HW7

APPLY DECISION TREE TO PREDICT IMDB_SCORE (LOW OR HIGH) 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	
0	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	76
1	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	30
2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	2
3	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	44
4	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	
...	...	...	...	...	...	...	...	
5038	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	
5039	NaN	43.0	43.0	NaN	319.0	Valorie Curry	841.0	
5040	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	
5041	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	
5042	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	

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

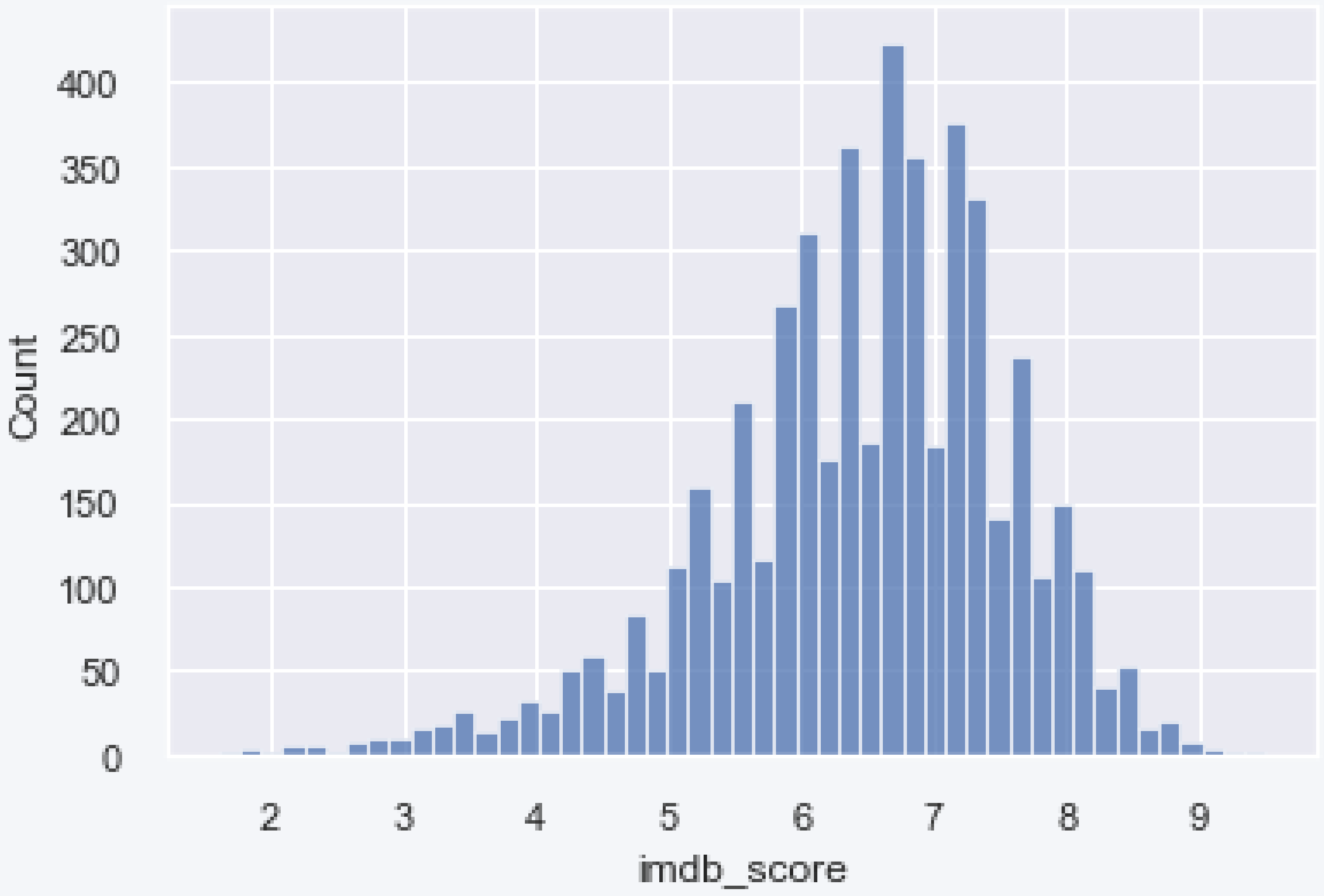
```
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   director_name              4939 non-null   object
 1   num_critic_for_reviews     4993 non-null   float64
 2   duration                   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
```

- There are originally 5043 rows of data
- Multiple columns contain null values
- Fortunately, the "imdb_score" column that I will be predicting has no empty rows.
- Since the non-numeric values can be categorized and then binary encoding them. Thus, I will not only select the numeric columns, but keeping all the columns instead.

Step 2 - Exploring and Cleaning the Data

	imdb_score
count	5043.000
mean	6.442
std	1.125
min	1.600
25%	5.800
50%	6.600
75%	7.200
max	9.500



Visualizing the histogram of "imdb_score", it can be seen that the values appear to be somewhat **normally distributed**

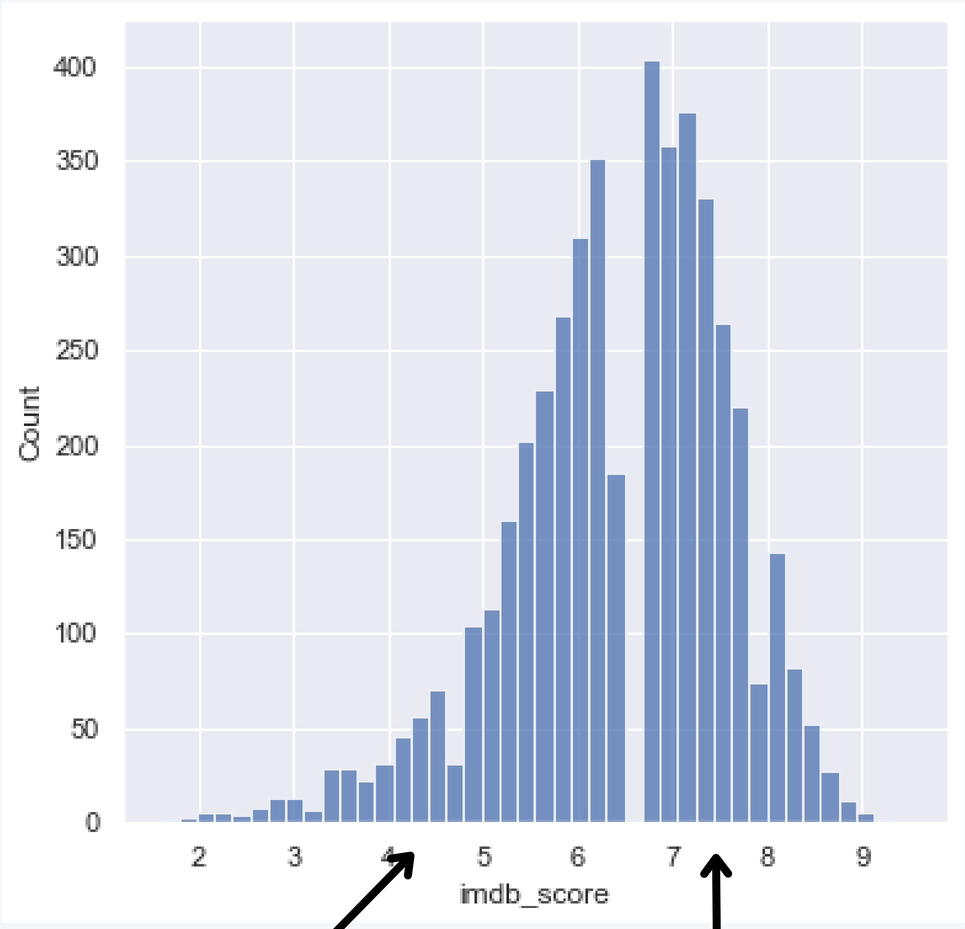
The mean imdb_score is at 6.442 points

Step 2 - Exploring and Cleaning the Data

Next, the imdb_score is categorized as a "high" or a "low".

Since there is practically no middle point to separate the high and low scores and in order to remove the ambiguity between films with close IMDb scores, the middle 10% percent of the data are removed.

The data between the 45-percentile and 55-percentile are dropped from the data frame, leaving 2 distinct sections to be categorized as a "high_score" represented by value of 1 in the high_imdb_score column, and a value of 0 in the same column for "low_score"



high_imdb_score = 0

high_imdb_score = 1

	high_imdb_score
0	1
1	1
2	1
3	1
4	1
...	...
5037	0
5038	1
5039	1
5040	0
5041	0

The new column is then added to the dataframe

the "movie_score_target" is a data frame as shown

Step 2 - Exploring and Cleaning the Data

```
gross                        17.529248
budget                        9.756098
aspect_ratio                  6.523895
title_year                    2.141582
director_facebook_likes       2.062265
num_critic_for_reviews        0.991473
actor_3_facebook_likes        0.456078
num_user_for_reviews          0.416419
duration                      0.297442
facenumber_in_poster          0.257783
actor_2_facebook_likes        0.257783
actor_1_facebook_likes        0.138806
cast_total_facebook_likes     0.000000
num_voted_users               0.000000
movie_facebook_likes          0.000000
imdb_score                    0.000000
dtype: float64
```

All the rows containing missing values are dropped as they make up only a small percentage of the whole data (and less then 30%)

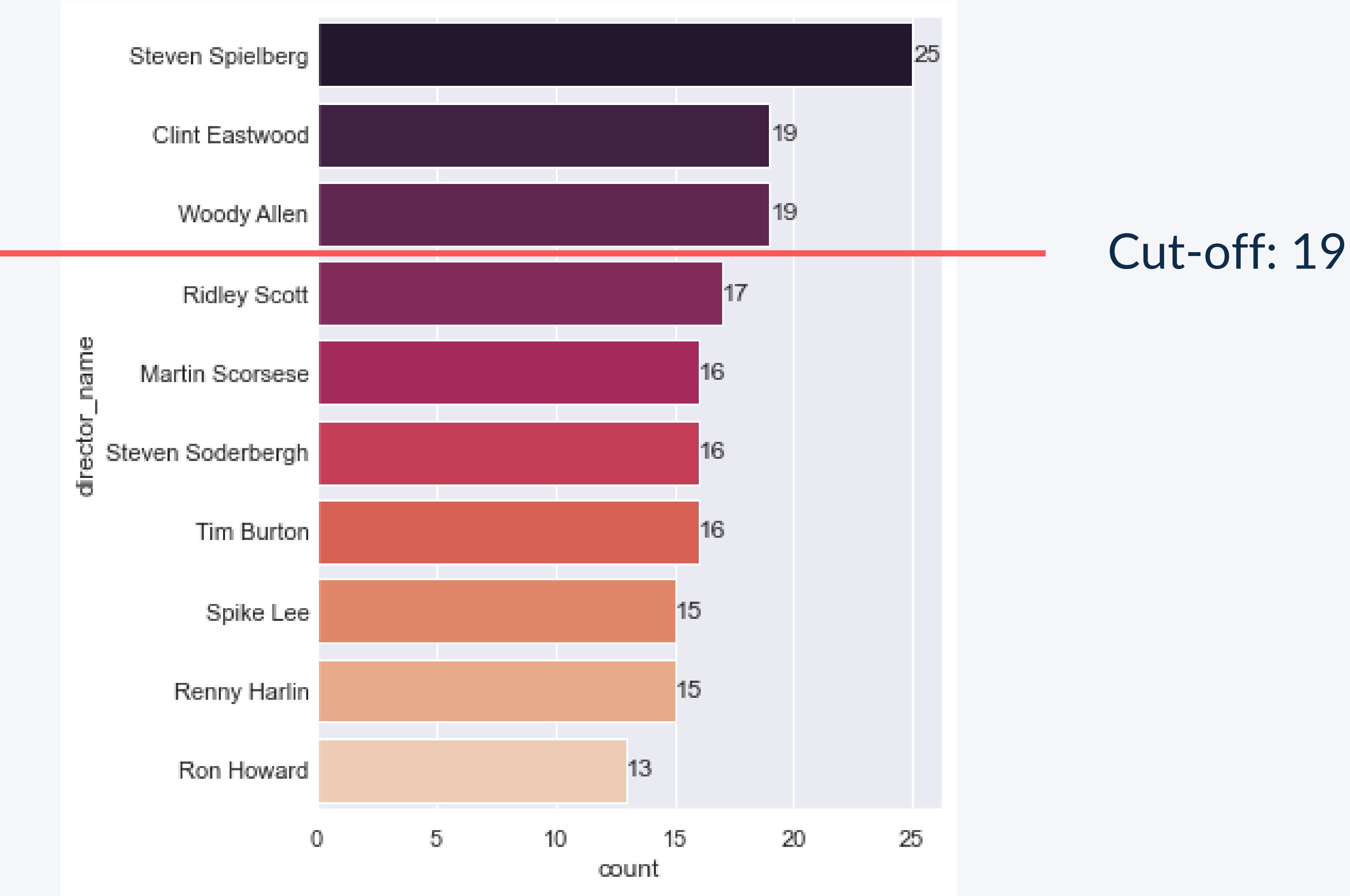
```
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   director_name              3441 non-null   object
 1   num_critic_for_reviews     3441 non-null   float64
 2   duration                   3441 non-null   float64
 3   director_facebook_likes    3441 non-null   float64
 4   actor_3_facebook_likes     3441 non-null   float64
 5   actor_2_name               3441 non-null   object
 6   actor_1_facebook_likes     3441 non-null   float64
 7   gross                      3441 non-null   float64
 8   genres                     3441 non-null   object
 9   actor_1_name               3441 non-null   object
 10  movie_title                3441 non-null   object
 11  num_voted_users            3441 non-null   int64
 12  cast_total_facebook_likes  3441 non-null   int64
 13  actor_3_name               3441 non-null   object
 14  facenumber_in_poster       3441 non-null   float64
 15  plot_keywords              3441 non-null   object
 16  movie_imdb_link            3441 non-null   object
 17  num_user_for_reviews       3441 non-null   float64
 18  language                   3441 non-null   object
 19  country                    3441 non-null   object
...
 26  movie_facebook_likes       3441 non-null   int64
 27  high_imdb_score            3441 non-null   int64
dtypes: float64(13), int64(4), object(11)
```

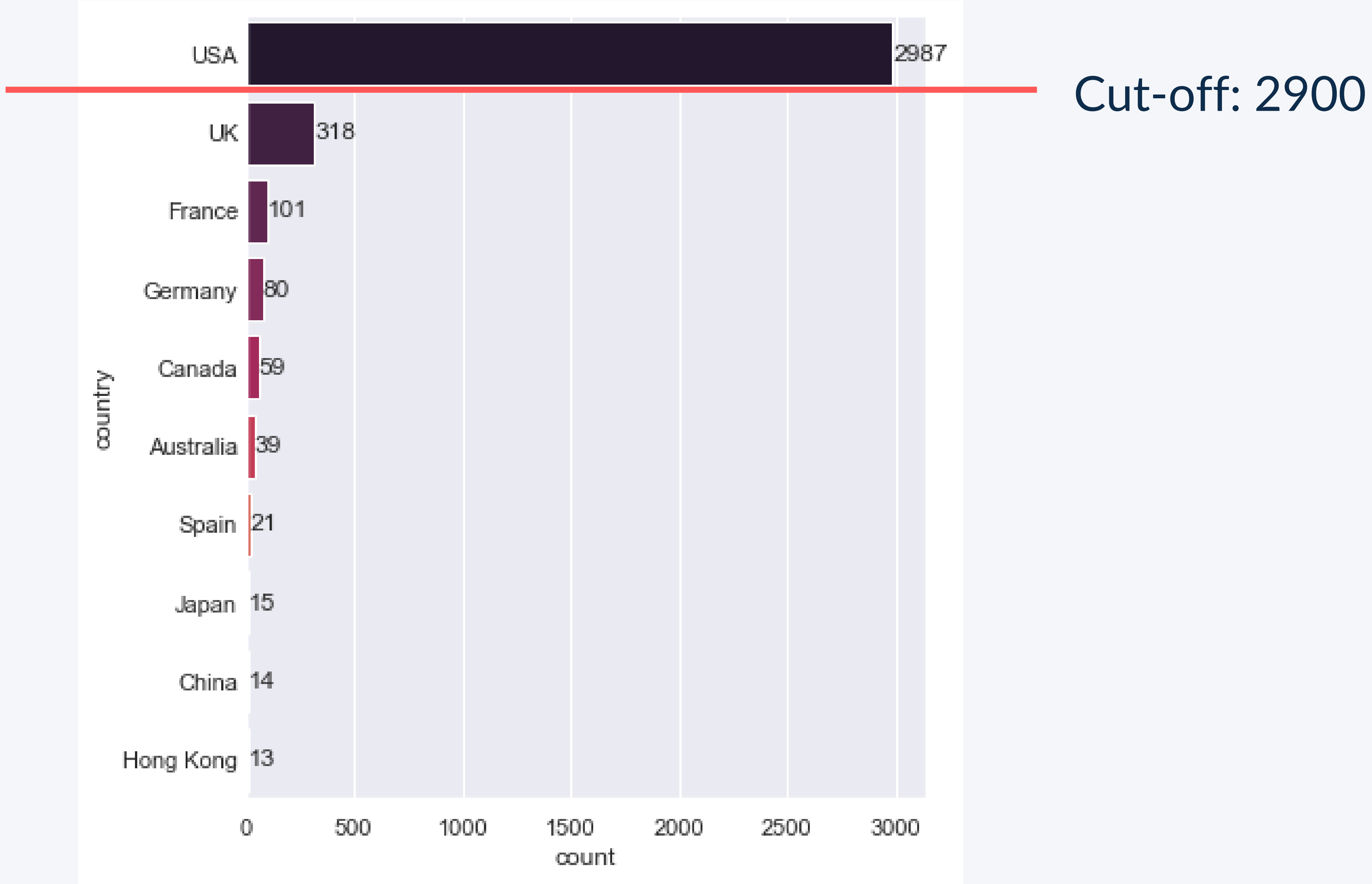
After dropping, there are a total of 3441 non-null numerical and string columns remaining

Step 3 - Preprocess the Data

Onehot encode categorical features

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





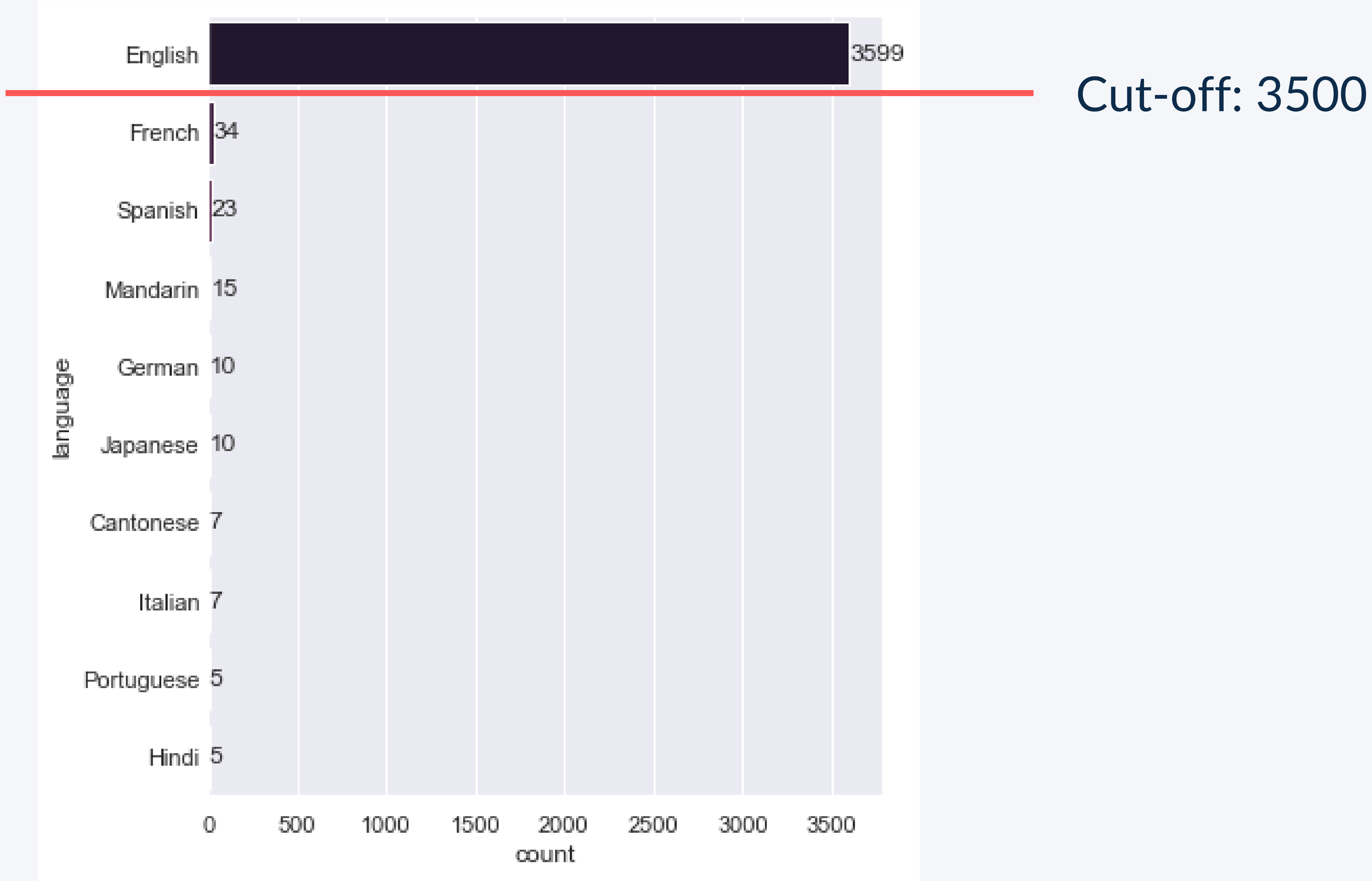
The cut-off can be set for the top 4 directors who made the most movie in this list. Other directors will be included to the director:others column

