- INTRODUCTION :-

In the digital age, personalized movie recommendations are crucial for enhancing user experience on streaming platforms. A movie rating prediction model uses machine learning to estimate the ratings users might give to unseen movies, thereby enabling tailored suggestions. By analyzing patterns in user behavior and movie features, these models can effectively predict preferences. This project aims to develop such a model, utilizing collaborative filtering and content-based techniques, to provide accurate and personalized movie recommendations. The process involves data collection, preprocessing, feature engineering, model training, and evaluation to achieve optimal prediction accuracy.

In [1]:
```
# import necessary libraries required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.neighbors import KNeighborsRegressor
```

In [2]:
```
# read the dataset into a dataframe
df = pd.read_csv("C:\\Users\\DELL\\Downloads\\Movie dataset.csv",encoding='latin1')
# show first five records of dataframe
df.head()
```

Out[2]:

	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)	-2019.0	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid
2	#Homecoming	-2021.0	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borthakur	Roy Angana
3	#Yaaram	-2019.0	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
4	...And Once Again	-2010.0	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Rituparna Sengupta	Antara Mali

3)Data Preprocessing :

In [3]:
```
# show the number of records and observations in the dataframe
df.shape
```

Out[3]:
```
(15509, 10)
```

In [4]:
```
# check out the information on the dataframe
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  float64
 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(2), object(8)
memory usage: 1.2+ MB
```

In [5]:
```
# check out the missing values in each observation
df.isna().sum()
```

Out[5]:
```
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 [6]:
```
# drop records with missing value in any of the following columns: Name, Year, Duration, Votes, Rating
df.dropna(subset=['Name', 'Year', 'Duration', 'Votes', 'Rating'], inplace=True)
df
```

Out[6]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
1	#Gadhvi (He thought he was Gandhi)	-2019.0	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid
3	#Yaaram	-2019.0	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
5	...Aur Pyaar Ho Gaya	-1997.0	147 min	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor
6	...Yahaan	-2005.0	142 min	Drama, Romance, War	7.4	1,086	Shoojit Sircar	Jimmy Sheirgill	Minissha Lamba	Yashpal Sharma
8	?: A Question Mark	-2012.0	82 min	Horror, Mystery, Thriller	5.6	326	Allyson Patel	Yash Dave	Muntazir Ahmad	Kiran Bhatia
...	...	...	...	...	...	...	...	...	...	...
15493	Zubaan	-2015.0	115 min	Drama	6.1	408	Mozez Singh	Vicky Kaushal	Sarah Jane Dias	Raaghavv Chanana
15494	Zubeidaa	-2001.0	153 min	Biography, Drama, History	6.2	1,496	Shyam Benegal	Karisma Kapoor	Rekha	Manoj Bajpayee
15503	Zulm Ki Zanjeer	-1989.0	125 min	Action, Crime, Drama	5.8	44	S.P. Muthuraman	Chiranjeevi	Jayamalini	Rajinikanth
15505	Zulmi	-1999.0	129 min	Action, Drama	4.5	655	Kuku Kohli	Akshay Kumar	Twinkle Khanna	Aruna Irani
15508	Zulm-O-Sitam	-1998.0	130 min	Action, Drama	6.2	20	K.C. Bokadia	Dharmendra	Jaya Prada	Arjun Sarja

5851 rows × 10 columns

In [7]:
```
# remove rows with duplicate movie records
df.drop_duplicates(subset=['Name', 'Year', 'Director'], keep='first', inplace=True)
```

In [8]:
```
# remove () from the Year column values and change the datatype to integer
df['Year'] = df['Year'].astype(int)
```

In [9]:
```
# remove minutes from the Duration column values
df['Duration'] = df['Duration'].str.replace(r' min', '').astype(int)
```

In [10]:
```
# remove commas from Votes column and convert to integer
df['Votes'] = df['Votes'].str.replace(',', '').astype(int)

# show the number of records and observations after cleaning the dataframe
df.shape
```

Out[10]:
```
(5850, 10)
```

In [11]:
```
# show the info on the cleaned dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5850 entries, 1 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Name      5850 non-null   object 
 1   Year      5850 non-null   int32  
 2   Duration  5850 non-null   int32  
 3   Genre     5819 non-null   object 
 4   Rating    5850 non-null   float64
 5   Votes     5850 non-null   int32  
 6   Director  5849 non-null   object 
 7   Actor 1   5775 non-null   object 
 8   Actor 2   5733 non-null   object 
 9   Actor 3   5687 non-null   object 
dtypes: float64(1), int32(3), object(6)
memory usage: 434.2+ KB
```

In [12]:
```
# show the statistics of the dataframe
df.describe()
```

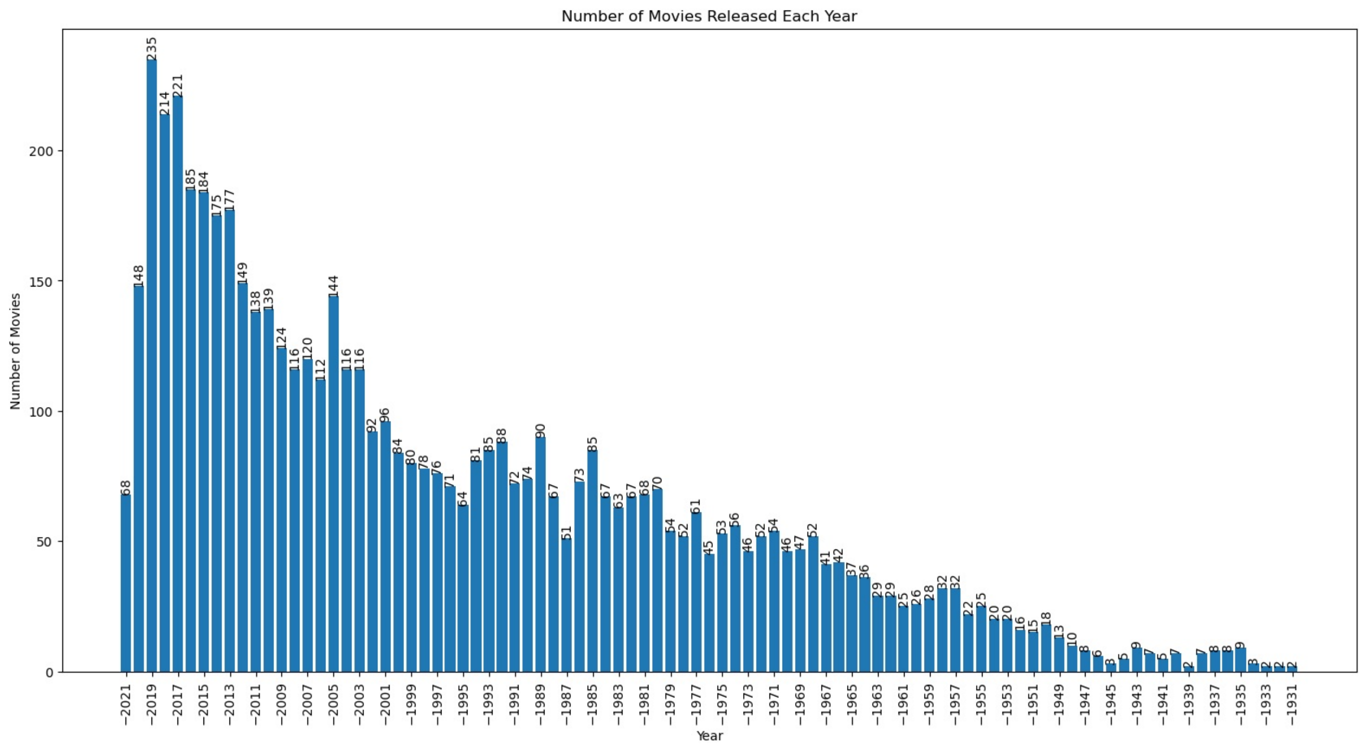
Out[12]:

	Year	Duration	Rating	Votes
count	5850.000000	5850.000000	5850.000000	5850.000000
mean	-1996.426496	132.293675	5.931504	2611.717949
std	19.902673	26.558025	1.389772	13434.933770
min	-2021.000000	21.000000	1.100000	5.000000
25%	-2013.000000	117.000000	5.000000	28.000000
50%	-2002.000000	134.000000	6.100000	119.000000
75%	-1983.000000	150.000000	7.000000	862.750000
max	-1931.000000	321.000000	10.000000	591417.000000

4)Exploratory Data Analysis (EDA):

i.Number of Movies each Year

In [13]:
```
# group the data by Year and count the number of movies in each year
yearly_movie_counts = df['Year'].value_counts().sort_index()

# create a bar chart
plt.figure(figsize=(18, 9))
plt.bar(yearly_movie_counts.index, yearly_movie_counts.values, color='tomato')
plt.xlabel('Year')
plt.ylabel('Number of Movies')
plt.title('Number of Movies Released Each Year')

# Show every second year on the x-axis and rotate x-labels for better readability
plt.xticks(yearly_movie_counts.index[::2], rotation=90)
bars=plt.bar(yearly_movie_counts.index, yearly_movie_counts.values)

for bar in bars:
    xval = bar.get_x() + bar.get_width() / 2
    yval = bar.get_height()
    plt.text(xval, yval, int(yval), ha='center', va='bottom', rotation= 90)

plt.show()
```



ii)Creating Genre Dummy Columns and Analyzing Movie Counts by Genre

In [14]:
```
# create dummy columns for each genre
dummies = df['Genre'].str.get_dummies(', ')
# creating a new dataframe which combines df and dummies
df_genre = pd.concat([df, dummies], axis=1)
df_genre
```

Out[14]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3	...	Music	Musical
1	#Gadhvi (He thought he was Gandhi)	-2019	109	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid	...	0	0
3	#Yaaram	-2019	110	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor	...	0	0
5	...Aur Pyaar Ho Gaya	-1997	147	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor	...	0	1
6	...Yahaan	-2005	142	Drama, Romance, War	7.4	1086	Shoojit Sircar	Jimmy Sheirgill	Minissha Lamba	Yashpal Sharma	...	0	0
8	?: A Question Mark	-2012	82	Horror, Mystery, Thriller	5.6	326	Allyson Patel	Yash Dave	Muntazir Ahmad	Kiran Bhatia	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
15493	Zubaan	-2015	115	Drama	6.1	408	Mozez Singh	Vicky Kaushal	Sarah Jane Dias	Raaghavv Chanana	...	0	0
15494	Zubeidaa	-2001	153	Biography, Drama, History	6.2	1496	Shyam Benegal	Karisma Kapoor	Rekha	Manoj Bajpayee	...	0	0
15503	Zulm Ki Zanjeer	-1989	125	Action, Crime, Drama	5.8	44	S.P. Muthuraman	Chiranjeevi	Jayamalini	Rajinikanth	...	0	0
15505	Zulmi	-1999	129	Action, Drama	4.5	655	Kuku Kohli	Akshay Kumar	Twinkle Khanna	Aruna Irani	...	0	0
15508	Zulm-O-Sitam	-1998	130	Action, Drama	6.2	20	K.C. Bokadia	Dharmendra	Jaya Prada	Arjun Sarja	...	0	0

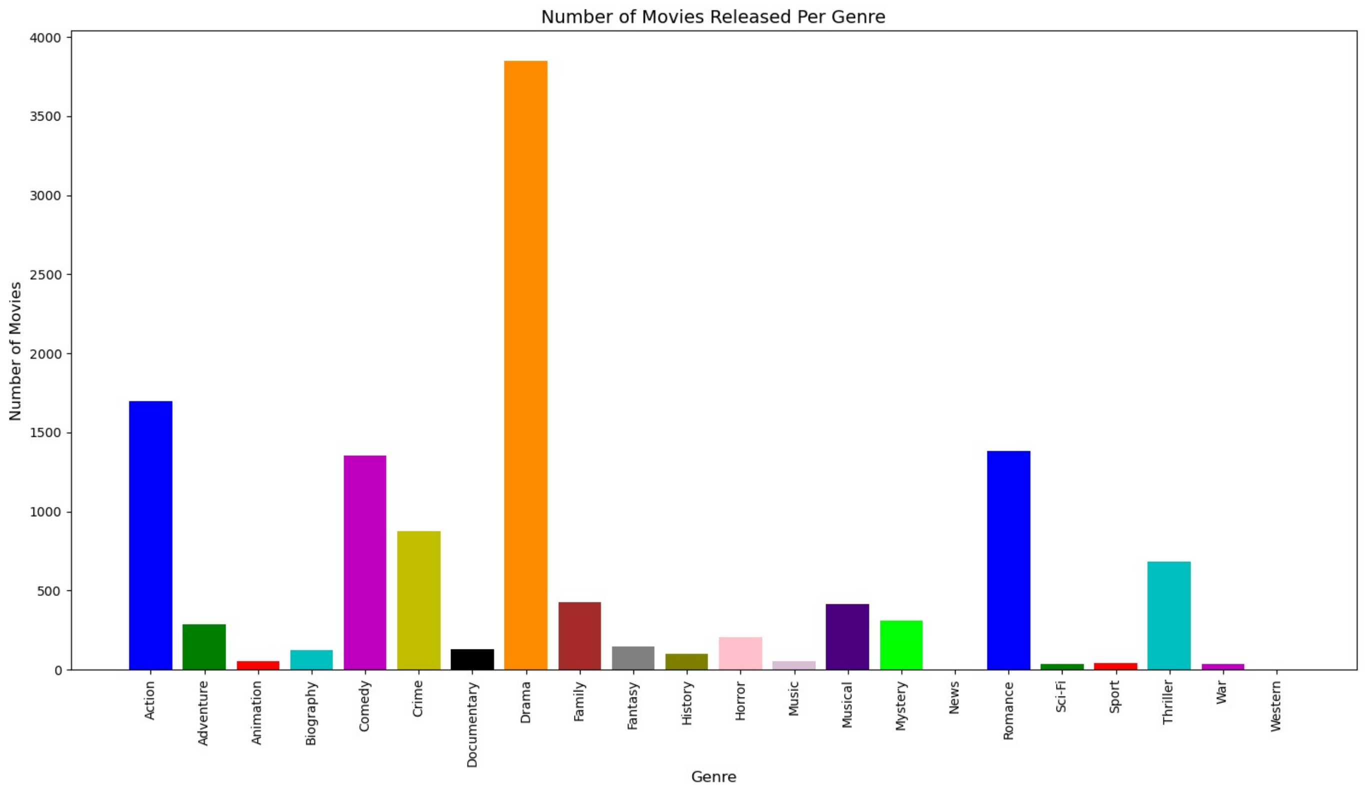
5850 rows × 32 columns

In [15]:
```
genre_columns = df_genre.columns[10:]  # Assuming genre columns start from the 11th column
genre_columns
```

```
Out[15]:  Index(['Action', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime',
                 'Documentary', 'Drama', 'Family', 'Fantasy', 'History', 'Horror',
                 'Music', 'Musical', 'Mystery', 'News', 'Romance', 'Sci-Fi', 'Sport',
                 'Thriller', 'War', 'Western'],
                dtype='object')
```

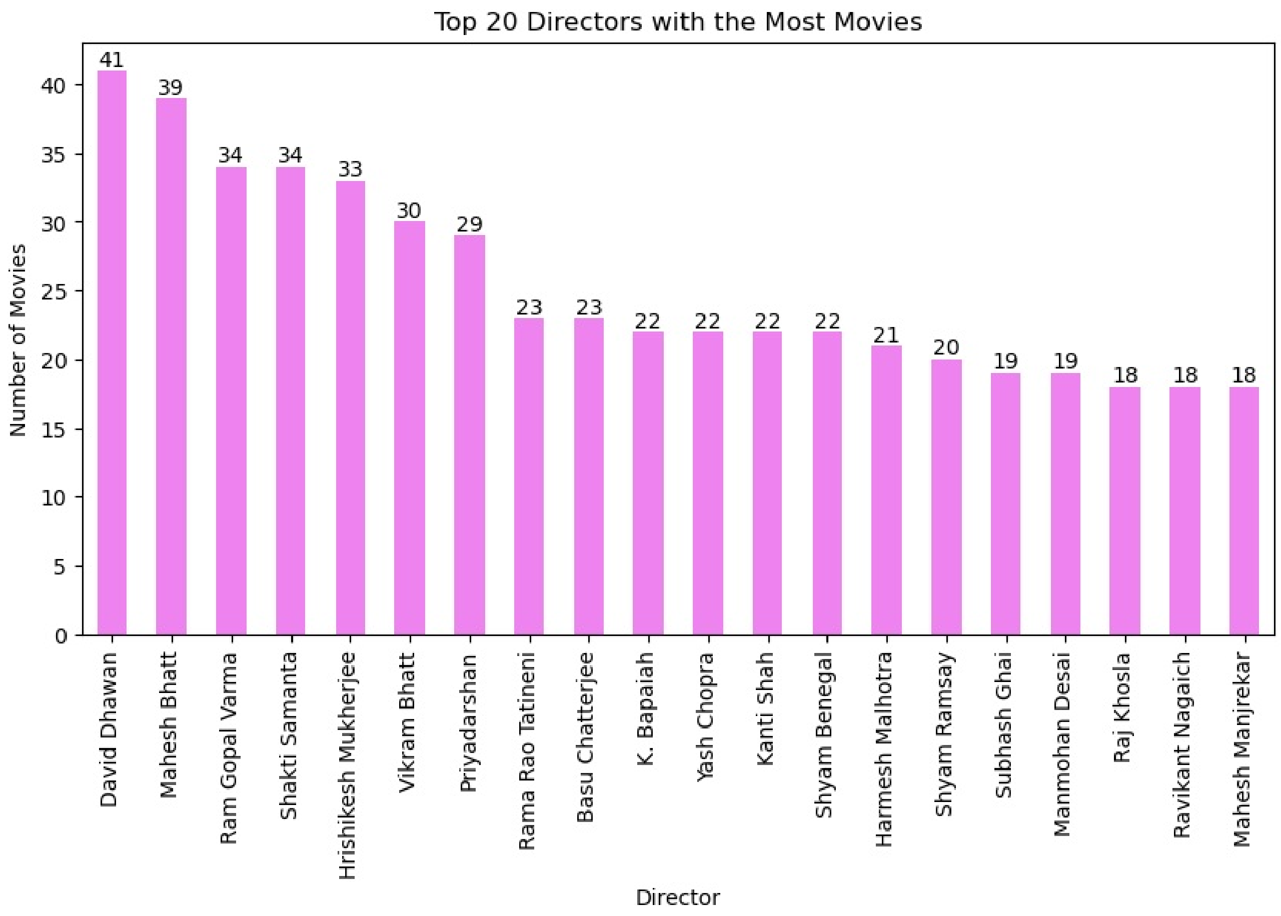
```
In [16]:  # group the data by genre_columns and count the number of movies in each genre
          genre_movie_counts = df_genre[genre_columns].sum().sort_index()
```

```
In [17]:  plt.figure(figsize=(18, 9))
          plt.bar(genre_movie_counts.index, genre_movie_counts.values, color=('b','g','r','c','m','y','k','darkorange','b
          plt.xlabel('Genre',fontsize=12)
          plt.ylabel('Number of Movies',fontsize=12)
          plt.title('Number of Movies Released Per Genre',fontsize=14)

          plt.xticks(rotation=90)

          plt.show()
```



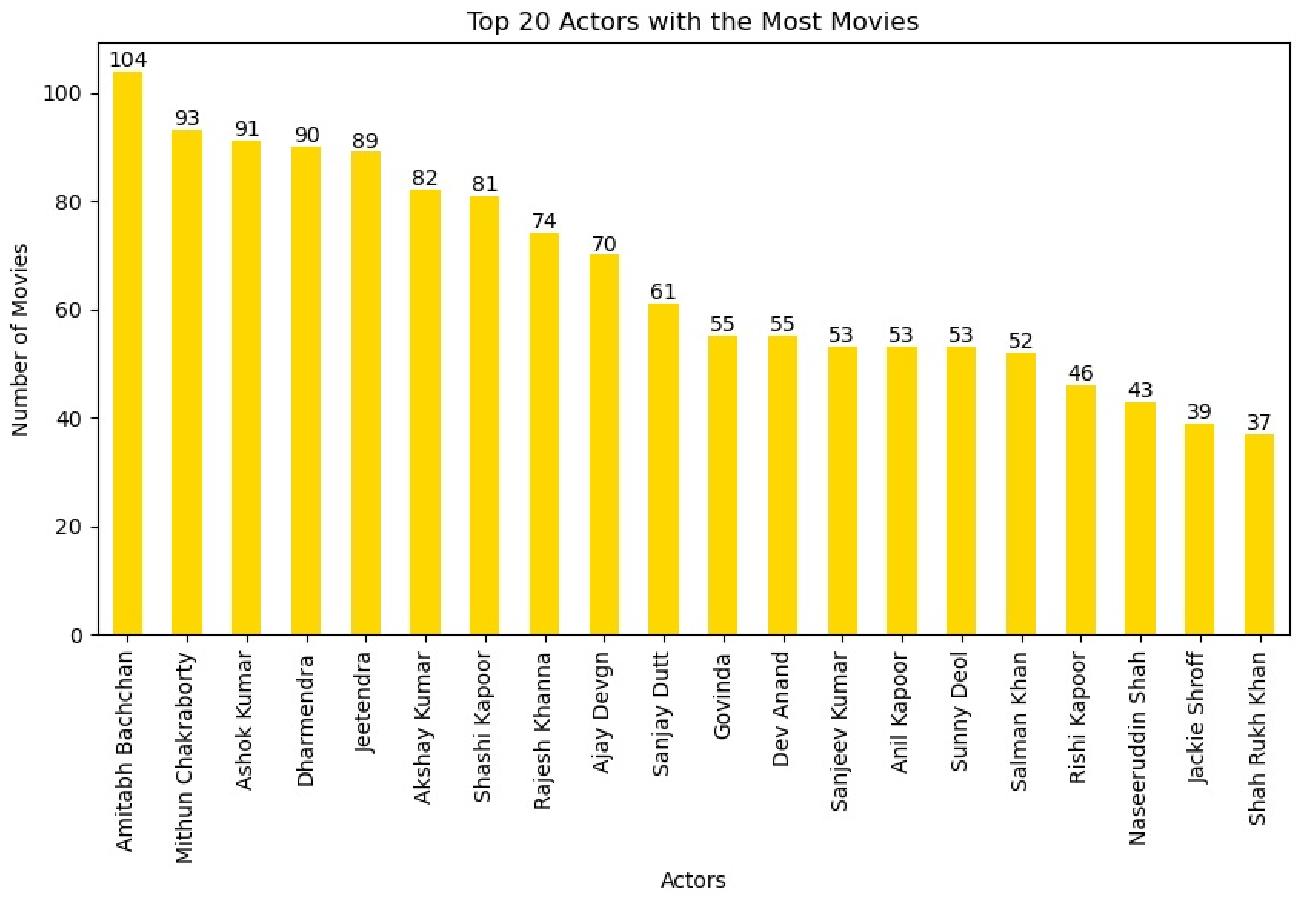
iii. Top 20 Directors with the Most Movies:

```
In [18]:  # Analyzing count of movies of each director
          director_movie_counts = df['Director'].value_counts()

          # Create a bar chart
          plt.figure(figsize=(10, 5))
          bars = director_movie_counts.head(20).plot(kind='bar', color='violet')
          plt.xlabel('Director')
          plt.ylabel('Number of Movies')
          plt.title('Top 20 Directors with the Most Movies')
          plt.xticks(rotation=90)

          # Add count labels on top of the bars
          for bar in bars.patches:
              xval = bar.get_x() + bar.get_width() / 2
              yval = bar.get_height()
              plt.text(xval, yval, int(yval), ha='center', va='bottom')

          plt.show()
```



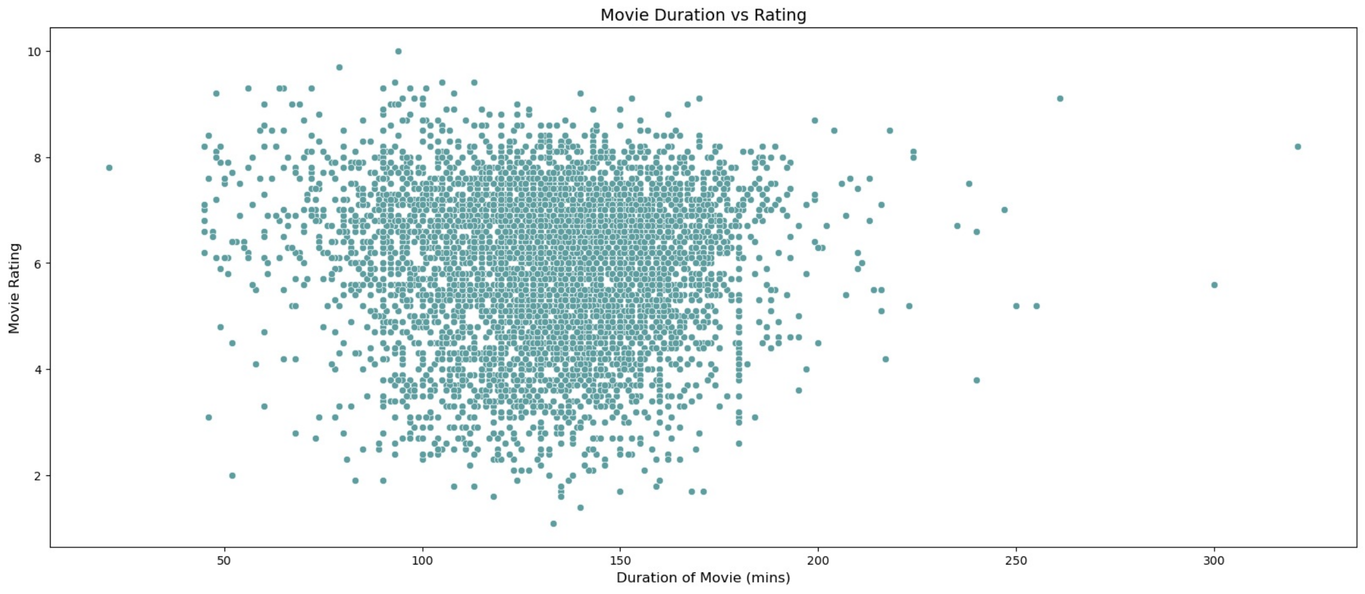
iv.Top 20 Actors with the Most Movies:

In [19]:
```
# To Count Top 20 movies for each actor
actor_movie_counts = df['Actor 1'].value_counts()

# Create a bar chart
plt.figure(figsize=(10, 5))
actor_movie_counts.head(20).plot(kind='bar', color='gold')
plt.xlabel('Actors')
plt.ylabel('Number of Movies')
plt.title('Top 20 Actors with the Most Movies')
plt.xticks(rotation=90)

# Add count labels on top of the bars
for i, v in enumerate(actor_movie_counts.head(20)):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.show()
```



v.Movie Duration vs. Rating Scatter plot :

In [20]:
```
plt.figure(figsize=(20, 8))
# create a scatter plot with Duration and Rating relationship
sns.scatterplot(x=df['Duration'], y=df['Rating'],  color = 'cadetblue')
plt.xlabel('Duration of Movie (mins)',fontsize=12)
plt.ylabel('Movie Rating',fontsize=12)
plt.title('Movie Duration vs Rating',fontsize=14)
plt.show()
```



In [40]:
```
import pandas as pd
from sklearn.feature_selection import chi2
```

5.Feature Analysis:

In [21]:
```
# dropping the columns from the dataframe since these are the least dependable observations for target variable
df.drop(['Name','Director','Actor 1','Actor 2','Actor 3'], axis=1,inplace=True)
# show first five records of the dataframe
df.head()
```

Out[21]:

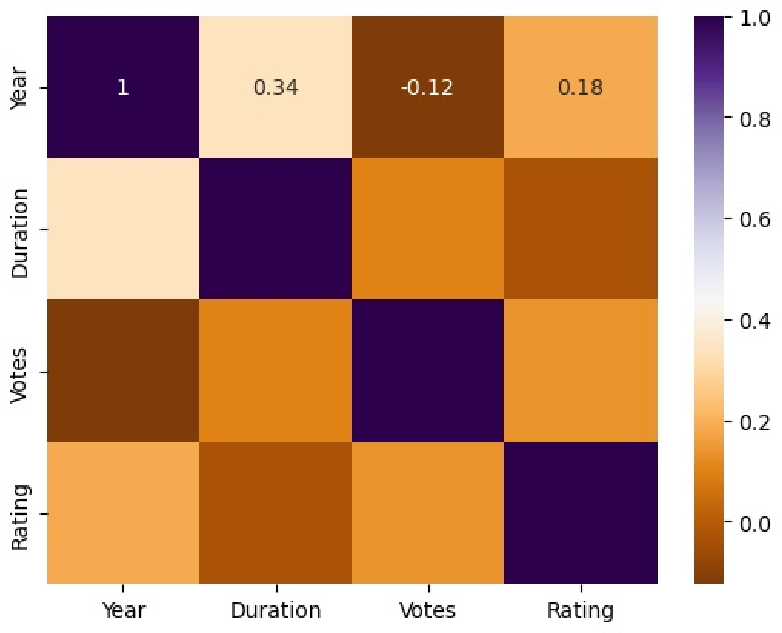
	Year	Duration	Genre	Rating	Votes
1	-2019	109	Drama	7.0	8
3	-2019	110	Comedy, Romance	4.4	35
5	-1997	147	Comedy, Drama, Musical	4.7	827
6	-2005	142	Drama, Romance, War	7.4	1086
8	-2012	82	Horror, Mystery, Thriller	5.6	326

In [29]:

```
df_1=df[['Year','Duration','Votes','Rating']]
X=df_1.corr()
sns.heatmap(X,cmap='PuOr',annot=True)
```

Out[29]:

```
<Axes: >
```



In [23]:

```
# creating target variable and learning observations for the model
X = df[['Year','Duration','Votes']]
y = df['Rating']

# split 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_state=231)
```

6.Machine Learning Modeling Techniques :

i.linear Regression Model :

In [24]:

```
# creating a liner regression model
lr = LinearRegression()

# training the data on linear regression model
lr.fit(X_train, y_train)

# predicting the test data on trained model
pred = lr.predict(X_test)

# evaluating linear regression model
r2_score(y_test,pred)
```

Out[24]:

```
0.008207910636609306
```

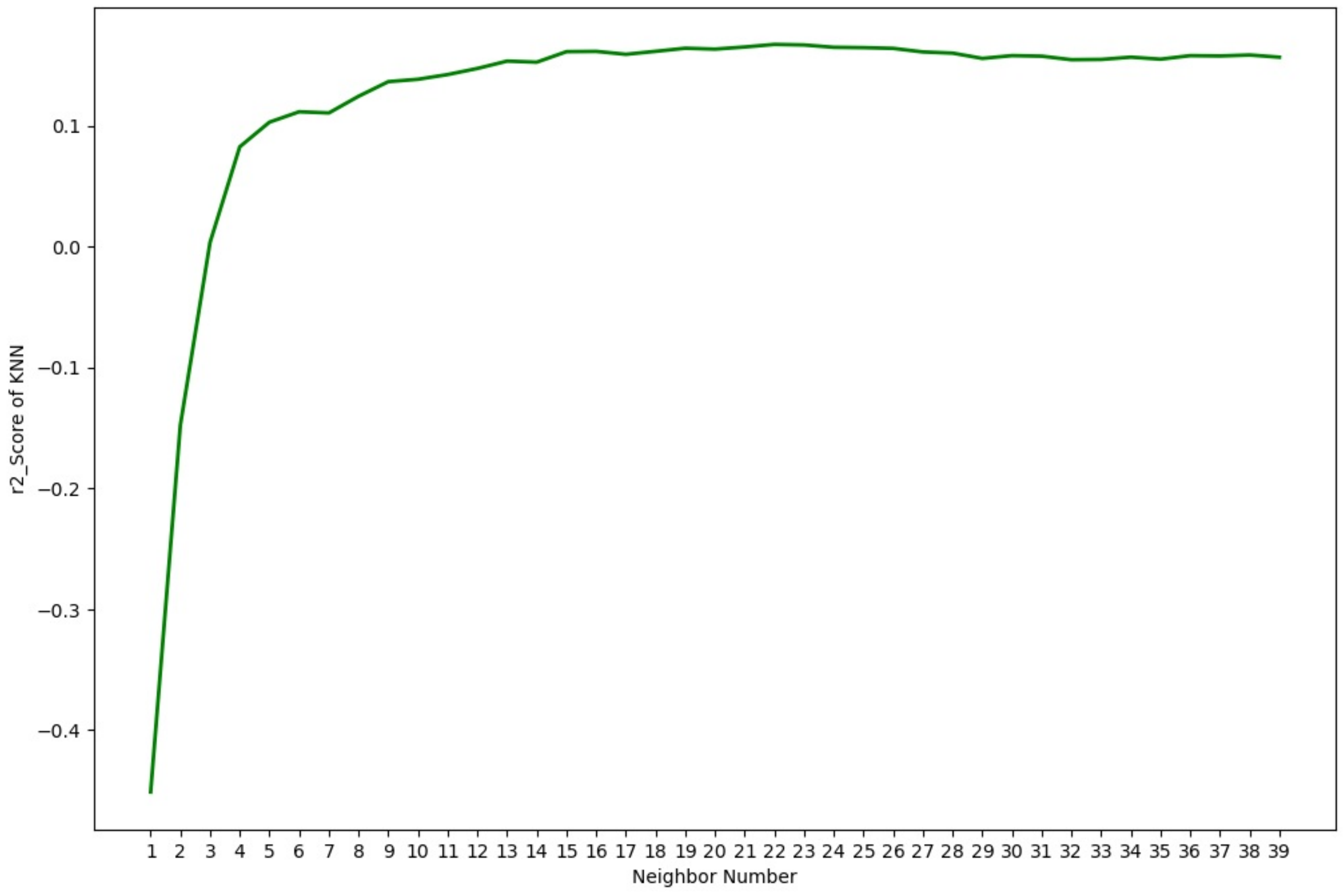
- interpretation:
  If the value of r2_score=0.008207910636609306 then the given model is good to fit for the given data.

ii.K- Nearest Neighbors (KNN) Regression Model :

In [56]:

```
# creating a range for number of neighbors parameter of the KNN model
kRange = range(1,40,1)

# creating an empty scores list
scores_list = []

# iterate every value in kRange list
for i in kRange:
    # create a K Nearest Neighbor model with i as number of neighbors
    regressor_knn = KNeighborsRegressor(n_neighbors = i)

    # fit training data to the KNN model
    regressor_knn.fit(X_train,y_train)
    # evaluate the model
    pred = regressor_knn.predict(X_test)

    # append the regression score for evaluation of the model to scores_list
    scores_list.append(r2_score(y_test,pred))
```

In [57]:
```
plt.figure(figsize=(12,8))
# create a line graph for showing regression score (scores_list) for respective number of neighbors used in the
plt.plot(kRange, scores_list, linewidth=2, color='green')
# values for x-axis should be the number of neighbors stored in kRange
plt.xticks(kRange)
plt.xlabel('Neighbor Number')
plt.ylabel('r2_Score of KNN')
plt.show()
```



In [58]:
```
# Creating a KNN model with best parameters i.e., number of neighbors = 23
regressor_knn = KNeighborsRegressor(n_neighbors = 23)

# fit training data to the KNN model
regressor_knn.fit(X_train,y_train)
# evaluate test data on the model
pred = regressor_knn.predict(X_test)
# show regression score
r2_score(y_test,pred)
```

Out[58]:
```
0.16686929930648098
```

iii.SGD (Stochastic Gradient Descent) Regression :

In [59]:
```
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import r2_score

# Create an instance of the SGDRegressor
sgd_regressor = SGDRegressor(max_iter=100, random_state=1)  # You can adjust the max_iter and random_state

# Fit the model to your training data
sgd_regressor.fit(X_train, y_train)

# Make predictions
pred = sgd_regressor.predict(X_test)

# Evaluate the model
```

```
r2 = r2_score(y_test, pred)

print("R-squared score:", r2)
```

```
R-squared score: -5.130041433561575e+32
```

iv.Random Forest Regression:

In [60]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

rf_regressor = RandomForestRegressor(n_estimators=100, random_state=1)
rf_regressor.fit(X_train, y_train)
rf_pred = rf_regressor.predict(X_test)
r2_rf = r2_score(y_test, rf_pred)
print(f'R-squared score (Random Forest): {r2_rf}')
```

```
R-squared score (Random Forest): 0.15491173885508136
```

v.Gradoemt Boosting Regression:

In [61]:
```
from sklearn.ensemble import GradientBoostingRegressor
gb_regressor = GradientBoostingRegressor(n_estimators=100, random_state=231)
gb_regressor.fit(X_train, y_train)
gb_pred = gb_regressor.predict(X_test)
r2_gb = r2_score(y_test, gb_pred)
print(f'R-squared score: {r2_gb}')
```

```
R-squared score: 0.2487452215098106
```

- CONCLUSION :-

Based on the R² scores obtained from various regression models applied to the dataset, we observe that the R² score is lowest for the Linear Regression model (R² = 0.0082). Therefore, we conclude that the given Linear Regression model is well-fitted for predicting movie ratings.

In [ ]: