Phase 1

In [53]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("C:\\Users\\Alvin Roy\\Downloads\\NetflixOriginals.csv", encoding=
df
```

Out[2]:

	Title	Genre	Premiere	Runtime	IMDB Score	Language
0	Enter the Anime	Documentary	August 5, 2019	58	2.5	English/Japanese
1	Dark Forces	Thriller	August 21, 2020	81	2.6	Spanish
2	The App	Science fiction/Drama	December 26, 2019	79	2.6	Italian
3	The Open House	Horror thriller	January 19, 2018	94	3.2	English
4	Kaali Khuhi	Mystery	October 30, 2020	90	3.4	Hindi
...	...	...	...	...	...	...
579	Taylor Swift: Reputation Stadium Tour	Concert Film	December 31, 2018	125	8.4	English
580	Winter on Fire: Ukraine's Fight for Freedom	Documentary	October 9, 2015	91	8.4	English/Ukranian/Russian
581	Springsteen on Broadway	One-man show	December 16, 2018	153	8.5	English
582	Emicida: AmarElo - It's All For Yesterday	Documentary	December 8, 2020	89	8.6	Portuguese
583	David Attenborough: A Life on Our Planet	Documentary	October 4, 2020	83	9.0	English

584 rows × 6 columns

In [3]:
```
df.info
```

Out[3]:
```
<bound method DataFrame.info of                                                 Title
Genre  \
0                                 Enter the Anime           Documentary
1                                     Dark Forces              Thriller
2                                         The App  Science fiction/Drama
3                                  The Open House       Horror thriller
4                                     Kaali Khuhi               Mystery
..                                            ...                   ...
579        Taylor Swift: Reputation Stadium Tour          Concert Film
580  Winter on Fire: Ukraine's Fight for Freedom           Documentary
581                       Springsteen on Broadway          One-man show
582    Emicida: AmarElo - It's All For Yesterday           Documentary
583     David Attenborough: A Life on Our Planet           Documentary

              Premiere  Runtime  IMDB Score                 Language
0       August 5, 2019       58         2.5          English/Japanese
1      August 21, 2020       81         2.6                  Spanish
2    December 26, 2019       79         2.6                  Italian
3     January 19, 2018       94         3.2                  English
4     October 30, 2020       90         3.4                    Hindi
..                 ...      ...         ...                      ...
579  December 31, 2018      125         8.4                  English
580    October 9, 2015       91         8.4  English/Ukranian/Russian
581  December 16, 2018      153         8.5                  English
582   December 8, 2020       89         8.6               Portuguese
583    October 4, 2020       83         9.0                  English

[584 rows x 6 columns]>
```

In [4]:
```
df.describe
```

Out[4]:
```
<bound method NDFrame.describe of                                                 Title
Genre  \
0                                 Enter the Anime           Documentary
1                                     Dark Forces              Thriller
2                                         The App  Science fiction/Drama
3                                  The Open House       Horror thriller
4                                     Kaali Khuhi               Mystery
..                                            ...                   ...
579        Taylor Swift: Reputation Stadium Tour          Concert Film
580  Winter on Fire: Ukraine's Fight for Freedom           Documentary
581                       Springsteen on Broadway          One-man show
582    Emicida: AmarElo - It's All For Yesterday           Documentary
583     David Attenborough: A Life on Our Planet           Documentary

              Premiere  Runtime  IMDB Score                 Language
0       August 5, 2019       58         2.5          English/Japanese
1      August 21, 2020       81         2.6                  Spanish
2    December 26, 2019       79         2.6                  Italian
3     January 19, 2018       94         3.2                  English
4     October 30, 2020       90         3.4                    Hindi
..                 ...      ...         ...                      ...
579  December 31, 2018      125         8.4                  English
580    October 9, 2015       91         8.4  English/Ukranian/Russian
581  December 16, 2018      153         8.5                  English
582   December 8, 2020       89         8.6               Portuguese
583    October 4, 2020       83         9.0                  English

[584 rows x 6 columns]>
```

In [40]:
```
df.rename(columns = {'IMDB Score':'IMDBScore'}, inplace = True)
df
```

Out[40]:

	Title	Genre	Premiere	Runtime	IMDBScore	Language
0	Enter the Anime	Documentary	August 5, 2019	58	2.5	English/Japanese
1	Dark Forces	Thriller	August 21, 2020	81	2.6	Spanish
2	The App	Science fiction/Drama	December 26, 2019	79	2.6	Italian
3	The Open House	Horror thriller	January 19, 2018	94	3.2	English
4	Kaali Khuhi	Mystery	October 30, 2020	90	3.4	Hindi
...	...	...	...	...	...	...
579	Taylor Swift: Reputation Stadium Tour	Concert Film	December 31, 2018	125	8.4	English
580	Winter on Fire: Ukraine's Fight for Freedom	Documentary	October 9, 2015	91	8.4	English/Ukranian/Russian
581	Springsteen on Broadway	One-man show	December 16, 2018	153	8.5	English
582	Emicida: AmarElo - It's All For Yesterday	Documentary	December 8, 2020	89	8.6	Portuguese
583	David Attenborough: A Life on Our Planet	Documentary	October 4, 2020	83	9.0	English

584 rows × 6 columns

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Title         0
Genre         0
Premiere      0
Runtime       0
IMDB Score    0
Language      0
dtype: int64
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Title', 'Genre', 'Premiere', 'Runtime', 'IMDB Score', 'Language'], dtype
='object')
```

In [7]:
```
language=df['Language'].value_counts().sort_values(ascending=False)
language=language[:10]
language
```

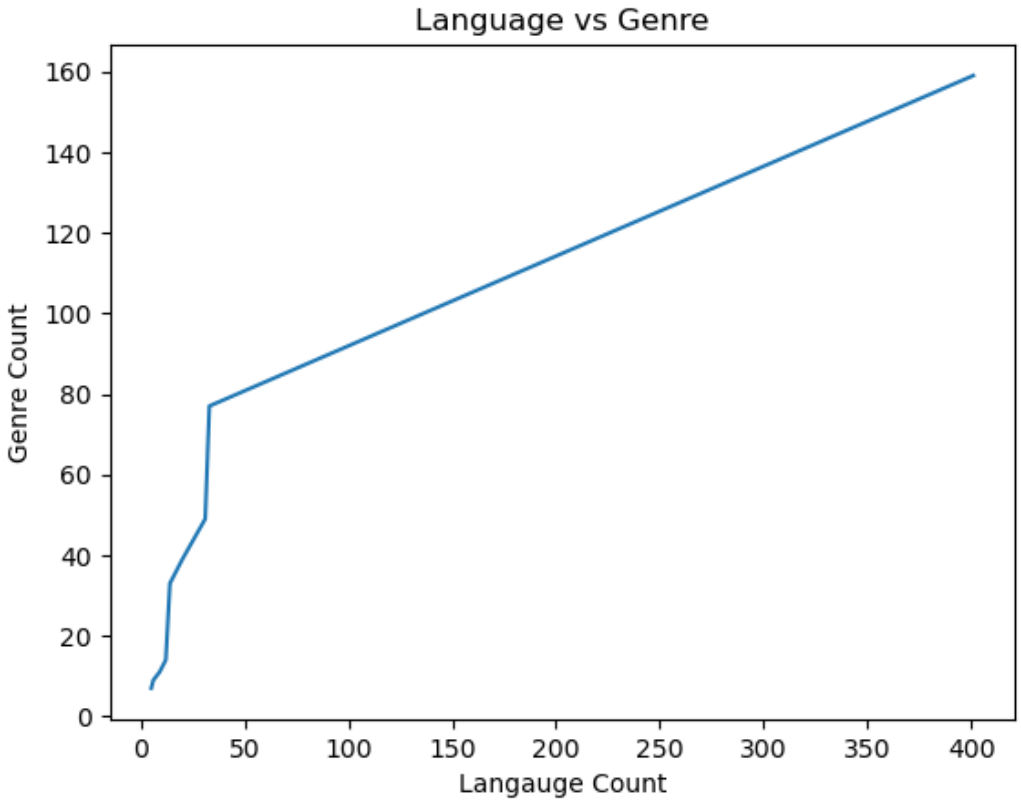
Out[7]:
```
Language
English       401
Hindi          33
Spanish        31
French         20
Italian        14
Portuguese     12
Indonesian      9
Korean          6
Japanese        6
German          5
Name: count, dtype: int64
```

In [8]:
```
Genre=df["Genre"].value_counts().sort_values(ascending=False)
Genre=Genre[:10]
Genre
```

Out[8]:
```
Genre
Documentary        159
Drama               77
Comedy              49
Romantic comedy     39
Thriller            33
Comedy-drama        14
Crime drama         11
Biopic               9
Horror               9
Action               7
Name: count, dtype: int64
```

In [9]:
```
plt.plot(language,Genre)
plt.xlabel("Langauge Count")
plt.ylabel("Genre Count")
plt.title("Language vs Genre")
```

Out[9]:
```
Text(0.5, 1.0, 'Language vs Genre')
```



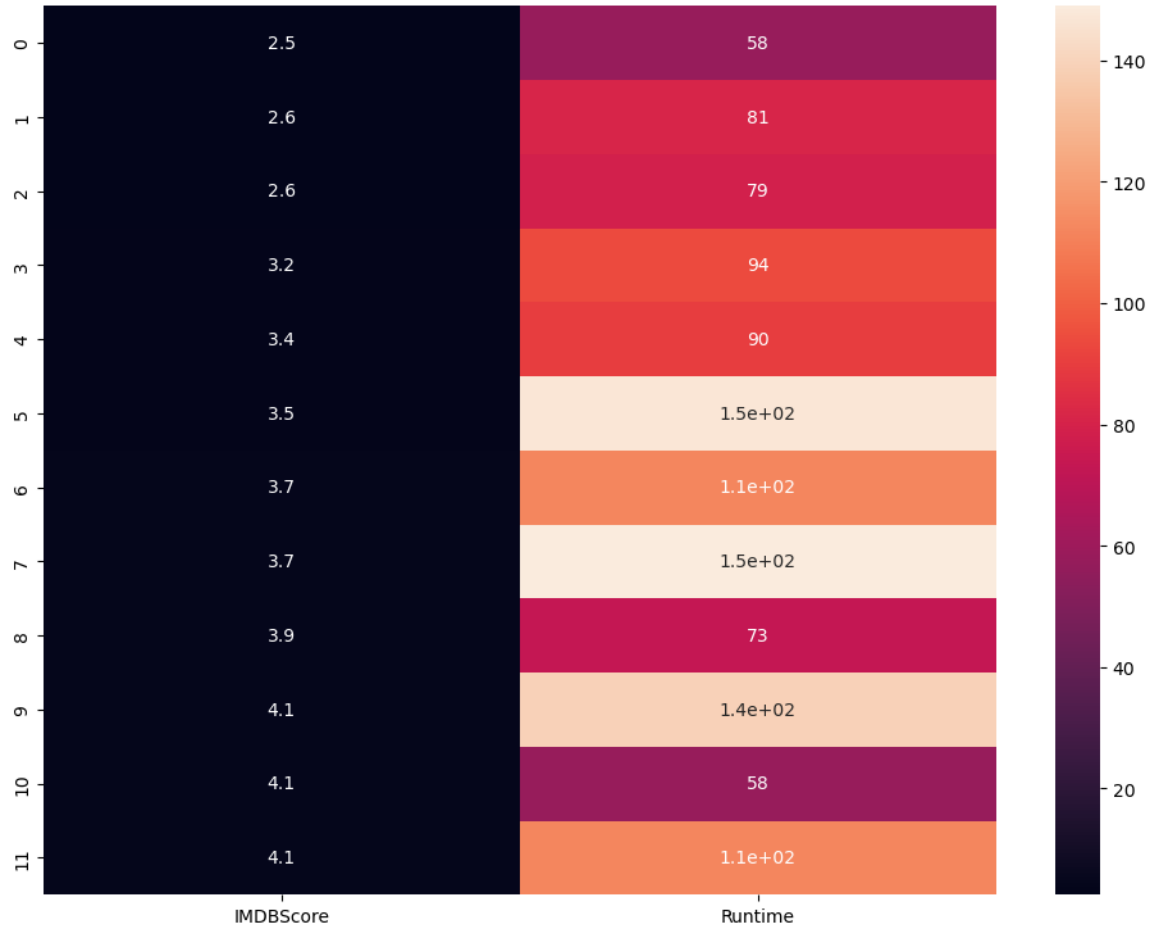
In [58]:
```
cols=["IMDBScore","Runtime"]
x=df[cols].head(12)

x.corr()
```

