IMDB Movie Data Analysis

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
data=pd.read_csv('datasets/movies.csv')
```

data exploration

In [ ]:
```
data.head()
```

In [ ]:
```
data.info()
```

remvoing columns we dont need

In [ ]:
```
data.drop(['Poster','Movie URL'],axis=1,inplace=True)
```

checking for null values

In [ ]:
```
data.isnull().sum()
```

In [ ]:
```
data=data.dropna()
```

checking for duplicates

In [ ]:
```
dupli=data.duplicated()
sum(dupli)
```

there are no duplicates

In [ ]:
```
data.info()
```

removing outliers

In [ ]:
```
data.describe().round()
```

In [ ]:
```
sns.set(style='whitegrid',palette='deep')
sns.boxplot(x=data['Length in Min'])
```

considering movies with runtime less than 4 hours

In [ ]:
```
data = data[data['Length in Min']<240]
```

Analysis

In [ ]:
```
g=data.groupby('Release Year').agg({'IMDB Rating':'mean','Rating Count':'sum',
g
```

Number of movies released over the years

In [ ]:
```
sns.lineplot(x=g.index,y='Title',data=g)
plt.ylabel('No. of movies released')
plt.title('Number of movies released over the years')
```

IMDB rating over the years

In [ ]:
```
sns.lineplot(x=g.index,y='IMDB Rating',data=g)
plt.title('IMDB Rating over the years')
```

rating count over the years

In [ ]:
```
sns.lineplot(x=g.index,y='Rating Count',data=g)
plt.title('number of ratings over the years')
plt.ylim(0, g['Rating Count'].max() + 100)
plt.ylabel('rating count in M')
```

length of movies over the years

In [ ]:
```
sns.lineplot(x=g.index,y='Length in Min',data=g)
```

correlation analysis

In [ ]:
```
plt.figure(figsize=(3,3))
corr=data.iloc[:,1:5].corr()
sns.heatmap(corr,cmap='viridis')
```

there isnt much correlation among the variables

Genres with highest number of movies

In [ ]:
```
genres=data.Genres.str.split("|",expand=True)
genres.head(1)
```

In [ ]:
```
melted_df = genres.melt(value_vars=[0, 1, 2], value_name='genre')
melted_df = melted_df.dropna()
melted_df
genre_counts = melted_df['genre'].value_counts().head(10)
genre_counts.plot(kind='bar',edgecolor='k')
```

director with most movies

In [ ]:
```
directors=data.Directors.str.split("|",expand=True)
directors.head(3)
```

creating single list

In [ ]:
```
melted_df = directors.melt(value_vars=[0, 1, 2, 3], value_name='director')
melted_df = melted_df.dropna()
melted_df = melted_df[melted_df['director'] != '']
```

getting count of directors

In [ ]:
```
director_counts = melted_df['director'].value_counts().head(10)
```

In [ ]:
```
director_counts.plot(kind='bar',edgecolor='k')
plt.ylim(0, 130)
plt.title('director with highest number of movies')
```

directors with highest mean rating count

In [ ]:
```
data['Directors'] = data['Directors'].str.split('|').apply(sorted)
data['Directors'] = data['Directors'].apply(', '.join)
```

In [ ]:
```
# Filter directors with at least 10 movies and calculate mean rating count
director_ratings = data.groupby('Directors').filter(lambda x: x['Title'].count
director_ratings = director_ratings.groupby('Directors')['Rating Count'].mean(

# Select top 10 directors with highest mean rating count
top_10_directors = director_ratings.sort_values(ascending=False).head(10)
top_10_directors.plot(kind='bar',edgecolor='k')
plt.title('top 10 directors with highest mean rating count')
```

top 10 highest rated movies

In [ ]:
```
sorted_df = data.sort_values(by='IMDB Rating', ascending=False)
sorted_df=sorted_df.head(10)
sorted_df[['Title','Release Year','IMDB Rating']]
```

In [ ]:
```
data
```

In [ ]:
```
df = data.copy()
all_stars = df['Stars'].str.split('|',expand=True)
x=all_stars.iloc[:,:3]
df_comb= pd.concat([df[['Release Year', 'Length in Min']], x], axis=1)
df_comb.head(2)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder


label_encoder = LabelEncoder()
df_comb[0] = label_encoder.fit_transform(df_comb[0])
df_comb[1] = label_encoder.fit_transform(df_comb[1])
df_comb[2] = label_encoder.fit_transform(df_comb[2])
df_comb
```

In [ ]:
```
y = pd.Series(df['IMDB Rating'])
y.shape
```

In [ ]:
```

X_train, X_test, y_train, y_test = train_test_split(df_comb, y, test_size=0.2,
model = LinearRegression()
model.fit(X_train, y_train)

test_r2 = r2_score(y_test, model.predict(X_test))
```

In [ ]:

In [ ]: