

Predicting IMDb Scores Using Machine Learning

Phase 5 Submission Document

Team Members

ABIN BINU JACOB – 961721104002

SELVAK S - 961721104011

ARUN D – 961721104304

VASANTH P.S – 961721104317

SANTHOSH KUMAR R – 961721104314

Project: Predicting IMDb Scores

Topic: Start building the **predicting IMDB scores** model by loading and preprocessing the dataset.

Predicting IMDb Scores



Introduction

Predicting scores on the Internet Movie Database (IMDb) is a fascinating and challenging task that has gained significant attention in the fields of data science and machine learning. IMDb is a popular platform where users rate and review movies and television shows, providing valuable insights into public perception and preference. Predicting IMDb scores involves the use of various techniques, algorithms, and data sources to estimate the rating a movie or TV show is likely to receive based on its attributes and historical data.

In this endeavor, data scientists and researchers leverage a multitude of factors, such as genre, director, cast, plot, budget, release date, and more, to develop predictive models. These models aim to provide a numerical score that closely aligns with the IMDb rating, offering valuable insights to filmmakers, studios, and streaming services for decision-making, marketing, and content creation.

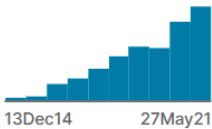
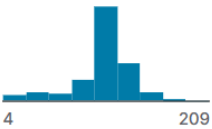
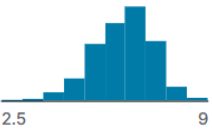
The prediction of IMDb scores has a wide range of applications, including aiding movie studios in making production and marketing decisions, assisting viewers in selecting content to watch, and supporting critics and enthusiasts in evaluating the quality of a film or TV show. With advancements in machine learning and the availability of extensive movie-related datasets, the accuracy of IMDb score predictions continue to improve, making it a compelling area for both research and practical use. Whether you're a data scientist, a movie buff, or an industry professional, the ability to forecast IMDb scores contributes to a deeper understanding of audience preferences and helps shape the ever-evolving world of cinema and entertainment.

Data Source

A good data source for house price prediction using machine learning should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: (<https://www.kaggle.com/datasets/luisortner/netflix-original-films-imdb-scores>)

Given Data Set:

▲ Title	▲ Genre	📅 Premiere	# Runtime	# IMDB Score	▲ Langu
Title of film	Genre(s) of film	Date the film premiered	Runtime in minutes	IMDB score as of 06/01/21	Language 06/01/21
584 unique values	Documentary 27% Drama 13% Other (348) 60%				English Hindi Other (15
Enter the Anime	Documentary	August 5, 2019	58	2.5	English
Dark Forces	Thriller	August 21, 2020	81	2.6	Spanish
The App	Science fiction/Drama	December 26, 2019	79	2.6	Italian
The Open House	Horror thriller	January 19, 2018	94	3.2	English
Kaali Khuhi	Mystery	October 30, 2020	90	3.4	Hindi
Drive	Action	November 1, 2019	147	3.5	Hindi
Leyla Everlasting	Comedy	December 4, 2020	112	3.7	Turkish
The Last Days of American Crime	Heist film/Thriller	June 5, 2020	149	3.7	English
Paradox	Musical/Western/Fant	March 23, 2018	73	3.9	English

Here's a list of tools and software commonly used in the process:

1. Programming Language:

Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy, pandas, scikit-learn, and more.

2. Integrated Development Environment (IDE):

IDEs like PyCharm, VSCode, and RStudio provide a user-friendly environment for writing and debugging code.

3. Python:

Python is the go-to programming language for data analysis and machine learning. Many libraries and frameworks are available to work with data and build predictive models, including NumPy, Pandas, Scikit-Learn, and TensorFlow.

4. Jupyter Notebook:

Jupyter Notebook is an interactive environment widely used for data exploration, analysis, and model development. It allows for easy experimentation and documentation.

5. R:

R is another programming language used for statistical analysis and data visualization. It has a strong community of users in the data science field.

6. SQL Databases:

Databases like MySQL, PostgreSQL, or SQLite are often used to store and manage movie-related data, such as cast and crew information, budget details, and release dates.

7. Web Scraping Tools:

Tools like BeautifulSoup and Scrapy are used to collect data from IMDb's website or other sources. Web scraping helps gather information about movies and TV shows for analysis.

8. Data Visualization Tools:

Tools like Matplotlib, Seaborn, and Plotly are used to create visualizations to better understand the data and the relationships between different variables.

9. Machine Learning Libraries:

Scikit-Learn, XGBoost, LightGBM, and Keras (for deep learning) are commonly used for building predictive models to estimate IMDb scores.

10. Feature Engineering Tools:

Feature engineering is crucial in creating relevant predictors. Python libraries like Feature-engine and Featuretools can be helpful.

11. Text Analysis Libraries:

Natural language processing (NLP) libraries like NLTK and spaCy are used for sentiment analysis and text-based features extraction from reviews and plot summaries.

12. Big Data Tools:

In cases where large datasets are involved, tools like Apache Spark can be used for distributed data processing and machine learning.

13. Cloud Computing Platforms:

Services like Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure provide cloud-based resources for handling large-scale data and machine learning tasks.

14. Machine Learning Platforms:

AutoML platforms like Google AutoML and H2O.ai provide automated machine learning solutions that can be used for model building and optimization.

15. Version Control:

Tools like Git and GitHub/GitLab are essential for tracking changes in code and collaborating with other team members on the project.

16. Containerization:

Docker and container orchestration tools like Kubernetes can be used for packaging and deploying machine learning models.

17. Data Analysis and Visualization Software:

Tools like Tableau, Power BI, or even Excel can be used for additional data analysis and visualization, especially in the context of reporting and sharing results.

18. Statistical Analysis Software:

Software like IBM SPSS or SAS can be used for advanced statistical analysis, especially in more traditional statistical modeling approaches.

The specific tools and software used may vary depending on the project's scale, the preferences of the data scientists, and the available resources. These tools collectively assist in collecting, analyzing, and modeling data to predict IMDb scores accurately.

1. DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT

1. Empathize with Users and Stakeholders:

Understand the needs and expectations of filmmakers, viewers, and industry stakeholders. Conduct interviews, surveys, and focus groups to gather insights into what factors contribute to their perception of movie quality and how they use IMDb scores in decision-making.

2. Define the Problem:

Clearly define the problem statement based on user insights. Identify the specific challenges in predicting IMDb scores, such as the influence of different factors like genre, cast, or release date on the overall rating.

3. Ideate Innovative Solutions:

Brainstorm with your team to generate innovative ideas for improving the accuracy of IMDb score predictions. Consider leveraging advanced data analysis techniques, incorporating sentiment analysis from user reviews, or exploring novel data sources that may provide additional context for predicting scores.

4. Prototype and Experiment:

Develop prototypes of your predictive models and test them with real IMDb data. Experiment with various algorithms and features to understand their impact on the accuracy of predictions. Use A/B testing to compare the performance of different models and select the most effective one.

5. Test and Refine:

Gather feedback from users and stakeholders on the prototype predictions. Use this feedback to refine the model, ensuring it accurately captures the features that contribute to IMDb scores. Incorporate user feedback to fine-tune the model and improve its predictive capabilities.

6. Iterate and Improve:

Continuously iterate on the predictive model based on ongoing feedback and new data. Monitor the model's performance over time and make necessary adjustments to account for changes in user preferences, industry trends, and any modifications to the IMDb rating system.

7. Implement and Monitor:

Deploy the refined predictive model in a real-world setting and closely monitor its performance. Collect data on how well the model predicts IMDb scores compared to actual user ratings. Regularly update the model to adapt to changes in the movie industry and evolving user preferences.

8. Collaborate Across Disciplines:

Foster collaboration between data scientists, movie industry experts, and user experience designers to ensure that the predictive model addresses the diverse needs of various stakeholders. Encourage cross-functional teamwork to incorporate different perspectives and expertise throughout the design and implementation process.

2. DESIGN INTO INNOVATION

Data Overview

We collected a comprehensive dataset of movies, including various features such as genres, cast, director, budget, and release year. The dataset also includes IMDb scores, which serve as our target variable

Data Processing

To prepare the data for modelling, we executed the following preprocessing tasks

- Handling missing data
- Encoding categorical variables
- Normalizing or scaling numeric features
- Addressing outliers, if necessary

Feature Engineering

Feature engineering involved extracting valuable features from the dataset, including

- Genre-based features
- Actor and director-related features
- Budget normalization
- Release year transformation

Model Selection

We experimented with several machine learning models suitable for regression tasks, including

- Linear Regression
- Random Forest
- Gradient Boosting
- Neural Networks

Model Training

The selected model was trained on a portion of the dataset, and hyperparameters were tuned for optimal performance

Model Evaluation

The model's performance was assessed using various evaluation metrics, including

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- R-squared (R2)

These metrics provided insights into the model's predictive accuracy and its ability to estimate IMDb scores effectively

Program:

In[1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from plotnine import *
```

Introduction

To Predict an IMDb Scores using Machine Language

Background

This dataset contains the information about the movies. For a movie to be commercial success, it depends on various factors like director, actors, critic reviews and viewers reaction. IMDb score is one of the important factors to measure the movie's success.

Description of dataset attributes

- Color :- Movie is black or coloured
- Director_name:- Name of the movie director
- num_critic_for_reviews :- No of critics for the movie
- duration:- movie duration in minutes
- director_facebook_likes:-Number of likes for the Director on his
- Facebook Page oactor_3_facebook_likes:- No of likes for the actor 3 on his/her facebook Page
- actor2_name:- name of the actor 2
- actor_1_facebook_likes:- No of likes for the actor 1 on his/her facebook Page
- gross:- Gross earnings of the movie in Dollars ogenres:- Film categorization like 'Animation', 'Comedy',
- 'Romance', 'Horror', 'Sci-Fi', 'Action', 'Family' o actor_1_name:- Name of the actor 1 o movie_title:-Title of the movie
- num_voted_users:-No of people who voted for the
- movie ocast_total_facebook_likes:- Total facebook like for the movie oactor_3_name:- Name of the actor 3
- facenumber_in_poster:- No of actors who featured in the movie poster
- plot_keywords:-Keywords describing the movie plots
- movie_imdb_link:-Link of the movie link
- num_user_for_reviews:- Number of users who gave a review
- language:- Language of the movie ocountry:- Country
- where movie is produced ocontent_rating:- Content rating of the

- movie_obudget:- Budget of the movie in Dollars
- title_year:- The year in which the movie is released
- actor_2_facebook_likes:- facebook likes for the actor 2
- imdb_score:- IMDB score of the movie
- aspect_ratio :- Aspect ratio the movie was made in
- movie_facebook_likes:- Total no of facebook likes for the movie

Case Study

The dataset here gives the massive information about the movies and their IMDB scores respectively. We are going to analyze each and every factor which can influence the IMDB ratings so that we can predict better results. The movie with the higher IMDB score is more successful as compared to the movies with low IMDB score.

Data Processing

In[2]:

```
movie_df=pd.read_csv("/kaggle/input/imdb-5000-movie-dataset/movie_metadata.csv")
```

In[3]:

```
movie_df.head(10)
```

Out[3]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action Advent
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action Advent
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Action Advent
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action Thriller
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN	Documentary
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0	73058679.0	Action Advent
6	Color	Sam Raimi	392.0	156.0	0.0	4000.0	James Franco	24000.0	336530303.0	Action Advent
7	Color	Nathan Greno	324.0	100.0	15.0	284.0	Donna Murphy	799.0	200807262.0	Adventure Ani
8	Color	Joss Whedon	635.0	141.0	0.0	19000.0	Robert Downey Jr.	26000.0	458991599.0	Action Advent
9	Color	David Yates	375.0	153.0	282.0	10000.0	Daniel Radcliffe	25000.0	301956980.0	Adventure Far

In[4]:

```
movie_df.shape
```

Out[4]:

```
(5043, 28)
```

In[5]:

```
movie_df.dtypes
```

Out[5]:

```
color                object
director_name        object
num_critic_for_reviews  float64
duration             float64
director_facebook_likes float64
actor_3_facebook_likes float64
actor_2_name         object
actor_1_facebook_likes float64
gross               float64
genres              object
actor_1_name         object
movie_title          object
num_voted_users      int64
cast_total_facebook_likes int64
actor_3_name         object
facenumber_in_poster float64
plot_keywords        object
movie_imdb_link       object
num_user_for_reviews float64
language            object
country             object
content_rating       object
budget              float64
title_year           float64
actor_2_facebook_likes float64
imdb_score           float64
aspect_ratio         float64
movie_facebook_likes  int64
dtype: object
```

In[6]:

movie_df.describe().T

Out[6]:

	count	mean	std	min	25%	50%	75%	max
num_critic_for_reviews	4993.0	1.401943e+02	1.216017e+02	1.00	50.00	110.00	195.00	8.130000e+02
duration	5028.0	1.072011e+02	2.519744e+01	7.00	93.00	103.00	118.00	5.110000e+02
director_facebook_likes	4939.0	6.865092e+02	2.813329e+03	0.00	7.00	49.00	194.50	2.300000e+04
actor_3_facebook_likes	5020.0	6.450098e+02	1.665042e+03	0.00	133.00	371.50	636.00	2.300000e+04
actor_1_facebook_likes	5036.0	6.560047e+03	1.502076e+04	0.00	614.00	988.00	11000.00	6.400000e+05
gross	4159.0	4.846841e+07	6.845299e+07	162.00	5340987.50	25517500.00	62309437.50	7.605058e+08
num_voted_users	5043.0	8.366816e+04	1.384853e+05	5.00	8593.50	34359.00	96309.00	1.689764e+06
cast_total_facebook_likes	5043.0	9.699064e+03	1.816380e+04	0.00	1411.00	3090.00	13756.50	6.567300e+05
facenumber_in_poster	5030.0	1.371173e+00	2.013576e+00	0.00	0.00	1.00	2.00	4.300000e+01
num_user_for_reviews	5022.0	2.727708e+02	3.779829e+02	1.00	65.00	156.00	326.00	5.060000e+03
budget	4551.0	3.975262e+07	2.061149e+08	218.00	6000000.00	20000000.00	45000000.00	1.221550e+10
title_year	4935.0	2.002471e+03	1.247460e+01	1916.00	1999.00	2005.00	2011.00	2.016000e+03
actor_2_facebook_likes	5030.0	1.651754e+03	4.042439e+03	0.00	281.00	595.00	918.00	1.370000e+05
imdb_score	5043.0	6.442138e+00	1.125116e+00	1.60	5.80	6.60	7.20	9.500000e+00
aspect_ratio	4714.0	2.220403e+00	1.385113e+00	1.18	1.85	2.35	2.35	1.600000e+01
movie_facebook_likes	5043.0	7.525965e+03	1.932045e+04	0.00	0.00	166.00	3000.00	3.490000e+05

In[7]:

```
movie_df.drop('movie_imdb_link', axis=1, inplace=True)
```

In[8]:

```
movie_df["color"].value_counts() movie_df.drop('color',axis=1,inplace=True)
```

In[9]: movie_df.columns

Out[9]:

```
Index(['director_name', 'num_critic_for_reviews', 'duration',  
      'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',  
      'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',  
      'movie_title', 'num_voted_users', 'cast_total_facebook_likes',  
      'actor_3_name', 'facenumber_in_poster', 'plot_keywords',  
      'num_user_for_reviews', 'language', 'country', 'content_rating',  
      'budget', 'title_year', 'actor_2_facebook_likes', 'imdb_score',  
      'aspect_ratio', 'movie_facebook_likes'],  
      dtype='object')
```

In[10]:

```
movie_df.isna().any()
```

Out[10]:

director_name	True
num_critic_for_reviews	True
duration	True
director_facebook_likes	True
actor_3_facebook_likes	True
actor_2_name	True
actor_1_facebook_likes	True
gross	True
genres	False
actor_1_name	True
movie_title	False
num_voted_users	False
cast_total_facebook_likes	False
actor_3_name	True
facenumber_in_poster	True
plot_keywords	True
num_user_for_reviews	True
language	True
country	True
content_rating	True
budget	True
title_year	True
actor_2_facebook_likes	True
imdb_score	False
aspect_ratio	True
movie_facebook_likes	False
dtype:	bool

In[11]:

```
movie_df.isna().sum()
```

Out[11]:

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
num_user_for_reviews	21
language	12
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
dtype:	int64

In[12]:

```
movie_df.dropna(axis=0,subset=['director_name',  
'num_critic_for_reviews','duration','director_facebook_likes','actor_3_facebook_likes','actor_2_name',  
'actor_1_facebook_likes','actor_1_name','actor_3_name','facenumber_in_poster',  
'num_user_for_reviews','language','country','actor_2_facebook_likes','plot_keywords'],inplace=True)
```

In[13]:

```
movie_df.shape
```

Out[13]:

```
(4737, 26)
```

In[14]:

```
movie_df["content_rating"].fillna("R", inplace = True
```

In[15]:

```
movie_df["budget"].fillna(movie_df["budget"].median(),inplace=True)
```

In[17]:

```
movie_df['gross'].fillna(movie_df['gross'].median(),inplace=True)
```

In[18]:

```
movie_df.isna().sum()
```

Out[18]:

```
director_name      0
num_critic_for_reviews  0
duration           0
director_facebook_likes  0
actor_3_facebook_likes  0
actor_2_name       0
actor_1_facebook_likes  0
gross              0
genres             0
actor_1_name       0
movie_title        0
num_voted_users    0
cast_total_facebook_likes  0
actor_3_name       0
facenumber_in_poster  0
plot_keywords      0
num_user_for_reviews  0
language           0
country            0
content_rating     0
budget             0
title_year         0
actor_2_facebook_likes  0
imdb_score         0
aspect_ratio       0
movie_facebook_likes  0
dtype: int64
```

In[19]:

```
movie_df.drop_duplicates(inplace=True) movie_df.shape
```

Out[19]:

```
(4695, 26)
```

In[20]:

```
movie_df["language"].value_counts()
```

Out[20]:

English	4405	Hungarian	1
French	69	Mongolian	1
Spanish	35	Greek	1
Hindi	25	Romanian	1
Mandarin	24	Bosnian	1
German	18	Telugu	1
Japanese	16	Maya	1
Russian	11	Polish	1
Italian	10	Filipino	1
Cantonese	10	Czech	1
Portuguese	8	Dzongkha	1
Korean	8	Kazakh	1
Danish	5	Vietnamese	1
Norwegian	4	Icelandic	1
Swedish	4	Aramaic	1
Hebrew	4		
Dutch	4		
Persian	4		
Arabic	3		
Thai	3		
Indonesian	2		
None	2		
Aboriginal	2		
Dari	2		
Zulu	2		

Name: language, dtype: int64

In[21]:

```
plt.figure(figsize=(40,10))
```

```
sns.countplot(movie_df["language"]) plt.show()
```

In[22]:

```
movie_df.drop('language',axis=1,inplace=True)
```

In[23]:

```
movie_df["Profit"]=movie_df['budget'].sub(movie_df['gross'], axis = 0) movie_df.head(5)
```

Out[23]:

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres
0	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action Adventure Fantasy
1	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action Adventure Fantasy
2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Action Adventure Thriller
3	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action Thriller
5	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0	73058679.0	Action Adventure Sci-Fi

In[24]:

```
movie_df['Profit_Percentage']=(movie_df["Profit"]/movie_df["gross"])*100 movie_df
```

Out[24]:

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres
0	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action Adventure
1	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action Adventure
2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Action Adventure
3	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action Thriller
5	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0	73058679.0	Action Adventure
...
5034	Neill Dela Liana	35.0	80.0	0.0	0.0	Edgar Tancangco	0.0	70071.0	Thriller
5035	Robert Rodriguez	56.0	81.0	0.0	6.0	Peter Marquardt	121.0	2040920.0	Action Crime Drama
5037	Edward Burns	14.0	95.0	0.0	133.0	Caitlin FitzGerald	296.0	4584.0	Comedy Drama
5038	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	26005908.0	Comedy Drama
5042	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	85222.0	Documentary

In[25]:

```
value_counts=movie_df["country"].value_counts() print(value_counts)
```

Out[25]:

USA	3568	Poland	2
UK	420	Taiwan	2
France	149	Iceland	2
Canada	107	Romania	2
Germany	96	Hungary	2
Australia	53	Greece	2
Spain	32	Soviet Union	1
India	27	Slovakia	1
China	24	Finland	1
Japan	21	Official site	1
Italy	20	Turkey	1
Hong Kong	16	Peru	1
New Zealand	14	Libya	1
South Korea	12	Afghanistan	1
Ireland	11	Cambodia	1
Denmark	11	Indonesia	1
Russia	11	Nigeria	1
Mexico	11	Kyrgyzstan	1
South Africa	8	Colombia	1
Brazil	8	New Line	1
Norway	7	Philippines	1
Netherlands	5	Bahamas	1
Sweden	5	Bulgaria	1
Thailand	4	Georgia	1
Iran	4	Aruba	1
Argentina	4	Chile	1
Czech Republic	3		
Switzerland	3		
Belgium	3		
Israel	3		
West Germany	3		

Name: country, dtype: int64

In[26]:

```
vals = value_counts[:2].index
```

```
print (vals)
```

```
movie_df['country'] = movie_df.country.where(movie_df.country.isin(vals), 'other')
```

In[27]:

```
movie_df["country"].value_counts()
```


Out[27]:

```
USA      3568
other     707
UK        420
Name: country, dtype: int64
```

In[28]:

```
movie_df.head(10)
```

Out[28]:

	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres
0	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action Adventure Fantasy
1	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action Adventure Fantasy
2	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Action Adventure Thriller
3	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action Thriller
5	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0	73058679.0	Action Adventure Sci-Fi
6	Sam Raimi	392.0	156.0	0.0	4000.0	James Franco	24000.0	336530303.0	Action Adventure Romance
7	Nathan Greno	324.0	100.0	15.0	284.0	Donna Murphy	799.0	200807262.0	Adventure Animation Comedy
8	Joss Whedon	635.0	141.0	0.0	19000.0	Robert Downey Jr.	26000.0	458991599.0	Action Adventure Sci-Fi
9	David Yates	375.0	153.0	282.0	10000.0	Daniel Radcliffe	25000.0	301956980.0	Adventure Family Fantasy
10	Zack Snyder	673.0	183.0	0.0	2000.0	Lauren Cohan	15000.0	330249062.0	Action Adventure Sci-Fi

3. BUILD LOADING AND PREPROCESSING THE DATASET

Data Processing

To prepare the data for modelling, we executed the following preprocessing tasks

- Handling missing data
- Encoding categorical variables
- Normalizing or scaling numeric features
- Addressing outliers, if necessary

Feature Engineering

Feature engineering involved extracting valuable features from the dataset, including

- Genre-based features
- Actor and director-related features
- Budget normalization
- Release year transformation

Model Selection

We experimented with several machine learning models suitable for

regression tasks, including

- Linear Regression
- Random Forest
- Gradient Boosting
- Neural Networks

Model Training

The selected model was trained on a portion of the dataset, and hyperparameters were tuned for optimal performance

Model Evaluation

The model's performance was assessed using various evaluation metrics, including

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- R-squared (R²)

These metrics provided insights into the model's predictive accuracy and its ability to estimate IMDb scores effectively

Program

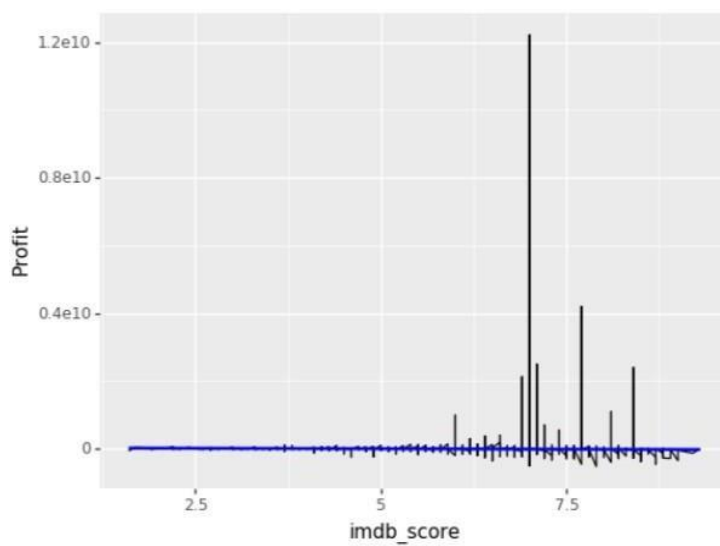
Data Visualization

In[29]:

```
(ggplot(movie_df)
+ aes(x='title_year')
+ geom_bar(size=20)
)
```

In[30]:

```
ggplot(aes(x='imdb_score', y='Profit'), data=movie_df) +\
geom_line() +\ stat_smooth(colour='blue', span=1)
```

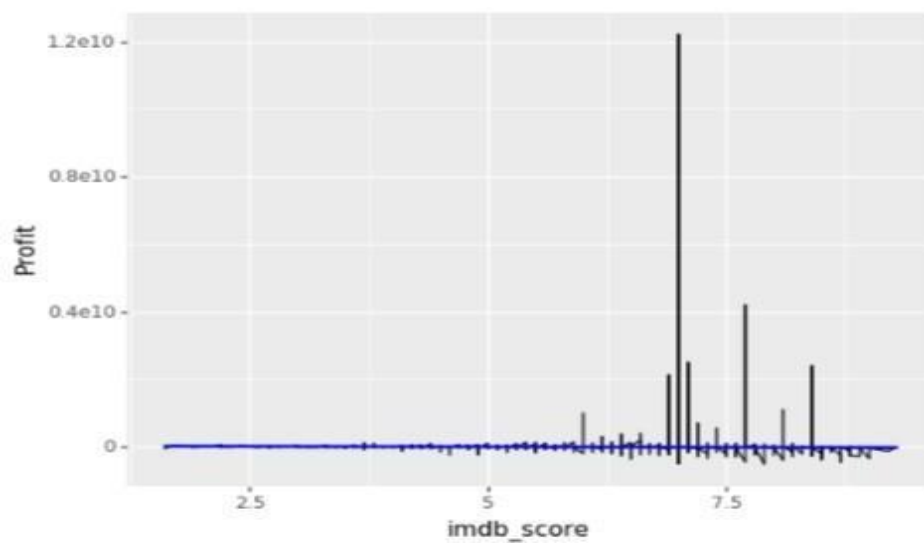


Out[30]:

<ggplot: (8779159653317)>

In[31]:

```
ggplot(aes(x='imdb_score', y='Profit'), data=movie_df) +\n  geom_line() +\n  stat_smooth(colour='blue', span=1)
```

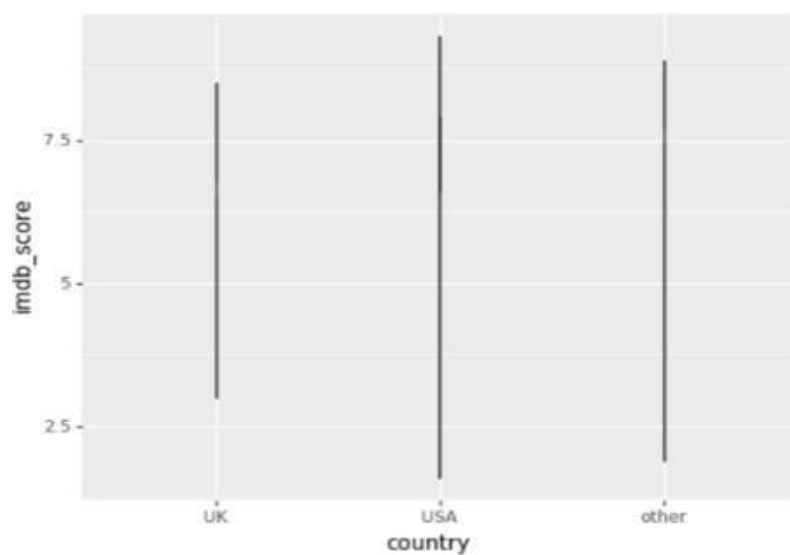


Out[31]:

```
<ggplot: (8779159577103)>
```

In[32]:

```
ggplot(aes(x='country', y='imdb_score'), data=movie_df) +\n  geom_line() +\n  stat_smooth(colour='blue', span=1)
```

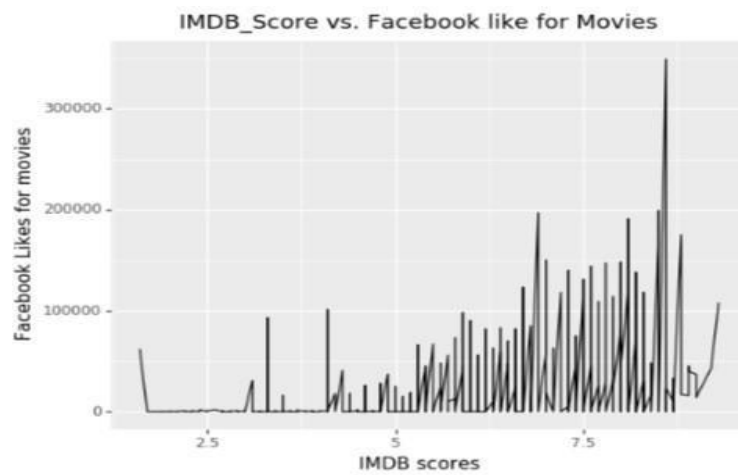


Out[32]:

<ggplot: (-9223363257695236576)>

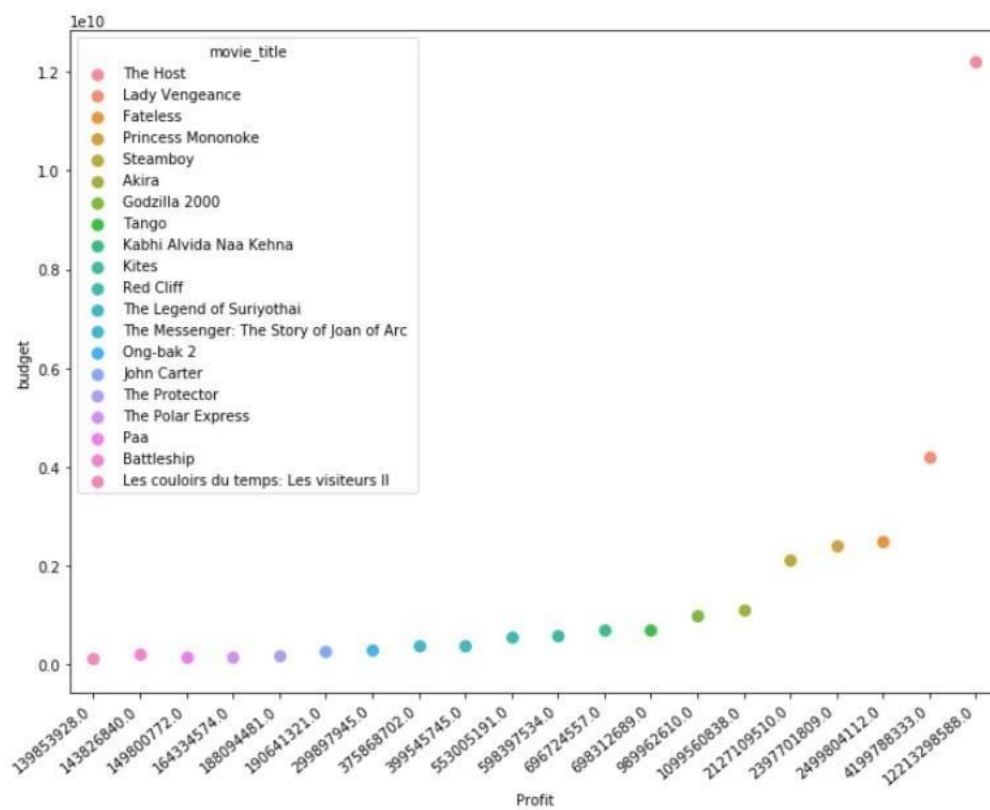
In[33]:

```
(ggplot(movie_df)\n  + aes(x='imdb_score', y='movie_facebook_likes')\n  + geom_line()\n  + labs(title='IMDB_Score vs. Facebook like for Movies', x='IMDB scores', y='Facebook Likes for\n  movies')\n)
```



Out[33]:

<ggplot: (8779159517781)



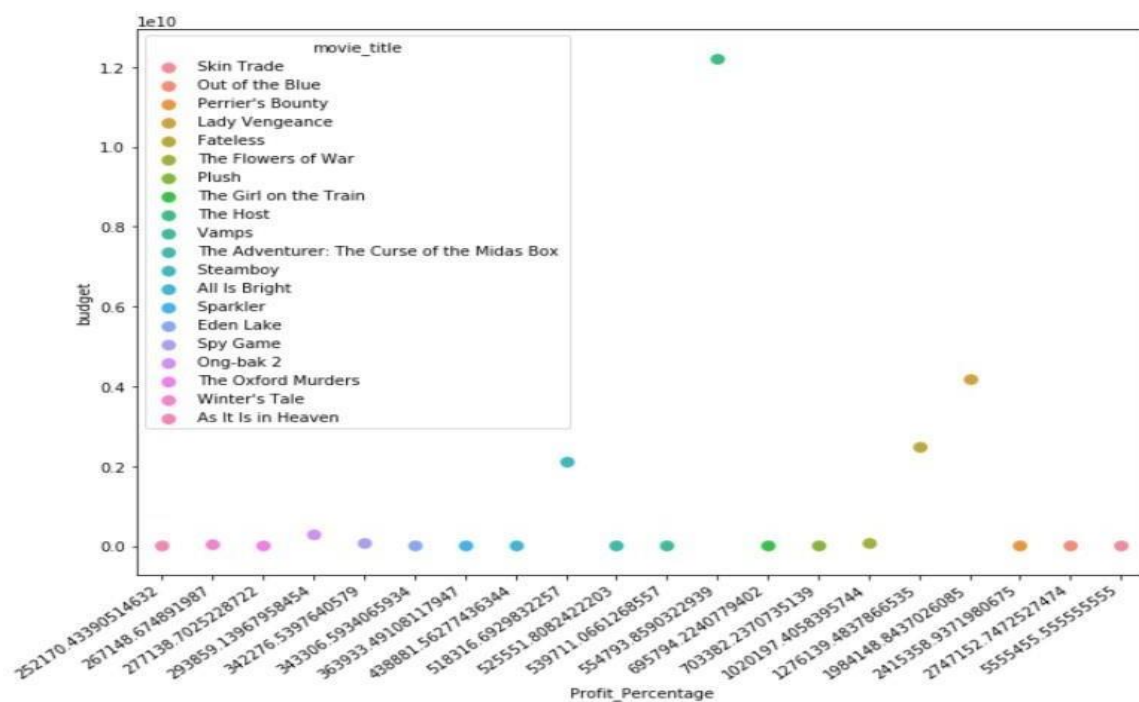
In[34]:

```
plt.figure(figsize=(10,8)) movie_df= movie_df.sort_values(by
='Profit', ascending=False)
movie_df_new=movie_df.head(20)

ax=sns.pointplot(movie_df_new['Profit'], movie_df_new['budget'],
hue=movie_df_new['movie_title'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout() plt.show()
```

In[35]:

```
plt.figure(figsize=(10,8)) movie_df= movie_df.sort_values(by
='Profit_Percentage', ascending=False) movie_df_new=movie_df.head(20)
ax=sns.pointplot(movie_df_new['Profit_Percentage'], movie_df_new['budget'],
hue=movie_df_new['movie_title']) ax.set_xticklabels(ax.get_xticklabels(),
rotation=40, ha="right") plt.tight_layout() plt.show()
```



In[36]:

```
plt.figure(figsize=(10,8))
```

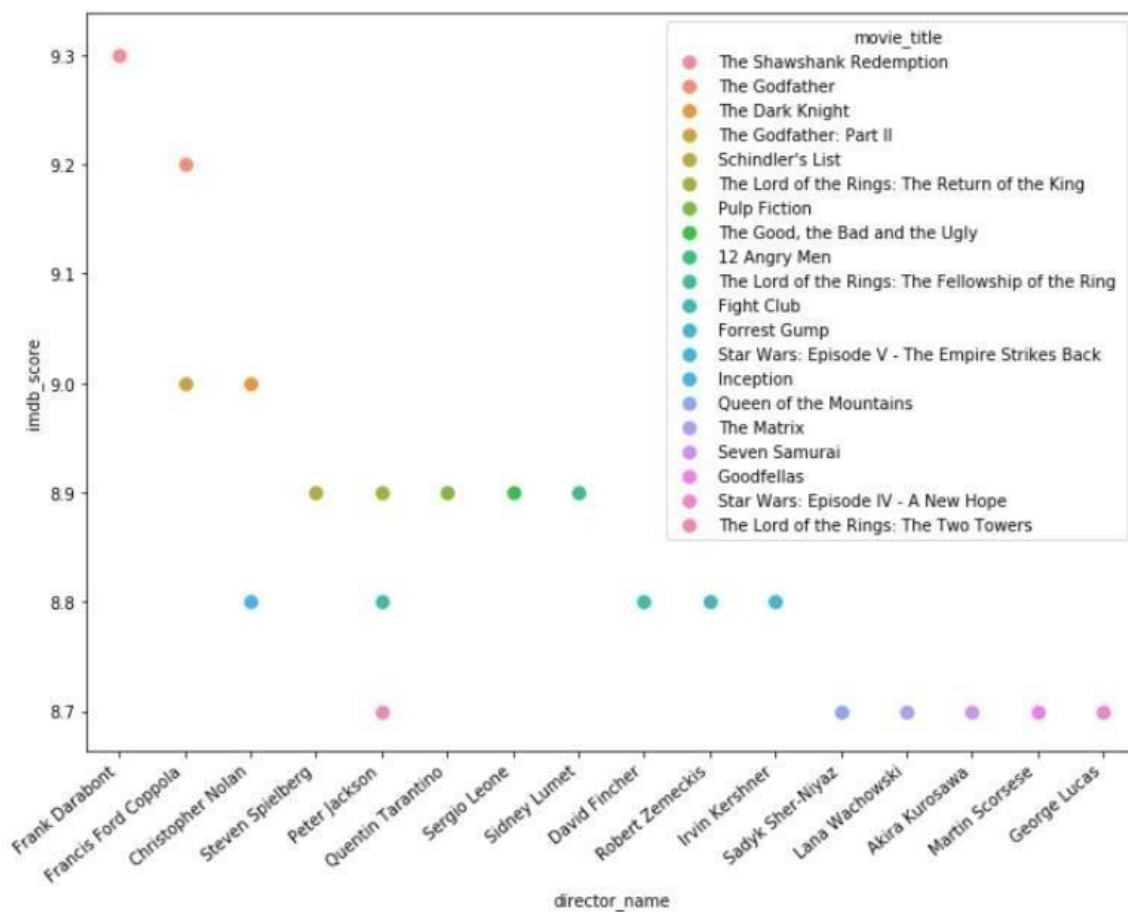
```
movie_df= movie_df.sort_values(by='imdb_score' , ascending=False)
```

```
movie_df_new=movie_df.head(20)
```

```
ax=sns.pointplot(movie_df_new['director_name'], movie_df_new['imdb_score'],
```

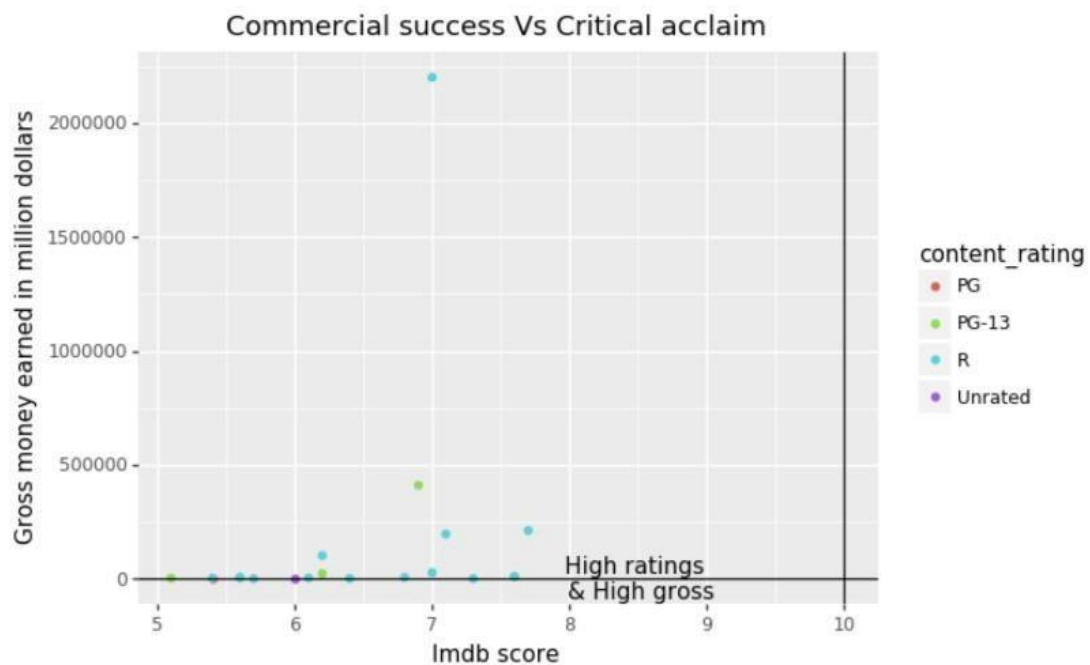
```
hue=movie_df_new['movie_title']) ax.set_xticklabels(ax.get_xticklabels(),
```

```
rotation=40, ha="right") plt.tight_layout() plt.show()
```



In[37]:

```
movie_df= movie_df.sort_values(by='Profit_Percentage' , ascending=False)
movie_df_new=movie_df.head(20)
(ggplot(movie_df_new)
 + aes(x='imdb_score', y='gross',color = "content_rating")
 + geom_point()
 + geom_hline(aes(yintercept = 600)) + geom_vline(aes(xintercept =
10)) + xlab("Imdb score") + ylab("Gross money earned in million
dollars") + ggtitle("Commercial success Vs Critical acclaim") +
annotate("text", x = 8.5, y = 700, label = "High ratings \n & High gross"))
```

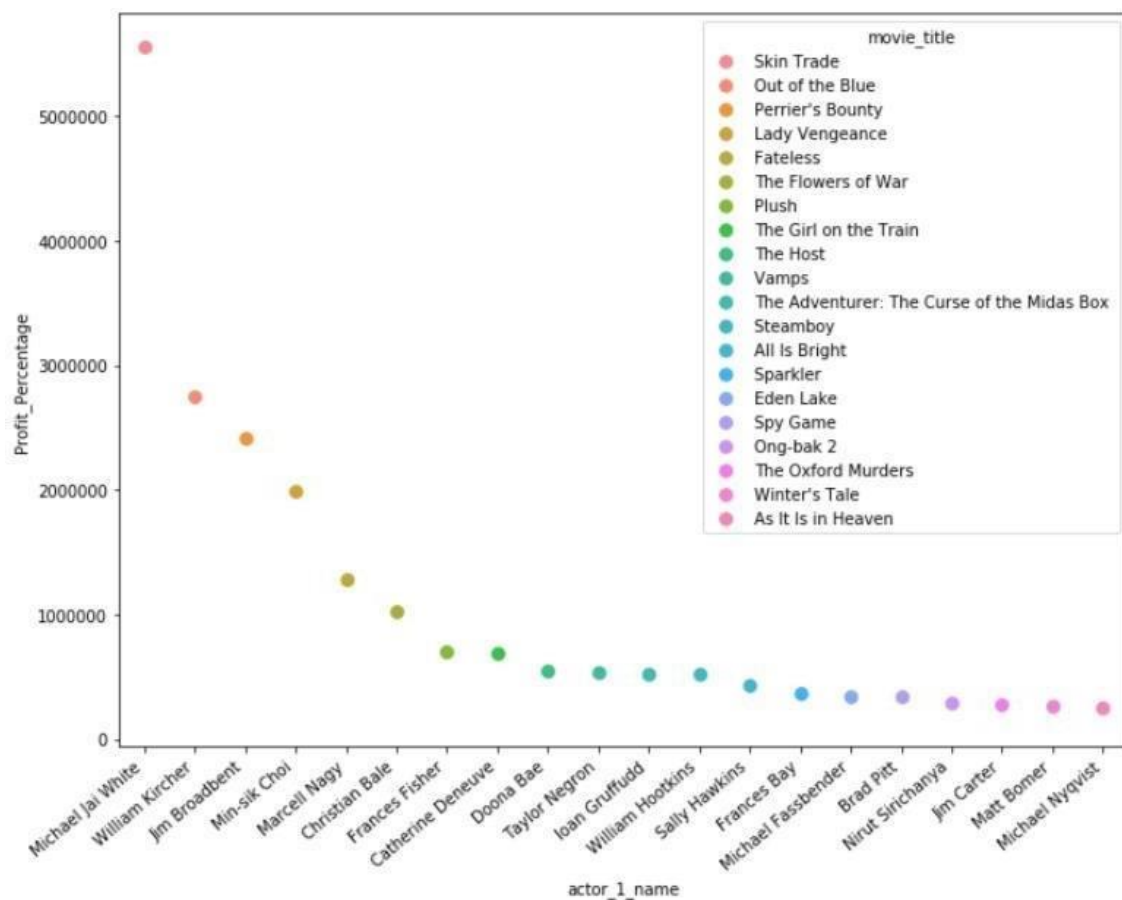


Out[37]:

```
ggplot: (8779159511195)>
```

In[38]:

```
plt.figure(figsize=(10,8)) movie_df= movie_df.sort_values(by
='Profit_Percentage' , ascending=False) movie_df_new=movie_df.head(20)
ax=sns.pointplot(movie_df_new['actor_1_name'], movie_df_new['Profit_Percentage'],
hue=movie_df_new['movie_title']) ax.set_xticklabels(ax.get_xticklabels(), rotation=40,
ha="right") plt.tight_layout() plt.show()
```



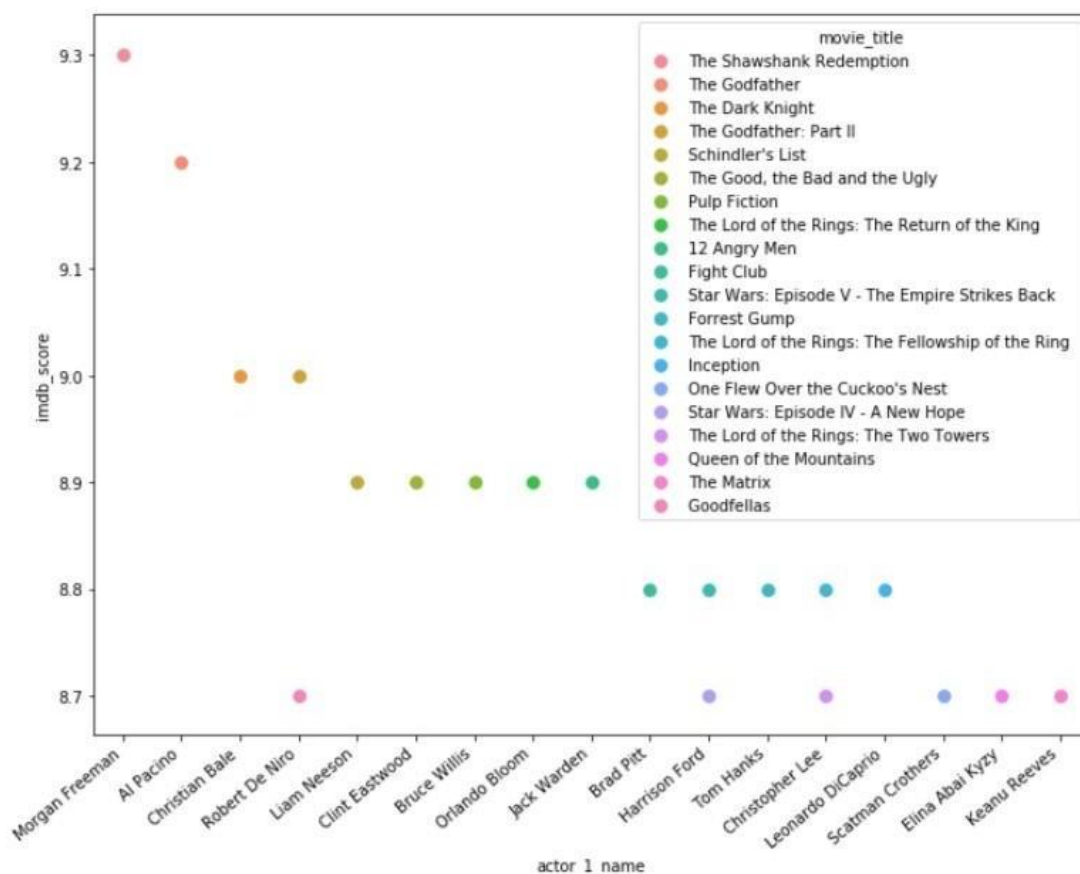
In[39]:

```
plt.figure(figsize=(10,8))

movie_df= movie_df.sort_values(by='imdb_score' , ascending=False)

movie_df_new=movie_df.head(20)

ax=sns.pointplot(movie_df_new['actor_1_name'], movie_df_new['imdb_score'],
hue=movie_df_new['movie_title']) ax.set_xticklabels(ax.get_xticklabels(),
rotation=40, ha="right") plt.tight_layout() plt.show()
```



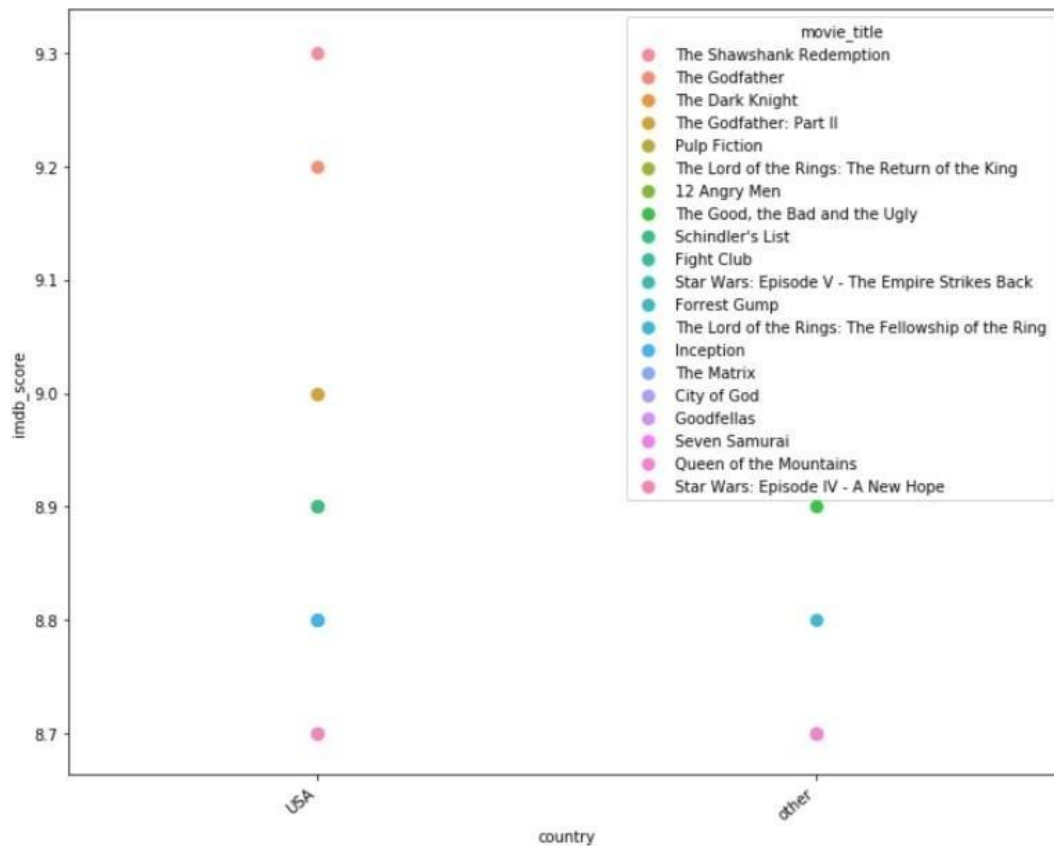
In[40]:

```
plt.figure(figsize=(10,8))

movie_df= movie_df.sort_values(by='imdb_score', ascending=False)

movie_df_new=movie_df.head(20)

ax=sns.pointplot(movie_df_new['country'],
movie_df_new['imdb_score'],          hue=movie_df_new['movie_title'])
ax.set_xticklabels(ax.get_xticklabels(),    rotation=40,    ha="right")
plt.tight_layout() plt.show()
```



4. PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION etc.

Feature Engineering

Feature engineering involved extracting valuable features from the dataset, including

- Genre-based features
 - Actor and director-related features
 - Budget normalization
 - Release year transformation
-
- It describes how the authors extracted additional features from the movie data to use in their prediction models
 - The features include one hot encoding of categorical data, mean rating of directors, writers, cast and production companies, and release month of the movie
 - This also shows how the movie ratings correlate with the extracted features using plots and a correlation matrix.

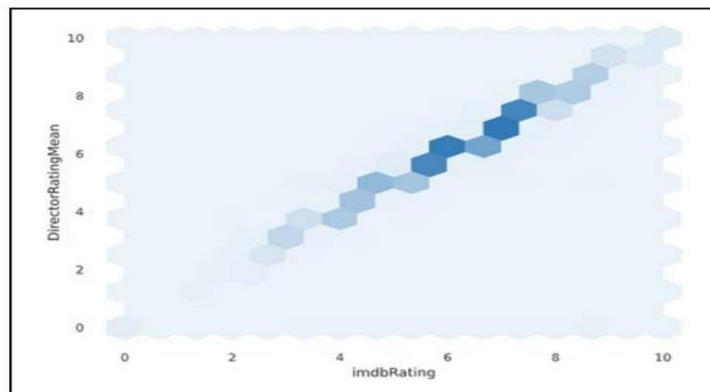


Fig. 1. Interaction of Director Mean Rating with imdbRating.

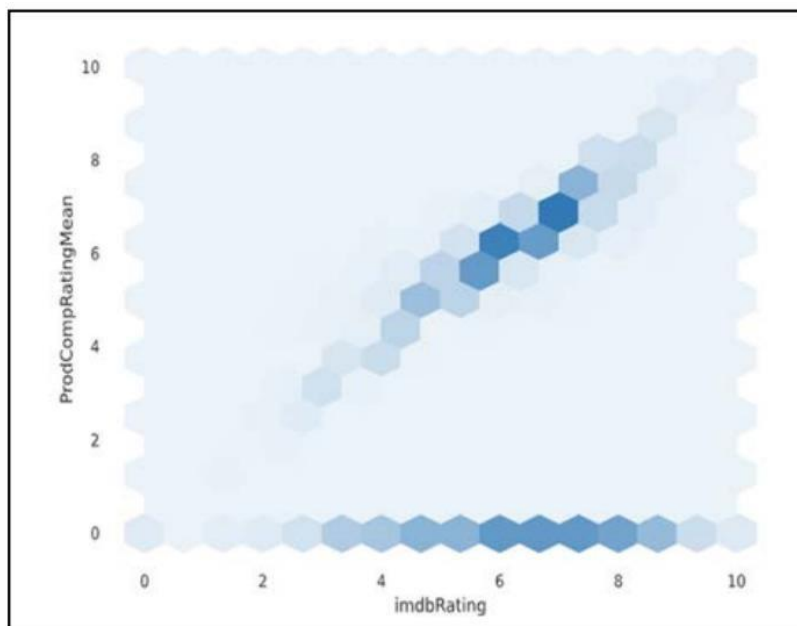


Fig. 2. Interaction of Production Company Ratings with imdb Ratings.

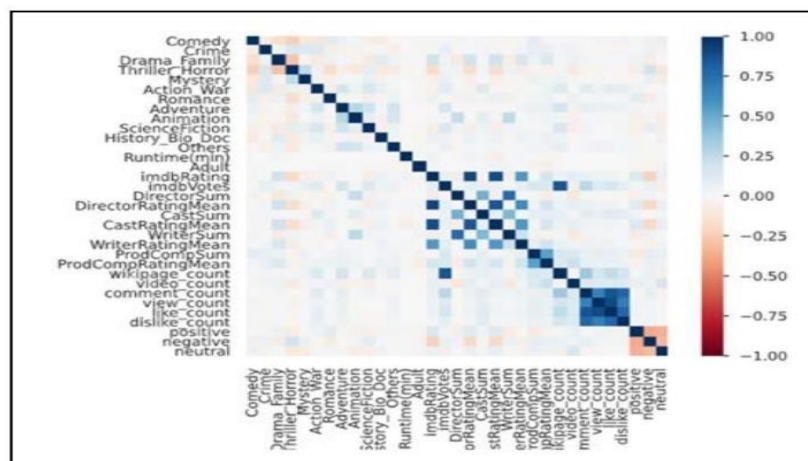


Fig. 3. Correlation Matrix.

Program

Removing Columns with names

In[41]:

```
movie_df.drop('director_name', axis=1, inplace=True)
```

In[42]:

```
movie_df.drop('actor_1_name', axis=1, inplace=True)
```

In[43]:

```
movie_df.drop('actor_2_name', axis=1, inplace=True)
```

In[44]:

```
movie_df.drop('actor_3_name', axis=1, inplace=True)
```

In[45]:

```
movie_df.drop('movie_title', axis=1, inplace=True)
```

In[46]:

```
movie_df.drop('plot_keywords', axis=1, inplace=True)
```


In[47]:

```
movie_df['genres'].value_counts()
```

Out[47]:

Drama	209
Comedy	186
Comedy Drama Romance	182
Comedy Drama	180
Comedy Romance	149
...	
Comedy Drama Horror	1
Mystery Western	1
Animation Comedy Drama Romance	1
Adventure Drama Romance Western	1
Drama War Western	1
Name: genres, Length: 875, dtype: int64	

In[48]:

```
movie_df.drop('genres',axis=1,inplace=True)
```

Remove the Linear dependent variables

In[49]:

```
movie_df.drop('Profit',axis=1,inplace=True)
```

In[50]:

```
movie_df.drop('Profit_Percentage',axis=1,inplace=True)
```

In[51]:

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
corr = movie_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

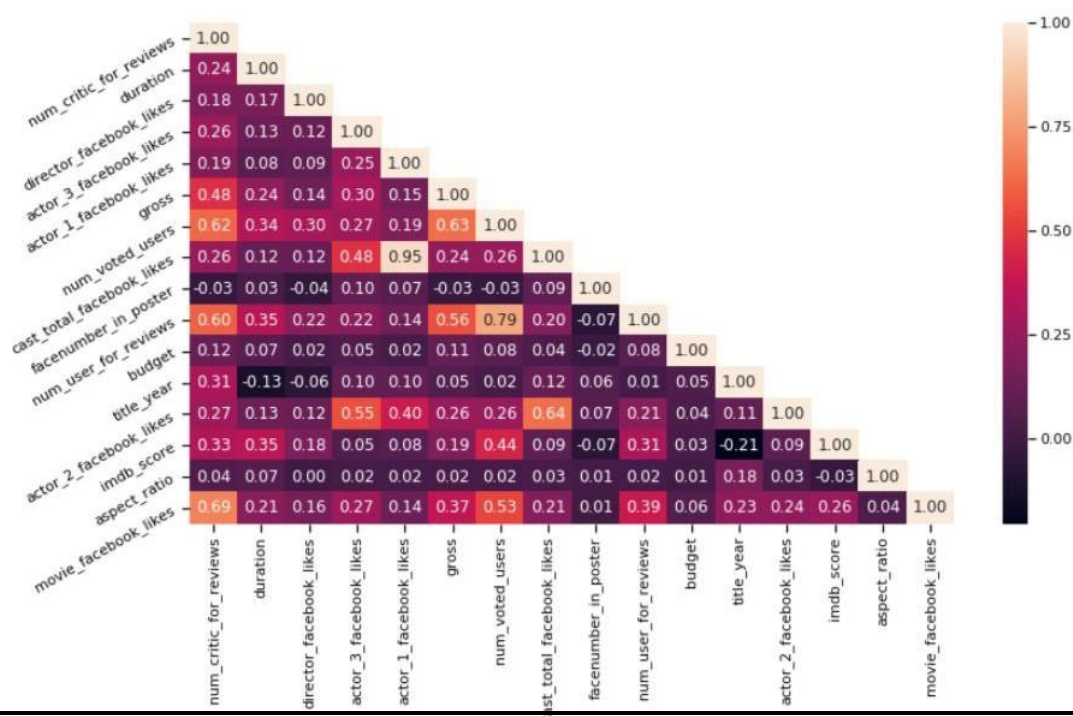
```
mask = np.zeros_like(corr)
```

```
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)
```



In[52]:

```
movie_df['Other_actor_facebbok_likes']=movie_df["actor_2_facebook_likes"] +  
movie_df['actor_3_facebook_likes']
```

In[53]:

```
movie_df.drop('actor_2_facebook_likes',axis=1,inplace=True)
```

In[54]:

```
movie_df.drop('actor_3_facebook_likes',axis=1,inplace=True)
```

In[55]:

```
movie_df.drop('cast_total_facebook_likes',axis=1,inplace=True)
```

In[56]:

```
movie_df['critic_review_ratio']=movie_df['num_critic_for_reviews']/movie_df['num_user_for_reviews']
```

In[57]:

```
movie_df.drop('num_critic_for_reviews',axis=1,inplace=True)  
movie_df.drop('num_user_for_reviews',axis=1,inplace=True)
```

In[58]:

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
corr = movie_df.corr()
```

```
sns.set_context("notebook", font_scale=1.0, rc={"lines.linewidth": 2.5})
```

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

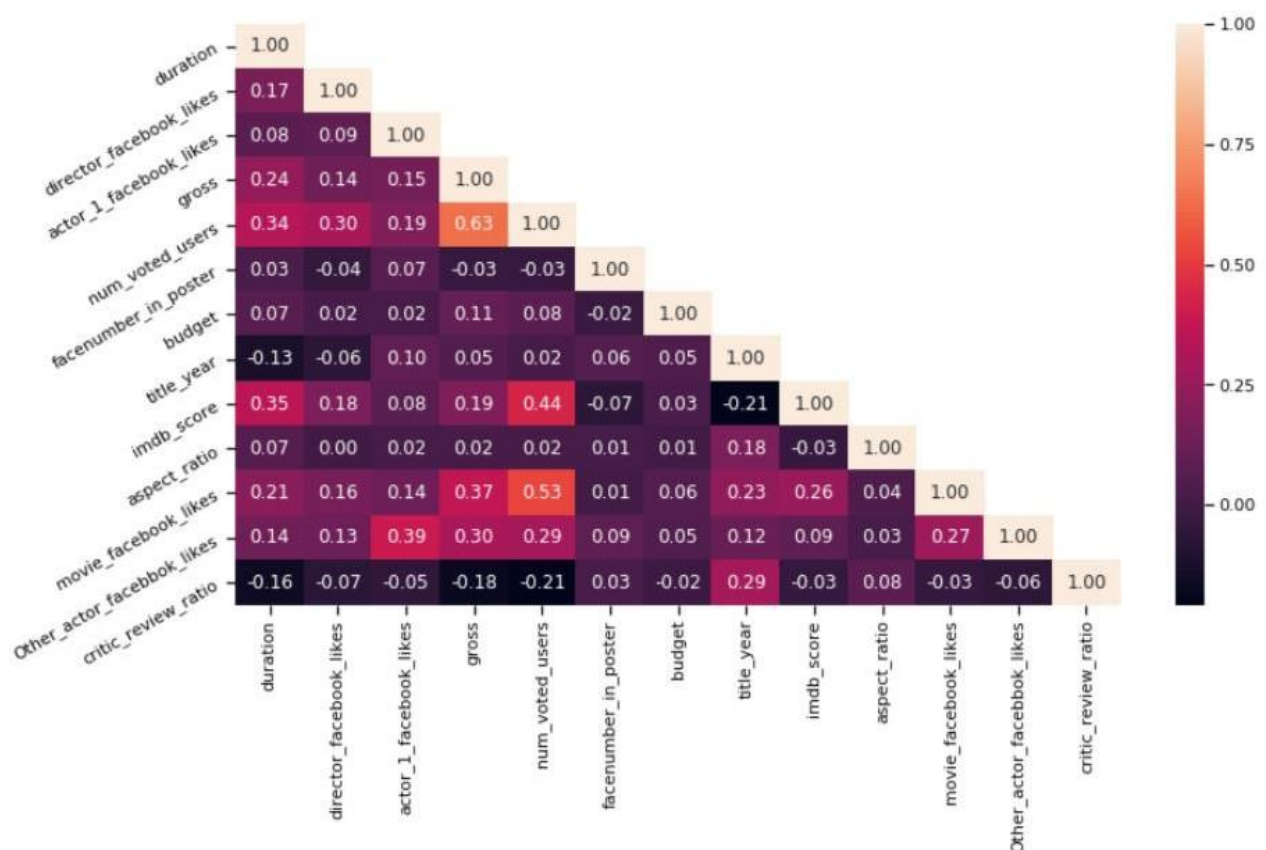
```
mask = np.zeros_like(corr)
```

```
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)
```



In[59]:

```
movie_df["imdb_binned_score"]=pd.cut(movie_df['imdb_score'], bins=[0,4,6,8,10],
right=True, labels=False)+1
```

In[60]:

```
movie_df.drop('imdb_score',axis=1,inplace=True)
```

In[61]:

```
movie_df.head(5)
```

Out[61]:

	duration	director_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	facenumber_in_poster	country	content_rating	budget	title_year
1937	142.0	0.0	11000.0	28341469.0	1689764	0.0	USA	R	25000000.0	1994.0
3466	175.0	0.0	14000.0	134821952.0	1155770	1.0	USA	R	6000000.0	1972.0
66	152.0	22000.0	23000.0	533316061.0	1676169	0.0	USA	PG-13	185000000.0	2008.0
2837	220.0	0.0	22000.0	57300000.0	790926	1.0	USA	R	13000000.0	1974.0
3355	178.0	16000.0	13000.0	107930000.0	1324680	1.0	USA	R	8000000.0	1994.0

Model Selection

We experimented with several machine learning models suitable for regression tasks, including

- Linear Regression
- Random Forest
- Gradient Boosting

- Neural Networks
- RandomForestRegressor – n_estimators, max_depth, min_samples_split, min_samples_leaf, random_state.
- XGBRegressor – n_estimators, learning rate, objective, reg_lambda, reg_alpha.
- The linear regression as a supervised machine learning algorithm that predicts a dependent variable based on one or more independent features.
- It distinguishes between univariate and multivariate linear regression based on the number of independent features.
- Goal of linear regression is to find the best linear equation that represents the relationship between the dependent and independent variables.
- The slope of the line as an indicator of how much the dependent variable changes for a unit change in the independent variable.

Program

Classification Model Selection

Logistic Regression

In[66]:

```
from sklearn.linear_model import LogisticRegression  
logit =LogisticRegression()  
logit.fit(X_train,np.ravel(y_train,order='C'))  
y_pred=logit.predict(X_test)
```

In[67]:

```
from sklearn import metrics  
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(cnf_matrix)  
  
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Out[67]:

```
[[ 0  24  22  0]  
 [ 0  93 285  0]  
 [ 0  67 851  6]  
 [ 0   1  28 32]]  
Accuracy: 0.6926898509581263
```

KNN

In[68]:

```
from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(n_neighbors=22)  
knn.fit(X_train, np.ravel(y_train,order='C'))  
knnpred = knn.predict(X_test)  
cnf_matrix = metrics.confusion_matrix(y_test, knnpred)  
print(cnf_matrix)  
print("Accuracy:",metrics.accuracy_score(y_test, knnpred))
```

Out[68]:

```
[[ 0  24  22  0]
 [ 0 155 223  0]
 [ 0 153 771  0]
 [ 0   2  47 12]]
Accuracy: 0.6657203690560681
```

SVC

In[69]:

```
from sklearn.svm import SVC
svc= SVC(kernel = 'sigmoid')
svc.fit(X_train, np.ravel(y_train,order='C'))
svcpred = svc.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, svcpred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, svcpred))
```

Out[69]:

```
[[ 1  36   9  0]
 [ 2 157 219  0]
 [ 4 175 730 15]
 [ 0   8  33 20]]
Accuracy: 0.6444286728176012
```


Naive Bayes

In[70]:

```
from sklearn.naive_bayes import GaussianNB
gaussiannb= GaussianNB()
gaussiannb.fit(X_train, np.ravel(y_train,order='C'))
gaussiannbpred = gaussiannb.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, gaussiannbpred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, gaussiannbpred))
```

Out[70]:

```
[[ 42   0   1   3]
 [326   1   1  50]
 [530   0   5 389]
 [   8   1   0  52]]
Accuracy: 0.07097232079488999
```

Decision Tree

In[71]:

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(criterion='gini')
dtree.fit(X_train, np.ravel(y_train,order='C'))
```

```
dtreepred = dtree.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, dtreepred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, dtreepred))
```

Out[71]:

```
[[  5  24  17   0]
 [ 24 194 160   0]
 [ 15 192 706  11]
 [  0   2  23  36]]
Accuracy: 0.6678495386799148
```

ADA Boosting

In[72]:

```
from sklearn.ensemble import AdaBoostClassifier
abcl = AdaBoostClassifier(base_estimator=dtree, n_estimators=60)
abcl=abcl.fit(X_train,np.ravel(y_train,order='C'))
abcl_pred=abcl.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, abcl_pred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, abcl_pred))
```

Out[72]:

```
[[ 7 25 14  0]
 [20 194 164  0]
 [13 205 689 17]
 [ 0  1 29 31]]
Accuracy: 0.6536550745209369
```

Random Forest

In[73]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 200)#criterion = entropy,gini
rfc.fit(X_train, np.ravel(y_train,order='C'))
rfcpred = rfc.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, rfcpred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, rfcpred))
```

Out[73]:

```
[[ 2 31 13  0]
 [ 0 177 201  0]
 [ 0  77 846  1]
 [ 0  0 34 27]]
Accuracy: 0.7466288147622427
```

Bagging Classifier

In[74]:

```
new_movie_df=movie_df.pop("imdb_binned_score")
```

In[75]:

```
from sklearn.ensemble import BaggingClassifier  
bgcl = BaggingClassifier(n_estimators=60, max_samples=.7 , oob_score=True)  
  
bgcl = bgcl.fit(movie_df, new_movie_df)  
print(bgcl.oob_score_)
```

Out[75]:

```
0.7429179978700745
```

Gradient Boosting

In[76]:

```
from sklearn.ensemble import GradientBoostingClassifier

gbcl = GradientBoostingClassifier(n_estimators = 50, learning_rate = 0.09, max_depth=5)

gbcl = gbcl.fit(X_train,np.ravel(y_train,order='C'))

test_pred = gbcl.predict(X_test)

cnf_matrix = metrics.confusion_matrix(y_test, test_pred)

print(cnf_matrix)

print("Accuracy:",metrics.accuracy_score(y_test, test_pred))
```

Out[76]:

```
[[ 1  38  7  0]
 [ 0 211 167  0]
 [ 2 100 817  5]
 [ 1   0  30 30]]
Accuracy: 0.751596877217885
```

XG Boosting

In[77]:

```
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train, np.ravel(y_train,order='C'))
xgbprd = xgb.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, xgbprd)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, xgbprd))
```

Out[77]:

```
[[ 1  36  9  0]
 [ 2 198 178  0]
 [ 1 101 818  4]
 [ 0   2  26 33]]
Accuracy: 0.7452093683463449
```

Model Training

The selected model was trained on a portion of the dataset, and hyperparameters were tuned for optimal performance

- The data collection procedure and the model implementation for predicting movie ratings using ensemble learning algorithms.
- It explains how the data was obtained from different sources such as IMDb, Wikipedia, and YouTube, and how it was pre-processed and transformed to extract relevant features.
- The use of two ensemble learning algorithms: Random Forest and XGBoost, and how they were implemented using Scikit-learn library in Python.
- The performance metrics used to evaluate the models, such as cross-validation score, mean squared error, root mean squared error, and mean absolute error.

Program

Splitting the Data into Training and test Data

In[64]:

```
X=pd.DataFrame(columns=['duration','director_facebook_likes','actor_1_facebook_likes','gross','num_voted_users','facenumber_in_poster','budget','title_year','aspect_ratio','movie_facebook_likes','Other_actor_facebbok_likes','critic_review_ratio','country_USA','country_other','content_rating_G','content_rating_GP','content_rating_M','content_rating_NC-17','content_rating_Not Rated','content_rating_PG','content_rating_PG-
```

```
13','content_rating_Passed','content_rating_R','content_rating_TV-14','content_rating_TV-  
G','content_rating_TV-PG','content_rating_Unrated','content_rating_X'],data=movie_df)  
  
y=pd.DataFrame(columns=['imdb_binned_score'],data=movie_df)  
  
from sklearn.model_selection import train_test_split  
  
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=100)
```

In[65]:

```
from sklearn.preprocessing import StandardScaler  
  
sc_X = StandardScaler()  
  
X_train = sc_X.fit_transform(X_train)  
  
X_test = sc_X.transform(X_test)
```

Model Evaluation

The model's performance was assessed using various evaluation metrics, including

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- R-squared (R2)
- Mean square error is a way of measuring how accurate a statistical model or prediction is. It is calculated by taking the average of the squared differences between the actual and predicted values.
- Mean square error can be used for different types of models, such as regression, classification, or neural networks
- Root mean square error (RMSE) is a measure of how well a statistical model or prediction fits the observed data.
- RMSE also has some limitations and assumptions that need to be checked before using it. For example, RMSE may not be

appropriate if the data has outliers, non-linear relationships, or non-normal errors.

TABLE IV. PERFORMANCE METRICS (MOVIE METADATA + EXTRACTED FEATURES)

Performance Metric	Random Forest	XGBoost
<i>Cross Validation Score</i>	0.894	0.931
<i>MSE</i>	0.28	0.18
<i>RMSE</i>	0.53	0.42
<i>MAE</i>	0.37	0.22

TABLE V. PERFORMANCE METRICS (MOVIE METADATA + EXTRACTED FEATURES + SOCIAL MEDIA DATA)

Performance Metric	Random Forest	XGBoost
<i>Cross Validation Score</i>	0.907	0.953
<i>MSE</i>	0.16	0.09
<i>RMSE</i>	0.40	0.30
<i>MAE</i>	0.25	0.13

These metrics provided insights into the model's predictive accuracy and its ability to estimate IMDb scores effectively

Model Comparison

In[78]:

```
from sklearn.metrics import classification_report
```

```
print('Logistic Reports\n',classification_report(y_test,y_pred))
```

```

print('KNN Reports\n',classification_report(y_test, knnpred))
print('SVC Reports\n',classification_report(y_test, svcpred))
print('Naive BayesReports\n',classification_report(y_test, gaussiannbpred))
print('Decision Tree Reports\n',classification_report(y_test, dtreepred))
print('Ada Boosting\n',classification_report(y_test, abcl_pred))
print('Random Forests Reports\n',classification_report(y_test, rfcpred))
print('Bagging Clasifier',bgcl.oob_score_)
print('Gradient Boosting',classification_report(y_test, test_pred))
print('XGBoosting\n',classification_report(y_test, xgbprd))

```

Out[78]:

Logistic Reports

	precision	recall	f1-score	support
1	0.00	0.00	0.00	46
2	0.50	0.25	0.33	378
3	0.72	0.92	0.81	924
4	0.84	0.52	0.65	61
accuracy			0.69	1409
macro avg	0.52	0.42	0.45	1409
weighted avg	0.64	0.69	0.65	1409

KNN Reports

	precision	recall	f1-score	support
1	0.00	0.00	0.00	46
2	0.46	0.41	0.44	378
3	0.73	0.83	0.78	924
4	1.00	0.20	0.33	61
accuracy			0.67	1409
macro avg	0.55	0.36	0.39	1409
weighted avg	0.64	0.67	0.64	1409

SVC Reports

	precision	recall	f1-score	support
1	0.14	0.02	0.04	46
2	0.42	0.42	0.42	378
3	0.74	0.79	0.76	924
4	0.57	0.33	0.42	61
accuracy			0.64	1409
macro avg	0.47	0.39	0.41	1409
weighted avg	0.62	0.64	0.63	1409

Naive BayesReports

	precision	recall	f1-score	support
1	0.05	0.91	0.09	46
2	0.50	0.00	0.01	378
3	0.71	0.01	0.01	924
4	0.11	0.85	0.19	61
accuracy			0.07	1409
macro avg	0.34	0.44	0.07	1409
weighted avg	0.61	0.07	0.02	1409

Decision Tree Reports

	precision	recall	f1-score	support
1	0.11	0.11	0.11	46
2	0.47	0.51	0.49	378
3	0.78	0.76	0.77	924
4	0.77	0.59	0.67	61
accuracy			0.67	1409
macro avg	0.53	0.49	0.51	1409
weighted avg	0.67	0.67	0.67	1409

Ada Boosting

	precision	recall	f1-score	support
1	0.17	0.15	0.16	46
2	0.46	0.51	0.48	378
3	0.77	0.75	0.76	924
4	0.65	0.51	0.57	61
accuracy			0.65	1409
macro avg	0.51	0.48	0.49	1409
weighted avg	0.66	0.65	0.66	1409

Random Forests Reports

	precision	recall	f1-score	support
1	1.00	0.04	0.08	46
2	0.62	0.47	0.53	378
3	0.77	0.92	0.84	924
4	0.96	0.44	0.61	61
accuracy			0.75	1409
macro avg	0.84	0.47	0.52	1409
weighted avg	0.75	0.75	0.72	1409

Bagging Clasifier 0.7429179978700745

Gradient Boosting	precision	recall	f1-score	support
1	0.25	0.02	0.04	46
2	0.60	0.56	0.58	378
3	0.80	0.88	0.84	924
4	0.86	0.49	0.62	61
accuracy			0.75	1409
macro avg	0.63	0.49	0.52	1409
weighted avg	0.73	0.75	0.74	1409

XGBoosting

	precision	recall	f1-score	support
1	0.25	0.02	0.04	46
2	0.59	0.52	0.55	378
3	0.79	0.89	0.84	924
4	0.89	0.54	0.67	61
accuracy			0.75	1409
macro avg	0.63	0.49	0.53	1409
weighted avg	0.72	0.75	0.73	1409

ADVANTAGES

1. Informing Audience Choices:

Predicting scores can help audiences make informed choices about which movies or shows to watch. By having an idea of how a particular title is likely to be rated, viewers can decide whether it aligns with their preferences and expectations.

2. Guiding Production Decisions:

For filmmakers and production companies, IMDb score predictions can serve as a valuable indicator of potential audience reception. This can guide decisions about marketing strategies, distribution plans, and even the content of future projects.

3. Market Analysis:

IMDb score predictions can provide valuable insights for market analysis and research. By understanding the potential reception of certain types of content, stakeholders in the entertainment industry can make informed decisions about investment, production, and distribution.

4. Enhancing User Experience:

Streaming platforms and movie recommendation systems can utilize IMDb score predictions to enhance user experience by suggesting content that aligns with the user's preferences. This can improve user engagement and satisfaction with the platform.

5. Building Publicity and Hype:

Predicting high scores on IMDb can contribute to building positive publicity and generating hype around a particular title. This can attract more viewers and create a buzz within the entertainment industry, leading to increased viewership and potentially higher revenue.

6. Improving Content Quality:

By analyzing IMDb score predictions, content creators can gather insights into what elements contribute to higher ratings. This information can be used to improve the quality of future content, leading to better-received productions and increased audience satisfaction.

7. Competitive Analysis:

IMDb score predictions can help in conducting competitive analysis within the entertainment industry. By comparing predicted scores of different titles, stakeholders can gain insights into market trends, audience preferences, and the performance of competitors.

8. Investment Decisions:

Investors and stakeholders in the entertainment industry can utilize IMDb score predictions to make more informed investment decisions. Predicted scores can provide valuable information about the potential success and profitability of a particular project, helping investors allocate resources more effectively.

Disadvantages

1. Inaccuracy and Bias:

IMDb score predictions may not always accurately reflect the actual reception of a movie or TV show. Predictions can be influenced by various biases, such as user demographics, early reviews, or promotional campaigns, leading to misleading or inaccurate estimations of the final rating.

2. Limited Evaluation Criteria:

IMDb score predictions often rely on limited evaluation criteria, such as user reviews and ratings, which may not capture the nuances of a production's quality or appeal. Important aspects such as cultural context, artistic merit, and critical acclaim may be overlooked, resulting in an oversimplified assessment of a title's potential success.

3. Impact on Creativity:

Overreliance on IMDb score predictions may influence content creators to prioritize elements that are more likely to boost ratings rather than focusing on creative innovation and originality. This can potentially lead to the production of formulaic or derivative content, limiting the diversity and richness of artistic expression.

4. Manipulation and Fraud:

The system of user reviews and ratings on IMDb can be susceptible to manipulation and fraud, such as fake reviews or rating manipulation through bots or coordinated efforts. This can distort the predicted scores and mislead both audiences and industry stakeholders.

5. Overemphasis on Commercial Success:

IMDb score predictions may contribute to an overemphasis on commercial success over artistic value. Filmmakers and producers might prioritize elements that are more likely to generate higher predicted scores, potentially leading to a focus on profit-driven strategies rather than the creation of meaningful and impactful content.

6. Underrepresentation of Niche Content:

IMDb score predictions might favor mainstream, popular, or commercially successful content, potentially overlooking niche or independent productions that cater to specific audiences or explore unconventional themes. This could contribute to a limited representation of diverse and innovative storytelling within the entertainment industry.

7. Negative Impact on Small Productions:

Inaccurate or low IMDb score predictions can have a detrimental impact on the success of small-scale or independent productions. Despite their potential artistic value, these titles may struggle to attract audiences and secure adequate funding or distribution opportunities due to unfavorable predicted scores.

8. Disregard for Evolving Audience Preferences:

IMDb score predictions may not adequately account for evolving audience preferences and changing cultural trends. As audience tastes and expectations shift over time, relying solely on past ratings and reviews may not accurately predict the reception of new and innovative content.

Benefits

1. Informed Viewing Decisions:

IMDb score predictions can help viewers make informed decisions about what movies or TV shows to watch, allowing them to choose content that aligns with their preferences and tastes.

2. Marketing and Promotion:

Predicting scores can assist filmmakers and studios in creating effective marketing and promotional strategies. Anticipated high ratings can be used to generate buzz and interest in upcoming releases, leading to increased viewership and ticket sales.

3. Financial Planning:

IMDb score predictions provide valuable insights for investors and stakeholders in the entertainment industry, aiding in financial planning and investment decisions. Anticipated high scores can indicate potential commercial success, encouraging investment in promising projects.

4. Content Development:

Predicted IMDb scores can guide content creators in understanding audience preferences and expectations. This knowledge can be used to develop content that is more likely to resonate with viewers, leading to higher ratings and increased audience satisfaction.

5. Industry Competitiveness:

IMDb score predictions contribute to a competitive environment in the entertainment industry, encouraging filmmakers and production companies to strive for higher quality content and better audience reception. This can lead to an overall improvement in the quality of movies and TV shows.

6. User Engagement:

Predicted IMDb scores can enhance user engagement on digital streaming platforms by helping users discover content that is likely to appeal to them. This can lead to increased user satisfaction and retention, ultimately benefiting the platform's viewership and revenue.

7. Talent Recognition:

Anticipated high scores on IMDb can help recognize and promote talented actors, directors, and other industry professionals. This recognition can open up opportunities for these individuals to work on more prominent projects, further elevating the quality of future productions.

8. Industry Feedback:

IMDb score predictions can serve as feedback for filmmakers and studios, highlighting areas for improvement and indicating which aspects of a production are well-received by audiences. This feedback can be used to refine future projects and enhance the overall quality of content.

Conclusion

The conclusion is that Random Forest Algorithm along with the gradient boosting have the accuracy of 74.5 and 75.5 respectively

This analysis compares certain attributes regarding Facebook and IMDB site against the gross revenue of a film. The higher number of Facebook likes from the primary actor and supporting actors plays a significant role in generating revenue from a film. Through both models and the sensitivity analysis, someone can easily see the support in this conclusion. Directors and producers can take this dataset and implement it into their thought process when planning their movie. Movie-goers can use this dataset to make the same predictions once the movie is announced with primary and supporting actors/actresses. It could possibly save movie-goers money when debating on whether to go see a movie or not.

This project has helped me substantially in practicing with running analysis on certain topics and generating a result. It has developed my skill in Excel and XL Miner by using the Missing Data handle, the Reduce Categories handle, the Data Partition handle, the Multiple Linear Regression handle, and a sensitivity analysis. Overall, the effectiveness of this project was very useful for me in the preparation for my career. I can take this project as proof of knowledge in these areas as well as knowing associated terms.