# IMDb PREDICTING SCORES

## TEAM MEMBER

732121104053:RANJANADEVI C



Problem Statement:

The task is to predict movie scores (ratings) using IMDb's available data. This is essentially a regression problem where we aim to forecast a movie's rating based on certain features available in the dataset.

Procedure:

Data Collection: Gather a comprehensive dataset from IMDb or other reliable sources. The dataset should contain various features like genre, director, cast, budget, runtime, release date, etc.

Data Preprocessing: Clean the dataset by handling missing values, encoding categorical variables, and scaling numerical features if necessary. Feature selection may be required to determine which attributes significantly impact the movie rating.

Feature Engineering: Extract valuable features or create new ones that might influence the movie's rating. This might involve sentiment analysis on reviews, creating a feature for the director's previous successes, or any other relevant metric.

Model Selection: Choose appropriate regression models (e.g., linear regression, decision trees, random forests, gradient boosting) to train on thedataset. Consider ensemble methods or even deep learning if necessary. Utilize cross-validation to evaluate each model's performance.

Model Training Split the dataset into training and testing sets. Train the selected models on the training set and fine-tune the hyperparameters using techniques like grid search or random search.

Model Evaluation: Evaluate the models' performance using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared to select the best performing mode

# PROGRAM:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import scipy.stats

from datetime import date

import plotly.graph\_objects as go

import plotly.express as px

*#Analysis the Data*

df = pd.read\_csv('../input/netflix-original-films-imdb-scores/NetflixOriginals.csv', encoding='latin-1')

df.head()

df.isna().sum()

df.dtypes

df["Runtime"]=pd.to\_numeric(df['Runtime'])

d1=df.groupby("Language").agg({"Runtime": "max"}).reset\_index()

df1=d1.loc[(d1['Runtime'])>=120]

sns.barplot(x="Runtime", y="Language", data=df1, palette="pastel").set\_title("Long-running films by language");

d2=df.loc[(pd.DatetimeIndex(df['Premiere'])>"2019-01-01") & (pd.DatetimeIndex(df['Premiere'])<"2020-06-01") & (df["Genre"]=="Documentary")]

(sns.barplot(x='IMDB Score', y='Title', data=d2.head(), palette= "Set2")

.set\_title("IMDB Score of Documentary Genre Between January 2019 and June 2020"));

d3=df.loc[df["Language"]=="English"].sort\_values(["IMDB Score"], ascending=False)

d3.reset\_index(drop=True)

d3[["Genre","IMDB Score"]].head(1)

df.loc[df["Language"]=="Hindi"][["Language",'Runtime']].groupby(["Language"]).agg({"Runtime":np.mean})

d5=df["Genre"].value\_counts().head()

sns.lineplot(y=d5.index, x=d5.values).set\_title("Genre Categories")

plt.show()

df["count"]=1

d6=df.groupby(["Language"]).count()["count"].sort\_values(ascending=False).head(3)

sns.barplot(x=d6.index, y=d6.values, palette="ocean").set\_title("Most Used Languages In Movies");

df[["Title","IMDB Score"]].sort\_values("IMDB Score", ascending=False).head(10).reset\_index(drop=True)

pear=scipy.stats.pearsonr(df["IMDB Score"], df["Runtime"])

spear=scipy.stats.spearmanr(df["IMDB Score"], df["Runtime"])

sns.scatterplot(x=df["IMDB Score"], y=df["Runtime"], color="green").set\_title("Correlation Between 'IMDB Score' and 'Runtime");

print(f"Correlation of Pearson: **{**pear[0]**}**")

print(f"Correlation of Spearman: **{**spear[0]**}**")

print("👎There is no correlation between Runtime and IMDB Score because the correlation coefficient is too low.")

Correlation of Pearson: -0.0408962914207887

Correlation of Spearman: -0.022930090743463378

👎There is no correlation between Runtime and IMDB Score because the correlation coefficient is too low.

d9= df.groupby("Genre").agg({"IMDB Score": "max"}).sort\_values("IMDB Score", ascending=False)[0:10].reset\_index()

sns.catplot(x="IMDB Score", y="Genre", data=d9, kind="point", color="purple");

d10=df[["Title","Runtime"]].sort\_values("Runtime", ascending=False).head(10).reset\_index(drop=True)

sns.barplot(y="Title", x="Runtime",errcolor="red", data=d10).set\_title("Top 10 Movies With The Highest Runtime");

df['year'] = pd.DatetimeIndex(df.Premiere).year.astype(int)

year = df['year'].value\_counts()

fig = px.bar(df, x=year.index, y=year.values, labels={"y":"Counts of Films", "x":"Year"})

fig.update\_traces(marker\_color='#aea1eb')

fig.show()

d12=df.groupby(["Language"]).agg({"IMDB Score":np.mean}).reset\_index().sort\_values("IMDB Score").head()

fig = px.bar(d12, y=d12["Language"], x=d12["IMDB Score"], labels={"y":"Language", "x":"IMDB Score"})

fig.update\_traces(marker\_color='#ffb3ba')

fig.show()

df[["year", "Runtime"]].groupby(["year"]).sum().sort\_values(["Runtime"], ascending=False).head(1)

d14=(df.groupby(["Language", "Genre"]).size()

.sort\_values(ascending=False)

.reset\_index(name='Count')

.drop\_duplicates(subset='Language'))

d14.head(10)

def find\_outlier(data):

sns.boxplot(x=data);

q1, q3= np.percentile(data, [25,75])

iqr=q3-q1

lower\_bound = q1 - (1.5 \* iqr)

upper\_bound = q3 + (1.5 \* iqr)

data\_outlier=data[(data<lower\_bound) | (data>upper\_bound)]

count\_of\_outliers=data[(data<lower\_bound) | (data>upper\_bound)].count()

print(f"Info about outlier data for **{**data.name**}**:")

print(f"**{**count\_of\_outliers**}** outlier datas.")

print("Outlier datas: ", data[(data<lower\_bound) | (data>upper\_bound)], sep="**\n**")

m=df["IMDB Score"]

n=df["Runtime"]

find\_outlier(m)

find\_outlier(n)

# OUTPUT:

In [2]:

*#Analysis the Data*

df = pd.read\_csv('../input/netflix-original-films-imdb-scores/NetflixOriginals.csv', encoding='latin-1')

df.head()

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 |

In [3]:

df.isna().sum()

Out[3]:

Title 0

Genre 0

Premiere 0

Runtime 0

IMDB Score 0

Language 0

dtype: int64

In [4]:

df.dtypes

Out[4]:

Title object

Genre object

Premiere object

Runtime int64

IMDB Score float64

Language object

dtype: object

According to the dataset, in which language are the long-running films created (Long-running films will be considered as 120 minutes and over.)?

In [5]:

df["Runtime"]=pd.to\_numeric(df['Runtime'])

In [6]:

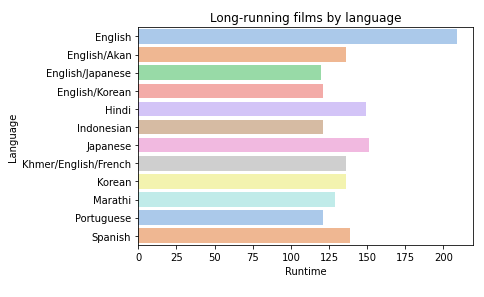
d1=df.groupby("Language").agg({"Runtime": "max"}).reset\_index()

In [7]:

df1=d1.loc[(d1['Runtime'])>=120]

In [8]:

sns.barplot(x="Runtime", y="Language", data=df1, palette="pastel").set\_title("Long-running films by language");



Find the IMDB values of the movies filmed in the 'Documentary' genre between January 2019 and June 2020.

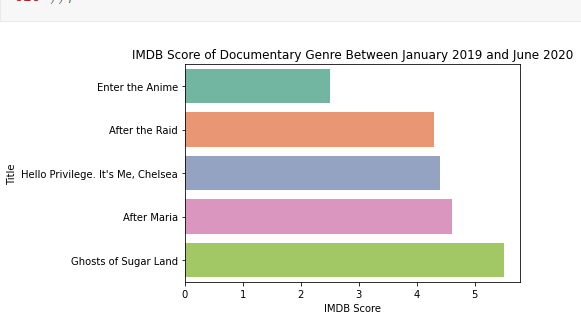
In [9]:

d2=df.loc[(pd.DatetimeIndex(df['Premiere'])>"2019-01-01") & (pd.DatetimeIndex(df['Premiere'])<"2020-06-01") & (df["Genre"]=="Documentary")]

In [10]:

(sns.barplot(x='IMDB Score', y='Title', data=d2.head(), palette= "Set2")

.set\_title("IMDB Score of Documentary Genre Between January 2019 and June 2020"));



Which genre has the highest IMDB rating among movies filmed in only English?

In [11]:

d3=df.loc[df["Language"]=="English"].sort\_values(["IMDB Score"], ascending=False)

d3.reset\_index(drop=True)

d3[["Genre","IMDB Score"]].head(1)

Out[11]:

|  | Genre | IMDB Score |
| --- | --- | --- |
| 583 | Documentary | 9.0 |

What is the average 'runtime' of movies filmed in 'Hindi'?

In [12]:

df.loc[df["Language"]=="Hindi"][["Language",'Runtime']].groupby(["Language"]).agg({"Runtime":np.mean})

Out[12]:

|  | Runtime |
| --- | --- |
| Language |  |
| Hindi | 115.787879 |

How many categories does the 'Genre' Column have and what are those categories?

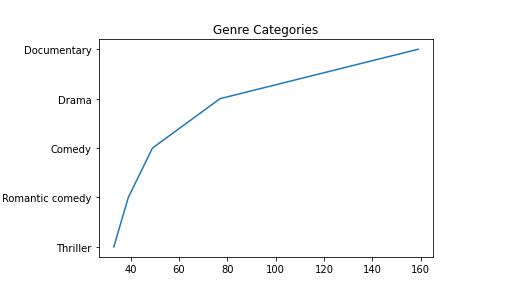
In [13]:

d5=df["Genre"].value\_counts().head()

In [14]:

sns.lineplot(y=d5.index, x=d5.values).set\_title("Genre Categories")

plt.show()



What are the three most used languages in movies?

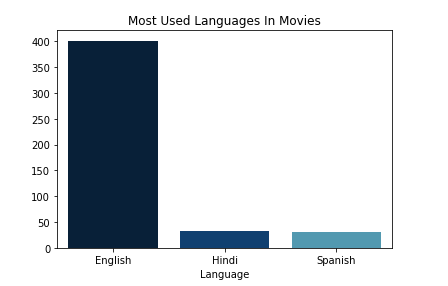
In [15]:

df["count"]=1

d6=df.groupby(["Language"]).count()["count"].sort\_values(ascending=False).head(3)

In [16]:

sns.barplot(x=d6.index, y=d6.values, palette="ocean").set\_title("Most Used Languages In Movies");



What are the top 10 movies with the highest IMDB rating?

In [17]:

df[["Title","IMDB Score"]].sort\_values("IMDB Score", ascending=False).head(10).reset\_index(drop=True)

Out[17]:

|  | Title | IMDB Score |
| --- | --- | --- |
| 0 | David Attenborough: A Life on Our Planet | 9.0 |
| 1 | Emicida: AmarElo - It's All For Yesterday | 8.6 |
| 2 | Springsteen on Broadway | 8.5 |
| 3 | Winter on Fire: Ukraine's Fight for Freedom | 8.4 |
| 4 | Taylor Swift: Reputation Stadium Tour | 8.4 |
| 5 | Ben Platt: Live from Radio City Music Hall | 8.4 |
| 6 | Dancing with the Birds | 8.3 |
| 7 | Cuba and the Cameraman | 8.3 |
| 8 | Klaus | 8.2 |
| 9 | 13th | 8.2 |

What is the correlation between IMDB score and 'Runtime'? Examine and visualize.

In [18]:

pear=scipy.stats.pearsonr(df["IMDB Score"], df["Runtime"])

In [19]:

spear=scipy.stats.spearmanr(df["IMDB Score"], df["Runtime"])

In [20]:

sns.scatterplot(x=df["IMDB Score"], y=df["Runtime"], color="green").set\_title("Correlation Between 'IMDB Score' and 'Runtime");

print(f"Correlation of Pearson: **{**pear[0]**}**")

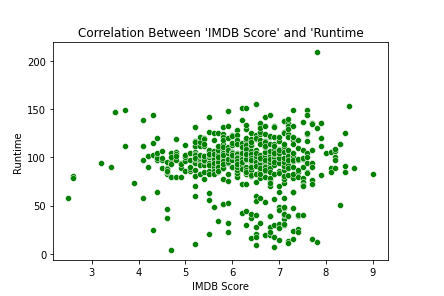
print(f"Correlation of Spearman: **{**spear[0]**}**")

print("👎There is no correlation between Runtime and IMDB Score because the correlation coefficient is too low.")

Correlation of Pearson: -0.0408962914207887

Correlation of Spearman: -0.022930090743463378

👎There is no correlation between Runtime and IMDB Score because the correlation coefficient is too low.



What are the top 10 'Genre' with the highest IMDB Score? Visualize it.

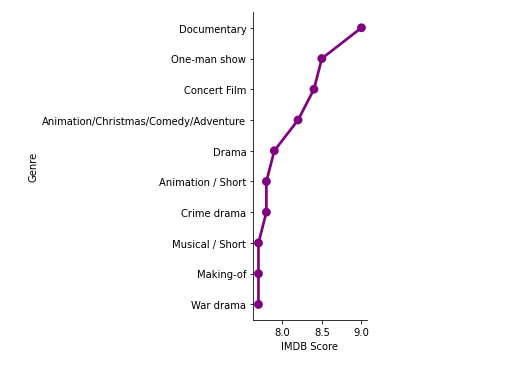
In [21]:

d9= df.groupby("Genre").agg({"IMDB Score": "max"}).sort\_values("IMDB Score", ascending=False)[

0:10].reset\_index()

In [22]:

sns.catplot(x="IMDB Score", y="Genre", data=d9, kind="point", color="purple");



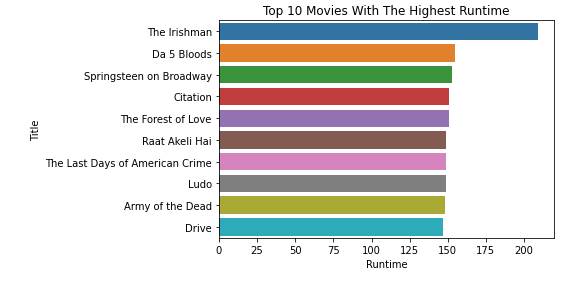
What are the top 10 movies with the highest 'runtime'? Visualize it.

In [23]:

d10=df[["Title","Runtime"]].sort\_values("Runtime", ascending=False).head(10).reset\_index(drop=True)

In [24]:

sns.barplot(y="Title", x="Runtime",errcolor="red", data=d10).set\_title("Top 10 Movies With The Highest Runtime");



In which year was the most movies released? Visualize it.

In [25]:

df['year'] = pd.DatetimeIndex(df.Premiere).year.astype(int)

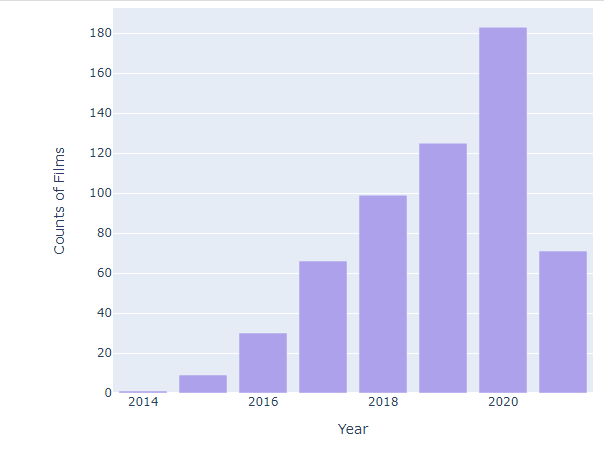
year = df['year'].value\_counts()

In [26]:

fig = px.bar(df, x=year.index, y=year.values, labels={"y":"Counts of Films", "x":"Year"})

fig.update\_traces(marker\_color='#aea1eb')

fig.show()



2014201620182020020406080100120140160180

YearCounts of Films

which language that movies have the lowest average IMDB rating

In [27]:

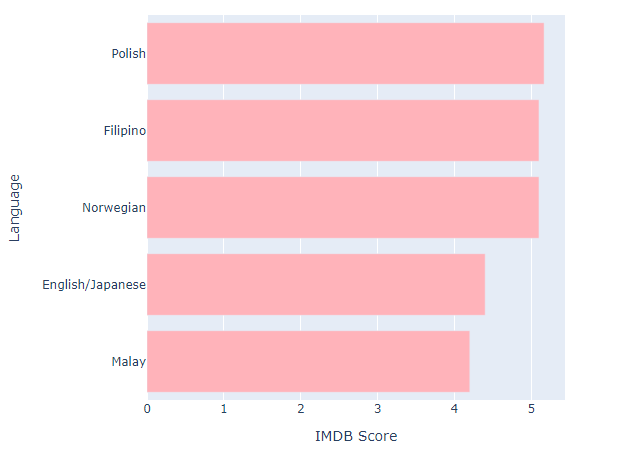
d12=df.groupby(["Language"]).agg({"IMDB Score":np.mean}).reset\_index().sort\_values("IMDB Score").head()

In [28]:

fig = px.bar(d12, y=d12["Language"], x=d12["IMDB Score"], labels={"y":"Language", "x":"IMDB Score"})

fig.update\_traces(marker\_color='#ffb3ba')

fig.show()



runtime

In [29]:

df[["year", "Runtime"]].groupby(["year"]).sum().sort\_values(["Runtime"], ascending=False).head(1)

Out[29]:

|  | Runtime |
| --- | --- |
| year |  |
| 2020 | 17384 |

What is the "Genre" and language

In [30]:

d14=(df.groupby(["Language", "Genre"]).size()

.sort\_values(ascending=False)

.reset\_index(name='Count')

.drop\_duplicates(subset='Language'))

d14.head(10)

Out[30]:

|  | Language | Genre | Count |
| --- | --- | --- | --- |
| 0 | English | Documentary | 120 |
| 5 | Hindi | Drama | 13 |
| 6 | Spanish | Drama | 8 |
| 10 | Portuguese | Comedy | 6 |
| 13 | French | Documentary | 6 |
| 15 | English/Spanish | Documentary | 5 |
| 23 | Italian | Drama | 4 |
| 31 | Indonesian | Drama | 3 |
| 39 | Korean | Drama | 2 |
| 45 | Japanese | Anime/Science fiction | 2 |

outlier data

In [31]:

def find\_outlier(data):

sns.boxplot(x=data);

q1, q3= np.percentile(data, [25,75])

iqr=q3-q1

lower\_bound = q1 - (1.5 \* iqr)

upper\_bound = q3 + (1.5 \* iqr)

data\_outlier=data[(data<lower\_bound) | (data>upper\_bound)]

count\_of\_outliers=data[(data<lower\_bound) | (data>upper\_bound)].count()

print(f"Info about outlier data for **{**data.name**}**:")

print(f"**{**count\_of\_outliers**}** outlier datas.")

print("Outlier datas: ", data[(data<lower\_bound) | (data>upper\_bound)], sep="**\n**")

In [32]:

m=df["IMDB Score"]

n=df["Runtime"]

In [33]:

find\_outlier(m)

Info about outlier data for IMDB Score:

9 outlier datas.

Outlier datas:

0 2.5

1 2.6

2 2.6

3 3.2

4 3.4

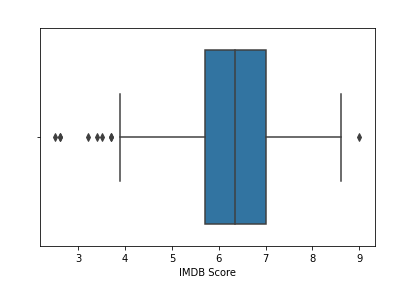
5 3.5

6 3.7

7 3.7

583 9.0

Name: IMDB Score, dtype: float64



In [34]:

find\_outlier(n)

Info about outlier data for Runtime:

75 outlier datas.

Outlier datas:

5 147

7 149

15 25

16 144

30 37

...

552 15

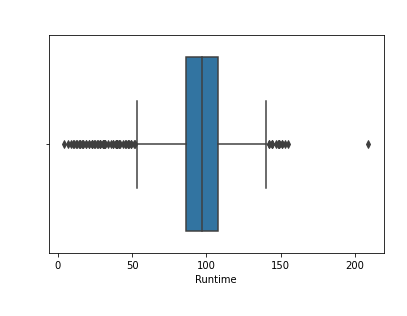
560 12

561 209

577 51

581 153

Name: Runtime, Length: 75, dtype: int64



Conclusion:

Based on the evaluation metrics, the model should provide insights into which features significantly influence the movie ratings. A well-performing model will assist in predicting IMDb scores more accurately, allowing for better decision-making in the film industry regarding what aspects might contribute to a movie's success.

Keep in mind that despite building a predictive model, predicting movie scores accurately can be challenging due to various subjective and dynamic factors that influence audience perceptions and preferences. Constant model refinement and adaptation might be necessary to improve the predictive accuracy.