169	7.1	Johnny Depp	514191	El
1	260000000	Walt Disney Animation Studios	USA	Nathan Greno	Animation	200821936	Tangled	PG	2010-11-24 00:00:00	100	7.8	Mandy Moore	325621	I Fogelr
2	258000000	Columbia Pictures	USA	Sam Raimi	Action	336530303	Spider-Man 3	PG-13	2007-05-04 00:00:00	139	6.2	Tobey Maguire	416842	S Ra
3	250000000	Walt Disney Pictures	USA	Rob Marshall	Action	241071802	Pirates of the Caribbean: On Stranger Tides	PG-13	2011-05-20 00:00:00	100	6.6	Johnny Depp	408968	El
4	250000000	Walt Disney	USA	Andrew	Action	73078100	John	PG-	2012-03-	132	6.6	Taylor	224224	And

	budget	company	country	director	genre	gross	name	rating	released	runtime	score	star	votes	wr
4	250000000	Disney Pictures	USA	Stanton	Action	73078100	Carter	13	09 00:00:00	132	6.6	Kitsch	224234	Star

Exploring the dataset statistics:

In [18]:

```
#Group the dataset into different genres to find the average rating among different genres:
genres = movies_dataset.groupby(['genre'])['score'].mean()

print(genres)

print(movies_dataset.describe())
```

```
genre
Action       6.106086
Adventure    6.354847
Animation    6.746931
Biography    7.040390
Comedy       6.165769
Crime        6.756322
Drama        6.715720
Family       5.792857
Fantasy      5.775000
Horror       5.687004
Musical      6.650000
Mystery      6.342105
Romance      6.126667
Sci-Fi       6.253846
Thriller     5.877778
War          6.400000
Western      6.200000
Name: score, dtype: float64
             budget         gross     runtime        score         votes  \
count  6.820000e+03  6.820000e+03  6820.00000  6820.000000  6.820000e+03
mean   2.458113e+07  3.349783e+07   106.55132     6.374897  7.121952e+04
std    3.702254e+07  5.819760e+07    18.02818     1.003142  1.305176e+05
min    0.000000e+00  7.000000e+01    50.00000     1.500000  2.700000e+01
25%    0.000000e+00  1.515839e+06    95.00000     5.800000  7.665250e+03
50%    1.100000e+07  1.213568e+07   102.00000     6.400000  2.589250e+04
75%    3.200000e+07  4.006534e+07   115.00000     7.100000  7.581225e+04
max    3.000000e+08  9.366622e+08   366.00000     9.300000  1.861666e+06

              year
count  6820.000000
mean   2001.000293
std       8.944501
min    1986.000000
25%    1993.000000
50%    2001.000000
75%    2009.000000
max    2016.000000
```

3 Hypothesis about the dataset:

1. Movies with budgets greater than $80M tend to perform better in terms of IMDB score, than those below this budget
2. Movies by Major Production Houses such as "Walt Disney Productions" and "Warner Bros. Productions", tend to perform better, on avergae, than other smaller Production Houses, in terms of IMDB score
3. Movies with positive gross margins (with multiplier ratios* > 1.5) tend to perform better (i.e. have higher IMDB score) than those with gross margins less than 1.5. (In other words, box-office successful movies tend go be likened by Critics as well as the movie-going public)

*Multiplier Ratio = ((domestic + international revenue)/ budget)

Clean the dataset to remove the '0' and 'null' values in 'budget' column & using a minimum threshold of $100,000 for a movie budget:

In [6]:

```
budget_col = movies_dataset.loc[movies_dataset['budget']<= 100000]
budget_threshold_set = movies_dataset.loc[movies_dataset['budget'] > 100000]
```

```
budget_threshold_set = movies_dataset.loc[movies_dataset['budget'] > 100000]

print(budget_col.shape)
print(budget_threshold_set.shape)
budget_col.head()
```

```
(2210, 15)
(4610, 15)
```

Out[6]:

	budget	company	country	director	genre	gross	name	rating	released	runtime	score	star	votes	writer	y
25	0	TriStar Pictures	USA	John Badham	Comedy	40697761	Short Circuit	PG	1986-05-09 00:00:00	98	6.6	Ally Sheedy	47068	S.S. Wilson	1
26	0	Neue Constantin Film	Italy	Jean-Jacques Annaud	Crime	7153487	The Name of the Rose	R	1986-09-24 00:00:00	130	7.8	Sean Connery	86991	Umberto Eco	1
27	0	TriStar Pictures	USA	Sidney J. Furie	Action	24159872	Iron Eagle	PG-13	1986-01-17 00:00:00	117	5.3	Louis Gossett Jr.	11304	Kevin Alyn Elders	1
32	0	Gaumont	France	Jean-Jacques Beineix	Drama	2003822	Betty Blue	Not specified	1986-11-07 00:00:00	120	7.4	Jean-Hugues Anglade	14562	Philippe Djian	1
35	0	Columbia Pictures Corporation	USA	John G. Avildsen	Action	115103979	The Karate Kid Part II	PG	1986-06-20 00:00:00	113	5.9	Pat Morita	58370	Robert Mark Kamen	1

Filtering the dataset to only include the "interested features", which will be used to perform analysis on the hypothesis made about this data

In [8]:

```
#Check to see if the distribution of the residuals in normal:
fil_datatset_1 = budget_threshold_set[['budget','company','director','genre','score']]

print(fil_datatset_1.shape)
fil_datatset_1.head()
```

```
(4610, 5)
```

Filtering the dataset to include only Production Houses and Directors who've made at least 5 movies

In [28]:

```
from sklearn.preprocessing import MinMaxScaler

print('List of unique Production Houses in the data set, before filtering: ',
len(set(fil_datatset_1['company'])))
print('List of unique Directors in the data set, before filtering: ', len(set(budget_threshold_set
['director'])))

#Get the list of all Production Houses who've Produced at least 5 movies:
get_companies = fil_datatset_1['company'].value_counts().to_frame().reset_index()
get_companies = get_companies.rename(columns = {'index':'company','company':'count'})

get_directors = fil_datatset_1['director'].value_counts().to_frame().reset_index()
get_directors = get_directors.rename(columns = {'index':'director','director':'count'})

#Filter data where value_counts is less than 5:
get_companies['count'] = pd.to_numeric(get_companies['count'])
get_directors['count'] = pd.to_numeric(get_directors['count'])
filtered_comps = get_companies.loc[get_companies['count']>=5]
filtered_comps_1 = get_companies.loc[get_companies['count']<5]

filtered_direcs = get_directors.loc[get_directors['count']>=5]

list_of_companies = list(set(filtered_comps['company']))
```

```
list_of_directors = list(set(filtered_direcs['director']))

print('')
print('List of unique Production Houses in the data set, After filtering: ', len(list_of_companies
))
print('List of unique Directors in the data set, After filtering: ', len(list_of_directors))

#Filtering the dataset to only include Producers and Directors who've Produces and Directed at lea
st 5 movies, respectively:
fil_datatset_2 = fil_datatset_1[fil_datatset_1['company'].isin(list_of_companies)]
fil_datatset_3 = fil_datatset_2[fil_datatset_2['director'].isin(list_of_directors)]

print('')
print('Shape of the Filtered DataFrame:')
print(fil_datatset_3.shape)
```

```
List of unique Production Houses in the data set, before filtering:  1319
List of unique Directors in the data set, before filtering:  1878

List of unique Production Houses in the data set, After filtering:  106
List of unique Directors in the data set, After filtering:  251

Shape of the Filtered DataFrame:
(1460, 9)
```

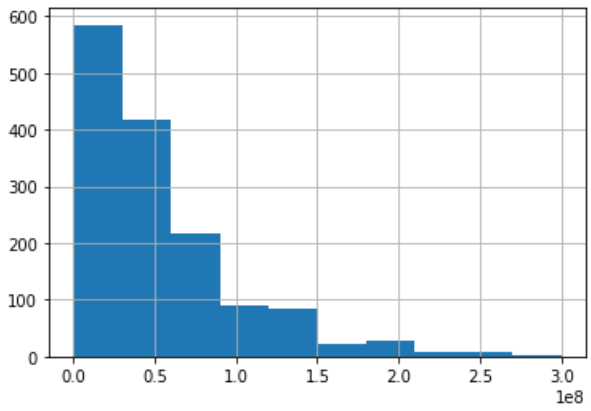
Normalizing and scaling the 'budget' feature space to be used for regression modelling & ensuring independence of the feature space with respect to other feature variables (i.e. Check for Normality to ensure regression models can be accurately applied on the 'interested feature' space):

In [39]:

```
#Explore the feature space of relationship with target variable of the DataFrame:
other_comps = set(filtered_comps_1['company'])
fil_datatset_3 = fil_datatset_3.replace(other_comps, 'other_comps')
comp_one_hot_enc = pd.get_dummies(fil_datatset_3['company'], drop_first = True)

new_dataf = pd.DataFrame()
fil_datatset_3_2 = fil_datatset_3[['budget','company','director', 'score',]]

#Merging the one-hot-encoded dataset to budget column:
new_dataf = pd.concat([fil_datatset_3_2, comp_one_hot_enc], axis = 1)

print(new_dataf['budget'].describe())
print(new_dataf['budget'].hist())

#Find the mean, min and max of this data set to normalize the residuals of the dataset:
mean_budget = new_dataf['budget'].mean()
max_budget = new_dataf['budget'].max()
min_budget = new_dataf['budget'].min()

print(mean_budget, max_budget, min_budget)

#Normalizing the independent variable ('budget') via applying simulating a Min-Max_Scaler funtion
through computing
#the Maximum, Minimum and Average budget of movies in the DataFrame:
new_val = []
for i in new_dataf['budget']:
    new_val.append((i-mean_budget)/(max_budget - min_budget))

#Printing the histogram of normaled bufget values to check for normality, kurtosis and skewness:
import matplotlib.pyplot as plt
plt.hist(new_val)
plt.show()

#Since the dataset is heavily 'right skewed', we will apply another normaliing technique:
#We expect RobustScaling to also exhibit a similar kind of 'right skewedness', however, the values
of normalized
#valrable (Budget) will take on a larger range than (0,1):

from sklearn.preprocessing import RobustScaler
reshape_budget_2 = np.array(new_dataf['budget']).reshape(-1,1)
Robust_Scaler = RobustScaler()
transformed_budget_2 = Robust_Scaler.fit_transform(reshape_budget_2)
new_dataf['trans_budget'] = transformed_budget_2
```

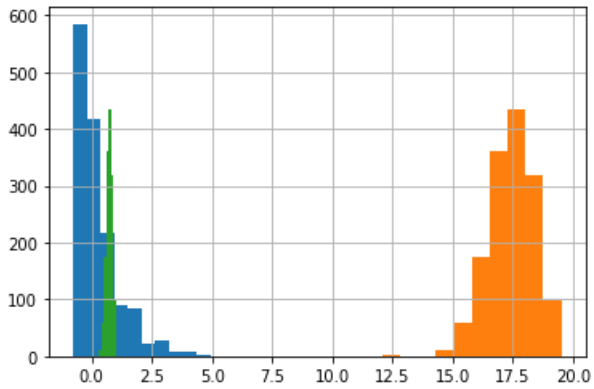
```
print(new_dataf['trans_budget'].hist())

#The data is still heavily 'right skewed'

#We will now apply the log transformation, as looking at the 'skewness' of the dataset, we see tha
t lower budget
#values are exponentially higher than high budget movies. Applying the natural log should help to
normalize this
#distribution and remove this skewness:

#Apply log-normalization transformation:
log_trns_budget = np.log(new_dataf.budget + 1)
new_dataf['log_trns_budget'] = log_trns_budget
print(new_dataf['log_trns_budget'].describe())
print(new_dataf['log_trns_budget'].hist())

#To run a Linear Regression model, we can further narrow the range of values using MinMaxScaler():
min_max_sc = MinMaxScaler()
reshape_lognorm_budget = np.array(new_dataf['log_trns_budget']).reshape(-1,1)
log_norm_budget = min_max_sc.fit_transform(reshape_lognorm_budget)
#transformed_budget_val = transformed_budget.transform(fil_datatset_1['budget'])
new_dataf['normalized_budget'] = log_norm_budget

print(new_dataf['normalized_budget'].describe())
print(new_dataf['normalized_budget'].hist())

#As can be seen from the 2nd histogram below, the frequeny distribution of the original 'budget' d
ata is 'right skewed':
#While the 'log tranformed' budget dataset is normally distributed.
#On further normalizing the 'log transformed' dataset using MinMaxScaler(), the varibale range is
shrunk between 0 and 1:
```

```
count    1.460000e+03
mean     5.463960e+07
std      4.717827e+07
min      1.750000e+05
25%      2.075000e+07
50%      4.000000e+07
75%      7.350000e+07
max      3.000000e+08
Name: budget, dtype: float64
AxesSubplot(0.125,0.125;0.775x0.755)
54639602.74041096 300000000 175000
```




```
AxesSubplot(0.125,0.125;0.775x0.755)
count    1460.000000
mean       17.448976
std         0.917204
min        12.072547
25%        16.847836
50%        17.504390
75%        18.112727
max        19.519293
Name: log_trns_budget, dtype: float64
AxesSubplot(0.125,0.125;0.775x0.755)
count    1460.000000
mean        0.721984
std         0.123168
```

```
min          0.000000
25%          0.641258
50%          0.729425
75%          0.811117
max          1.000000
Name: normalized_budget, dtype: float64
AxesSubplot(0.125,0.125;0.775x0.755)
```



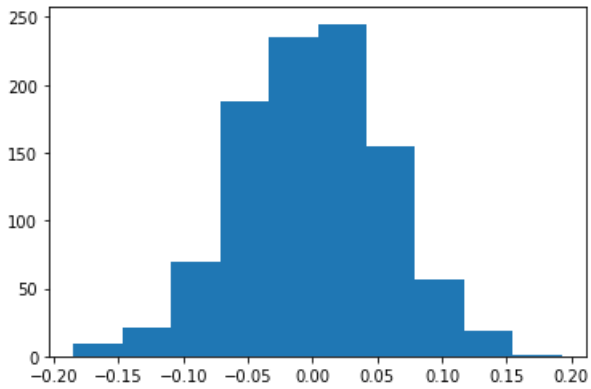
Performing the signifcance level (p-value test), to guage if a 'higher budget' movie tends to critically and popularly do better (i.e. higher IMDB score) than those with 'lower budget'

In [33]:

```
other_comps = set(filtered_comps_1['company'])

fil_datatset_3 = fil_datatset_3.replace(other_comps, 'other_comps')
comp_one_hot_enc = pd.get_dummies(fil_datatset_3['company'], drop_first = True)

new_dataf = pd.DataFrame()
fil_datatset_3_2 = fil_datatset_3[['budget','company','director', 'score',]]

#Merging the one-hot-encoded dataset to budget column:
new_dataf = pd.concat([fil_datatset_3_2, comp_one_hot_enc], axis = 1)

print(fil_datatset_3.shape)

#Making 'score' as the last column in DF:
mid = new_dataf['score']
new_dataf.drop(labels=['score'], axis=1, inplace = True)
new_dataf.insert(95, 'score', mid)
#print(new_dataf.head())

new_dataf = new_dataf.reset_index(drop = True)
#print(new_dataf.head())

#Fit a linear model on the dataset:
from sklearn.linear_model import LinearRegression

linear_model = LinearRegression()

from sklearn.model_selection import train_test_split

new_dataf_1 = new_dataf.copy()
new_dataf_1 = new_dataf_1.drop(columns = ['director', 'company'])

X_data = new_dataf_1.iloc[:, :-1]
Y_data = new_dataf_1['score']

print(X_data.shape, Y_data.shape)

#Convert Y_data into a 1-D array:
Y_data = Y_data.to_numpy()
Y_data = Y_data.reshape(-1,1)

#Train teh dataset and predict movie scores based on training input dataset:
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size = 0.28, random_state
= 0)
linear_model.fit(X_train, Y_train)
pred_values = linear_model.predict(X_test)
```

```
X_train_1 = new_dataf.iloc[:, :-1]
X_train_1.drop(columns = ['company'], axis = 1, inplace = True)

#Label encode the 'director' categorical feature using "Target Encoding" via the
'category_encoders' package in Python
from category_encoders import TargetEncoder
enc = TargetEncoder(cols = ['director'])
training_set = enc.fit_transform(X_train_1, Y_data)


training_set.drop(columns = ['score'], axis = 1, inplace = True)

#1. Check if movies with higher budgets have higher IMDB ratings or not. Whether this test is stat
istically significant
#A. Take the mean budget and the mean IMDB score for the corresponding budgeted movies and take th
e P-value via
# taking the mean difference betwen the sample population and dividing it by the sample frequncy o
r sample size
# Null Hypothesis is that there's no statistical significance between IMDB ratings of high budget
films (>=$100M) v.s.
# those with lower budgets (< $80M):

high_budget_movies = fil_datatset_3_2[fil_datatset_3_2['budget']>=80000000]
mean_sc_high_budget = high_budget_movies.score.mean()

lower_budget_movies = fil_datatset_3_2[fil_datatset_3_2['budget']<80000000]
mean_sc_low_budget = lower_budget_movies.score.mean()

count_of_high_budg_obs = high_budget_movies.score.count()
count_of_low_budg_obs = lower_budget_movies.score.count()

#Computing the proportion of movies which are high budget from the entire sample dataset
#(This will be our 'A Priori' probablity of a movie being 'High Budget'):
perc_high_budget = count_of_high_budg_obs/(count_of_high_budg_obs + count_of_low_budg_obs)

#Creating an iteration of 1000 samples from the Population, to compute the p_value of scores betwe
en 'High Budget' and
#'Lower Budget' movies being statistically the same or different:
# We will take our 'alpha' threshold to be 0.05 or 5% significance level

sample_diff = []
for i in range(1000):
    high_budget = []
    low_budget = []
    for val in fil_datatset_3_2.score:
        rand_val = np.random.random()

        if rand_val <= perc_high_budget:
            high_budget.append(val)

        else:
            low_budget.append(val)
    mean_diff = np.mean(high_budget) - np.mean(low_budget)
    sample_diff.append(mean_diff)

#Printing the histogram of Sample means between scores of 'High Budget' v.s. 'Lower Budget' movies
(Chck for normality):
import matplotlib.pyplot as plt
plt.hist(sample_diff)
plt.show()

#Finding p-value:
total = 0
for difference in sample_diff:
    if abs(difference) > 0.1:
        total += 1

p_value = total/1000

print(p_value)
```

```
(1460, 9)
(1460, 95) (1460,)
0.22465753424657534
0.081
```



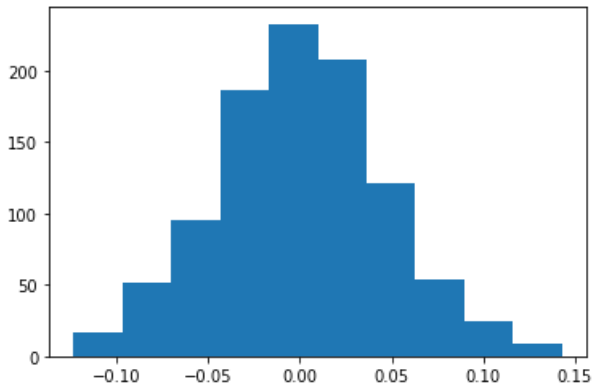
Performing the signifcance level (p-value test), to guage if movies produced by 'major production houses' tend to critically and popularly do better (i.e. higher IMDB score) than those produced by other, 'smaller production houses'.

In [35]:

```
#Implement 'Target Encoding' on the companies
#3. Check if movies produced by certain companies are sigificantly better rated (in terms of IMDB
score) than other movie producing companies
#One more hypothesis:
#Steps to be executed:
#1. Filter out companies who've produced less than 5 companies
#2. Group the companies between "Major Production Houses" v.s. "Others"
#3. Compute the probability of a company being a "Major Production House" v.s. being "Others"
#4. Simulate the random sampling of movies produced by a "Major Prodcution House" v.s. "Others" in
a sample frequency of 1000

#Definition of "Major Production House" is the list provided here:
major_prod_house = ['Universal Pictures', 'Pixar Animation Studios', 'Imagine Entertainment','New
Line Cinema', 'Walt Disney Animation Studios', 'DreamWorks Animation', \
                    'The Weinstein Company', 'Fox Searchlight Pictures', 'DreamWorks','Lucasfilm','
Twentieth Century Fox Film Corporation','Walt Disney Feature Animation' \
                   'Fox 2000 Pictures', 'Columbia Pictures Corporation', 'Metro-Goldwyn-Mayer (MGM)
','Columbia Pictures','BBC Films', 'Warner Bros.', 'Paramount Pictures' \
                   'Sony Pictures Classics', 'Touchstone Pictures', 'Lionsgate','British
Broadcasting Corporation (BBC)','Lions Gate Films','Marvel Studios']

fil_datatset_3_3 = fil_datatset_3_2.copy()
fil_datatset_3_major_prod = fil_datatset_3_3[fil_datatset_3_3['company'].isin(major_prod_house)]
fil_datatset_3_others = fil_datatset_3_3[~fil_datatset_3_3['company'].isin(major_prod_house)]

major_prod_mean = fil_datatset_3_major_prod.score.mean()
other_prod_mean = fil_datatset_3_others.score.mean()

#Computing the frequency of movie produced by a 'Major Production House' v.s. those produced by ot
her Production Houses:
count_of_major_prod_obs = fil_datatset_3_major_prod.score.count()
count_of_other_prod_obs = fil_datatset_3_others.score.count()

#Computing the proportion of movies which are produced by Major Production Houses from the entire
sample dataset
#(This will be our 'A Priori' probablity of a movie being produced by a Major Production House):
perc_major_prod = count_of_major_prod_obs/(count_of_major_prod_obs + count_of_other_prod_obs)

#Creating an iteration of 1000 samples from the Population, to compute the p_value of scores betwe
en movies produced by
#a 'Major Production House' and those produced by other Production Houses  being statistically the
same or different:
# We will take our 'alpha' threshold to be 0.05 or 5% significance level

mean_diffs = []
for i in range(1000):
    major_prod_house = []
    other_prod_house = []
    for val in fil_datatset_3_3.score:
        rand_val = np.random.random()

        if rand_val <= perc_major_prod:
            major_prod_house.append(val)
```

```
        else:
            other_prod_house.append(val)
    mean_diff = np.mean(major_prod_house) - np.mean(other_prod_house)
    mean_diffs.append(mean_diff)


#Printing the histogram of Sample means between scores of 'Major Prod Houses' v.s. 'Other Prod Hou
ses' movies (Chck for normality):
import matplotlib.pyplot as plt
plt.hist(mean_diffs)
plt.show()

#Computing the p_value:
tot = 0
for mean_val in mean_diffs:
    if mean_val > 0.1:
        tot +=1

p_val = tot/1000
print("The p_value for the movie ratings being statistically significantly diff for Major Prod Hou
ses v.s. Other Prod Houses is")
print(p_val)
```

```
0.5780821917808219
```




```
The p_value for the movie ratings being statistically significantly diff for Major Prod Houses
v.s. Other Prod Houses is
0.021
```

Other normalization and scaling measures for the feature space:

Polynomial scaling to remove skewness in the dataset:

Shortcomings:

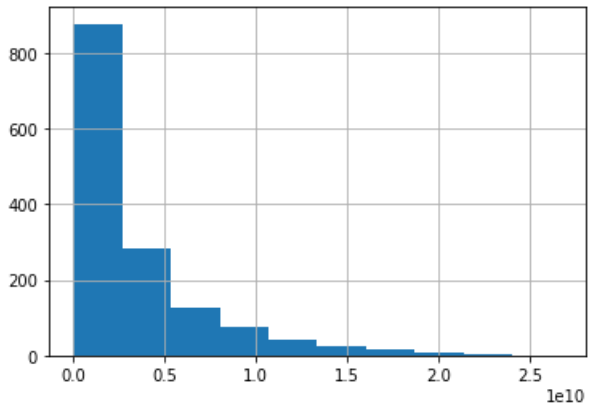
Polynomial scaling between 2 variables, where 1 of them is categorical may lead to inaccurate model predictions, as the categories will inherit a fales sense of 'order or sequence', while there may not exist any, in reality. 'Target encoding' is chosen for encoding categories as this allows the categories to be weighed per mean target value (i.e. IMDB score) corresponding to that category.

In [43]:

```
fil_datatset_4['comp_cat'] = fil_datatset_4.company.astype("category").cat.codes

fil_datatset_4.head()

#Checking for interaction between and 'comp_cat' & 'budget' using Polynomial Feature engineering:
from sklearn.preprocessing import PolynomialFeatures

poly_1 = PolynomialFeatures(degree = 2)
features = ['budget', 'comp_cat']
poly_comp_budgets = poly_1.fit_transform(fil_datatset_4[features])
print(poly_comp_budgets[:5])

new_df = pd.DataFrame(poly_comp_budgets, columns = poly_1.get_feature_names(input_features = featur
es))
```

```
new_df['score'] = fil_datatset_4['score']

print(new_df.shape)
new_df.head()

print(new_df['budget comp_cat'].hist())
```

```
[[1.0000e+00 3.0000e+08 8.9000e+01 9.0000e+16 2.6700e+10 7.9210e+03]
 [1.0000e+00 2.5800e+08 1.7000e+01 6.6564e+16 4.3860e+09 2.8900e+02]
 [1.0000e+00 2.5000e+08 8.9000e+01 6.2500e+16 2.2250e+10 7.9210e+03]
 [1.0000e+00 2.5000e+08 9.0000e+01 6.2500e+16 2.2500e+10 8.1000e+03]
 [1.0000e+00 2.4500e+08 5.3000e+01 6.0025e+16 1.2985e+10 2.8090e+03]]
(1460, 7)
AxesSubplot(0.125,0.125;0.775x0.755)
```



Next Steps:

From our EDA and hypothesis testing of the 2 laid out hypothesis, we found out that the budget of a movie has a significant impact on the IMDB score of the movie, with higher budget movies (typically with budgets > $80 Million) having a better IMDB score than lower budget movies. Our 2nd hypothesis test got us to the conclusion that movies produced by 'Major Production Houses' are Not significantly different in terms of IMDB ratings, as compared to other 'Smaller Production Houses'.

There could be multiple reasons for this result:

1. Movies produced by 'Major Production Houses' typically have higher number of votes cast which determines the IMDB score, due to substantial Marketing efforts to show the movie to a wider demographic audience and geographical regions. This results in viewers providing their ratings, thereby having an effect of normalizing the score. One wy to cater to this issue is to scale the IMDB score by the number of votes cast, to make it a fair comparison.
2. Movies produced by 'Major Production Houses' typically have 'higher expactation' from movie-goers especially if the movie is part of a series, such as the Marvel franshise or Bourne Series.

Another note to take here is the interaction of 'production houses' with 'directors' and 'budgets'. It is typically found that movies produced by certain production houses typically partner with a certain set of directors and typically have budgets exceeding $80 Million.

It would be wise to perform a Multi-Collinearity test here, to measure the dependency of 1 feature with another and eventually eliminate features which have a high correlation / dependnec on other features in the dataset.

Once the fetaures are engineered and made independent, having normally distributed residuals, as we've perform here, a simple regression model can be run to predict the IMDB score of newer movies which haven't been released yet. Similarly, we can solve a classification problem, by utilizing these engineered features to classify if the movie is predicted to perform well (above a certain IMDB score) or have a tendency to be un-popular among movie-goers (below a certain IMDB score).