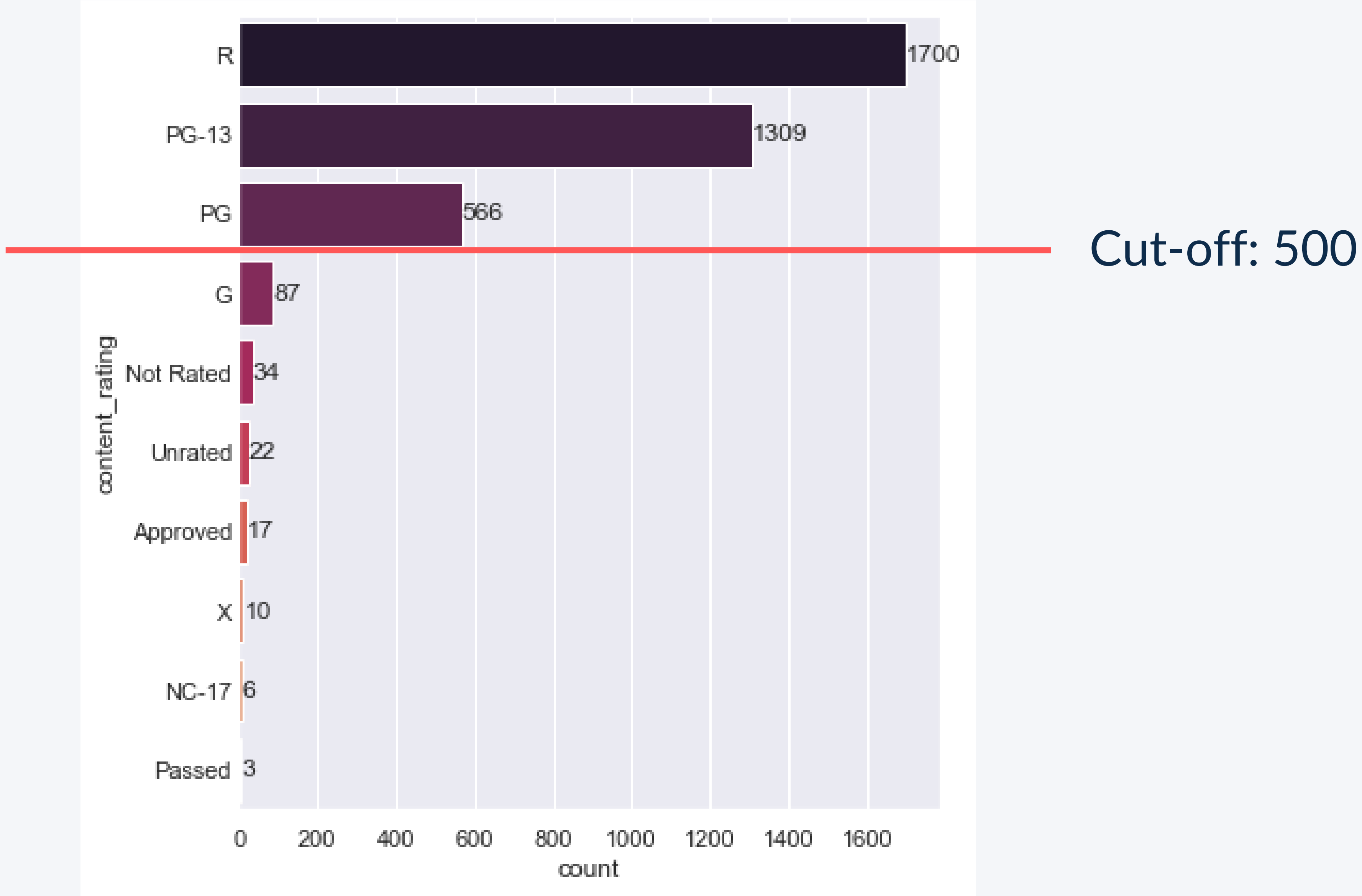
The USA dominates other countries in this list, so there can be 2 columns created, country:USA and country:others.

Step 3 - Preprocess the Data

Onehot encode categorical features

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





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 3 - Preprocess the Data

Onehot encode categorical features

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:Steven Spielberg
director_name:Woody Allen
director_name:others

country

country:USA
country:others

genres

genres:Comedy
genres:Comedy|Drama
genres:Comedy|Drama|Romance
genres:Drama
genres:others

language

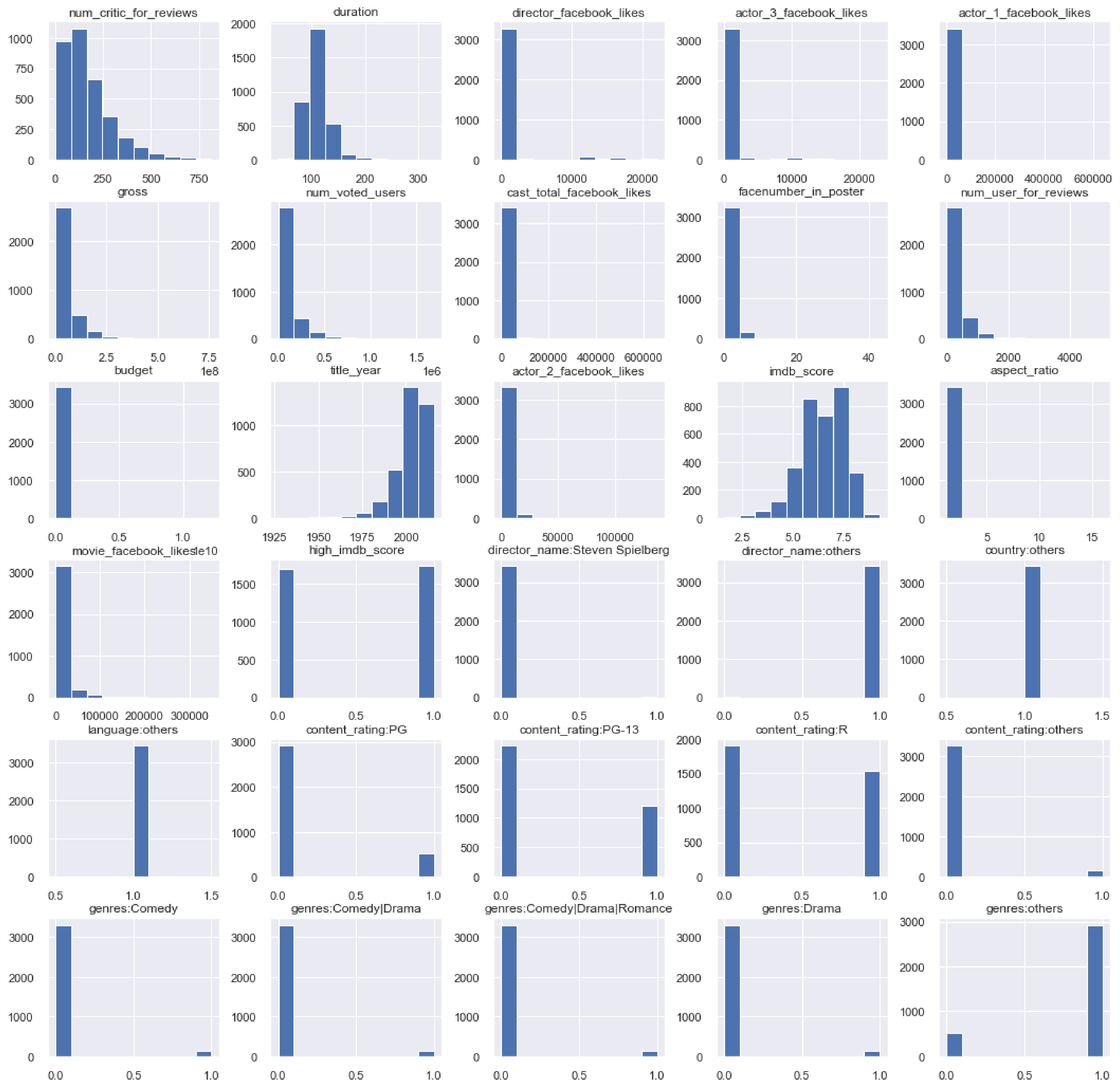
language:English
language:others

content_rating

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

Step 3 - Preprocess the Data

Histogram of the features



The histograms of the selected features are as shown.

Since the data have been OneHot encoded, many histograms show the bar that is either 0 or 1. Some numerical values like title_year, num_critic_for_reviews, duration are more skewed, and imdb_score appears to be somewhat normally distributed.

Step 3 - Preprocess the Data

Split train-test

```
X_train, X_test, y_train, y_test = train_test_split(df_dropped, movie_score_target, test_size=0.2,shuffle=True,stratify=movie_score_target,random_state=30)
✓ 0.0s
```

20% of the data will be randomly split to be used for evaluating the performance of the model, and the remaining 80% will be used for training the model

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

Step 3.1 - Model the Data - DecisionTreeClassifier (without GridSearchCV)

DecisionTreeClassifier (without GridSearchCV)

```
#Create DT Classifier
DT2=DecisionTreeClassifier(criterion='entropy',max_depth=5, min_samples_split=300)

#Train the model using the training sets
DT2.fit(X_train, y_train)

#Predict the response for test dataset
y_pred =DT2.predict(X_test)
```
✓ 0.0s

Since the model can become very complex due to the number of features available here, it is important to prevent the model from becoming too complex, leading to overfitting.

This is done by setting the minimum training inputs on each leaf, by setting the **min_samples_split to 300** to limit the number of leaves

By default, the max_depth is set to None and the model will be as complex as necessary to fit the data provided. However, this can quickly result in overfitting the training data. By setting the **max_depth is set to 5,** I limit the depth of the decision tree to 5 levels and combat overfitting my train data.

Step 3.1 - Model the Data - DecisionTreeClassifier (without GridSearchCV)

Evaluate classification result of test dataset

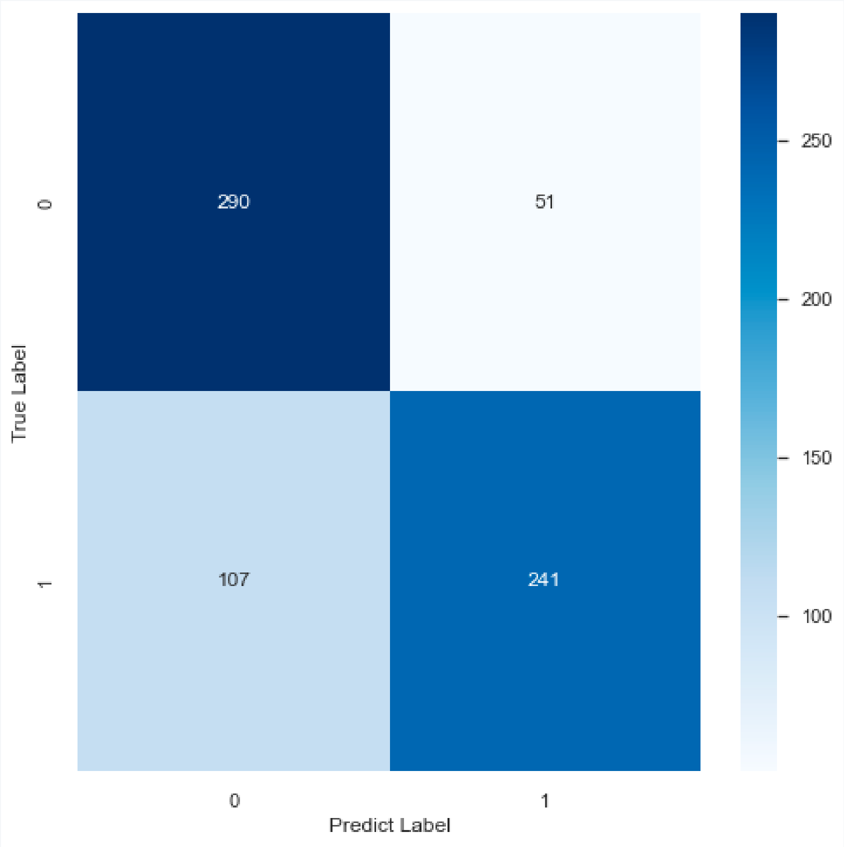
```
Classification Report:
                 precision    recall  f1-score   support

high_imdb_score       0.73      0.85      0.79       341
 low_imdb_score       0.83      0.69      0.75       348

       accuracy                           0.77       689
      macro avg       0.78      0.77      0.77       689
   weighted avg       0.78      0.77      0.77       689


Accuracy on train:   0.778

Accuracy on test:  0.771
```



Precision

- Of all the movies that the model predicted to have a high IMDB score, 73% actually have a high IMDB score
- Of all the movies that the model predicted to have a low IMDB score, 83% actually have a low IMDB score

Recall

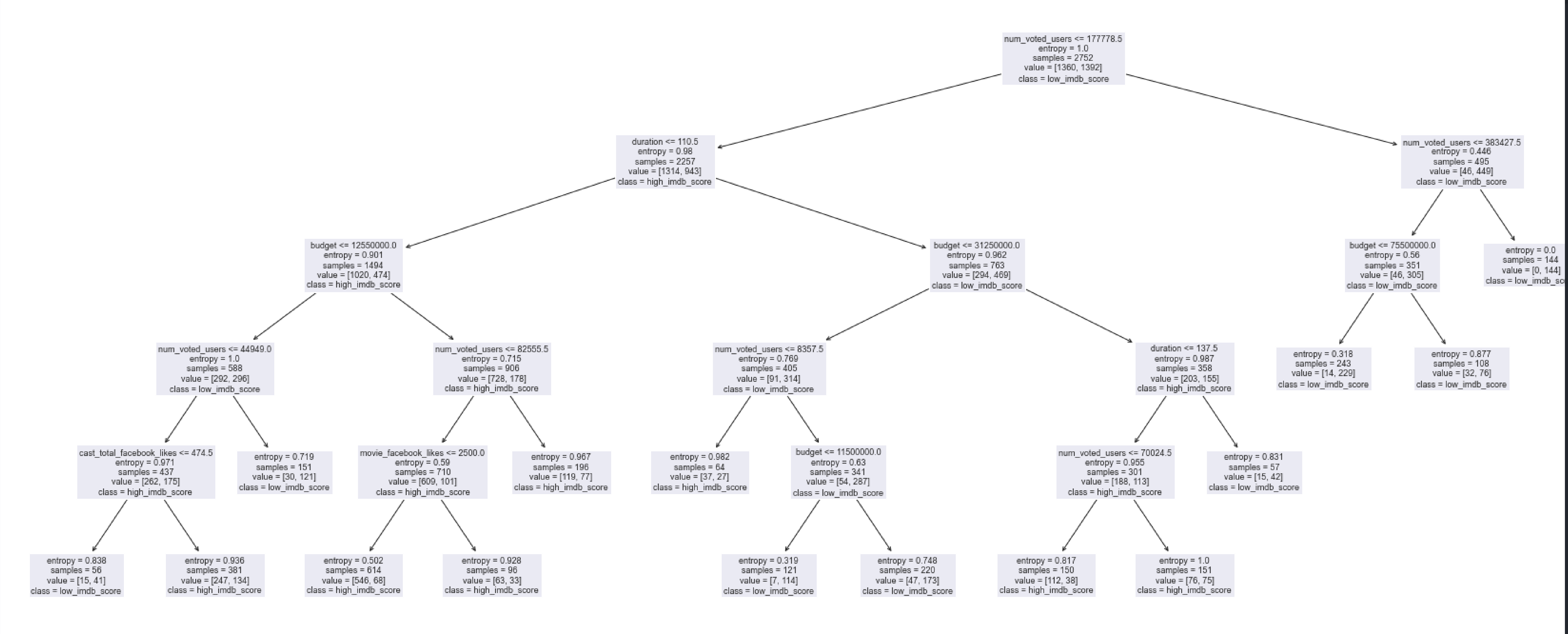
- Of all the movies that actually have a high IMDB score, 85% are correctly identified
- Of all the movies that actually have a low IMDB score, 69% are correctly identified

Accuracy: model correctly classified 77.1% of the instances (test data)

Note: the test data accuracy is very close to the train data accuracy, suggesting that the model does not overfit or underfit.

Step 3.1 - Model the Data - DecisionTreeClassifier (without GridSearchCV)

Evaluate classification result of test dataset




```
|--- num_voted_users <= 177778.50
|   |--- duration <= 110.50
|   |   |--- budget <= 12550000.00
|   |   |   |--- num_voted_users <= 44949.00
|   |   |   |   |--- cast_total_facebook_likes <= 474.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- cast_total_facebook_likes >  474.50
|   |   |   |   |   |--- class: 0
|   |   |   |--- num_voted_users >  44949.00
|   |   |   |   |--- class: 1
|   |   |--- budget >  12550000.00
|   |   |   |--- num_voted_users <= 82555.50
|   |   |   |   |--- movie_facebook_likes <= 2500.00
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- movie_facebook_likes >  2500.00
|   |   |   |   |   |--- class: 0
|   |   |   |--- num_voted_users >  82555.50
|   |   |   |   |--- class: 0
|   |--- duration >  110.50
|   |   |--- budget <= 31250000.00
|   |   |   |--- num_voted_users <= 8357.50
|   |   |   |   |--- class: 0
|   |   |   |--- num_voted_users >  8357.50
|   |   |   |   |--- budget <= 11500000.00
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- budget >  11500000.00
|   |   |   |   |   |--- class: 1
|   |   |--- budget >  31250000.00
|   |   |   |--- duration <= 137.50
|   |   |   |   |--- num_voted_users <= 70024.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- num_voted_users >  70024.50
|   |   |   |   |   |--- class: 0
|   |   |   |--- duration >  137.50
|   |   |   |   |--- class: 1
|--- num_voted_users >  177778.50
|   |--- num_voted_users <= 383427.50
|   |   |--- budget <= 75500000.00
|   |   |   |--- class: 1
|   |   |--- budget >  75500000.00
|   |   |   |--- class: 1
|   |--- num_voted_users >  383427.50
|   |   |--- class: 1
```

The decision tree has a depth of 5, as dictated by the max_depth parameter earlier.
Note that only the nodes with over 200 samples as dictated by *min_samples_split = 200*.

In doing so, the model didn't becomes maximally complex, which is a good sign for preventing overfitting.

Step 3.1 - Model the Data - DecisionTreeClassifier (without GridSearchCV)

Interpret the feature importance

	features	impact
0	num_critic_for_reviews	0.000000
1	duration	0.166531
2	director_facebook_likes	0.000000
3	gross	0.013643
4	num_voted_users	0.533789
5	cast_total_facebook_likes	0.023190
6	facenumber_in_poster	0.000000
7	num_user_for_reviews	0.000000
8	budget	0.239100
9	title_year	0.000000
10	aspect_ratio	0.000000
11	movie_facebook_likes	0.023748
12	director_name:Steven Spielberg	0.000000
13	director_name:others	0.000000
14	country:others	0.000000
15	language:others	0.000000
16	content_rating:PG	0.000000
17	content_rating:PG-13	0.000000
18	content_rating:R	0.000000
19	content_rating:others	0.000000
20	genres:Comedy	0.000000
21	genres:Comedy|Drama	0.000000
22	genres:Comedy|Drama|Romance	0.000000
23	genres:Drama	0.000000
24	genres:others	0.000000

Of all the features used, only a few of them are considered impactful, including

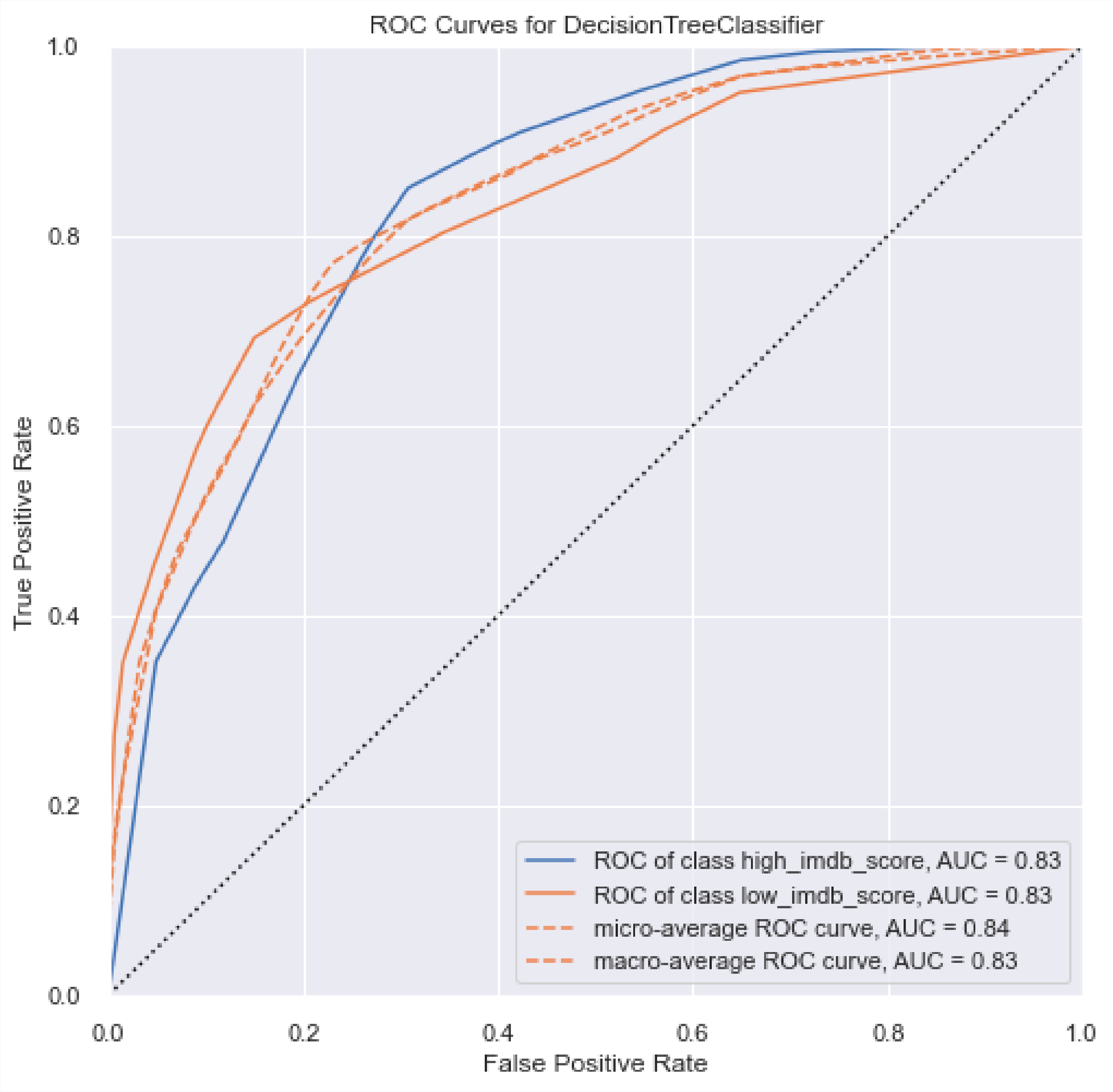
- duration
- budget
- num_voted_users

Some features do impact the model, but to a much smaller degree

- movie_facebook_likes
- cast_total_facebook_likes
- gross

Step 3.1 - Model the Data - DecisionTreeClassifier (without GridSearchCV)
Interpret the classification model performance

ROC: 0.83190077
AUC: 0.83190077



The F1-score for the "high_imdb_score" class is 0.79, and the F1-score for the "low_imdb_score" class is 0.75.

The ROC and AUC are at 0.83190077, which is considered decent.

Considering both metrics, I would consider that this model provides acceptable performance for predicting a social science problem.

Step 3.2 - Model the Data - DecisionTreeClassifier + gridserchcv

DecisionTreeClassifier + gridsearchcv

```
DT3=DecisionTreeClassifier()
cv = StratifiedKFold(8)
param_val = [{'criterion':['entropy','gini'],'max_depth':[1,6],'min_samples_split':np.arange(2,10)}]
#grid search configuration
grid = GridSearchCV(DT3, param_val, cv = cv,scoring='roc_auc_ovr')
#fitting into our data
grid.fit(X_train, y_train)

#Predict the response for test dataset

y_pred_2=grid.predict(X_test)
y_pred_2_prob=grid.predict_proba(X_test)
```

GridSearchCV allows you to find the best hyperparameters for the decision tree by performing an exhaustive search over the specified parameter grid.

Thus, the prediction of the model with the use of gridsearchcv should technically improve the model performance

Since the model can become very complex due to the number of features available here, it is important to prevent the model from becoming too complex, leading to overfitting.

This is done by setting the minimum training inputs on each leaf, by setting the **min_samples_split [2, 4, 6, 8, 10]** to limit the number of leaves

By default, the max_depth is set to None and the model will be as complex as necessary to fit the data provided. However, this can quickly result in overfitting the training data. By setting the **max_depth is set to 6,** I limit the depth of the decision tree to 6 levels and combat overfitting my train data.

Note: the max_depth of 6 is the best depth that results the highest ROC AUC scores. This will be shown later in the slides.

Step 3.2 - Model the Data - DecisionTreeClassifier + gridserchcv

Evaluate classification result of test dataset

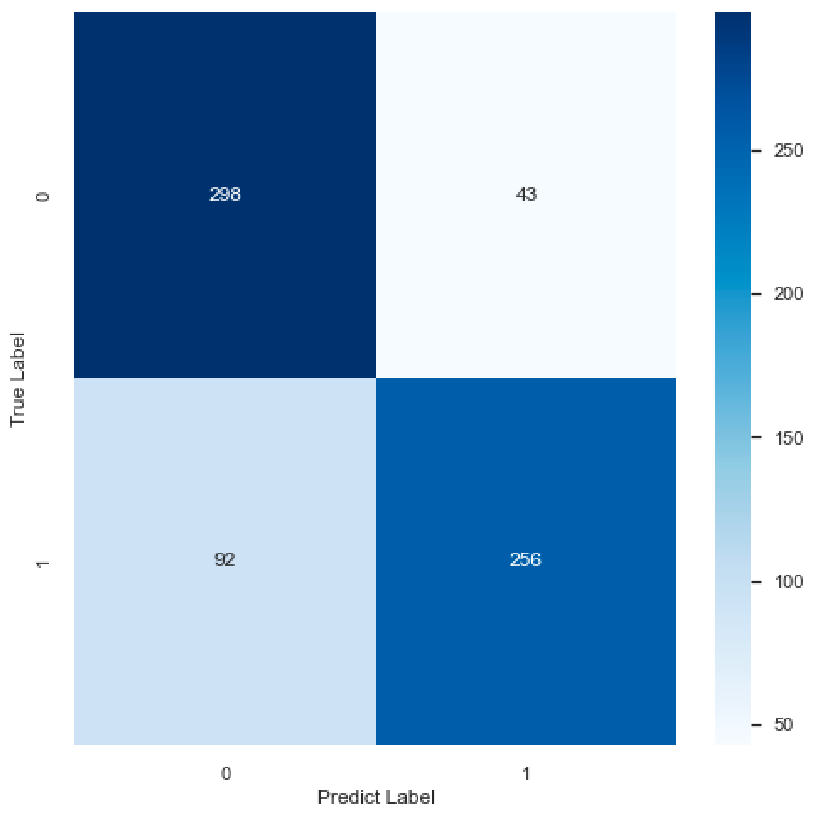
```
Classification Report:
                  precision    recall  f1-score   support

high_imdb_score       0.76      0.87      0.82       341
 low_imdb_score       0.86      0.74      0.79       348

       accuracy                           0.80       689
      macro avg       0.81      0.80      0.80       689
   weighted avg       0.81      0.80      0.80       689


Accuracy on train:   0.826

Accuracy on test:  0.804
```



Precision

- Of all the movies that the model predicted to have a high IMDB score, 76% actually have a high IMDB score
- Of all the movies that the model predicted to have a low IMDB score, 86% actually have a low IMDB score

Recall

- Of all the movies that actually have a high IMDB score, 87% are correctly identified
- Of all the movies that actually have a low IMDB score, 74% are correctly identified

Accuracy: model correctly classified 80.4% of the instances

Note: the test data accuracy is relatively close to the train data accuracy, suggesting that the model does not overfit or underfit.

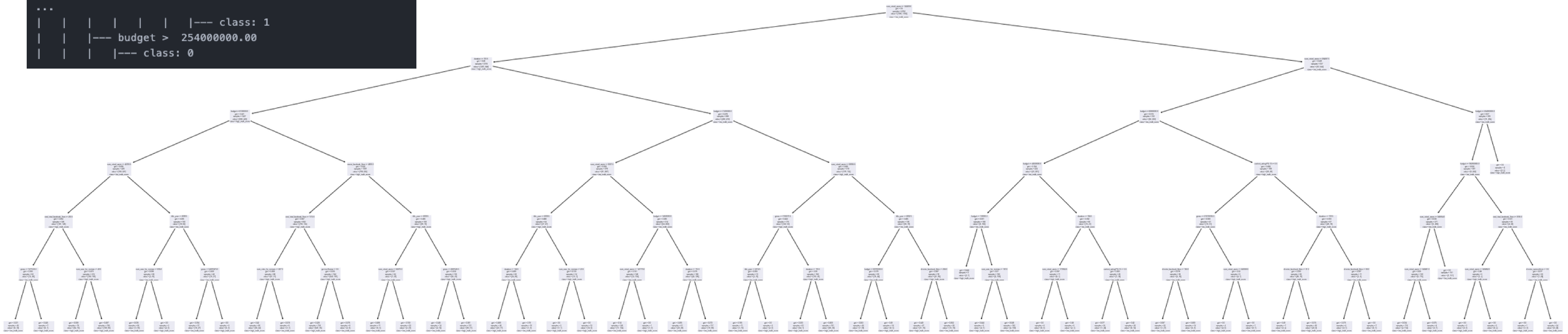
Step 3.2 - Model the Data - DecisionTreeClassifier + gridserchcv

