Practical 12

implementation of any real timeapplication using suitable machine learning algorithem technique.

Topic:Movie Rating Predicton

Input:Dataset containing data like num_critic_for_reviews ,duration ,num_voted_users ,num_user_for_reviews ,movie_facbook_likes ,director_facebook_likes etc.

Output:Predict rating based on this input (X) dataset using RandomForestRegressor.

In [356]:
```
import pandas as pd
import tkinter as tk
from tkinter import filedialog, messagebox
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import defaultdict,Counter
from sklearn.model_selection import train_test_split as tts
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
import ipywidgets as widgets
from IPython.display import display
import warnings
```

In [357]:
```
warnings.filterwarnings('ignore')
```

In [358]:
```
df = pd.read_csv("movie_metadata.csv")
```

In [359]:
```
df.sample(5)
```

Out[359]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_fa
1469	Color	John Singleton	180.0	106.0	309.0	
1509	Color	Joachim Rønning	58.0	93.0	18.0	
2208	Color	John Luessenhop	282.0	92.0	34.0	
4811	Color	Roger Nygard	43.0	86.0	3.0	
3992	Black and White	Richard Brooks	79.0	134.0	174.0	

5 rows × 28 columns

In [360]:
```
df.columns
```

Out[360]:
```
Index(['color', '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 [361]:
```
df.shape
```

Out[361]:
```
(5043, 28)
```

In [362]:
```
df.isnull().sum()
```

Out[362]:
```
color                       19
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
```

In [363]:
```
df['gross'].fillna(df['gross'].median(),inplace=True)
df['budget'].fillna(df['budget'].median(),inplace=True)
df.dropna(inplace=True)
```

In [364]:
```
df.shape
```

Out[364]:
```
(4411, 28)
```

About the Data (EDA)

Creating a new column to show main genre of movie. This will help in sorting out movies according to their genre types

In [365]:
```
df['main_genre'] = df['genres'].apply(lambda x: x.split('|')[0] if '|' in x el
```

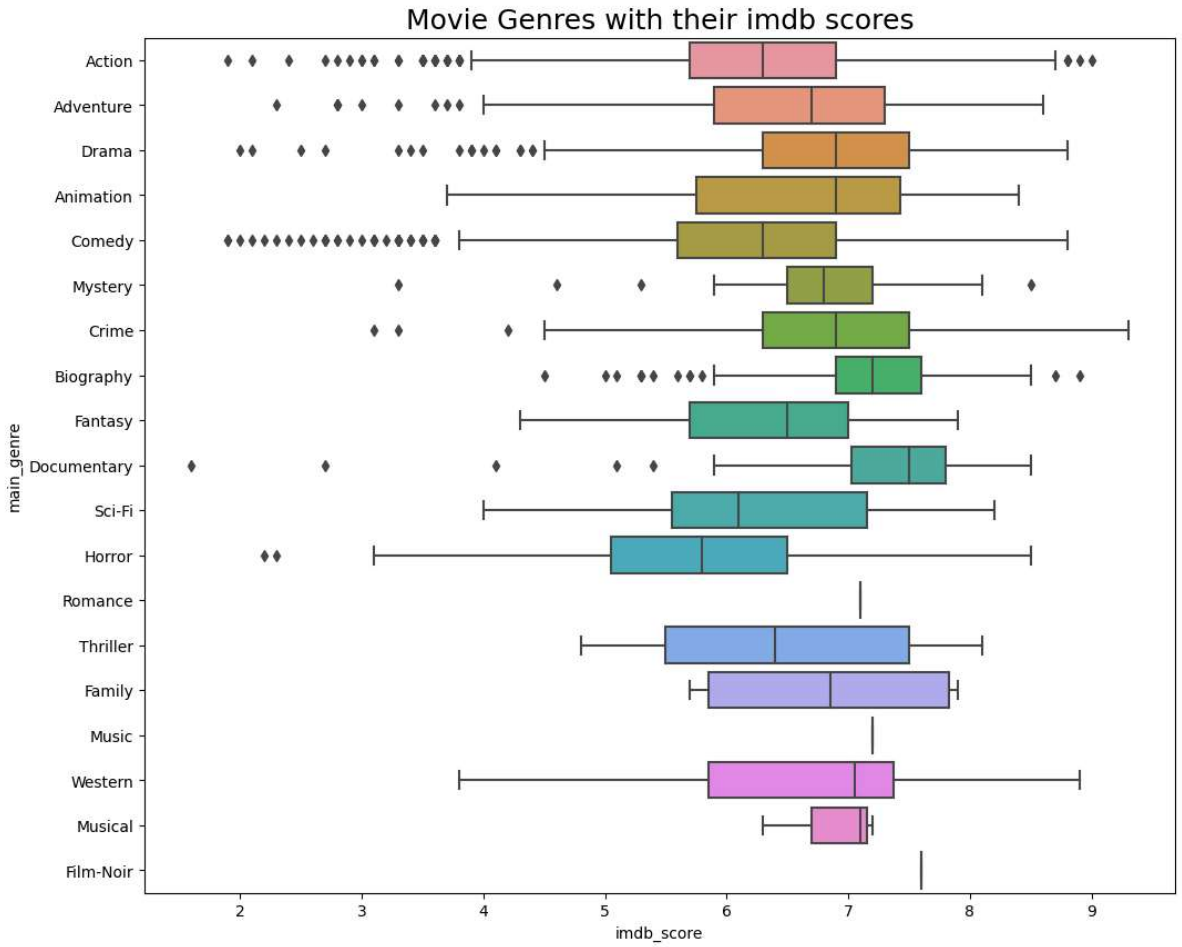
In [366]:
```
df.sample(2)
```

Out[366]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_fac
3019	Color	Mary Lambert	98.0	103.0	52.0	
710	Color	Jay Roach	141.0	106.0	116.0	

2 rows × 29 columns

In [367]:
```
plt.figure(figsize=(12,10))
sns.boxplot(x='imdb_score',y='main_genre',data=df)
plt.title('Movie Genres with their imdb scores',fontsize=18)
plt.show()
```

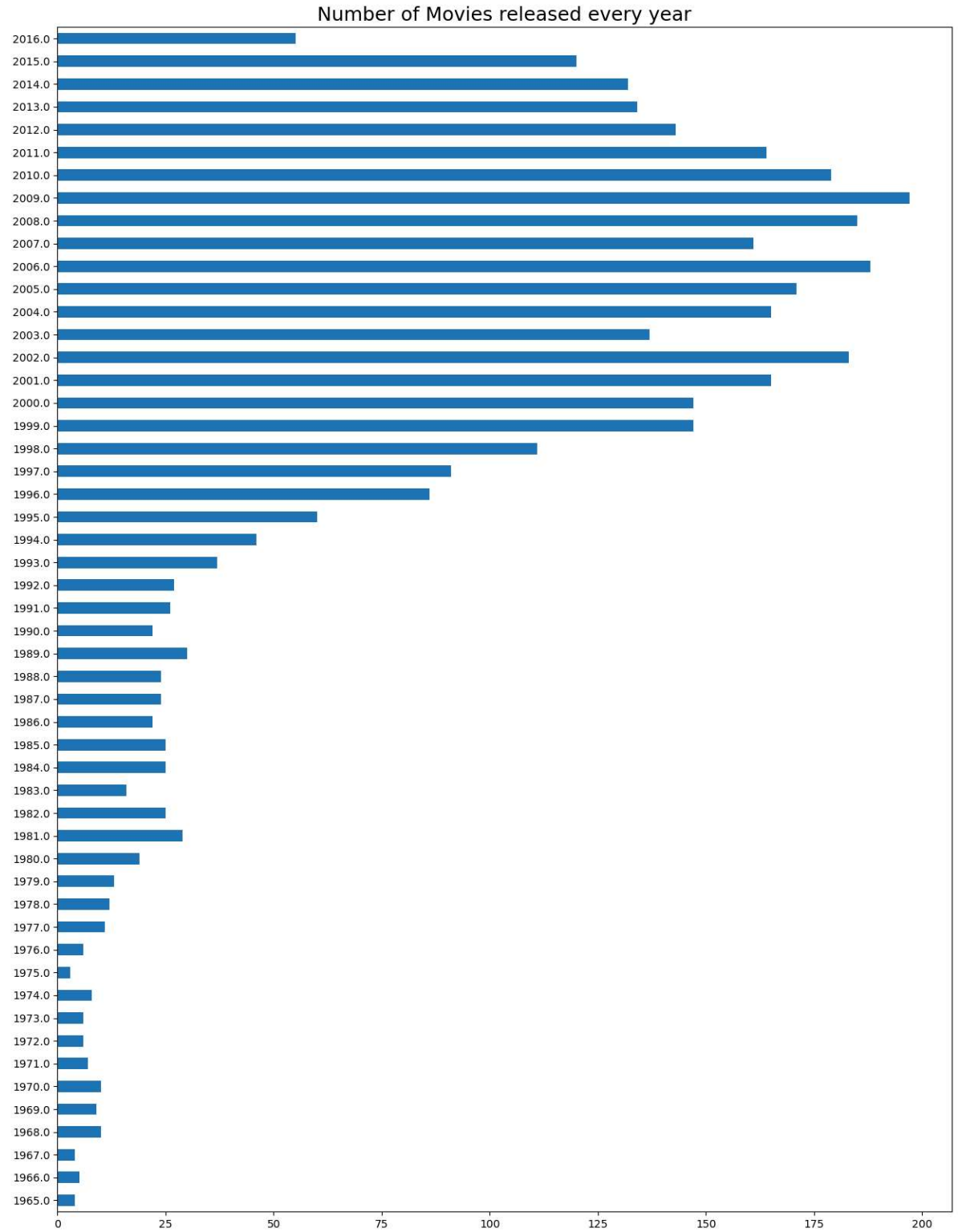


In [368]:
```
numeric_cols = df.select_dtypes(include=np.number).columns
z_scores = np.abs((df[numeric_cols] - df[numeric_cols].mean()) / df[numeric_co
threshold = 3
df = df[(z_scores < threshold).all(axis=1)]
```

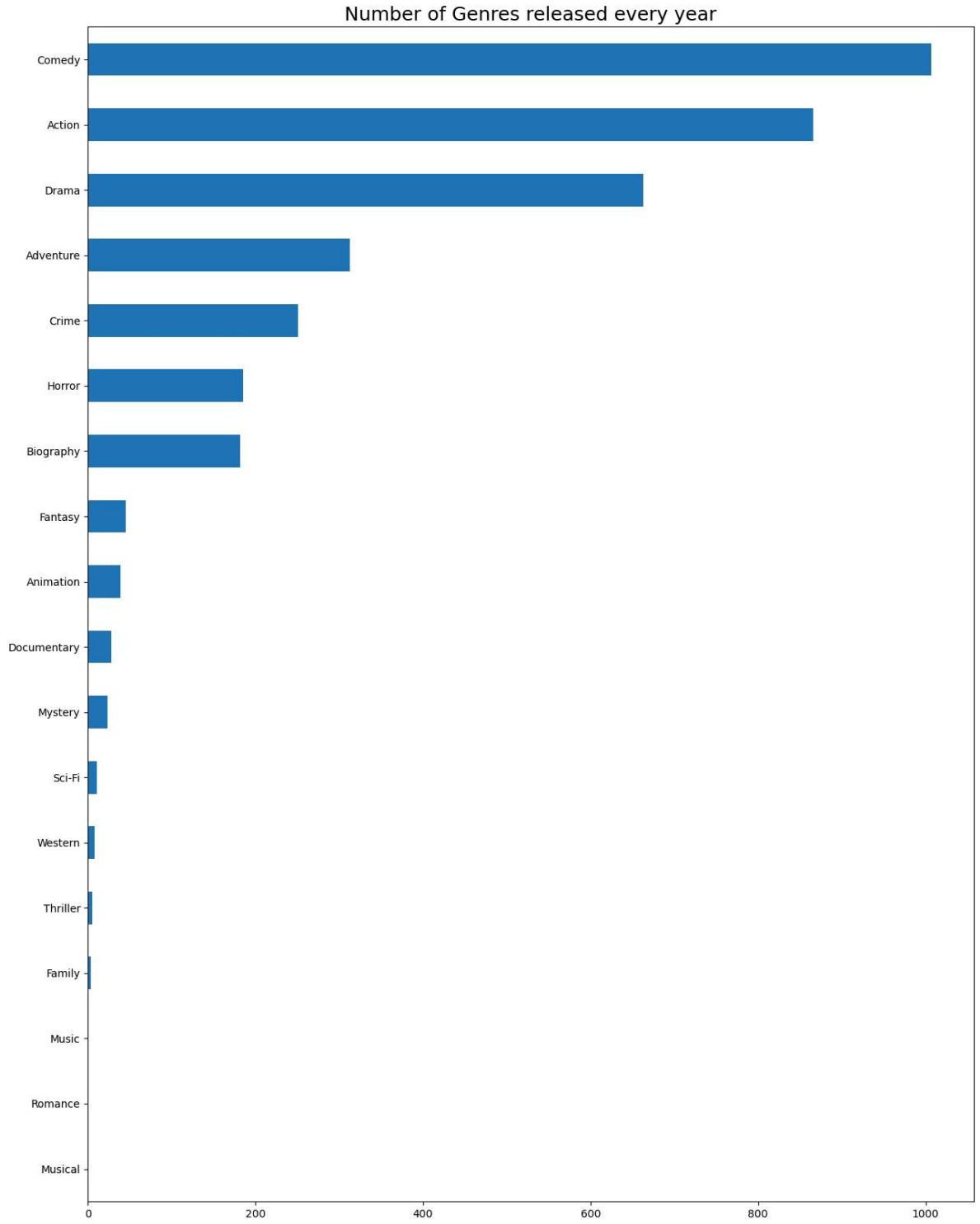
In [369]:
```
df.shape
```

Out[369]: (3632, 29)

In [370]:
```
df.title_year.value_counts(dropna=True).sort_index().plot(kind='barh',figsize=
plt.title("Number of Movies released every year",fontsize=18)
plt.show()
```



In [371]:
```
df.main_genre.value_counts(dropna=True).sort_values().plot(kind='barh',figsize
plt.title("Number of Genres released every year",fontsize=18)
plt.show()
```



In [ ]:

In [372]:
```
# Movies with the lowest Imdb rating
df[df['imdb_score']==3.3]
```

Out[372]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_fac
313	Color	Pitof	212.0	87.0	26.0	
1303	Color	William Malone	104.0	101.0	37.0	
1934	Color	Bob Spiers	33.0	93.0	14.0	
2519	Color	Ian Iqbal Rashid	62.0	94.0	8.0	
2550	Color	Perry Andelin Blake	56.0	80.0	11.0	
2935	Color	Tamra Davis	111.0	93.0	33.0	
3197	Color	Tom Brady	49.0	97.0	105.0	
3230	Color	Alan Metter	17.0	83.0	3.0	
3799	Color	Klaus Menzel	13.0	103.0	34.0	
4019	Color	Uwe Boll	17.0	86.0	892.0	
4127	Color	Philip Zlotorynski	14.0	80.0	0.0	
4285	Color	Gary Rogers	7.0	120.0	0.0	
4769	Color	Tamra Davis	111.0	93.0	33.0	

13 rows × 29 columns

In [373]:
```
# Movie with the highest Imdb rating
df[df['imdb_score']==8.9]
```

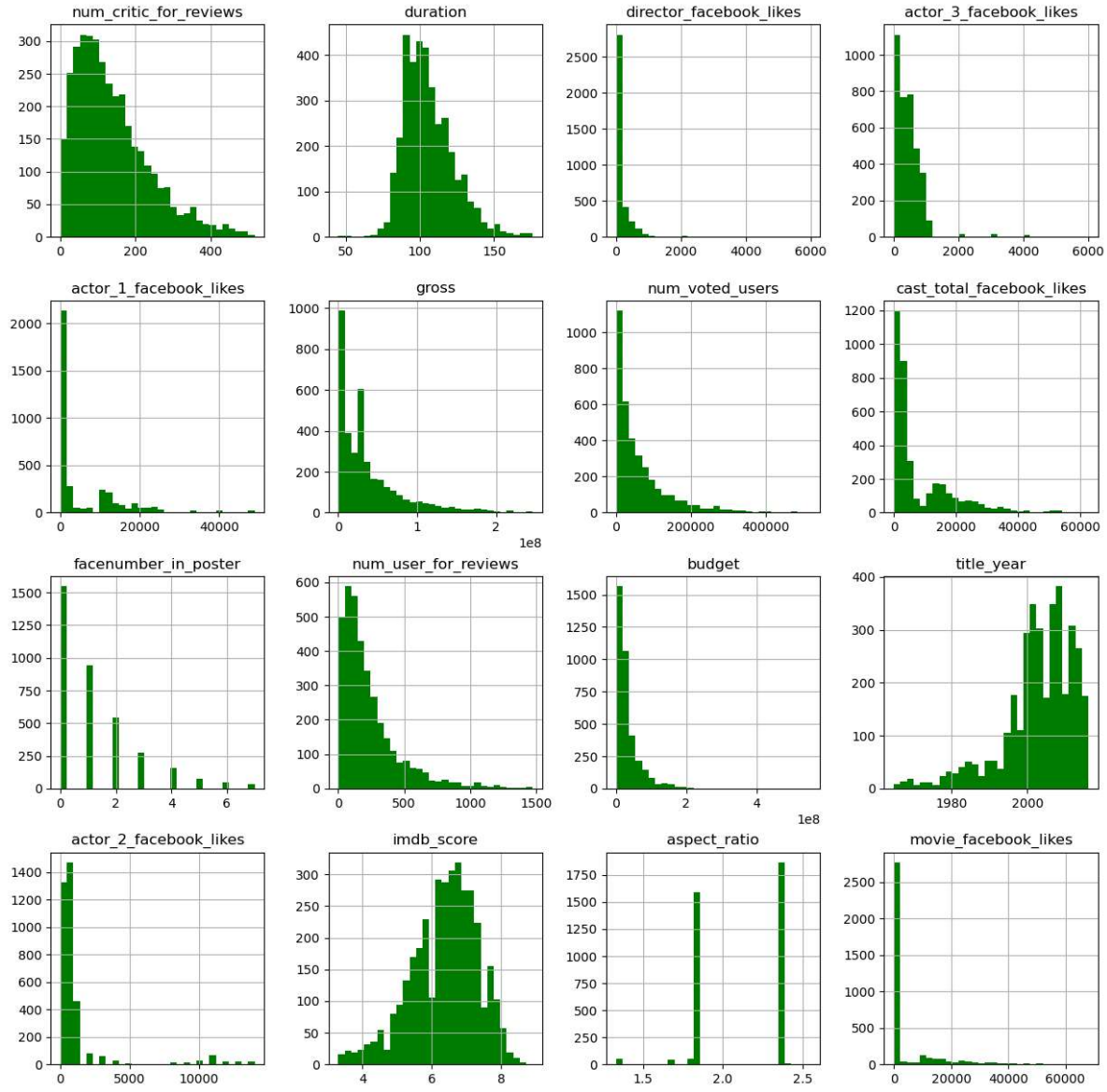
Out[373]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_fac
4498	Color	Sergio Leone	181.0	142.0	0.0	

1 rows × 29 columns

Histogram of all columns in df

In [374]:
```
df.hist(bins=30,figsize=(15,15),color='g')
plt.show()
```



Adding New Column that shows number of genres in movie

In [375]:
```
df['num_genres'] = df.genres.apply(lambda x: len(x.split('|')))
```

In [376]:
```
df.sample(2)
```

Out[376]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_fac
768	Color	John Pasquin	111.0	115.0	11.0	
3682	Color	Aki Kaurismäki	205.0	93.0	592.0	

2 rows × 30 columns

In [377]:
```
df.num_genres.max()
```

Out[377]: 8

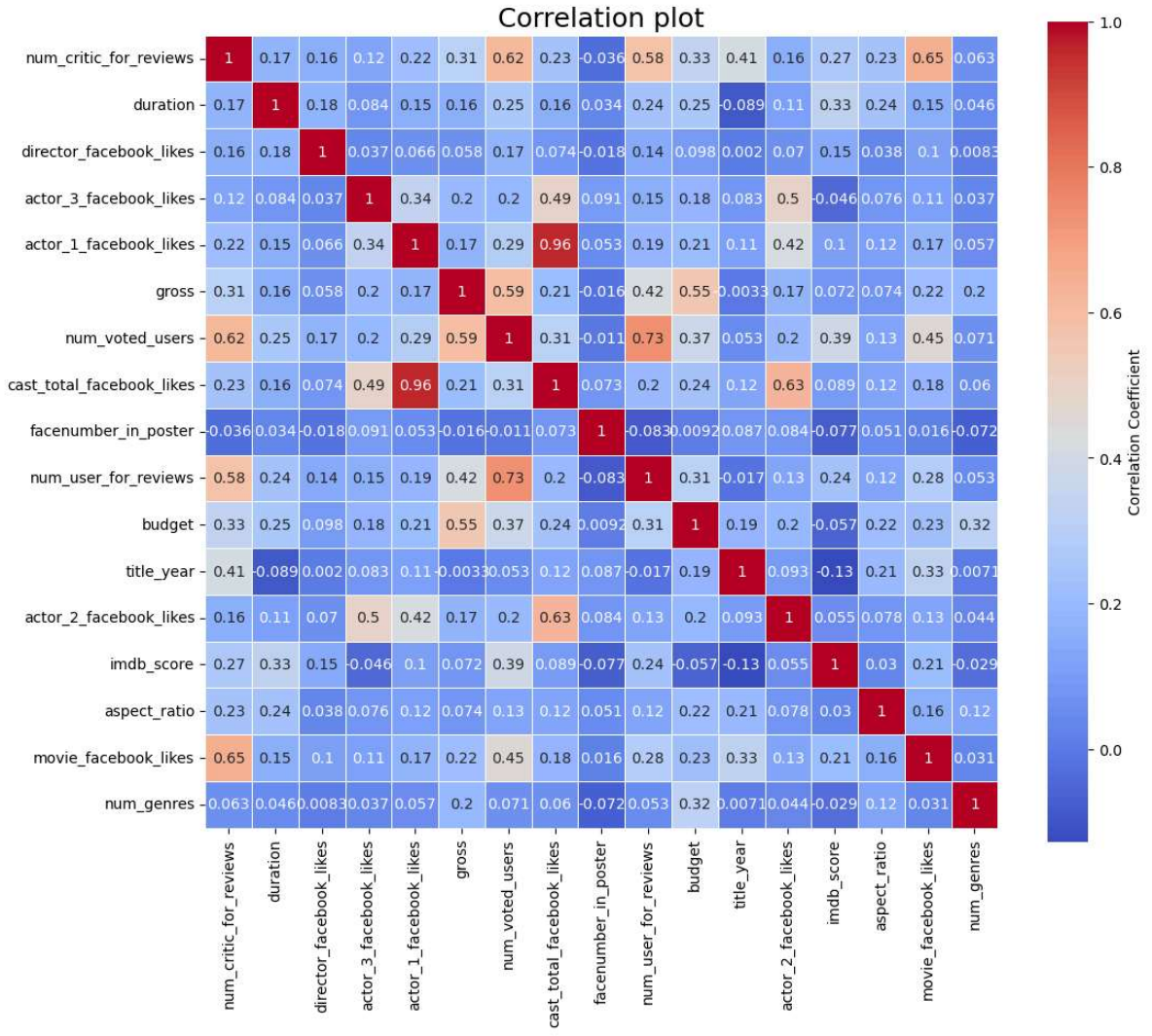
In [378]:
```
df[df.num_genres==8]
```

Out[378]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_fac
902	Color	Don Bluth	78.0	94.0	383.0	
1387	Color	John Frankenheimer	126.0	124.0	287.0	
2340	Color	Kevin Munroe	138.0	107.0	14.0	

3 rows × 30 columns

In [379]:
```
plt.figure(figsize=(12,10))
sns.heatmap(df.corr(),annot=True,linewidths=.5,
            cmap='coolwarm',square=True,cbar_kws={'label': 'Correlation Coeffi
plt.title("Correlation plot",fontsize=18)
plt.show()
```



In [ ]:
```
```

Selected Cols for model

num_critic_for_reviews
duration
num_voted_users
num_user_for_reviews
movie_facbook_likes
director_facebook_likes

In [380]:
```
X = df[['num_critic_for_reviews','duration','num_voted_users','num_user_for_re
y_rating = df['imdb_score']
y_status = df[['budget', 'gross']].apply(lambda row: 1 if row['budget']*3 <= r
```

In [381]:
```
X.shape,y_rating.shape,y_status.shape
```

Out[381]:
```
((3632, 6), (3632,), (3632,))
```

In [382]:
```
X_train, X_test, y_train_rating, y_test_rating = tts(X, y_rating, test_size=0.
X_train_status, X_test_status, y_train_status, y_test_status = tts(X, y_status
```

In [383]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [384]:
```
critic_reviews = widgets.FloatText(description="Critic Reviews:")
duration = widgets.FloatText(description="Duration (minutes):")
voted_users = widgets.FloatText(description="Voted Users:")
user_reviews = widgets.FloatText(description="User Reviews:")
movie_likes = widgets.FloatText(description="Movie Likes:")
director_likes = widgets.FloatText(description="Director Likes:")
```

In [385]:
```
display(critic_reviews, duration, voted_users, user_reviews, movie_likes, dire
```

Critic Revie… 0

Duration (… 0

Voted Users: 0

User Revie… 0

Movie Likes: 0

Director Lik… 0

In [386]:
```
knn = KNeighborsRegressor(n_neighbors=12)
knn.fit(X_train_scaled, y_train_rating)
pred_imdb_score_knn = knn.predict([[critic_reviews.value, duration.value, vote
print("Predicted IMDb score (KNN):", pred_imdb_score_knn[0])
```

```
Predicted IMDb score (KNN): 6.133333333333334
```

In [387]:
```
mse_rf = mean_squared_error(y_test_rating, rf_regressor.predict(X_test_scaled))
print("Mean Squared Error of RandomForestRegressor:", mse_rf)
```

```
Mean Squared Error of RandomForestRegressor: 0.6346543865199454
```

In [388]:
```
RandomForestRegressor(n_estimators=100, random_state=42)
it(X_train_scaled, y_train_rating)
e_rf = rf_regressor.predict([[critic_reviews.value, duration.value, voted_users
ed IMDb score (Random Forest):", pred_imdb_score_rf[0])
```

```
Predicted IMDb score (Random Forest): 6.505999999999999
```

In [389]:
```
mse_rf = mean_squared_error(y_test_rating, rf_regressor.predict(X_test_scaled))
print("Mean Squared Error of RandomForestRegressor:", mse_rf)
```

```
Mean Squared Error of RandomForestRegressor: 0.6346543865199454
```

In [ ]:

In [ ]: