In [1]:
```
# import libaries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [21]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn
import scipy.stats as stats
import seaborn as sns
import plotly.express as px

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import accuracy_score, mean_absolute_error, mean_squared_error, r
import warnings
warnings.filterwarnings('ignore')
plt.rcParams["figure.figsize"] = [10,5]
# Ignore warnings
import warnings
# Set the warning filter to ignore FutureWarning
warnings.simplefilter(action = "ignore", category = FutureWarning)
```

In [33]:
```
import pandas as pd

# Escape each backslash with another backslash
file_path = "C:\\Users\\ARUTHRA D\\Downloads\\IMDb Movies India.csv (1)\\IMDb Movies I
df = pd.read_csv(file_path, encoding='latin1')
```

In [34]:
```
print(df.head())
```

```
                                 Name    Year Duration            Genre  \
0                                         NaN      NaN            Drama
1  #Gadhvi (He thought he was Gandhi)  (2019)  109 min            Drama
2                          #Homecoming  (2021)   90 min   Drama, Musical
3                              #Yaaram  (2019)  110 min  Comedy, Romance
4                    ...And Once Again  (2010)  105 min            Drama

   Rating Votes             Director       Actor 1             Actor 2  \
0     NaN   NaN       J.S. Randhawa      Manmauji              Birbal
1     7.0     8       Gaurav Bakshi  Rasika Dugal      Vivek Ghamande
2     NaN   NaN  Soumyajit Majumdar  Sayani Gupta   Plabita Borthakur
3     4.4    35          Ovais Khan       Prateik          Ishita Raj
4     NaN   NaN        Amol Palekar  Rajat Kapoor  Rituparna Sengupta

           Actor 3
0  Rajendra Bhatia
1    Arvind Jangid
2       Roy Angana
3  Siddhant Kapoor
4      Antara Mali
```

In [36]:
```
print(df.head())
print(df.columns)
```

```
                                 Name    Year Duration            Genre  \
0                                         NaN      NaN            Drama
1  #Gadhvi (He thought he was Gandhi)  (2019)  109 min            Drama
2                          #Homecoming  (2021)   90 min   Drama, Musical
3                              #Yaaram  (2019)  110 min  Comedy, Romance
4                    ...And Once Again  (2010)  105 min            Drama

   Rating Votes             Director       Actor 1             Actor 2  \
0     NaN   NaN       J.S. Randhawa      Manmauji              Birbal
1     7.0     8       Gaurav Bakshi  Rasika Dugal      Vivek Ghamande
2     NaN   NaN  Soumyajit Majumdar  Sayani Gupta   Plabita Borthakur
3     4.4    35          Ovais Khan       Prateik          Ishita Raj
4     NaN   NaN        Amol Palekar  Rajat Kapoor  Rituparna Sengupta

           Actor 3
0  Rajendra Bhatia
1    Arvind Jangid
2       Roy Angana
3  Siddhant Kapoor
4      Antara Mali
Index(['Name', 'Year', 'Duration', 'Genre', 'Rating', 'Votes', 'Director',
       'Actor 1', 'Actor 2', 'Actor 3'],
      dtype='object')
```

In [37]:
```
# Data preprocessing
# Check for missing values
print(df.isnull().sum())
```

```
Name           0
Year         528
Duration    8269
Genre       1877
Rating      7590
Votes       7589
Director     525
Actor 1     1617
Actor 2     2384
Actor 3     3144
dtype: int64
```

In [39]:
```
# Ensure the correct target column is used (inspect the column names)
print("Columns:", df.columns)
```

```
Columns: Index(['Name', 'Year', 'Duration', 'Genre', 'Rating', 'Votes', 'Director',
       'Actor 1', 'Actor 2', 'Actor 3'],
      dtype='object')
```

In [40]:
```
# Assuming the correct target column is named 'rating' or similar, adjust accordingly
target_column = 'rating'
```

In [43]:
```
# Drop rows with missing target values
df = df.dropna(subset=[target_column])

# Example: Encoding categorical variables
df['genre_encoded'] = df['genre'].astype('category').cat.codes
df['director_encoded'] = df['director'].astype('category').cat.codes
df['actors_encoded'] = df['actors'].astype('category').cat.codes

# Select relevant features
X = df[['genre_encoded', 'director_encoded', 'actors_encoded']]
y = df[target_column]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_25248\1847689084.py in ?()
      1 # Drop rows with missing target values
----> 2 df = df.dropna(subset=[target_column])
      3
      4 # Example: Encoding categorical variables
      5 df['genre_encoded'] = df['genre'].astype('category').cat.codes

~\anaconda3\Lib\site-packages\pandas\core\frame.py in ?(self, axis, how, thresh, subs
et, inplace, ignore_index)
   6403             ax = self._get_axis(agg_axis)
   6404             indices = ax.get_indexer_for(subset)
   6405             check = indices == -1
   6406             if check.any():
-> 6407                 raise KeyError(np.array(subset)[check].tolist())
   6408             agg_obj = self.take(indices, axis=agg_axis)
   6409
   6410         if thresh is not no_default:

KeyError: ['rating']
```

In [44]:
```
# Assuming the correct target column is named 'rating' or similar, adjust accordingly
target_column = 'rating'  # Adjust if the actual column name is different
if target_column not in df.columns:
    print(f"Column '{target_column}' not found in the dataset. Please verify the corre
else:
    # Drop rows with missing target values
    df = df.dropna(subset=[target_column])

    # Example: Encoding categorical variables
    df['genre_encoded'] = df['genre'].astype('category').cat.codes
    df['director_encoded'] = df['director'].astype('category').cat.codes
    df['actors_encoded'] = df['actors'].astype('category').cat.codes

    # Select relevant features
    X = df[['genre_encoded', 'director_encoded', 'actors_encoded']]
    y = df[target_column]

    # Splitting the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

    # Train a linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

    # Evaluate the model
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'Mean Squared Error: {mse}')
    print(f'R-squared: {r2}')

    # Plot actual vs predicted ratings
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=y_test, y=y_pred)
    plt.xlabel('Actual Ratings')
    plt.ylabel('Predicted Ratings')
    plt.title('Actual vs Predicted Movie Ratings')
    plt.show()
```

```
Column 'rating' not found in the dataset. Please verify the correct column name for r
atings.
```

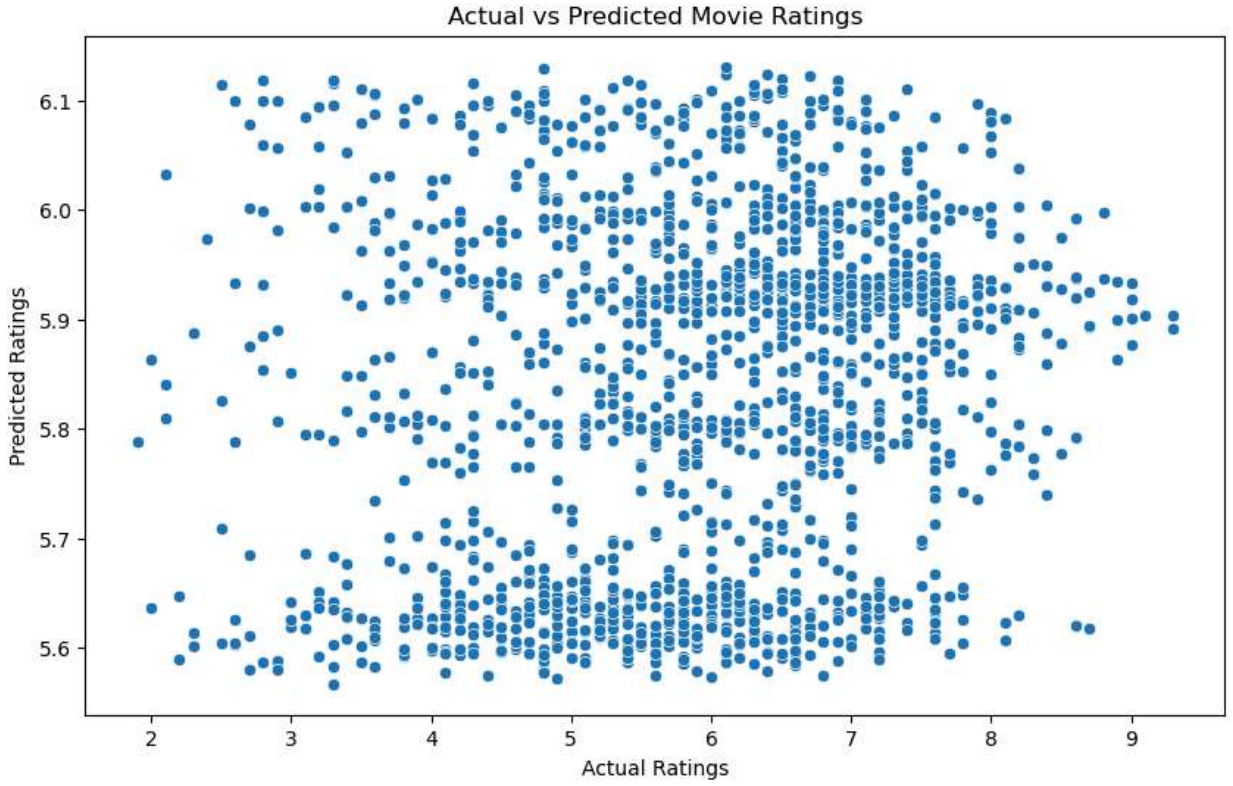
In [49]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = 'C:/Users/ARUTHRA D/Downloads/IMDb Movies India.csv (1)/IMDb Movies India.
df = pd.read_csv(file_path, encoding='latin1')

# Display the first few rows and columns of the dataset
print("First few rows of the dataset:")
print(df.head())

print("\nColumns in the dataset:")
print(df.columns)

# Check for missing values
print("\nMissing values in each column:")
print(df.isnull().sum())

# Assuming the correct target column is 'Rating'
target_column = 'Rating'
```

```
First few rows of the dataset:
                                  Name    Year Duration            Genre  \
0                                          NaN      NaN            Drama
1  #Gadhvi (He thought he was Gandhi)  (2019)  109 min            Drama
2                          #Homecoming  (2021)   90 min   Drama, Musical
3                              #Yaaram  (2019)  110 min  Comedy, Romance
4                   ...And Once Again  (2010)  105 min            Drama

   Rating Votes             Director       Actor 1             Actor 2  \
0     NaN   NaN       J.S. Randhawa      Manmauji              Birbal
1     7.0     8       Gaurav Bakshi  Rasika Dugal      Vivek Ghamande
2     NaN   NaN  Soumyajit Majumdar  Sayani Gupta   Plabita Borthakur
3     4.4    35          Ovais Khan       Prateik          Ishita Raj
4     NaN   NaN        Amol Palekar  Rajat Kapoor  Rituparna Sengupta

           Actor 3
0  Rajendra Bhatia
1    Arvind Jangid
2       Roy Angana
3  Siddhant Kapoor
4      Antara Mali

Columns in the dataset:
Index(['Name', 'Year', 'Duration', 'Genre', 'Rating', 'Votes', 'Director',
       'Actor 1', 'Actor 2', 'Actor 3'],
      dtype='object')

Missing values in each column:
Name           0
Year         528
Duration    8269
Genre       1877
Rating      7590
Votes       7589
Director     525
Actor 1     1617
Actor 2     2384
Actor 3     3144
dtype: int64
```

In [48]:
```
if target_column not in df.columns:
    print(f"Column '{target_column}' not found in the dataset. Please verify the corre
else:
    # Drop rows with missing target values
    df = df.dropna(subset=[target_column])

    # Example: Encoding categorical variables
    df['genre_encoded'] = df['Genre'].astype('category').cat.codes
    df['director_encoded'] = df['Director'].astype('category').cat.codes
    df['actor1_encoded'] = df['Actor 1'].astype('category').cat.codes
    df['actor2_encoded'] = df['Actor 2'].astype('category').cat.codes
    df['actor3_encoded'] = df['Actor 3'].astype('category').cat.codes

    # Select relevant features
    X = df[['genre_encoded', 'director_encoded', 'actor1_encoded', 'actor2_encoded', '
    y = df[target_column]

    # Splitting the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

    # Train a linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

    # Evaluate the model
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'Mean Squared Error: {mse}')
    print(f'R-squared: {r2}')

    # Plot actual vs predicted ratings
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=y_test, y=y_pred)
    plt.xlabel('Actual Ratings')
    plt.ylabel('Predicted Ratings')
    plt.title('Actual vs Predicted Movie Ratings')
    plt.show()
```

```
Mean Squared Error: 1.8154893850164557
R-squared: 0.023481628556979106
```



In [50]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn
import scipy.stats as stats
import seaborn as sns
import plotly.express as px

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import accuracy_score, mean_absolute_error, mean_squared_error, r
import warnings
warnings.filterwarnings('ignore')
plt.rcParams["figure.figsize"] = [10,5]
# Ignore warnings

import warnings
# Set the warning filter to ignore FutureWarning
warnings.simplefilter(action = "ignore", category = FutureWarning)
```

In [51]:
```
df.head()
```

Out[51]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	Birbal	Rajendra Bhatia
1	#Gadhvi (He thought he was Gandhi)	(2019)	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid
2	#Homecoming	(2021)	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borthakur	Roy Angana
3	#Yaaram	(2019)	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
4	...And Once Again	(2010)	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Rituparna Sengupta	Antara Mali

In [52]:
```
df.shape
```

Out[52]:
```
(15509, 10)
```

In [53]:
```
df.isnull().sum()
```

Out[53]:
```
Name           0
Year         528
Duration    8269
Genre       1877
Rating      7590
Votes       7589
Director     525
Actor 1     1617
Actor 2     2384
Actor 3     3144
dtype: int64
```

In [54]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15509 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Name      15509 non-null  object 
 1   Year      14981 non-null  object 
 2   Duration  7240 non-null   object 
 3   Genre     13632 non-null  object 
 4   Rating    7919 non-null   float64
 5   Votes     7920 non-null   object 
 6   Director  14984 non-null  object 
 7   Actor 1   13892 non-null  object 
 8   Actor 2   13125 non-null  object 
 9   Actor 3   12365 non-null  object 
dtypes: float64(1), object(9)
memory usage: 1.2+ MB
```

In [55]:
```
# Locating rows with missing values in columns from 1 to 9

nulls = df[df.iloc[:, 1:9].isna().all(axis=1)]
nulls.head()
```

Out[55]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
1836	Bang Bang Reloaded	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1920	Battle of bittora	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2653	Campus	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3403	Dancing Dad	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3807	Dial 100	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [56]:
```
#Checking if there are any typos

for col in df.select_dtypes(include = "object"):
    print(f"Name of Column: {col}")
    print(df[col].unique())
    print('\n', '-'*60, '\n')
```

```
Name of Column: Name
[' ' '#Gadhvi (He thought he was Gandhi)' '#Homecoming' ... 'Zulmi Raj'
 'Zulmi Shikari' 'Zulm-O-Sitam']

 ------------------------------------------------------------

Name of Column: Year
[nan '(2019)' '(2021)' '(2010)' '(1997)' '(2005)' '(2008)' '(2012)'
 '(2014)' '(2004)' '(2016)' '(1991)' '(1990)' '(2018)' '(1987)' '(1948)'
 '(1958)' '(2017)' '(2020)' '(2009)' '(2002)' '(1993)' '(1946)' '(1994)'
 '(2007)' '(2013)' '(2003)' '(1998)' '(1979)' '(1951)' '(1956)' '(1974)'
 '(2015)' '(2006)' '(1981)' '(1985)' '(2011)' '(2001)' '(1967)' '(1988)'
 '(1995)' '(1959)' '(1996)' '(1970)' '(1976)' '(2000)' '(1999)' '(1973)'
 '(1968)' '(1943)' '(1953)' '(1986)' '(1983)' '(1989)' '(1982)' '(1977)'
 '(1957)' '(1950)' '(1992)' '(1969)' '(1975)' '(1947)' '(1972)' '(1971)'
 '(1935)' '(1978)' '(1960)' '(1944)' '(1963)' '(1940)' '(1984)' '(1934)'
 '(1955)' '(1936)' '(1980)' '(1966)' '(1949)' '(1962)' '(1964)' '(1952)'
 '(1933)' '(1942)' '(1939)' '(1954)' '(1945)' '(1961)' '(1965)' '(1938)'
 '(1941)' '(1931)' '(1937)' '(2022)' '(1932)' '(1923)' '(1915)' '(1928)'
```

In [57]:
```
df.dropna(subset=['Name', 'Year', 'Duration', 'Rating', 'Votes', 'Director', 'Actor 1'
df['Name'] = df['Name'].str.extract('([A-Za-z\s\'\-]+)')
df['Year'] = df['Year'].str.replace(r'[()]', '', regex=True).astype(int)
df['Duration'] = pd.to_numeric(df['Duration'].str.replace(r' min', '', regex=True), er
df['Genre'] = df['Genre'].str.split(', ')
df = df.explode('Genre')
df['Genre'].fillna(df['Genre'].mode()[0], inplace=True)
df['Votes'] = pd.to_numeric(df['Votes'].str.replace(',', ''), errors='coerce')
df
```

Out[57]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
1	Gadhvi	2019	109	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid
3	Yaaram	2019	110	Comedy	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
3	Yaaram	2019	110	Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
5	Aur Pyaar Ho Gaya	1997	147	Comedy	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor
5	Aur Pyaar Ho Gaya	1997	147	Drama	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor
...	...	...	...	...	...	...	...	...	...	...
15503	Zulm Ki Zanjeer	1989	125	Drama	5.8	44	S.P. Muthuraman	Chiranjeevi	Jayamalini	Rajinikanth
15505	Zulmi	1999	129	Action	4.5	655	Kuku Kohli	Akshay Kumar	Twinkle Khanna	Aruna Irani
15505	Zulmi	1999	129	Drama	4.5	655	Kuku Kohli	Akshay Kumar	Twinkle Khanna	Aruna Irani
15508	Zulm-O-Sitam	1998	130	Action	6.2	20	K.C. Bokadia	Dharmendra	Jaya Prada	Arjun Sarja
15508	Zulm-O-Sitam	1998	130	Drama	6.2	20	K.C. Bokadia	Dharmendra	Jaya Prada	Arjun Sarja

12008 rows × 10 columns

In [58]:
```
duplicate = df.groupby(['Name', 'Year']).filter(lambda x: len(x) > 1)
duplicate.head(5)
```

Out[58]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
3	Yaaram	2019	110	Comedy	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
3	Yaaram	2019	110	Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
5	Aur Pyaar Ho Gaya	1997	147	Comedy	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor
5	Aur Pyaar Ho Gaya	1997	147	Drama	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor
5	Aur Pyaar Ho Gaya	1997	147	Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor

In [59]:
```
df = df.drop_duplicates(subset=['Name'], keep=False)
```

In [60]:
```
df.describe()
```

Out[60]:

	Year	Duration	Rating	Votes
count	1528.000000	1528.000000	1528.000000	1528.000000
mean	1997.972513	123.823953	5.976243	552.479712
std	21.181921	25.108144	1.412547	4311.631841
min	1931.000000	45.000000	1.600000	5.000000
25%	1985.000000	107.000000	5.100000	14.000000
50%	2004.000000	126.000000	6.100000	34.000000
75%	2016.000000	140.000000	7.000000	127.250000
max	2021.000000	300.000000	9.400000	101014.000000

In [61]:
```
df.describe(include = 'O')
```

Out[61]:

	Name	Genre	Director	Actor 1	Actor 2	Actor 3
count	1528	1528	1528	1528	1528	1528
unique	1528	20	1114	1010	1131	1154
top	Gadhvi	Drama	Kanti Shah	Mithun Chakraborty	Mithun Chakraborty	Pran
freq	1	789	13	22	12	16

In [62]:
```
# Find the row with the highest number of votes
max_votes_row = df[df['Votes'] == df['Votes'].max()]

# Get the name of the movie with the highest votes
movie_highest_votes = max_votes_row['Name'].values[0]

# Find the number of votes for the movie with the highest votes
votes_highest_votes = max_votes_row['Votes'].values[0]

print("Movie with the highest votes:", movie_highest_votes)
print("Number of votes for the movie with the highest votes:", votes_highest_votes)
print('\n', '='*100, '\n')


# Find the row with the lowest number of votes
min_votes_row = df[df['Votes'] == df['Votes'].min()]

# Get the name of the movie with the lowest votes
movie_lowest_votes = min_votes_row['Name'].values[0]

# Find the number of votes for the movie with the lowest votes
votes_lowest_votes = min_votes_row['Votes'].values[0]

print("Movie with the highest votes:", movie_lowest_votes)
print("Number of votes for the movie with the highest votes:", votes_lowest_votes)
```

```
Movie with the highest votes: My Name Is Khan
Number of votes for the movie with the highest votes: 101014

 ====================================================================================================

Movie with the highest votes: Anmol Sitaare
Number of votes for the movie with the highest votes: 5
```

In [63]:
```
# Find the row with the highest rating
max_rating_row = df[df['Rating'] == df['Rating'].max()]
movie_highest_rating = max_rating_row['Name'].values[0]
votes_highest_rating = max_rating_row['Votes'].values[0]

print("Movie with the highest rating:", movie_highest_rating)
print("Number of votes for the movie with the highest rating:", votes_highest_rating)
print('\n', '='*100, '\n')


# Find the row with the lowest rating
min_rating_row = df[df['Rating'] == df['Rating'].min()]
movie_lowest_rating = min_rating_row['Name'].values[0]
votes_lowest_rating = min_rating_row['Votes'].values[0]

print("Movie with the highest rating:", movie_lowest_rating)
print("Number of votes for the movie with the highest rating:", votes_lowest_rating)
```

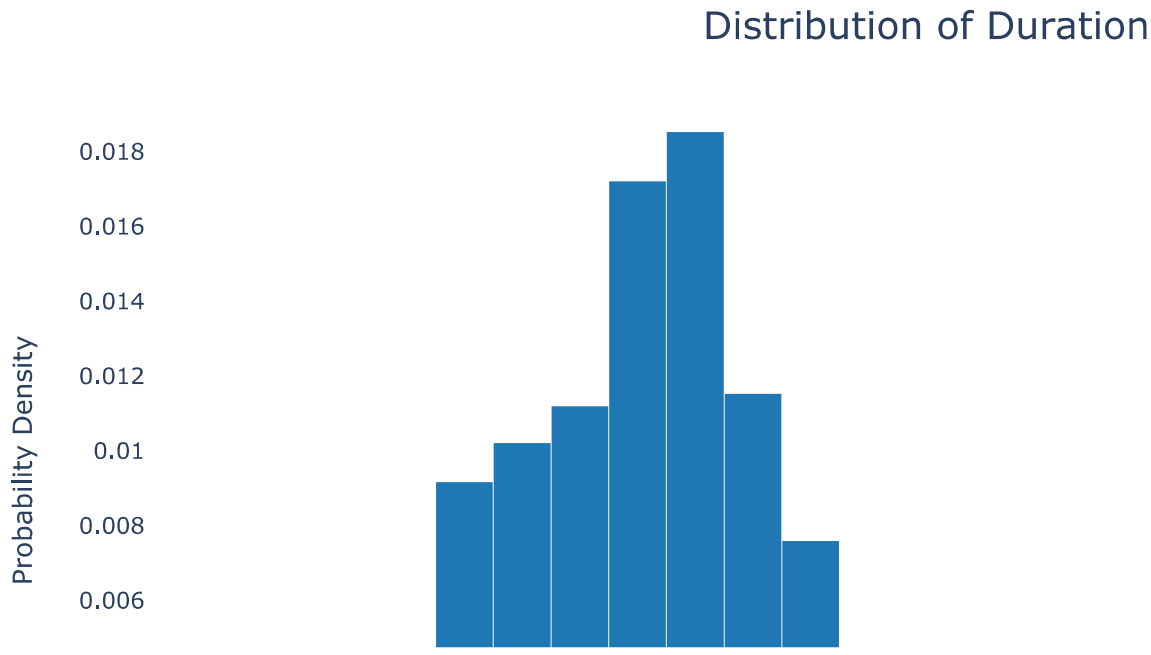
```
Movie with the highest rating: June
Number of votes for the movie with the highest rating: 18

 ========================================================================================
================

Movie with the highest rating: Mumbai Can Dance Saalaa
Number of votes for the movie with the highest rating: 43
```

In [64]:
```
# Group the dataset by the 'Director' column and count the number of movies each direct
director_counts = df['Director'].value_counts()

# Find the director with the highest number of movies directed
most_prolific_director = director_counts.idxmax()
num_movies_directed = director_counts.max()

print("Director with the most movies directed:", most_prolific_director)
print("Number of movies directed by", most_prolific_director, ":", num_movies_directed
print('\n', '='*100, '\n')


# Group the dataset by the 'Director' column and count the number of movies each direct
director_counts = df['Director'].value_counts()

# Find the director with the lowest number of movies directed
least_prolific_director = director_counts.idxmin()
num_movies_directed = director_counts.min()

print("Director with the most movies directed:", least_prolific_director)
print("Number of movies directed by", most_prolific_director, ":", num_movies_directed
```

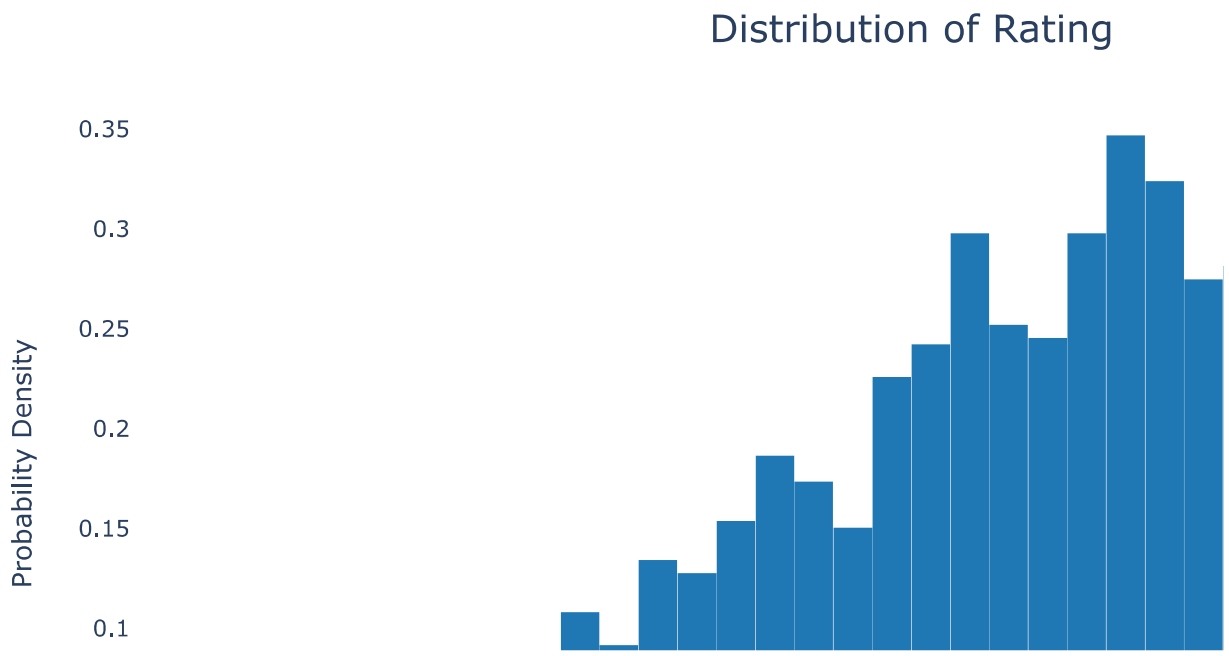
```
Director with the most movies directed: Kanti Shah
Number of movies directed by Kanti Shah : 13

 ========================================================================================
================

Director with the most movies directed: Sikandar Khanna
Number of movies directed by Kanti Shah : 1
```

In [65]:
```
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2',

# Create histogram plot using Plotly Express
fig_year = px.histogram(df, x='Year', histnorm='probability density', nbins=30, color_
fig_year.update_traces(selector=dict(type='histogram'))
fig_year.update_layout(
    title='Distribution of Year',
    title_x=0.5,
    title_pad=dict(t=20),
    title_font=dict(size=20),
    xaxis_title='Year',
    yaxis_title='Probability Density',
    xaxis=dict(showgrid=False),
    yaxis=dict(showgrid=False),
    bargap=0.02,
    plot_bgcolor='white'
)
```

In [66]:
```
fig_duration = px.histogram(df, x = 'Duration', histnorm='probability density', nbins
fig_duration.update_traces(selector=dict(type='histogram'))
fig_duration.update_layout(title='Distribution of Duration', title_x=0.5, title_pad=di
fig_duration.show()
```

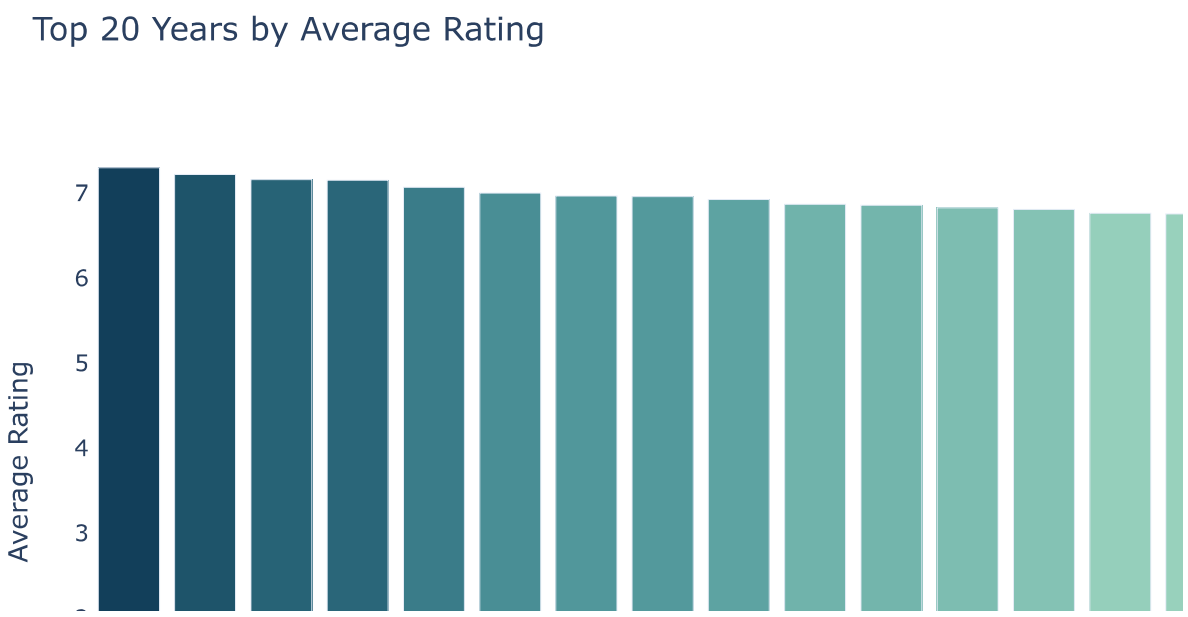


In [67]:
```
fig_rating = px.histogram(df, x = 'Rating', histnorm='probability density', nbins = 40
fig_rating.update_traces(selector=dict(type='histogram'))
fig_rating.update_layout(title='Distribution of Rating', title_x=0.5, title_pad=dict(t
fig_rating.show()
```

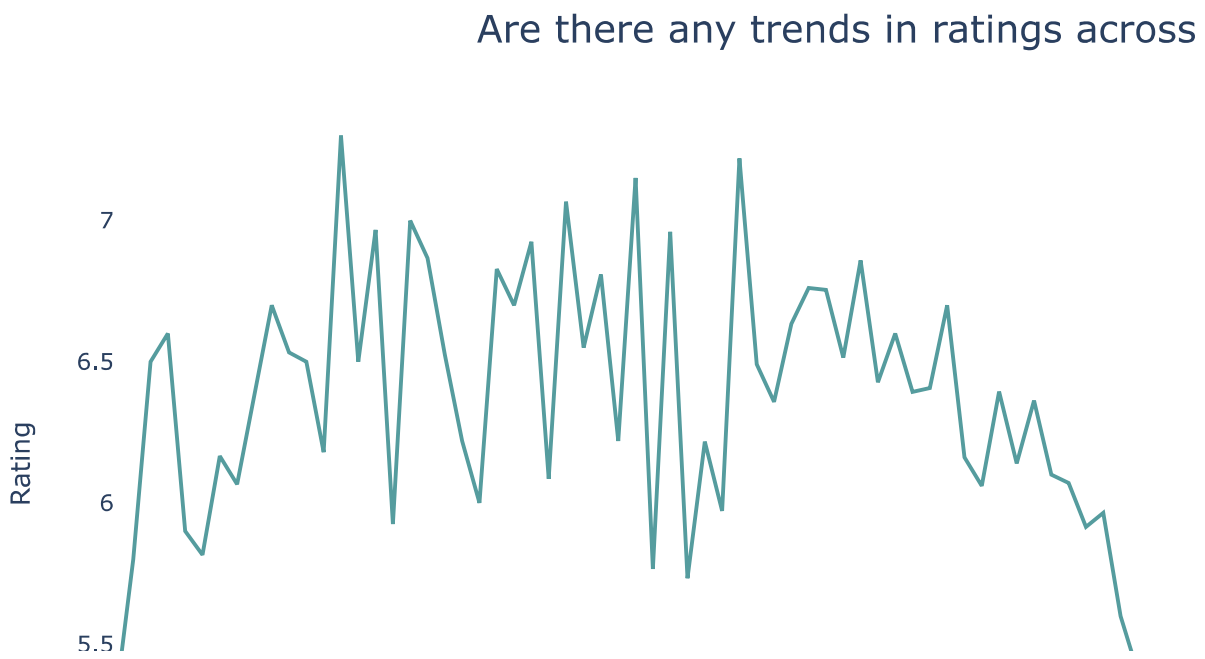


In [68]:
```
fig_votes = px.box(df, x = 'Votes', color_discrete_sequence = colors)
fig_votes.update_layout(title='Distribution of Votes', title_x=0.5, title_pad=dict(t=2
fig_votes.show()
```

In [69]:
```
year_avg_rating = df.groupby('Year')['Rating'].mean().reset_index()

top_5_years = year_avg_rating.nlargest(20, 'Rating')
fig = px.bar(top_5_years, x='Year', y='Rating', title='Top 20 Years by Average Rating'
fig.update_xaxes(type='category')
fig.update_layout(xaxis_title='Year', yaxis_title='Average Rating', plot_bgcolor = 'wh
fig.show()
```

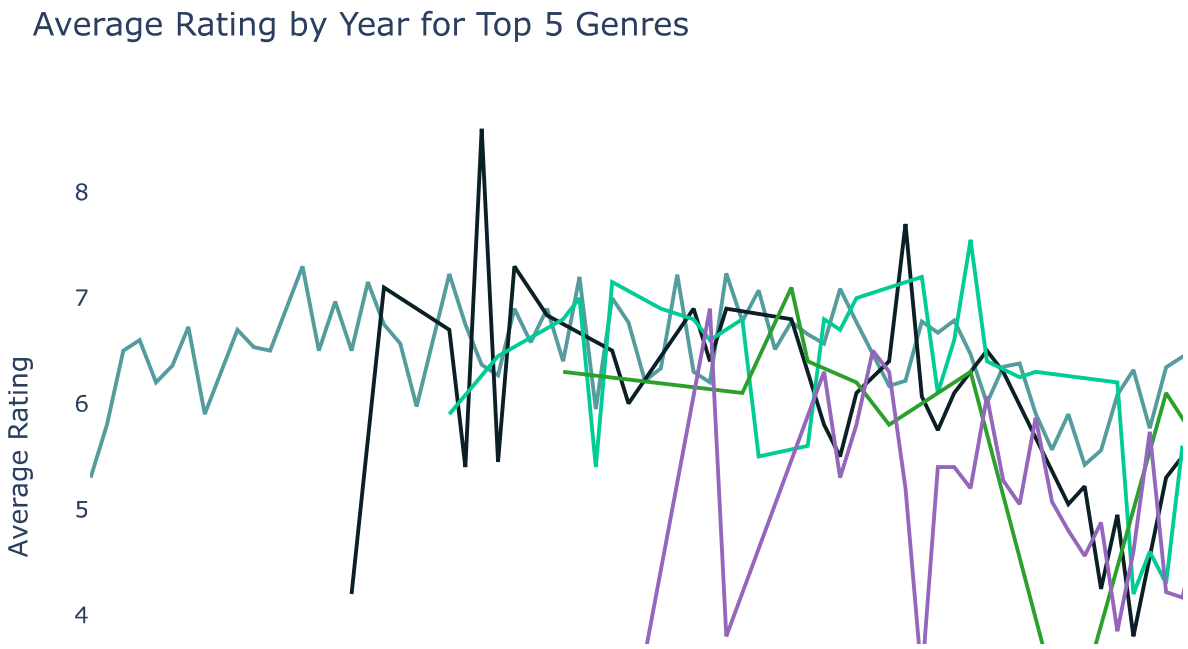


In [70]:
```
# Group data by Year and calculate the average rating
average_rating_by_year = df.groupby('Year')['Rating'].mean().reset_index()

# Create the line plot with Plotly Express
fig = px.line(average_rating_by_year, x='Year', y='Rating', color_discrete_sequence=['
fig.update_layout(title='Are there any trends in ratings across year?', title_x=0.5, t
fig.show()
```

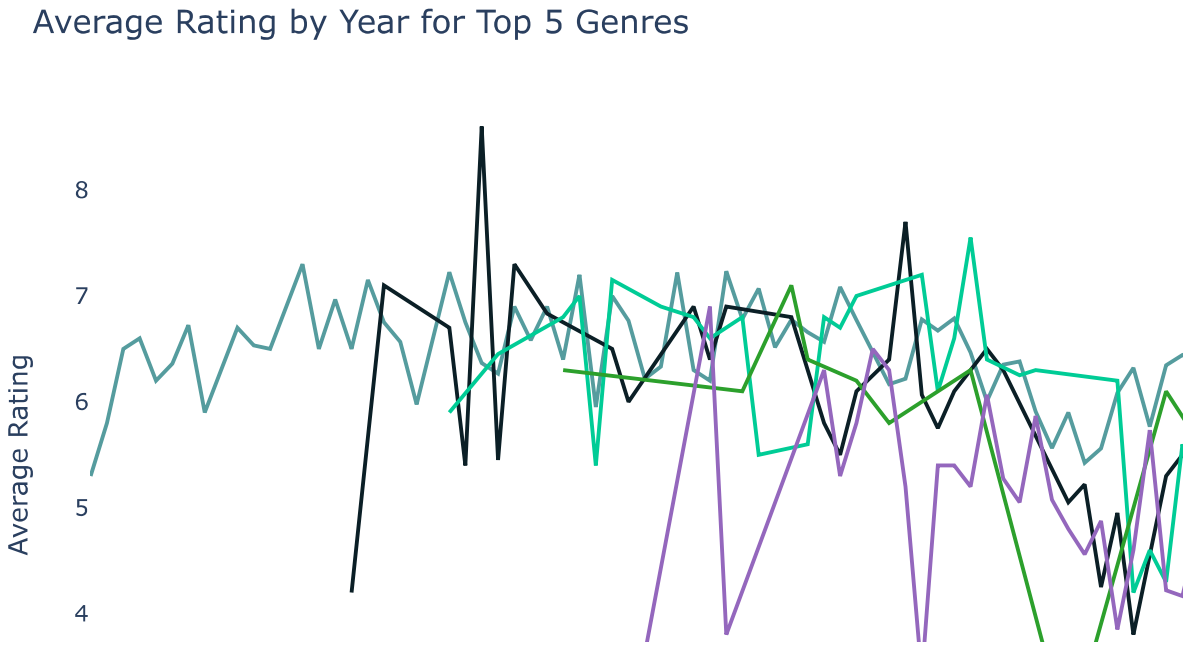


In [71]:
```
# Group data by Year and calculate the average rating
average_rating_by_year = df.groupby('Year')['Votes'].mean().reset_index()

# Create the line plot with Plotly Express
fig = px.line(average_rating_by_year, x='Year', y='Votes', color_discrete_sequence=['#
fig.update_layout(title='Are there any trends in votes across year?', title_x=0.5, tit
fig.show()
```

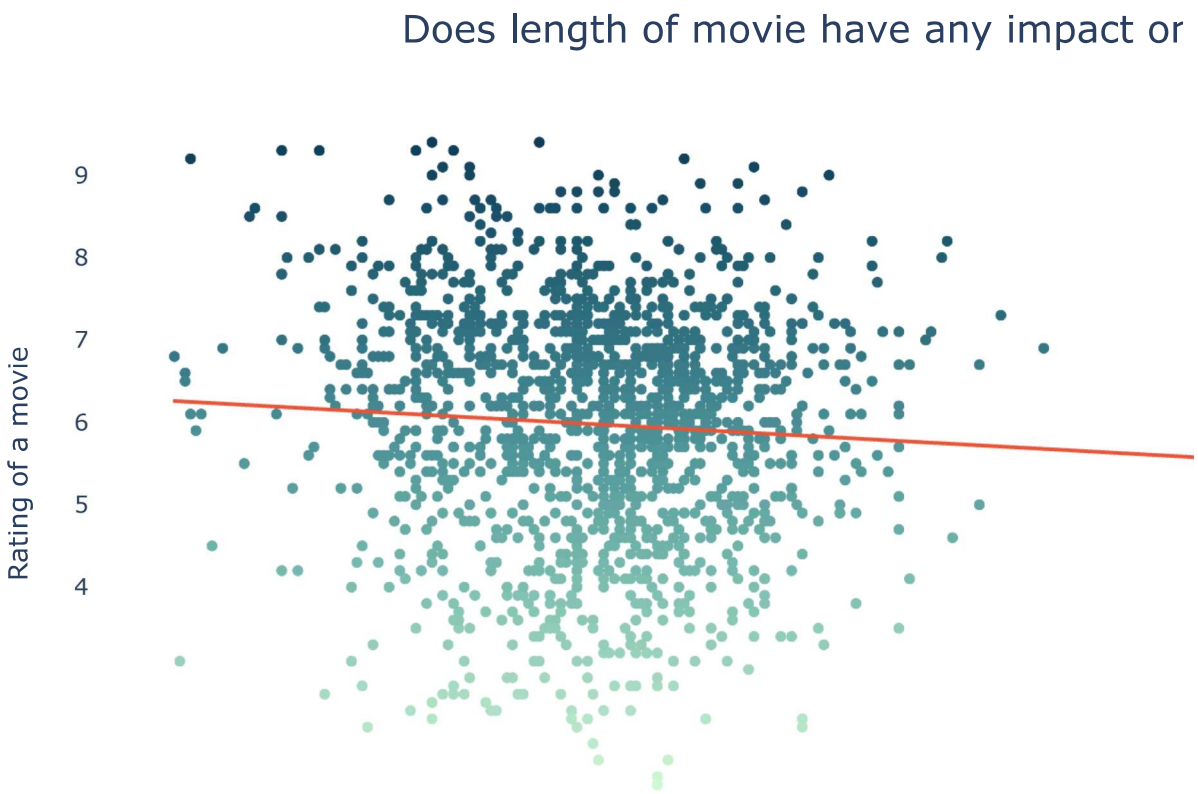
In [73]:
```
# Group data by Year and calculate the average rating
average_rating_by_year = df.groupby(['Year', 'Genre'])['Rating'].mean().reset_index()

# Get the top 5 genres
top_5_genres = df['Genre'].value_counts().head(5).index

# Filter the data to include only the top 5 genres
average_rating_by_year = average_rating_by_year[average_rating_by_year['Genre'].isin(t

# Create the line plot with Plotly Express
fig = px.line(average_rating_by_year, x='Year', y='Rating', color = "Genre", color_dis

# Customize the layout
fig.update_layout(title='Average Rating by Year for Top 5 Genres', xaxis_title='Year',
```



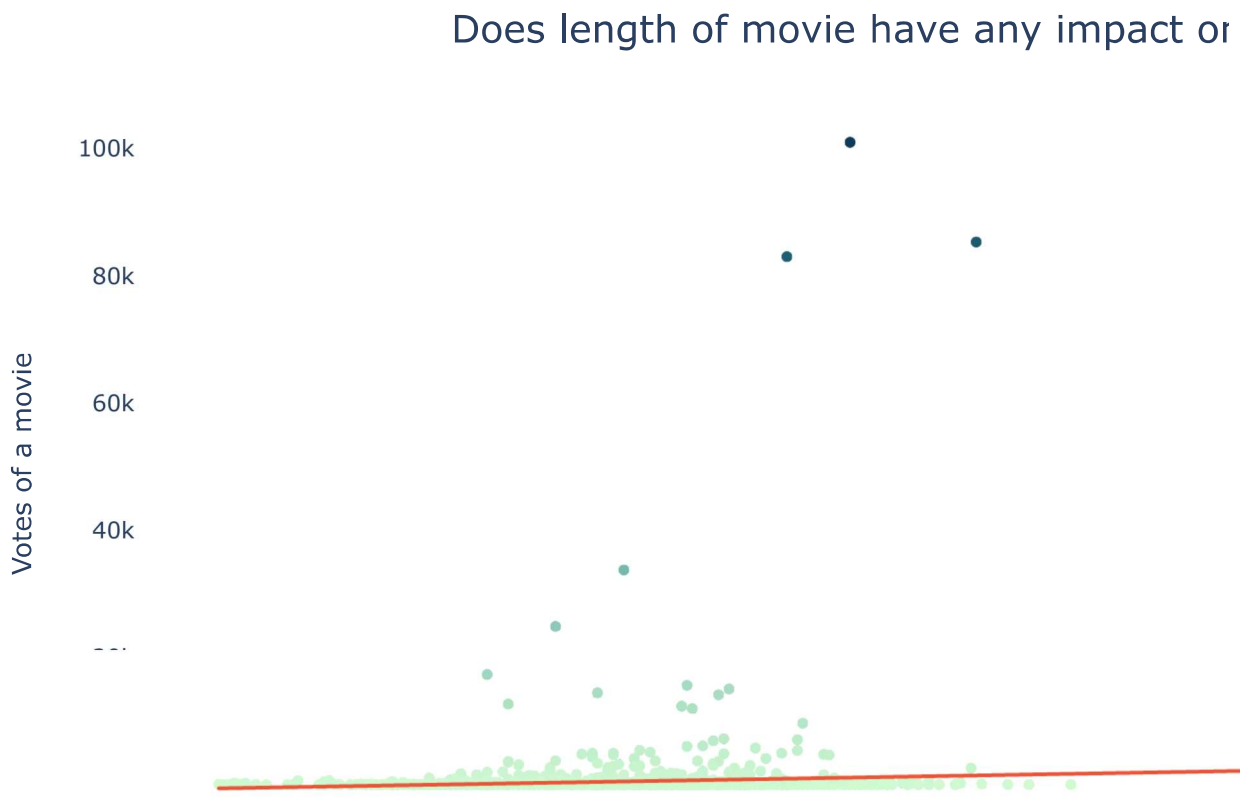
In [74]:
```
fig.show()
```



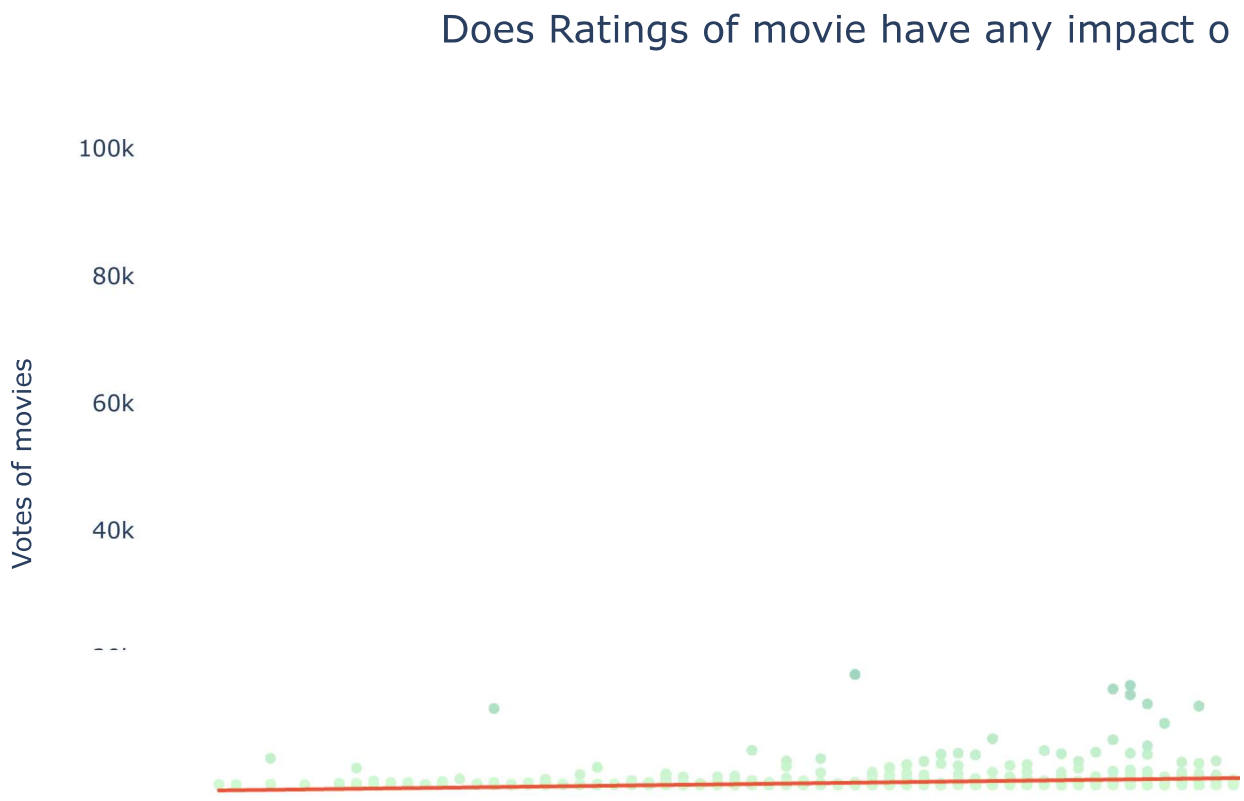
In [75]:
```
fig_dur_rat = px.scatter(df, x = 'Duration', y = 'Rating', trendline='ols', color = "R
fig_dur_rat.update_layout(title='Does length of movie have any impact on rating?', tit
fig_dur_rat.show()
```



In [76]:
```
fig_dur_votes = px.scatter(df, x = 'Duration', y = 'Votes', trendline='ols', color = "
fig_dur_votes.update_layout(title='Does length of movie have any impact on Votes?', ti
fig_dur_votes.show()
```



In [77]:
```
fig_rat_votes = px.scatter(df, x = 'Rating', y = 'Votes', trendline='ols', color = "Vo
fig_rat_votes.update_layout(title='Does Ratings of movie have any impact on Votes?', t
fig_rat_votes.show()
```



In [78]:
```
df.drop('Name', axis = 1, inplace = True)
# Grouping the columns with their average rating and then creating a new feature

genre_mean_rating = df.groupby('Genre')['Rating'].transform('mean')
df['Genre_mean_rating'] = genre_mean_rating

director_mean_rating = df.groupby('Director')['Rating'].transform('mean')
df['Director_encoded'] = director_mean_rating

actor1_mean_rating = df.groupby('Actor 1')['Rating'].transform('mean')
df['Actor1_encoded'] = actor1_mean_rating

actor2_mean_rating = df.groupby('Actor 2')['Rating'].transform('mean')
df['Actor2_encoded'] = actor2_mean_rating

actor3_mean_rating = df.groupby('Actor 3')['Rating'].transform('mean')
df['Actor3_encoded'] = actor3_mean_rating
# Keeping the predictor and target variable

X = df[[ 'Year', 'Votes', 'Duration', 'Genre_mean_rating','Director_encoded','Actor1_e
y = df['Rating']
# Splitting the dataset into training and testing parts

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=20)
```

In [79]:
```
lr = LinearRegression()
lr.fit(X_train,y_train)
lr_pred = lr.predict(X_test)
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
rf_pred = rf.predict(X_test)
# Evaluating the performance of trained algos

print('The performance evaluation of Logistic Regression is below: ', '\n')
print('Mean squared error: ',mean_squared_error(y_test, lr_pred))
print('Mean absolute error: ',mean_absolute_error(y_test, lr_pred))
print('R2 score: ',r2_score(y_test, lr_pred))
print('\n', '='*100, '\n')

print('The performance evaluation of Random Forest Regressor is below: ', '\n')
print('Mean squared error: ',mean_squared_error(y_test, rf_pred))
print('Mean absolute error: ',mean_absolute_error(y_test, rf_pred))
print('R2 score: ',r2_score(y_test, rf_pred))
```

```
The performance evaluation of Logistic Regression is below:

Mean squared error:  0.1397803872882309
Mean absolute error:  0.27146950876584425
R2 score:  0.926720988593957

 ====================================================================================
================

The performance evaluation of Random Forest Regressor is below:

Mean squared error:  0.11856241503267971
Mean absolute error:  0.19207516339869374
R2 score:  0.9378443805167561
```

In [80]:
```
X.head()
```

Out[80]:

	Year	Votes	Duration	Genre_mean_rating	Director_encoded	Actor1_encoded	Actor2_encoded	Actor3
1	2019	8	109	6.420152	7.000	6.850000	7.000000	
10	2004	17	96	6.420152	6.200	5.766667	5.100000	
11	2016	59	120	4.698529	5.900	5.900000	5.900000	
30	2005	1002	116	6.420152	6.525	6.900000	6.866667	
32	1993	15	168	6.420152	5.400	5.600000	6.400000	

In [81]:
```
y.head()
```

Out[81]:
```
1     7.0
10    6.2
11    5.9
30    7.1
32    5.6
Name: Rating, dtype: float64
```

In [82]:
```
data = {'Year': [2005], 'Votes': [1002], 'Duration': [116], 'Genre_mean_rating': [6.4]
df = pd.DataFrame(data)
predicted_rating_lr = rf.predict(df)

# Display the predicted rating
print("Predicted Rating:", predicted_rating_lr[0])
```

```
Predicted Rating: 6.877000000000007
```

In [83]:
```
predicted_rating_rf = rf.predict(df)

# Display the predicted rating
print("Predicted Rating:", predicted_rating_rf[0])
```

```
Predicted Rating: 6.877000000000007
```

In [ ]: