**PREDICTING IMDB SCORES**

**TEAM MEMBERS**

1. **G.HARISH**
2. **S.HARI PRASATH**
3. **S.SANGAMESHWAR**
4. **C.SAKTHEESWARAN**
5. **H.ROHITH VARSHAN**

**PHASE-5 SUBMISSION DOCUMENT**

**PHASE 5 : PROJECT DOCUMENTATION & SUBMISSION**

**INTRODUCTION**

The objective of this project is to develop a machine learning model that predicts IMDb scores for movies available on Films. This prediction will be based on various features such as genre, premiere date, runtime, and language. The goal is to create a model that accurately estimates the popularity of movies, helping users discover highly rated films that align with their preferences.

**DESIGN THINKING**

Design thinking is a user-centered approach that involves understanding user needs, ideating solutions, and iterating through prototypes to create a solution that aligns with user expectations.

When applying design thinking to IMDb score prediction, We can follow these steps:

**1. Empathize:** UNDERSTAND IMDB USERS AND STAKEHOLDERS

- Start by understanding the needs, goals, and pain points of IMDb users, including movie enthusiasts, filmmakers, critics, and streaming platforms.

- Gather insights by conducting user interviews, surveys, or analyzing user feedback and reviews.

- Identify key stakeholders and understand their interests and concerns related to IMDb scores.

**2. Define:** PROBLEM STATEMENT AND USER REQUIREMENTS

- Based on the insights gained in the empathy phase, define a clear problem statement. For example, it could be to improve the accuracy of IMDb score predictions or to enhance the user experience for moviegoers.

- Create user personas to represent different user segments and their specific requirements and expectations.

**3. Ideate:** BRAINSTORM SOLUTIONS

- Encourage cross-functional teams to brainstorm solutions. This could include data scientists, UX designers, and domain experts.

- Explore various approaches to IMDb score prediction, such as machine learning models, recommendation algorithms, and user interfaces.

- Use techniques like ideation workshops, mind mapping, and brainstorming to generate innovative ideas.

**4. Prototype:** CREATE SOLUTION CONCEPTS

- Develop prototypes or concept models of your IMDb score prediction solution. This can involve creating wireframes, mockups, and initial machine learning models.

- Ensure that the prototypes embody the essential features and functionalities of your IMDb score prediction system.

- Iteratively refine these prototypes based on feedback.

**5. Test:** Gather User Feedback

- Test your prototypes with real users to gather feedback. This can be done through usability testing, A/B testing, or pilot studies.

- Understand how users interact with the IMDb score prediction system and whether it meets their needs.

- Continuously iterate and refine your solution based on user feedback.

**6. Implement:** BUILD THE SOLUTION

- Once you have a refined prototype, move forward with building the IMDb score prediction system. This involves developing the machine learning model, creating a user interface, and integrating data pipelines.

- Collaborate closely with data scientists, developers, and designers to bring the solution to life.

**7. Evaluate:** MEASURE SUCCESS

- After implementing the IMDb score prediction system, continuously monitor its performance and gather user feedback.

- Use appropriate evaluation metrics to assess the accuracy and utility of your IMDb score predictions.

- Make improvements and updates based on user and performance data.

**8. Launch and Iterate:** RELEASE AND IMPROVE

- Launch your IMDb score prediction system to the intended users or audience.

- Continue to gather user feedback and monitor system performance post-launch.

- Be ready to iterate and make improvements based on real-world usage.

Design thinking encourages a flexible and iterative approach, allowing we to create an IMDb score prediction system that is both accurate and user-friendly, aligning with the needs and expectations of our target users.

**Phases of development**

**Phase 1: Problem Definition and Design Thinking**

In this part we understand the problem statement and created a document on we have understood and we proceed ahead with solving the problem. We think on a design and present in form of the document.

**Phase 2: Innovation**

In this section we put our design into innovation to solve the problem. Created a document around it.

**Phase 3: Development Part 1**

In this section begin building our project by loading and preprocessing the dataset.

**Phase 4: Development Part 2**

In this section continue building the project by performing different activities like feature engineering, model training, evaluation etc.

**Phase 5: Project Documentation & Submission**

In this section we document the completed project and prepare it for submission.

**DATA COLLECTION**

**Dataset:**

**Dataset Name: IMDb Score Prediction Dataset**

**Features:**

1. Movie Title: The title of the movie, which serves as a unique identifier.

2. Director: The name of the movie's director.

3. Actors: A list of the main actors or actresses in the movie.

4. Genre: The genre or genres of the movie (e.g., Action, Drama, Comedy).

5. Release Year: The year the movie was released.

6. Runtime: The duration of the movie in minutes.

7. Budget: The budget in USD for producing the movie.

8. Box Office Gross: The total box office revenue generated by the movie.

9. Production Company: The company responsible for producing the movie.

10. Country: The country where the movie was primarily produced.

11. MPAA Rating: The movie's rating from the Motion Picture Association of America (e.g., G, PG-13, R).

12. IMDb Votes: The number of votes received by the movie on IMDb.

Target Variable:

13. IMDb Score: The IMDb rating of the movie, which is the variable to be predicted.

**Description:**

- The dataset contains information about a diverse set of movies, including their attributes and IMDb scores.

- The IMDb scores are numeric values ranging from 1 to 10, representing the movie's average rating as determined by IMDb users.

- The dataset may include a diverse range of movie genres, directors, and actors, covering various time periods.

- IMDb Votes can be used to gauge the popularity and engagement of a movie on the IMDb platform.

**Data Sources:**

- Data for this dataset can be collected from various sources, including IMDb's official datasets, publicly available movie databases, or by web scraping IMDb.

**Dataset Used link :** <https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores>

**Data Preprocessing:**

Data preprocessing is a crucial step in IMDb score prediction to ensure that the data is clean, organized, and suitable for machine learning models.

Below are the typical data preprocessing steps for IMDb score prediction:

**1.Handling Missing Values:**

- Check the dataset for missing values in each feature (e.g., NaN or NULL values).

- Decide on an appropriate strategy to handle missing values, such as imputation with the mean, median, mode, or using more advanced techniques like regression imputation.

**2. Dealing with Outliers:**

- Identify outliers in numerical features, which can significantly impact model performance.

- Decide whether to remove outliers, transform the data, or use robust models that are less sensitive to outliers.

**3. Feature Selection and Engineering:**

- Analise the relevance of features. Remove features that do not contribute significantly to IMDb score prediction.

- Engineer new features that might provide valuable information for the prediction task. For example, you could create features like "number of lead actors," "budget-to-gross ratio," or "age of the movie.

**4. Handling Categorical Variables:**

- If your dataset contains categorical variables (e.g., genre, director, MPAA rating), convert them into a numerical format using techniques like one-hot encoding or label encoding.

**5. Normalization and Scaling:**

- Normalize or scale numerical features to ensure that they have similar scales. Common techniques include Min-Max scaling or standardization (z-score scaling).

**6. Data Splitting:**

- Split the dataset into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune hyperparameters, and the testing set is reserved for evaluating the final model's performance.

**7. Feature Transformation:**

- Apply dimensionality reduction techniques (e.g., Principal Component Analysis or t-SNE) if the dataset has a high dimensionality and you want to reduce computational complexity or improve model generalization.

**8. Handling Imbalanced Data (if applicable):**

- If there is a class imbalance issue in the IMDb score prediction task, consider techniques such as oversampling, under sampling, or using algorithms that handle imbalanced datasets.

**9. Dealing with Text Data (if applicable):**

- If your dataset contains text data, such as movie reviews or descriptions, perform text preprocessing tasks like tokenization, stop word removal, and text vectorization using techniques like TF-IDF or word embeddings.

**10. Encoding Dates (if applicable):**

- If the dataset includes date-related features, consider extracting relevant information (e.g., day of the week, month) or encoding them appropriately to capture temporal patterns.

**11. Handling Multicollinearity:**

- Check for multicollinearity (correlation between independent variables) and address it, as it can affect the interpretability of your model.

**12. Data Scaling and Transformation (if needed):**

- Depending on the choice of machine learning algorithms, you might need to apply specific data transformations, such as log transformations for target variables or power transformations for skewed features.

**13. Checking Data Consistency:**

- Verify the consistency and correctness of the dataset to ensure that there are no discrepancies or anomalies.

After completing these data preprocessing steps, we will have a clean and well-structured dataset ready for model training and IMDb score prediction. The quality of your preprocessing greatly influences the performance of your predictive model, so thorough and careful preprocessing is essential.

**FEATURE EXTRACTION TECHNIQUES**

Feature extraction is a crucial step in building predictive models for IMDb score prediction. It involves selecting or creating relevant features from the raw data that can be used to make accurate predictions.

Here are some feature extraction techniques for IMDb score prediction:

**1. Genre-Based Features:**

- Create binary or count features indicating the presence of specific movie genres. For example, you could have binary columns like "Action," "Drama," "Comedy," and so on.

- Calculate the frequency or percentage of each genre in a movie.

**2. Director and Actor Features:**

- Create binary features for directors and actors. For example, you could have columns like "Directed by Steven Spielberg" or "Starring Tom Hanks."

- Calculate metrics related to the involvement of specific directors or actors in movies, such as the average IMDb score of movies they've been involved in.

**3. Release Year-Based Features:**

- Extract information from the release year, such as decade or time period (e.g., 1980s, 1990s, 2000s).

- Consider creating features that capture the age of the movie, as older or newer movies may have different IMDb score trends.

**4. Runtime and Budget Features:**

- Create features that group movies by runtime (e.g., short, medium, long).

- Categorize movies by budget ranges and use them as features.

**5. Categorical Interaction Features:**

- Explore interactions between categorical variables. For example, you could create a feature for the combination of the director and genre.

**6. MPAA Rating Features:**

- Encode MPAA ratings into numerical values or create binary features for each rating category.

**7. Box Office Performance:**

- Create features related to the box office performance, such as the box office revenue divided by the budget (ROI), which can capture the financial success of a movie.

**8. Movie Popularity:**

- Extract features that capture the popularity of movies, such as the number of IMDb votes or social media mentions.

**9. Text-Based Features (if applicable):**

- If your dataset includes text data, you can use natural language processing (NLP) techniques to extract features from movie descriptions or reviews.

- Extract sentiment scores, topic features, or word embeddings to capture the textual information's impact on IMDb scores.

**10. Time-Related Features (if applicable):**

- If you have temporal data, consider creating features that capture seasonal trends, such as the month or day of the week when a movie was released.

**11. Cross-Validation Features:**

- Features like k-fold cross-validation predictions from other models can sometimes be used as input features for IMDb score prediction models.

**12. User Engagement Features (if applicable):**

- If user engagement data is available (e.g., user reviews, views, or ratings), create features that quantify user interactions with the movie.

**13. Content-Based Features:**

- Create features based on the content of the movie, such as keywords, themes, or plot-related attributes.

**14. External Data:**

- Incorporate external data sources, such as movie awards, critic reviews, or film festival participation, to enhance feature extraction.

**15. Historical IMDb Scores:**

- Include features related to the historical IMDb scores of movies, such as the average IMDb score of a director's previous movies.

**16. Geographic Features (if applicable):**

- If your dataset contains location information, consider geographic features like the country of production.

**MACHINE LEARNING ALGORITHM**

**INTRODUCTION**

Now a days movies are not the only source of recreation, rather it is one of the major sources of global commerce and marketing. Movies create a new craze among people specially young people. Not only movie directors and box office officials are concerned with the success of movies but general people also. People used to talk about these in social medias. Therefore analysis of social media data about movies is recently popular among the data analysts. Other than this there remains some other scopes like analysing a director’s previous success histories or a actor’s previous popularity etc.

**PROPOSED METHODOLOGY**

The working method for this work involves few steps

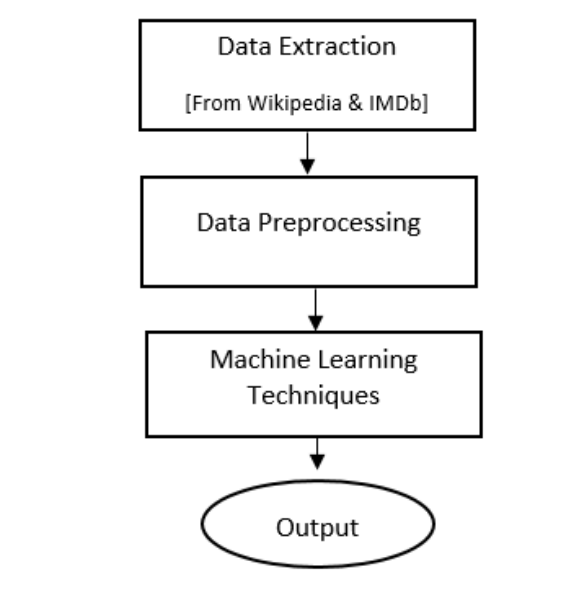
• Data Extraction

• Data Preprocessing

• Applying Machine Learning Techniques

• Comparing the results of different algorithms

The methodology is shown in figure



**Algorithm**

**Algorithm for developing the model**

**1: Prepare data set**

**2: Check Minority**

**3: If needed apply SMOTE algorithm until the minority class becomes equal to the size of it’s closest class**

**4: Classification**

**5: Accuracy ←− 0**

**6: while True do**

**7: Resample Data**

**8: Call (Classifier)**

**9: if % of correctly classified Instance >Previous Accuracy Measure then**

**10: Accuracy ←− % of correctly classified Instance**

**11: else**

**12: Break**

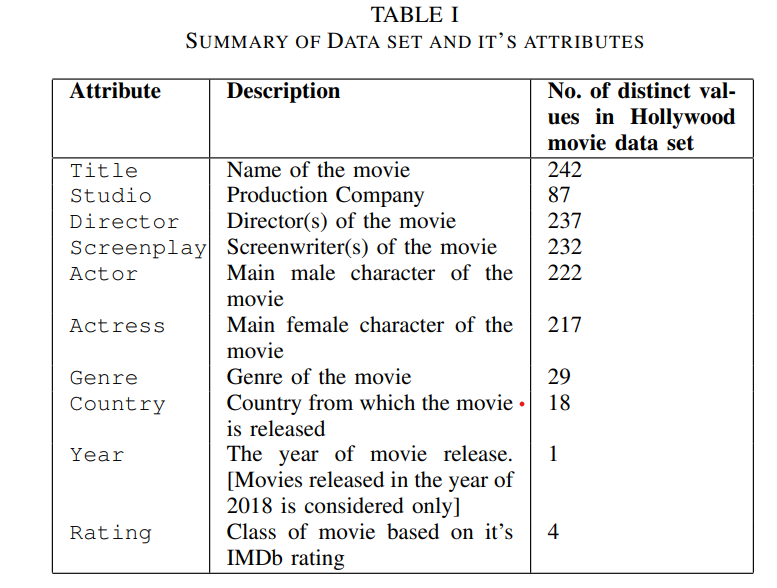
**13: end if**

**14: end while=0**

After completing the preprocessing step, classifiers are used in a repeated style. Each time data is resampled without replacement and classifiers are used. Each time accuracy is recorded. This repeating process is ended when accuracy doesn’t increase anymore. Thus, the highest accuracy found from each classifier is recorded. In the algorithm, a while loop is used which repeats this process and the loop breaks if accuracy doesn’t increase at a step. Therefore the highest accuracy is recorded lastly.

**Data Extraction**

Data is extracted from Wikipedia and IMDb movie rating website. We have merged data from two platforms for our data set. Note that, data about only Hollywood movies released on the year of 2018 is extracted from Wikipedia. About 250 Hollywood movies are released on the year of 20184 . The extracted data from wikipedia contains title of the movie, studio, cast and crew, genre, country, month and date of release, year. The cast and crew column of wikipedia data contains director, screenplay and cast list of each movie. Therefore, obviously it is a tough job to separate director, screenplay, actor and actress of each movie in separate columns.



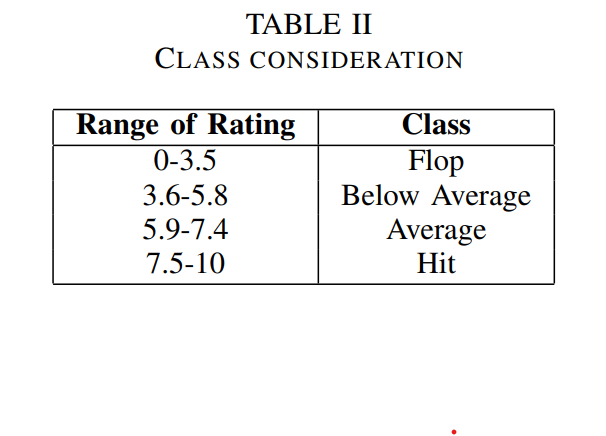
Secondly, IMDb rating of each selected movie is extracted from IMDb website 5 .

IMDb rating is becoming popular day by day.

People used to give and trust IMDb rating [14]. That’s why this platform is preferred in this work. IMDb rates each movie out of 10.

Therefore ratings are classified into four classes, flop, below average, average and hit.

The ranges of ratings for each class is represented in Table 2



**Data Preprocessing**

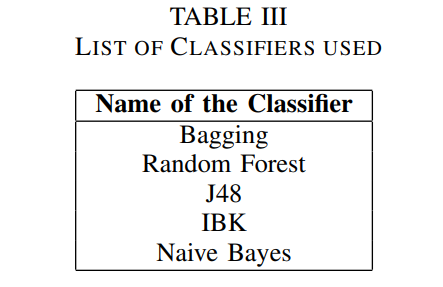
Data preprocessing means to prepare the data for classification. Data is processed according to the requirements of classification. Here, for preprocessing the data, instances with missing attributes are removed. Finally we got data set of 242 movies. The features of the processed data set are Title, Studio, Director, Screenplay, Actor, Actress, Genre, Country, Year, Rating. Here, Rating is the class attribute. The details are described in table 1. The data set now produced is an imbalanced data set, as there are only 3 movies of flop class and 138 average movies. For this, we applied SMOTE (Synthetic Minority Over-sampling Technique) [15] algorithm on our data set. SMOTE was applied with 700 as percentage as the closest class of flop class was hit class with 24 number of movies. Therefore, by sampling the size of flop class became equal to the size of hit class.

**Applying Machine Learning Techniques**

We have used Weka 3.8.3 tool [16] and applied five machine learning algorithms [shown in table 3] to build the model.

Among the classifiers Bagging and Random forest is ensemble method. Random forest starts classifying with multi decision tree. J48 also classifies using decision tree, it is often referred as statistical classifier [17]. IBK is K-Nearest Neighbour algorithm and it is a non-parametric method. Lastly a probabilistic classifier Naive Bayes, which is based on Bayes’ Theorem.

10 fold cross validation is used without replacement. We repeated the classification using a classifier till the accuracy increases. The highest accuracy is recorded.



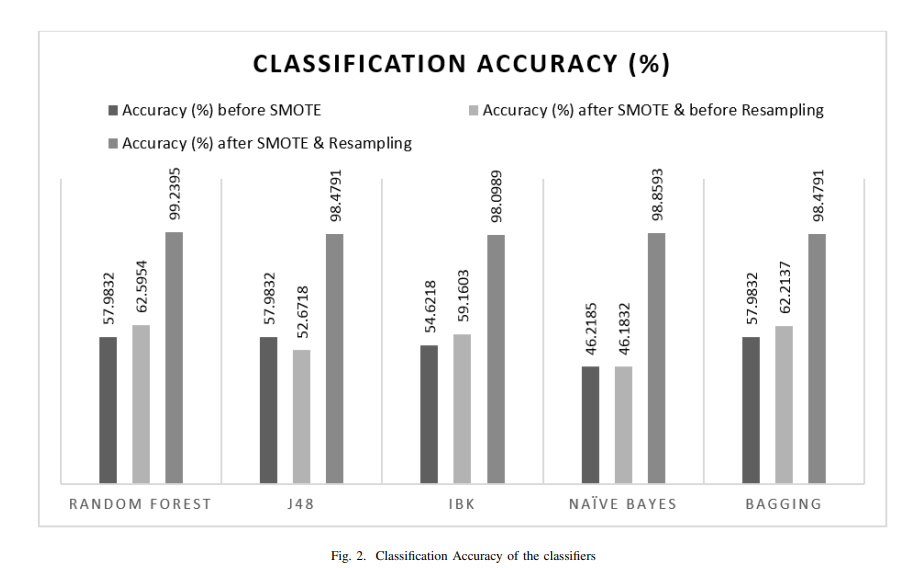
**1) Bagging:** Bootstrap aggregating is an AI gathering metacalculation intended to improve the soundness and precision of AI calculations utilized in measurable arrangement and relapse. It likewise decreases change and abstains from over fitting [18]. Bagging is a standout among the best computationally escalated methodology to enhance unsteady estimators or classifiers, helpful particularly for high dimensional informational collection issues [18].

**2) Random Forest:** Random Forest or discretionary decision woods are a gathering learning procedure for portrayal. The backslide and various end overs that works by using the planning time to build enormous decision trees [19]. The classes use the yielding technique (game plan) or mean figure (backslide) of the individual trees [19]. Random Forests are a mixture of tree pointers to such a degree. That the tree depends upon the estimations of a subjective vector analysed self-sufficiently.

**3) J48:** The C4.5 algorithm is used for constitution to select trees. Trees are called as J48 in weka which are experimented [20]. Channels have similar characteristics as classifiers. That are composed in a chain of importance: full name of weka is called as J48 [20]. Information mining incorporates the conscious examination of gigantic enlightening lists.

**4) IBK:** A non-parametric estimation system that deals with k-nearest neighbours is called IBK. That is needed for portrayal and backslide. In the two cases, the data contains the k nearest planning points. That is of reference in the component space [21]. In k-NN request, the yield is a class interest. Pushed by applying Text Categorization to organizing web list things [21]. IBK selects the number nearest neighbours between 1.

**5) Naive Bayes:** A classifier is an AI model that is used to isolate different things. It is used to subject to explicit features. A Naive Bayes classifier is a anticipated AI model. This model is utilized for classification task. The core of the classifier depends on the Bayes hypothesis [22]. Implementing Bayes theorem, we can evaluate the prospect measure of A event, knowing the fact that B has taken place. Here, B is the proof and A would be the hypothesis [23]. The supposition is made based on the hypothesis. That is the predictors of the features are not dependent. That is existence of one fixed feature does not influence other. That’s why naive is being called [24].



**EVALUATION METRICS**

To compute the performance of each classifiers we have used root mean squared error and kappa statistics along with classification accuracy. RMSE is applied to measure the difference of the actual value and the predicted value done by a classifier.

RMSE value is always non-negative and value of 0 represents perfect fit.



where, yˆi , is the value predicted by the classifier and yi is the actual result.

**RESULT**

This section describes about the accuracies of the classifiers at different stages of our work.

We can divide our procedure in three stages. The first stage is to apply the classifiers on our data set before applying SMOTE technique.

As our data set is imbalanced, so before applying SMOTE, it’s accuracy is very low. The accuracies are illustrated in figure 2.

Here, Random forest, J48 and bagging gives same classification accuracy and this is the highest among the five classifiers.

To make the data set a balanced one, SMOTE is applied afterwards. Thus, the accuracies increase. At this stage, again random forest gives highest accuracy. Like the previous stage, Naive Bayes gives the lowest accuracy.

Now at the final stage, satisfying outcome is acquired by applying classification algorithms by resampling the data set without replacement [25].

In algorithm 1, this resampling and determining the accuracy is done in a loop. The loop continues when the accuracies increase in each turn. It breaks, when the accuracy starts to decrease. Thus, the highest accuracy obtained by each classifiers is listed and shown in figure 2.

Among the five classifiers, random forest gives the highest classification accuracy. Though all the five classifiers give significantly good accuracy here, that is above 90%.

Figure 2 illustrates the kappa statistics and root mean squared error of the classifiers. Here, random forest again gives maximum kappa statistics and minimum root mean squared error. The lazy classifier IBK gives the minimum accuracy with highest root mean squared error.

Therefore, the performance metrics of the five classifiers state the conclusion that random forest classifier gives the best outcomes in terms of accuracy, kappa statistics and also root mean squared error for our data set.

**PROGRAM**

import pandas as pd

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

dataset = pd.read\_csv("../Dataset/IMDB\_movie\_reviews\_details.csv")

dataset

dataset.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 10 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 id 1000 non-null int64

1 name 1000 non-null object

2 year 1000 non-null object

3 runtime 1000 non-null int64

4 genre 1000 non-null object

5 rating 1000 non-null float64

6 meta score 841 non-null float64

7 timeline 1000 non-null object

8 votes 1000 non-null object

9 gross 829 non-null object

D types: float64(2), int64(2), object(6)

memory usage: 54.8+ KB

dataset.head()

dataset.isna().sum()

id 0

name 0

year 0

runtime 0

genre 0

rating 0

meta score 159

timeline 0

votes 0

gross 171

d type: int64

Deleting first column

df = dataset

df = df.drop(['id'], axis=1)

df.head()

Year column converted to int

df['year'] = df['year'].str.replace('I', '')

df['year'] = df['year'].str.replace(' ', '')

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

df.info()

<class 'pandas.core.frame.DataFrame'>

Range Index: 1000 entries, 0 to 999

Data columns (total 9 columns):

# Column Non-Null Count D type

--- ------ -------------- -----

0 name 1000 non-null object

1 year 1000 non-null int64

2 runtime 1000 non-null int64

3 genre 1000 non-null object

4 rating 1000 non-null float64

5 meta score 841 non-null float64

6 timeline 1000 non-null object

7 votes 1000 non-null object

8 gross 829 non-null object

d types: float64(2), int64(2), object(5)

memory usage: 50.8+ KB

Votes column converted to int

df['votes'] = df['votes'].str.replace(',', '')

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

df.info()

<class 'pandas.core.frame.DataFrame'>

Range Index: 1000 entries, 0 to 999

Data columns (total 9 columns):

# Column Non-Null Count D type

--- ------ -------------- -----

0 name 1000 non-null object

1 year 1000 non-null int64

2 runtime 1000 non-null int64

3 genre 1000 non-null object

4 rating 1000 non-null float64

5 meta score 841 non-null float64

6 timeline 1000 non-null object

7 votes 1000 non-null int64

8 gross 829 non-null object

D types: float64(2), int64(3), object(4)

memory usage: 54.8+ KB

Gross column converted to int

df['gross'] = df['gross'].str.replace('$', '')

df['gross'] = df['gross'].str.replace('M', '')

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

df.info()

<class 'pandas.core.frame.DataFrame'>

Range Index: 1000 entries, 0 to 999

Data columns (total 9 columns):

# Column Non-Null Count D type

--- ------ -------------- -----

0 name 1000 non-null object

1 year 1000 non-null int64

2 runtime 1000 non-null int64

3 genre 1000 non-null object

4 rating 1000 non-null float64

5 meta score 841 non-null float64

6 timeline 1000 non-null object

7 votes 1000 non-null int64

8 gross 829 non-null float64

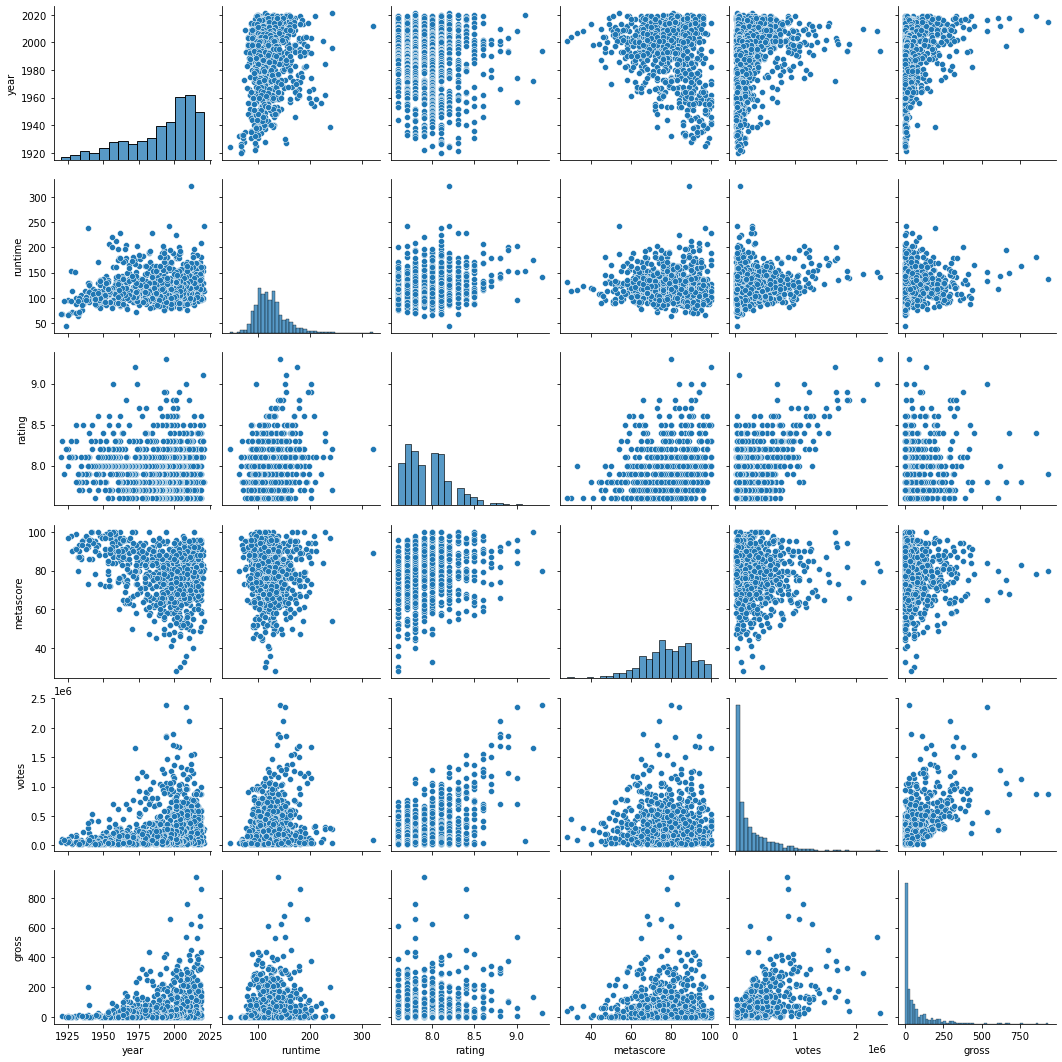
D types: float64(3), int64(3), object(3)

memory usage: 58.7+ KB

Checking how numbers correlate

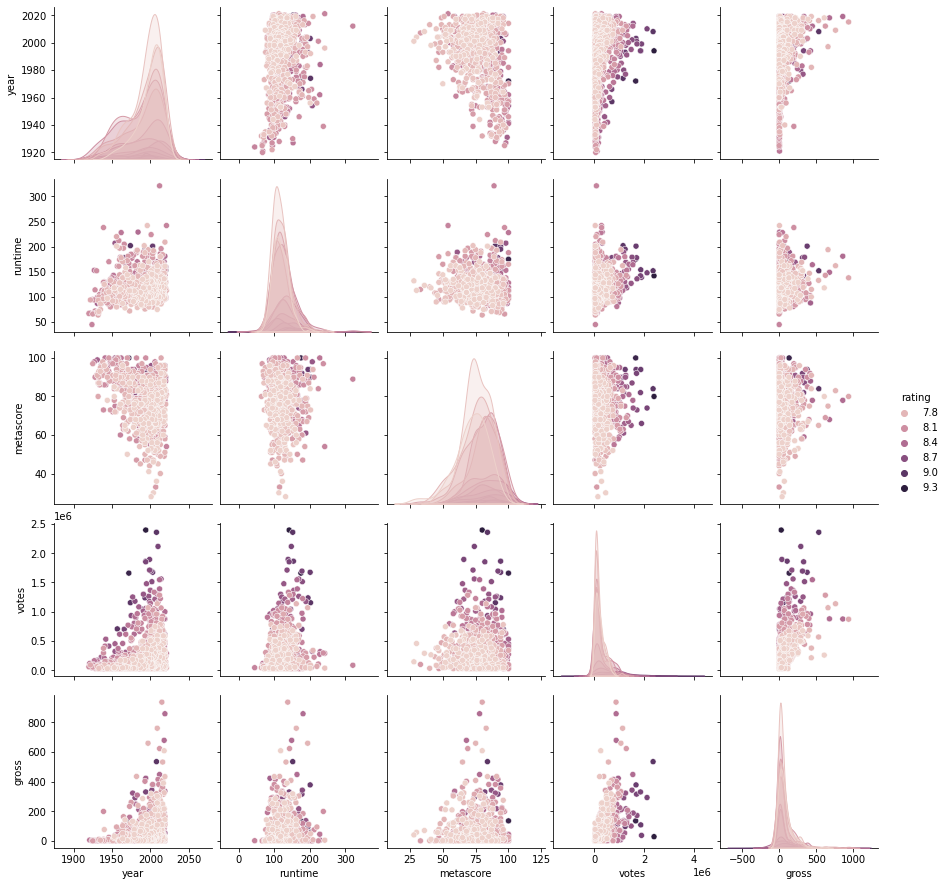
sns.pairplot(df)

<seaborn.axisgrid.PairGrid at 0x336aeef8>



sns.pairplot(df, hue='rating')

<seaborn.axisgrid.PairGrid at 0x3876cd18>



plt.figure()

cor = df.corr()

sns.heatmap(cor, annot=True, cmap='coolwarm')

plt.ylim()

(6.0, 0.0)



Feature Engineering

Df

df[["genre\_1","genre\_2","genre\_3"]] = df['genre'].str.split(',', n = 3, expand=True)

df = df.drop(['genre'], axis=1)

df['genre\_1'] = df['genre\_1'].str.replace(' ', '')

df['genre\_2'] = df['genre\_2'].str.replace(' ', '')

df['genre\_3'] = df['genre\_3'].str.replace(' ', '')

l1 = df.genre\_1.unique()

l2 = df.genre\_2.unique()

l3 = df.genre\_3.unique()

l = list(l1) + list(l2) + list(l3)

l = [i for i in l if i]

l = list(set(l))

print(l)

['Action', 'History', 'Mystery', 'Sport', 'Romance', 'Western', 'Thriller', 'Musical', 'Animation', 'Film-Noir', 'Music', 'Horror', 'War', 'Sci-Fi', 'Comedy', 'Adventure', 'Fantasy', 'Crime', 'Family', 'Biography', 'Drama']

len(l)

21

listofzeros = [0] \* 1000

for genre in l:

    df[genre] = listofzeros

df.head()

for genre in l:

    for x in range(1000):

        if df.at[x, 'genre\_1'] == genre or df.at[x, 'genre\_2'] == genre or df.at[x, 'genre\_3'] == genre:

            df.at[x, genre] = 1

df.head()

Checking how genre correlate

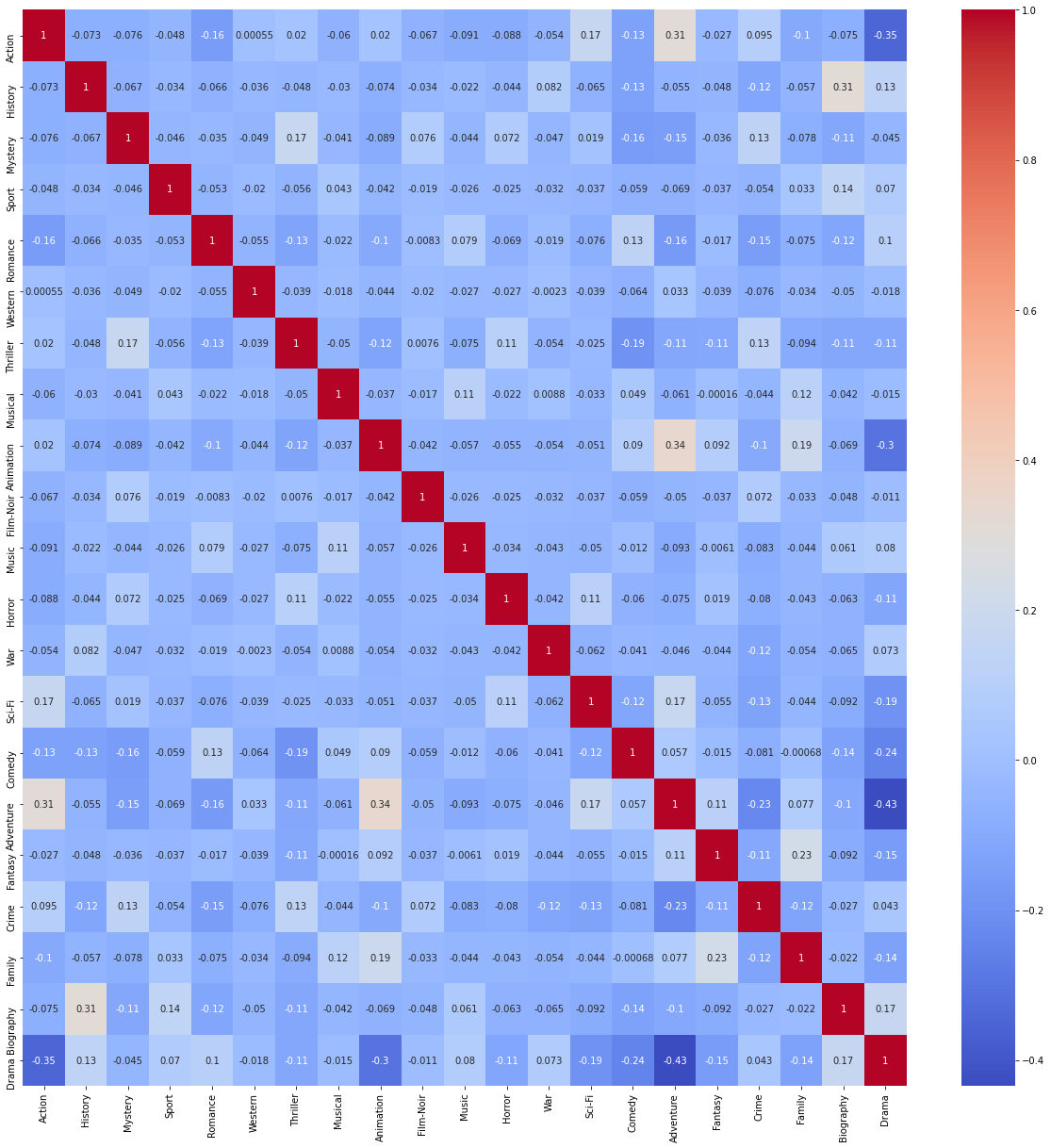
plt.figure(figsize=(21,21))

cor = df[l].corr()

sns.heatmap(cor, annot=True, cmap='coolwarm')

plt.ylim()

(21.0, 0.0)



Removing unwanted columns for model training

df.info()

<class 'pandas.core.frame.DataFrame'>

Rang Index: 1000 entries, 0 to 999

Data columns (total 32 columns):

# Column Non-Null Count D type

--- ------ -------------- -----

0 name 1000 non-null object

1 year 1000 non-null int64

2 runtime 1000 non-null int64

3 rating 1000 non-null float64

4 meta score 841 non-null float64

5 timeline 1000 non-null object

6 votes 1000 non-null int64

7 gross 829 non-null float64

8 genre\_1 1000 non-null object

9 genre\_2 892 non-null object

10 genre\_3 643 non-null object

11 Action 1000 non-null int64

12 History 1000 non-null int64

13 Mystery 1000 non-null int64

14 Sport 1000 non-null int64

15 Romance 1000 non-null int64

16 Western 1000 non-null int64

17 Thriller 1000 non-null int64

18 Musical 1000 non-null int64

19 Animation 1000 non-null int64

20 Film-Noir 1000 non-null int64

21 Music 1000 non-null int64

22 Horror 1000 non-null int64

23 War 1000 non-null int64

24 Sci-Fi 1000 non-null int64

25 Comedy 1000 non-null int64

26 Adventure 1000 non-null int64

27 Fantasy 1000 non-null int64

28 Crime 1000 non-null int64

29 Family 1000 non-null int64

30 Biography 1000 non-null int64

31 Drama 1000 non-null int64

D types: float64(3), int64(24), object(5)

memory usage: 230.5+ KB

#Not taking 'Horror' column to avoid dummy variable trap

df\_model = df[['year', 'runtime', 'votes', 'metascore', 'gross', 'Mystery', 'Drama', 'Musical', 'Fantasy', 'Adventure', 'Western', 'Thriller', 'War', 'Biography', 'Family', 'Sport', 'Film-Noir', 'Music', 'Sci-Fi', 'Animation', 'Romance', 'Crime', 'Action', 'Comedy', 'History', 'rating']]

df\_model.head()

df\_model.info()

<class 'pandas.core.frame.DataFrame'>

Range Index: 1000 entries, 0 to 999

Data columns (total 26 columns):

# Column Non-Null Count D type

--- ------ -------------- -----

0 year 1000 non-null int64

1 runtime 1000 non-null int64

2 votes 1000 non-null int64

3 meta score 841 non-null float64

4 gross 829 non-null float64

5 Mystery 1000 non-null int64

6 Drama 1000 non-null int64

7 Musical 1000 non-null int64

8 Fantasy 1000 non-null int64

9 Adventure 1000 non-null int64

10 Western 1000 non-null int64

11 Thriller 1000 non-null int64

12 War 1000 non-null int64

13 Biography 1000 non-null int64

14 Family 1000 non-null int64

15 Sport 1000 non-null int64

16 Film-Noir 1000 non-null int64

17 Music 1000 non-null int64

18 Sci-Fi 1000 non-null int64

19 Animation 1000 non-null int64

20 Romance 1000 non-null int64

21 Crime 1000 non-null int64

22 Action 1000 non-null int64

23 Comedy 1000 non-null int64

24 History 1000 non-null int64

25 rating 1000 non-null float64

D types: float64(3), int64(23)

memory usage: 203.2 KB

**Calculating mean and filling missing data**

X = df\_model.iloc[:,:-1].values

y = df\_model.iloc[:,25].values

np.set\_printoptions(suppress=True)

X

array([[ 1994., 142., 2394059., ..., 0., 0., 0.],

[ 1972., 175., 1658439., ..., 0., 0., 0.],

[ 2020., 153., 78266., ..., 0., 0., 0.],

...,

[ 1953., 118., 37753., ..., 0., 0., 0.],

[ 1953., 118., 44086., ..., 0., 0., 0.],

[ 1944., 97., 26903., ..., 0., 0., 0.]])

X[0,:]

array([ 1994. , 142. , 2394059. , 80. , 28.34,

0. , 1. , 0. , 0. , 0. ,

0. , 0. , 0. , 0. , 0. ,

0. , 0. , 0. , 0. , 0. ,

0. , 0. , 0. , 0. , 0. ])

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=True)

#X\_train, X\_test

#y\_train, y\_test

**Implementing Algorithms**

**Linear Regression**

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train,y\_train)

y\_pred = regressor.predict(X\_test)

accuracy = regressor.score(X\_test,y\_test)

print('Accuracy of the model is',accuracy\*100,'%')

**Accuracy of the model is 49.27193177432364 %**

**Polynomial Regression**

from sklearn.preprocessing import PolynomialFeatures

for degree in range(1,4):

    poly\_reg = PolynomialFeatures(degree = degree)

    X\_poly = poly\_reg.fit\_transform(X)

    lin\_reg2 = LinearRegression()

    lin\_reg2.fit(X\_poly,y)

    accuracy = lin\_reg2.score(poly\_reg.fit\_transform(X),y)

    print('Accuracy of the model is with degree',str(degree),'=',accuracy\*100,'%')

Accuracy of the model is with degree 1 = 44.74549675905853 %

Accuracy of the model is with degree 2 = 55.163495947730986 %

Accuracy of the model is with degree 3 = 69.56899235247278 %

**Random Forest Regression**

from sklearn.ensemble import RandomForestRegressor

x = 500

regressor = RandomForestRegressor(n\_estimators = x)

regressor.fit(X,y)

accuracy = regressor.score(X,y)

print('Accuracy of the model with',str(x),'n\_estimators','=',accuracy\*100,'%')

Accuracy of the model with 500 n\_estimators = 93.97201024913363 %

**SVR**

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

sc\_y = StandardScaler()

X = sc\_X.fit\_transform(X)

y = sc\_y.fit\_transform(y.reshape(-1,1))

from sklearn.svm import SVR

regressor = SVR(kernel='rbf')

regressor.fit(X,y)

accuracy = regressor.score(X,y)

print('Accuracy of the model is',accuracy\*100,'%')

Accuracy of the model is 59.742573708014255 %

README FILE :

**# IMDb Price Prediction**

IMDb is the world’s most popular and authoritative source for movie, TV, and celebrity content. IMDb users often look at ratings to get an idea of how good movies are, so that they can decide what movies to watch or which ones to prioritize. However, movies that are not yet released don’t have ratings, and even the ones with few votes often change as more users vote. Therefore, I wrote code to predict IMDb ratings of new movies based on various features, such as budget, actors, directors, writers, release year, genres, and plot. While others have used linear regressions to predict ratings of movies in general, those predictions rely on features like movie earnings or number of votes, which would not be available for new movies. I instead implemented two more algorithms to test the predictions and its accuracy.

![](/Images/imdb1.png)

**## Dataset**

The dataset which is used here, is collected from Kaggle website. Here is the link of the dataset : https://www.kaggle.com/datasets/luiscorter netflix-original-films-imdb-scores

**## Goal**

The goal of this project is to make a prediction model which will predict the rating of the movies using different parameters.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**## What have I done?**

1. Importing all the required libraries. Check [`requirements.txt`](/requirements.txt).

2. Upload the dataset and the Jupyter Notebook file.

3. Exploratory Data Analysis

4. Data Processing

5. Prediction Models

    - Linear Regression

    - Polynomial Regression

    - Random Forest

    - SVR

8. Validation Process

9. Conclusion

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**## Libraries used**

1. Numpy

2. Pandas

3. Matplotlib

4. Sklearn

5. Seaborn

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**## Exploratory Data Analysis**

1. **\*\*Correlation between numerical parameters\*\***

![](/Images/imdb2.png)

2. **\*\*Confusion matrix between numerical parmameters\*\***

![](/Images/imdb3.png)

3. **\*\*Confusion matrix after feature engineering\*\***

![](/Images/imdb4.png)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**## Model comparison**

Here I have deployed four algorithms to deploy the models, now let's check the accuracy scores for these models.

|Models|Accuray Score|

|-|-|

|Linear Regression|0.49|

|Polynomial Regression|0.69|

|Random Forest Regression|0.93|

|SVR|0.59|

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CONCLUSION**

As the business market of film industries are becoming huge day by day, competition here is also growing complex. Therefore, predicting movie’s rating is growing complex also. Our model is developed on a real world data set and it is collected from two platforms, Wikipedia and IMDb.

The model can also be used to predict some other ratings like Rotten Tomato or Metacritic. Other than films, TV shows, music shows, etc. can be predicted by our model using the features of our model. describes some features having more influence on movie success and some other features having less or no influence. According to , budget is having a small positive influence but cast or actor/actress doesn’t have any influence on Russian film industry.

Thus, our work can be done by a weighted feature classification to reflect these influences. Along with hollywood movie dataset, bollywood or other movie dataset can be used to make the model more efficient.

The database can be enriched by including the movies released on recent years. That is, along with 2023, movies from 2022 or 2021 can be included.

In conclusion, predicting IMDb scores is a complex task that involves analysing various factors such as user reviews, critic ratings, genre, director, and more. Machine learning models, like regression or deep learning, can be used for this purpose, but the accuracy of predictions may vary. Careful feature selection and data preprocessing are crucial for improving the model's performance. Additionally, domain-specific knowledge and continuous model evaluation are essential to create a reliable IMDb score prediction system.

**THANK YOU**