Import Library

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('IMDb Movies India.csv', encoding='latin-1')
```

```
df
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Act
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	E
1	#Gadhvi (He thought he was Gandhi)	(2019)	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Gham
2	#Homecoming	(2021)	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borth
3	#Yaaram	(2019)	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita
4	...And Once Again	(2010)	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Ritup Seng
...	...	...	...	...	...	...	...	...	
15504	Zulm Ko Jala Doonga	(1988)	NaN	Action	4.6	11	Mahendra Shah	Naseeruddin Shah	Sumeet S
15505	Zulmi	(1999)	129 min	Action, Drama	4.5	655	Kuku Kohli	Akshay Kumar	Twinkle Kha
15506	Zulmi Raj	(2005)	NaN	Action	NaN	NaN	Kiran Thej	Sangeeta Tiwari	
15507	Zulmi Shikari	(1988)	NaN	Action	NaN	NaN	NaN	NaN	
15508	Zulm-O-Sitam	(1998)	130 min	Action, Drama	6.2	20	K.C. Bokadia	Dharmendra	Jaya P

15509 rows × 10 columns

```
df.describe()
```

	Rating
count	7919.000000
mean	5.841621
std	1.381777
min	1.100000
25%	4.900000
50%	6.000000
75%	6.800000
max	10.000000

```
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
```

```
df.dtypes
```

```
Name         object
Year         object
Duration     object
Genre        object
Rating      float64
Votes        object
Director     object
Actor 1      object
Actor 2      object
Actor 3      object
dtype: object
```

```
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
```

```
df.shape
```

```
(15509, 10)
```

```
df.dropna(inplace=True)
df.head(5)
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor
1	#Gadhvi (He thought he was Gandhi)	(2019)	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamanc
3	#Yaaram	(2019)	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita R
5	...Aur Pyaar Ho Gaya	(1997)	147 min	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachcha
6	...Yahaan	(2005)	142 min	Drama, Romance, War	7.4	1,086	Shoojit Sircar	Jimmy Sheirgill	Minissha Lamb
8	?: A Question Mark	(2012)	82 min	Horror, Mystery, Thriller	5.6	326	Allyson Patel	Yash Dave	Muntazir Ahma

```
df.isnull().sum()
```

```
Name        0
Year        0
Duration    0
Genre       0
Rating      0
Votes       0
Director    0
Actor 1     0
Actor 2     0
Actor 3     0
dtype: int64
```

```
df.shape
```

```
(5659, 10)
```

```
df['Year'] = df['Year'].str.extract('(\d+)')  # Extract numeric part of the string
df['Year'] = pd.to_numeric(df['Year'], errors='coerce')  # Convert to numeric
```

```
df['Duration'] = df['Duration'].str.extract('(\d+)')
df['Duration'] = pd.to_numeric(df['Duration'], errors='coerce')
```

```
df["Year"].head()
```

```
1    2019
3    2019
5    1997
6    2005
8    2012
Name: Year, dtype: int64
```

```
genre=df['Genre']
genre.head(5)
```

```
1                       Drama
3              Comedy, Romance
5       Comedy, Drama, Musical
6          Drama, Romance, War
8    Horror, Mystery, Thriller
Name: Genre, dtype: object
```

```
genres=df['Genre'].str.split(',',expand=True)
genres.head(5)
```

	0	1	2
1	Drama	None	None
3	Comedy	Romance	None
5	Comedy	Drama	Musical
6	Drama	Romance	War
8	Horror	Mystery	Thriller

```
genre_counts = {}
for genre in genres.values.flatten():
    if genre is not None:
        if genre in genre_counts:
            genre_counts[genre] += 1
        else:
            genre_counts[genre] = 1

genereCounts = {genre: count for genre, count in sorted(genre_counts.items())}
for genre, count in genereCounts.items():
    print(f"{genre}: {count}")
```

```
Action: 34
Adventure: 172
Biography: 31
Comedy: 355
Crime: 604
Drama: 1954
Family: 364
Fantasy: 115
History: 91
Horror: 74
Music: 50
Musical: 322
Mystery: 245
News: 1
Romance: 1221
Sci-Fi: 28
Sport: 38
Thriller: 590
War: 30
Western: 1
Action: 1652
Adventure: 105
Animation: 40
Biography: 84
Comedy: 989
Crime: 271
Documentary: 48
Drama: 1842
Family: 52
Fantasy: 31
History: 8
Horror: 128
Music: 3
Musical: 90
```

```
Mystery: 59
Romance: 159
Sci-Fi: 4
Sport: 2
Thriller: 89
War: 3
```

```
genresPie = df['Genre'].value_counts()
genresPie.head(5)
```

```
Drama                   844
Drama, Romance          332
Action, Crime, Drama    329
Action, Drama           206
Comedy, Drama           205
Name: Genre, dtype: int64
```

```
genrePie = pd.DataFrame(list(genresPie.items()))
genrePie = genrePie.rename(columns={0: 'Genre', 1: 'Count'})
genrePie.head(5)
```

	Genre	Count
0	Drama	844
1	Drama, Romance	332
2	Action, Crime, Drama	329
3	Action, Drama	206
4	Comedy, Drama	205

```
df['Votes'] = df['Votes'].str.replace(',', '').astype(int)
df["Votes"].head(5)
```

```
1       8
3      35
5     827
6    1086
8     326
Name: Votes, dtype: int64
```

```
df["Director"].nunique()
```

```
2431
```

```
directors = df["Director"].value_counts()
directors.head(5)
```

```
David Dhawan            41
Mahesh Bhatt            39
Ram Gopal Varma         33
Hrishikesh Mukherjee    33
Shakti Samanta          33
Name: Director, dtype: int64
```

```
actors = pd.concat([df['Actor 1'], df['Actor 2'], df['Actor 3']]).dropna().value_counts()
actors.head(5)
```

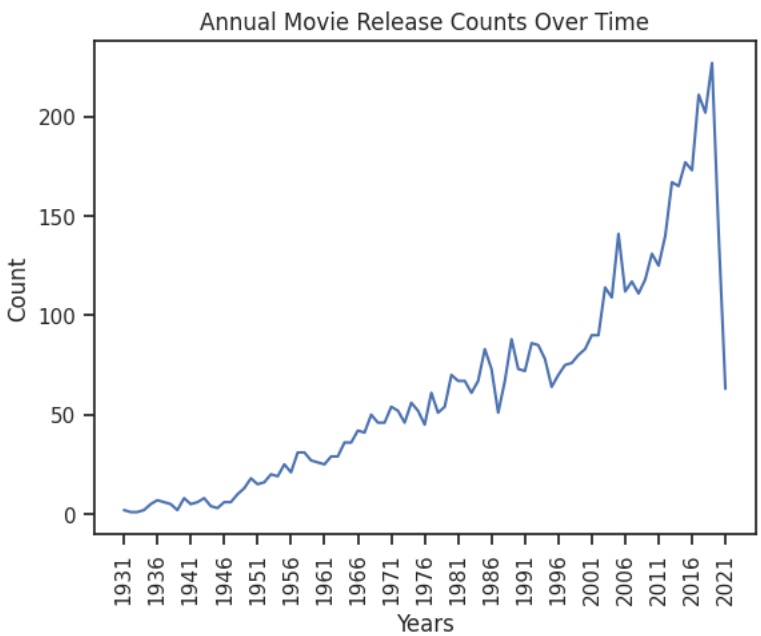
```
Mithun Chakraborty    160
Amitabh Bachchan      148
Dharmendra            146
Ashok Kumar           124
Akshay Kumar          120
dtype: int64
```

THE DATA VISUALIZATION

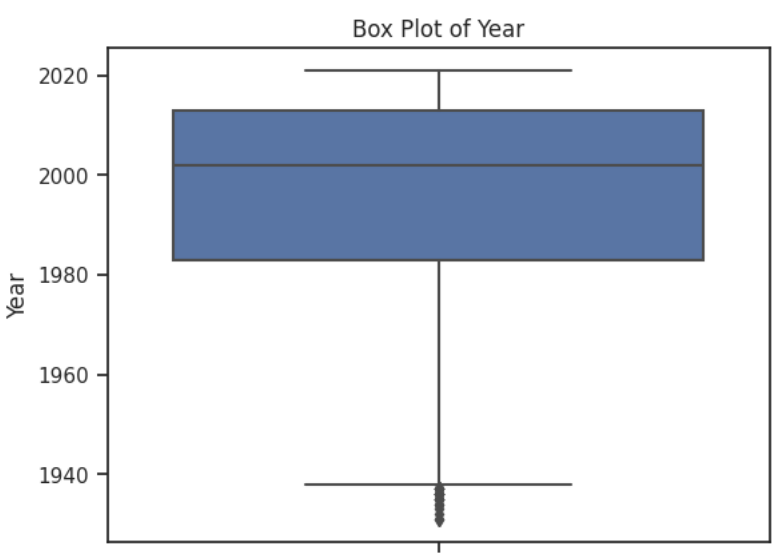
```
import seaborn as sb
import plotly.express as px
import matplotlib.pyplot as mpl
from wordcloud import WordCloud
```

```
sb.set(style = "ticks")
```

```
ax = sb.lineplot(data=df['Year'].value_counts().sort_index())
tick_positions = range(min(df['Year']), max(df['Year']) + 1, 5)
ax.set_title("Annual Movie Release Counts Over Time")
ax.set_xticks(tick_positions)
ax.set_xticklabels(tick_positions, rotation = 90)
ax.set_xlabel("Years")
ax.set_ylabel("Count")
mpl.show()
```

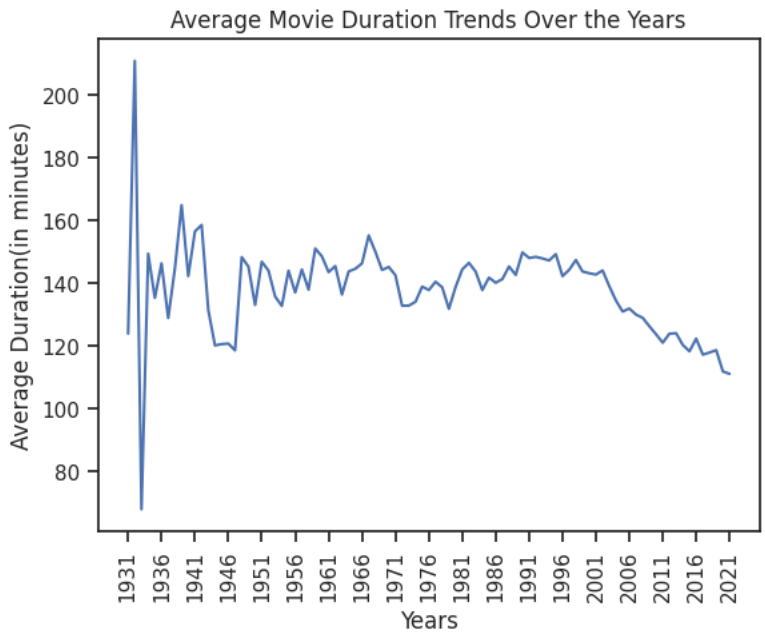