Out[58]:

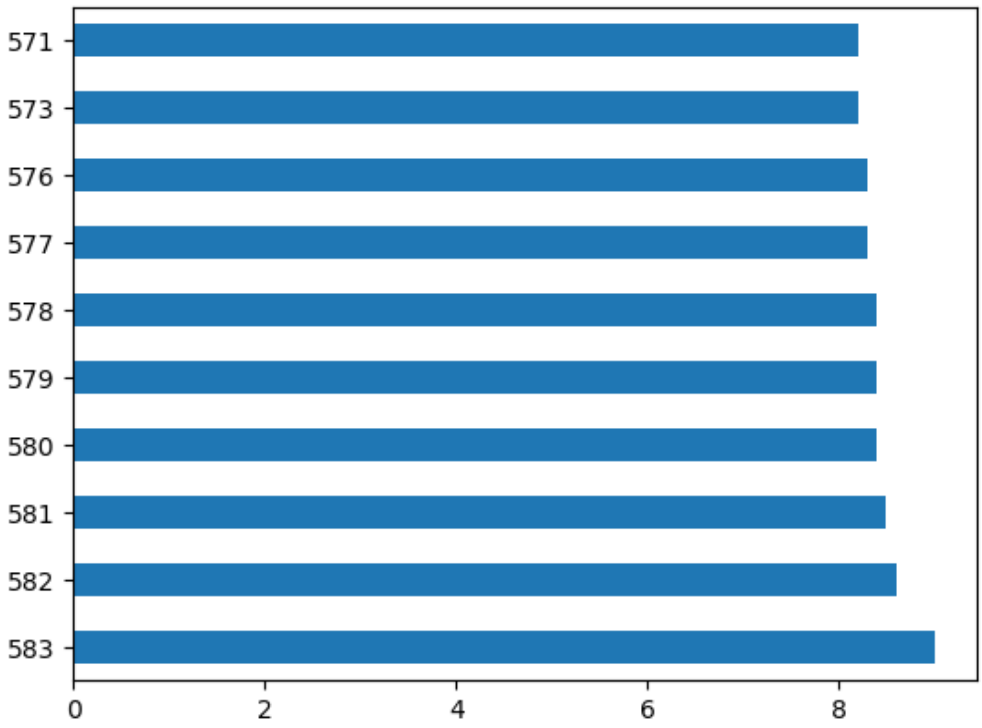
	IMDBScore	Runtime
IMDBScore	1.00000	0.40416
Runtime	0.40416	1.00000

In [60]:
```
plt.figure(figsize=(12,9))
sns.heatmap(x,annot=True)
```

Out[60]: <Axes: >

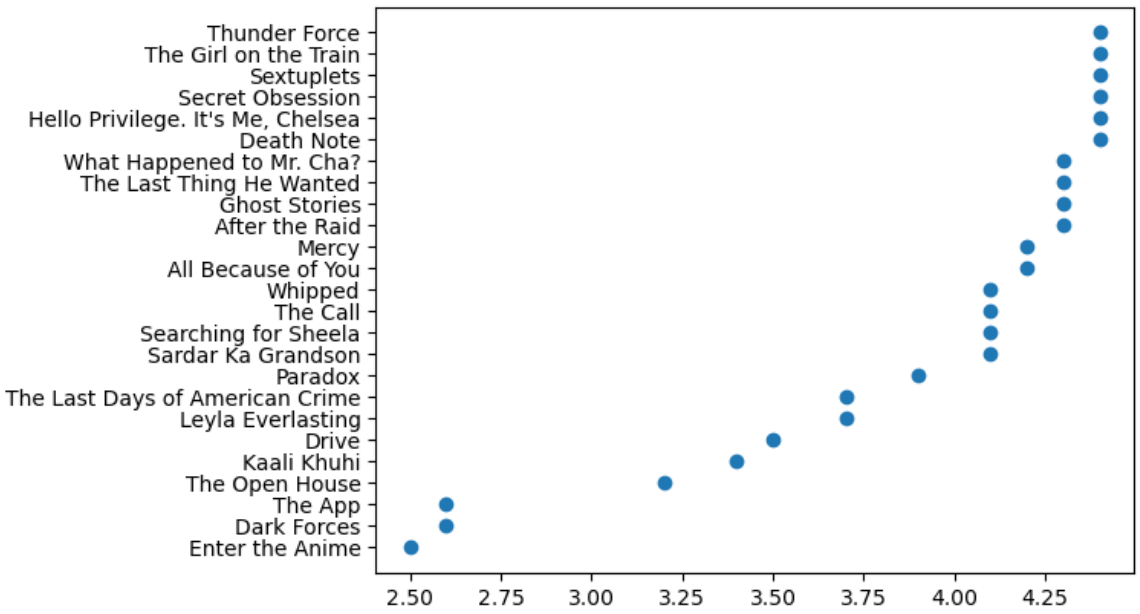


In [62]:
```
score=df["IMDBScore"].sort_values(ascending=False)[:10]
score.plot.barh()
plt.show()
```



In [72]:
```
plt.scatter(df["IMDBScore"][:25],df["Title"][:25])
plt.figure(figsize=(30,30))
```

Out[72]:
```
<Figure size 3000x3000 with 0 Axes>
```




```
<Figure size 3000x3000 with 0 Axes>
```

In [83]:
```
from sklearn.preprocessing import LabelEncoder
object_cols = ["Title","Genre"]
label_encoder = LabelEncoder()
df2=df.copy()
for col in object_cols:
    label_encoder.fit(df2[col])
    df2[col] = label_encoder.transform(df2[col])
df2
```

Out[83]:

	Title	Genre	Premiere	Runtime	IMDBScore	Language
0	147	45	August 5, 2019	58	2.5	English/Japanese
1	120	106	August 21, 2020	81	2.6	Spanish
2	433	93	December 26, 2019	79	2.6	Italian
3	500	63	January 19, 2018	94	3.2	English
4	243	73	October 30, 2020	90	3.4	Hindi
...	...	...	...	...	...	...
579	425	40	December 31, 2018	125	8.4	English
580	575	45	October 9, 2015	91	8.4	English/Ukranian/Russian
581	410	74	December 16, 2018	153	8.5	English
582	145	45	December 8, 2020	89	8.6	Portuguese
583	121	45	October 4, 2020	83	9.0	English

584 rows × 6 columns

In [ ]:
```
df2.drop(["Premiere"],axis=1,inplace=True)
```

In [ ]:
```
df2.drop(["Language"],axis=1,inplace=True)
```

In [93]:
```
df2.drop(["IMDBScore"],axis=1,inplace=True)

df2
```

Out[93]:

	Title	Genre	Runtime
0	147	45	58
1	120	106	81
2	433	93	79
3	500	63	94
4	243	73	90
...	...	...	...
579	425	40	125
580	575	45	91
581	410	74	153
582	145	45	89
583	121	45	83

584 rows × 3 columns

In [85]:
```
y=df["IMDBScore"]
y
```

Out[85]:
```
0      2.5
1      2.6
2      2.6
3      3.2
4      3.4
      ...
579    8.4
580    8.4
581    8.5
582    8.6
583    9.0
Name: IMDBScore, Length: 584, dtype: float64
```

In [94]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [95]:
```
xtrain,xtest,ytrain,ytest=train_test_split(df2,y,test_size=0.2,random_state=42)
```

In [96]:
```
print(xtrain.shape,ytrain.shape)
```

```
(467, 3) (467,)
```

In [97]:
```
print(xtest.shape,ytest.shape)
```

```
(117, 3) (117,)
```

In [98]:
```
lr=LinearRegression()
```

In [99]:
```
lr.fit(xtrain,ytrain)
```

Out[99]:
```
▾ LinearRegression
LinearRegression()
```

In [100]:
```
lr.fit(xtest,ytest)
```

Out[100]:
```
LinearRegression
LinearRegression()
```

In [176]:
```
lr.score(xtrain,ytrain)
```

Out[176]:
```
0.031119103871776854
```

In [105]:
```
y_pred=lr.predict(xtest)
x=[[147,45,58]]
predict=lr.predict(x)
predict
```

```
C:\ProgramData\anaconda3navAIMLDS\Lib\site-packages\sklearn\base.py:464: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted with feat
ure names
  warnings.warn(
```

Out[105]:
```
array([6.3065107])
```

In [104]:
```
print(lr.score(xtest, ytest))
```

```
0.019033211731676047
```

In [127]:
```
# Assuming df2 is your first DataFrame and df is your second DataFrame

# Filter rows in df2 where 'Title' is equal to 147
filtered_rows_df2 = df2[df2['Title'] == 147]

# Get the indices of the filtered rows
indices = filtered_rows_df2.index

# Use the indices to access the corresponding rows in df
resulting_rows_df = df.loc[indices]

# Now, resulting_rows_df contains the rows from df where 'Title' is equal to 147 in
resulting_rows_df
```

Out[127]:

	Title	Genre	Premiere	Runtime	IMDBScore	Language
0	Enter the Anime	Documentary	August 5, 2019	58	2.5	English/Japanese

In [151]:
```
from sklearn.ensemble import RandomForestRegressor

# create regressor object
regressor = RandomForestRegressor(n_estimators=100,
                                  random_state=0)

# fit the regressor with x and y data
regressor.fit(df2, y)
```

Out[151]:
```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

In [152]:
```
Y_pred = regressor.predict(xtest)
```

In [153]:
```
print(regressor.score(xtest, ytest))
```

```
0.8844757926660178
```

In [177]:
```
print(regressor.score(xtrain, ytrain))
```

```
0.8848317625613643
```

In [154]:
```
x=[[147,45,58]]
predict=regressor.predict(x)
predict
```

```
C:\ProgramData\anaconda3navAIMLDS\Lib\site-packages\sklearn\base.py:464: UserWarni
ng: X does not have valid feature names, but RandomForestRegressor was fitted with
feature names
  warnings.warn(
```

Out[154]:
```
array([3.752])
```

In [168]:
```
from sklearn.metrics import accuracy_score
```

In [204]:
```
import numpy as np

# Assuming ytest is a Pandas Series
ytest_array = np.array(ytest)
reshaped_ytest = ytest_array.reshape(-1, 1)
print(reshaped_ytest)

ytpred_array = np.array(y_pred)
reshaped_ypred = ytpred_array.reshape(-1, 1)
print(reshaped_ypred)
```

```
[6.30083192]
 [6.04885355]
 [6.35645981]
 [6.32326864]
 [6.2720176 ]
 [6.35540359]
 [6.08248077]
 [6.33297099]
 [6.30542195]
 [6.3537964 ]
 [6.32050545]
 [5.99773693]
 [6.07130372]
 [6.27105608]
 [5.96658017]
 [6.11709374]
 [6.50468044]
 [6.32829845]
 [6.23605459]
 [5.99847588]]
```

In [209]:
```
from sklearn.metrics import mean_squared_error, mean_squared_log_error,r2_score
```

In [212]:
```
mse = mean_squared_error(ytest, y_pred)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
r2_score=r2_score(ytest,y_pred)  # For the Linear Regression
print(rmse,mse,r2_score)
```

```
1.00905205719926 1.018186054138059 0.019033211731676047
```

In [224]:
```
Y_pred1 = Y_pred.astype(int)
Y_pred1
```

Out[224]:
```
array([6, 6, 5, 6, 7, 6, 5, 5, 5, 6, 7, 5, 7, 5, 6, 6, 5, 5, 8, 6, 7, 5,
       3, 7, 7, 5, 5, 7, 6, 4, 5, 6, 5, 6, 5, 5, 6, 7, 7, 7, 6, 6, 6, 5,
       7, 6, 4, 6, 6, 6, 5, 6, 5, 5, 5, 6, 7, 6, 6, 6, 5, 6, 6, 5, 5, 6,
       5, 7, 6, 5, 6, 6, 5, 5, 6, 6, 5, 5, 6, 7, 6, 4, 5, 6, 7, 5, 6, 7,
       6, 6, 6, 7, 5, 6, 6, 6, 6, 4, 6, 7, 6, 7, 6, 5, 6, 7, 6, 6, 5, 5,
       5, 5, 6, 7, 5, 7, 6])
```

In [226]:
```
mse = mean_squared_error(ytest, Y_pred1)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print(rmse,mse)
```

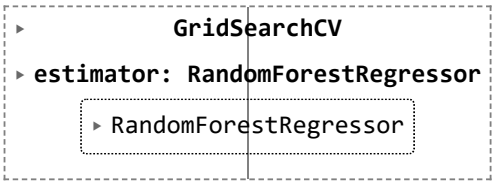
```
0.6717116801102846 0.45119658119658124
```

In [229]:
```
from sklearn.metrics import r2_score

# Assuming ytest contains actual labels and Y_pred1 contains predicted labels
r2_result = r2_score(ytest, Y_pred1)  # Calculate the R^2 score
r2_result
```

Out[229]:
```
0.565296677031442
```

In [230]:
```
from sklearn.model_selection import train_test_split, GridSearchCV

param_grid = {
    'n_estimators': [100, 200, 300],        # Number of trees
    'max_depth': [None, 10, 20, 30],       # Maximum depth of each tree
    'min_samples_split': [2, 5, 10],      # Minimum samples required to split a node
    'min_samples_leaf': [1, 2, 4]         # Minimum samples required at each leaf no
}
grid_search = GridSearchCV(estimator=regressor, param_grid=param_grid, cv=5, scoring
grid_search.fit(xtrain, ytrain)
```

Out[230]:



In [232]:
```
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Train the model with the best hyperparameters
best_rf_regressor = grid_search.best_estimator_
best_rf_regressor.fit(xtrain, ytrain)

# Make predictions on the test set
ypred = best_rf_regressor.predict(xtest)

# Evaluate the model
mse = mean_squared_error(ytest, ypred)
rmse = np.sqrt(mse)
r2 = r2_score(ytest, ypred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Best Hyperparameters: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_spli
t': 10, 'n_estimators': 100}
Mean Squared Error: 0.7810925200023665
Root Mean Squared Error: 0.8837943878540792
R-squared: 0.24745991405688794
```

Phase 2

Neural Network

In [233]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import layers
```

In [236]:

```
x = df[['Genre', 'Runtime', 'Language']]
y = df['IMDBScore']

# Encode categorical variables (Genre and Language) using one-hot encoding
x = pd.get_dummies(x, columns=['Genre', 'Language'], drop_first=True)

# Split the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_stat

# Standardize features (optional but can help neural networks converge faster)
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

# Build the neural network model
model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(x_train.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)  # Output layer for regression
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mean_absolute_e

# Train the model
model.fit(x_train, y_train, epochs=100, batch_size=32, validation_split=0.2)

# Evaluate the model on the test set
loss, mae = model.evaluate(x_test, y_test)
print(f"Mean Absolute Error on Test Set: {mae}")

# Make predictions
y_pred = model.predict(x_test)
```

```
Epoch 1/100
WARNING:tensorflow:From C:\ProgramData\anaconda3navAIMLDS\Lib\site-packages\kera
s\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated.
Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\ProgramData\anaconda3navAIMLDS\Lib\site-packages\kera
s\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_func
tions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions
instead.

12/12 [==============================] - 2s 26ms/step - loss: 33.2785 - mean_abs
olute_error: 5.6386 - val_loss: 28.2573 - val_mean_absolute_error: 5.1245
Epoch 2/100
12/12 [==============================] - 0s 6ms/step - loss: 22.5230 - mean_abso
lute_error: 4.4576 - val_loss: 19.7774 - val_mean_absolute_error: 4.1564
Epoch 3/100
12/12 [==============================] - 0s 6ms/step - loss: 14.6240 - mean_abso
lute_error: 3.4155 - val_loss: 12.6452 - val_mean_absolute_error: 3.2267
Epoch 4/100
12/12 [==============================] - 0s 6ms/step - loss: 9.4104 - mean_absol
```

In [237]:
```
y_pred
```

Out[237]:
```
array([[6.1771364],
       [4.839865 ],
       [5.5921364],
       [6.9115047],
       [6.8096337],
       [7.0572805],
       [6.0356617],
       [5.3591495],
       [5.0072885],
       [5.0961967],
       [7.2368755],
       [7.0786686],
       [7.1232686],
       [5.360663 ],
       [6.6132364],
       [5.377314 ],
       [5.0788856],
       [4.9926243],
       [7.2731147],
```

In [239]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load the dataset
# data = pd.read_csv("NetflixOriginals.csv")

# Preprocess the data
# Assuming you want to use 'Genre', 'Runtime', and 'Language' as features
X = df[['Genre', 'Runtime', 'Language']]
y = df['IMDBScore']

# Encode categorical variables (Genre and Language) using Label Encoding
label_encoders = {}
for col in ['Genre', 'Language']:
    label_encoders[col] = LabelEncoder()
    X[col] = label_encoders[col].fit_transform(X[col])

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat

# Standardize features (optional but can help gradient boosting)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Build and train the gradient boosting model
regressor = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth
regressor.fit(X_train, y_train)

# Make predictions
y_pred = regressor.predict(X_test)

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

print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared (R2): {r2}")
```

```
Mean Absolute Error: 0.6737418660432508
Mean Squared Error: 0.7646858702215756
R-squared (R2): 0.2632668272200527

C:\Users\Alvin Roy\AppData\Local\Temp\ipykernel_9888\1193338420.py:20: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.o
rg/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X[col] = label_encoders[col].fit_transform(X[col])
C:\Users\Alvin Roy\AppData\Local\Temp\ipykernel_9888\1193338420.py:20: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.o
rg/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X[col] = label_encoders[col].fit_transform(X[col])
```