APPLIED DATA SCIENCE GROUP-2
IMDB SCORE PREDICTION (PHASE - 5)
DEVELOPMENT & SUBMISSION

PROBLEM STATEMENT

The problem statement for the IMDb score prediction project:

"In the film industry, the success of a movie is often measured by its rating. The Internet Movie Database (IMDb) is one of the most popular platforms where movies are rated by viewers. These ratings play a crucial role in influencing audience viewership and are of great interest to film production companies, directors, and investors.

The task at hand is to build a predictive model that can accurately estimate the IMDb score of a movie before its release based on various factors such as director, actors, budget, genres, etc. This would provide valuable insights into the potential success of a movie and could guide decision-making processes in film production and marketing strategies.

The challenge lies in selecting relevant features from the available data, handling missing or inconsistent data, choosing an appropriate regression algorithm for prediction, and validating the model’s performance using suitable metrics. The goal is to achieve a model with high accuracy and robustness that can generalize well to new, unseen movie data."

DESIGN AND ANALYSIS

The design and analysis for the IMDb score prediction project can be broken down into several steps:

1. Problem Understanding:

The first step is to understand the problem at hand. The goal is to predict the IMDb score of a movie based on various features. This is a regression problem as the IMDb score is a continuous variable.

2. Data Collection:

The next step is to collect data that can be used to train the model. The dataset used in this project contains movie metadata from IMDb, including features such as director, actors, budget, genres, etc.

3. Exploratory Data Analysis (EDA):

This involves understanding the dataset through descriptive statistics and visualizations. It helps in identifying patterns, correlations, and outliers in the data.

4. Data Preprocessing:

This step involves cleaning the data and making it suitable for modeling. It includes handling missing values, encoding categorical variables, feature scaling, etc.

5. Feature Selection:

Not all features in the dataset may be useful for predicting the IMDb score. Feature selection methods can be used to select the most relevant features.

6. Model Building:

This involves choosing a suitable regression algorithm and training it on the preprocessed data. The choice of algorithm depends on the nature of the data and the problem.

7. Model Evaluation:

After training the model, it’s important to evaluate its performance using suitable metrics. For regression problems, metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared can be used.

8. Model Optimization:

Based on the evaluation results, the model may need to be optimized by tuning its hyperparameters or using different algorithms.

9. Model Deployment:

Once the model is optimized and achieves satisfactory performance, it can be deployed for use in predicting IMDb scores of new movies.Throughout these steps, it’s important to document each step and decision made for future reference and reproducibility.

Phase-1:

Problem Definition and Design Thinking

- Introduction
- Primary goals
- Sample dataset
- Problem
- Design & Analysis
- Summary

Phase-2: Innovation

Consider exploring advanced regression techniques like Gradient Boosting or Neural Networks for improved prediction accuracy

- Introduction
- Dataset details
- Libraries
- How to train and test
- Process flow

Phase 3 – Development part -1

Begin building the IMDb score prediction model by loading and preprocessing the dataset.

- Loading the dataset
- Preprocessing the dataset

Phase 4 – Development part -2

Continue building the IMDb score prediction model by feature engineering, model training, and evaluation

- Model training
- Feature engineering
- Evaluation

Phase 5 – Project Documentation & Submission

Document the IMDb score prediction project and prepare it for submission.

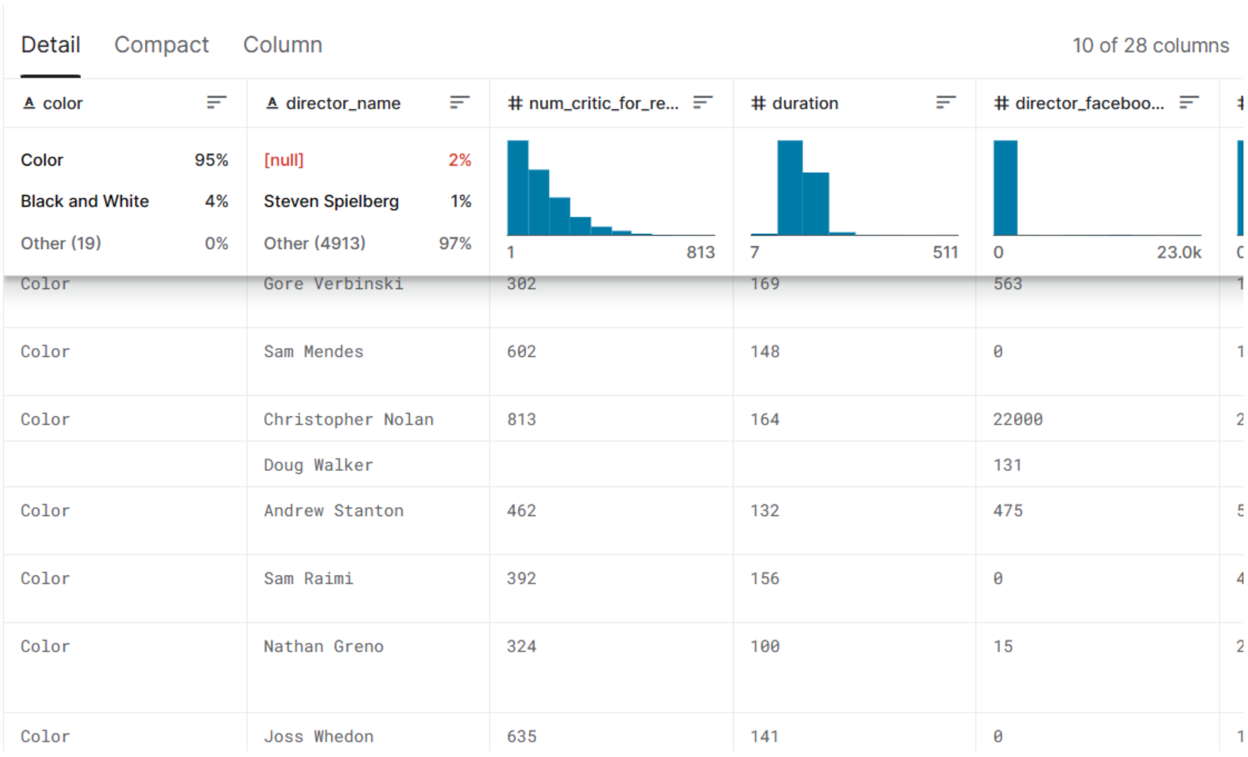
- Documentation
- Submission

DATASET

dataset: http://www.kaggle.com

dataset name: IMDB score Prediction

dataset link: https://www.kaggle.com/datasets/bobirino/movie-metadata



Detail Compact Column 10 of 28 columns

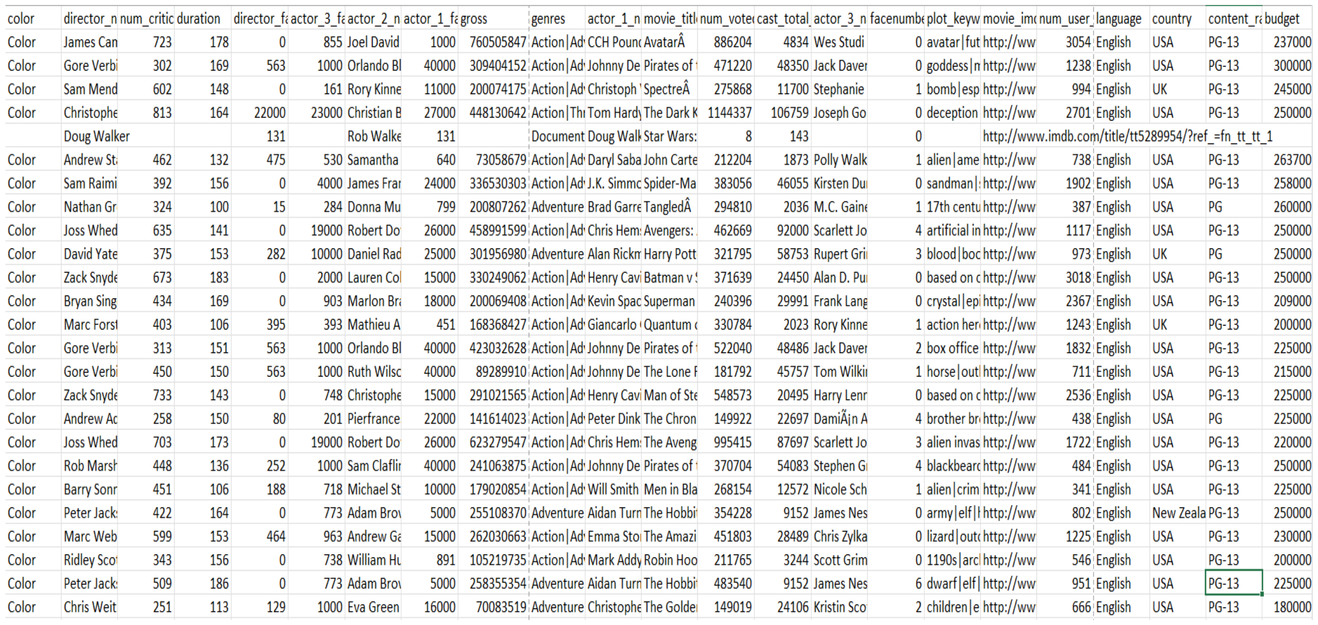
color	director_name	# num_critic_for_re...	# duration	# director_faceboo...	
Color 95%	[null] 2%				
Black and White 4%	Steven Spielberg 1%				
Other (19) 0%	Other (4913) 97%	1 – 813	7 – 511	0 – 23.0k	
Color	Gore Verbinski	302	169	563	1
Color	Sam Mendes	602	148	0	1
Color	Christopher Nolan	813	164	22000	2
	Doug Walker			131	
Color	Andrew Stanton	462	132	475	5
Color	Sam Raimi	392	156	0	4
Color	Nathan Greno	324	100	15	2
Color	Joss Whedon	635	141	0	1

STEPS FOR PREPROCESSING DATA:

1. **Dataset Acquisition**: The initial step is to gather the dataset that you'll be working with.
2. **Library Importation**: Import all necessary libraries that will be used for data preprocessing.
3. **Dataset Importation**: Load the dataset into your working environment.
4. **Handling Missing Values**: Identify any missing values in the dataset and decide on the best strategy to handle them, such as imputation or deletion.
5. **Categorical Data Encoding**: Convert categorical data into a format that can be easily understood by machine learning algorithms, typically through one-hot encoding or label encoding.
6. **Dataset Division**: Split the dataset into a training set and a testing set. This allows for proper evaluation of the model's performance.
7. **Feature Scaling**: Normalize or standardize features to ensure that they're on a similar scale and to prevent any one feature from dominating others.

1.Acquire the dataset:

Dataset link: https://www.kaggle.com/datasets/bobirino/movie-metadata



color	director_n	num_critic	duration	director_fa	actor_3_fa	actor_2_n	actor_1_fa	gross	genres	actor_1_n	movie_titl	num_vote	cast_total_	actor_3_n	facenumbe	plot_keyw	movie_imc	num_user_	language	country	content_ra	budget
Color	James Can	723	178	0	855	Joel David	1000	760505847	Action|Ad	CCH Pounc	AvatarÂ	886204	4834	Wes Studi	0	avatar|fut	http://ww	3054	English	USA	PG-13	237000
Color	Gore Verbi	302	169	563	1000	Orlando Bl	40000	309404152	Action|Ad	Johnny De	Pirates of	471220	48350	Jack Daver	0	goddess|n	http://ww	1238	English	USA	PG-13	300000
Color	Sam Mend	602	148	0	161	Rory Kinne	11000	200074175	Action|Ad	Christoph	SpectreÂ	275868	11700	Stephanie	1	bomb|esp	http://ww	994	English	UK	PG-13	245000
Color	Christophe	813	164	22000	23000	Christian B	27000	448130642	Action|Th	Tom Hardy	The Dark K	1144337	106759	Joseph Go	0	deception	http://ww	2701	English	USA	PG-13	250000
	Doug Walker			131		Rob Walke	131		Document	Doug Walk	Star Wars:	8	143		0		http://www.imdb.com/title/tt5289954/?ref_=fn_tt_tt_1					
Color	Andrew St	462	132	475	530	Samantha	640	73058679	Action|Ad	Daryl Saba	John Carte	212204	1873	Polly Walk	1	alien|ame	http://ww	738	English	USA	PG-13	263700
Color	Sam Raimi	392	156	0	4000	James Frar	24000	336530303	Action|Ad	J.K. Simmc	Spider-Ma	383056	46055	Kirsten Du	0	sandman|	http://ww	1902	English	USA	PG-13	258000
Color	Nathan Gr	324	100	15	284	Donna Mu	799	200807262	Adventure	Brad Garre	TangledÂ	294810	2036	M.C. Gaine	1	17th centu	http://ww	387	English	USA	PG	260000
Color	Joss Whed	635	141	0	19000	Robert Do	26000	458991599	Action|Ad	Chris Hem	Avengers:	462669	92000	Scarlett Jo	4	artificial in	http://ww	1117	English	USA	PG-13	250000
Color	David Yate	375	153	282	10000	Daniel Rad	25000	301956980	Adventure	Alan Rickm	Harry Pott	321795	58753	Rupert Gri	3	blood|boc	http://ww	973	English	UK	PG	250000
Color	Zack Snyde	673	183	0	2000	Lauren Col	15000	330249062	Action|Ad	Henry Cavi	Batman v	371639	24450	Alan D. Pu	0	based on c	http://ww	3018	English	USA	PG-13	250000
Color	Bryan Sing	434	169	0	903	Marlon Bra	18000	200069408	Action|Ad	Kevin Spac	Superman	240396	29991	Frank Lang	0	crystal|epi	http://ww	2367	English	USA	PG-13	209000
Color	Marc Forst	403	106	395	393	Mathieu A	451	168368427	Action|Ad	Giancarlo	Quantum	330784	2023	Rory Kinne	1	action her	http://ww	1243	English	UK	PG-13	200000
Color	Gore Verbi	313	151	563	1000	Orlando Bl	40000	423032628	Action|Ad	Johnny De	Pirates of	522040	48486	Jack Daver	2	box office	http://ww	1832	English	USA	PG-13	225000
Color	Gore Verbi	450	150	563	1000	Ruth Wilsc	40000	89289910	Action|Ad	Johnny De	The Lone F	181792	45757	Tom Wilki	1	horse|outl	http://ww	711	English	USA	PG-13	215000
Color	Zack Snyde	733	143	0	748	Christophe	15000	291021565	Action|Ad	Henry Cavi	Man of Ste	548573	20495	Harry Lenr	0	based on c	http://ww	2536	English	USA	PG-13	225000
Color	Andrew Ac	258	150	80	201	Pierfrance	22000	141614023	Action|Ad	Peter Dink	The Chron	149922	22697	DamiÃ¡n A	4	brother br	http://ww	438	English	USA	PG	225000
Color	Joss Whed	703	173	0	19000	Robert Do	26000	623279547	Action|Ad	Chris Hem	The Aveng	995415	87697	Scarlett Jo	3	alien invas	http://ww	1722	English	USA	PG-13	220000
Color	Rob Marsh	448	136	252	1000	Sam Claflir	40000	241063875	Action|Ad	Johnny De	Pirates of	370704	54083	Stephen G	4	blackbearc	http://ww	484	English	USA	PG-13	250000
Color	Barry Sonr	451	106	188	718	Michael St	10000	179020854	Action|Ad	Will Smith	Men in Bla	268154	12572	Nicole Sch	1	alien|crim	http://ww	341	English	USA	PG-13	225000
Color	Peter Jack	422	164	0	773	Adam Brov	5000	255108370	Adventure	Aidan Turn	The Hobbi	354228	9152	James Nes	0	army|elf|l	http://ww	802	English	New Zeala	PG-13	250000
Color	Marc Web	599	153	464	963	Andrew Ga	15000	262030663	Action|Ad	Emma Sto	The Amazi	451803	28489	Chris Zylka	0	lizard|outc	http://ww	1225	English	USA	PG-13	230000
Color	Ridley Sco	343	156	0	738	William Hu	891	105219735	Action|Ad	Mark Addy	Robin Hoo	211765	3244	Scott Grim	0	1190s|arcl	http://ww	546	English	USA	PG-13	200000
Color	Peter Jack	509	186	0	773	Adam Brov	5000	258355354	Adventure	Aidan Turn	The Hobbi	483540	9152	James Nes	6	dwarf|elf|	http://ww	951	English	USA	PG-13	225000
Color	Chris Weit	251	113	129	1000	Eva Green	16000	70083519	Adventure	Christophe	The Golde	149019	24106	Kristin Sco	2	children|e	http://ww	666	English	USA	PG-13	180000

DESCRIBE DATA USED

To predict IMDb scores, a wide variety of data can be used. The accuracy and effectiveness of the prediction can depend on the complexity and size of the dataset, as well as the specific machine learning or statistical techniques used.

1. Movie Metadata: This includes information like the movie's title, release date, genre, director, writer, and production company.

2.Cast and Crew Information: Data about the actors, actresses, and key crew members, such as the director, producer, and writer.

3. Box Office Performance: Details about a movie's financial success, including its budget, gross revenue, and profitability.

4. Movie Plots and Summaries: Descriptions of the movie's plot or summary, which can be used for natural language processing and sentiment analysis.

5. User Reviews and Ratings: IMDb has user-generated reviews and ratings, which can be used as features. Sentiment analysis on user reviews can provide valuable insights.

6. Movie Posters and Trailers: Visual data like movie posters and trailers can be analyzed for aesthetic and marketing factors.

7. Awards and Nominations: Information about awards won or nominated for can indicate a movie's quality and critical reception.

8.Runtime: The duration of the movie can sometimes be a predictor of IMDb score, as very long or very short movies may have different audience expectations.

9.Release Information: Data about the geographical and temporal release of the movie can be relevant, as release strategies vary.

10.Social Media Data: Information about a movie's presence on social media platforms, such as the number of followers on official pages, can be used to gauge audience engagement.

Innovation & Design:

- **Personalized Recommendations:**

  Develop a recommendation system that suggests movies based on a user's IMDb ratings and preferences. Use machine learning algorithms to make these recommendations more accurate over time.

- **Visual Data Analytics:**

  Create interactive visualizations that allow users to explore IMDb data. Visualize trends, ratings distribution, and other insights that go beyond a simple numeric score.

- **Crowdsourced Reviews:**

  Enable users to submit detailed reviews alongside their ratings. Implement sentiment analysis to summarize the overall sentiment of reviews.

- **Filter and Search Enhancement:**

  Improve IMDb's filtering and search capabilities. Add advanced filters such as genre-specific, release year, or director-based searches.

- **IMDb Score Predictions:**

  Develop a model to predict IMDb scores for movies before they are released, considering factors like the cast, crew, and pre-release buzz.

- **User-Generated Lists:**

  Allow users to create and share lists of their favorite movies, creating a sense of community and enabling users to discover new films.

- **Mobile App Integration:**

  Create a mobile app that seamlessly integrates these features, making it easy for users to access IMDb's enhanced functionalities on the go.

- **Data Insights for Filmmakers:**

  Offer insights to filmmakers and studios on how their movies are rated by users, potentially helping them make improvements in future projects.

- **Rating Aggregation:**

  Aggregate IMDb ratings with ratings from other sources like Rotten Tomatoes or Metacritic to provide a more comprehensive view of a movie's reception.

- **Accessibility and User-Centric Design:**

  Ensure the platform is accessible to all users, including those with disabilities, and prioritize a user-centric design for a seamless experience.

- **Community Engagement:**

  Implement features like forums or discussion boards where users can engage in meaningful discussions about movies and ratings.

- **Data Security and Privacy:**

  Pay strict attention to data security and user privacy, especially when handling user-generated content and personal preferences.

Details of Libraries:

- **IMDbPY (Python Library):**

  IMDbPY is a Python package specifically designed to access and retrieve data from the IMDb website. You can use it to fetch movie details, cast information, user reviews, and IMDb ratings.

- **OMDb API:**

  The Open Movie Database (OMDb) API provides access to a vast amount of movie-related data, including IMDb ratings, plot summaries, release dates, and more. It's easy to use and doesn't require an IMDb API key.

- **TMDb API:**

  The Movie Database (TMDb) API offers movie information, including user ratings and reviews. While not IMDb-specific, it can complement IMDb data for a broader perspective.

- **Beautiful Soup (Python Library):**

  Beautiful Soup is a Python library for web scraping. You can use it to extract data from IMDb web pages when IMDbPY doesn't provide the specific data you need.

- **Pandas (Python Library):**

  Pandas is a powerful data manipulation and analysis library for Python. It's great for handling, cleaning, and analyzing IMDb data obtained through IMDbPY or web scraping.

- **Matplotlib and Seaborn (Python Libraries):**

  Matplotlib and Seaborn are popular Python libraries for data visualization. You can use them to create various charts and plots to visualize IMDb ratings distribution, trends, and correlations.

- **Scikit-learn (Python Library):**

  If your project involves machine learning or predictive modeling based on IMDb data, Scikit-learn is a go-to library for tasks like building recommendation systems or predicting IMDb scores.

- **D3.js (JavaScript Library):**

  If you plan to create interactive web visualizations for IMDb data, D3.js is a powerful JavaScript library for data-driven graphics. It can be used to create dynamic and interactive data visualizations.

- **SQLite (Database):**

  SQLite is a lightweight database engine that you can use to store and manage IMDb-related data locally, making it easier to query and analyze large datasets.

- **Flask (Python Framework):**

  If you're building a web application to present IMDb□related data, Flask is a lightweight Python web framework that can help you develop the backend of your application.

- **React or Vue.js (JavaScript Frameworks):**

  For the frontend of a web application, React or Vue.js can be useful frameworks to create responsive and interactive user interfaces.

• **Heroku or AWS (Cloud Services):**

If you plan to deploy your IMDb-related project online, platforms like Heroku or AWS can host your web application and databases in the cloud.

EVALUATION AND METRICS

When building a model for IMDb score prediction, it's important to evaluate its performance to determine how well it predicts IMDb scores. Here are common evaluation metrics and techniques used in IMDb score prediction:

1.Mean Absolute Error (MAE):

- MAE measures the average absolute difference between the predicted IMDb scores and the actual IMDb scores. A lower MAE indicates better performance.

2.Mean Squared Error (MSE):

- MSE calculates the average of the squared differences between predicted and actual IMDb scores. It penalizes larger errors more heavily. Smaller MSE values are better.

3.Root Mean Squared Error (RMSE):

- RMSE is the square root of MSE, and it provides a measure of the average prediction error in the same units as the IMDb scores. Lower RMSE values indicate better performance.

4.R-squared (R2) Score:

- R2 measures the proportion of the variance in IMDb scores that is explained by the model. It ranges from 0 to 1, with higher values indicating better model fit. An R2 of 1 means the model perfectly predicts IMDb scores.

5.Coefficient of Determination (COD):

- COD, similar to R2, measures how well the model explains the variance in IMDb scores. It can help assess the goodness of fit.

6.Adjusted R-squared:

- Adjusted R-squared takes into account the number of predictors in the model and adjusts R2 accordingly. It helps prevent overfitting by penalizing the inclusion of unnecessary features.

7.Percentile Ranking:

- You can calculate the percentile rank of the model's predictions to determine how often it correctly predicts IMDb scores compared to other movies.

8.Residual Analysis:

- Analyze the model's residuals (the differences between predicted and actual scores). Plotting residuals can help identify patterns or heteroscedasticity, which can indicate model deficiencies.

9. Cross-Validation:

- Use cross-validation techniques (e.g., k-fold cross-validation) to assess how well the model generalizes to unseen data. Cross-validation provides a more robust estimate of the model's performance.

10.Bias-Variance Tradeoff:

Consider the balance between bias and variance. High bias indicates underfitting, while high variance indicates overfitting. Achieving the right balance is crucial for model performance.

IMDB SCORE PREDICTION IMPLEMENTATION

The task is to build a model for predicting the IMDB rating for the various movies

Importing the required library for the task

In [1]:

```
import numpy as np
import pandas as pd
import csv
from scipy import stats
import matplotlib.pyplot as plt
```

In [2]:

```
rawdata = pd.read_csv('movie_metadata.csv')
```

In [3]:

```
rawdata.head()
```

In [4]:

```
rawdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 29 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   color                      5024 non-null   object 
 1   director_name              4939 non-null   object 
 2   num_critic_for_reviews     4993 non-null   float64
 3   duration                   5028 non-null   float64
 4   director_facebook_likes    4939 non-null   float64
 5   actor_3_facebook_likes     5020 non-null   float64
 6   actor_2_name               5030 non-null   object 
 7   actor_1_facebook_likes     5036 non-null   float64
 8   gross                      4159 non-null   float64
 9   genres                     5043 non-null   object 
 10  actor_1_name               5036 non-null   object 
 11  movie_title                5043 non-null   object 
 12  num_voted_users            5043 non-null   int64  
 13  cast_total_facebook_likes  5043 non-null   int64  
 14  actor_3_name               5020 non-null   object 
 15  facenumber_in_poster       5030 non-null   float64
 16  plot_keywords              4890 non-null   object 
 17  movie_imdb_link            5043 non-null   object 
 18  num_user_for_reviews       5022 non-null   float64
 19  language                   5029 non-null   object 
 20  country                    5038 non-null   object 
 21  content_rating             4740 non-null   object 
 22  budget                     4551 non-null   float64
 23  title_year                 4935 non-null   float64
 24  actor_2_facebook_likes     5030 non-null   float64
 25  imdb_score                 5043 non-null   float64
 26  aspect_ratio               4714 non-null   float64
 27  movie_facebook_likes       5043 non-null   int64  
 28  Unnamed: 28               0 non-null      float64
dtypes: float64(14), int64(3), object(12)
memory usage: 1.1+ MB
```

There are many variables in this dataset which account of integers, float values and categorical values. All these varibales are not useful while considering to build a model for determining the IMDB ratings for the movies. These variables need to be analyzed for their usefullness.

In [5]:

```
rawdata.isnull().sum(axis = 0)
```

Out[5]:

```
color                        19
director_name               104
num_critic_for_reviews       50
duration                     15
director_facebook_likes     104
actor_3_facebook_likes       23
actor_2_name                 13
actor_1_facebook_likes        7
gross                       884
genres                        0
actor_1_name                  7
movie_title                   0
num_voted_users               0
cast_total_facebook_likes     0
actor_3_name                 23
facenumber_in_poster         13
plot_keywords               153
movie_imdb_link               0
num_user_for_reviews         21
language                     14
country                       5
content_rating              303
budget                      492
title_year                  108
actor_2_facebook_likes       13
imdb_score                    0
aspect_ratio                329
movie_facebook_likes          0
Unnamed: 28                5043
dtype: int64
```

Finding out the null values in the dataset help us to find anomalies which can be taken care of as they would not help to create good models for continious predictions.

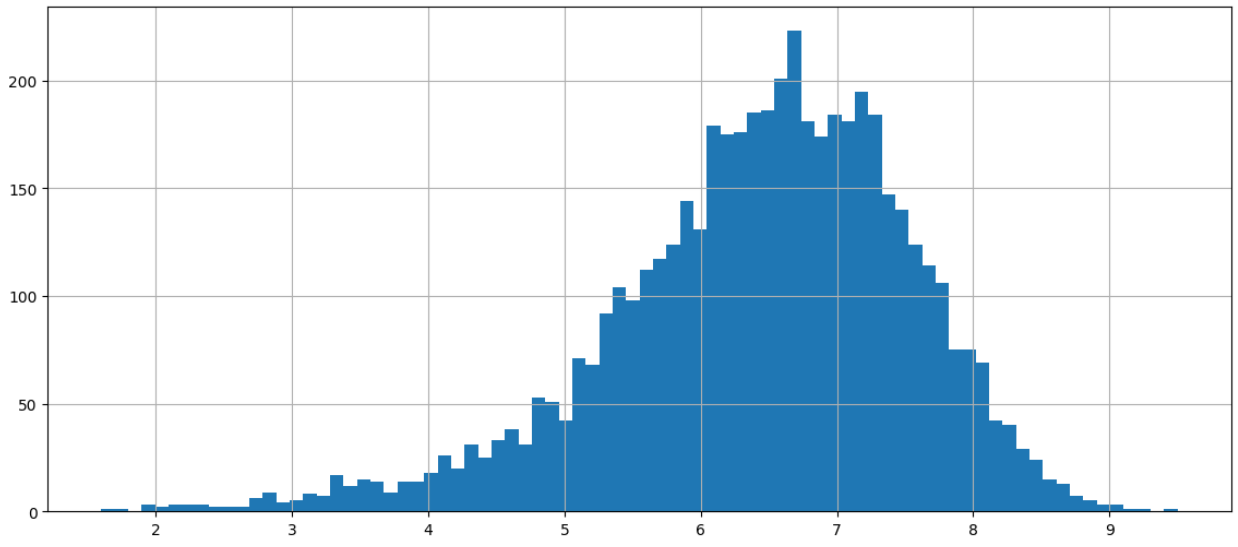
Histograms allow us to figure out if the data is normally distributed and not skewed for particular values. Below is the histogram for the IMDB scores. We can see that there are many movies which have scores between the range of 5 and 8, which resembles a good chunk of data.

In [6]:

```
figg = rawdata['imdb_score'].hist(bins=80, figsize=[14,6])
figg.plot()
```

Out[6]:

```
[]
```



Finding the standard deviation and variance of the scores for the movies

In [7]:

```
rawdata['imdb_score'].std()
```

Out[7]:

```
1.125115865732819
```

In [8]:

```
rawdata['imdb_score'].var()
```

Out[8]:

```
1.2658857113237107
```

Analysis of the data to find various information

Finding the genres and top 10 countries for movies with IMDB scores greater than 7

In [9]:

```
country = rawdata['country'].value_counts()
goodmovies = rawdata.loc[rawdata['imdb_score'] >= 7]
genrecountry = goodmovies.groupby('genres')['country'].count()
ascgenre = genrecountry.sort_values(ascending=False)
top10country = country[:10]
```

In [10]:

```
ascgenre
```

Out[10]:

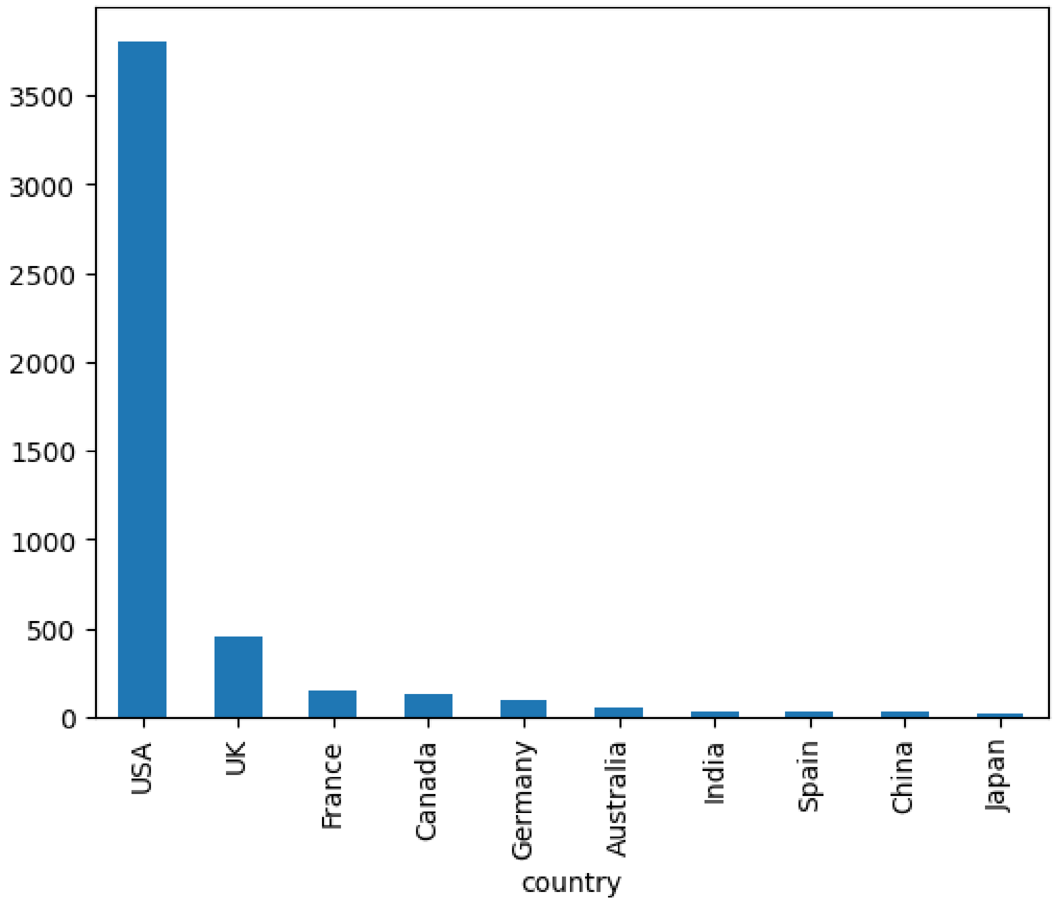
```
genres
Drama                          132
Drama|Romance                   82
Comedy|Drama                    73
Comedy|Drama|Romance            59
Crime|Drama|Thriller            47
```

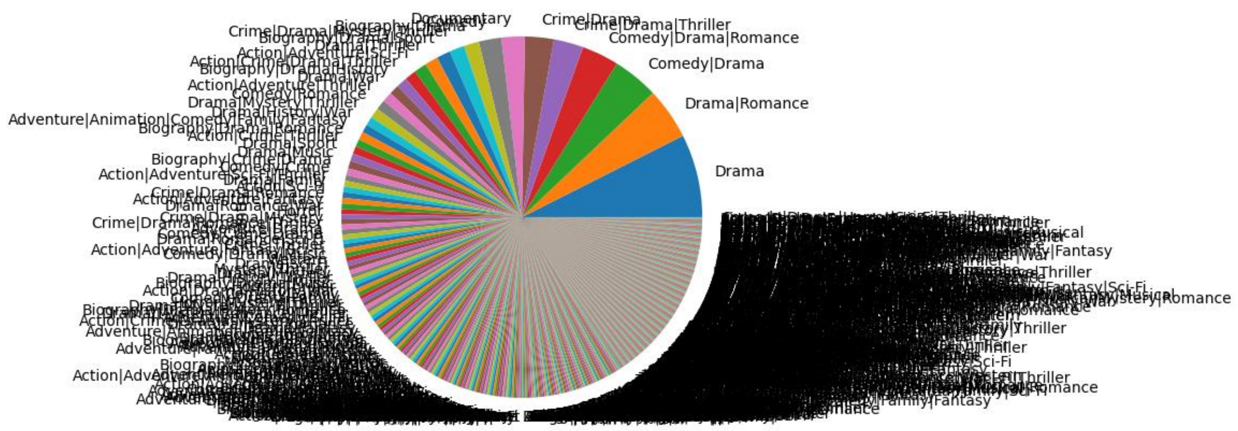
```
                                    ...
Action|Drama|Fantasy|Romance             1
Biography|Documentary|Sport              1
Action|Drama|Fantasy|Sci-Fi              1
Comedy|Drama|Horror|Sci-Fi|Thriller      1
Adventure|Drama|Fantasy|Mystery          0
Name: country, Length: 501, dtype: int64
```

In [11]:

```
top10country.plot(kind = 'bar')
plt.show()
fig1, ax1 = plt.subplots()
ascgenre.plot(kind = 'pie')
labels = 'Action', 'Adventure', 'Drama', 'Animation', 'Comedy', 'Mystery', 'Crime', 'Biography', 'Fantasy',
'Documentary', 'Sci-Fi', 'Horror', 'Romance', 'Family', 'Western', 'Musical'

ax1.axis('equal')
plt.show()
```





From the above charts we can figure out that USA produced the most films based of the data available followed by other countries in the list. The pie chart explains the different genres of films that had IMDB scores of more than 7. Action, adventure and drama lead other genres and are considered more popular for good movie types.

Finding the counts of film ratings for each and every country in the dataset.

In [12]:

```
from pandas import DataFrame
countryrating = rawdata.groupby(['country','content_rating']).size()
rating = DataFrame(countryrating)
rating
```

Out[12]:

```
                                           0

     country     content_rating

 Afghanistan              PG-13            1

                              R            3
   Argentina
                          Unrated          1

       Aruba                  R            1

   Australia                  G            2

         ...                ...          ...
```

		0
country	**content_rating**	
USA	**Unrated**	38
	X	12
West Germany	**M**	1
	PG	1
	R	1

176 rows × 1 columns

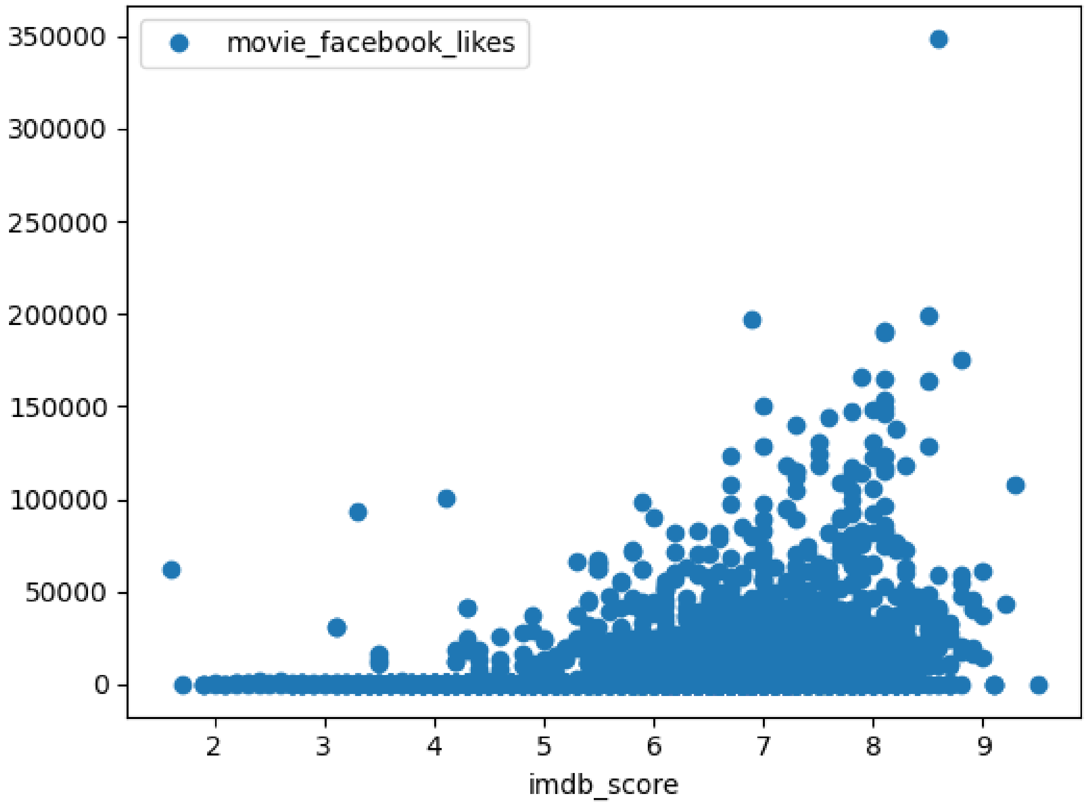
Finding relationship between IMDB score and facebook likes

In [13]:

```
rawdata.plot(x='imdb_score', y='movie_facebook_likes', style='o')
```

Out[13]:

```
<Axes: xlabel='imdb_score'>
```



Processing Data for building machine learning models to predict IMDB ratings which are a series of continious values and not classes

In [14]:

```
newdata = 
rawdata[['color','num_critic_for_reviews','duration','director_facebook_likes','actor_1_facebook_likes','actor_2_face
book_likes','actor_3_facebook_likes','gross','genres','movie_title','num_voted_users','cast_total_facebook_likes','face
number_in_poster','num_user_for_reviews','language','country','content_rating','budget','title_year','aspect_ratio','mo
vie_facebook_likes','imdb_score']]
```

In [15]:

```
newdata[:5]
```

Out[15]:

Null values have to be taken care of before feeding it to the model

In [16]:

```
newdata = newdata.fillna(value=0)
```

In [17]:

```
newdata.isnull().sum(axis = 0)
```

Out[17]:

```
color                        0
num_critic_for_reviews       0
duration                     0
director_facebook_likes      0
actor_1_facebook_likes       0
actor_2_facebook_likes       0
actor_3_facebook_likes       0
gross                        0
genres                       0
movie_title                  0
num_voted_users              0
cast_total_facebook_likes    0
facenumber_in_poster         0
num_user_for_reviews         0
language                     0
country                      0
content_rating               0
budget                       0
title_year                   0
aspect_ratio                 0
movie_facebook_likes         0
imdb_score                   0
dtype: int64
```

Writing a function to convert all the categorical varibale in the dataset to numeric values

In [18]:

```
class MultiColumnLabelEncoder:
    def __init__(self,columns = None):
        self.columns = columns # array of column names to encode

    def fit(self,X,y=None):
        return self # not relevant here

    def transform(self,X):
        '''
        Transforms columns of X specified in self.columns using
        LabelEncoder(). If no columns specified, transforms all
        columns in X.
        '''
        output = X.copy()
        if self.columns is not None:
            for col in self.columns:
                output[col] = LabelEncoder().fit_transform(output[col])
        else:
            for colname,col in output.iteritems():
                output[colname] = LabelEncoder().fit_transform(col)
        return output

    def fit_transform(self,X,y=None):
        return self.fit(X,y).transform(X)
```

In [19]:

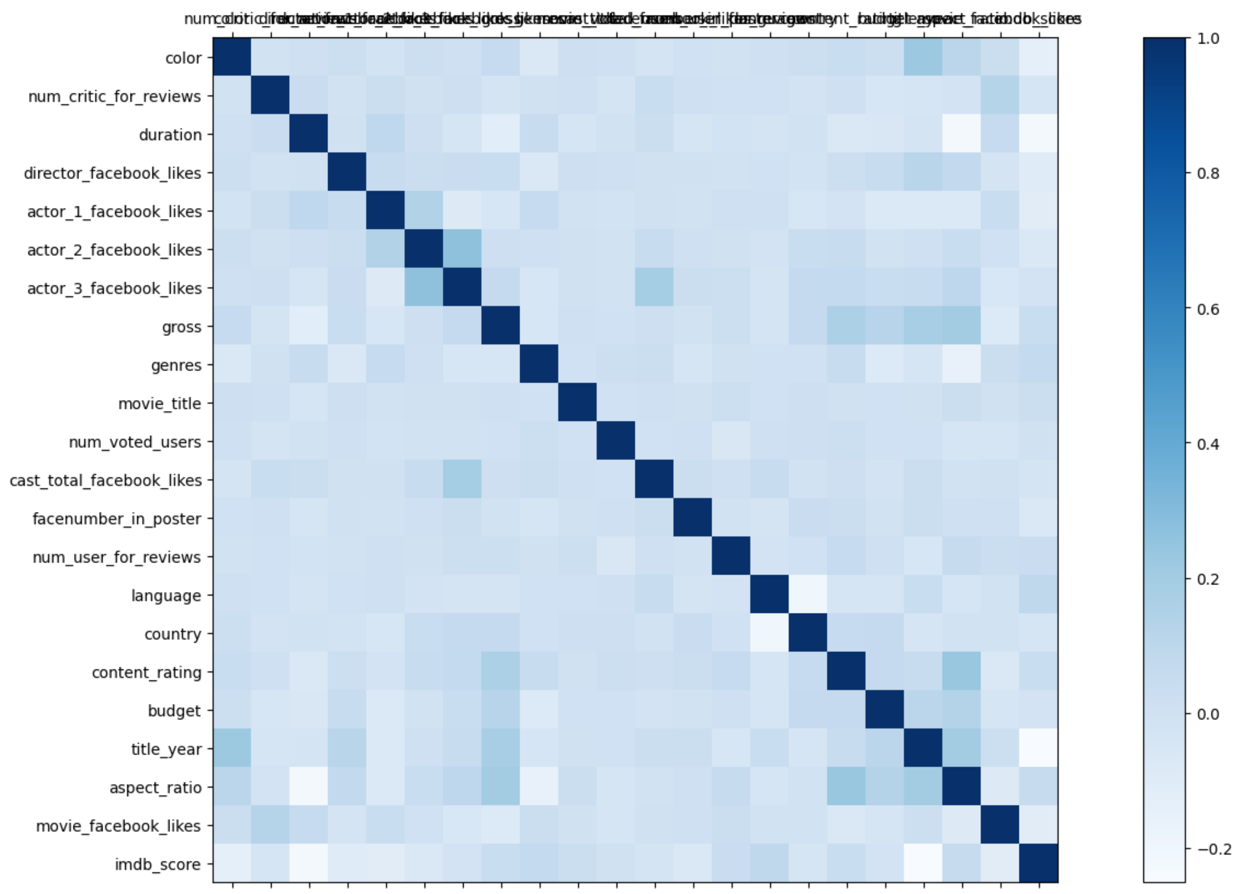
```
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
newdata = MultiColumnLabelEncoder(columns = 
['color','num_critic_for_reviews','duration','director_facebook_likes','actor_1_facebook_likes','actor_2_facebook_likes','actor_3_facebook_likes','gross','genres','movie_title','num_voted_users','cast_total_facebook_likes','facenumber_in_poster','num_user_for_reviews','language','country','content_rating','budget','title_year','aspect_ratio','movie_facebook_likes','imdb_score']).fit_transform((newdata).astype(str))
```

One of the main task herre is to find correlations between the variables in the dataset. If there are variables that are closely related to each other it would be a redundant variable which would be taking up resources during training and might not yield a good model for predicting the scores.

In [20]:

```
from matplotlib import cm as cm
import pandas
import numpy

names = 
['color','num_critic_for_reviews','duration','director_facebook_likes','actor_1_facebook_likes','actor_2_facebook_likes','actor_3_facebook_likes','gross','genres','movie_title','num_voted_users','cast_total_facebook_likes','facenumber_in_poster','num_user_for_reviews','language','country','content_rating','budget','title_year','aspect_ratio','movie_facebook_likes','imdb_score']
correlations = newdata.corr()
# plot correlation matrix
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111)
cax = ax.matshow(correlations,cmap=plt.cm.Blues)
fig.colorbar(cax)
ticks = numpy.arange(0,22,1)
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(names)
ax.set_yticklabels(names)
plt.show()
```



From the above correlation matrix we can see that the selected variable are not closely related to each other and can be used for training the models.

Splitting the data into train and test sets

In [21]:

```
from sklearn.model_selection import train_test_split
newdata = newdata.drop(columns=['imdb_score'])
finaldata = pd.concat([newdata, rawdata['imdb_score']], axis=1)
x = finaldata.loc[:, finaldata.columns != 'imdb_score']
y = finaldata['imdb_score']
```

In [22]:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

```
(4034, 21) (4034,)
(1009, 21) (1009,)
```

Building different models to figure out which performs the best for regression

Simple linear regression model

In [23]:

```
from sklearn import linear_model
lm = linear_model.LinearRegression()
```

```
model = lm.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [24]:

```
from sklearn import metrics
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
1.0417184593862583
```

From the above results we can see that the model doesn't perform well to predict the exact scores on the test set. This needs to improved and there are many ways to approcah such problems. One of the problem is dimensionality reduction, where we get only those features which seem important for predicting.

Random forest for extracting features which seem important. Random frest being a ensemble model helps to select features based on selecting from many subsets. This helps to eliminate features which would be biased and give equal importance to all.

In [25]:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(random_state=1, max_depth=100)
rf.fit(X_train, y_train)
```

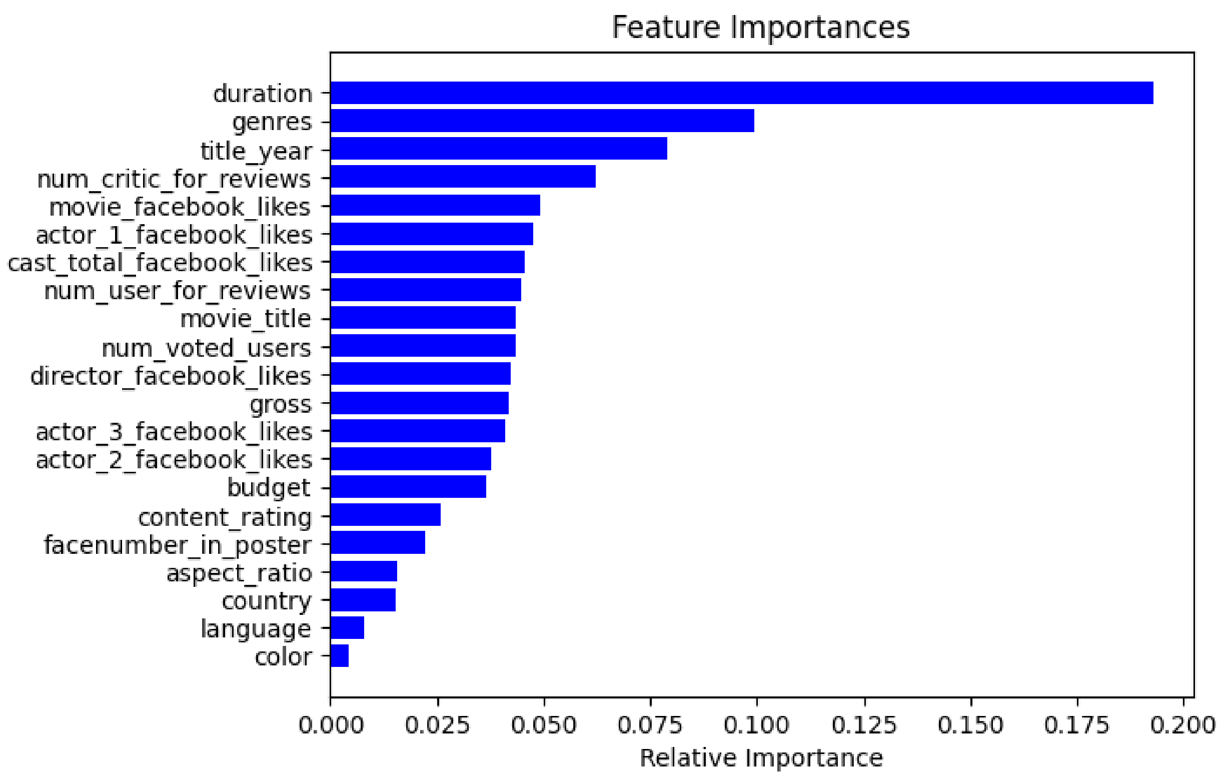
Out[25]:

```
RandomForestRegressor(max_depth=100, random_state=1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [26]:

```
features = x.columns
importances = rf.feature_importances_
indices = np.argsort(importances)[:22]  # top 10 features
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



From the above methods for feature selection we can see that 'duration' has come up as most important feature while determing the features to predict score for the movies. But duration isn't really a feature we can rely on and other variables has to be taken into account.

In [27]:

```
ranfor =
finaldata[['duration','title_year','movie_facebook_likes','num_critic_for_reviews','num_user_for_reviews','genres','ca
st_total_facebook_likes','num_voted_users']]
```

In [28]:

```
X_train, X_test, y_train, y_test = train_test_split(ranfor, y, test_size=0.2)
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

```
(4034, 8) (4034,)
(1009, 8) (1009,)
```

Random forest model

In [29]:

```
from sklearn.ensemble import RandomForestRegressor
rf1 = RandomForestRegressor(random_state=9, max_depth=100)
rf1.fit(X_train, y_train)
```

Out[29]:

```
RandomForestRegressor(max_depth=100, random_state=9)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:

```
y_pred = rf1.predict(X_test)
```

In [31]:

```
from sklearn.metrics import r2_score
coefficient_of_dermination = r2_score(y_test,y_pred)
print(coefficient_of_dermination*100)
```

```
31.967307067121308
```

In [32]:

```
ssr = np.sum((y_pred - y_test)**2)
```

In [33]:

```
print(ssr)
```

```
893.7796450000002
```

From the above result we can see that the model has a low accuracy because it is a regression problema and it's very difficult to predict continious variables

Using PCA for dimensionality reduction instead of Random Forest

In [34]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=4)
pca_result = pca.fit_transform(ranfor)
```

In [35]:

```
X_train, X_test, y_train, y_test = train_test_split(pca_result, y, test_size=0.2)
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

```
(4034, 4) (4034,)
(1009, 4) (1009,)
```

In [36]:

```
from sklearn import linear_model
lm = linear_model.LinearRegression()
model1 = lm.fit(X_train, y_train)
y_pred = model1.predict(X_test)
```

In [37]:

```
from sklearn.ensemble import RandomForestRegressor
rf2 = RandomForestRegressor(random_state=5, max_depth=1000)
rf2.fit(X_train, y_train)
```

Out[37]:

```
RandomForestRegressor(max_depth=1000, random_state=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:

```
y_pred = rf2.predict(X_test)
```

RMSE vs R2

In [39]:

```
from sklearn import metrics
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
1.1272981104093134
```

In [40]:

```
from sklearn.metrics import r2_score
coefficient_of_dermination = r2_score(y_test,y_pred)
print(coefficient_of_dermination*100)
```

```
1.37188201899473
```

Building a XGBoost model for prediction

In [41]:

```
import xgboost as xgb
xg_reg = xgb.XGBRegressor(objective ='reg:linear', colsample_bytree = 0.3, learning_rate = 0.1, max_depth = 5,
alpha = 10, n_estimators = 100)
xg_reg.fit(X_train,y_train)
preds = xg_reg.predict(X_test)
```

```
[23:01:28] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07593ffd91cd9da33-
1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squ
arederror.
```

In [42]:

```
coefficient_of_dermination = r2_score(y_test,preds)
print(coefficient_of_dermination*100)
print(np.sqrt(metrics.mean_squared_error(y_test,preds)))
```

```
1.686629884237012
1.125497922947681
```

We can see that results from XGBoost has slightly increased and still isn't best for predicting the right scores and can be improved using cross validation

Applying Cross validation with XGBoost to build a more robust model than the previous ones

In [43]:

```
data_dmatrix = xgb.DMatrix(data=x,label=y)
```

In [44]:

```
params = {"objective":"reg:linear",'colsample_bytree': 0.3,'learning_rate': 0.1,
                'max_depth': 5, 'alpha': 10}

cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=4,
                    num_boost_round=50,early_stopping_rounds=10,metrics="rmse", as_pandas=True, seed=123)
```

```
[23:01:28] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07593ffd91cd9da33-
1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squ
arederror.
[23:01:28] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07593ffd91cd9da33-
1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squ
arederror.
[23:01:28] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07593ffd91cd9da33-
1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squ
arederror.
[23:01:28] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07593ffd91cd9da33-
1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squ
arederror.
```

In [45]:

```
cv_results.head()
```

Out[45]:

	train-rmse-mean	train-rmse-std	test-rmse-mean	test-rmse-std
0	5.463068	0.014895	5.463214	0.049900
1	4.940707	0.013058	4.940953	0.051221
2	4.472980	0.011846	4.473388	0.052249
3	4.053943	0.010850	4.053918	0.053153
4	3.678052	0.009804	3.677957	0.052937

In [46]:

```
print((cv_results["test-rmse-mean"]).tail(1))
```

```
49    0.944848
Name: test-rmse-mean, dtype: float64
```

In [47]:

```
xg_reg = xgb.train(params=params, dtrain=data_dmatrix, num_boost_round=10)
```

```
[23:01:29] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07593ffd91cd9da33-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squarederror.
```

Conclusion

We can see that the RMSE value for predicting has improved significantly from the previous results thanks to Cross validation. This can further be improved by fine tuning the parameters and improving the model performance. For now the XGBoost model with cross validation helps to predict the rating best.

The other way to improve the predictions is to convert the imdb_scores from continious values to classes. This way the problem gets converted into a classification problem with 10 different classes and classification algorithms can be used to predict the imdb score classes.