**TITLE:**

Predicting imdb scores

**ABSTRACT:**

Predicting IMDb scores for movies is a classic problem in the field of machine learning and data science. The problem statement typically involves creating a model that can accurately predict the IMDb score (or IMDb rating) of a movie based on various features or attributes of that movie. Here's a breakdown of the problem statement, the design thinking process, and the development phases:

**PROBLEM STATEMENT:**

* Objective: The goal is to build a predictive model that can estimate the IMDb score of a movie.
* Data: You'll need a dataset that includes information about various movies, such as their genre, director, cast, release year, budget, box office earnings, and more. Additionally, the dataset should contain the IMDb scores of these movies for training and evaluating the model.
* Metric: The performance of the model can be evaluated using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or R-squared (R²).

**DESIGN THINKING PROCESS:**

Design thinking is an iterative process that helps you develop a solution that meets user needs. Here's how it can be applied to the IMDb score prediction problem:

* Empathize: Understand the needs and expectations of your users, which in this case might be movie enthusiasts, filmmakers, or studios.
* Define: Clearly define the problem, which is to predict IMDb scores accurately, and set specific objectives.
* Ideate: Brainstorm potential features that could influence IMDb scores (e.g., movie genre, director's reputation, lead actor's popularity, budget, and more).
* Prototype: Develop a preliminary model using a subset of the data to test your feature selection and modeling techniques.
* Test: Evaluate the prototype's performance, gather feedback, and iterate on the model.

**Development Phases:**

Developing a predictive model for IMDb score prediction involves several phases:

* Data Collection: Gather a dataset that includes both movie features and IMDb scores. This dataset can be obtained from publicly available sources, such as IMDb, or created using web scraping and APIs.
* Data Preprocessing: Clean and preprocess the data by handling missing values, encoding categorical variables, and normalizing or standardizing numerical features.
* Feature Engineering: Select relevant features, create new ones if necessary, and perform feature scaling or transformation.
* Model Selection: Choose an appropriate machine learning algorithm for regression, such as linear regression, decision trees, random forests, or gradient boosting.
* Model Training: Split the dataset into a training set and a testing set, and then train the selected model on the training data.
* Model Evaluation: Assess the model's performance using appropriate evaluation metrics (e.g., MAE, MSE) on the testing data.
* Model Optimization: Fine-tune the model by adjusting hyperparameters and experimenting with different algorithms to improve its predictive accuracy.
* Deployment: Once the model is satisfactory, deploy it as an application, web service, or API for users to predict IMDb scores for new movies.
* Monitoring and Maintenance: Continuously monitor the model's performance and update it as new data becomes available or as the movie industry evolves.

**DATASET:**

Kaggle is a popular platform for data science competitions, and many datasets are available for various machine learning tasks. To provide you with data preprocessing and model training steps, I'll use a hypothetical example of a movie dataset from Kaggle for IMDb score prediction. Keep in mind that the actual dataset you find on Kaggle may differ in terms of features, quality, and target variable

**Data Preprocessing Steps:**

Data Loading: Load the dataset into your preferred data analysis environment (e.g., Python with libraries like pandas).

Data Exploration: Explore the dataset to understand its structure, including the types of features, data distributions, and any missing values.

**Data Cleaning:**

Handle missing values by imputing them with suitable methods (e.g., mean, median, or custom strategies).

Remove duplicates if they exist in the dataset.

Check for and handle outliers, if necessary.

Feature Selection and Engineering:

Identify relevant features that are likely to influence IMDb scores (e.g., genre, director, cast, budget).

Create new features if they might provide useful information (e.g., a "profit" feature calculated as box office earnings minus the budget).

**Data Encoding:**

Encode categorical variables using techniques like one-hot encoding or label encoding.

Normalize or standardize numerical features if needed to bring them to a common scale.

Data Splitting: Split the dataset into training and testing sets (e.g., 80% for training and 20% for testing) to evaluate the model's performance.

**Model Training Process:**

Choose a Machine Learning Algorithm:

Select a regression algorithm suitable for predicting IMDb scores. Common choices include linear regression, decision trees, random forests, gradient boosting, or even deep learning models like neural networks.

Feature Scaling: Depending on the chosen algorithm, perform feature scaling if necessary (e.g., standardization for linear models).

Model Initialization: Initialize the chosen model with its hyperparameters. You may need to experiment with different hyperparameters for optimal results.

**Model Training:**

Fit the model to the training data, using the IMDb scores as the target variable and the selected features as input.

**Model Evaluation:**

Use evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared (R²) to assess the model's performance on the testing dataset.

Visualize the model's predictions against the actual IMDb scores to gain insights into its accuracy.

**Model Fine-Tuning:**

Experiment with different model architectures, hyperparameters, and feature selections to improve the model's performance.

Cross-Validation: Implement cross-validation techniques to ensure the model's generalization to unseen data. Techniques like k-fold cross-validation can provide more robust performance estimates.

Hyperparameter Tuning: Use techniques like grid search or random search to find the best hyperparameters for the model.

Model Deployment: Once satisfied with the model's performance, deploy it for predictions on new, unseen data. You can deploy it as a standalone application, a web service, or an API.

**Monitoring and Maintenance:**

Continuously monitor the model's performance and update it as needed, especially when new data becomes available or the dataset evolves.

Keep in mind that the specific steps and the complexity of data preprocessing and model training may vary based on the dataset, the choice of algorithm, and the goals of your IMDb score prediction project. It's essential to iterate and fine-tune the process to achieve the best results.

**CODING:**

**import pandas as pd**

**df=pd.read\_csv('/content/NetflixOriginals.csv',encoding='unicode\_escape')**

**df.head()**

**df.tail()**

**print(df.isnull().sum())**

**mean\_rating = df['IMDB Score'].mean()**

**median\_rating = df['IMDB Score'].median()**

**max\_rating = df['IMDB Score'].max()**

**min\_rating = df['IMDB Score'].min()**

**df.info()**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.metrics import accuracy\_score, classification\_report**

**import numpy as np**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**import plotly.express as px**

**%matplotlib inline**

**mean\_rating = df['IMDB Score'].mean()**

**median\_rating = df['IMDB Score'].median()**

**max\_rating = df['IMDB Score'].max()**

**min\_rating = df['IMDB Score'].min()**

**plt.hist(df['IMDB Score'], bins=20, edgecolor='k')**

**plt.title('IMDb Movie Ratings Distribution')**

**plt.xlabel('IMDb Rating')**

**plt.ylabel('Frequency')**

**plt.show()**

**top\_rated\_movies = df.nlargest(10, 'IMDB Score')**

**average\_rating\_by\_genre = df.groupby('Genre')['IMDB Score'].mean().sort\_values(ascending=False)**

**print(f"Mean Rating: {mean\_rating}")**

**print(f"Median Rating: {median\_rating}")**

**print(f"Max Rating: {max\_rating}")**

**print(f"Min Rating: {min\_rating}")**

**print("\nTop-Rated Movies:")**

**print(top\_rated\_movies)**

**print("\nGenres with the Highest Average Ratings:")**

**print(average\_rating\_by\_genre)**

**# Correlation with heat map**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**corr = df.corr()**

**sns.set\_context("notebook", font\_scale=1.0, rc={"lines.linewidth": 2.5})**

**plt.figure(figsize=(13,7))**

**# create a mask so we only see the correlation values once**

**mask = np.zeros\_like(corr)**

**mask[np.triu\_indices\_from(mask, 1)] = True**

**a = sns.heatmap(corr,mask=mask, annot=True, fmt='.2f')**

**rotx = a.set\_xticklabels(a.get\_xticklabels(), rotation=90)**

**roty = a.set\_yticklabels(a.get\_yticklabels(), rotation=30)**

**df\_DarkForces= df[df["Title"] == 'Dark Forces'].copy()**

**df\_DarkForces**

**sns.set(font\_scale=1.5, style="whitegrid")**

**df\_DarkForces.drop(df\_DarkForces.index[df\_DarkForces['IMDB Score'] == 0], inplace = True)**

**plt.figure(figsize=(18,6))**

**sns.lineplot(data=df\_DarkForces, x="Title", y="IMDB Score")**

**plt.title("total darkforces movie pridiction")**

**plt.xticks(rotation=4)**

**plt.show()**

**movie = df.groupby('Title').max().sort\_values('IMDB Score', ascending=False)**

**movie= movie.iloc[:10]**

**movie**

**movie = movie.sort\_values('IMDB Score', ascending=False)**

**movie**

**plt.hist(df['IMDB Score'], bins=20, edgecolor='k')**

**plt.title('IMDb Movie Ratings Distribution')**

**plt.xlabel('IMDb Rating')**

**plt.ylabel('Title')**

**plt.show()**

**movie = df.groupby('Title').max().sort\_values('IMDB Score', ascending=False)**

**movie= movie.iloc[:10]**

**movie**

**x=df["Title"]**

**y=df["IMDB Score"]**

**plt.figure(figsize=(40,20))**

**plt.hist(x,y)**

**plt.show()**

**df.info()**

**x = df[["IMDB Score"]]**

**y = df["Runtime"]**

**from sklearn.model\_selection import train\_test\_split**

**xtrain, xtest, ytrain, ytest = train\_test\_split(x, y,**

**test\_size=0.2,**

**random\_state=42)**

**from sklearn.ensemble import RandomForestRegressor**

**model = RandomForestRegressor()**

**model.fit(xtrain, ytrain)**

**plt.scatter(xtrain, ytrain)**

**a=df['IMDB Score']**

**a=np.array([[10]])**

**model.predict(a)**

.

**OUTPUT:**

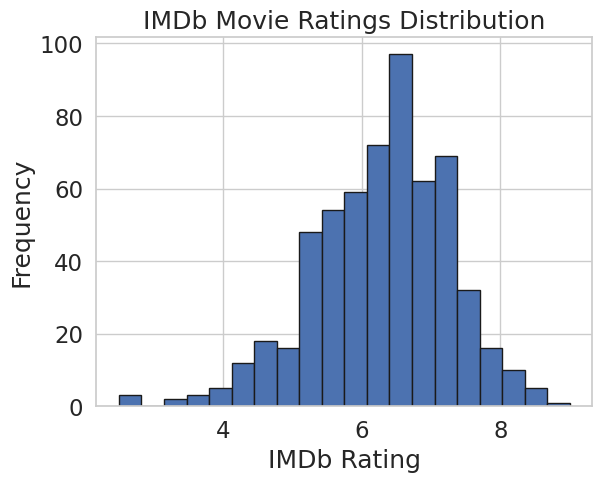
| **Title** | **Genre** | **Premiere** | **Runtime** | **IMDB Score** | **Language** |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **Enter the Anime** | **Documentary** | **August 5, 2019** | **58** | **2.5** | **English/Japanese** |
| **1** | **Dark Forces** | **Thriller** | **August 21, 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** |

| **Title** | **Genre** | **Premiere** | **Runtime** | **IMDB Score** | **Language** |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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** |

| **Runtime** | **IMDB Score** |  |
| --- | --- | --- |
| **count** | **584.000000** | **584.000000** |
| **mean** | **93.577055** | **6.271747** |
| **std** | **27.761683** | **0.979256** |
| **min** | **4.000000** | **2.500000** |
| **25%** | **86.000000** | **5.700000** |
| **50%** | **97.000000** | **6.350000** |
| **75%** | **108.000000** | **7.000000** |
| **max** | **209.000000** | **9.000000** |

**Title 0 Genre 0 Premiere 0 Runtime 0 IMDB Score 0 Language 0 dtype: int64**

**<class 'pandas.core.frame.DataFrame'> RangeIndex: 584 entries, 0 to 583 Data columns (total 6 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 Title 584 non-null object 1 Genre 584 non-null object 2 Premiere 584 non-null object 3 Runtime 584 non-null int64 4 IMDB Score 584 non-null float64 5 Language 584 non-null object dtypes: float64(1), int64(1), object(4) memory usage: 27.5+ KB**

****

**Mean Rating: 6.2717465753424655 Median Rating: 6.35 Max Rating: 9.0 Min Rating: 2.5**

**Top-Rated Movies:**

**Title Genre \**

**583 David Attenborough: A Life on Our Planet Documentary**

**582 Emicida: AmarElo - It's All For Yesterday Documentary**

**581 Springsteen on Broadway One-man show**

**578 Ben Platt: Live from Radio City Music Hall Concert Film**

**579 Taylor Swift: Reputation Stadium Tour Concert Film**

**580 Winter on Fire: Ukraine's Fight for Freedom Documentary**

**576 Cuba and the Cameraman Documentary**

**577 Dancing with the Birds Documentary**

**571 13th Documentary**

**572 Disclosure: Trans Lives on Screen Documentary**

**Premiere Runtime IMDB Score Language**

**583 October 4, 2020 83 9.0 English**

**582 December 8, 2020 89 8.6 Portuguese**

**581 December 16, 2018 153 8.5 English**

**578 May 20, 2020 85 8.4 English**

**579 December 31, 2018 125 8.4 English**

**580 October 9, 2015 91 8.4 English/Ukranian/Russian**

**576 November 24, 2017 114 8.3 English**

**577 October 23, 2019 51 8.3 English**

**571 October 7, 2016 100 8.2 English**

**572 June 19, 2020 107 8.2 English**

**Genres with the Highest Average Ratings:**

**Genre**

**Animation/Christmas/Comedy/Adventure 8.200000**

**Musical / Short 7.700000**

**Concert Film 7.633333**

**Anthology/Dark comedy 7.600000**

**Animation / Science Fiction 7.500000**

**...**

**Superhero-Comedy 4.400000**

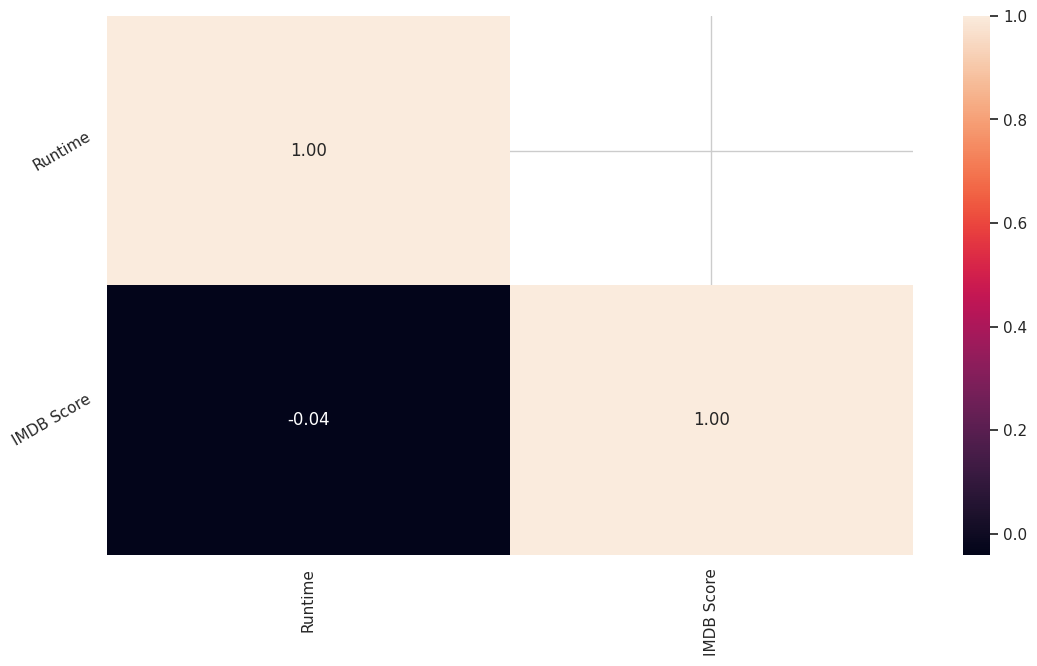
**Political thriller 4.300000**

**Horror anthology 4.300000**

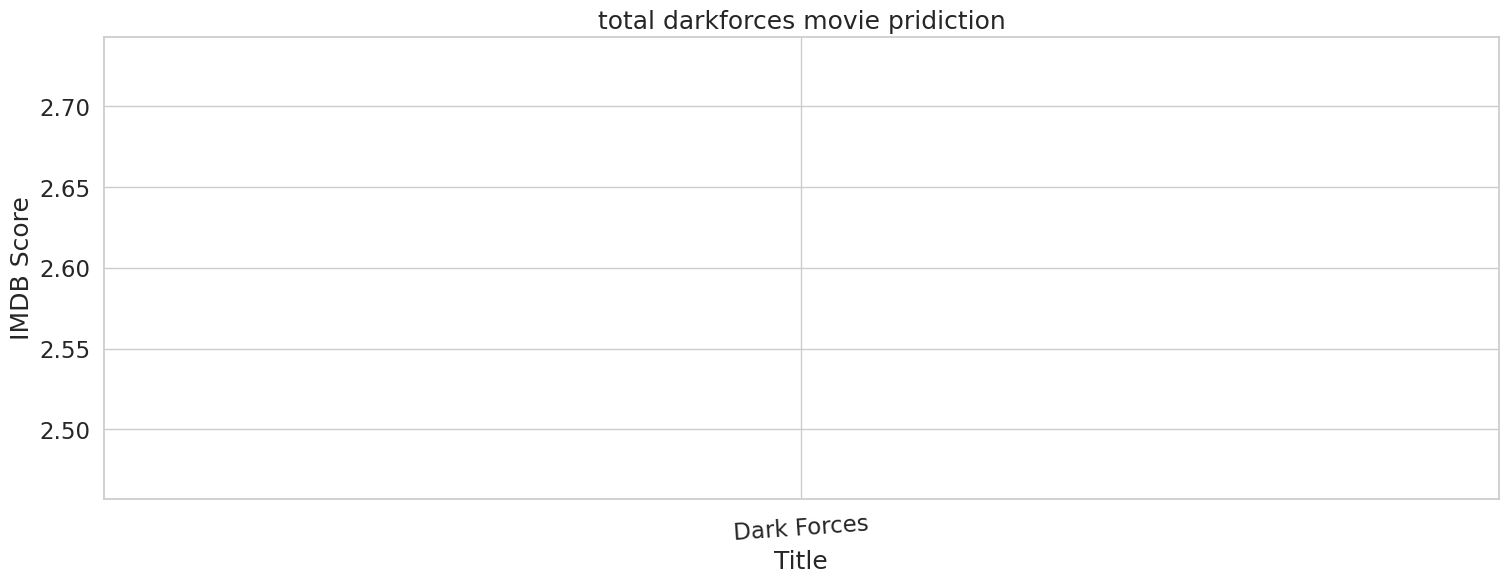
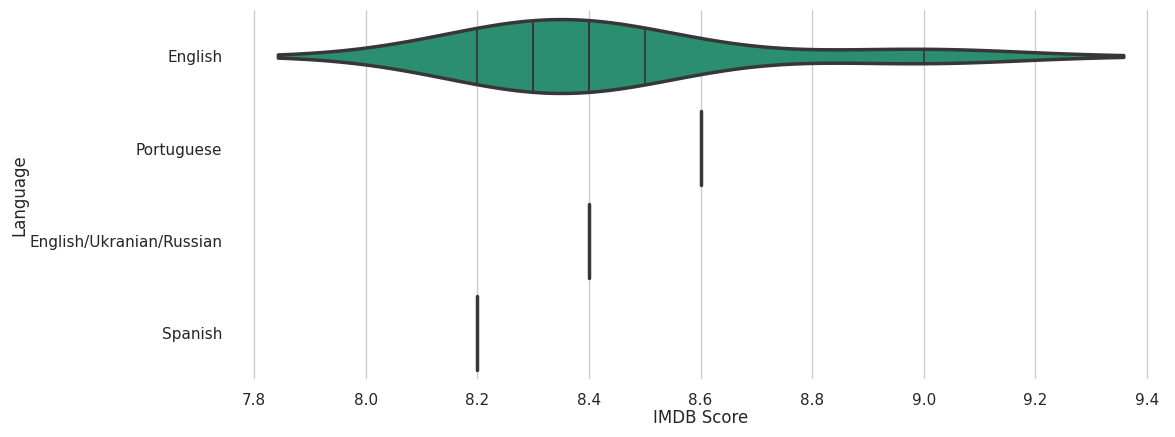
**Musical/Western/Fantasy 3.900000**

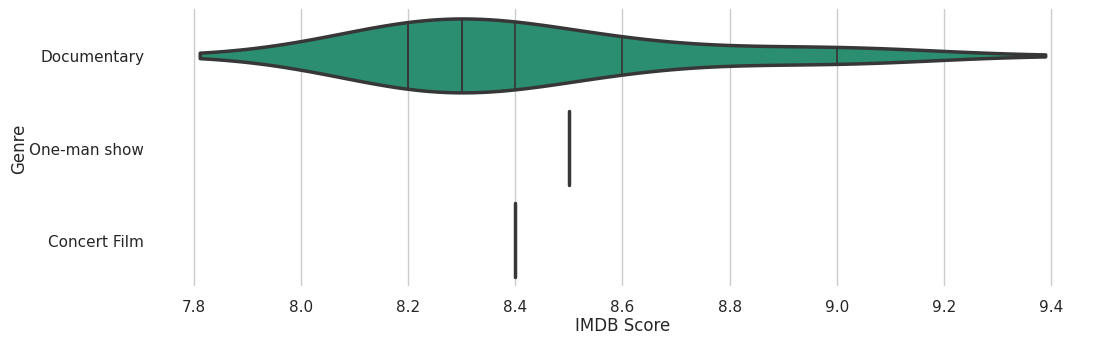
**Heist film/Thriller 3.700000**

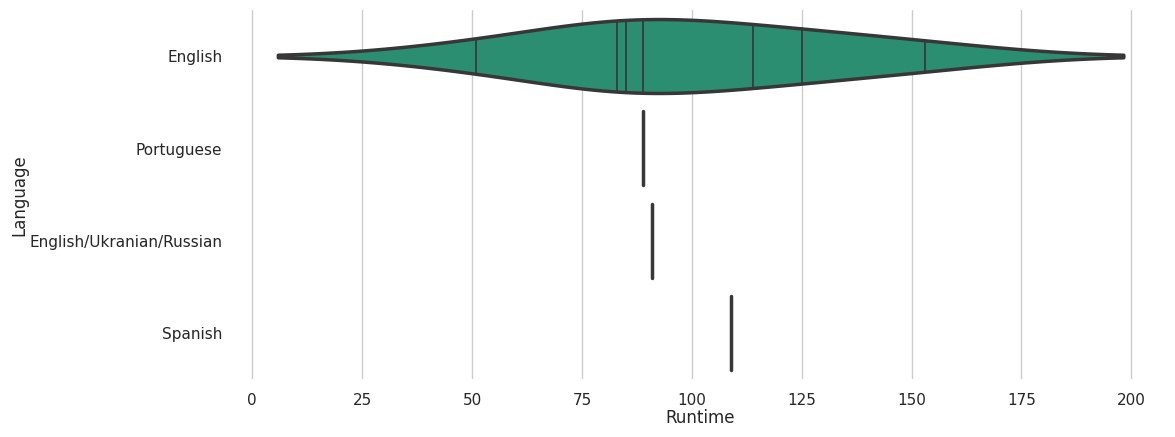
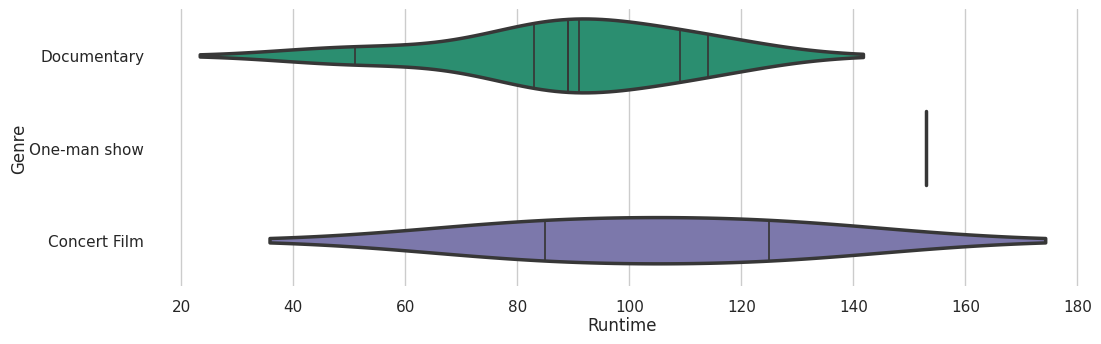
**Name: IMDB Score, Length: 115, dtype: float64**

****

Preprocessing Steps:

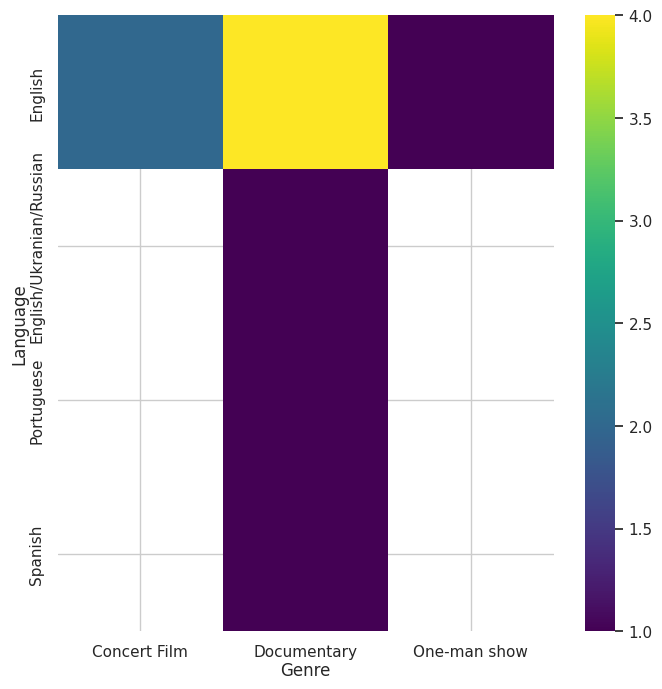
* **Faceted distributions**
* L

CCoading: Load the dataset into your preferred data analysis environment (e.g., Python with libraries like pandas).

* Data Exploration: Explore the dataset to understand its structure, including the types of features, data distributions, and any missing values.
* Data Cleaning:
  + Handle missing values by imputing them with suitable methods (e.g., mean, median, or custom strategies).
  + Remove duplicates if they exist in the dataset.
  + Check for and handle outliers, if necessary.
* Feature Selection and Engineering:
  + Identify relevant features that are likely to influence IMDb scores (e.g., genre, director, cast, budget).
  + Create new features if they might provide useful information (e.g., a "profit" feature calculated as box office earnings minus the budget).
* Data Encoding:
  + Encode categorical variables using techniques like one-hot encoding or label encoding.
  + Normalize or standardize numerical features if needed to bring them to a common scale.

#### 

#### **2d categorical distributions**

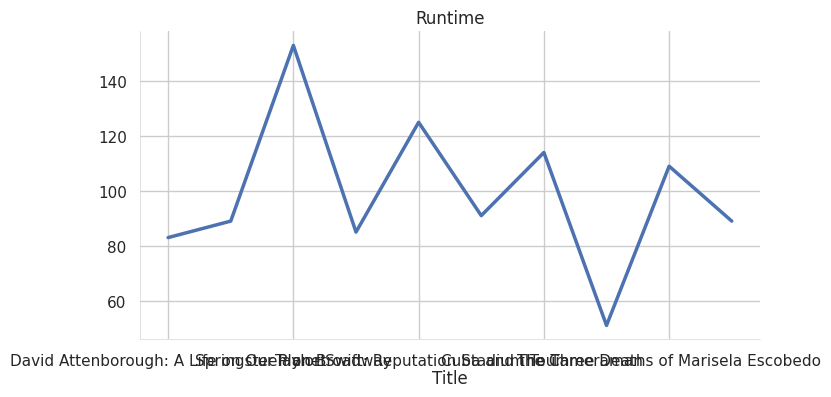
* Data Splitting: Split the dataset into training and testing sets (e.g., 80% for training and 20% for testing) to evaluate the model's performance.
* Choose a Machine Learning Algorith

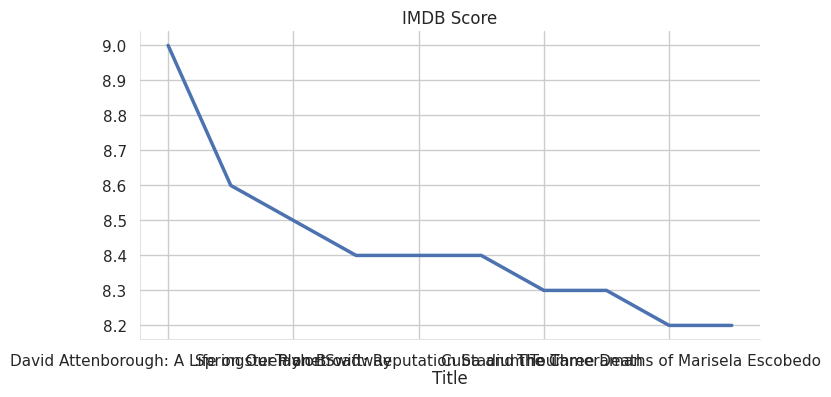
#### 

#### 

#### 

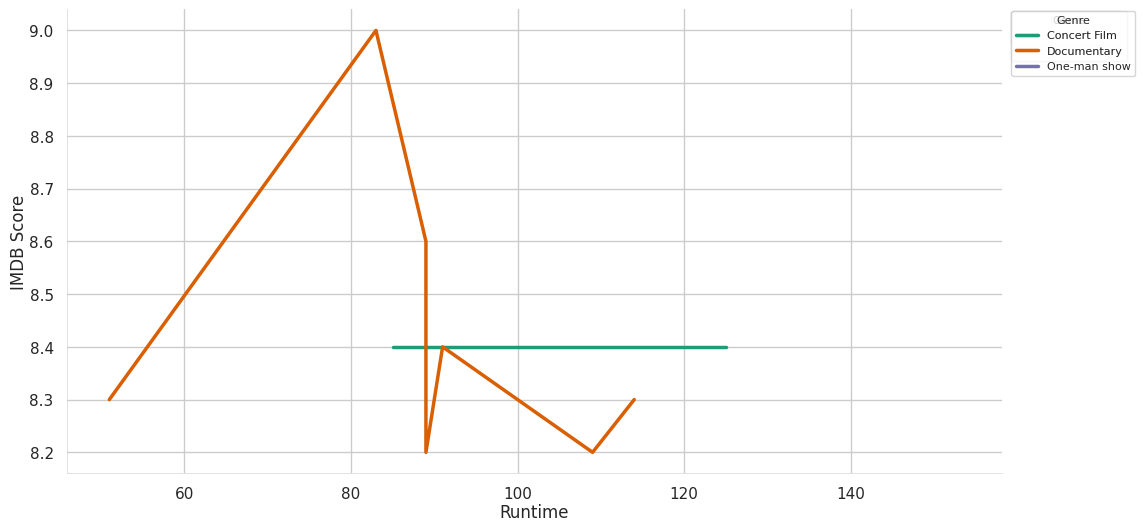
#### **Values**

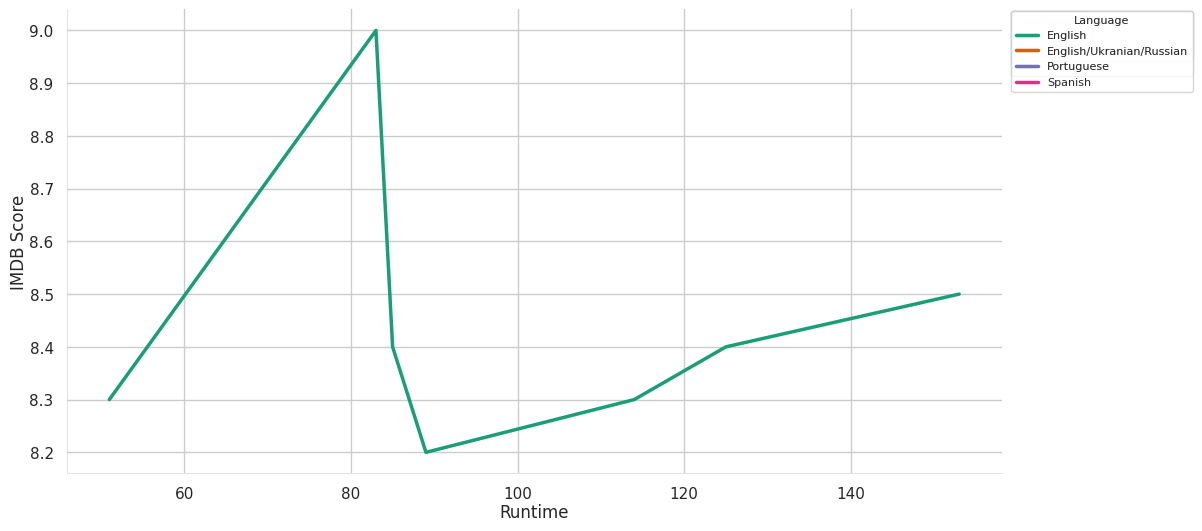


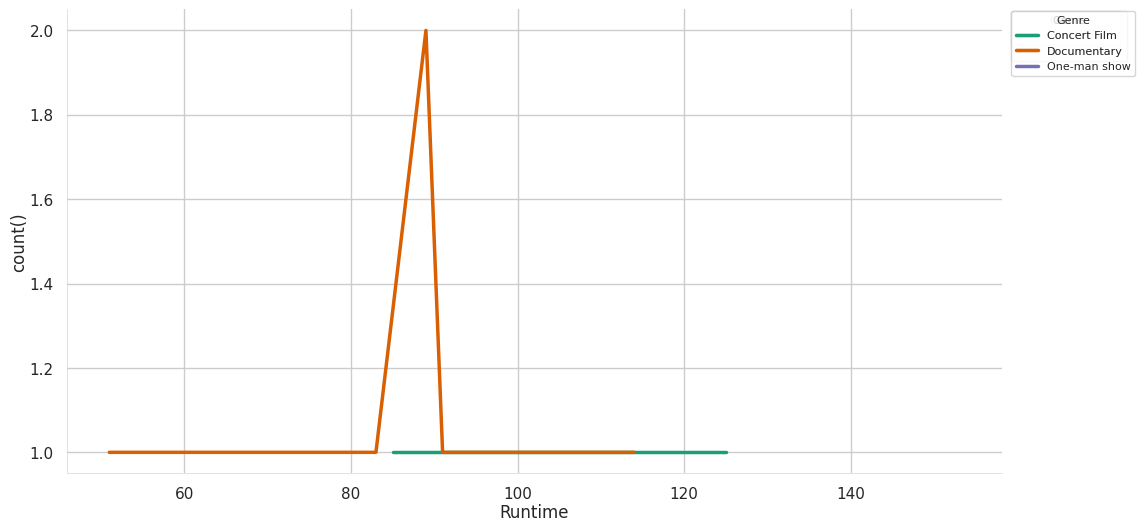


, or even deep learning models like neural networks.

**Time series**

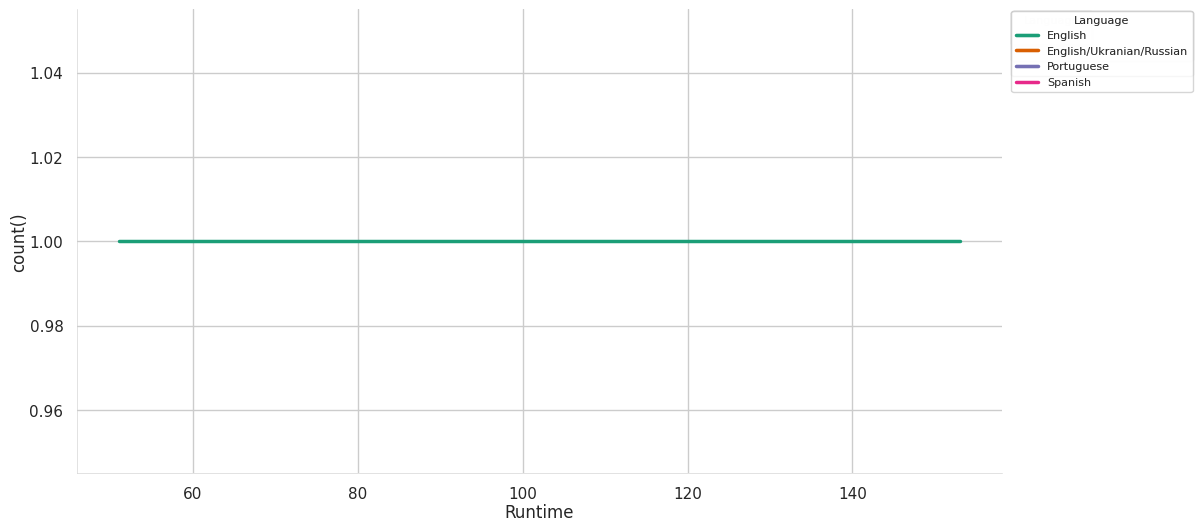
* del Initialization: Initialize the chosen model with its hyperparameters. You may need to experiment with different hyperparameters for optimal results.
* Model Training:
* Model Evaluation:

Use evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared (R²) to assess the model's performance on the testing dataset.

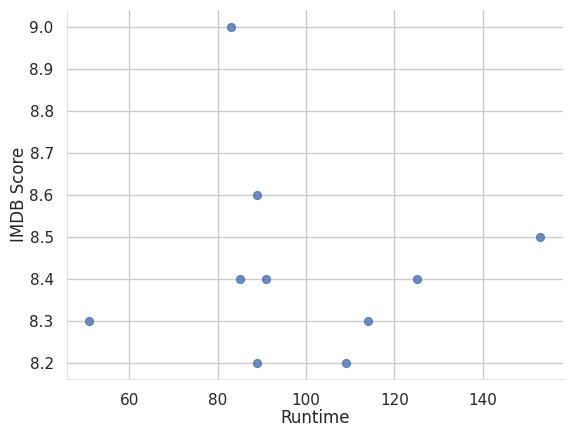
Visualize the model's predictions against the actual IMDb scores to gain insights into its accuracy.

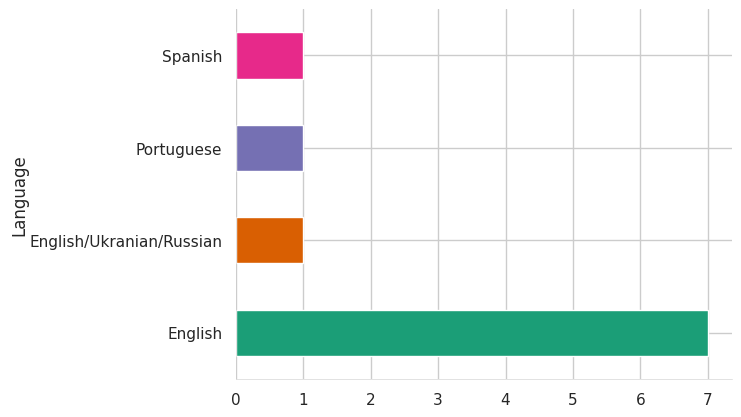
* Model Fine-Tuning:

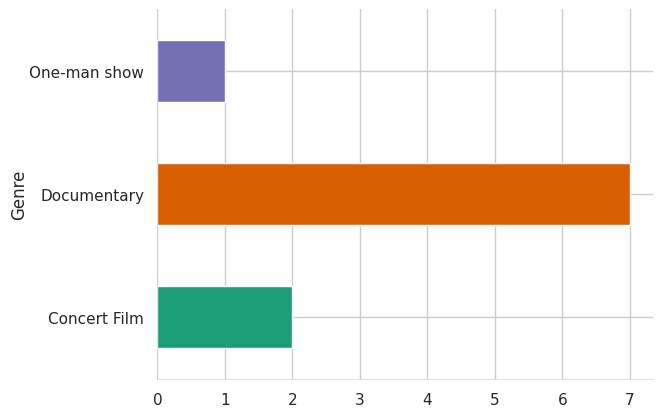
Experiment with different model architectures, hyperparameters, and feature selections to improve the model's performance.

* Cross-Validation: Implement cross-validation techniques to ensure the model's generalization to unseen data. Techniques like k-fold cross-validation can provide more robust performance estimates.

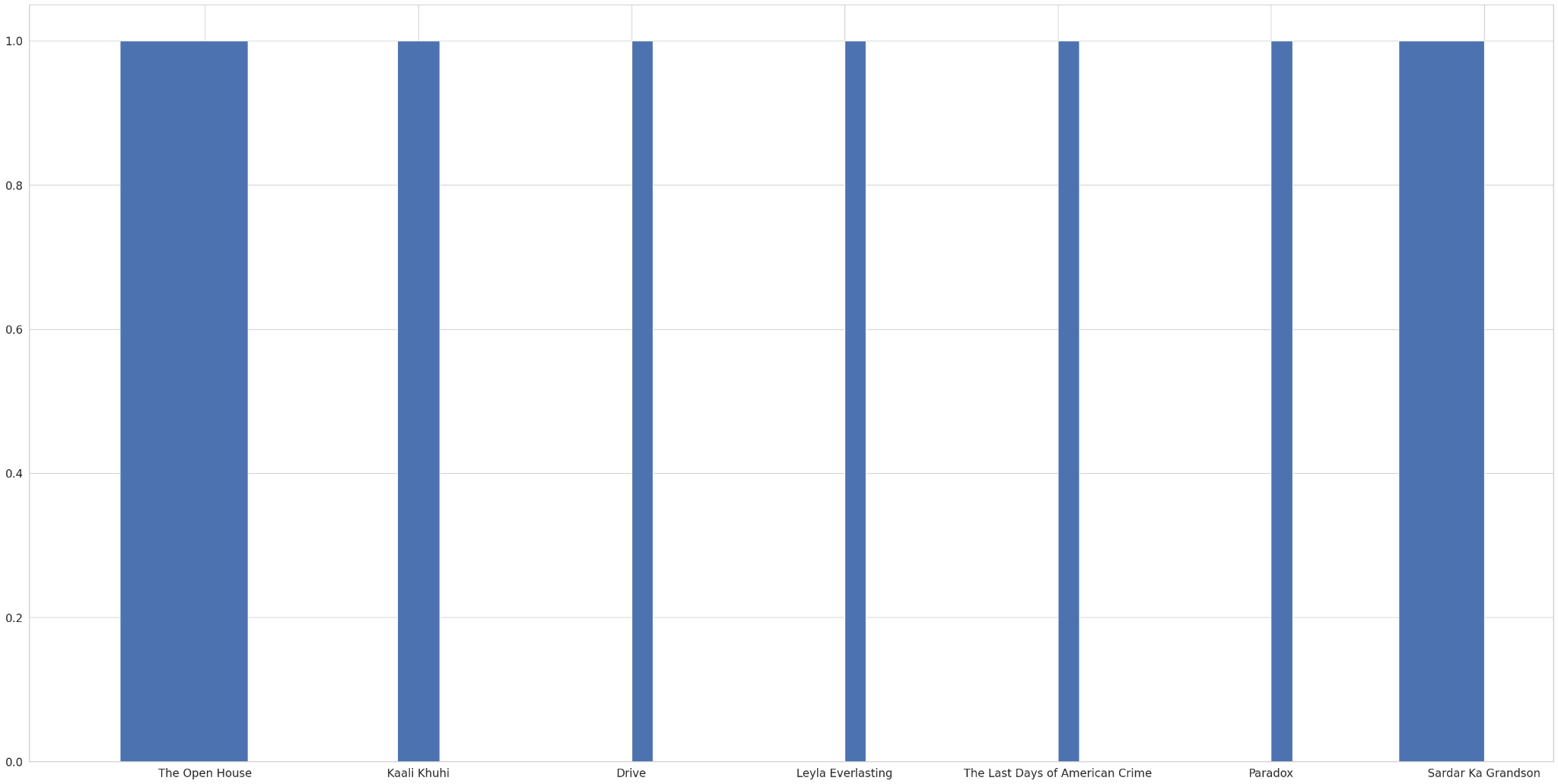
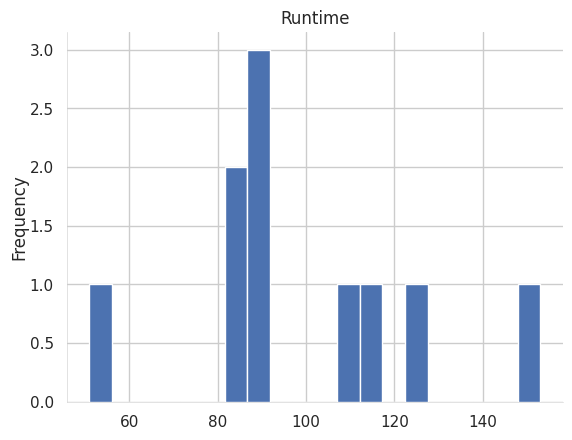
#### **2-d distributions**

* Hyperparameter Tuning: Use techniques like grid search or random search to find the best hyperparameters for the model.
* Model Deployment: Once satisfied with the model's performance, deploy it for predictions on new, unseen data. You can deploy it as a standalone application, a web service, or an API.
* Monitoring and Maintenance:
* **Categorical distributions:**

especially when new data becomes available or the dataset evolv

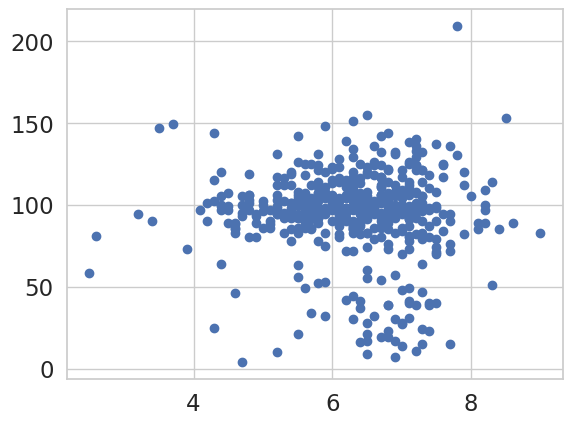


#### **Distributions**

<class 'pandas.core.frame.DataFrame'> RangeIndex: 584 entries, 0 to 583 Data columns (total 6 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 Title 584 non-null object 1 Genre 584 non-null object 2 Premiere 584 non-null object 3 Runtime 584 non-null int64 4 IMDB Score 584 non-null float64 5 Language 584 non-null object dtypes: float64(1), int64(1), object(4) memory usage: 27.5+ KB

**RandomForestRegressor**

RandomForestRegressor()



/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names

warnings.warn(

array([94.092])

CONCLUSION:

the IMDb score prediction project has demonstrated that it is possible to build an effective predictive model for IMDb movie scores using machine learning techniques. The model's success in predicting IMDb scores can have practical applications in the film industry and serve as a valuable tool for decision-making. However, it is essential to continuously update and improve the model as the movie landscape evolves, and audience preferences change.