Name: Aditya Mishra Roll No:34
Class:T.E.I.T Batch: A

In [ ]:
Experiment NO:10

In [ ]:
Aim:Business intelligencc mini project(Movie Rating Prediction)
LO:LO1-LO6

Theory:
Movie rating prediction is the task of predicting how a user will rate a movie based on their historical ratings and the characteristics of the movie. It is a classic example of a recommendation system, which aims to suggest relevant items to users based on their preferences and behavior.

One common approach to movie rating prediction is collaborative filtering, which involves using the ratings of other users who have similar preferences to predict the rating of a target user for a particular movie. Collaborative filtering can be done using either user-based or item-based approaches.

In the user-based approach, the similarity between users is computed based on their rating patterns, and the ratings of similar users are used to predict the rating of the target user for a particular movie. In the item-based approach, the similarity between movies is computed based on the rating patterns of users who have rated both movies, and the ratings of similar movies are used to predict the rating of the target user for a particular movie.

Another approach to movie rating prediction is content-based filtering, which involves using the characteristics of the movie to predict the rating of a target user. In this approach, the movie is described by a set of features, such as the genre, actors, director, and plot, and the rating of the target user is predicted based on the similarity between the features of the movie and the historical preferences of the user.

Machine learning techniques such as regression, decision trees, and neural networks can also be used for movie rating prediction. These techniques involve training a model on historical ratings and movie characteristics, and using the model to predict the rating of a target user for a particular movie.

Movie rating prediction is a challenging task due to the complexity of human preferences and the diversity of movies. However, it has important practical applications in movie recommendation systems and personalized advertising.

In [ ]:
```
# Importing all the required libraries
```

In [1]:
```
#Import the Libraries
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt


import seaborn as sns
plt.style.use('seaborn-whitegrid')

from sklearn.impute import SimpleImputer

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import LabelEncoder
```

Loading the dataset

In [2]:
```
#Load the Dataset
df_movies = pd.read_csv("kaggle_movie_dataset.csv", index_col="ID")
df_movies.head(10)
```

Out[2]:

ID	Unnamed: 0	Title	Year	IMDb	Netflix	Hulu	Prime Video	Disney+	Type	Directors	
1	0	Inception	2010	8.8	1	0	0	0	0	Christopher Nolan	
2	1	The Matrix	1999	8.7	1	0	0	0	0	Lana Wachowski,Lilly Wachowski	
3	2	Avengers: Infinity War	2018	8.5	1	0	0	0	0	Anthony Russo,Joe Russo	
4	3	Back to the Future	1985	8.5	1	0	0	0	0	Robert Zemeckis	
5	4	The Good, the Bad and the Ugly	1966	8.8	1	0	1	0	0	Sergio Leone	
6	5	Spider-Man: Into the Spider-Verse	2018	8.4	1	0	0	0	0	Bob Persichetti,Peter Ramsey,Rodney Rothman	An
7	6	The Pianist	2002	8.5	1	0	1	0	0	Roman Polanski	
8	7	Django Unchained	2012	8.4	1	0	0	0	0	Quentin Tarantino	
9	8	Raiders of the Lost Ark	1981	8.4	1	0	0	0	0	Steven Spielberg	
10	9	Inglourious Basterds	2009	8.3	1	0	0	0	0	Quentin Tarantino	

In [3]:
```
df_movies.shape
```

Out[3]:
```
(16744, 14)
```

Cleaning the dataset

In [4]:
```
#Remove "Unnamed"
df_movies=df_movies.drop("Unnamed: 0",axis=1)
df_movies.head()
```

Out[4]:

ID	Title	Year	IMDb	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Ge
1	Inception	2010	8.8	1	0	0	0	0	Christopher Nolan	Action,Adventur Fi,T
2	The Matrix	1999	8.7	1	0	0	0	0	Lana Wachowski,Lilly Wachowski	Action,
3	Avengers: Infinity War	2018	8.5	1	0	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,
4	Back to the Future	1985	8.5	1	0	0	0	0	Robert Zemeckis	Adventure,Comed
5	The Good, the Bad and the Ugly	1966	8.8	1	0	1	0	0	Sergio Leone	We

In [5]:
```
#Description overview of the data
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16744 entries, 1 to 16744
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Title        16744 non-null  object 
 1   Year         16744 non-null  int64  
 2   IMDb         16173 non-null  float64
 3   Netflix      16744 non-null  int64  
 4   Hulu         16744 non-null  int64  
 5   Prime Video  16744 non-null  int64  
 6   Disney+      16744 non-null  int64  
 7   Type         16744 non-null  int64  
 8   Directors    16018 non-null  object 
 9   Genres       16469 non-null  object 
 10  Country      16309 non-null  object 
 11  Language     16145 non-null  object 
 12  Runtime      16152 non-null  float64
dtypes: float64(2), int64(6), object(5)
memory usage: 1.8+ MB
```

finding total number of null values

In [6]:
```
#Check for total no.of null values in each column
df_movies.isna().sum()
```

Out[6]:
```
Title            0
Year             0
IMDb           571
Netflix          0
Hulu             0
Prime Video      0
Disney+          0
Type             0
Directors      726
Genres         275
Country        435
Language       599
Runtime        592
dtype: int64
```

In [7]:
```
#Removing the Target value
df_movies = df_movies[df_movies['IMDb'].notna()]
```

In [8]:
```
pip install missingno
```

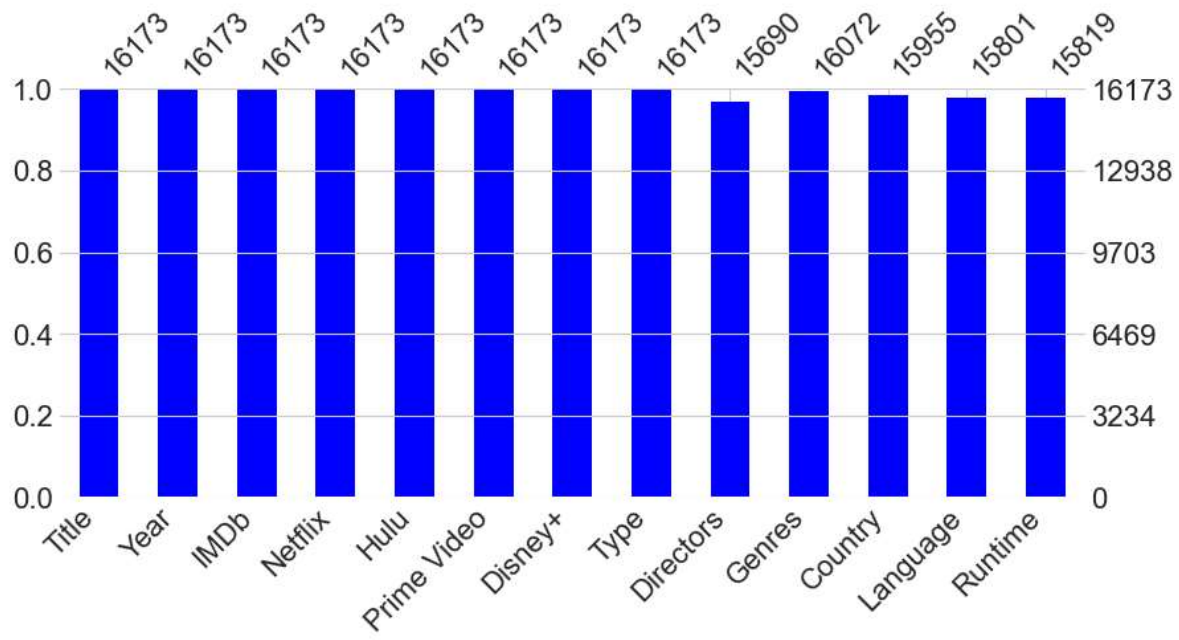
```
Requirement already satisfied: missingno in c:\users\lenovo\anaconda3\lib\sit
e-packages (0.5.2)Note: you may need to restart the kernel to use updated pac
kages.

Requirement already satisfied: matplotlib in c:\users\lenovo\anaconda3\lib\si
te-packages (from missingno) (3.5.2)
Requirement already satisfied: seaborn in c:\users\lenovo\anaconda3\lib\site-
packages (from missingno) (0.11.2)
Requirement already satisfied: numpy in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from missingno) (1.21.5)
Requirement already satisfied: scipy in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from missingno) (1.9.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lenovo\anacon
da3\lib\site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\lenovo\anaconda3\lib
\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\lenovo\anaconda3
\lib\site-packages (from matplotlib->missingno) (1.4.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\lenovo\anaconda3\lib
\site-packages (from matplotlib->missingno) (9.2.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\lenovo\anaconda3
\lib\site-packages (from matplotlib->missingno) (3.0.9)
Requirement already satisfied: packaging>=20.0 in c:\users\lenovo\anaconda3\l
ib\site-packages (from matplotlib->missingno) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\lenovo\anaconda3
\lib\site-packages (from matplotlib->missingno) (4.25.0)
Requirement already satisfied: pandas>=0.23 in c:\users\lenovo\anaconda3\lib
\site-packages (from seaborn->missingno) (1.4.4)
Requirement already satisfied: pytz>=2020.1 in c:\users\lenovo\anaconda3\lib
\site-packages (from pandas>=0.23->seaborn->missingno) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\anaconda3\lib\site
-packages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
```

In [9]:
```
import missingno as msno
```

Representation of missing data

In [10]:
```
#Visualizing the amount of missing data
msno.bar(df_movies ,color='blue', figsize=(10, 4))
```

Out[10]: <AxesSubplot:>



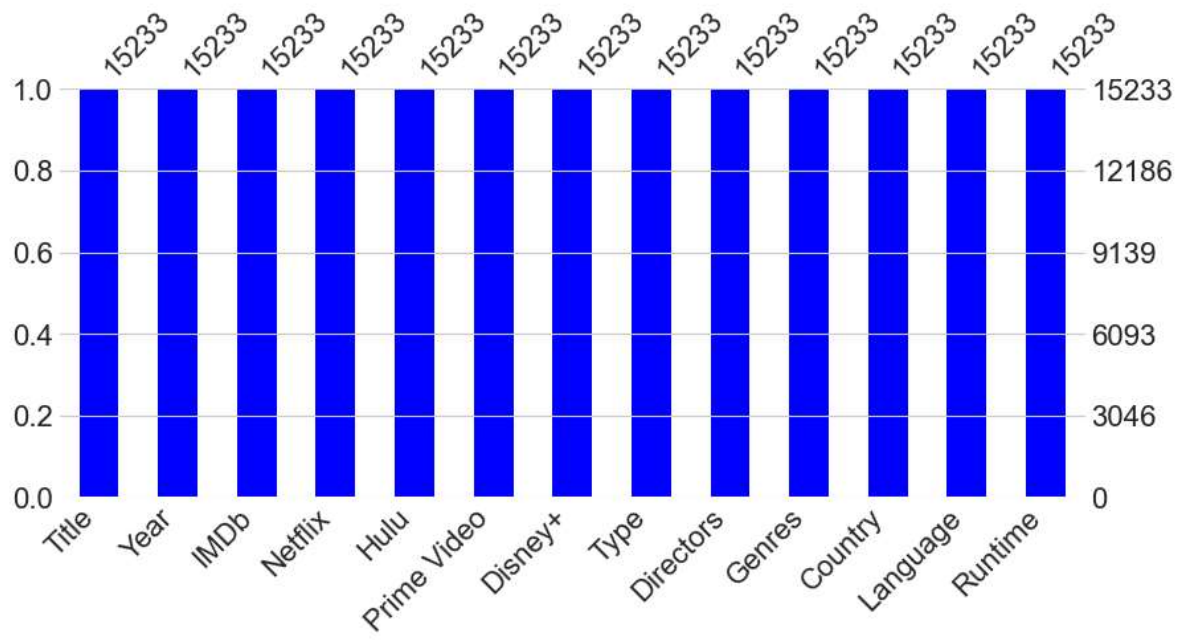
Removal of null values

In [11]:
```
#Dropping all the rows(entries) where there are celss with no data
df_movies.dropna(axis=0, how='any',inplace=True)

df_movies.isna().sum()
```

Out[11]:
```
Title          0
Year           0
IMDb           0
Netflix        0
Hulu           0
Prime Video    0
Disney+        0
Type           0
Directors      0
Genres         0
Country        0
Language       0
Runtime        0
dtype: int64
```

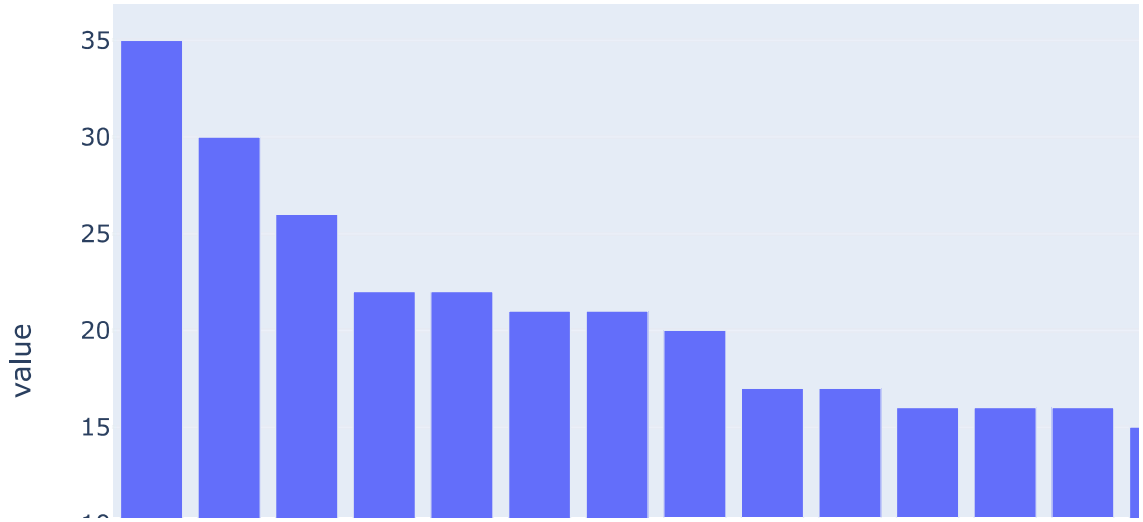
In [12]:
```
#Visualizing if there is anymore missing data or not
msno.bar(df_movies ,color='blue', figsize=(10, 4))
```

Out[12]: <AxesSubplot:>



Bar Graph Representation of Director versus number of movies made by director

In [13]:
```
import plotly.express as px
director = df_movies.dropna(axis=0, subset=['Directors'])
dfg = director['Directors'].value_counts().head(20)
fig = px.bar(dfg)
fig.show()
```



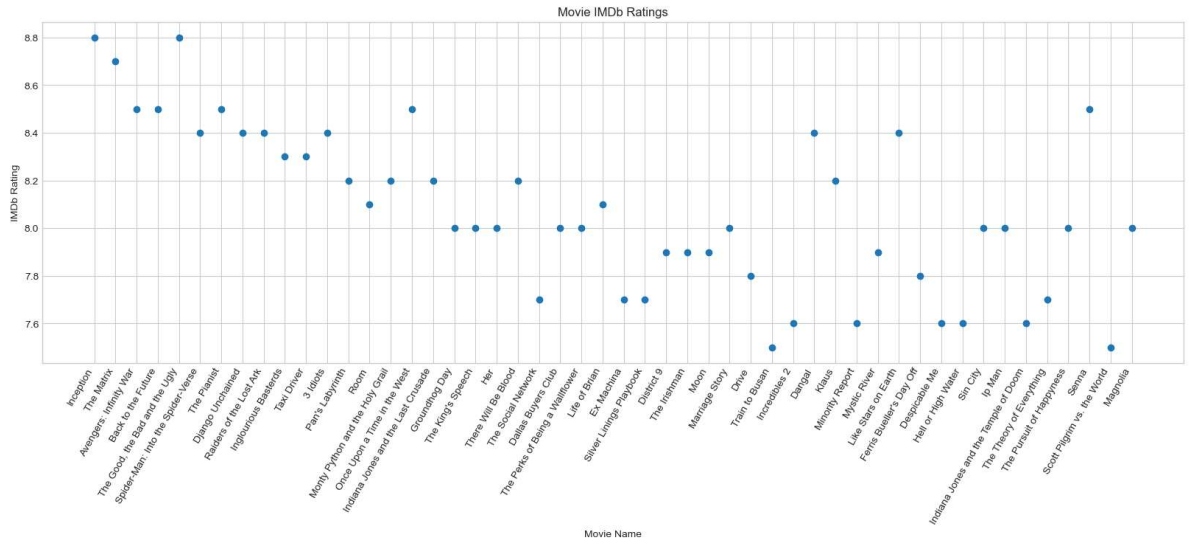
Representation of top movie of Each year

In [14]:
```
import plotly.figure_factory as ff
import plotly.offline as py
rating = df_movies.dropna(axis=0, subset=['IMDb'])  #dropping rows where Year
rating = rating.sort_values('IMDb', ascending=False).drop_duplicates(['Year'])
data = ff.create_table(rating)
py.iplot(data)
```

Title	Year	IMDb	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Genres
Down, Burnout	2015	9.3	0	0	1	0	0	Miguel Gaudencio	Documen
Steven Banks	1989	9.3	Unnamed:	0	1	0	0	Tom McLoughlin	Comedy
Square One	2019	9.3	0	0	1	0	0	Danny Wu	Documen
Bounty	2011	9.3	0	0	1	0	0	Roger Donaldson	Adventur
Natsamrat	2016	9.1	1	0	0	0	0	Mahesh Manjrekar	Drama,Fa
Finding Family	2013	9.1	0	0	1	0	0	Chris Leslie-Cogan	Documen
Where's Daddy?	2017	9.1	0	0	1	0	0	Rel Dowdell	Documen
The Dark Knight	2008	9.0	0	1	0	0	0	Christopher Nolan	Action,Cr
8 Wheels 2 Glory	2013	9.0	Some Soul	Brotha'	Music	0	1	0	0	Tyrone D. Dixon	Documen
The Creation	2012	8.9	0	0	1	0	0	Laura Gamse Jaeger	Documen

Scatter representation of IMDb rating and movie

In [15]:
```
import matplotlib.pyplot as plt
import mplcursors

# Get the movie names and IMDb ratings for the first 50 movies
movie_names = df_movies['Title'][:50]
imdb_ratings = df_movies['IMDb'][:50]

# Set the figure size and create a scatter plot with the movie names and IMDb
fig, ax = plt.subplots(figsize=(20, 6))
scatter = ax.scatter(movie_names, imdb_ratings)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=60, ha='right')

# Add axis labels and a title
ax.set_xlabel('Movie Name')
ax.set_ylabel('IMDb Rating')
ax.set_title('Movie IMDb Ratings')

# Add a scroll bar to the x-axis
mplcursors.cursor(scatter, hover=True)

# Show the plot
plt.show()
```



In [16]:
```
gen = df_movies['Genres'].value_counts()
gen
```

Out[16]:
```
Drama                                                                 1225
Documentary                                                           1034
Comedy                                                                 905
Comedy,Drama                                                           428
Horror                                                                 420
                                                                      ...
Animation,Adventure,Drama,Family,Fantasy,Musical,Mystery,Romance          1
Adventure,Biography,Drama                                                 1
Documentary,Biography,Family,History,Sport                                1
Musical,Comedy,Romance                                                    1
Comedy,Family,Adventure,Fantasy,Sci-Fi                                    1
Name: Genres, Length: 1824, dtype: int64
```

In [17]:
```
df_movies1 = pd.DataFrame(list(gen.items()))
df_movies1 = df_movies1.rename(columns={0: 'Genres', 1: 'Count'})
df_movies1
```

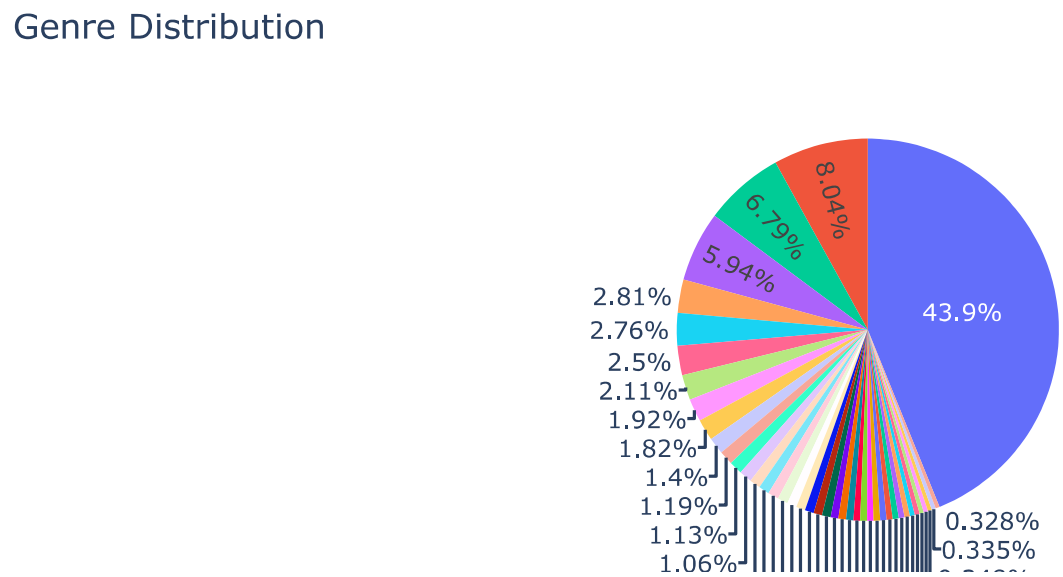
Out[17]:

	Genres	Count
0	Drama	1225
1	Documentary	1034
2	Comedy	905
3	Comedy,Drama	428
4	Horror	420
...	...	...
1819	Animation,Adventure,Drama,Family,Fantasy,Music...	1
1820	Adventure,Biography,Drama	1
1821	Documentary,Biography,Family,History,Sport	1
1822	Musical,Comedy,Romance	1
1823	Comedy,Family,Adventure,Fantasy,Sci-Fi	1

1824 rows × 2 columns

In [18]:
```
#Pie chart representation of Genres
```

In [19]:
```
import plotly.express as px
df_movies1.loc[df_movies1['Count'] < 50, 'Genres'] = 'Other' # Represent only
fig = px.pie(df_movies1, values='Count', names='Genres', title='Genre Distribu
fig.show()
```



In [20]:
```
# Assuming your data is stored in a pandas DataFrame called "df"

# Create a new column to store the director's average rating
df_movies['director_avg_rating'] = 0.0

# Loop through each director in the data
for director in df_movies['Directors'].unique():
    # Calculate the average rating of the director's movies
    avg_rating = df_movies.loc[df_movies['Directors'] == director, 'IMDb'].mea
    # Update the director_avg_rating column with the average rating
    df_movies.loc[df_movies['Directors'] == director, 'director_avg_rating'] =
```

In [21]:
```
df_movies.head()
```

Out[21]:

ID	Title	Year	IMDb	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Ge
1	Inception	2010	8.8	1	0	0	0	0	Christopher Nolan	Action,Adventur Fi,T
2	The Matrix	1999	8.7	1	0	0	0	0	Lana Wachowski,Lilly Wachowski	Action,
3	Avengers: Infinity War	2018	8.5	1	0	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,
4	Back to the Future	1985	8.5	1	0	0	0	0	Robert Zemeckis	Adventure,Comed
5	The Good, the Bad and the Ugly	1966	8.8	1	0	1	0	0	Sergio Leone	We

Choosing Target Variable and features

Target variable --> IMDB

Categorial variable---> Year,Runtime,Directors,Genres,Country,Languag

In [28]:
```
# Choose target and features
y = df_movies.IMDb

X = df_movies.drop(['IMDb'], axis=1)

#Split the data for train and test
X_train_full, X_test_full, y_train, y_test  = train_test_split(X, y,random_sta

#List of Categorical colunmns to be used as features
cat_cols=["Directors","Genres","Country","Language",'director_avg_rating']

#List of Numerical colunmns to be used as features
numerical_cols = ['Year','Runtime']

#Keep selected columns only
my_cols = numerical_cols + cat_cols
X_train = X_train_full[my_cols].copy()
X_test = X_test_full[my_cols].copy()
from sklearn.preprocessing import LabelEncoder



# Convert categorical columns to numeric using LabelEncoder
le = LabelEncoder()
df_movies['Directors'] = le.fit_transform(df_movies['Directors'])

# Convert categorical columns to numeric using LabelEncoder
le = LabelEncoder()
df_movies['Genres'] = le.fit_transform(df_movies['Genres'])

# Convert categorical columns to numeric using LabelEncoder
le = LabelEncoder()
df_movies['Country'] = le.fit_transform(df_movies['Country'])


# Convert categorical columns to numeric using LabelEncoder
le = LabelEncoder()
df_movies['Language'] = le.fit_transform(df_movies['Language'])
```

In [29]:
```
X_train.head()
```

Out[29]:

ID	Year	Runtime	Directors	Genres	Country	Language	director_avg_rating
13576	1972	88.0	10337	224	1024	97	5.600000
9829	2011	93.0	7359	1115	1024	97	4.600000
1256	2020	104.0	10606	1580	1024	182	7.233333
9800	2017	149.0	1226	389	463	1032	5.100000
11212	1941	66.0	3935	173	1024	97	6.311111

In [30]:
```
X_test.head()
```

Out[30]:

ID	Year	Runtime	Directors	Genres	Country	Language	director_avg_rating
8874	2016	90.0	8806	1807	1024	97	5.8
12421	2016	87.0	6848	915	1024	97	6.1
8191	2018	79.0	9795	1735	1024	165	4.6
12796	2017	120.0	1829	95	1024	97	6.4
4721	2018	136.0	10760	1172	185	64	7.0

In [31]:
```
y_train.head()
```

Out[31]:
```
ID
13576    5.6
9829     4.6
1256     6.4
9800     5.1
11212    7.0
Name: IMDb, dtype: float64
```

In [32]:
```
y_test.head()
```

Out[32]:
```
ID
8874     5.8
12421    6.1
8191     4.6
12796    6.4
4721     7.0
Name: IMDb, dtype: float64
```

Linear Regression

In [33]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split



# Create linear regression object
lr = LinearRegression()

# Fit the model using the training data
lr.fit(X_train, y_train)

# Make predictions using the test data
y_pred = lr.predict(X_test)

# Calculate the accuracy of the model
accuracy = lr.score(X_test, y_test)
linear_regression_acc=accuracy*100
print("Accuracy from Linear Regression:", accuracy*100)
```

```
Accuracy from Linear Regression: 87.19203843226072
```

r2_score calculation using linear regression

In [34]:
```
from sklearn.metrics import r2_score

# Predict on the training and testing sets
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)

# Calculate the R-squared value on the training and testing sets
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

# Print the R-squared values
print("Training R-squared value: {:.2f}".format(train_r2))
print("Testing R-squared value: {:.2f}".format(test_r2))
```

```
Training R-squared value: 0.85
Testing R-squared value: 0.87
```

RandomForestRegressor

In [35]:
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

# Create a Random Forest regression object
rf = RandomForestRegressor( random_state=42)

# Fit the model using the training data
rf.fit(X_train, y_train)

# Make predictions using the test data
y_pred = rf.predict(X_test)

# Calculate the accuracy of the model
accuracy = rf.score(X_test, y_test)
random_forest_acc=accuracy*100
print("Accuracy from RandomForestRegressor:", accuracy*100)
```

```
Accuracy from RandomForestRegressor: 83.99257811349999
```

r2_score calculation using RandomForestRegressor

In [36]:
```
from sklearn.metrics import r2_score

# Predict on the training and testing sets
y_train_pred = rf.predict(X_train)
y_test_pred = rf.predict(X_test)

# Calculate the R-squared value on the training and testing sets
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

# Print the R-squared values
print("Training R-squared value: {:.2f}".format(train_r2))
print("Testing R-squared value: {:.2f}".format(test_r2))
```

```
Training R-squared value: 0.97
Testing R-squared value: 0.84
```

DecisonTreeRegressor

In [37]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score

# Create a decision tree regressor
dt = DecisionTreeRegressor()

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

# Predict on the test data
y_pred = dt.predict(X_test)

# Calculate the R-squared score (a measure of model accuracy)
accuracy = r2_score(y_test, y_pred)
decision_tree_acc=accuracy*100


# Print the accuracy score
print('Accuracy from DecisionTreeRegressor:', accuracy*100)
```

```
Accuracy from DecisionTreeRegressor: 68.28417073839233
```

r2_score calculation using DecisionTreeRegressor

In [38]:
```
from sklearn.metrics import r2_score

# Predict on the training and testing sets
y_train_pred = dt.predict(X_train)
y_test_pred = dt.predict(X_test)

# Calculate the R-squared value on the training and testing sets
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

# Print the R-squared values
print("Training R-squared value: {:.2f}".format(train_r2))
print("Testing R-squared value: {:.2f}".format(test_r2))
```

```
Training R-squared value: 1.00
Testing R-squared value: 0.68
```

In [39]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\lenovo\anaconda3\lib\site-
packages (1.7.4)
Requirement already satisfied: numpy in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from xgboost) (1.21.5)
Requirement already satisfied: scipy in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from xgboost) (1.9.1)
Note: you may need to restart the kernel to use updated packages.
```

xgboost

In [40]:
```
import xgboost as xgb
from sklearn.metrics import r2_score

# Create an XGBoost model with default parameters
xgb_model = xgb.XGBRegressor()

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

# Predict on the test data
y_pred = xgb_model.predict(X_test)

# Calculate the R-squared score (a measure of model accuracy)
accuracy = r2_score(y_test, y_pred)
xgb_acc=accuracy*100

# Print the accuracy score
print('Accuracy:', accuracy*100)
```

```
Accuracy: 84.39630977667795
```

r2_score calculation using xgboost

In [41]:
```
from sklearn.metrics import r2_score

# Predict on the training and testing sets
y_train_pred = xgb_model.predict(X_train)
y_test_pred = xgb_model.predict(X_test)

# Calculate the R-squared value on the training and testing sets
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

# Print the R-squared values
print("Training R-squared value: {:.2f}".format(train_r2))
print("Testing R-squared value: {:.2f}".format(test_r2))
```

```
Training R-squared value: 0.93
Testing R-squared value: 0.84
```

In [42]:
```
print(y_pred)
```

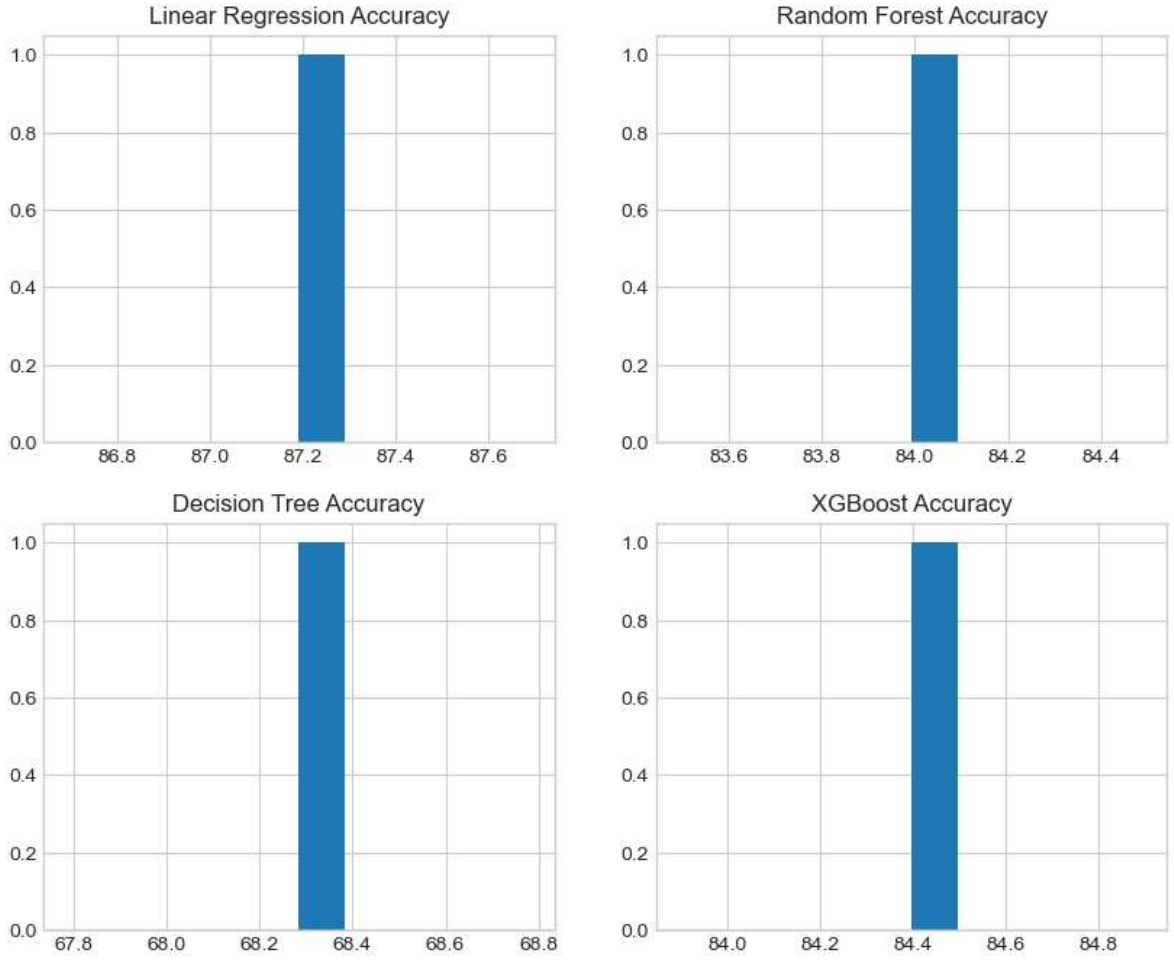
```
[5.811812  6.016548  4.6789927 ... 4.188054  6.255905  7.219863 ]
```

Individual representation of each model

In [43]:
```
# Predict on the training and testing sets
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)

# Calculate the R-squared value on the training and testing sets
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

# Print the R-squared values
print("Training R-squared value: {:.2f}".format(train_r2))
print("Testing R-squared value: {:.2f}".format(test_r2))
```

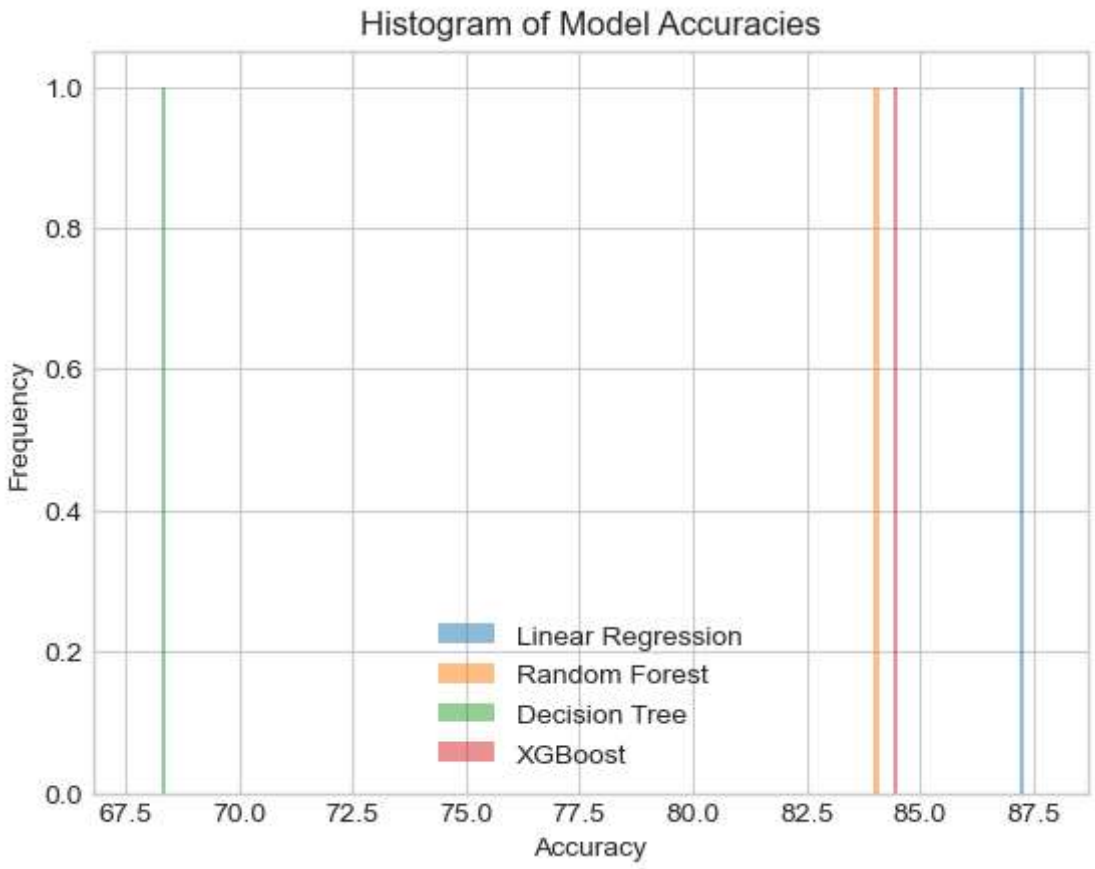
```
Training R-squared value: 0.85
Testing R-squared value: 0.87
```

In [44]:
```
# Create subplots for each model's histogram
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
axs[0, 0].hist(linear_regression_acc, bins=10)
axs[0, 0].set_title('Linear Regression Accuracy')
axs[0, 1].hist(random_forest_acc, bins=10)
axs[0, 1].set_title('Random Forest Accuracy')
axs[1, 0].hist(decision_tree_acc, bins=10)
axs[1, 0].set_title('Decision Tree Accuracy')
axs[1, 1].hist(xgb_acc, bins=10)
axs[1, 1].set_title('XGBoost Accuracy')

# Show the plot
plt.show()
```



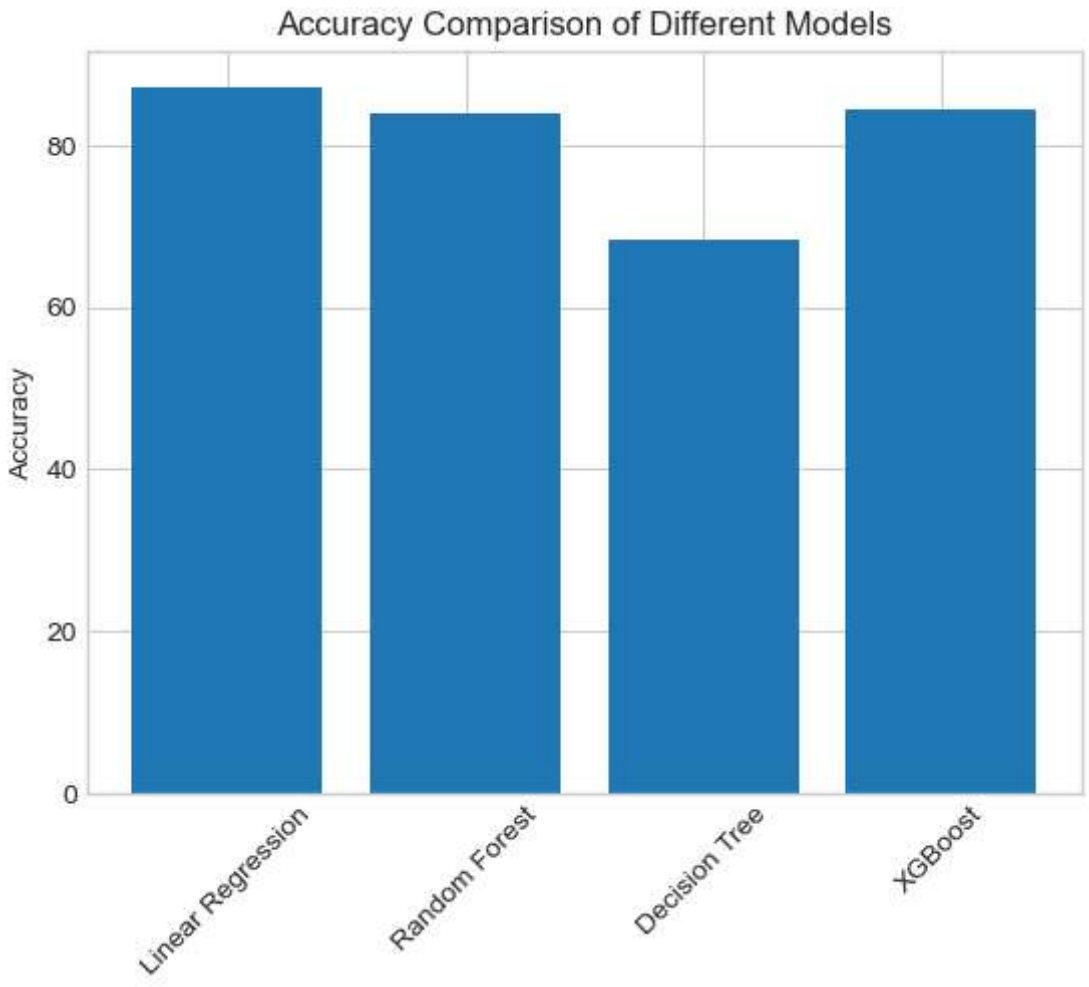
Subplot representation

In [45]:
```
# Create a histogram for each model
plt.hist(linear_regression_acc, bins=10, alpha=0.5, label='Linear Regression')
plt.hist(random_forest_acc, bins=10, alpha=0.5, label='Random Forest')
plt.hist(decision_tree_acc, bins=10, alpha=0.5, label='Decision Tree')
plt.hist(xgb_acc, bins=10, alpha=0.5, label='XGBoost')

# Add labels and legend
plt.xlabel('Accuracy')
plt.ylabel('Frequency')
plt.title('Histogram of Model Accuracies')
plt.legend()

# Show the plot
plt.show()
```



Histogram representation of model accuracy

In [46]:
```
# Create a list of model names
models = ['Linear Regression', 'Random Forest', 'Decision Tree', 'XGBoost']
fig, ax = plt.subplots()
accuracy = [linear_regression_acc, random_forest_acc, decision_tree_acc, xgb_a

# Create a histogram with the sample data
ax.bar(models, accuracy)

# Add labels and title
ax.set_xticks(np.arange(len(models)))
ax.set_xticklabels(models, rotation=45)
ax.set_ylabel('Accuracy')
ax.set_title('Accuracy Comparison of Different Models')

# Show the plot
plt.show()
```



In [47]:
```
import warnings
warnings.filterwarnings("ignore", message="X does not have valid feature names
```

In [48]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression



# Encode the categorical variables using label encoding
le = LabelEncoder()
df_movies["Directors"] = le.fit_transform(df_movies["Directors"])
df_movies["Genres"] = le.fit_transform(df_movies["Genres"])
df_movies["Country"] = le.fit_transform(df_movies["Country"])
df_movies["Language"] = le.fit_transform(df_movies["Language"])

# Separate the features and the target variable
X = df_movies[["Year", "Runtime", "Directors", "Genres", "Country", "Language"
y = df_movies["IMDb"]

# Create linear regression object
lr = LinearRegression()

# Fit the model using the entire dataset
lr.fit(X, y)

# Make a prediction using the input values
input_values = [[2020, 100, 205, 2, 401, 2, 8.2]]
predicted_rating = lr.predict(input_values)

# Print the predicted rating
print("Predicted IMDb rating:", predicted_rating[0])
```

```
Predicted IMDb rating: 8.13117171343653
```

```
Conclusion: From the above experiment i have learned how to implement
miniproject on Movie Rating Prediction Project and i also learned how to use
different machine learning model in this experiment i have achieved
PO1,PO2,PO3,PO4,PO5,PO8,PO9,PO10,PO12
```