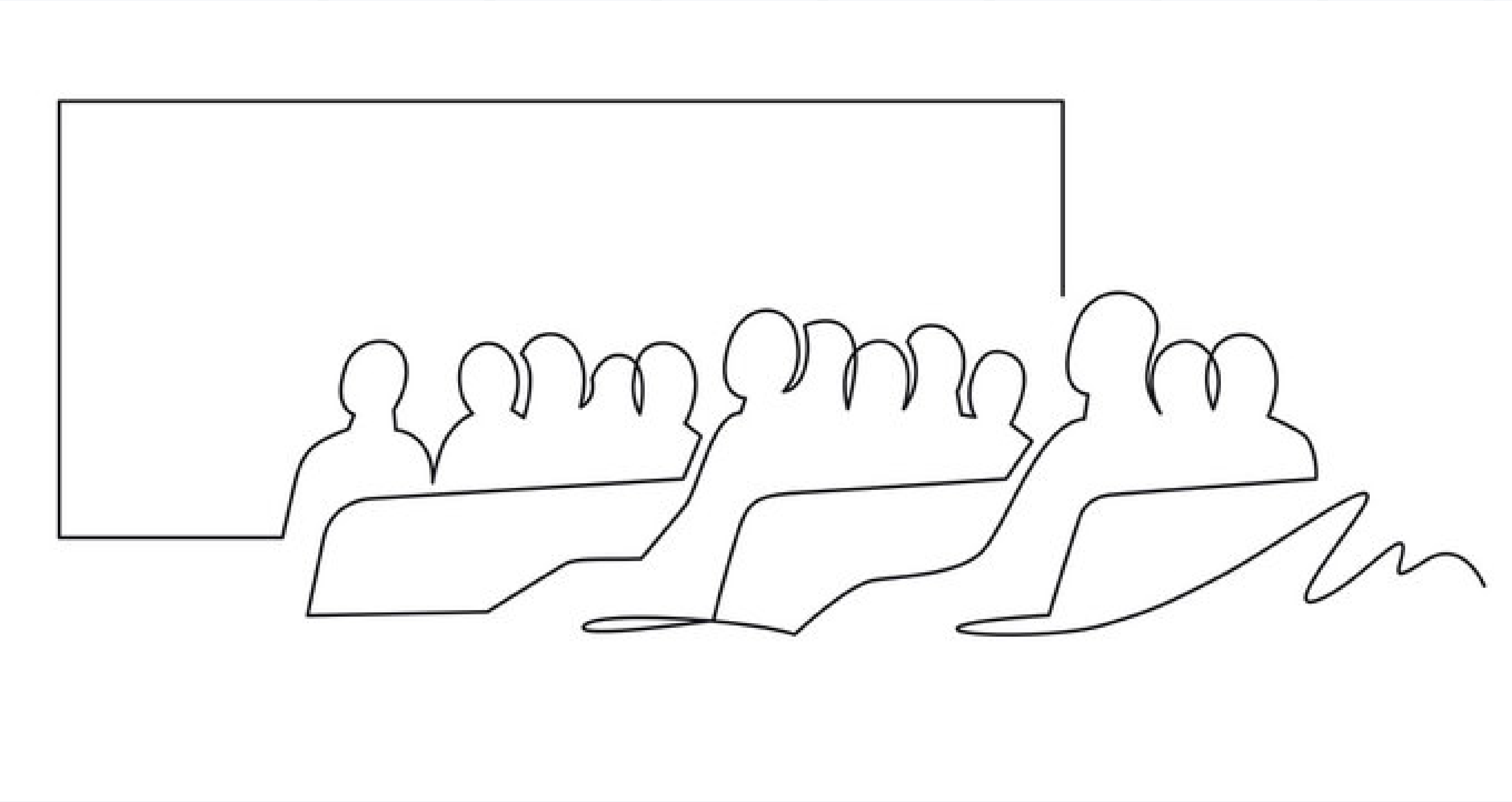
HW4

APPLY LINEAR REGRESSION TO PREDICT THE NUMBER OF CRITICAL REVIEWS ON IMDB



Dhanabordee Mekintharanggur
6238077121

Step 1 - Loading the Data

The data is downloaded as a .csv file format and is placed in the same directory as the .ipynb file

The csv is read using pd.read_csv to read the data in the dataframe format for further exploration and preprocessing

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
0	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0
1	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0
2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0
3	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0
4	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN
...	...	...	...	...	...	...	...	...
5038	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	NaN
5039	NaN	43.0	43.0	NaN	319.0	Valorie Curry	841.0	NaN
5040	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	NaN
5041	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	10443.0
5042	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	85222.0

5043 rows × 27 columns

Notice that there are NaN values present in multiple columns. This has to be dealt with in the next step.

Step 2 - Exploring and Cleaning the Data

There are originally 5043 rows of data

Multiple columns contain null values

The "num_critic_for_reviews" column, the y-axis that we aimed to predict using regression has a total of 50 null values, and the column cannot be removed considering the missing values make up less than 30%. The "num_critic_for_reviews" doesn't have a readily reliable metric to be used for an approximation of value either. The best option is to delete all the 50 rows that contain the missing values for this column.

```
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
...
 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
```

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

Step 2 - Exploring and Cleaning the Data

Regarding the other features, none has over 30% missing values either. Replacing the missing values by mean or median will not make sense as these values vary greatly among the movies. So, no column should be removed. Instead, the rows that contain missing values in any of the columns shall be specifically removed.

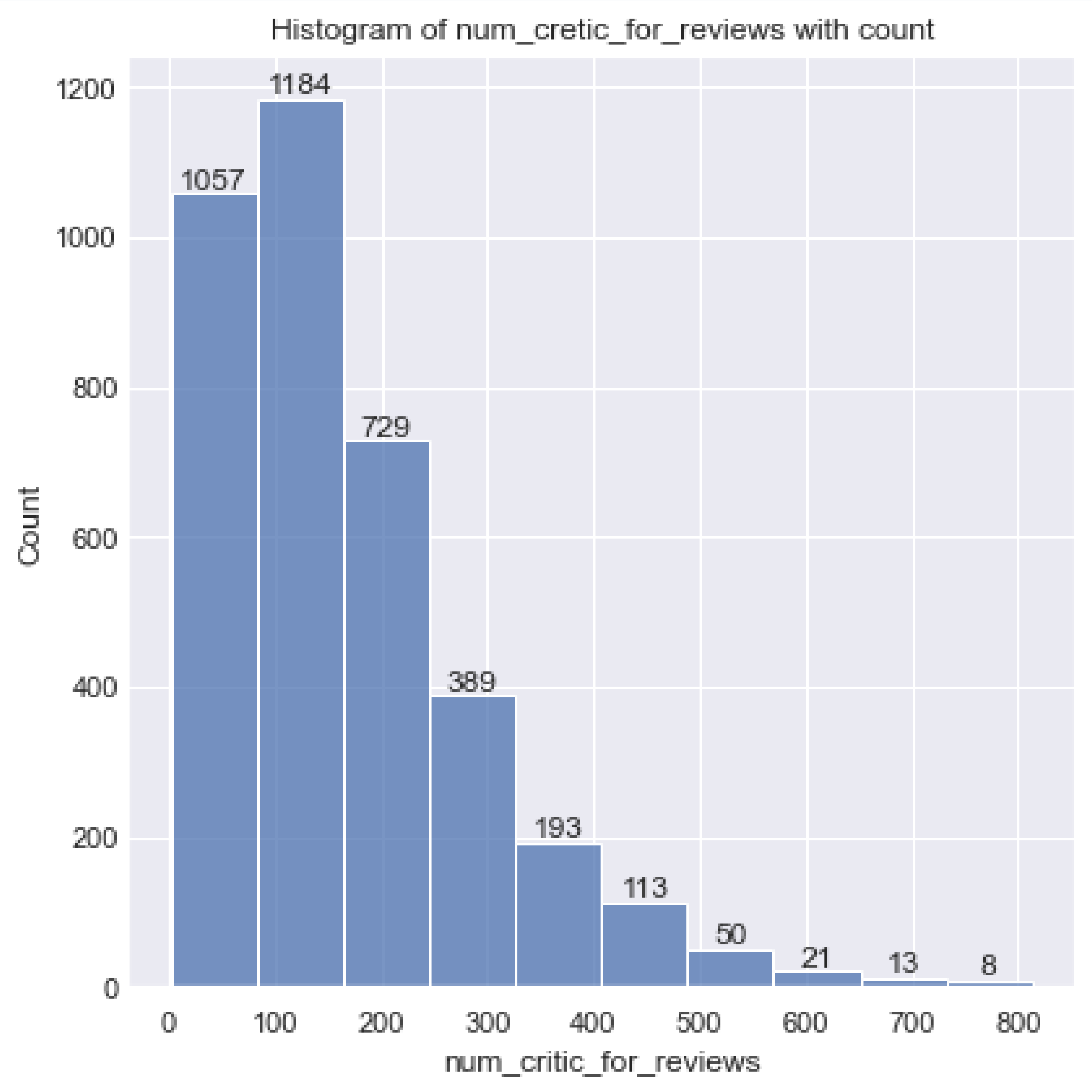
After all the rows containing the null values in all the features are removed, there are a total of 3757 rows remaining, making up 74.50% of the original dataset.

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

```
The remaining rows make up 0.7449930596866944 of the original dataset
```

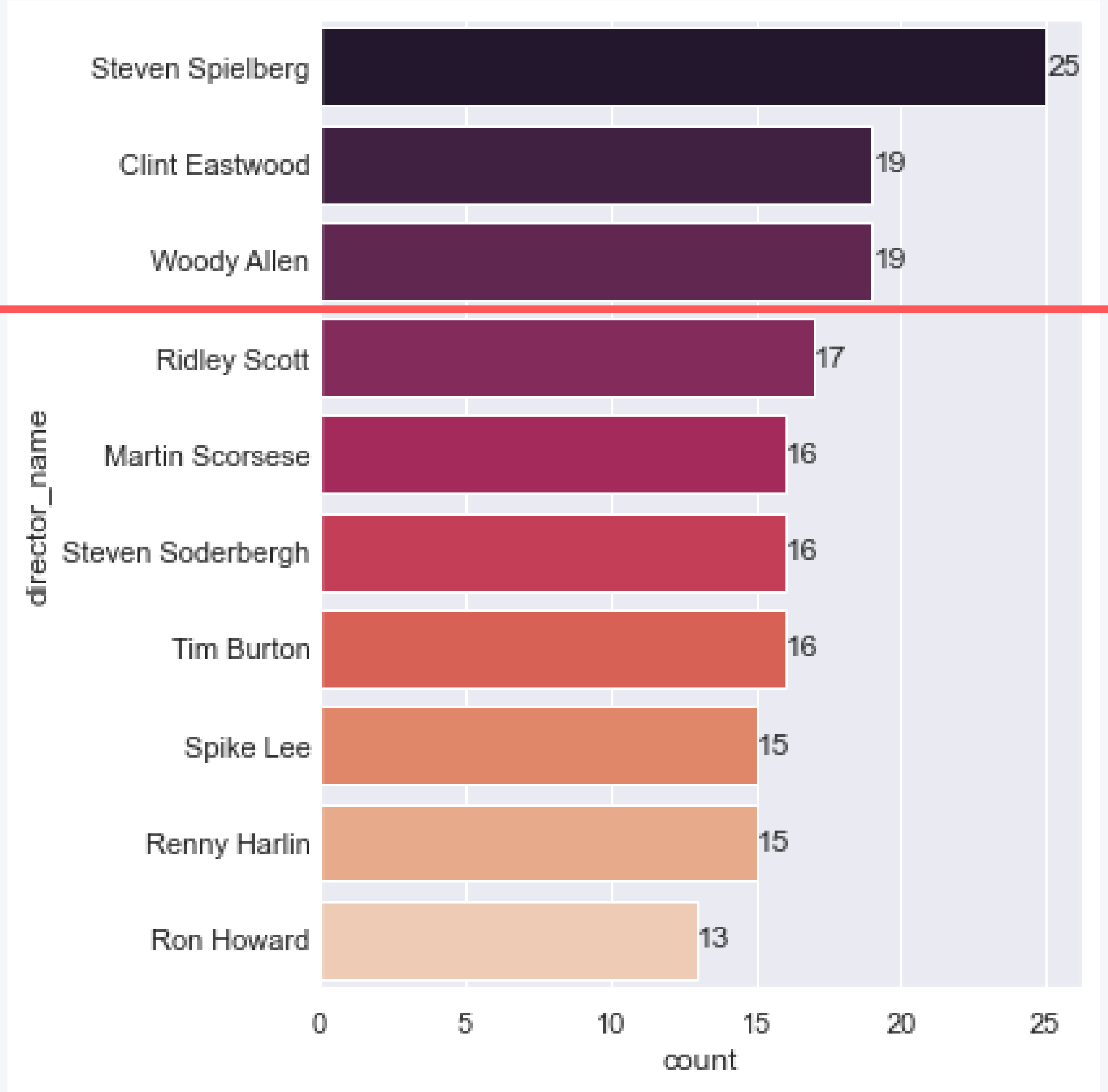
Step 2 - Exploring and Cleaning the Data

Visualizing the histogram of num_critic_for_reviews, it can be seen that the number of critics is skewed towards the right in this dataset with the range 0-300 being the most common



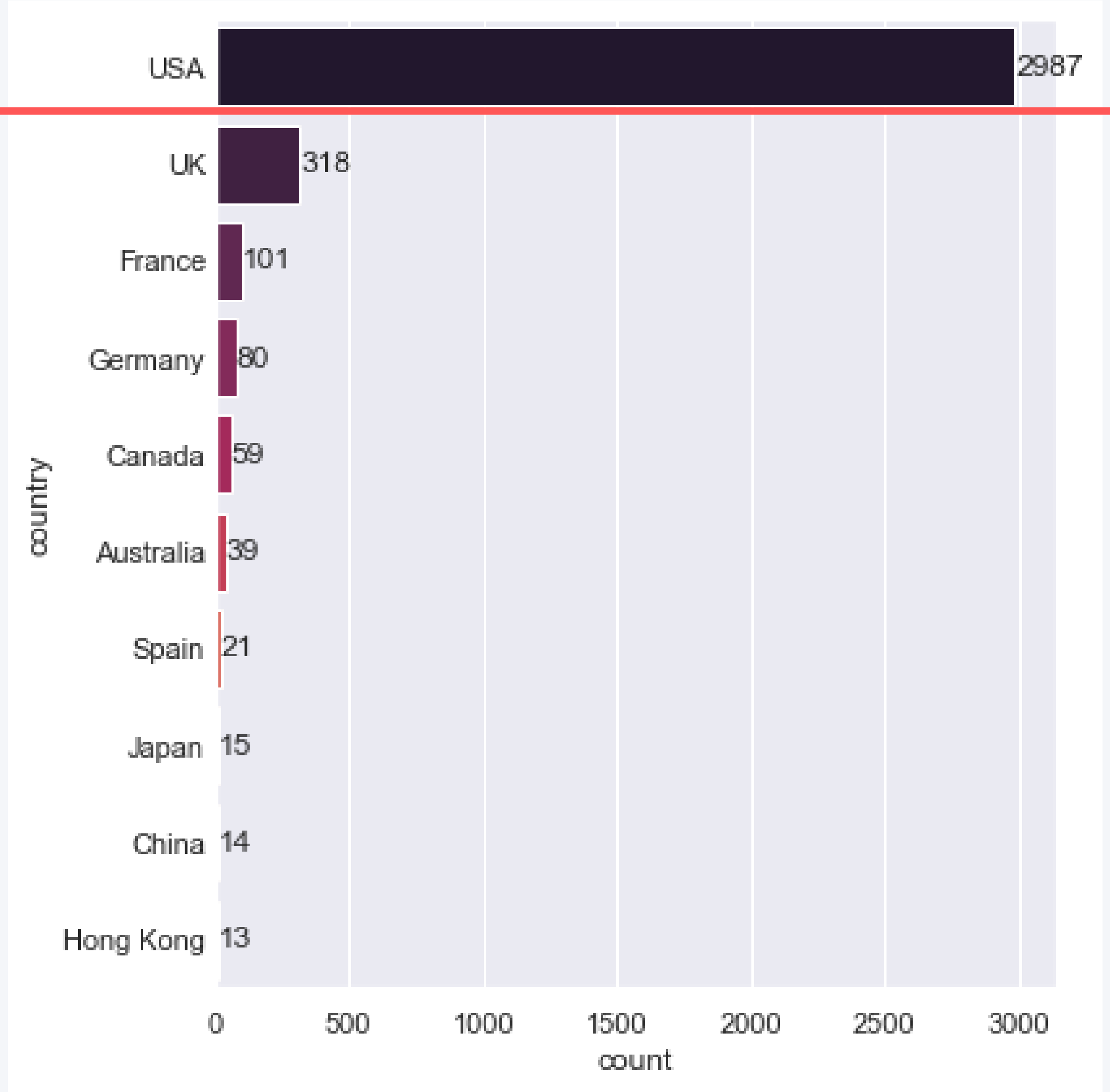
Step 2 - Exploring and Cleaning the Data

Since the dataset contains many columns that are not numerical, I explored the possibility of converting them into numerical values for further interpretation. To do so, start by visualizing the interesting columns' histogram to help identify the values that can be used as new columns



Cut-off: 19

The cut-off can be set for the top 4 directors who made the most movie in this list. Other directors will be included to the director:others column

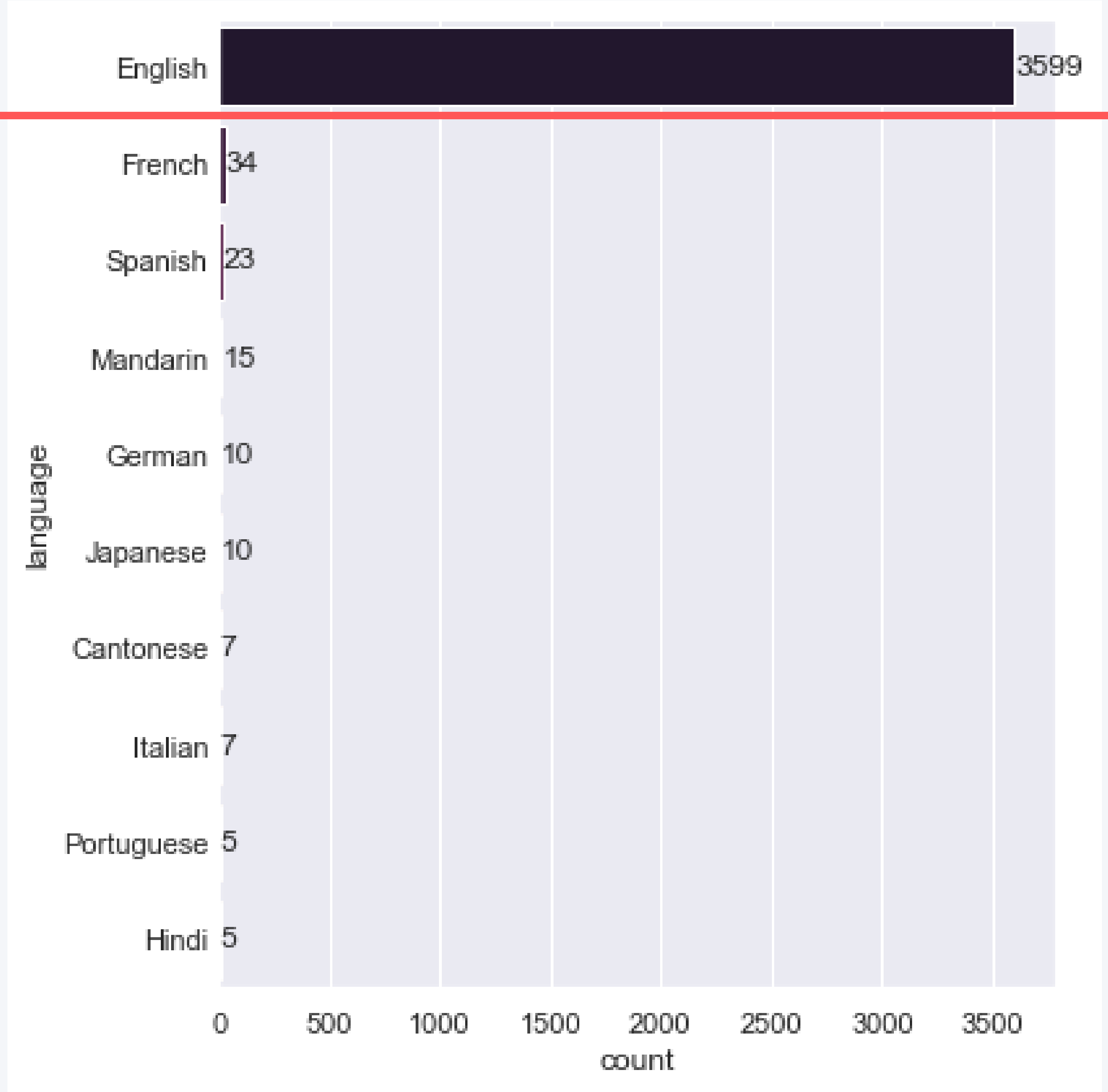


Cut-off: 2900

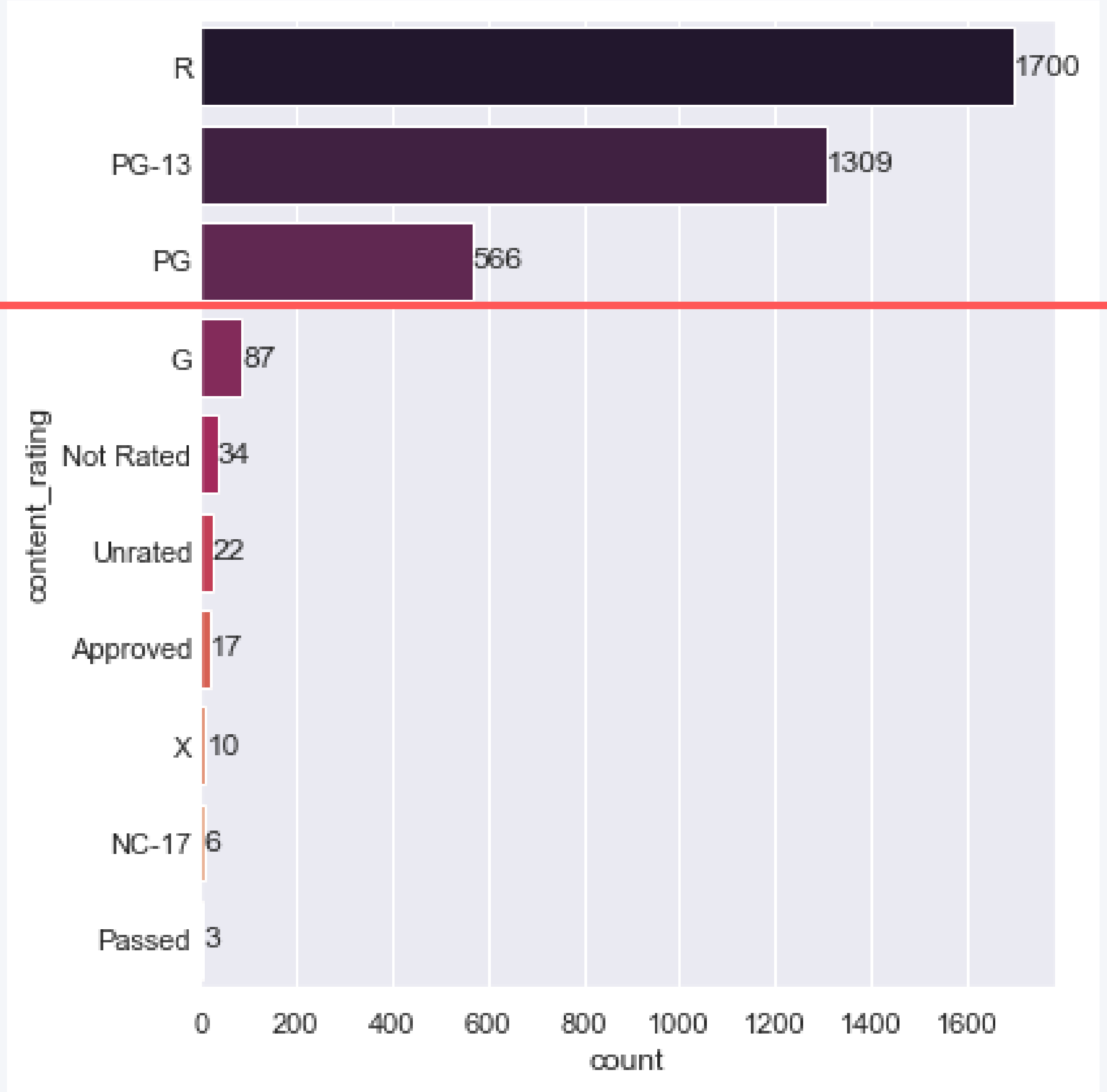
The USA dominates other countries in this list, so there can be 2 columns created, country:USA and country:others.

Step 2 - Exploring and Cleaning the Data

Since the dataset contains many columns that are not numerical, I explored the possibility of converting them into numerical values for further interpretation. To do so, start by visualizing the interesting columns' histogram to help identify the values that can be used as new columns



Cut-off: 3500



Cut-off: 500

The English-speaking films dominate other languages in this list, so there can be 2 columns created, language:English and language:others.

The top 3 most common content ratings can be used as the new categories and the other can be included to the content_rating:others column

Step 2 - Exploring and Cleaning the Data

To separate the most frequent values of the previously-mentioned columns into new columns, the OneHotEncoder from sklearn can be utilized and the new numeric columns generated are as follow:

director_name

director_name:Clint Eastwood
director_name:Steven Spielberg
director_name:Woody Allen
director_name:others

country

country:USA
country:others

language

language:English
language:others

content_rating

content_rating:PG
content_rating:PG-13
content_rating:R
content_rating:others

Step 2 - Exploring and Cleaning the Data

The other columns including: 'movie_title', 'actor_2_name', 'actor_3_name', 'actor_1_name', 'plot_keywords', 'movie_imdb_link' are then dropped. The reasons for dropping these columns are as follow:

'movie_title': Every new movie will have a unique title. Using this feature for developing the model will not be useful

'actor_[1, 2, 3]_name': There are too much variation and too many actors to effectively create new columns specifically for them and have them potentially be significant

'**plot_keywords**': Plot keywords are often unique, and includes the name of the movie. It is not helpful form groups based on such a variable value as such

'**movie_imdb_link**': For the same reason as dropping 'movie_title', each entry is unique and will not be helpful in modeling

```
RangeIndex: 3757 entries, 0 to 3756
Data columns (total 32 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   director_name              3757 non-null   object
 1   num_critic_for_reviews     3757 non-null   float64
 2   duration                   3757 non-null   float64
 3   director_facebook_likes    3757 non-null   float64
 4   actor_3_facebook_likes     3757 non-null   float64
 5   actor_1_facebook_likes     3757 non-null   float64
 6   gross                      3757 non-null   float64
 7   num_voted_users            3757 non-null   int64
 8   cast_total_facebook_likes  3757 non-null   int64
 9   facenumber_in_poster       3757 non-null   float64
 10  num_user_for_reviews       3757 non-null   float64
 11  language                   3757 non-null   object
 12  country                    3757 non-null   object
 13  content_rating             3757 non-null   object
 14  budget                     3757 non-null   float64
 15  title_year                 3757 non-null   float64
 16  actor_2_facebook_likes     3757 non-null   float64
 17  imdb_score                 3757 non-null   float64
 18  aspect_ratio               3757 non-null   float64
 19  movie_facebook_likes       3757 non-null   int64
...
 30  content_rating:R           3757 non-null   float64
 31  content_rating:others      3757 non-null   float64
dtypes: float64(25), int64(3), object(4)
memory usage: 939.4+ KB
```

These are the resulting columns after dropping the unused features and adding the OneHot columns

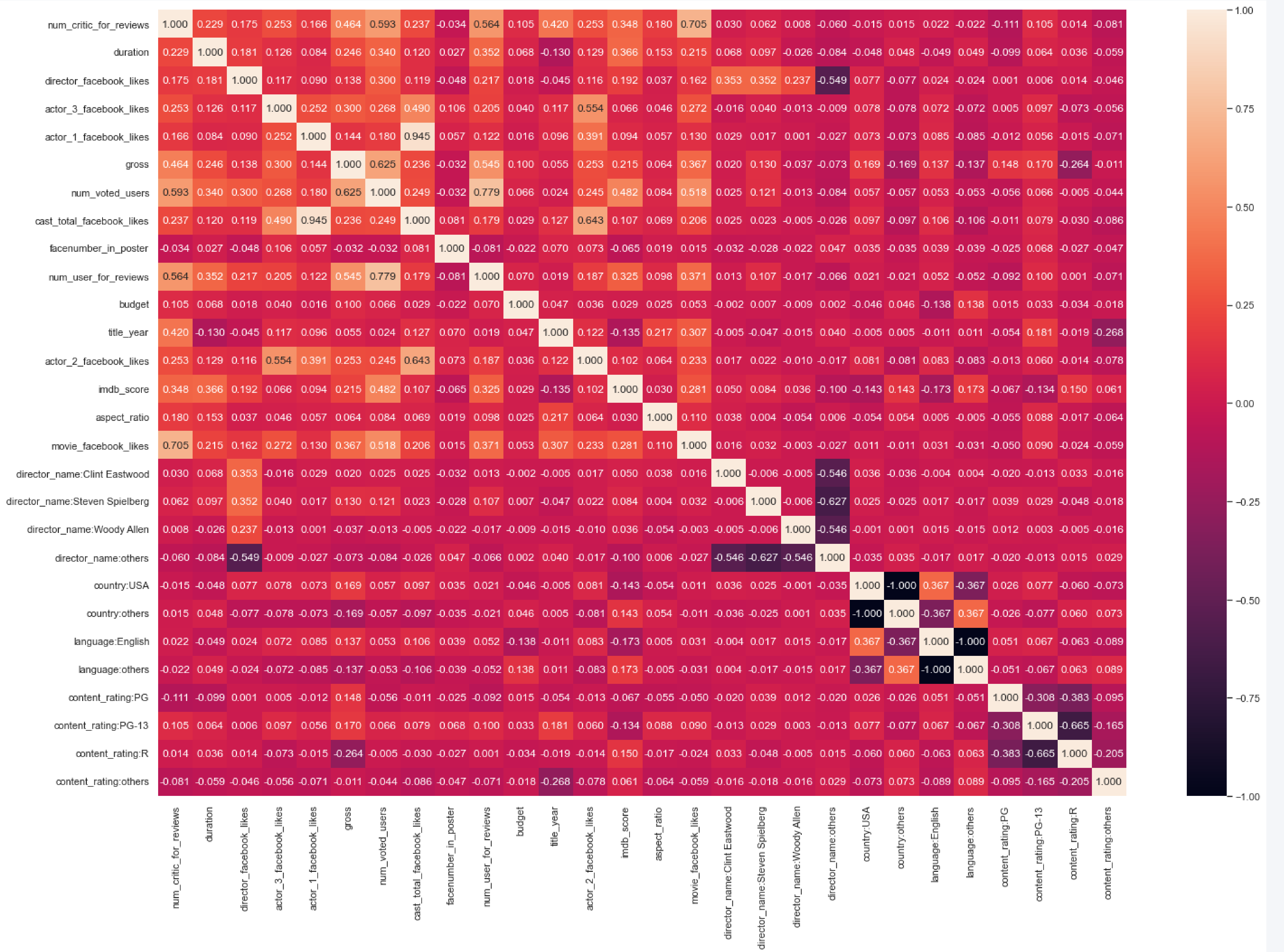
Step 2 - Exploring and Cleaning the Data

The heatmap reveals the correlation between each features.

To **remove the noisy values**, identify the X features with a correlation coefficient of > 0.5 or <-0.5 with the Y feature

The top 3 features that have the highest correlation with num_critic_for_reviews are selected: 'movie_facebook_likes', 'num_voted_users', and 'num_user_for_reviews'

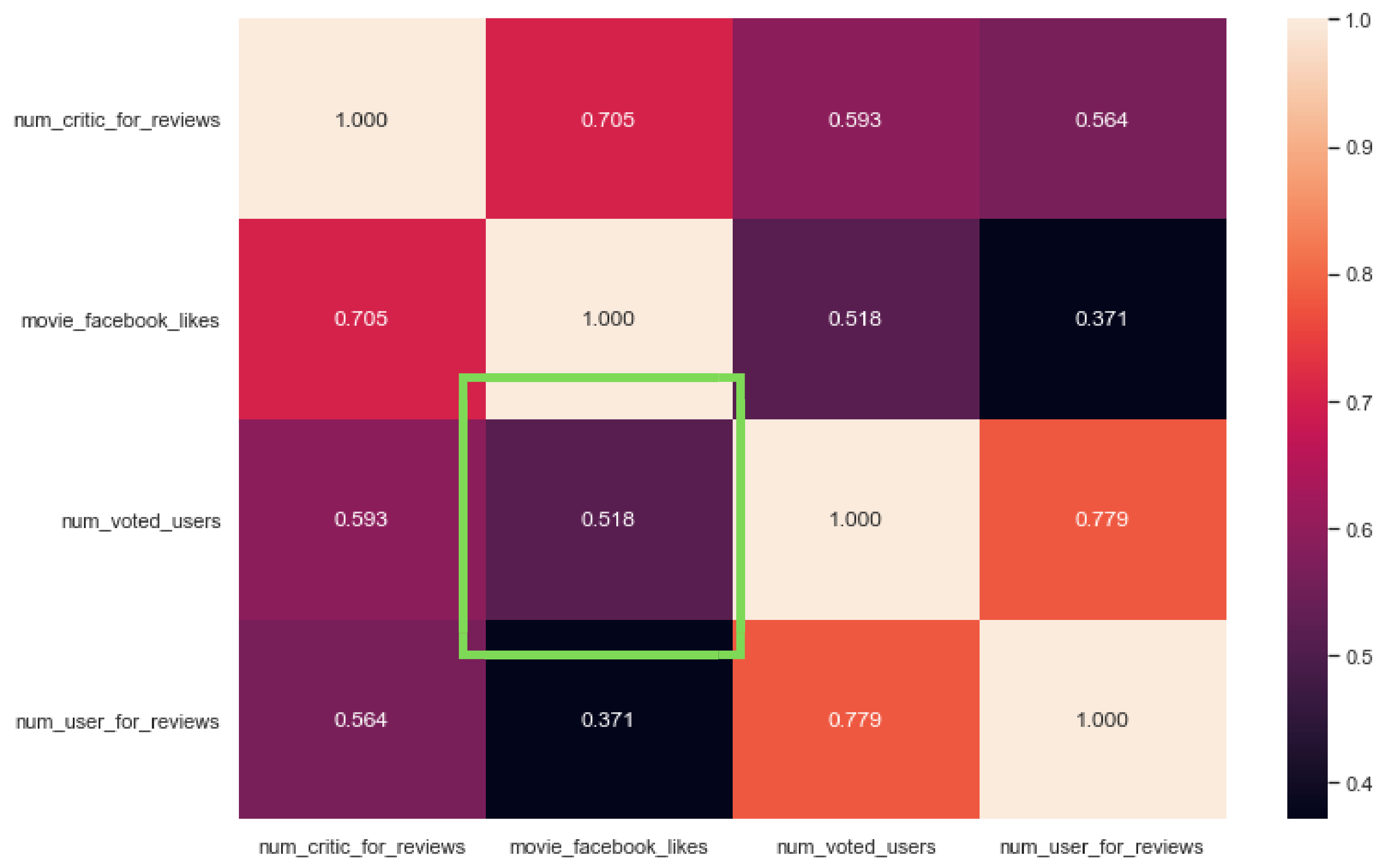
These features are good candidates to be used to create a regression model to predict 'num_critic_for_reviews' later on



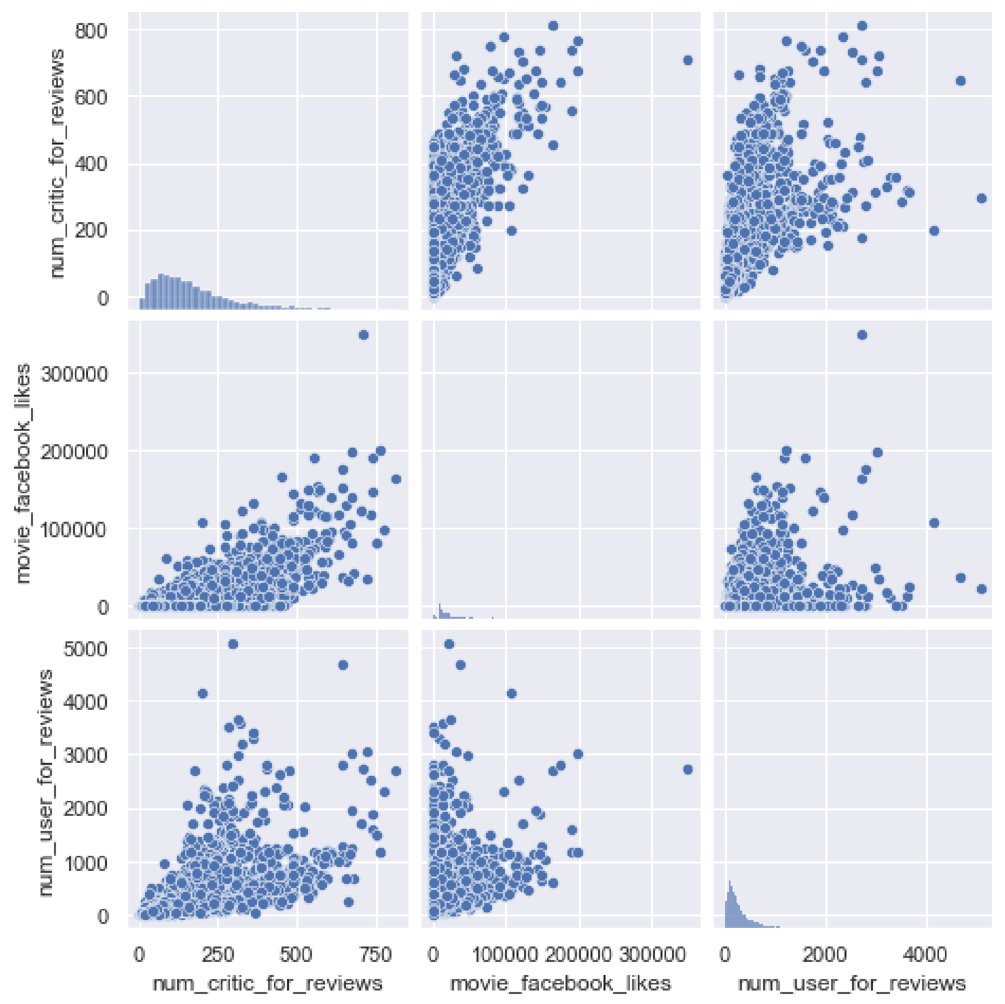
Step 2 - Exploring and Cleaning the Data

Next, the collinearity is removed. Notice that 'num_voted_users' and 'movie_facebook_likes' has a higher correlation than 0.5.

Thus, 'num_voted_users' is dropped to prevent collinearity since 'movie_facebook_likes' has a higher correlation with 'num_critic_for_reviews'



The resulting pair plot shows the scatterplot graph between each feature and shows a slightly linear trend, signifying that these values are suitable for developing the linear regression model



Step 2 - Exploring and Cleaning the Data

Before developing the model, apply the MinMaxScaler.

This scaling is performed to normalize the values of the features, as linear regression models are sensitive to the scale of the features.

Lastly, separate the model into train and test data

```
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y,test_size=0.3,random_state=20,shuffle=True)
```

30% of the data will be randomly split to be used for evaluating the performance of the model, and the remaining 70% will be used for training the model

Since this is not a time-series data, the dataset can be shuffled

Step 3 - Data Model Training

Least squares linear regression is tested in this project.

The model is fitted with

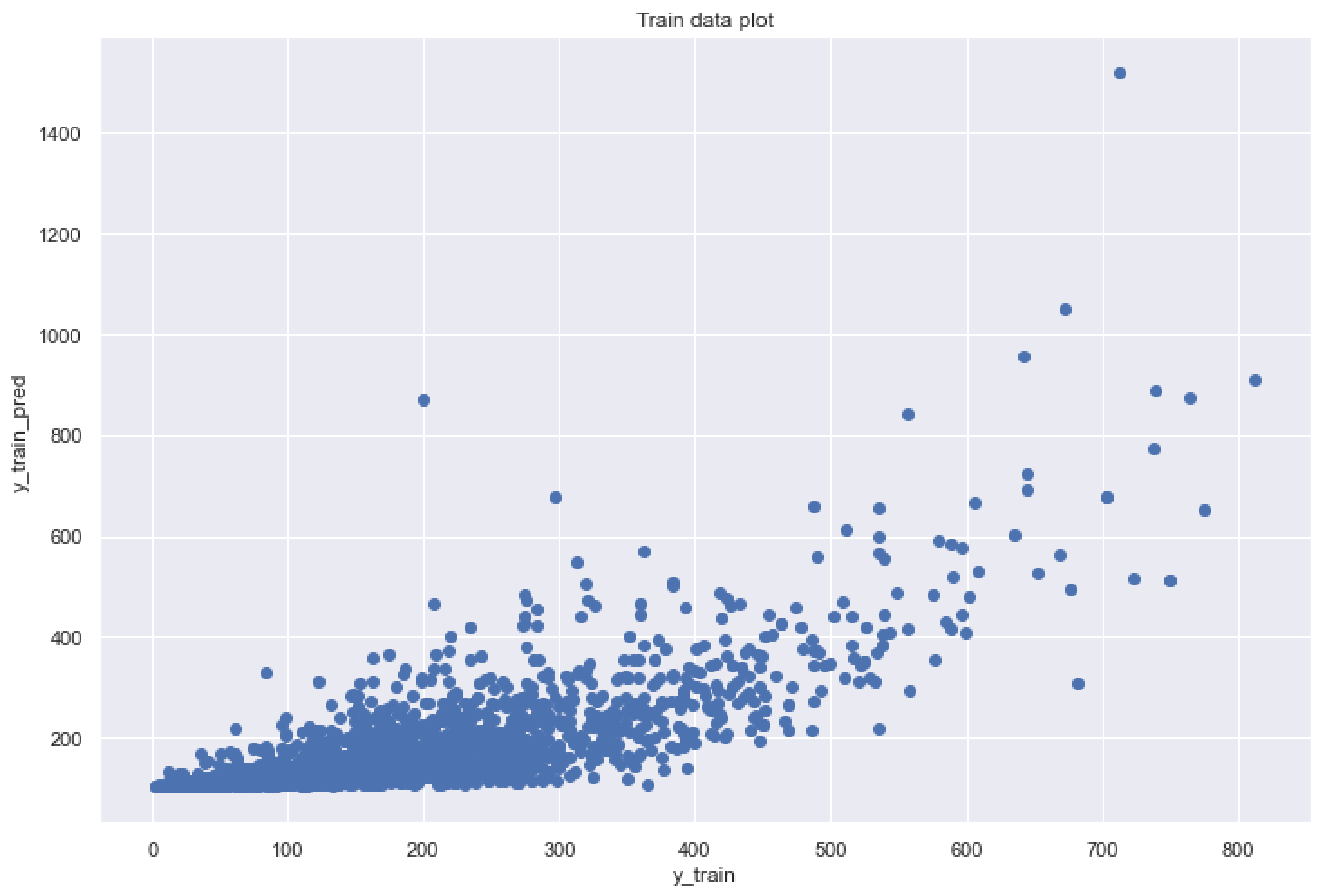
- x_train: data frame consisting of 'movie_facebook_likes' and 'num_user_for_reviews'
- y_train: train data frame consisting of 'num_critic_for_reviews'
- the default parameters are used in this case
- Displays the result of the model traning

```
------------------
Train
------------------
Train Intercept: 102.92803704397518
Train Coefficient: [1141.65249328  506.32856662]

Train r2 score: 0.5979657434270407

Train MAE score: 0.8113609836098132

Train MSE score: 6143.688921746239
```



Step 4 - Data Model Testing

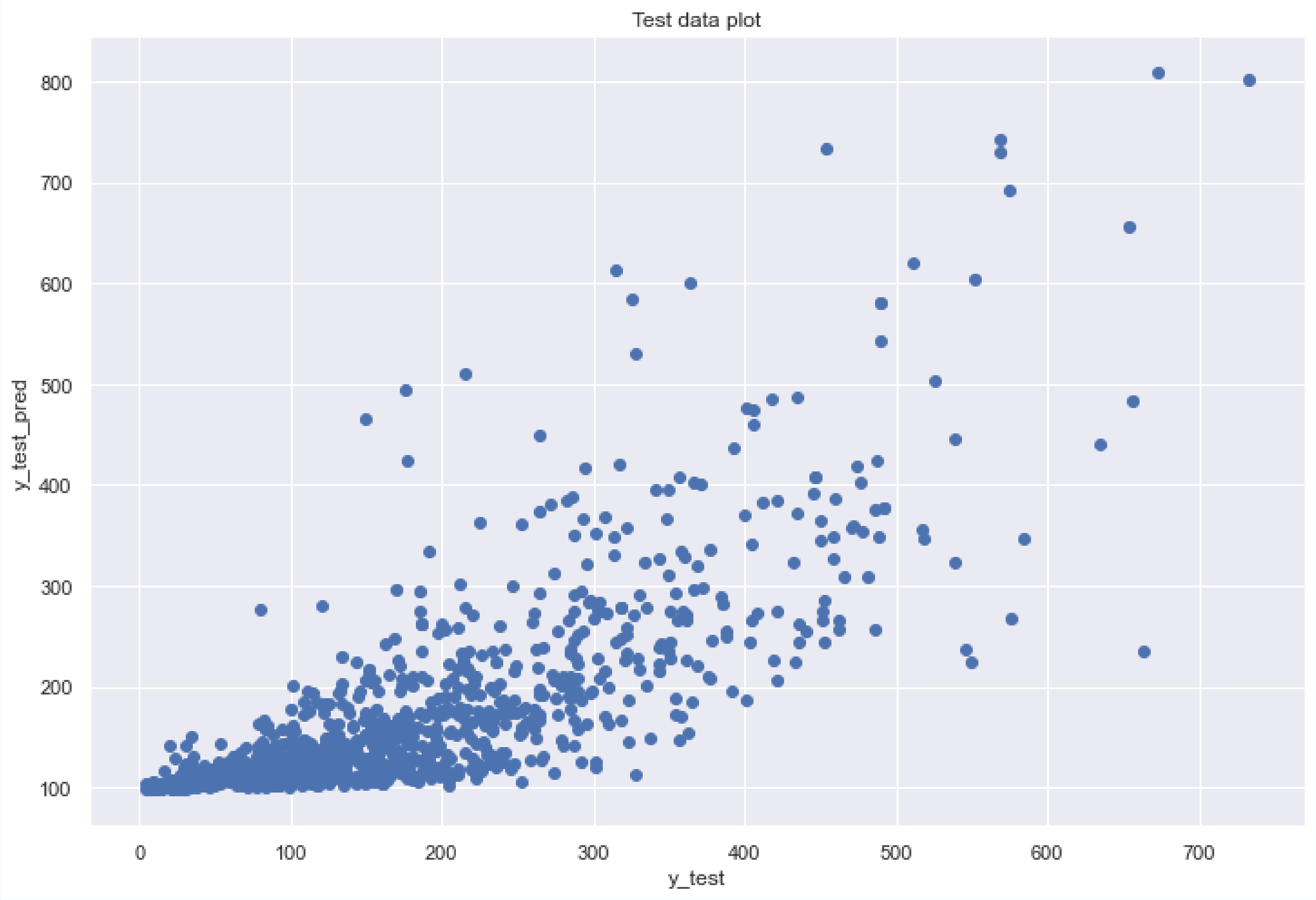
- The test data are then used to fit the model and make a prediction
- Displays the result of the model testing
- It appears that the model is able to perform better with the test data in regards to the r2, MSE and MAE scores

```
------------------
Test
------------------
Train Intercept: 96.98329410505347
Train Coefficient: [1195.26196807  600.18277279]

 Test r2 score: 0.6177409866117874

 Test MAE: 0.7487624582815583

 Test MSE: 5781.317384901998
```



Step 5 - Results Evaluation (Part 1)

Model Scores

- As seen during the data model testing, the model performs slightly better with the test data compared to the train data
- This could happen due to "luck" that the test data might have fit the model better, resulting in a higher r2, MAE, and MSE scores compared to the training data

Linear Regression Equation

- The linear regression equation is as follow:
- **y = 1141.65249328*x1 + 506.32856662*x2 + 102.92803704397518**
  - x1 and x2 values are the input variables for 'movie_facebook_likes' and 'num_user_for_reviews', respectively

Results

- As a social science project, the r2 score of 0.6177 is considered as acceptable

Improvement

- To improve the performance of the prediction, it maybe necessary to obtain more features that are more closely related to the 'num_critic_for_reviews' so that it would result in a higher correlation and eventually a better prediction
- Other models may also be considered such as logistic regression or neural network

Step 5 - Exploration of a Model using Categorized the Data

Categorical data ('title_year')

- I try looking into other features that could be useful in categorizing the data into groups to be used with the SGD Regression
- The ColumnSelector, MinMaxScaler, OneHotEncoder, and FeatureUnion are used to create pipes for preparing the data
- The feature I have selected is the 'title_year'
- Although 'title_year' has a correlation with 'num_critic_for_reviews' of 0.42, which is lower than the suggested 0.5 in the lecture slides, I believe it is worth considering to be used as a category

```
Categorical features used:  ['title_year']
Final features used:  ['movie_facebook_likes', 'num_user_for_reviews', 1927.0, 1929.0, 1936.0, 1937.0, 1939.0, 1946.0,
1947.0, 1950.0, 1953.0, 1957.0, 1959.0, 1960.0, 1963.0, 1964.0, 1965.0, 1966.0, 1967.0, 1968.0, 1969.0, 1970.0, 1971.0,
1972.0, 1973.0, 1974.0, 1975.0, 1976.0, 1977.0, 1978.0, 1979.0, 1980.0, 1981.0, 1982.0, 1983.0, 1984.0, 1985.0, 1986.0,
1987.0, 1988.0, 1989.0, 1990.0, 1991.0, 1992.0, 1993.0, 1994.0, 1995.0, 1996.0, 1997.0, 1998.0, 1999.0, 2000.0, 2001.0,
2002.0, 2003.0, 2004.0, 2005.0, 2006.0, 2007.0, 2008.0, 2009.0, 2010.0, 2011.0, 2012.0, 2013.0, 2014.0, 2015.0, 2016.0]
```

Step 5 - Exploration of a Model using Categorized the Data

```
-----------------
Train
-----------------
Intercept: [28.96472306]
Coefficient: [ 6.74756489e+02  6.59968217e+02  4.42752298e+01 -2.31926038e-02
  1.81776859e+01  2.50866525e+01  2.69214810e+01  1.10613366e+01
  1.12236583e+01 -4.83603986e+00  8.02674974e+00  1.65787646e+01
  2.57964101e+01  2.88786558e+01  2.14905966e+01  5.02885705e+01
  1.52139201e+01  7.29666839e+00  1.89354282e+01  3.60697211e+00
  2.57219214e+01  1.66434795e+01  1.83335680e+01 -3.32813569e+01
  3.14060552e+01  6.22034923e+01 -5.63487119e+00  1.49000173e+01
  7.07405085e+00  1.33382972e+00  5.03149286e+01  4.77658044e+01
  6.19359263e+01  6.05824862e+01  1.34906188e+01  4.47001231e+01
  3.42345459e+01  2.01033244e+01  2.13411981e+01  3.13884311e+01
  1.65754444e+01  2.29835766e+01  2.18528499e+00  1.07641379e+01
 -3.20740205e+00 -1.20595723e+01  1.39661253e+01  8.15063853e+00
  2.18331495e+01  1.62846283e+01  2.19888494e+01  3.83087456e+01
...
```

```
------------------
Test
------------------

 Test r2 score: 0.7446870026740564

 Test MAE: 0.5386795648987315

 Test MSE: 3861.3751889033433
```


```
-----------------
Cross Validation
-----------------
0.73 accuracy with a standard deviation of 0.01
the intercept is 28.96
the coef of movie_facebook_likesis 674.76
the coef of num_user_for_reviewsis 659.97
the coef of 1927.0is 44.28
the coef of 1929.0is -0.02
the coef of 1936.0is 18.18
the coef of 1937.0is 25.09
the coef of 1939.0is 26.92
the coef of 1946.0is 11.06
the coef of 1947.0is 11.22
the coef of 1950.0is -4.84
the coef of 1953.0is 8.03
the coef of 1957.0is 16.58
the coef of 1959.0is 25.80
...
the coef of 2013.0is 161.87
the coef of 2014.0is 133.45
the coef of 2015.0is 116.90
the coef of 2016.0is 118.75
```

Step 5 - Exploration of a Model using Categorized the Data

Categorical data ('title_year')

- I try looking into other features that could be useful in categorizing the data into groups to be used with the SGD Regression
- The ColumnSelector, MinMaxScaler, OneHotEncoder, and FeatureUnion are used to create pipes for preparing the data
- The feature I have selected is the 'title_year'
- Although 'title_year' has a correlation with 'num_critic_for_reviews' of 0.42, which is lower than the suggested 0.5 in the lecture slides, I believe it is worth considering to be used as a category

```
Categorical features used:  ['decade']
Final features used:  ['movie_facebook_likes', 'num_user_for_reviews', '1920s ', '1930s ', '1940s ', '1950s ', '1960s ',
'1970s ', '1980s ', '1990s ', '2000s ', '2010s ']
```

Step 5 - Exploration of a Model using Categorized the Data

```
-----------------
Train
-----------------
Intercept: [7.03162478]
Coefficient: [689.13876079 638.61541549  48.26057762  63.92413969  29.43103925
  43.60647601  60.83561998  50.17269121  55.72193202  36.41694626
  93.60971228 167.69663183]

Train r2 score: 0.6944060470977949

Train MAE score: 0.6360597710911559

Train MSE score: 4669.935838308854
```

```
------------------
Test
------------------
Test r2 score: 0.7085297685984144

Test MAE: 0.5864210495853096

Test MSE: 4408.220230171704
```


```
-----------------
Cross Validation
-----------------
0.70 accuracy with a standard deviation of 0.02
the intercept is 7.03
the coef of movie_facebook_likesis 689.14
the coef of num_user_for_reviewsis 638.62
the coef of 1920s is 48.26
the coef of 1930s is 63.92
the coef of 1940s is 29.43
the coef of 1950s is 43.61
the coef of 1960s is 60.84
the coef of 1970s is 50.17
the coef of 1980s is 55.72
the coef of 1990s is 36.42
the coef of 2000s is 93.61
the coef of 2010s is 167.70
```

Step 5 - Results Evaluation (Part 2)

Categorical Values

- Using the ('title_year') as a categorical value shows a significant improvement to the test results in terms of the r2 score, MAE, and MSE as a whole.
- The train and test scores are very close to each other, suggesting that the model does not overfit
- However, to prevent this from possibly happening due to the number of features created from categorizing by 'title_year', I grouped the data into decades as a preemptive measure
- The r2, MAE, and MSE worsen compared to using all the 'title_year' values but is still a better result than simply doing the Least squares linear regression originally

Note

- This is not to say that the suggestion to only use x features that correlate highly with the y feature with correlation > 0.5 should be neglected, but is just to offer a perspective that using certain value for categories can improve the regression model's performance