Evaluate classification result of test dataset

```
|--- num_voted_users <= 142409.50
|   |--- duration <= 110.50
|   |   |--- budget <= 8100000.00
|   |   |   |--- num_voted_users <= 42093.50
|   |   |   |   |--- cast_total_facebook_likes <= 474.50
|   |   |   |   |   |--- gross <= 7271535.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- gross >  7271535.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- cast_total_facebook_likes >  474.50
|   |   |   |   |   |--- num_user_for_reviews <= 49.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- num_user_for_reviews >  49.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |--- num_voted_users >  42093.50
|   |   |   |   |--- title_year <= 2008.50
|   |   |   |   |   |--- num_user_for_reviews <= 874.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- num_user_for_reviews >  874.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- title_year >  2008.50
|   |   |   |   |   |--- gross <= 54097470.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- gross >  54097470.00
|   |   |   |   |   |   |--- class: 0
...
|   |   |   |   |   |   |--- class: 1
|   |   |--- budget >  254000000.00
|   |   |   |--- class: 0
```

The decision tree has a depth of 6, as dictated by the max_depth parameter earlier.

In doing so, the model didn't becomes maximally complex, which is a good sign for preventing overfitting.



Step 3.2 - Model the Data - DecisionTreeClassifier + gridserchcv
Interpret the feature importance

	features	impact
0	num_critic_for_reviews	0.000000
1	duration	0.166531
2	director_facebook_likes	0.000000
3	gross	0.013643
4	num_voted_users	0.533789
5	cast_total_facebook_likes	0.023190
6	facenumber_in_poster	0.000000
7	num_user_for_reviews	0.000000
8	budget	0.239100
9	title_year	0.000000
10	aspect_ratio	0.000000
11	movie_facebook_likes	0.023748
12	director_name:Steven Spielberg	0.000000
13	director_name:others	0.000000
14	country:others	0.000000
15	language:others	0.000000
16	content_rating:PG	0.000000
17	content_rating:PG-13	0.000000
18	content_rating:R	0.000000
19	content_rating:others	0.000000
20	genres:Comedy	0.000000
21	genres:Comedy|Drama	0.000000
22	genres:Comedy|Drama|Romance	0.000000
23	genres:Drama	0.000000
24	genres:others	0.000000

Of all the features used, only a few of them are considered impactful, including

- duration
- budget
- num_voted_users

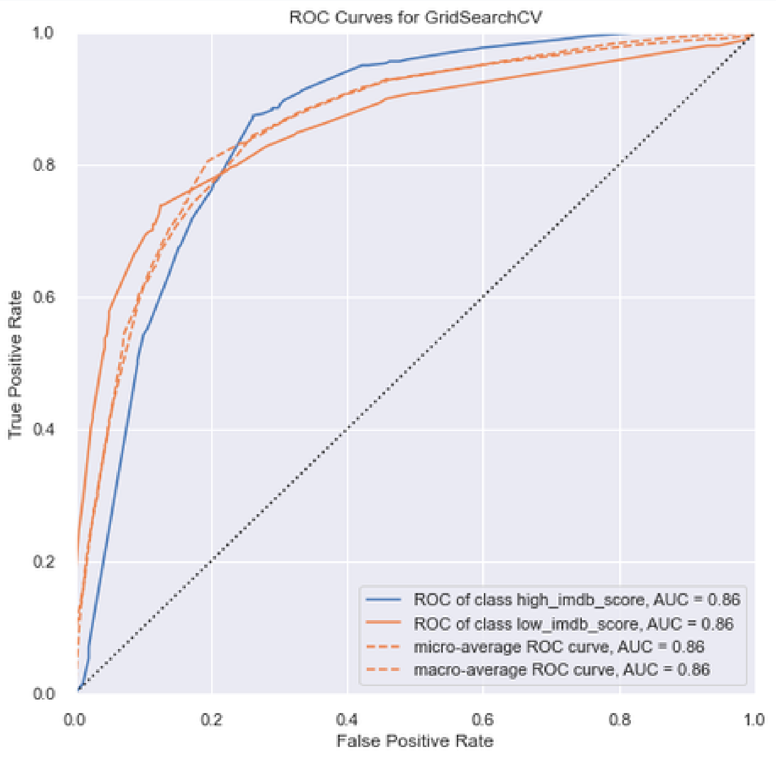
Some features do impact the model, but to a much smaller degree

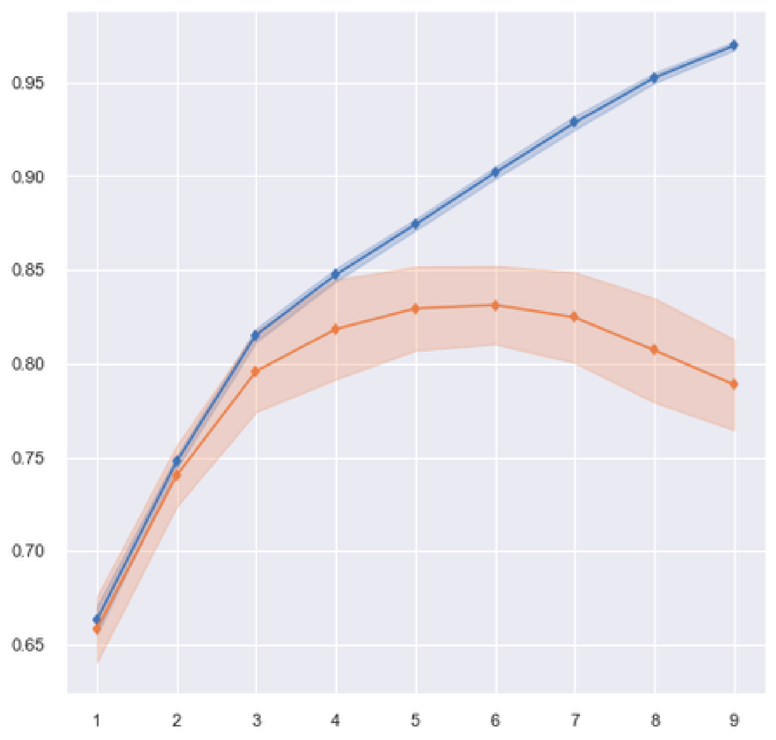
- movie_facebook_likes
- cast_total_facebook_likes
- gross

All the same features are still considered impactful, although their weights are changed

Step 3.2 - Model the Data - DecisionTreeClassifier + gridserchcv

Interpret the classification model performance





The F1-score for the "high_imdb_score" class is 0.82, and the F1-score for the "low_imdb_score" class is 0.79.

The ROC and AUC are at 0.8564524555903867, which is considered decent.

Considering both metrics, I would consider that this model provides acceptable performance for predicting a social science problem.

The ValidationCurve plots the ROC AUC score of the training data in blur and the ROC AUC score of the testing data in orange.

It can be observed that the score reaches its maximum at max_depth = 6. Thus, the model has been modified to use 6 as the max depth as shown in the previous slides.

Model Improvement

To further improve this model, there are some actions that can be taken

1. Tune the hyperparameters
    a. max_depth, min_samples_split, min_samples_leaf, and max_features can be tuned to find the most suitable values that yield the best result for the high_imdb_score prediction
2. Feature selection
    a. Since there are many features, the features can be narrowed down after the most important features have been identified.
    b. Unfortunately, there weren't many important features present, so I did not exclude any features in this exercise
3. Use larger dataset
    a. with more data to train the model, the model's performance can be improved and I can reduce the risk of overfitting the model