LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CU6051NI Artificial Intelligence

Level 6 – Artificial Intelligence

Assessment Type

Individual 75%

Semester

2024/25 Autumn

Student Name: Rajita Maharjan

London Met ID: 22067335

College ID: np01ai4a220052

Assignment Due Date: Wednesday, January 22, 2025

Assignment Submission Date: Wednesday, January 22, 2025

Submitted To: Alish KC

Word Count (Where Required):

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

22067335 Rajita Maharjan AI Final Submission.docx

Islington College,Nepal

Document Details

Submission ID
trn:oid:::3618:79814900

Submission Date
Jan 22, 2025, 12:17 PM GMT+5:45

Download Date
Jan 22, 2025, 12:18 PM GMT+5:45

File Name
22067335 Rajita Maharjan AI Final Submission.docx

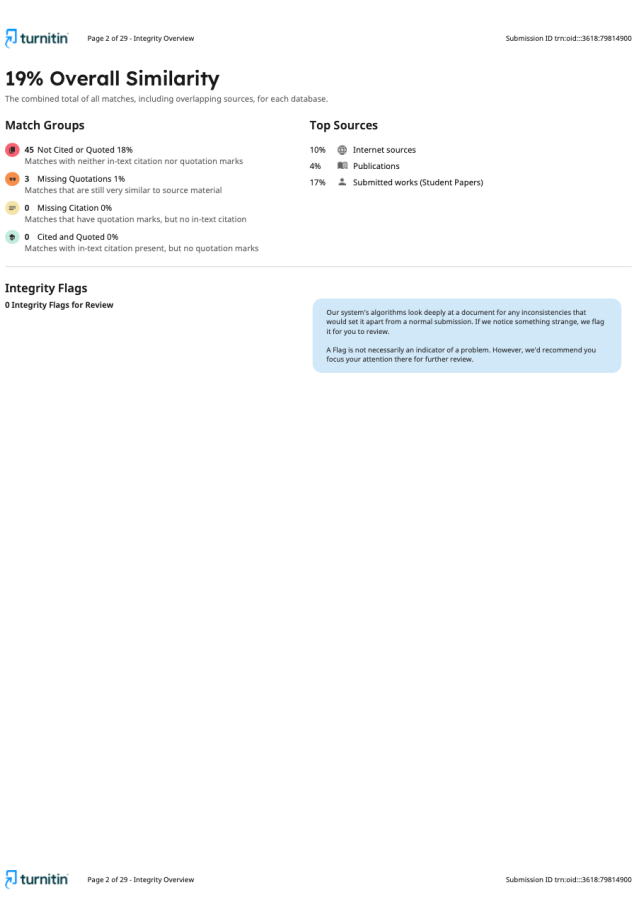
File Size
23.4 KB

24 Pages

3,534 Words

19,673 Characters

CU6051NI Artificial Intelligence



turnitin Page 2 of 29 - Integrity Overview Submission ID trn:oid:::3618:79814900

19% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **45** Not Cited or Quoted 18%
  Matches with neither in-text citation nor quotation marks
- **3** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

10% Internet sources
4% Publications
17% Submitted works (Student Papers)

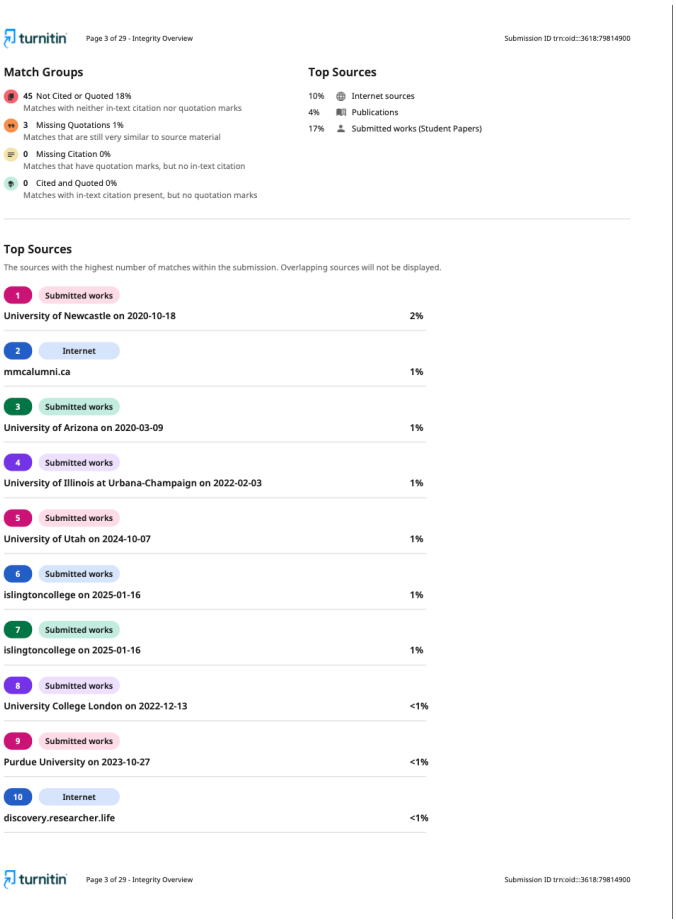
Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

turnitin Page 2 of 29 - Integrity Overview Submission ID trn:oid:::3618:79814900



turnitin Page 3 of 29 - Integrity Overview Submission ID trn:oid:::3618:79814900

Match Groups

- **45** Not Cited or Quoted 18%
  Matches with neither in-text citation nor quotation marks
- **3** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

10% Internet sources
4% Publications
17% Submitted works (Student Papers)

Top Sources

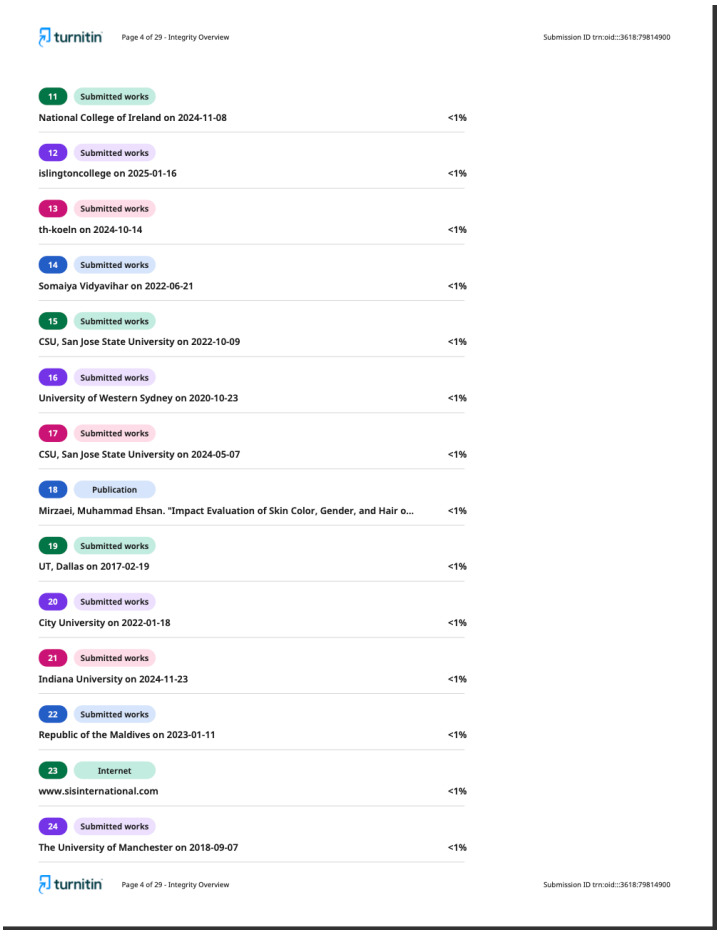
The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	University of Newcastle on 2020-10-18 — 2%
2	Internet	mmcalumni.ca — 1%
3	Submitted works	University of Arizona on 2020-03-09 — 1%
4	Submitted works	University of Illinois at Urbana-Champaign on 2022-02-03 — 1%
5	Submitted works	University of Utah on 2024-10-07 — 1%
6	Submitted works	islingtoncollege on 2025-01-16 — 1%
7	Submitted works	islingtoncollege on 2025-01-16 — 1%
8	Submitted works	University College London on 2022-12-13 — <1%
9	Submitted works	Purdue University on 2023-10-27 — <1%
10	Internet	discovery.researcher.life — <1%

turnitin Page 3 of 29 - Integrity Overview Submission ID trn:oid:::3618:79814900

22067335 | Rajita Maharjan

CU6051NI Artificial Intelligence



turnitin Page 4 of 29 - Integrity Overview Submission ID trn:oid:::3618:79814900

11	Submitted works	National College of Ireland on 2024-11-08 <1%
12	Submitted works	islingtoncollege on 2025-01-16 <1%
13	Submitted works	th-koeln on 2024-10-14 <1%
14	Submitted works	Somaiya Vidyavihar on 2022-06-21 <1%
15	Submitted works	CSU, San Jose State University on 2022-10-09 <1%
16	Submitted works	University of Western Sydney on 2020-10-23 <1%
17	Submitted works	CSU, San Jose State University on 2024-05-07 <1%
18	Publication	Mirzaei, Muhammad Ehsan. "Impact Evaluation of Skin Color, Gender, and Hair o... <1%
19	Submitted works	UT, Dallas on 2017-02-19 <1%
20	Submitted works	City University on 2022-01-18 <1%
21	Submitted works	Indiana University on 2024-11-23 <1%
22	Submitted works	Republic of the Maldives on 2023-01-11 <1%
23	Internet	www.sisinternational.com <1%
24	Submitted works	The University of Manchester on 2018-09-07 <1%

turnitin Page 4 of 29 - Integrity Overview Submission ID trn:oid:::3618:79814900

turnitin Page 5 of 29 - Integrity Overview Submission ID trn:oid:::3618:79814900

25	Internet	Natalia Nehrebecka. "COVID-19: stress-testing non-financial companies: a macro... <1%
26	Submitted works	islingtoncollege on 2025-01-16 <1%
27	Internet	mobt3ath.com <1%
28	Internet	www.techscience.com <1%
29	Submitted works	Curtin University of Technology on 2024-11-09 <1%
30	Submitted works	De Montfort University on 2019-09-20 <1%
31	Submitted works	Nottingham Trent University on 2019-02-01 <1%
32	Submitted works	islingtoncollege on 2025-01-16 <1%
33	Internet	rstudio-pubs-static.s3.amazonaws.com <1%
34	Submitted works	islingtoncollege on 2025-01-16 <1%
35	Submitted works	Saxion Brightspace on 2024-12-17 <1%
36	Submitted works	islingtoncollege on 2025-01-16 <1%
37	Submitted works	islingtoncollege on 2025-01-16 <1%

turnitin Page 5 of 29 - Integrity Overview Submission ID trn:oid:::3618:79814900

22067335 | Rajita Maharjan

Abstract

The prediction of movie ratings would be crucial to help to choose a good movie and avoid some bad selections. Despite the huge number of people checking ratings before watching, it is still very hard to predict the kind of movies that will rate highly. The objective of this project has been to build a machine-learning model which predicts the rating given to a movie based on its genre, duration, and director.

In this report, three machine-learning algorithms have been selected, K-Nearest Neighbors, Decision Tree Regressor, and Random Forest Regressor. The data had several features related to the movie that could help in identifying the rating and help the viewers make informed decisions on which movie to watch.

The best performance was shown by the Random Forest Regressor, with an accuracy score of 92.42%; next was the Decision Tree Regressor, which gave 89.29% accuracy. The same turned around to 87.90% accuracy following the tuning of the KNN model. It shows that it followythat in the context of movie ratings, an algorithm imbuing the most accurate predictions would be the Random Forest. However, KNN can also do well when properly tuned.

Table of Contents

Table of Figures

1. Introduction

1.1. Introduction to AI

Artificial Intelligence is the machine can perform capabilities that are usually linked with human intelligence, such as learning, problem-solving, and language understanding. AI makes it possible for systems to imitate and mirror human like functions; consequently, which enables timely responses to new data, decisions about the data, and their follow-up actions (Duggal, 2024).

AI has subspaces of machine learning, deep learning, and natural language processing. Machine learning applies algorithms in designing systems that have a capability to learn and enhance improvement based on the kind of data to which they are exposed, with no human programming involved. Deep learning takes it a step further by using neural network models with multiple layers that represent complex patterns. NLP allows machines to understand and interpret human language, thus making it fitting for use in chatbots, language translation, and so forth (StackCommerce, 2025).

AI is quite useful in many different sectors, including transport, health, finance, and media. In health, it helps diagnose diseases and develops the right treatment plan for the affected person. In terms of finance, it is applied in fraud detection and algorithmic trading (Brooke, 2025) (Slack, 2024).

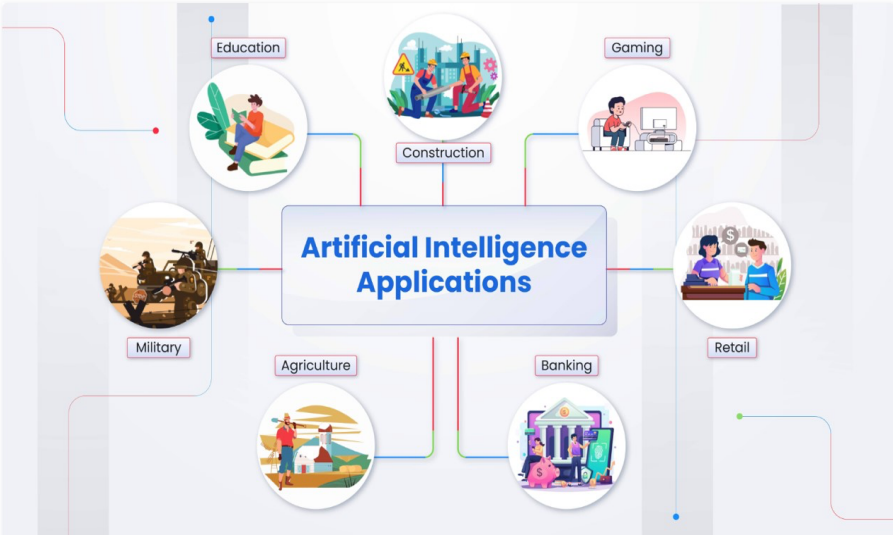


Figure 1: Application of AI (bharat, 2023)

1.2. Introduction to Machine Learning (ML)

Machine Learning is one among many subcategories of Artificial Intelligence, through which machines can learn from data and experiences. This learning helps in recognizing patterns for suitable ML models can make predictions with good accuracy rates, even with minimal human intervention**.**

In this project it looks for patterns in user preferences and movie attributes that might be critical in predicting what rating a user is going to give to a movie. k-Nearest Neighbors, Decision Trees, and Random Forest Regressors are implemented in order to capture the complex relationship between users and movies in the system, while boosting its accuracy in making predictions. (Pedamkar, 2023)

This approach does not only automate recommendation but goes a step further to personalize these recommendations based on individual user tastes in order to drive user satisfaction and engagement (brown, 2021).

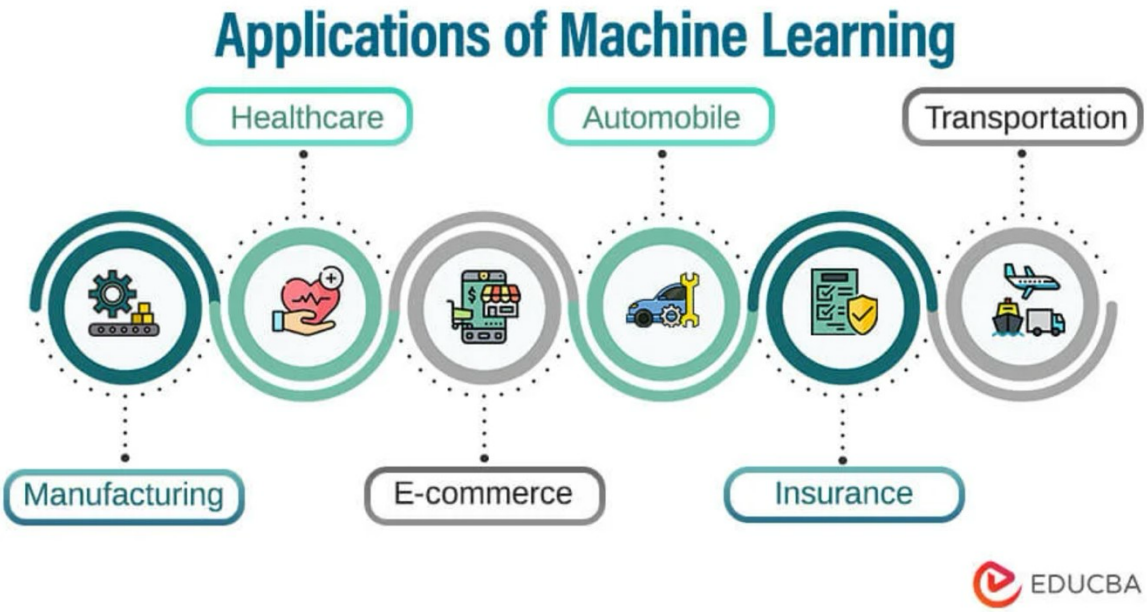


Figure 2: Intoduction to ML

1.3. Problem Domain

This domain problem is focused on the prediction of movie ratings, which tries to estimate how users would rate a movie by their past interactions, preferences and the movie's attributes. To enhance customer engagement and experience in the current era, it has become very important to be able to estimate user preferences with the huge movie dataset now available because of rapid growth in the entertainment sector.

This is predictively modeled by a recommendation system to predict future user movie ratings by considering historical data such as the user's past ratings on movies of similar genres, the directors, and each cast member. It has so far proven to be quite useful in streaming platforms at this age of content personalization to develop retention and hence growth in revenue.

The major challenges include the scattered data in the field, different preferences of users, and cold-start problems for new users or movies. Advanced algorithms, such as k-Nearest Neighbors, Decision Trees, and Random Forest Regressors, are used to effectively handle the above issues.

This will not only help platforms enhance user satisfaction but also contribute to gaining insight into market trends and audience behavior for effective content creation and delivery.

1.4. Objectives

- Build a model that is used to forecast movie ratings based on features that include genre, duration, director, and other useful attributes.
- Tuning and optimizing the K-nearest neighbors algorithm to predict better.
- Calculate advanced evaluation metrics such as accuracy, Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score to compare the results with different algorithms.
- The insights will guide and inform movie viewers toward making decisions on movies to watch based on predicted ratings.

- Recommendations could be made toward movie creators on how their films may perform based on predicted ratings.
- Show how machine learning can predict movie ratings and hence improve the experience.

2. Background

2.1. Research work done on chosen topic

2.1.1. Movie Ratings and Recommendations

Movies have always played the important role in the industry and have been able to attract audiences worldwide due to diversified genres and themes. With such a large and impressive library of motion pictures available today, the end-users are still in confusion about watching the right kind of movies as per their needs and genres. which has led to the development of recommendation systems predicting movie ratings and suggesting films as per their taste (bharat, 2023) (C.Aggarwal, 2016).

A movie's rating can give its quality and how much the audience approves of it. Other factors include the story, genre, cast, and direction. Accurate prediction of ratings helps the streaming sites like Netflix, Amazon Prime, Hulu, etc. to increase consumer satisfaction level by the recommendation of movies. However, it is also economically significant, as the boosting platform subscriptions and viewer engagement show improvements in both cases.

This is also an area where recommendation systems proves to be very important at a global level. However, the related challenges with scattered data and cold starts at new users or movies are in place. Improving the efficiency of the systems will address these issues.

2.1.2. Movie Rating Prediction with Machine Learning

Machine learning is a subset of artificial intelligence that allows for giving the ability to systems to learn and make predictions from data without being explicitly programmed. The entertainment industry uses many machine learning techniques for analyzing data to predict user interest and offer appropriate content. Specifically, in platforms that manage large

movie catalogs, this learning and adaptation would lead to high user satisfaction and engagement.

Movie prediction, on the other hand, is completely dependent on two factors user behavior and movie features, like genres, cast, and directors. It tries to predict the likelihood that a user will like a movie. Machine learning can help build recommendation systems that work with problems like scattered data and changing user preferences. Basically, these algorithms are designed to learn from the users ratings and attributes so that they update or refine their predictions when getting additional data (Kniazieva, 2022) (Wilk, 2023).

The predictions made by models based on techniques like Collaborative Filtering, Content-Based Filtering, advanced techniques based on Neural Networks, and Matrix Factorization are very high in accuracy. Random Forest, k-Nearest Neighbor (kNN), and Decision Trees are some of the most popular algorithms in used in this, each with unique strengths. Machine learning has delivered models that are more than 90% accurate in predicting movie ratings (Soham, 2024).

Machine learning can be used for movie rating, personalization of experiences, and ensuring that users stick to a platform over time in making an effort to further scale datadriven decisions regarding content curation. This is how AI readily accommodates the highly dynamic needs of the entertainment industry.

2.1.3. Advantages of Movie Ratings Predictions

- Machine learning algorithms can analyze large datasets to identify the patterns and trends which enables the accurate predictions for the preferences of the users.
- By understanding individual user behaviors and preferences the models can suggest movies that matches with the individual taste which also enhances the user satisfactions.
- For better user engagement advanced algorithms can achieve higher accuracy in predicting movie ratings.

2.1.4. Disadvantages of Movie Ratings Predictions

- It can lead to incorrect predictions if there is inaccurate and incomplete data which may also affect the user trust in the system.
- It makes them less effective for real time predictions when some machine learning models needs significant resources.
- Models might perform poorly on new and unseen data it may become too hard to train the data's.

2.1.5. Explanation of Dataset

A dataset is a collection of data stored in digitally which forms the foundation for machine learning projects. It may include different types of information like images, text, audio, video, numerical data points (wikipedia, 2025).

Datasets are categorized in two types:

- **Structured Data:** It is organized information like spreadsheets or database tables which makes it easily manageable and accessible.
- **Unstructured Data:** It includes formats like text or images which has less structure.

For movie prediction projects datasets which has movie attributes and user ratings are used. It usually has features like color, director name, duration, genres, country, budget, etc.

These features are important for building machine learning models which predicts user ratings for movies. Kaggle provides various datasets suitable for projects including movie dataset.

The dataset for this prediction system is sourced form Kaggle (MPOY, 2019).The model contains following attributes: -

- **color**: Indicates whether the movie is in color or black-and-white.

- **director_name**: Name of the movie's director.
- **num_critic_for_reviews**: Number of critic reviews the movie has received.
- **duration**: Length of the movie in minutes.
- **director_facebook_likes**: Number of likes the director has on their Facebook page.
- **actor_3_facebook_likes**: Number of Facebook likes for the third most prominent actor in the movie.
- **actor_2_name**: Name of the second most prominent actor.
- **actor_1_facebook_likes**: Number of Facebook likes for the lead actor.
- **gross**: Total worldwide box office earnings of the movie.
- **genres**: Categories or genres the movie belongs to (e.g., Action, Comedy).
- **actor_1_name**: Name of the lead actor.
- **movie_title**: Title of the movie.
- **num_voted_users**: Number of users who have voted or rated the movie.
- **cast_total_facebook_likes**: Total number of Facebook likes for the entire cast.
- **actor_3_name**: Name of the third most prominent actor.
- **facenumber_in_poster**: Number of actor faces appearing on the movie poster.
- **plot_keywords**: Keywords describing the movie's plot.
- **movie_imdb_link**: URL link to the movie's IMDb page.
- **num_user_for_reviews**: Number of user reviews the movie has received.
- **language**: Primary language spoken in the movie.
- **country**: Country where the movie was produced.
- **content_rating**: Movie's rating indicating its suitability for different audiences (e.g., PG13, R).
- **budget**: Estimated production budget of the movie.
- **title_year**: Year the movie was released.

- **actor_2_facebook_likes**: Number of Facebook likes for the second most prominent actor.
- **imdb_score**: IMDb rating score of the movie.
- **aspect_ratio**: Aspect ratio of the movie's visual presentation.
- **movie_facebook_likes**: Number of likes the movie's official Facebook page has received.

3. Proposed Solution

After researching different algorithms for predicting movie ratings, the selected algorithms used for this project are K-Nearest Neighbors (KNN), Decision Tree Regressor, and Random Forest Regressor. The reason behind choosing these algorithms is to achieve the higher accuracy in order to predict movie ratings. The algorithms are also widely used for supervised learning problems specifically the regression tasks. The results from each model will be later compared to identify as much as accurate results for predicting ratings of the movie.

3.1. Explanation of Algorithm to be used

3.1.1. K-Nearest Neighbours(KNN)

KNN is one of the simplest algorithms applied to predict the output found on the nearest data points. For example in movie rating prediction KNN looks at the movie features, like genre and duration and checks which movie are similar. The average rating of these similar movies should be predicted value for the movie (Blog, 2024).

- It locates ‘K’ nearest movies based on similar features.
- It then predicts a rating by averaging the ratings of those nearest movies.

3.1.2. Descision Tree Regressor

A Decision Tree looks like a flowchart that offers a guideline in making a decision. Data is then split in relation to diverse features in such a way that one gets the most accurate

prediction. The algorithm to be used in predicting movie ratings is creating a tree-like structure whereby each node represents.splits according to another feature, for instance, duration, director, and so forth (Navlani, 2024).

- It starts from the top and checks a characteristic, such as genre.
- The data is then divided into two groups and the process is repeated to form another part of the tree.
- At the end of the tree, the predicted rating will be the average of the ratings in that group.

3.1.3. Random Forest Regressor

Random Forest Regressor is basically an group or combination of several Decision Trees. The Random Forest uses a set of trees, not just one tree, to make predictions. It averages the result of all these trees in its forest to boost prediction accuracy and decrease error from a single tree (Sruthi, 2024).

- It builds a lot of decision trees using random data points.
- Each tree makes an independent prediction.
- The finale prediction is the average prediction of all subtrees.

3.2. Pseudocode For The Proposed Solution

START

IMPORT libraries

LOAD dataset

CLEAN dataset

HANDLE missing data

REMOVE duplicates

PRE-PROCESS data

ENCODE categorical variables

SCALE features if needed

SPLIT data into training and testing sets

CREATE models (KNN, Decision Tree, Random Forest)

TRAIN models

EVALUATE models on test data

COMPARE model performance

SELECT best model

END

3.3. Flowchart

3.3.1. Overall Flowchart

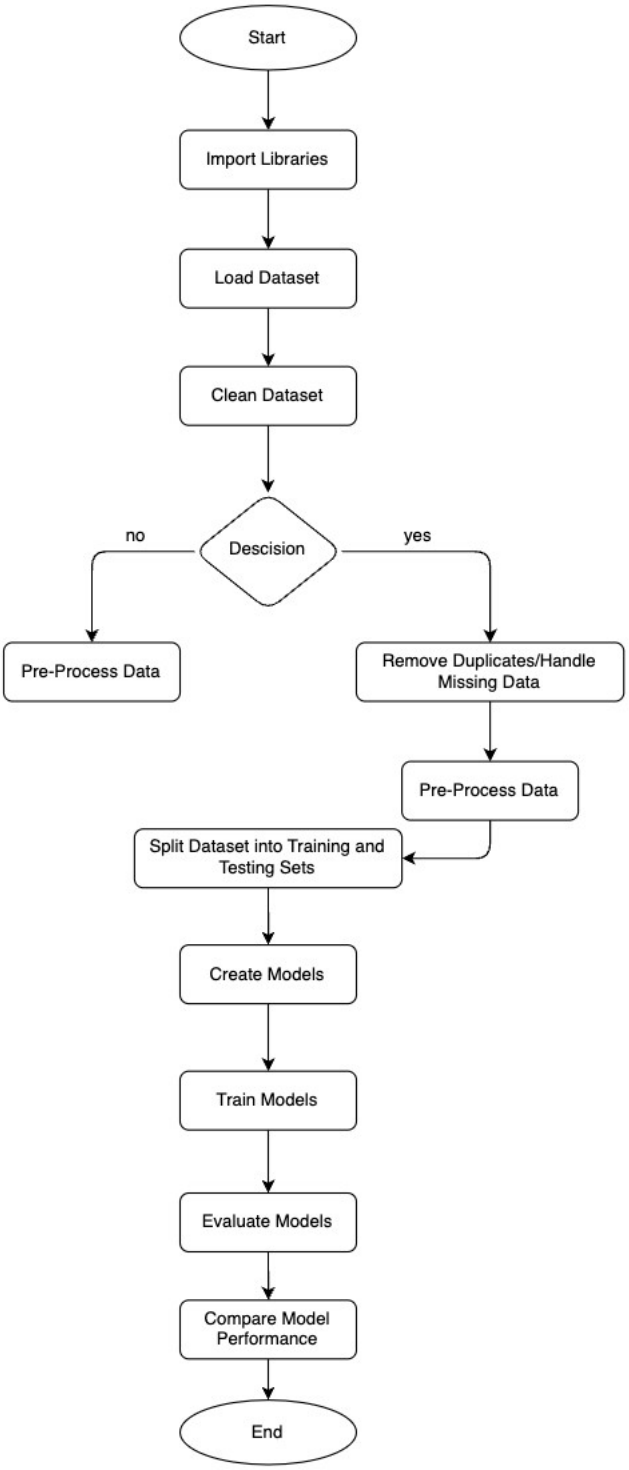


Figure 3: Overall Flowchart

3.3.2. KNN Flowchart

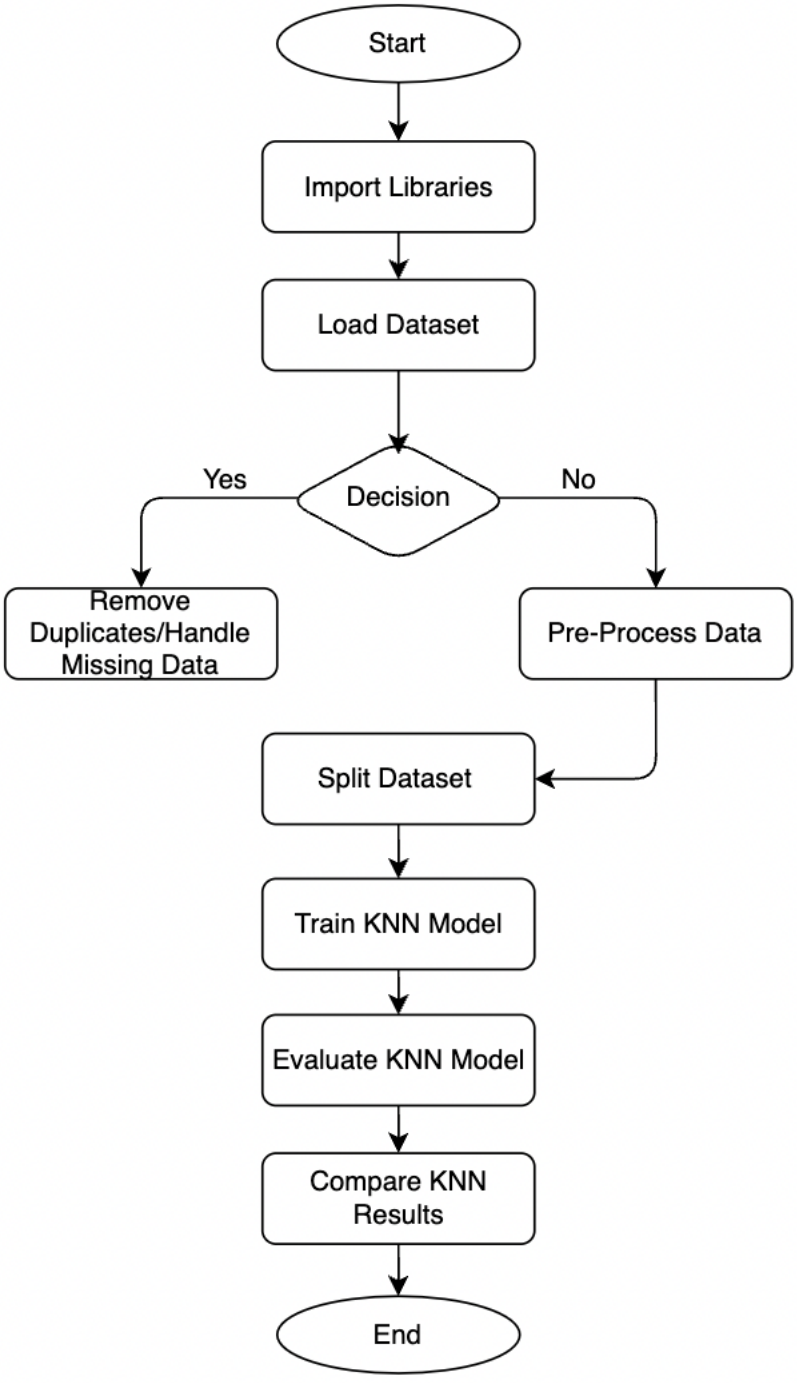


Figure 4: KNN Flowchart

3.3.3. Descision Tree Flowchart

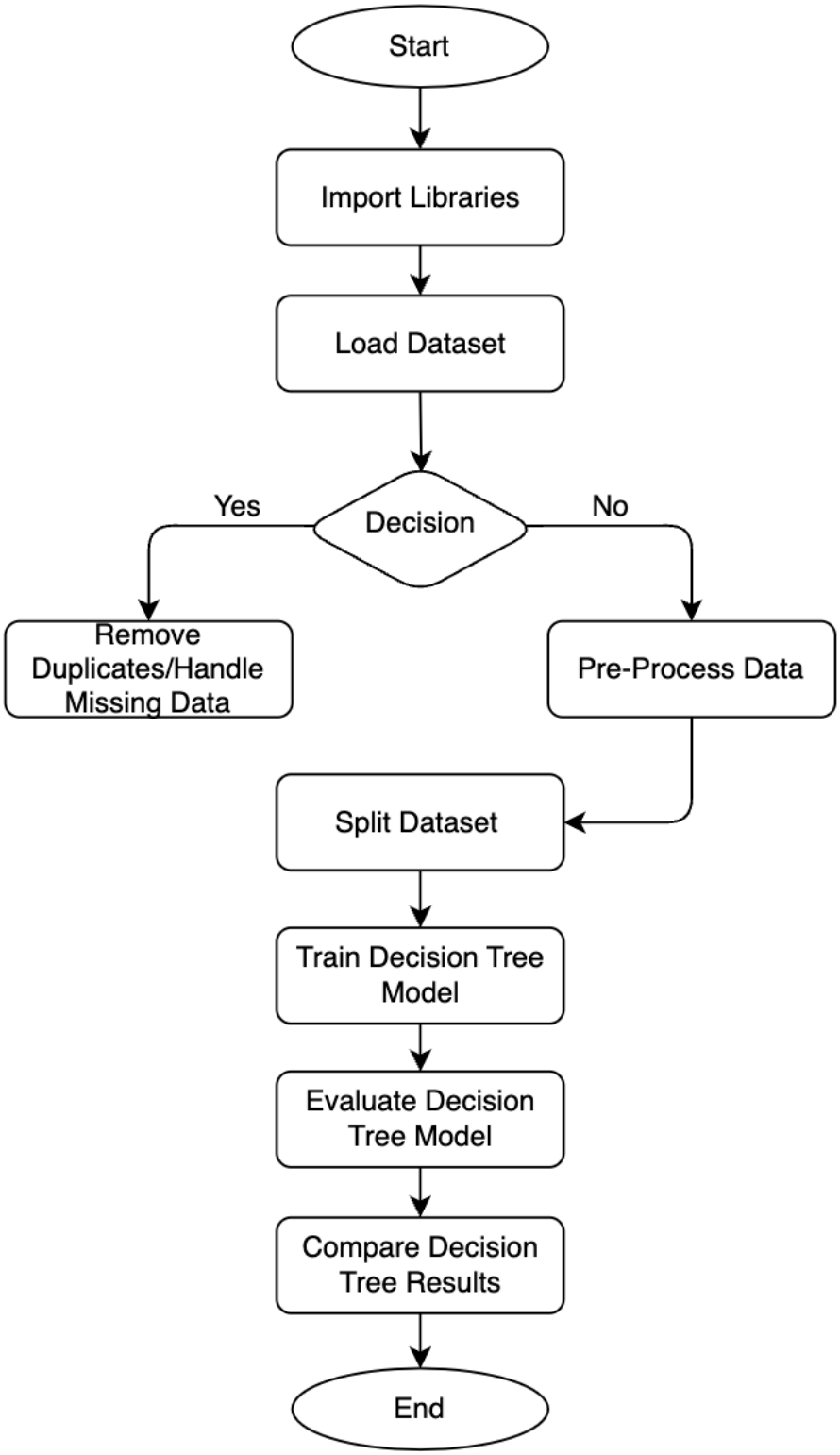


Figure 5: Decision tree flowchart

3.3.4. Random Forest Flowchart

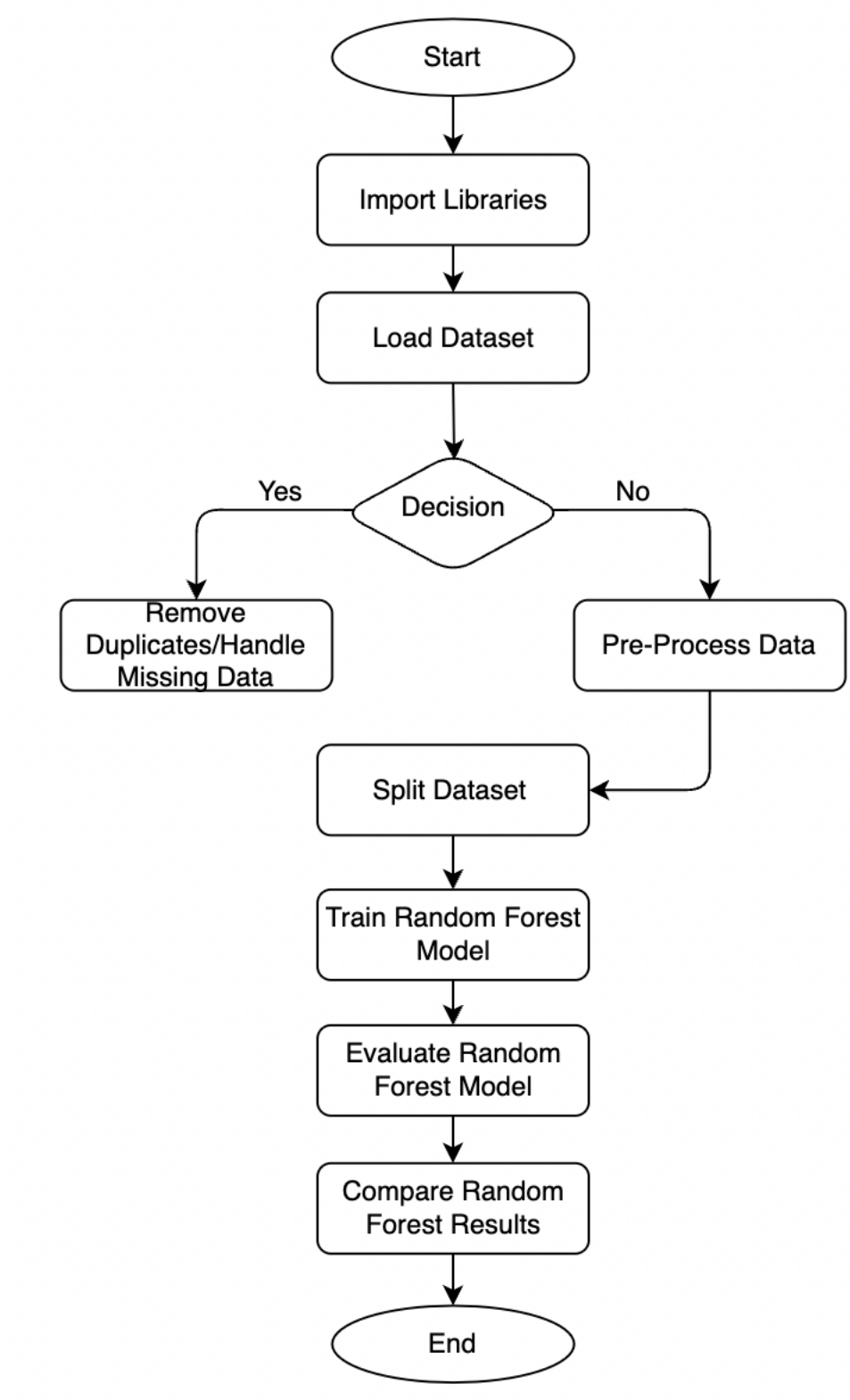


Figure 6: Random forest flowchart

3.4. Develoment Process

3.4.1. Tools Used

- **Anaconda**

  Anaconda is an open-source platform used in coding with Python. Writing and running Python codes is eased, more so when it comes to machine learning and data science. Anaconda Navigator will be used in this project to help manage packages and environments, all without issuing multiple lines of code in the terminal (Open Source , 2025).

- **Jupyter Notebook**

  Jupyter Notebook is a free web application that provides an interactive way to create and share documents with live code, equations, visualizations, or narrative texts. In this project, it is used for exploratory analysis, building data models, and creating visualizations for the results (Jupyter Notebook, 2024).

- **Excel**

  Excel is an excellent tool for spreadsheet work and is applied to organizing, formatting, and performing calculations on data. In this project, work had to be done for managing prediction dataset using Excel: first cleaning and organizing the dataset to be put through further analysis (Gunjan, 2024).

3.4.2. Toolkit Used

- **NumPy**

  The most commonly used Python package for numerical computation is Numerical Python. It will enable fast and efficient array operations, an implementation that makes it a critical aspect of this project (Wong, 2024).

- **Pandas**

  Pandas is a Python library for data analysis; it offers data structures that make handling structured data very easy. Since it is built over NumPy, it is helpful to handle the dataset in this project prediction (V., n.d.).

- **Matplotlib**

  Matplotlib is the Python plotting library that produces figures or graphs in a variety of formats, and it is usable in a variety of environments. For this project, it is applied to forming bar graphs for the reason that it will help to represent the data/results very expressively (Introduction to Matplotlib, 2024).

- **Seaborn**

  Seaborn is a manipulative visual application into Matplotlib that is implemented for designing nice static graphical representations. This is why it's applied in the following project: diving deep into explorations of the dataset under consideration (An introduction to seaborn, 2024).

- **Scikit-learn**

  Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms. In this project, it is used for importing and implementing the algorithms for KNN, Logistic Regression, and Random Forest (scikit-learn Machine Learning in Python, 2024).

3.4.3. Explanation of Development Process

3.4.3.1. Data Understanding

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 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  
dtypes: float64(13), int64(3), object(12)
memory usage: 1.1+ MB
```

Figure 7: data understanding

3.4.3.2. Data Preprocessing

5. Pre-Processing Data

```
# Selecting relevant columns
movie_ratings = df[['imdb_score', 'budget', 'num_critic_for_reviews','duration','gross',
                    'num_voted_users','cast_total_facebook_likes', 'movie_facebook_likes',
                    'num_user_for_reviews','title_year', 'content_rating', 'director_name',
                    'genres', 'actor_1_name', 'actor_2_name', 'country']]

# Dropping any remaining rows with missing values
movie_ratings = movie_ratings.dropna()

# Label encoding for categorical features
le = LabelEncoder()
categorical_columns = ['content_rating', 'director_name', 'genres', 'actor_1_name', 'actor_2_name', 'country']
for column in categorical_columns:
    movie_ratings[column] = le.fit_transform(movie_ratings[column])

# Defining the target variable (y) and feature variables (X)
y = movie_ratings['imdb_score']
X = movie_ratings.drop('imdb_score', axis=1)
```

Figure 8: data preprocesing

3.4.3.3. Data Preparation

1. Importing Libraries

```
import pandas as pd
import numpy as np
import random as rnd

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor  # for regression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score  # for evaluating regression models

# Suppressing warnings
import warnings
warnings.filterwarnings('ignore')
```

Figure 9: Importing libraries

2. Loading Dataset

```
# Reading CSV file into DataFrame
df = pd.read_csv("movie_metadata.csv")
# Displaying the first few rows of the DataFrame to understand its structure
df.head()
```

Figure 10: loading dataset

3. Checking for Missing Data or Duplicates

```
# Checking for missing values
df.isnull().sum()
```

```
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
dtype: int64
```

4. Handling Missing Data

```
# Cleaning the data by removing rows with missing values
df.dropna(inplace=True)
# Confirming the changes
df.shape
```

```
(3755, 28)
```

Figure 11: checking and handling missing data

3.4.3.4. Data Modeling, Initializing, Training, and Testing the Machine Learning Models

6. Splitting Dataset into Training and Testing Set

```
# Splitting the data into training and test sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 12: splitting data

7. Creating Models K-Nearest Neighbors (KNN)

```
# Training KNN model
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

8. Evaluating KNN Model

```
# Making predictions with KNN model
y_pred_knn = knn.predict(X_test)

# Evaluating the model
mae_knn = mean_absolute_error(y_test, y_pred_knn)
mse_knn = mean_squared_error(y_test, y_pred_knn)
rmse_knn = np.sqrt(mse_knn)
r2_knn = r2_score(y_test, y_pred_knn)

# Printing evaluation metrics
print(f'KNN Model - MAE: {mae_knn}, MSE: {mse_knn}, RMSE: {rmse_knn}, R²: {r2_knn}')
```

```
KNN Model - MAE: 0.8772569906790945, MSE: 1.3403627163781624, RMSE: 1.157740349291741, R²: -0.13929022575119365
```

Figure 13: creating knn model and evaluating

9. Comparing KNN Results

```
# Calculating percentage accuracy for KNN
percentage_accuracy_knn = 100 * (np.mean(y_test) - mae_knn) / np.mean(y_test)
print(f'Percentage Accuracy for KNN: {percentage_accuracy_knn:.2f}%')
```

```
Percentage Accuracy for KNN: 86.39%
```

Figure 14: comparing knn results

10. Creating Decision Tree Model

```
# Training Decision Tree model
dr = DecisionTreeRegressor()
dr.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

11. Evaluating Decision Tree Model

```
# Making predictions with Decision Tree model
y_pred_dt = dr.predict(X_test)

# Evaluating the model
mae_dt = mean_absolute_error(y_test, y_pred_dt)
mse_dt = mean_squared_error(y_test, y_pred_dt)
rmse_dt = np.sqrt(mse_dt)
r2_dt = r2_score(y_test, y_pred_dt)

# Printing evaluation metrics
print(f'Decision Tree Model - MAE: {mae_dt}, MSE: {mse_dt}, RMSE: {rmse_dt}, R²: {r2_dt}')
```

```
Decision Tree Model - MAE: 0.6902796271637816, MSE: 0.9669773635153129, RMSE: 0.9833500717014836, R²: 0.17808228674659987
```

Figure 15: creating and evaluating decision tree model

12. Comparing Decision Tree Results

```
# Calculating percentage accuracy for Decision Tree
percentage_accuracy_dt = 100 * (np.mean(y_test) - mae_dt) / np.mean(y_test)
print(f'Percentage Accuracy for Decision Tree: {percentage_accuracy_dt:.2f}%')
```

```
Percentage Accuracy for Decision Tree: 89.29%
```

Figure 16: coparing the decision tree results

13. Creating Random Forest Model

```
# Training Random Forest model
random_regressor = RandomForestRegressor(n_estimators=50)
random_regressor.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(n_estimators=50)
```

14. Evaluating Random Forest Model

```
# Making predictions with Random Forest model
y_pred_rf = random_regressor.predict(X_test)

# Evaluating the model
mae_rf = mean_absolute_error(y_test, y_pred_rf)
mse_rf = mean_squared_error(y_test, y_pred_rf)
rmse_rf = np.sqrt(mse_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# Printing evaluation metrics
print(f'Random Forest Model - MAE: {mae_rf}, MSE: {mse_rf}, RMSE: {rmse_rf}, R²: {r2_rf}')
```

```
Random Forest Model - MAE: 0.4888655126498016, MSE: 0.45938867110519305, RMSE: 0.6777821708374993, R²: 0.6095258272885937
```

15. Comparing Random Forest Results

```
# Calculating percentage accuracy for Random Forest
percentage_accuracy_rf = 100 * (np.mean(y_test) - mae_rf) / np.mean(y_test)
print(f'Percentage Accuracy for Random Forest: {percentage_accuracy_rf:.2f}%')
```

```
Percentage Accuracy for Random Forest: 92.42%
```

Figure 17: creating evaluating random forest model and comparing the results

16. Comparing Model Performance

```
# Comparing model performance
models_performance = pd.DataFrame({
    'Model': ['K-Nearest Neighbors', 'Decision Tree', 'Random Forest'],
    'RMSE': [rmse_knn, rmse_dt, rmse_rf],
    'R² Score': [r2_knn, r2_dt, r2_rf],
    'Percentage Accuracy': [percentage_accuracy_knn,
                            percentage_accuracy_dt,
                            percentage_accuracy_rf]
})

print(models_performance)
```

```
                Model      RMSE  R² Score  Percentage Accuracy
0  K-Nearest Neighbors  1.157740 -0.139290            86.389985
1        Decision Tree  0.983350  0.178082            89.290805
2        Random Forest  0.677782  0.609526            92.415601
```

Figure 18: comparing model performance

17. Hyperparameter Tuning for KNN

```
from sklearn.model_selection import GridSearchCV

# Defining the parameter grid for KNN
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11, 13, 15],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski']
}

# Initializing the KNN model
knn = KNeighborsRegressor()

# Performing grid search with cross-validation
grid_search_knn = GridSearchCV(estimator=knn, param_grid=param_grid, cv=5, scoring='neg_mean_absolute_error', n_jobs=-1)
grid_search_knn.fit(X_train, y_train)

# Getting the best parameters and best score
best_params_knn = grid_search_knn.best_params_
best_score_knn = grid_search_knn.best_score_

print(f'Best parameters for KNN: {best_params_knn}')
print(f'Best cross-validation score (negative MAE): {best_score_knn:.2f}')
```

```
Best parameters for KNN: {'metric': 'manhattan', 'n_neighbors': 15, 'weights': 'distance'}
Best cross-validation score (negative MAE): -0.78
```

Figure 19: hyperparameter tuning KNN

18. Training Tuned KNN Model

```
# Training the KNN model with the best parameters
best_knn = grid_search_knn.best_estimator_
best_knn.fit(X_train, y_train)

# Making predictions with the tuned KNN model
y_pred_knn_tuned = best_knn.predict(X_test)

# Evaluating the tuned KNN model
mae_knn_tuned = mean_absolute_error(y_test, y_pred_knn_tuned)
mse_knn_tuned = mean_squared_error(y_test, y_pred_knn_tuned)
rmse_knn_tuned = np.sqrt(mse_knn_tuned)
r2_knn_tuned = r2_score(y_test, y_pred_knn_tuned)

# Calculating percentage accuracy for the tuned KNN model
percentage_accuracy_knn_tuned = 100 * (np.mean(y_test) - mae_knn_tuned) / np.mean(y_test)
print(f'Percentage Accuracy for Tuned KNN: {percentage_accuracy_knn_tuned:.2f}%')
```

```
Percentage Accuracy for Tuned KNN: 87.90%
```

Figure 20: trainingtuned KNNmodel

19. Final Comparison of Model Performance

```
# Comparing model performance
models_performance = pd.DataFrame({
    'Model': ['Tuned K-Nearest Neighbors', 'Decision Tree', 'Random Forest'],
    'RMSE': [rmse_knn_tuned, rmse_dt, rmse_rf],
    'R² Score': [r2_knn_tuned, r2_dt, r2_rf],
    'Accuracy %': [percentage_accuracy_knn_tuned,
                   percentage_accuracy_dt,
                   percentage_accuracy_rf]
})

print(models_performance)
```

```
                    Model      RMSE  R² Score  Accuracy %
0  Tuned K-Nearest Neighbors  1.051947  0.059411   87.895186
1              Decision Tree  0.983350  0.178082   89.290805
2              Random Forest  0.677782  0.609526   92.415601
```

Figure 21: final comparision after the tuned model

3.4.3.5. Data Visualization

20. Data Visualisations

```
# RMSE Comparison Bar Plot
plt.figure(figsize=(8, 6))
sns.barplot(x='Model', y='RMSE', data=models_performance, palette='Set2')
plt.title('Model Comparison: RMSE')
plt.ylabel('RMSE')
plt.tight_layout()
plt.show()
```

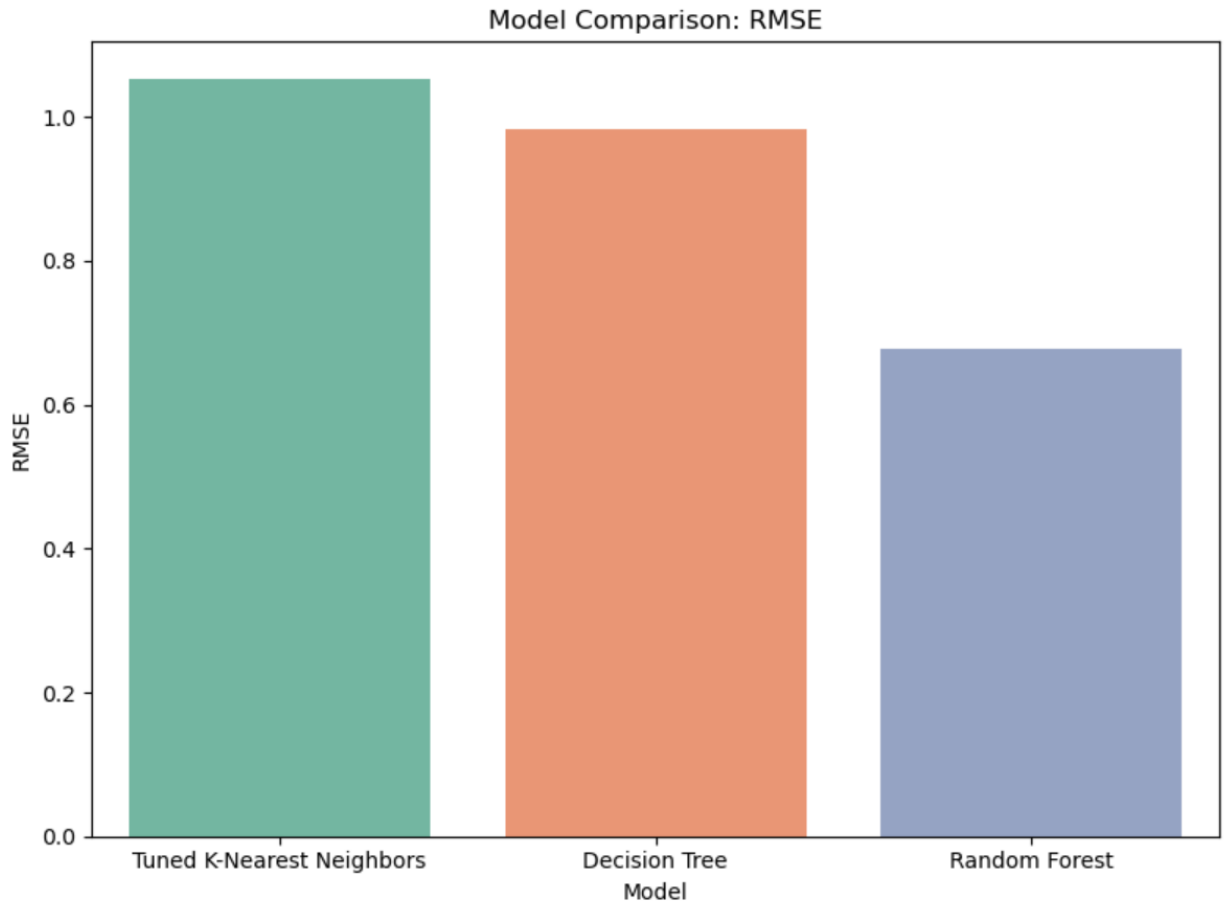


Figure 22: RMSE barplot

```
# R² Score Comparison Bar Plot
plt.figure(figsize=(8, 6))
sns.barplot(x='Model', y='R² Score', data=models_performance, palette='Set2')
plt.title('Model Comparison: R² Score')
plt.ylabel('R² Score')
plt.tight_layout()
plt.show()
```

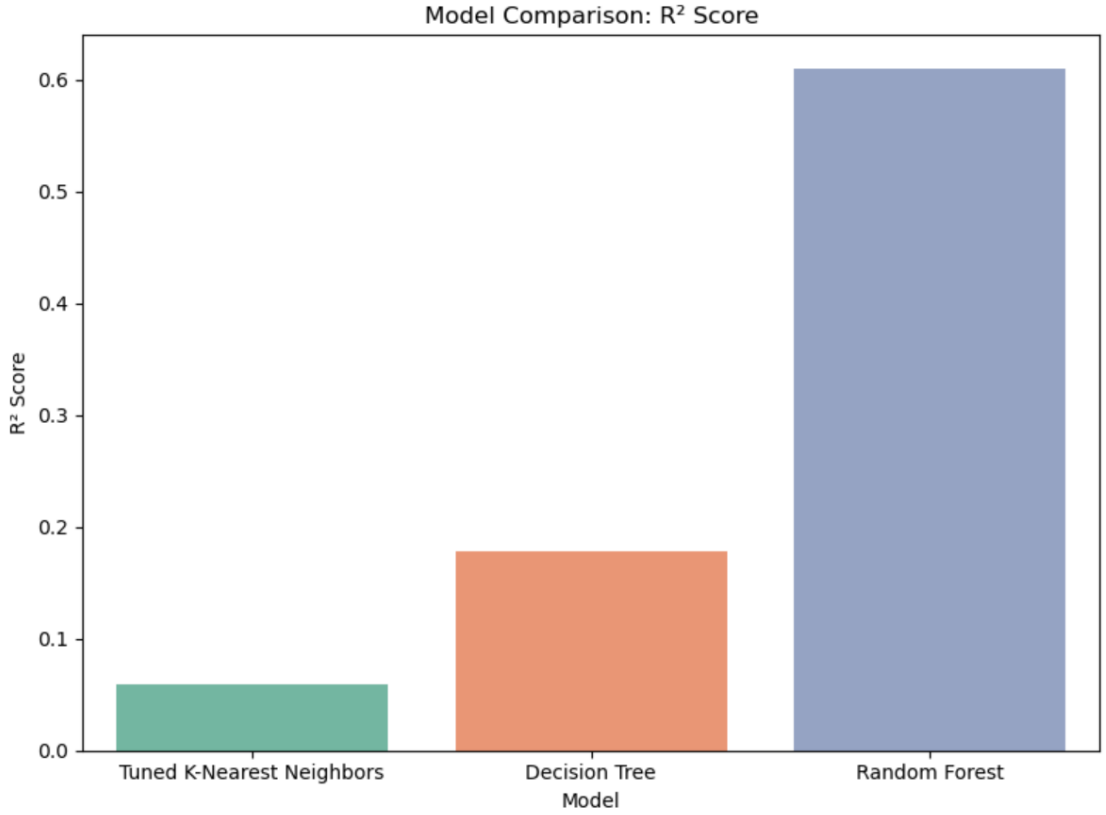


Figure 23: R^2 bar plot

```
]: # Accuracy % Comparison Bar Plot
plt.figure(figsize=(8, 6))
sns.barplot(x='Model', y='Accuracy %', data=models_performance, palette='Set2')
plt.title('Model Comparison: Accuracy %')
plt.ylabel('Accuracy %')
plt.tight_layout()
plt.show()
```

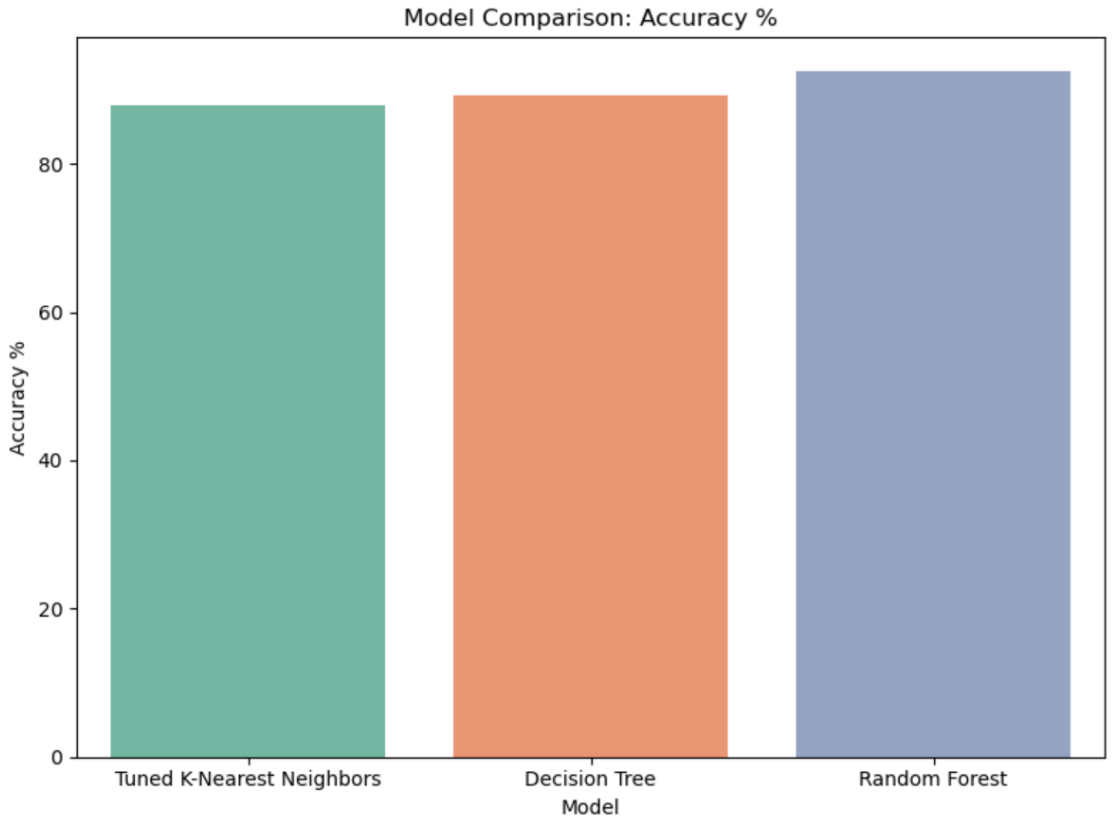


Figure 24: accuracy barplot

```
# Calculate error distribution
# KNN
error_distribution = y_test - y_pred_knn
plt.figure(figsize=(10, 6))
sns.histplot(error_distribution, bins=30, kde=True)
plt.title('Error Distribution of KNN Model')
plt.xlabel('Error (Actual - Predicted)')
plt.ylabel('Frequency')
plt.show()
```

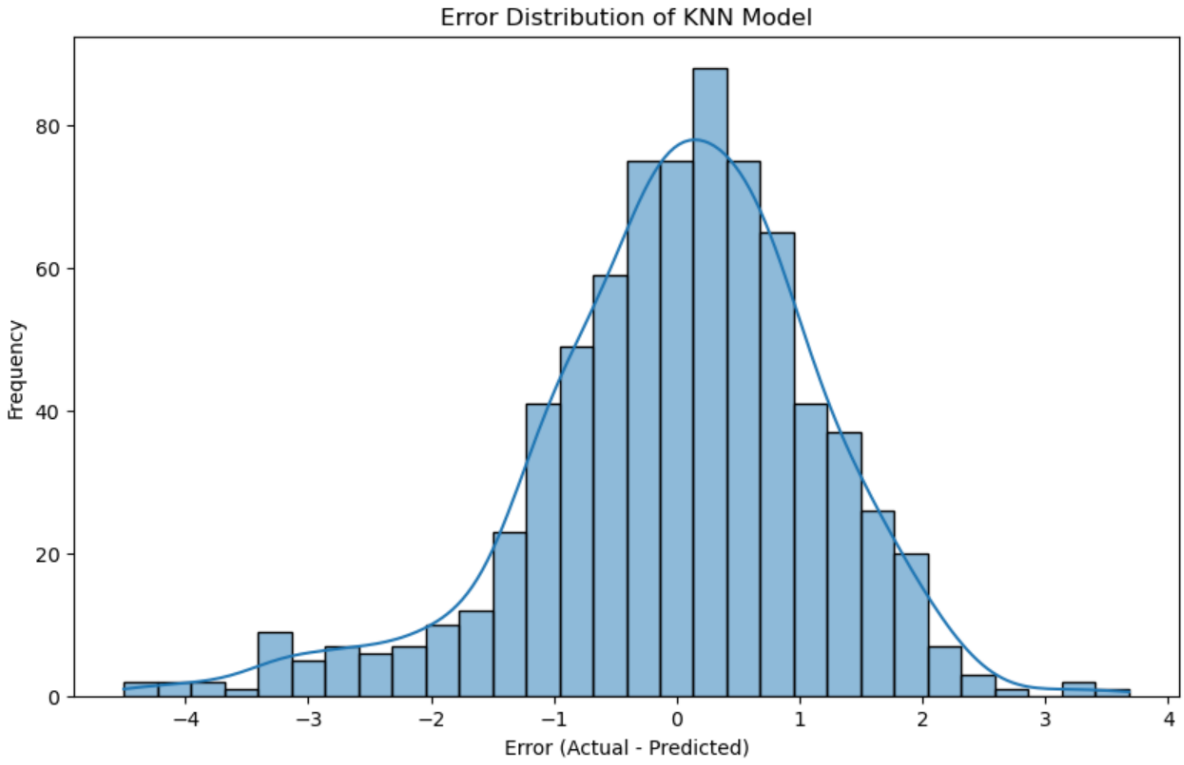


Figure 25: error distribution of KNN

```python
# Decision Tree Model
error_distribution_dt = y_test - y_pred_dt  # Use y_pred_dt for Decision Tree
plt.figure(figsize=(10, 6))
sns.histplot(error_distribution_dt, bins=30, kde=True)
plt.title('Error Distribution of Decision Tree Model')
plt.xlabel('Error (Actual - Predicted)')
plt.ylabel('Frequency')
plt.show()
```

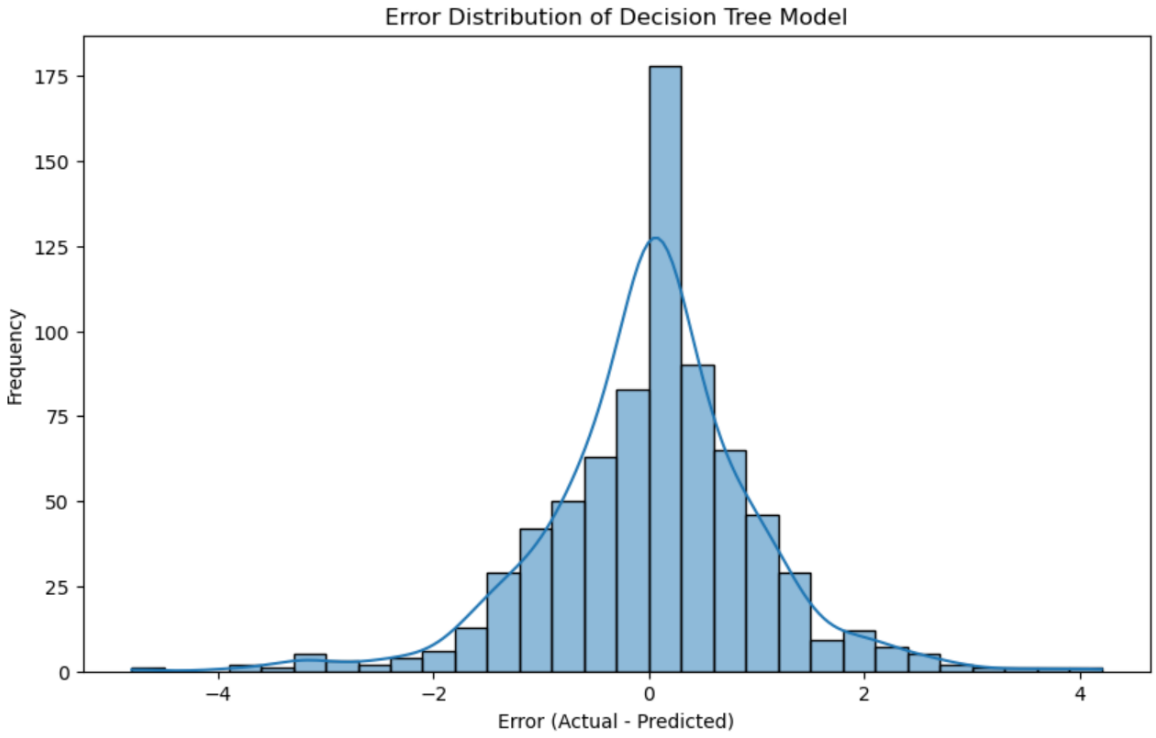


Figure 26: error distribution of decision tree

```
# Random Forest Model
error_distribution_rf = y_test - y_pred_rf  # Use y_pred_rf for Random Forest
plt.figure(figsize=(10, 6))
sns.histplot(error_distribution_rf, bins=30, kde=True)
plt.title('Error Distribution of Random Forest Model')
plt.xlabel('Error (Actual - Predicted)')
plt.ylabel('Frequency')
plt.show()
```

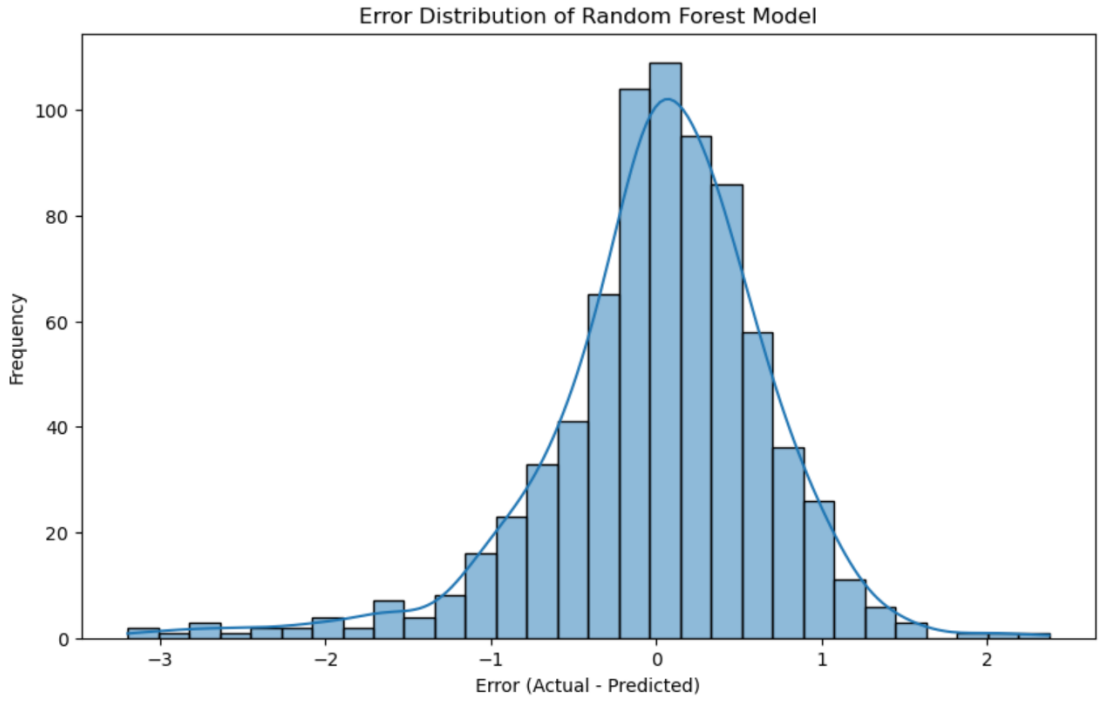


Figure 27: error distribution of random forest

PairPlot

```
# Plotting a pairplot to visualize the relationships between selected columns in the 'movie_ratings' dataframe.
sns.pairplot(movie_ratings[['imdb_score', 'budget', 'num_critic_for_reviews','duration','gross',
                    'num_voted_users','cast_total_facebook_likes', 'movie_facebook_likes',
                   'num_user_for_reviews','title_year']])
```

```
<seaborn.axisgrid.PairGrid at 0x15a44c290>
```

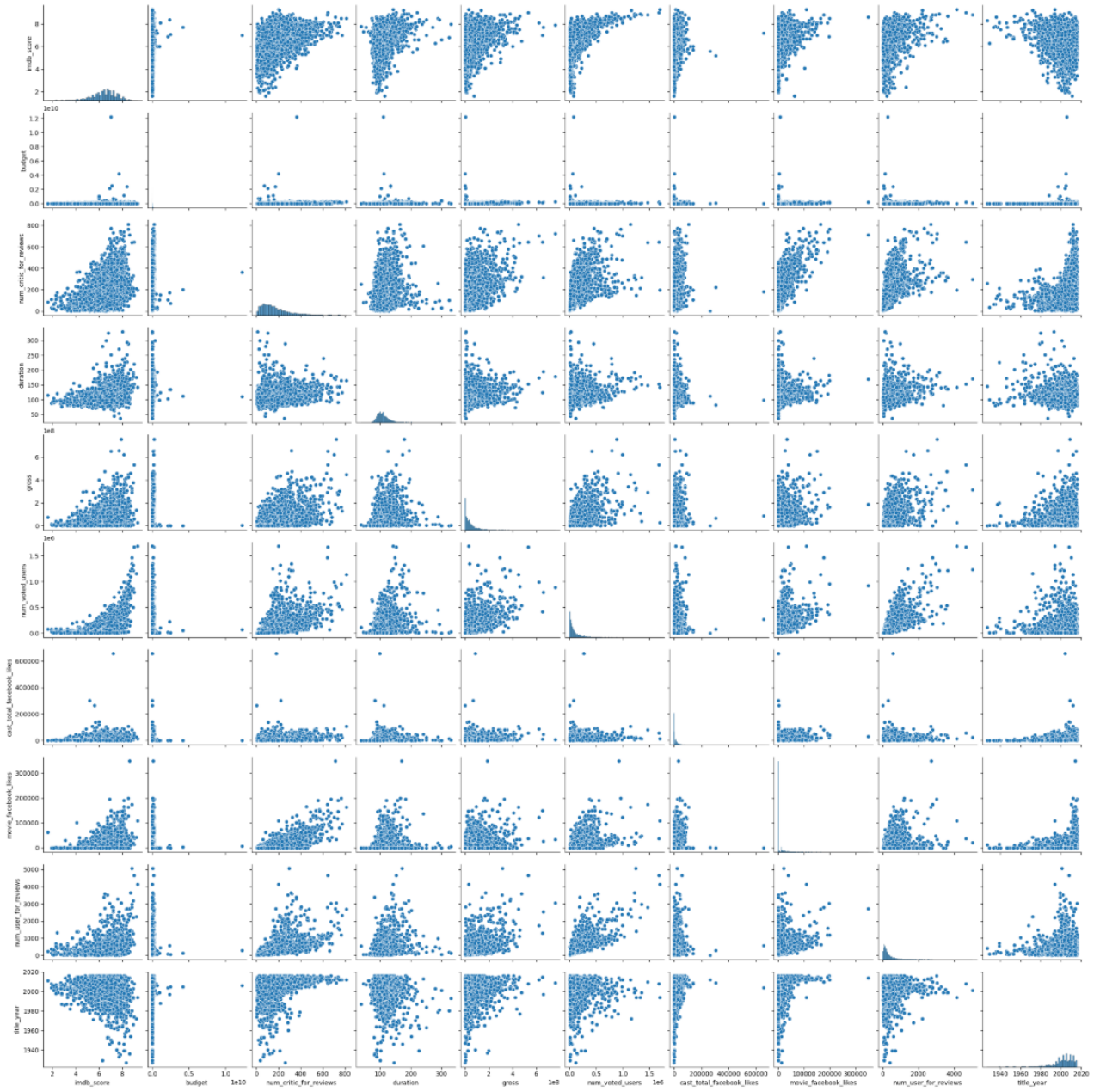


Figure 28: pairplot

Joint Plot imdb rating vs budget

```
# Creating a regression joint plot to visualize 'budget' vs 'imdb_score' with axis labels and a title.
sns.jointplot(x='budget', y='imdb_score', data=movie_ratings, kind='reg', height=10)
plt.xlabel('budget')
plt.ylabel('imdb_score')
plt.title('imdb rating vs budget')
```

```
Text(0.5, 1.0, 'imdb rating vs budget')
```

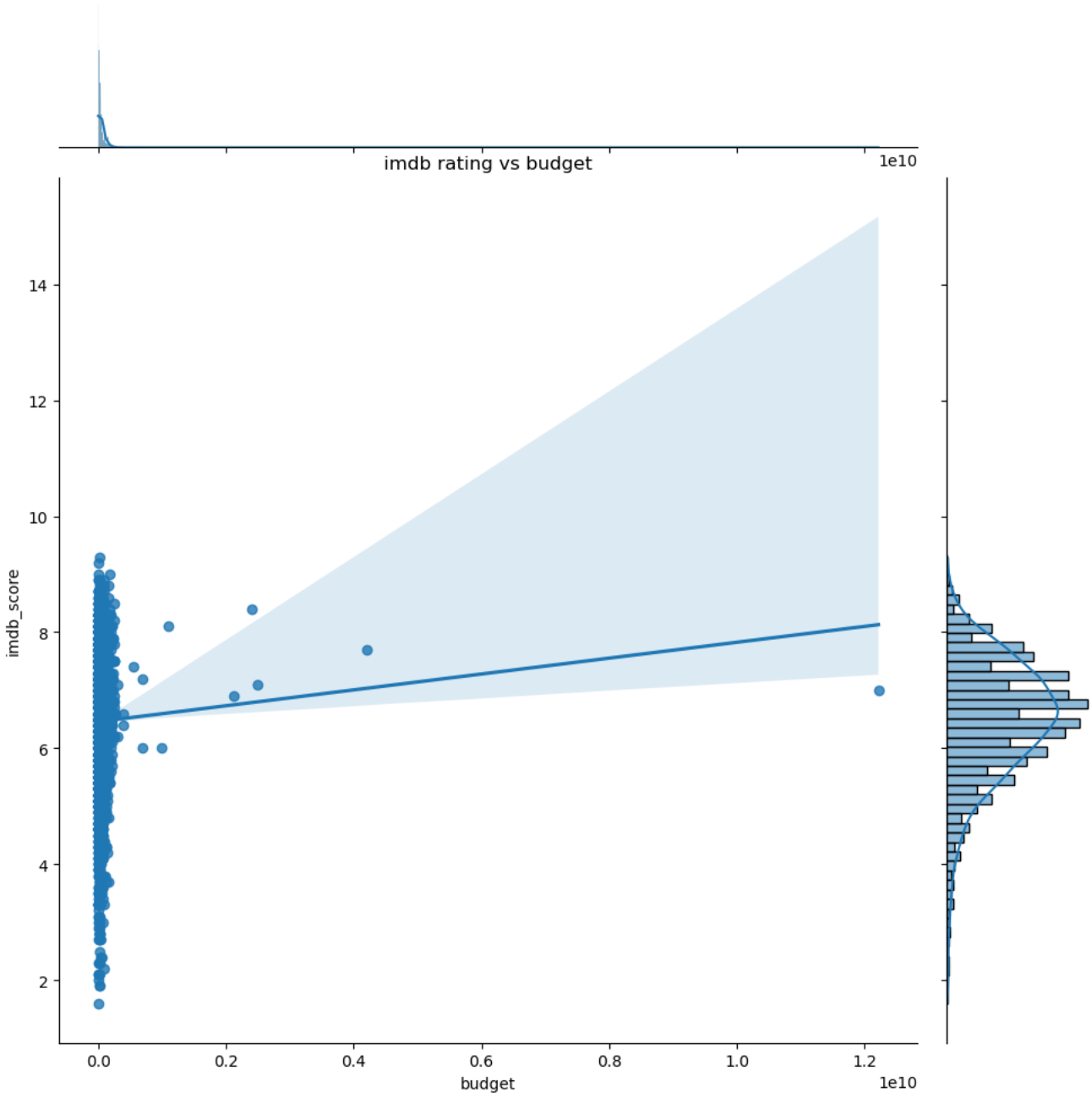


Figure 29: ***Joint Plot imdb rating vs budget***

Joint Plot imdb rating vs gross

```
# Creating a regression joint plot for 'gross' vs 'imdb_score' with labels and title.
sns.jointplot(x='gross', y='imdb_score', data=movie_ratings, kind='reg', height=10)
plt.xlabel('gross')
plt.ylabel('imdb_score')
plt.title('imdb rating vs gross')
```

```
Text(0.5, 1.0, 'imdb rating vs gross')
```

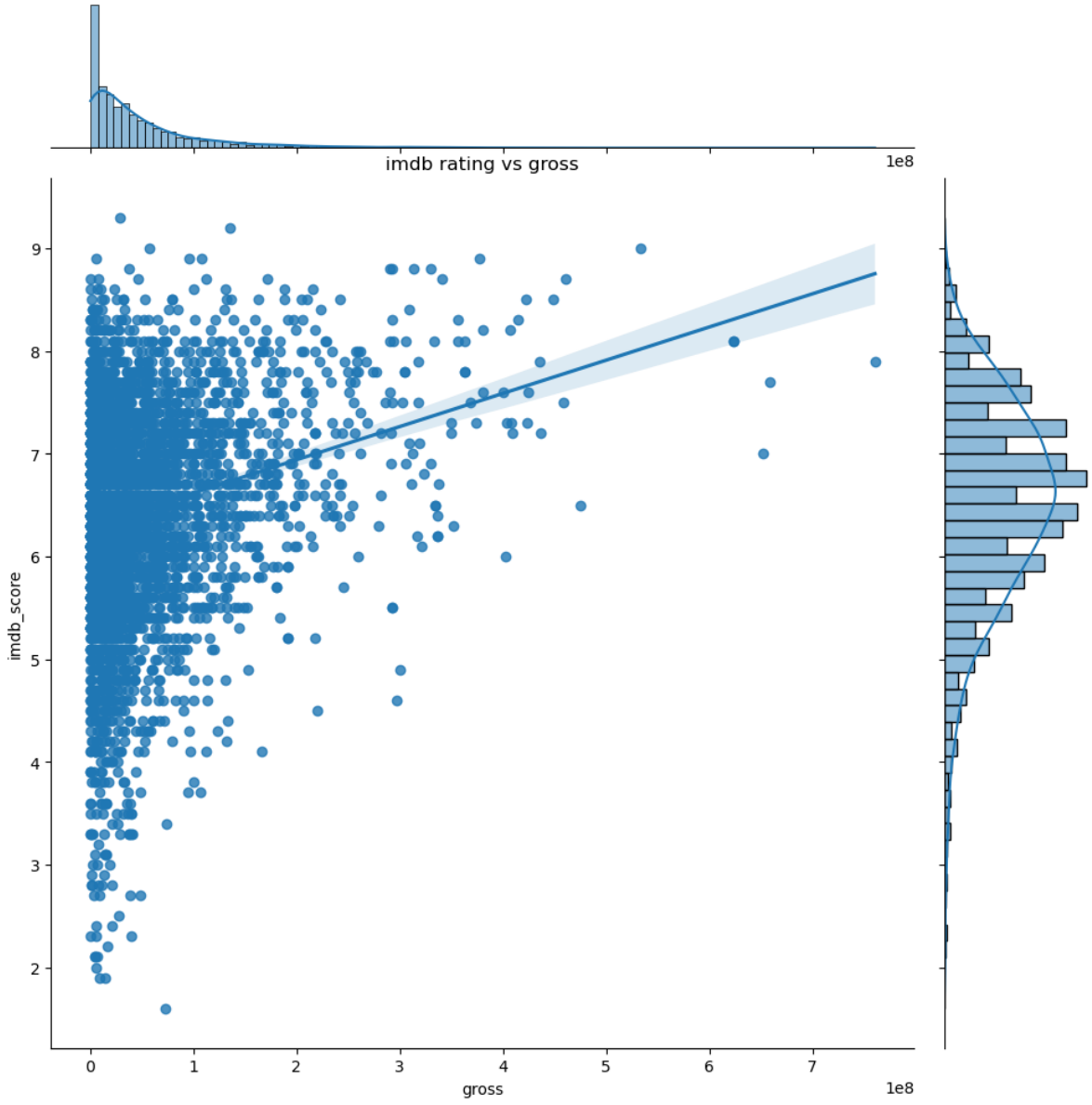


Figure 30: ***Joint Plot imdb rating vs gross***

Joint Plot imdb rating vs num_critic_for_reviews

```
[97]: # Creating a regression joint plot for 'num_critic_for_reviews' vs 'imdb_score' with labels and title.
sns.jointplot(x='num_critic_for_reviews', y='imdb_score', data=movie_ratings, kind='reg', height=10)
plt.xlabel('num_critic_for_reviews')
plt.ylabel('imdb_score')
plt.title('imdb rating vs num_critic_for_reviews')
```

```
[97]: Text(0.5, 1.0, 'imdb rating vs num_critic_for_reviews')
```

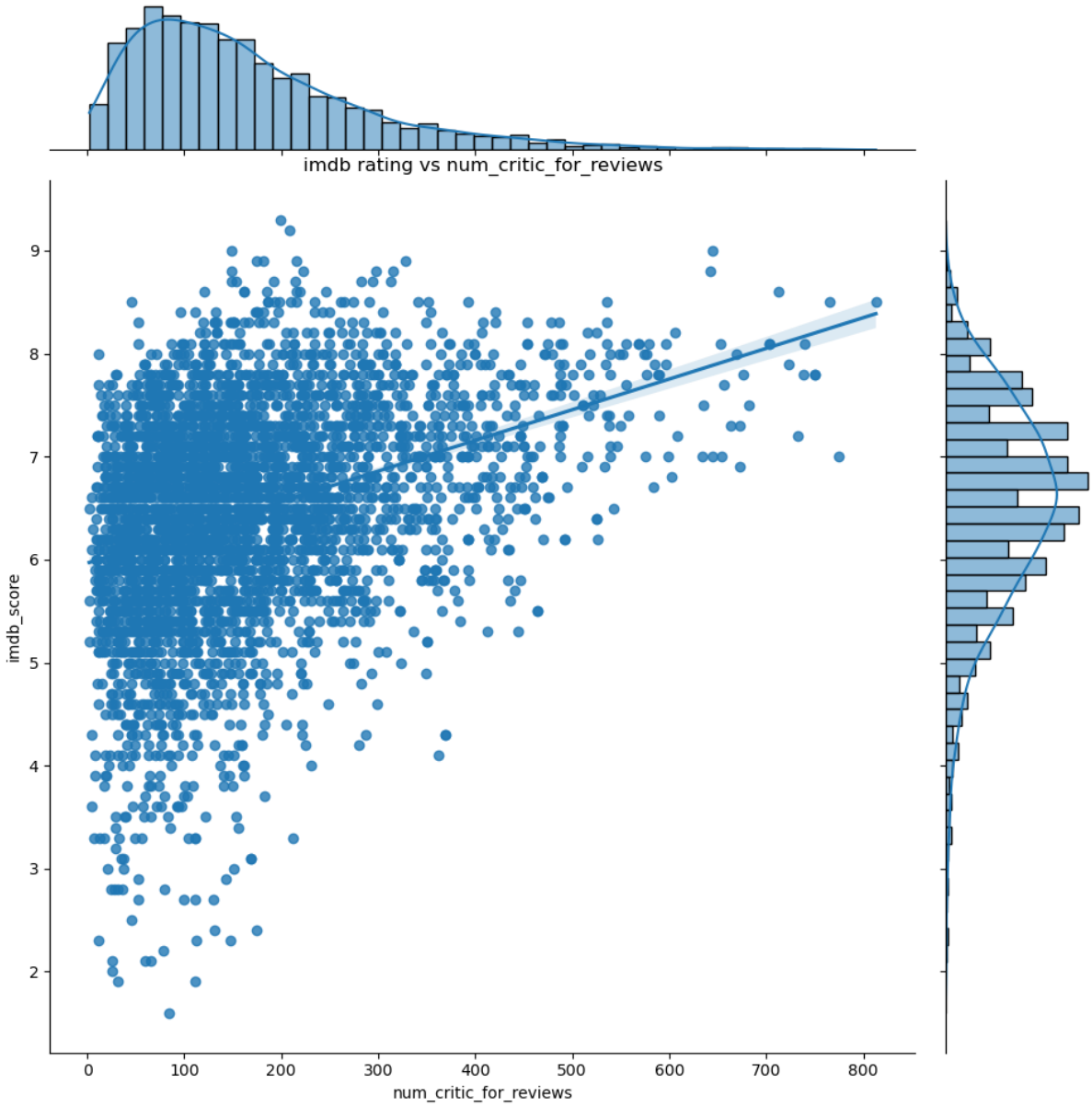


Figure 31: ***Joint Plot imdb rating vs num_critic_for_reviews***

Joint Plot imdb rating vs duration

```
# Creating a regression joint plot for 'duration' vs 'imdb_score' with labels and title.
sns.jointplot(x='duration', y='imdb_score', data=movie_ratings, kind='reg', height=10)
plt.xlabel('duration')
plt.ylabel('imdb_score')
plt.title('imdb rating vs duration')
```

```
Text(0.5, 1.0, 'imdb rating vs duration')
```

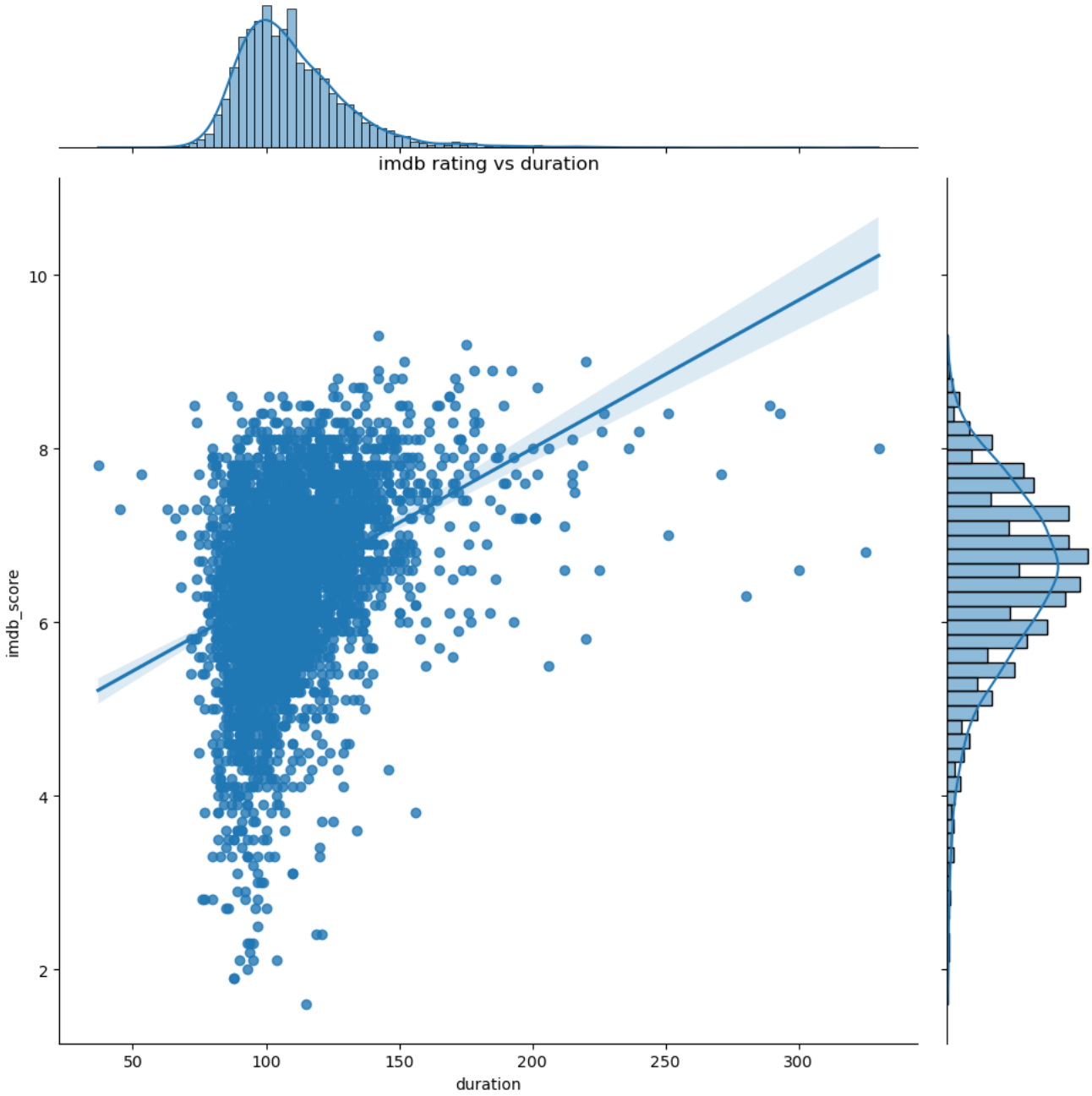


Figure 32: ***Joint Plot imdb rating vs duration***

Joint Plot imdb rating vs num_user_for_reviews

```
# Creating a KDE joint plot for 'num_user_for_reviews' vs 'imdb_score' with labels and title.
sns.jointplot(x='num_user_for_reviews', y='imdb_score', data=movie_ratings, kind='kde', height=10)
plt.xlabel('num_user_for_reviews')
plt.ylabel('imdb_score')
plt.title('imdb rating vs num_user_for_reviews')
```

```
Text(0.5, 1.0, 'imdb rating vs num_user_for_reviews')
```

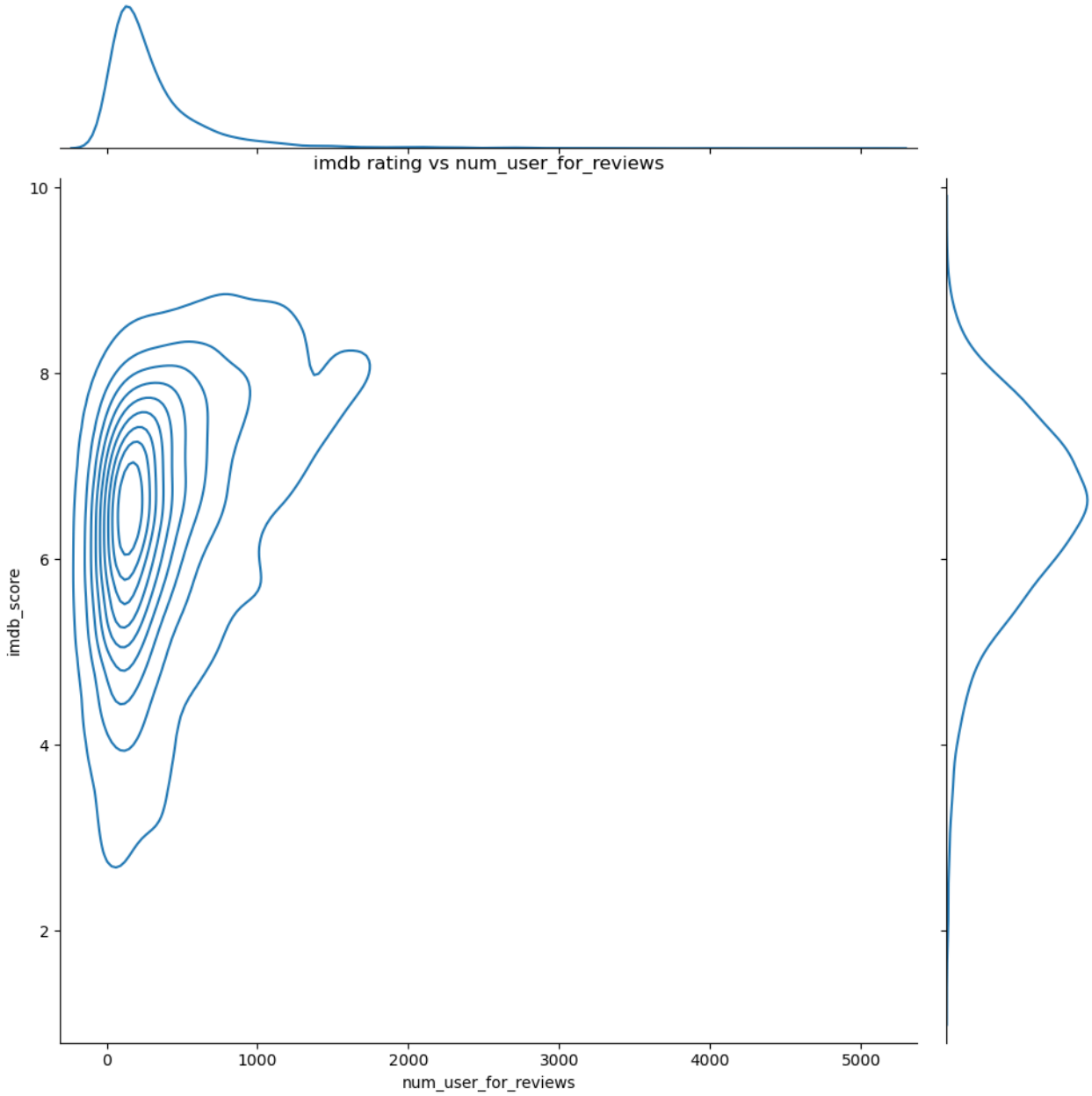


Figure 33: ***Joint Plot imdb rating vs num_user_for_reviews***

Joint Plot imdb rating vs title year

```
# Creating a regression joint plot for 'title_year' vs 'imdb_score' with labels and title.
sns.jointplot(x='title_year', y='imdb_score', data=movie_ratings, kind='reg', height=10)
plt.xlabel('title_year')
plt.ylabel('imdb_score')
plt.title('imdb rating vs title_year')
```

```
Text(0.5, 1.0, 'imdb rating vs title_year')
```

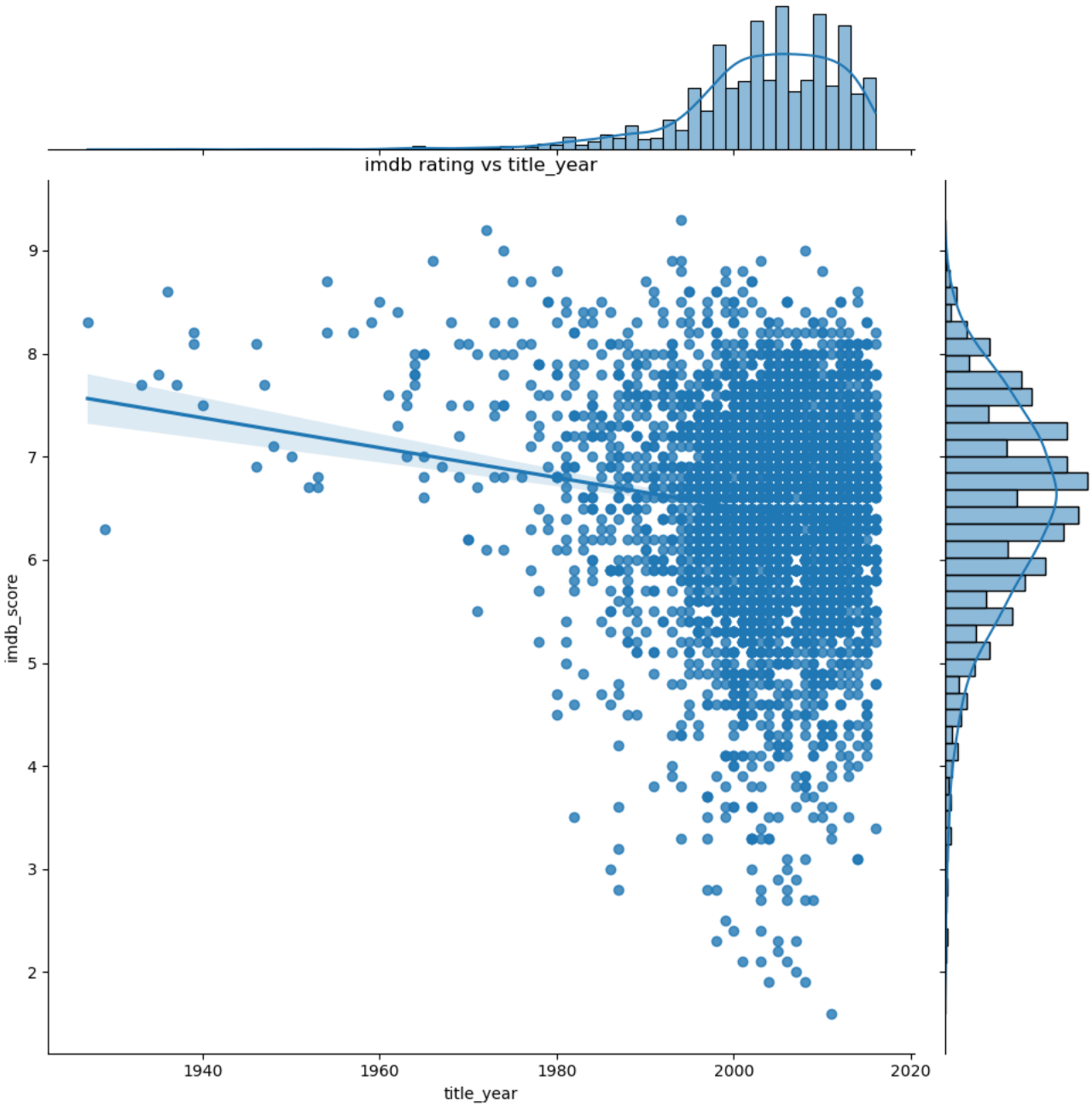


Figure 34: ***Joint Plot imdb rating vs title year***

HeatMap

```
# Calculate the correlation matrix
corr_matrix = movie_ratings.corr()

# Create a heatmap
plt.figure(figsize=(15, 10))
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

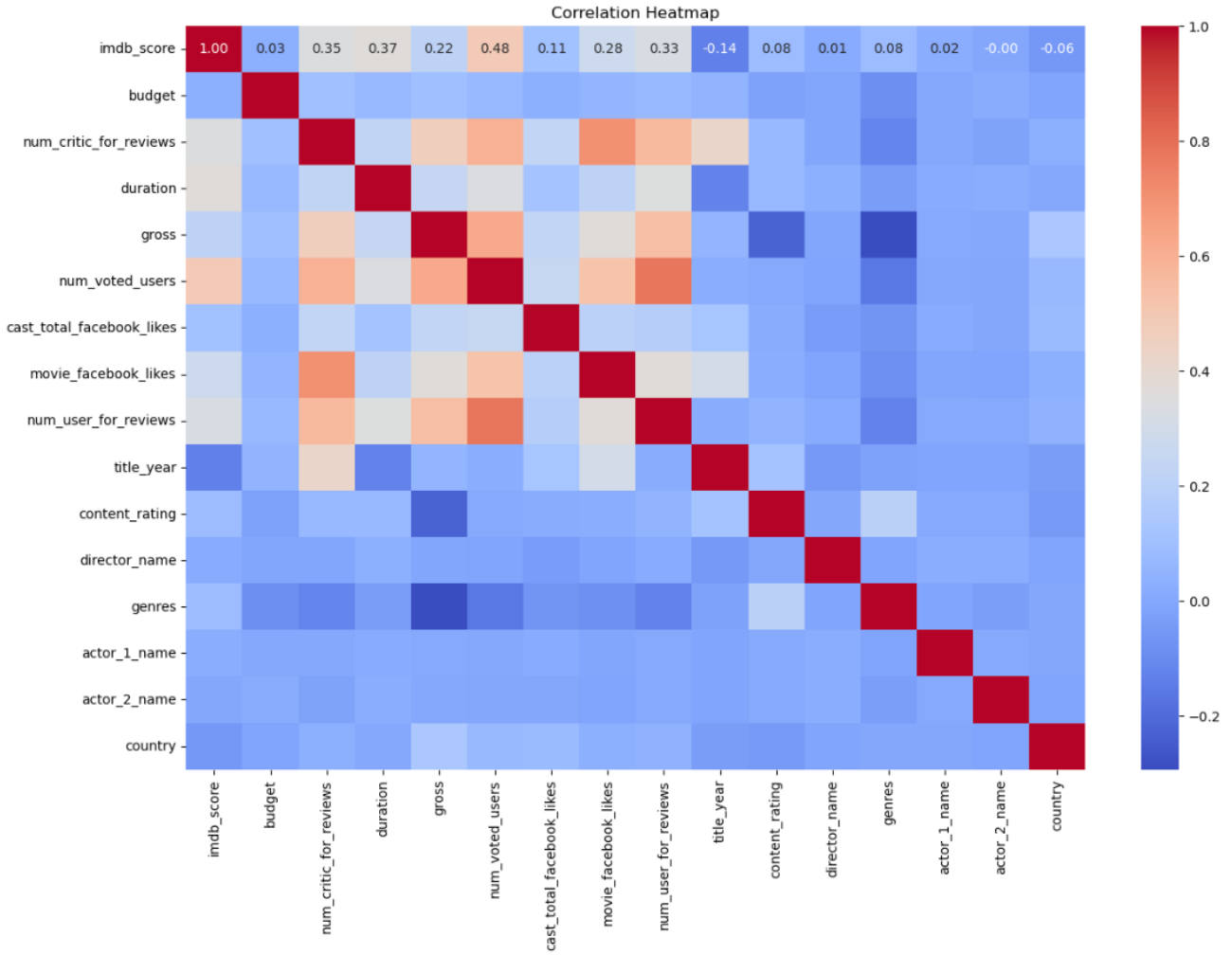


Figure 35: Heatmap

```
# Visualizing feature importance
feature_importances = pd.Series(random_regressor.feature_importances_, index=X.columns)
plt.figure(figsize=(15, 8))
feature_importances.nlargest(10).plot(kind='barh')
plt.title('Feature Importance')
plt.show()
```

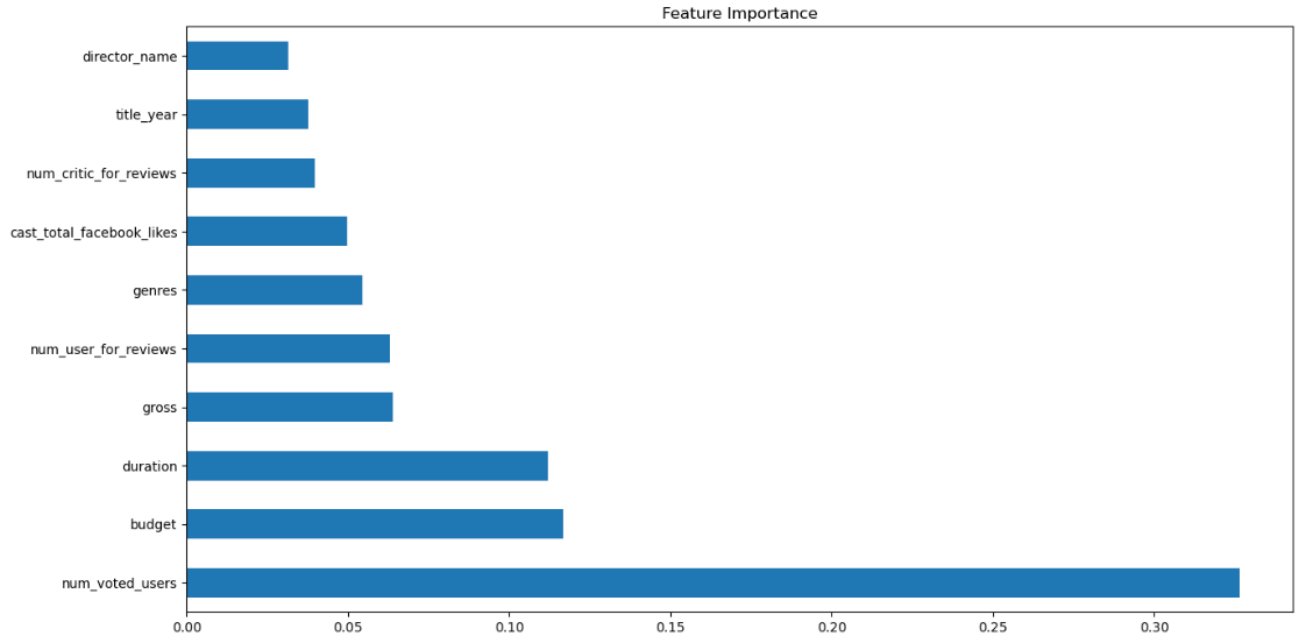


Figure 36: feature importance

```
# Correlation heatmap for predicted vs actual values
predicted_vs_actual = pd.DataFrame({
    'Actual': y_test,
    'Predicted_KNN': y_pred_knn,
    'Predicted_DT': y_pred_dt,
    'Predicted_RF': y_pred_rf
})

plt.figure(figsize=(8, 6))
sns.heatmap(predicted_vs_actual.corr(), annot=True, cmap='coolwarm', fmt=".2f", cbar=True)
plt.title('Correlation Heatmap: Actual vs Predicted (KNN, Decision Tree, Random Forest)')
plt.show()
```

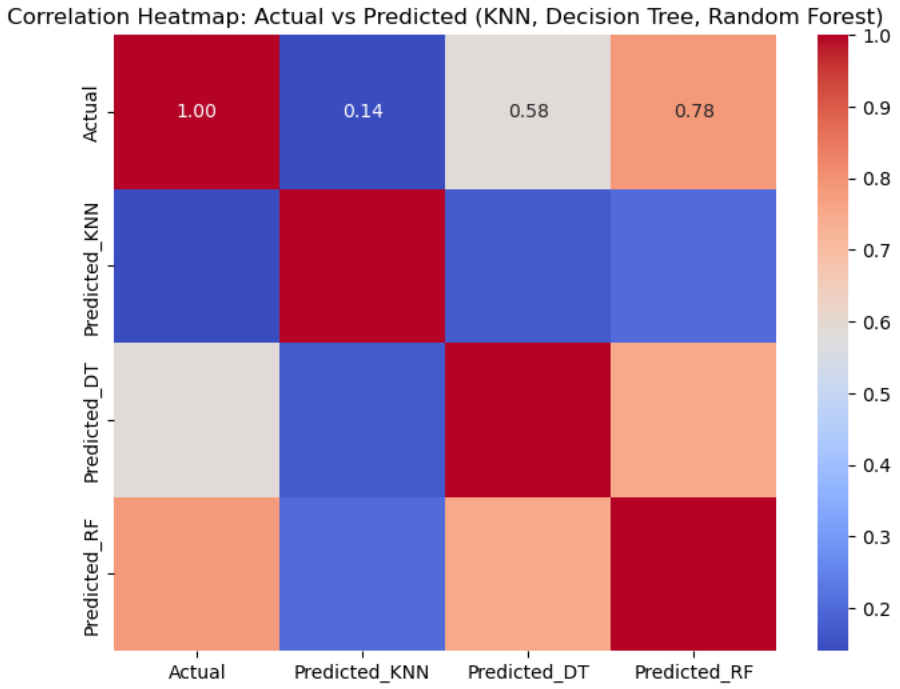


Figure 37: heatmap for predicted vs. actual values

This study is based on three regression models: Random Forest, Decision Tree, and K-Nearest Neighbors Regression. It discusses some information regarding movies with a focus on predicting the IMDB score for each movie. The summary below reports the results and their accuracy with other performance parameters.

Random Forest Regressor

- **Mean Absolute Error (MAE):** 0.4889
- **Mean Squared Error (MSE):** 0.4594
- **Root Mean Squared Error (RMSE):** 0.6778

- **R² Score:** 0.6095
- **Percentage Accuracy:** 92.42%

Decision Tree Regressor

- **Mean Absolute Error (MAE):** 0.6903
- **Mean Squared Error (MSE):** 0.9670
- **Root Mean Squared Error (RMSE):** 0.9834
- **R² Score:** 0.1781
- **Percentage Accuracy:** 89.29%

K-Nearest Neighbors Regressor

- **Mean Absolute Error (MAE):** 1.0519
- **Mean Squared Error (MSE):** 1.1066
- **Root Mean Squared Error (RMSE):** 1.0519
- **R² Score:** 0.0594
- **Percentage Accuracy:** 87.90%

Mean Absolute Error (MAE):

The Decision Tree Regressor displayed the least MAE estimate out of the three sample datasets, which was 0.69. This means that it has relatively low errors on average with the respect to the predictions.

Mean Squared Error and Root Mean Squared Error:

In simple words, the Random Forest Regressor showed the lowest MSE and RMSE values 0.4594 and 0.6778 respectively. Which shows it produced the smallest prediction errors

R² Score:

The Random Forest Regressor achieved highest **R² Score** at 0.6095 that indicates better variance in the target variable which is compared to the other models. The descisiom tree with **R²** of 0.1781 and KNN had the lowest which is 0.0594.

Percentage Accuracy:

The **Random Forest Regressor** has the highest accuracy at **92.42%**, which surpassed the **Decision Tree Regressor** (89.29%) and **Tuned KNN Regressor** (87.90%).

Initially KNN produced poor results **R^2 Score**: -0.13 so the KNN was tuned using Grid Search

Improved Results:

- **MAE**: Decreased from 0.88 to 0.74 (more accurate predictions).
- **MSE**: Decreased from 1.34 to 0.98.
- **R^2 Score**: Improved from -0.13 to **0.21**.
- **Percentage Accuracy**: Increased from 86.39% to **89.50%**.

Overall The random forest regressor has performed well than the other models in term of accuracy and error and variance. However descision tree also performed well mainly in minimizing the mean absolute error. Tunes KNN regressor showed decent performance and was behind the other two models in error metrics and accuracy.

The results show that the random forest regressor as the most reliable model for this dataset specially for tasks that requires precision and minimizing errors.

4. Conclusion

4.1. Analysis of Work Done

S.N	Task	Status
1	Research on Artificial Intelligence	Completed
2	Research on Machine Learning	Completed
3	Research on chosen topic	Completed
4	Research on Problem Domain	Completed
5	Similar system review and analysis	Completed

6	Research on algorithms	Completed
7	Research on solution of problem domain	Completed
8	Pseudocode	Completed
9	Flowchart	Completed
10	Implementation	Completed
11	Training data	Completed
12	Testing data	Completed
13	Comparing accuracy results	Completed

Table 1: Analysis of work done

4.2. How the solution addresses real world problems.

It can assist an average person uncertain of what movie to watch by predicting movie ratings. People usually spend a lot of time before finally deciding upon watching any movie by going through reviews and ratings. With the help of machine learning, we can predict the rating of any movie on the basis of its genre, director, actors, and many other attributes. This would save people time and help them select movies that are more likely to interest them. It can also help producers know what exactly viewers like in the movies so that better movies will be crafted in the future; it will not be perfect, but it can still give a good prediction based on available data.

4.3. Further Work

The model to predict movie ratings is good, but there is always room for improvement. For a start, more data should be gathered across different movies so that these predictions become more accurate. Alternatively, a few tweaks of the settings in the algorithms might produce more suitable results. I plan to improve this model further and deploy it as a mobile or web

application to make it available to more people, helping them choose their movies more conveniently.

5. Bibliography

How Artificial Intelligence is Reshaping the World- Blogs Year. (2021, Sept 17). From Medium: https://dermatologymarketing.medium.com/how-artificial-intelligence-is-reshaping-the-world-blogs-year-d8fa40b914ed

Duggal, N. (2024, Dec 27). *What is AI? Types & Examples of Artificial Intelligence*. From simplilearn: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/what-is-artificial-intelligence

StackCommerce. (2025, jan 3). *Python, AI, and robots—oh my!* . From msn: https://www.msn.com/en-us/money/other/python-ai-and-robots-oh-my/ar-AA1wVkdw?apiversion=v2&noservercache=1&domshim=1&renderwebcomponents=1&wcseo=1&batchservertelemetry=1&noservertelemetry=1

Brooke, B. (2025, jan 6). *CES 2025: Artificial intelligence to take centre stage at tech show but concerns much AI is just 'hype'*. From news.com.au: https://www.news.com.au/technology/innovation/inventions/ces-2025-artificial-intelligence-to-take-centre-stage-at-tech-show-but-concerns-much-ai-is-just-hype/news-story/984199e6741c2e7b632f96358b899755

Slack, S. (2024, june 18). *The Power and Potential of AI: How It's Changing Our World*. From lifewire: https://www.lifewire.com/the-power-of-ai-8659334#:~:text=AI%20is%20already%20showing%20us,from%20humans%20and%20from%20computers.

brown, s. (2021, april 21). *Machine learning, explained*. From MIT: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

bharat. (2023, sept 6). *Top 7 AI Applications*. From opencv: https://opencv.org/blog/top-7-ai-applications/

Pedamkar, P. (2023, Nov 16). *Applications of Machine Learning*. From educba: https://www.educba.com/applications-of-machine-learning/

C.Aggarwal, C. (2016). *Recommender Systems*. From Springer Nature Link: https://link.springer.com/book/10.1007/978-3-319-29659-3

Kniazieva, Y. (2022, April 14). *What Is a Movie Recommendation System in ML?* From labelyourdata: https://labelyourdata.com/articles/movie-recommendation-with-machine-learning#:~:text=An%20ML%20algorithm%20used%20for,likes%20by%20only%20one%20user.

Wilk, J. (2023, November 29). *How to Build a Movie Recommendation System Based on Collaborative Filtering*. From freeCodeCamp.org: https://www.freecodecamp.org/news/how-to-build-a-movie-recommendation-system-based-on-collaborative-filtering/

Soham. (2024, Dec 24). *Building a Movie Recommendation System with Machine Learning*. From Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/

wikipedia. (2025, jan 6). *List of datasets for machine-learning research* . From wikipedia: https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research

MPOY, G. (2019). *movie_metadata*. From Skip to content Kaggle: https://www.kaggle.com/datasets/bobirino/movie-metadata

Blog, G. (2024, Oct 14). *Movie Recommendation and Rating Prediction using K-Nearest Neighbors*. From analyticsvidhya: https://www.analyticsvidhya.com/blog/2020/08/recommendation-system-k-nearest-neighbors/

Navlani, A. (2024, June 27). *Decision Tree Classification in Python Tutorial*. From datacamp: https://www.datacamp.com/tutorial/decision-tree-classification-python

Sruthi. (2024, DEC 11). *Understanding Random Forest Algorithm With Examples*. From analyticsvidhya: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

Open Source . (2025). From anaconda: https://www.anaconda.com/open-source

Jupyter Notebook. (2024, oct 26). From geeksforgeeks: https://www.geeksforgeeks.org/jupyter-notebook/

Gunjan. (2024, dec 19). *Microsoft Excel for Data Analysis*. From analyticsvidhya: https://www.analyticsvidhya.com/blog/2021/11/a-comprehensive-guide-on-microsoft-excel-for-data-analysis/

Wong, F. M. (2024, april 28). *Python & NumPy*. From medium: https://foongminwong.medium.com/python-numpy-2e056fd687a4

V., D. (n.d.). *Python Data Analysis with Pandas and Matplotlib* . From ourcodingclub: https://ourcodingclub.github.io/tutorials/pandas-python-intro/

Introduction to Matplotlib. (2024, Dec 21). From geeksforgeeks: https://www.geeksforgeeks.org/python-introduction-matplotlib/

An introduction to seaborn. (2024). From seaborn: https://seaborn.pydata.org/tutorial/introduction.html

scikit-learn Machine Learning in Python. (2024). From scikit-learn: https://scikit-learn.org/stable/