```
ax = sb.boxplot(data=df, y='Year')
ax.set_ylabel('Year')
ax.set_title('Box Plot of Year')
mpl.show()
```

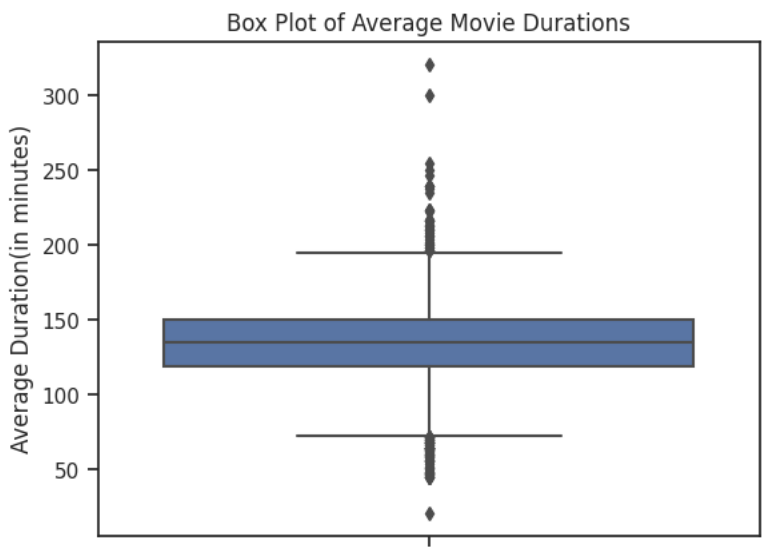



```
ax = sb.lineplot(data=df.groupby('Year')['Duration'].mean().reset_index(), x='Year', y='Duration')
tick_positions = range(min(df['Year']), max(df['Year']) + 1, 5)
ax.set_title("Average Movie Duration Trends Over the Years")
ax.set_xticks(tick_positions)
```

```
ax.set_xticklabels(tick_positions, rotation = 90)
ax.set_xlabel("Years")
ax.set_ylabel('Average Duration(in minutes)')
mpl.show()
```




```
ax = sb.boxplot(data=df, y='Duration')
ax.set_title("Box Plot of Average Movie Durations")
ax.set_ylabel('Average Duration(in minutes)')
mpl.show()
```



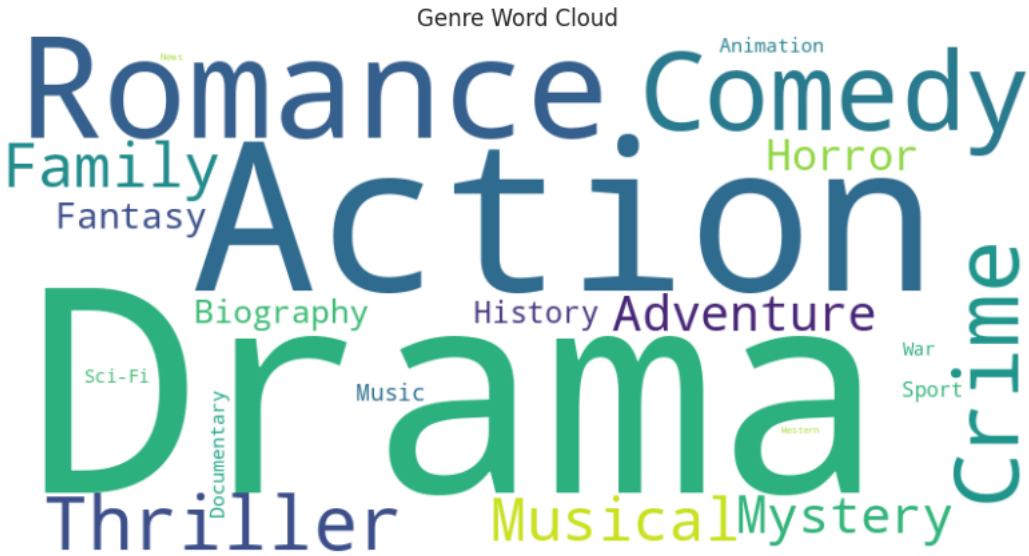

```
Q1 = df['Duration'].quantile(0.25)
Q3 = df['Duration'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df['Duration'] >= lower_bound) & (df['Duration'] <= upper_bound)]
df.head(5)
```

Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2

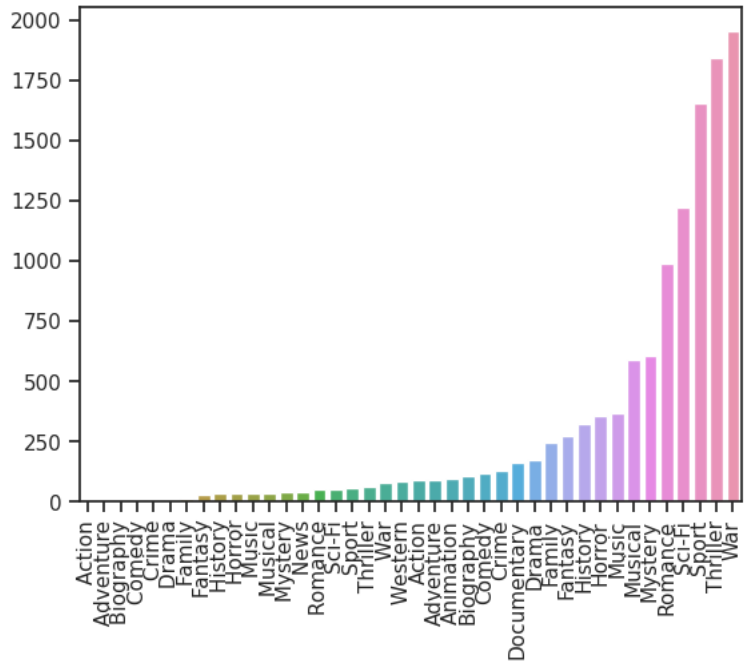
```
genre_counts = df['Genre'].str.split(', ', expand=True).stack().value_counts()

wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(genre_counts)

mpl.figure(figsize=(10, 6))
mpl.imshow(wordcloud, interpolation='bilinear')
mpl.axis('off')
mpl.title('Genre Word Cloud')
mpl.show()
```

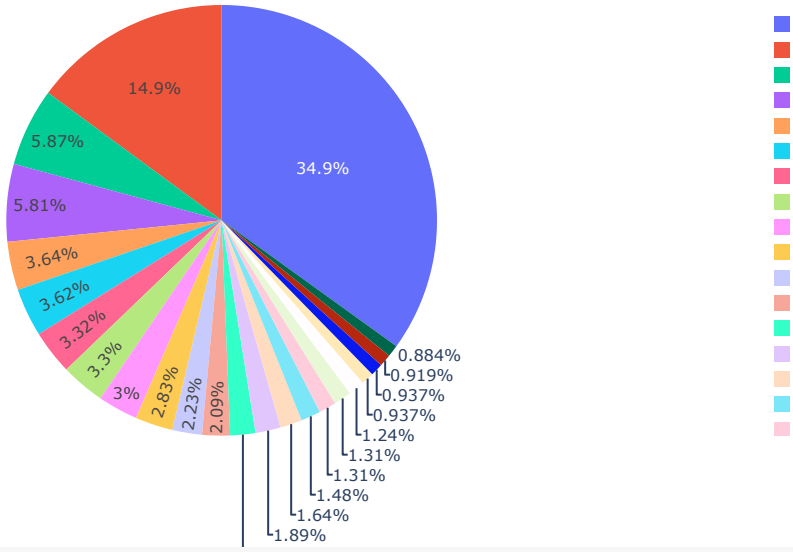



```
genreLabels = sorted(genereCounts.keys())
genreCounts = sorted(genereCounts.values())
ax = sb.barplot(x = genreLabels, y = genreCounts)
ax.set_xticklabels(labels=genreLabels, rotation = 90)
mpl.show()
```

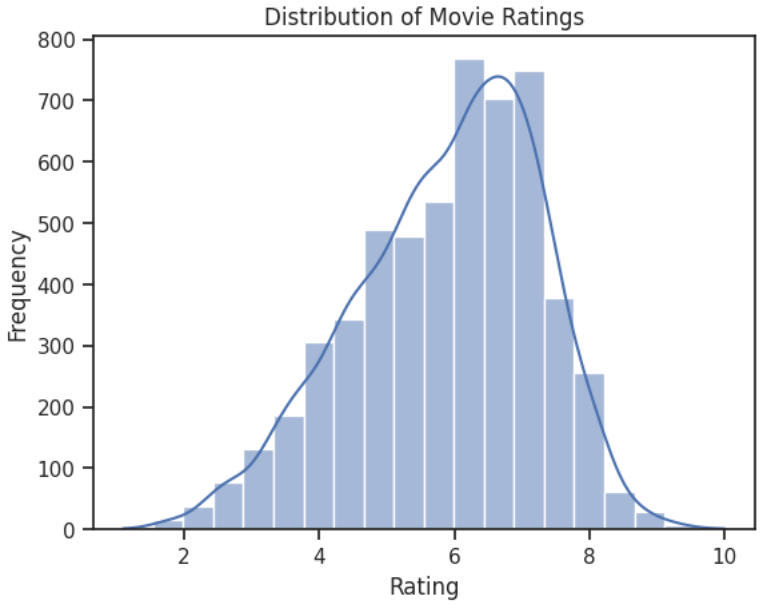



```
genrePie.loc[genrePie['Count'] < 50, 'Genre'] = 'Other'
ax = px.pie(genrePie, values='Count', names='Genre', title='More than one Genre of movies in Indian Cinema')
ax.show()
```

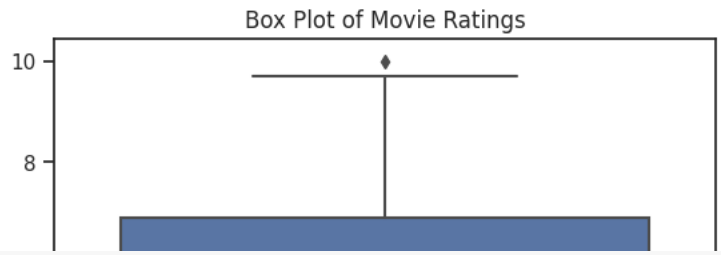
More than one Genre of movies in Indian Cinema




```
ax = sb.histplot(data = df, x = "Rating", bins = 20, kde = True)
ax.set_xlabel('Rating')
ax.set_ylabel('Frequency')
ax.set_title('Distribution of Movie Ratings')
mpl.show()
```



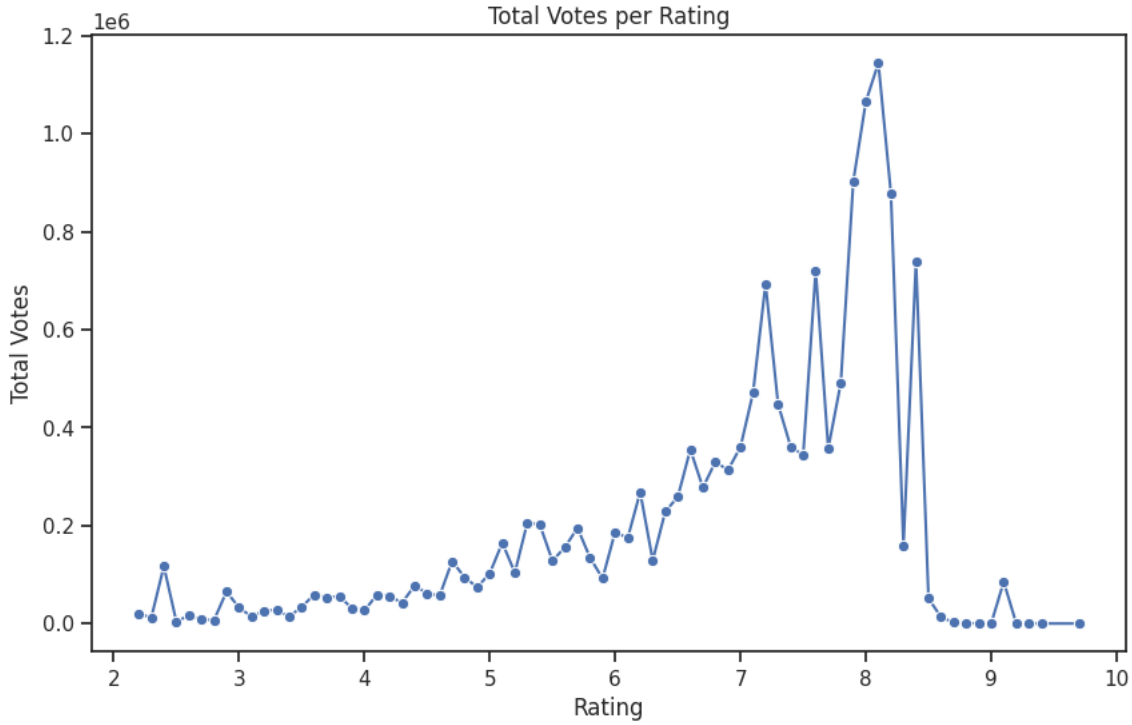

```
ax = sb.boxplot(data=df, y='Rating')
ax.set_ylabel('Rating')
ax.set_title('Box Plot of Movie Ratings')
mpl.show()
```



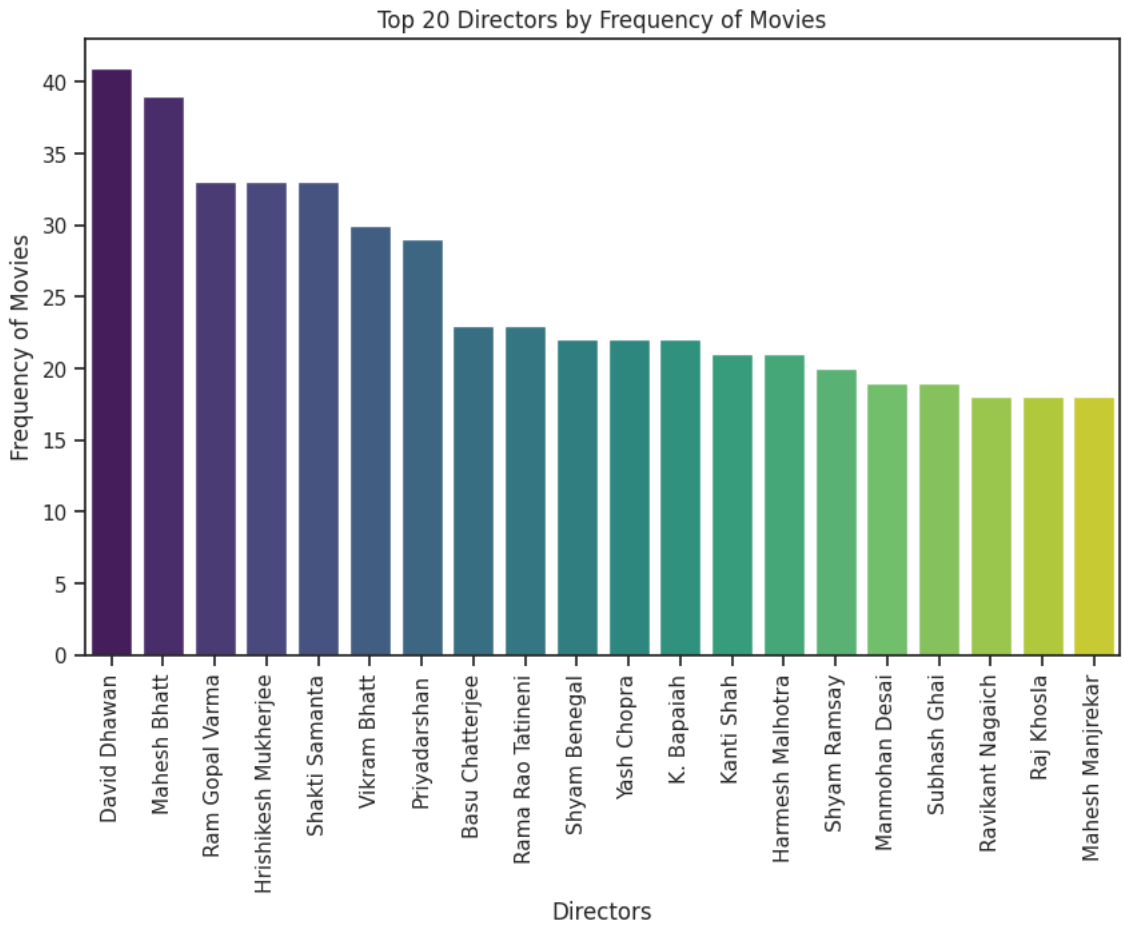

```
Q1 = df['Rating'].quantile(0.25)
Q3 = df['Rating'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df['Rating'] >= lower_bound) & (df['Rating'] <= upper_bound)]
df.head(5)
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2
1	#Gadhvi (He thought he was Gandhi)	2019	109	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande
3	#Yaaram	2019	110	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj
5	...Aur Pyaar Ho Gaya	1997	147	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan
6	...Yahaan	2005	142	Drama, Romance, War	7.4	1086	Shoojit Sircar	Jimmy Sheirgill	Minissha Lamba
8	?: A Question Mark	2012	82	Horror, Mystery, Thriller	5.6	326	Allyson Patel	Yash Dave	Muntazir Ahmad

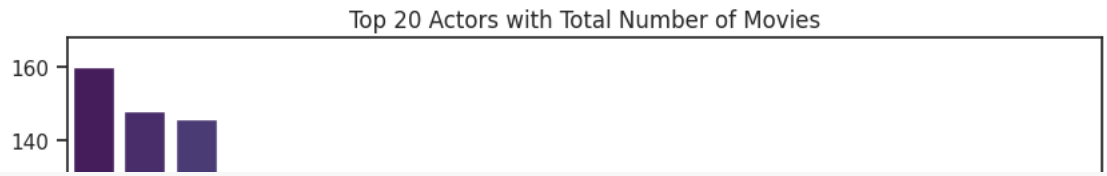
```
rating_votes = df.groupby('Rating')['Votes'].sum().reset_index()
mpl.figure(figsize=(10, 6))
ax_line_seaborn = sb.lineplot(data=rating_votes, x='Rating', y='Votes', marker='o')
ax_line_seaborn.set_xlabel('Rating')
ax_line_seaborn.set_ylabel('Total Votes')
ax_line_seaborn.set_title('Total Votes per Rating')
mpl.show()
```




```
mpl.figure(figsize=(10, 6))
ax = sb.barplot(x=directors.head(20).index, y=directors.head(20).values, palette='viridis')
ax.set_xlabel('Directors')
ax.set_ylabel('Frequency of Movies')
ax.set_title('Top 20 Directors by Frequency of Movies')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
mpl.show()
```



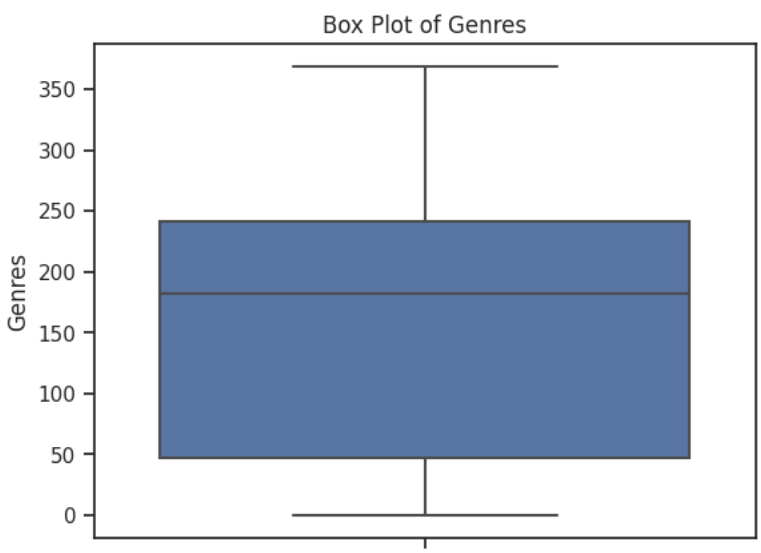

```
mpl.figure(figsize=(10, 6))
ax = sb.barplot(x=actors.head(20).index, y=actors.head(20).values, palette='viridis')
ax.set_xlabel('Actors')
ax.set_ylabel('Total Number of Movies')
ax.set_title('Top 20 Actors with Total Number of Movies')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
mpl.show()
```




```
df["Actor"] = df['Actor 1'] + ', ' + df['Actor 2'] + ', ' + df['Actor 3']
df["Directors"] = df['Director'].astype('category').cat.codes
df["Genres"] = df['Genre'].astype('category').cat.codes
df["Actors"] = df['Actor'].astype('category').cat.codes
df.head(5)
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3	Actor	Direct
1	#Gadhvi (He thought he was Gandhi)	2019	109	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid	Rasika Dugal, Vivek Ghamande, Arvind Jangid	
3	#Yaaram	2019	110	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor	Prateik, Ishita Raj, Siddhant Kapoor	1
5	...Aur Pyaar Ho Gaya	1997	147	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor	Bobby Deol, Aishwarya Rai Bachchan, Shammi Kapoor	1
6	...Yahaan	2005	142	Drama, Romance, War	7.4	1086	Shoojit Sircar	Jimmy Sheirgill	Minissha Lamba	Yashpal Sharma	Jimmy Sheirgill, Minissha Lamba, Yashpal Sharma	1
8	?: A Question Mark	2012	82	Horror, Mystery, Thriller	5.6	326	Allyson Patel	Yash Dave	Muntazir Ahmad	Kiran Bhatia	Yash Dave, Muntazir Ahmad, Kiran Bhatia	

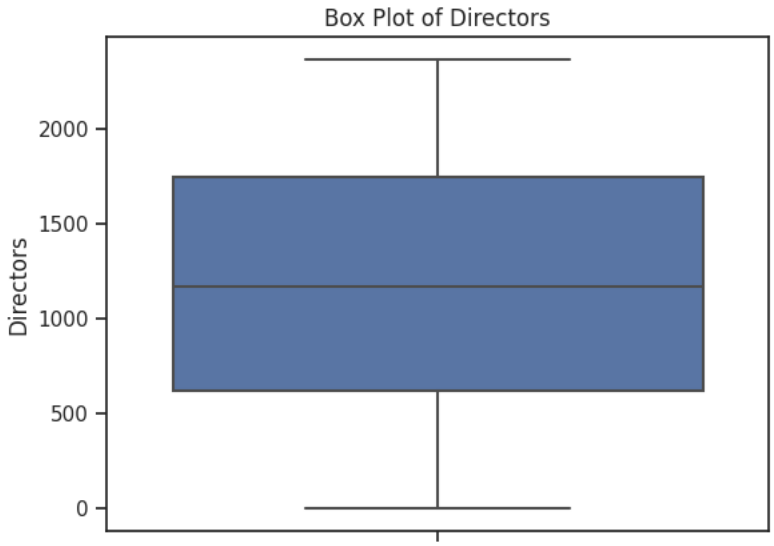
```
ax = sb.boxplot(data=df, y='Genres')
ax.set_ylabel('Genres')
ax.set_title('Box Plot of Genres')
mpl.show()
```




```
Q1 = df['Genres'].quantile(0.25)
Q3 = df['Genres'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df['Genres'] >= lower_bound) & (df['Genres'] <= upper_bound)]
df.head(5)
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3	Actor	Direct
1	#Gadhvi (He thought he was Gandhi)	2019	109	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid	Rasika Dugal, Vivek Ghamande, Arvind Jangid	
3	#Yaaram	2019	110	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor	Prateik, Ishita Raj, Siddhant Kapoor	1
5	...Aur Pyaar Ho Gaya	1997	147	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor	Bobby Deol, Aishwarya Rai Bachchan, Shammi Kapoor	1

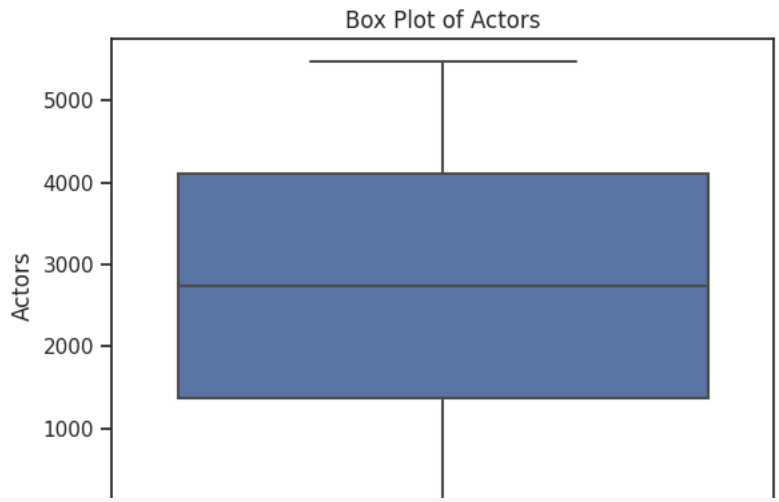
```
ax = sb.boxplot(data=df, y='Directors')
ax.set_ylabel('Directors')
ax.set_title('Box Plot of Directors')
mpl.show()
```




```
Q1 = df['Directors'].quantile(0.25)
Q3 = df['Directors'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df['Directors'] >= lower_bound) & (df['Directors'] <= upper_bound)]
df.head(5)
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3	Actor	Direct
1	#Gadhvi (He thought he was Gandhi)	2019	109	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid	Rasika Dugal, Vivek Ghamande, Arvind Jangid	
3	#Yaaram	2019	110	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor	Prateik, Ishita Raj, Siddhant Kapoor	1
5	...Aur Pyaar Ho Gaya	1997	147	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor	Bobby Deol, Aishwarya Rai Bachchan, Shammi Kapoor	1
6	...Yahaan	2005	142	Drama, Romance, War	7.4	1086	Shoojit Sircar	Jimmy Sheirgill	Minissha Lamba	Yashpal Sharma	Jimmy Sheirgill, Minissha Lamba, Yashpal Sharma	1
8	?: A Question Mark	2012	82	Horror, Mystery, Thriller	5.6	326	Allyson Patel	Yash Dave	Muntazir Ahmad	Kiran Bhatia	Yash Dave, Muntazir Ahmad, Kiran Bhatia	

```
ax = sb.boxplot(data=df, y='Actors')
ax.set_ylabel('Actors')
ax.set_title('Box Plot of Actors')
mpl.show()
```




```
Q1 = df['Actors'].quantile(0.25)
Q3 = df['Actors'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df['Actors'] >= lower_bound) & (df['Actors'] <= upper_bound)]
df.head(5)
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3	Actor	Direct
1	#Gadhvi (He thought he was Gandhi)	2019	109	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid	Rasika Dugal, Vivek Ghamande, Arvind Jangid	
3	#Yaaram	2019	110	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor	Prateik, Ishita Raj, Siddhant Kapoor	1
5	...Aur Pyaar Ho Gaya	1997	147	Comedy, Drama, Musical	4.7	827	Rahul Rawail	Bobby Deol	Aishwarya Rai Bachchan	Shammi Kapoor	Bobby Deol, Aishwarya Rai Bachchan, Shammi Kapoor	1
6	...Yahaan	2005	142	Drama, Romance, War	7.4	1086	Shoojit Sircar	Jimmy Sheirgill	Minissha Lamba	Yashpal Sharma	Jimmy Sheirgill, Minissha Lamba, Yashpal Sharma	1
8	?: A Question Mark	2012	82	Horror, Mystery, Thriller	5.6	326	Allyson Patel	Yash Dave	Muntazir Ahmad	Kiran Bhatia	Yash Dave, Muntazir Ahmad, Kiran Bhatia	

SPLITTING THE DATA

```
from sklearn.model_selection import train_test_split
```

```
Input = df.drop(['Name', 'Genre', 'Rating', 'Director', 'Actor 1', 'Actor 2', 'Actor 3', 'Actor'], axis=1)
Output = df['Rating']
```

```
Input.head(5)
```

	Year	Duration	Votes	Directors	Genres	Actors
1	2019	109	8	610	224	3788
3	2019	110	35	1305	182	3263
5	1997	147	827	1493	155	1091
6	2005	142	1086	1994	283	2036
8	2012	82	326	133	314	5437

```
Output.head(5)
```

```
1    7.0
3    4.4
```

```
5    4.7
6    7.4
8    5.6
Name: Rating, dtype: float64
```

```
x_train, x_test, y_train, y_test = train_test_split(Input, Output, test_size = 0.2, random_state = 1)
```

THE MODEL

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score as score
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
!pip install catboost
from catboost import CatBoostRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.23.5)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.3)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2023.3.post1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.1.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.12.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.43.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)
```

```
def evaluate_model(y_true, y_pred, model_name):
    print("Model: ", model_name)
    print("Accuracy = {:0.2f}%".format(score(y_true, y_pred)*100))
    print("Mean Squared Error = {:0.2f}\n".format(mean_squared_error(y_true, y_pred, squared=False)))
    return round(score(y_true, y_pred)*100, 2)
```

```
LR = LinearRegression()
LR.fit(x_train, y_train)
lr_preds = LR.predict(x_test)

RFR = RandomForestRegressor(n_estimators=100, random_state=1)
RFR.fit(x_train, y_train)
rf_preds = RFR.predict(x_test)

DTR = DecisionTreeRegressor(random_state=1)
DTR.fit(x_train, y_train)
dt_preds = DTR.predict(x_test)

XGBR = XGBRegressor(n_estimators=100, random_state=1)
XGBR.fit(x_train, y_train)
xgb_preds = XGBR.predict(x_test)

GBR = GradientBoostingRegressor(n_estimators=100, random_state=60)
GBR.fit(x_train, y_train)
gb_preds = GBR.predict(x_test)

LGBMR = LGBMRegressor(n_estimators=100, random_state=60)
LGBMR.fit(x_train, y_train)
lgbm_preds = LGBMR.predict(x_test)

CBR = CatBoostRegressor(n_estimators=100, random_state=1, verbose=False)
CBR.fit(x_train, y_train)
catboost_preds = CBR.predict(x_test)

KNR = KNeighborsRegressor(n_neighbors=5)
```

```
KNR.fit(x_train, y_train)
knn_preds = KNR.predict(x_test)
```

```
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000502 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1182
[LightGBM] [Info] Number of data points in the train set: 4425, number of used features: 6
[LightGBM] [Info] Start training from score 5.904429
```

```
LRScore = evaluate_model(y_test, lr_preds, "LINEAR REGRESSION")
RFScore = evaluate_model(y_test, rf_preds, "RANDOM FOREST")
DTScore = evaluate_model(y_test, dt_preds, "DECEISION TREE")
XGBScore = evaluate_model(y_test, xgb_preds, "EXTENDED GRADIENT BOOSTING")
GBScore = evaluate_model(y_test, gb_preds, "GRADIENT BOOSTING")
LGBScore = evaluate_model(y_test, lgbm_preds, "LIGHT GRADIENT BOOSTING")
CBRScore = evaluate_model(y_test, catboost_preds, "CAT BOOST")
KNNScore = evaluate_model(y_test, knn_preds, "K NEAREST NEIGHBORS")
```

```
Model:  LINEAR REGRESSION
Accuracy = 10.56%
Mean Squared Error = 1.26

Model:  RANDOM FOREST
Accuracy = 38.91%
Mean Squared Error = 1.04

Model:  DECEISION TREE
Accuracy = -20.64%
Mean Squared Error = 1.47

Model:  EXTENDED GRADIENT BOOSTING
Accuracy = 36.00%
Mean Squared Error = 1.07

Model:  GRADIENT BOOSTING
Accuracy = 40.21%
Mean Squared Error = 1.03

Model:  LIGHT GRADIENT BOOSTING
Accuracy = 42.65%
Mean Squared Error = 1.01

Model:  CAT BOOST
Accuracy = 40.97%
Mean Squared Error = 1.03

Model:  K NEAREST NEIGHBORS
Accuracy = 1.97%
Mean Squared Error = 1.32
```

```
models = pd.DataFrame(
    {
        "MODELS": ["Linear Regression", "Random Forest", "Decision Tree", "Gradient Boosting", "Extended Gradient Boosting", "Light Gradient
        "SCORES": [LRScore, RFScore, DTScore, GBScore, XGBScore, LGBScore, CBRScore, KNNScore]
    }
)
models.sort_values(by='SCORES', ascending=False)
```

	MODELS	SCORES
5	Light Gradient Boosting	42.65
6	Cat Boosting	40.97
3	Gradient Boosting	40.21
1	Random Forest	38.91
4	Extended Gradient Boosting	36.00
0	Linear Regression	10.56
7	K Nearest Neighbors	1.97
2	Decision Tree	-20.64