# MOVIE RECOMMENDATION SYSTEM

**By**

**Kevinbhai Barvaliya (0785329)**

**Under the Guidance of,**

**Dr. Savita Seharawat**

**GitHub**

# Table of Contents

# Abstract

Nowadays, the recommendation system has made getting the things effortlessly that we need. The main purpose of Movie recommendation systems is to help movie enthusiasts by suggesting what movie to watch without the hassle to have to go through the time-consuming process of deciding from a large collection of movies which go up to millions is tedious and confusing. In this paper, we aim to minimize the human effort by suggesting movies based on the user's interests and preferences. To handle such problems, we introduced a model based on content-based approach and sentimental analysis. This system recommends movies by matching examples provided by the user to movie contents, which system derives from the movie director, cast, genre gathered from movie files, without using any human generated metadata also shows if the reviews are good or bad.


# Introduction

## 1. Literature review

The movie recommendation system tries to recommend movies to different users based on their interests. This helps the user save time surfing the web and searching for movies from the thousands that already exist. A content-based recommendation system describes the movies that may be recommended to a user, based on data set.The system predicts what movies the user may like, based on the movies the user has liked in the past. Recommendation systems can recommend movies based on one or a combination of two or more factors. When designing a movie recommendation system, various factors are considered, such as the genre of the movie, the director or the actors in it.In this paper, the recommender system is built on the basis of movie description, actor, director and genre. A column will be created that will be the sum of all the four attributes, and it will be the dominant factor for this movie

recommender system.

The recommendation system tries to predict what interests users and recommends related items that are likely to be of interest to them. The growth in the amount of information available online and the increase in the number of Internet users has resulted in an overload of information making it difficult to find the right information at the right time, and the recommendation system solves this problem by filtering the desired data from a large amount of information that is generated based on interests the user or his preferences.
matchmaking, and a lot more.

## 2. Problem statement

- The goal of this project is to recommend movies based on user interest.
- This project will recommend movies based on similar content like actors, directors name, genre etc.

## 3. Objective

- This project is based on content-based movie recommendation.
- Improving accuracy of recommendation system
- Recommendation form huge data set of movies
- User interactive recommendation system.

## 4. Methodology

The content-based approach proposed an integrative method, based on weighted similarity measure to construct a movie recommendation system. The proposed movie recommendation system gives finer similarity metrics and quality than the existing Movie recommendation system but the computation time which is taken by the proposed recommendation system

is more than the existing recommendation system. For computing similarity between the different movies in the given dataset efficiently and in least time and to reduce computation time of the movie recommender engine we used cosine similarity measure.

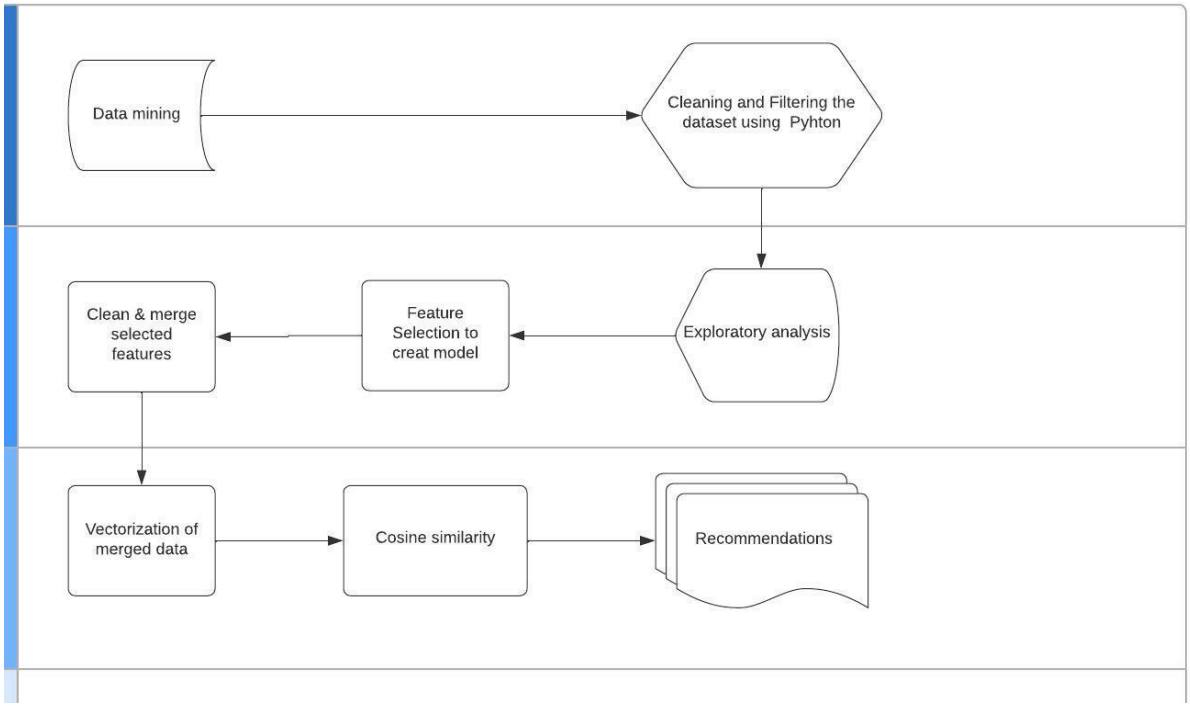



Figure 1: Methodology

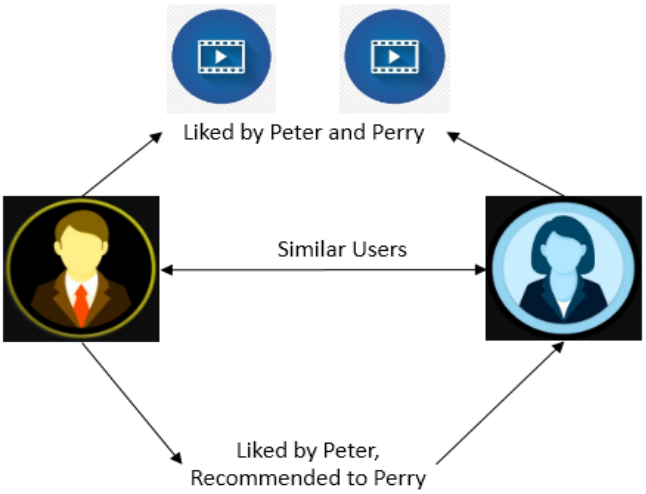
# Literature survey

Over the years, many recommendation systems have been developed using either collaborative, content based or hybrid filtering methods. These systems have been implemented using various big data and machine learning algorithms.

Recommender systems function with characteristic information and user-item interactions. Characteristic information is the information about the user and the items whereas user-item interaction is the information regarding ratings, the number of purchases, likes of the users, and many more. Based on this, the recommendation system can be developed using collaborative filtering, content-based filtering, or hybrid filtering **[Chen & Arslan (2008), Jalali, Mustapha, Sulaiman & Mamat (2010), Acilar & Arslan, (2009)].**

## 5. Collaborative Filtering.

This system identifies users with similar tastes and uses their opinion to recommend the same to other users with similar interests. It generates recommendations using information about rating profiles for different users or items. It has been implemented in different applications such as YouTube, Netflix, and Spotify. It is a widely used approach and is used as a part of the hybrid system

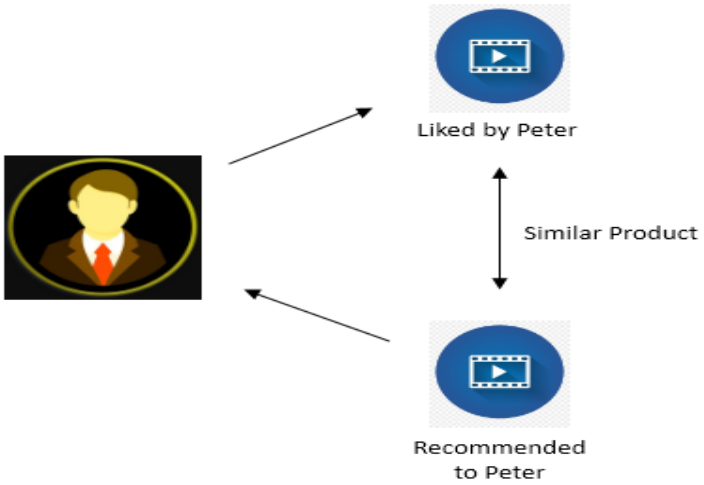



https://www.raco.cat/index.php/ELCVIA/article/download/373942/467477/

Figure 2: Collaborative Filter

## 6. Content-Based Filtering.

Content-based filtering methods are done based on user characteristics. This method is used in situations where data is known on an item such as name, location, or description and not on the user. It predicts the items based on the user's information and completely ignores contributions from other users as with the case of collaborative techniques. It uses the data that is provided by the user either explicitly or implicitly. When the user provides more content-based filtering mechanisms actions on the recommendations such as content-based recommender the engine becomes more and more accurate.




https://www.raco.cat/index.php/ELCVIA/article/download/373942/467477/

Figure 3: Content based Filter

## 7. Hybrid Approach.

A hybrid approach is a combination of collaborative filtering content-based filtering or any other approaches. Hybrid approaches can be implemented by making predictions separately on content-based and collaborative-based approaches and later combining them. It increases the accuracy and performance of the recommender systems.

# System specification

## 8. Hardware requirements

- A PC with windows OS
- Minimum 8 GB RAM
- Processor with 1.7-2.4gHz speed

## 9. Software specifications

- Anaconda
- PyCharm
- Python libraries

## 10. Software Requirements

### I. Anaconda distribution:

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.3

### II. Python libraries

For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as SKlearn, Numpy, pandas, Matplotlib, Stramlit framework, etc are needed.

**SKlearn:** It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

**NumPy**: NumPy is a general-purpose array-processing package. It

provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Pandas: Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Data frame.

**Beautiful Soup:** Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping

**Streamlit:** Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

# Implementation

## 11. Web scraping from IMDb

- To create a model, we scrapped 150k movies data from IMDb web site using python script.
- We fetched following features: Movie name, Rating, Released year, Metalcore, Genre, Movie description, Movie certificate, Votes, Goss, Director name, Actors name, Time, etc.
- Steps to implement web scraping in python to extract IMDb movies data.

  - Import required module.
    - → Beautiful Soup
    - → Requests
    - → Pandas

  - Access the HTML content from the webpage by assigning the URL and creating a soap object
  - Extract movie data from HTML content using Beautiful Soup.
  - After extracting data, create empty list to store data.
  - Convert list into Data Frame using Pandas library.
  - Export Data Frame in to CSV fire for further analysis

## I. Data Dictionary

| Field Name | Data Type | Description | Example |
|---|---|---|---|
| Movie name | Text | Name of movies | The Batman |
| Genre | Text | Category of movies | Action, Crime, Drama |
| Overview | Text | Short overview of movies | When a sadistic serial killer begins murder…. |
| Year | Numeric | Release year of movies | 2022 |
| Time | Numeric | Movie duration in minutes | 176 |
| Rating | Numeric | IMDb rating | 8.0 |
| Certificate | Text | Censer board certificate | PG |
| Votes | Numeric | Number of votes | 473729 |
| Gross (Million_$) | Numeric | Gross earning of movies in millions dollar | 232.64 |
| Meta score | Text | Ratings of the world's most respected critics | 72 |
| Director | Text | Director name | Matt Reeves |
| Actors | Text | List of actors name | Robert Pattinson, Zoë Kravitz, Jeffrey Wright |

Table 1: Data Dictionary

# 12. Data cleaning / Data preparation

Data cleaning is the process of ensuring accurate, consistent, and usable data. You can clean data by identifying errors or corruptions, correcting, or deleting them, or manually processing data as needed to prevent them from recurring. Most aspects of data cleaning can be done with software tools, but some must be done manually. This can be a daunting task, but it is an important part of managing company data.

## Data cleaning steps

### I. Remove irrelevant data

Firstly, we need to figure out irrelevant data from IMDb dataset and then find out way to manage that data. In our data set there are some columns (**Overview, Year, Time, Votes, Director, Overview**) which contains irrelevant data.

- **Overview column** contains lots of forwarding spaces, so we remove those spaces using strap() function.
- There are some roman data instead of year in the **Year column**, so we replace roman number will none data
- Replacing comma commas with none from **Time column.**
- some directors' names merged into the actor's column, so we must move those director name from actor column to director column.
- After cleaning data, we are exporting Data Frame as CSV.

### II. Remove duplicate values

- In IMDb dataset there were 43 duplicate records and I removes using **drop_duplicates()** function.

### III. Deal with missing data

| Column name | Number of null values |
|---|---|
| Movie name | _ 0 |
| Genre | 7973 |
| Overview | 0 |
| Year | 200 |
| Time | 31192 |
| Rating | 50631 |
| Certificate | 136032 |
| Votes | 50631 |
| Gross (Million_$) | 143242 |
| Meta score | 140588 |
| Director | 535 |
| Actors | 5757 |

Table 2: Missing values

- We can see that there are lots of null values in our data set. The Gross column contains almost 143k null values, if we remove rows that contain null values then we would be left with only a few rows, so for now, we are keeping all null values. We will remove null values as required

# 13. Exploratory analysis

## I. Insides about data-set

```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149922 entries, 0 to 149921
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Movie_name        149922 non-null  object
 1   Genre             141949 non-null  object
 2   Overview          116749 non-null  object
 3   Year              149722 non-null  float64
 4   Time              118730 non-null  float64
 5   Rating            99291 non-null   float64
 6   Certificate       13890 non-null   object
 7   Votes             99291 non-null   float64
 8   Gross(Million_$)  6680 non-null    float64
 9   Metascore         9334 non-null    float64
 10  Director          145597 non-null  object
 11  Actors            144165 non-null  object
dtypes: float64(6), object(6)
memory usage: 13.7+ MB
```

Figure 4 : information of data set

- Using **info()** function we can see number of record in the data frame, non-null values in each column, data type of columns, number of columns, size of data frame.

| | Year | Time | Rating | Votes | Gross(Million_$) | Metascore |
|---|---|---|---|---|---|---|
| **count** | 149722.000000 | 118730.000000 | 99291.000000 | 9.929100e+04 | 6680.000000 | 9334.000000 |
| **mean** | 2014.423131 | 96.054317 | 5.874749 | 5.425245e+03 | 23.508692 | 55.387079 |
| **std** | 4.720293 | 44.357386 | 1.433923 | 4.095977e+04 | 59.988082 | 17.136397 |
| **min** | 2005.000000 | 1.000000 | 1.000000 | 5.000000e+00 | 0.000000 | 1.000000 |
| **25%** | 2011.000000 | 82.000000 | 5.000000 | 2.500000e+01 | 0.050000 | 43.000000 |
| **50%** | 2015.000000 | 93.000000 | 6.000000 | 1.020000e+02 | 0.620000 | 56.000000 |
| **75%** | 2018.000000 | 107.000000 | 6.800000 | 5.620000e+02 | 18.360000 | 68.000000 |
| **max** | 2022.000000 | 10062.000000 | 10.000000 | 2.558124e+06 | 936.660000 | |

Table 3: Stats information

- Describe() function show the description of data set, in above screen shot we can see that it showing mean, standard deviation, min, max, Q1, Q2 and Q3 of the numeric columns.

## II. Correlation

| | Year | Time | Rating | Votes | Gross(Million_$) | Metascore |
|---|---|---|---|---|---|---|
| **Year** | 1.000000 | 0.003399 | 0.014505 | -0.039753 | 0.033765 | 0.097291 |
| **Time** | 0.003399 | 1.000000 | 0.070770 | 0.061983 | 0.168413 | 0.171672 |
| **Rating** | 0.014505 | 0.070770 | 1.000000 | 0.089860 | 0.176022 | 0.651316 |
| **Votes** | -0.039753 | 0.061983 | 0.089860 | 1.000000 | 0.725506 | 0.163125 |
| **Gross(Million_$)** | 0.033765 | 0.168413 | 0.176022 | 0.725506 | 1.000000 | 0.057768 |
| **Metascore** | 0.097291 | 0.171672 | 0.651316 | 0.163125 | 0.057768 | 1.000000 |

Table 3: Correlation

```
plt.figure(figsize=(10,7))
sns.heatmap(movies.corr())
```

```
<AxesSubplot:>
```

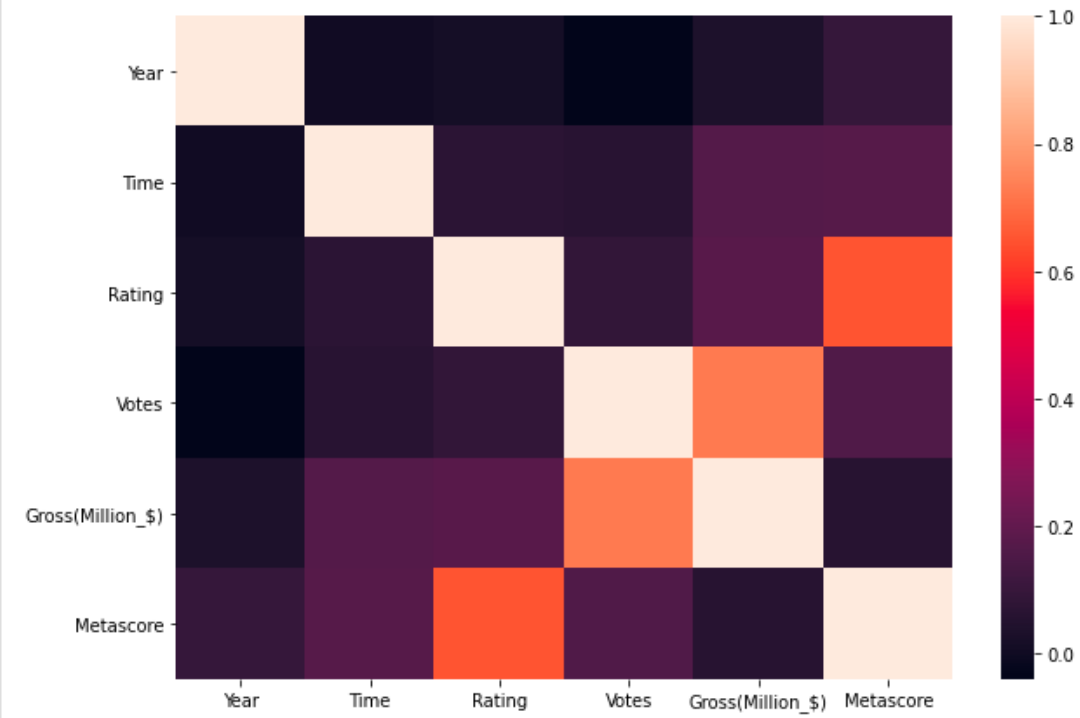



Figure 5: Correlatrion

- Using heat map we can see correlation between columns.
- Here, we can see that, correlation between rating and meta score, votes and gross is good, while other columns has very low correlation.

## III. Graphical EDA

- **Top voted Movies**

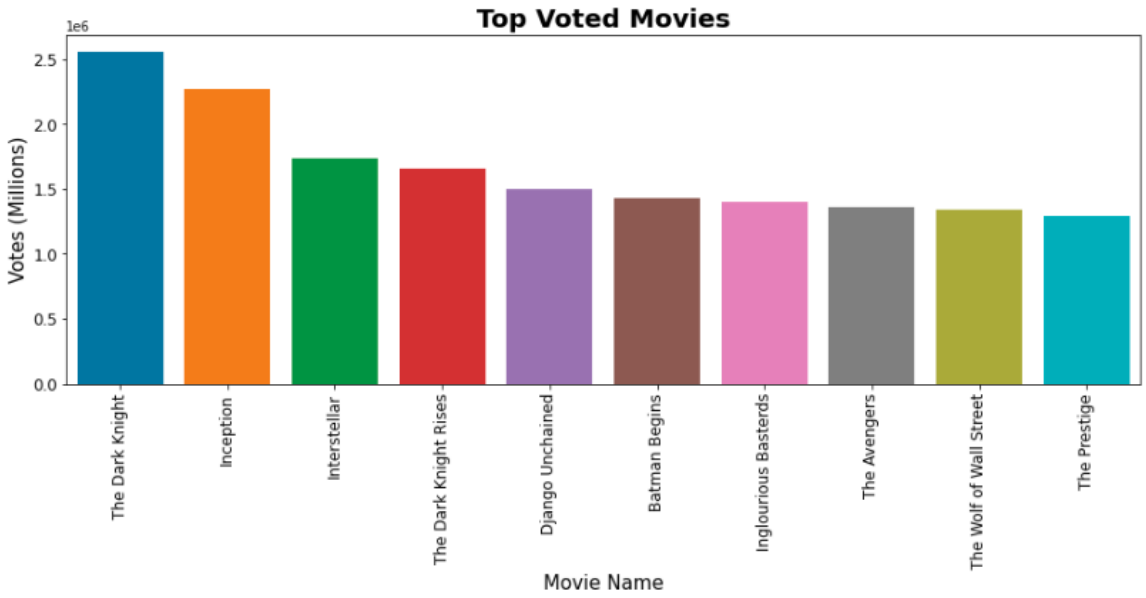



Figure 5: Top voted movies

- We can see in the above graph , 2.5 millions peoples voted to the dark night movie

- **Top rated Movies**

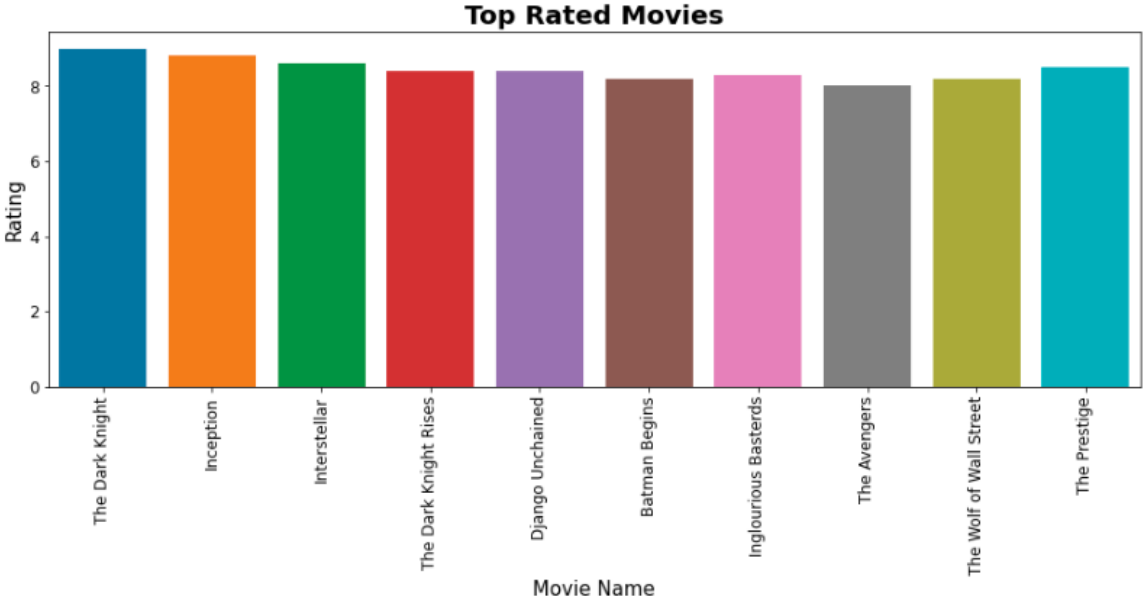



Figure 6: Top voted movie

- Here we can see top rated 10 movies, all ten top rated movies are lies between 8 to 9 rating.

- **Meta score of top-rated movies**

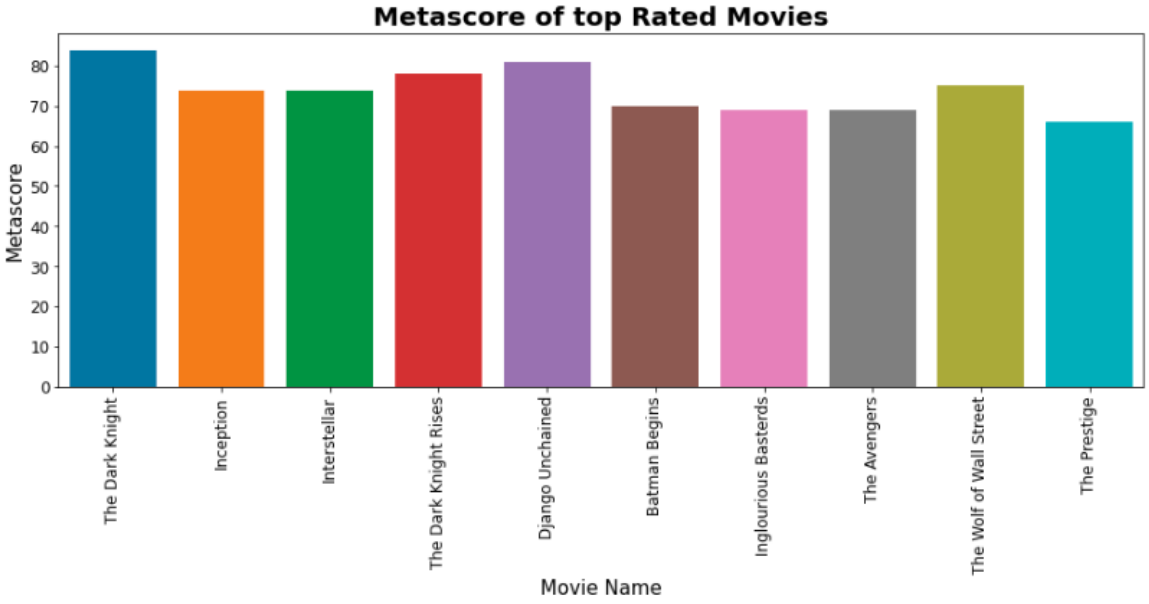



Figure 7: Meta score of top rated movies

- **Movies with highest gross**

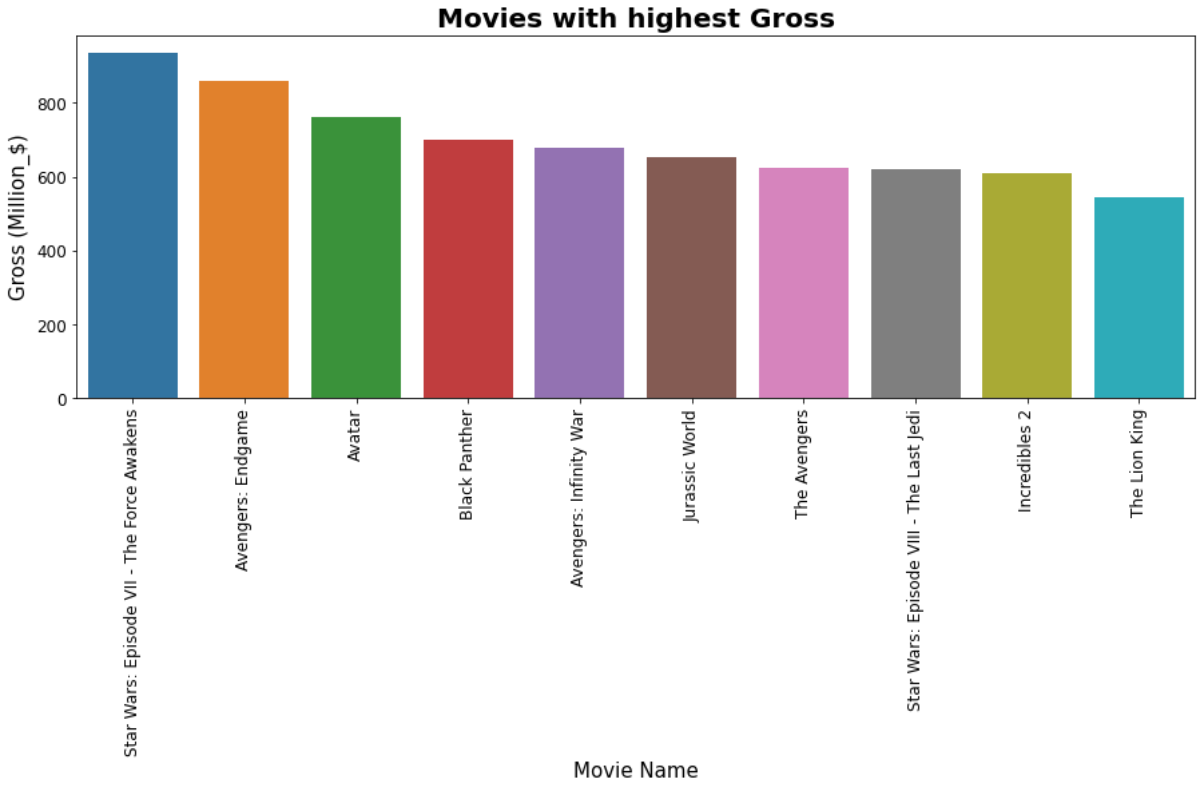


Figure 8: Movies with highest gross

- **Director's name who directed maximum movies**

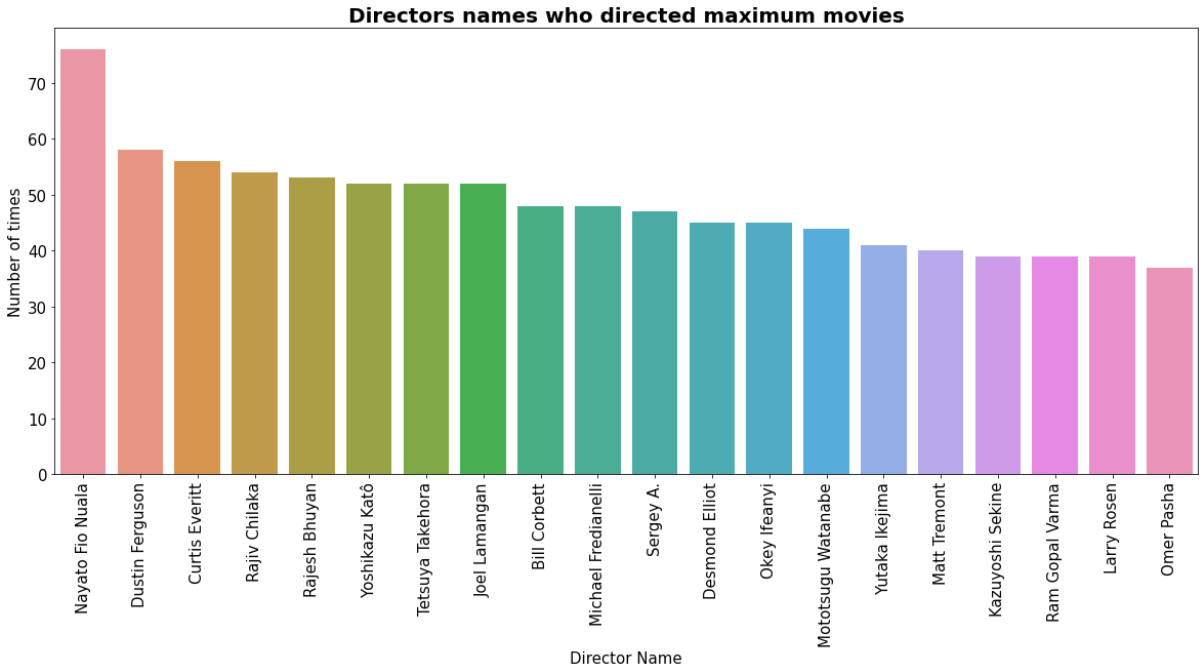



Figure 9: Top directors

- **Frequency of Genre in movies**

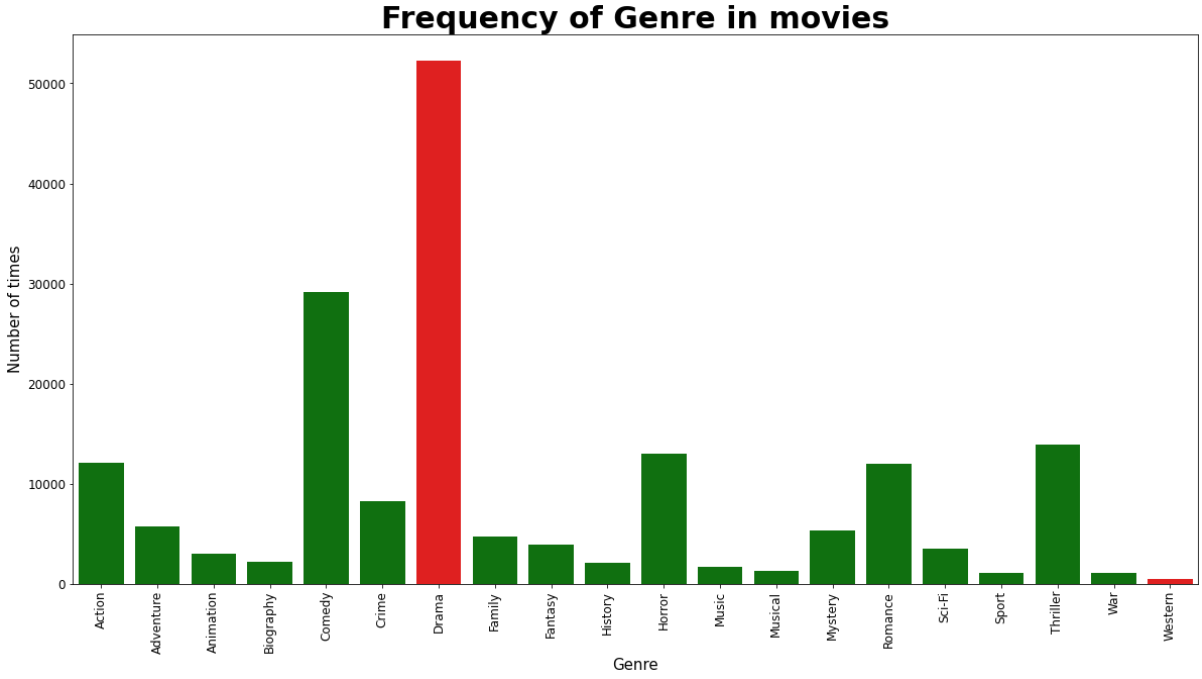



Figure 10: Frequency of genre

- **Highest average Rating by Genre.**

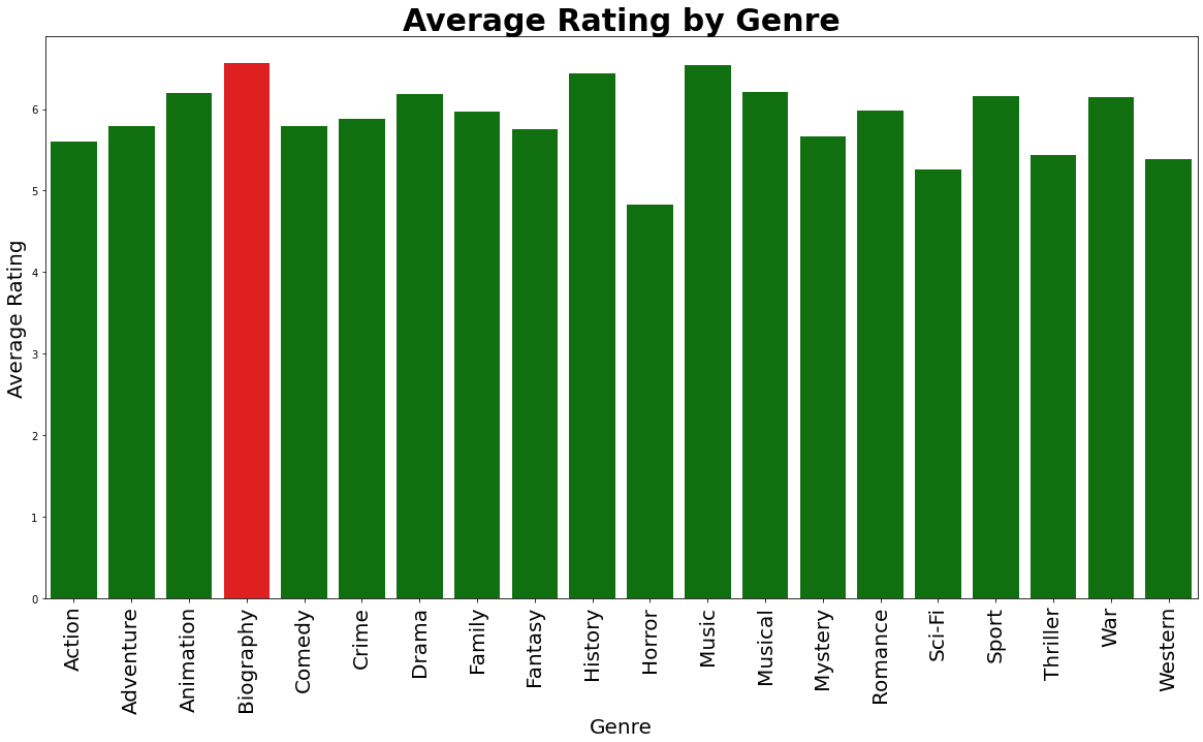



Figure 11: Highest average rating by genre

- **Lowest average Rating by Genre.**

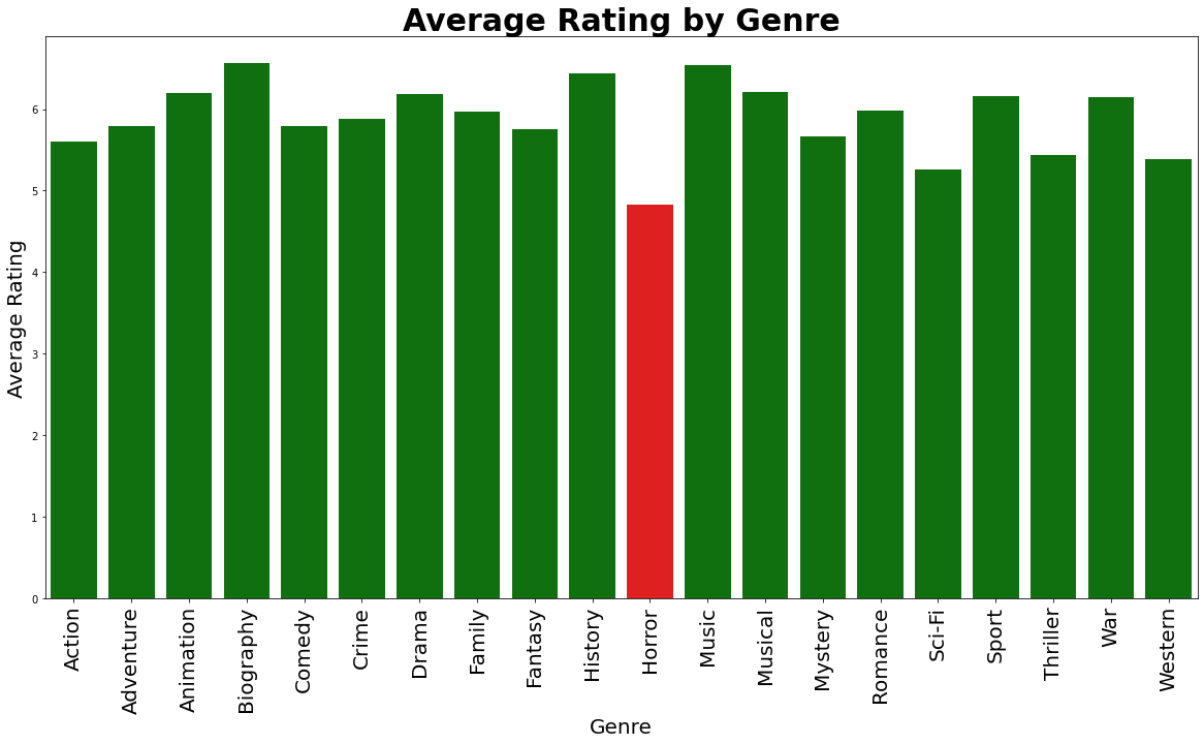


Figure 12: Lowest average rating by genr

- **Genre with highest average gross.**

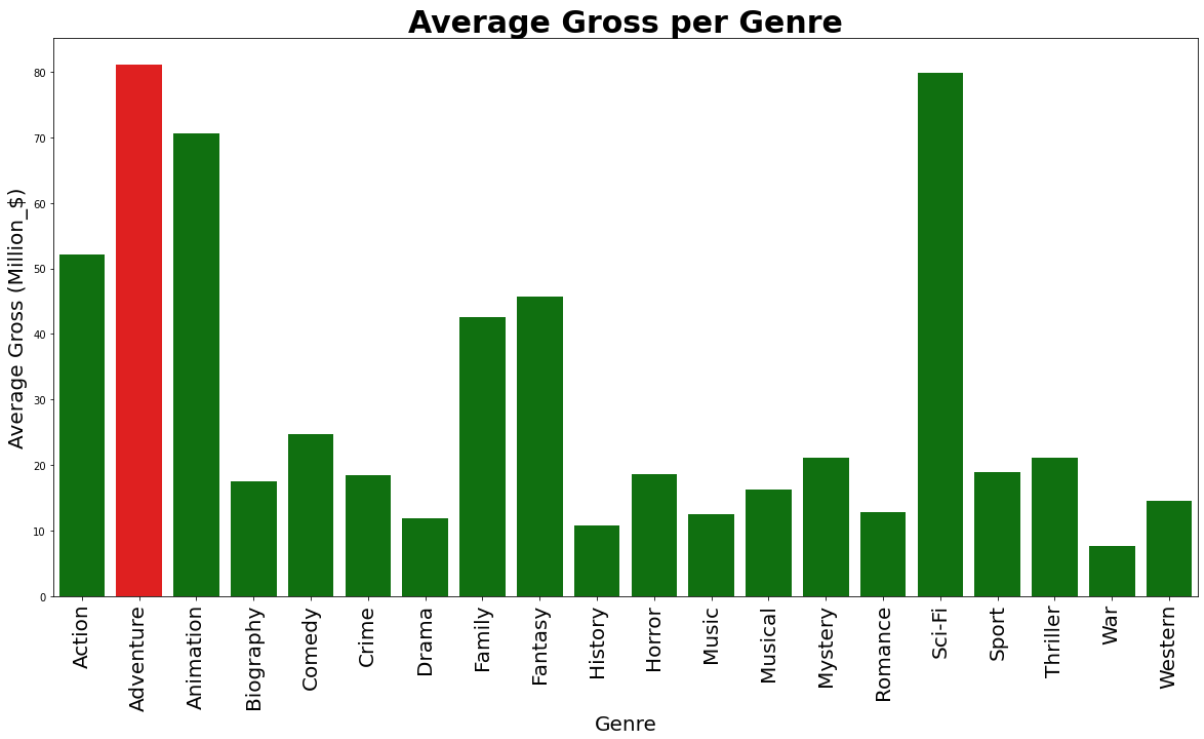



Figure 13: Highest average gross by genre

- **Genre with lowest average gross.**

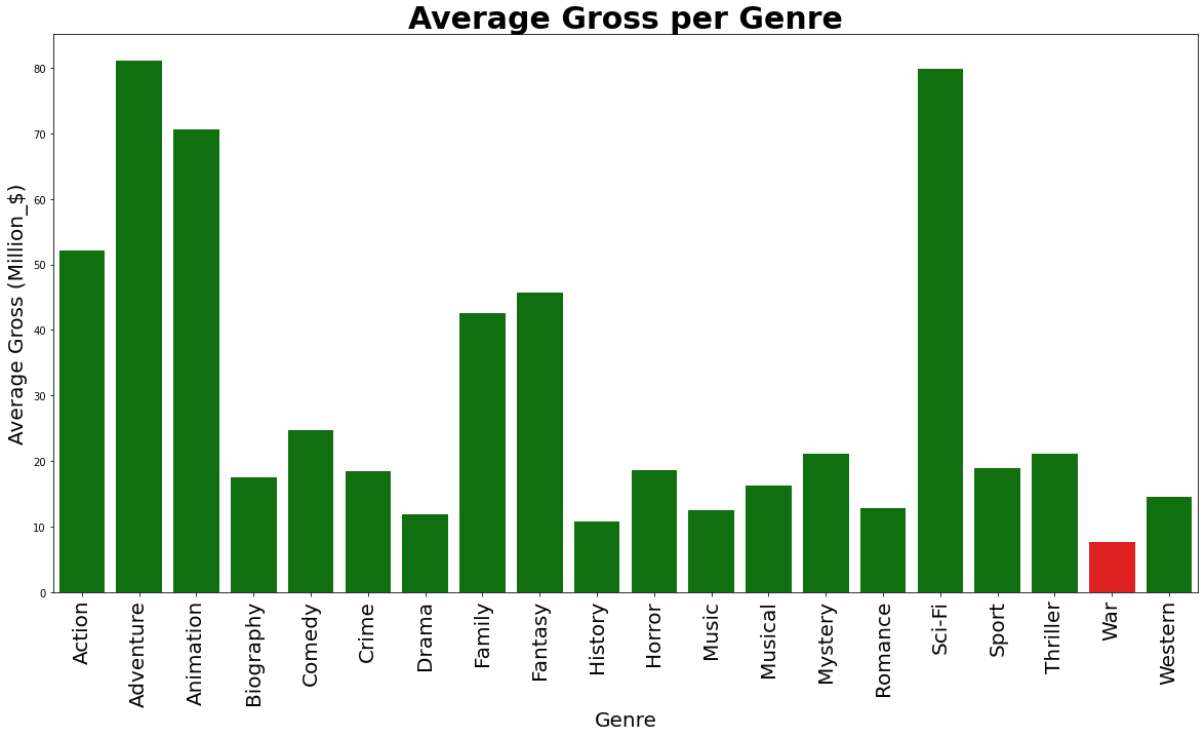


Figure 14: Lowest average gross by genre

- **Average meta score by genre**

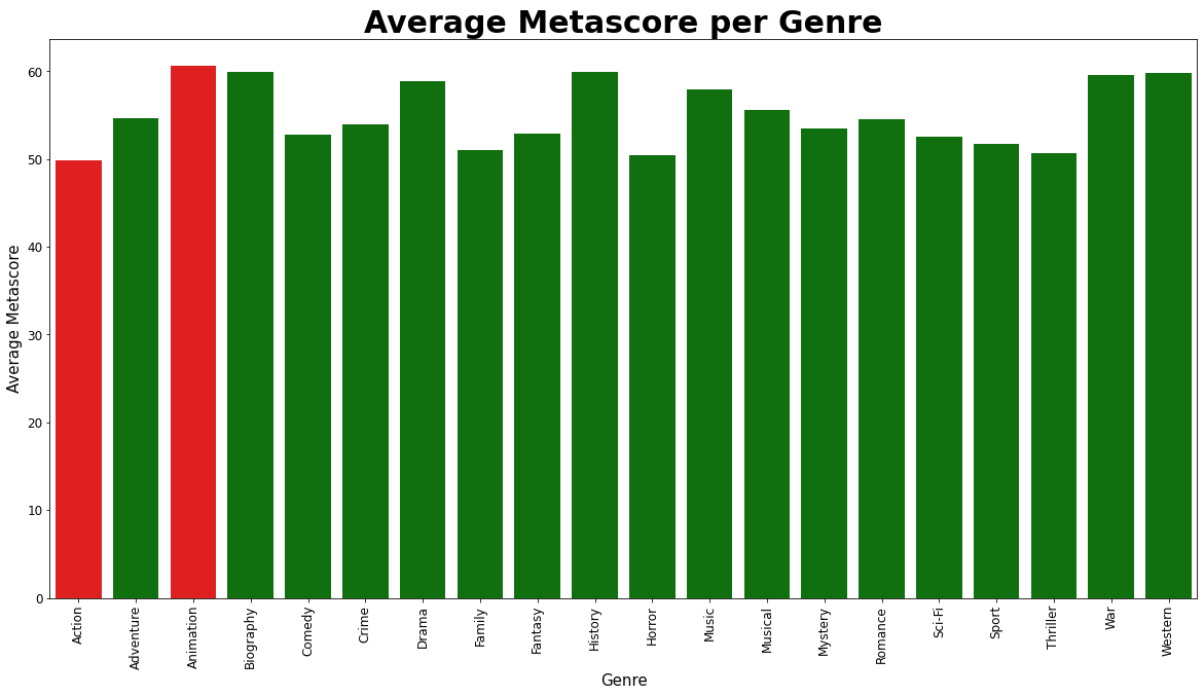



Figure 15: Average meta score by genre

# 14. Implementing Recommendation system

## I. Feature selection

- We are going to implement content-based recommendation system, so we must select features which contains text.
- There are total five feature which have text data, so we remove columns which doesn't have text data type.

| | Movie name | Genre | Overview | Director | Actors |
|---|---|---|---|---|---|
| **47** | The Dark Knight | Action Crime Drama | When the menace known as the Joker wreaks havo... | ChristopherNolan | ChristianBale HeathLedger AaronEckhart Michael... |
| **102** | Inception | Action Adventure Sci-Fi | A thief who steals corporate secrets through t... | ChristopherNolan | LeonardoDiCaprio JosephGordon-Levitt ElliotPag... |
| **83** | Interstellar | Adventure Drama Sci-Fi | A team of explorers travel through a wormhole ... | ChristopherNolan | MatthewMcConaughey AnneHathaway JessicaChastai... |
| **172** | The Dark Knight Rises | Action Crime Drama | Eight years after the Jokers reign of anarchy ... | ChristopherNolan | ChristianBale TomHardy AnneHathaway GaryOldman |
| **143** | Django Unchained | Drama Western | With the help of a German bountyhunter a freed... | QuentinTarantino | JamieFoxx ChristophWaltz LeonardoDiCaprio Kerr... |

Table 4: Features

## II. Clean selected features

- **Genre:** most of movies has more than one genre separated with comma, so we will replace comma with single space, because special characters might be affect to accuracy of recommendations.

- **Overview:** This column contains short description about movies. This column also contains some special characters, so we will use python code which remove all the special characters and keep only text and numbers.

- **Director:** First name and last name of director's are separated with single space. We will remove this space and merge both as a single word. If we keep it separate then it might give less accurate recommendation because many directors have same first name as well as last name, it might be confusing to find similarity.

- **Actors:** This column has same issue as director column**,** we will merge first name and last name of actors. In addition, this column contains multiple actors name and each name separated with comma, so we will replace this comma with none same as genre column.

- **Merging columns:** After cleaning all selected feature we will merge column so we can apply TFIDF.

## III. TF-IDF CountVectorizer

TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction.

Term frequency is defined as the number of times a word (i) appears in a document (j) divided by the total number of words in the document.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

| | action | adventure | american | based | begins | best | boy | brother | city |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9995** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 5: TF-IDF vector

## IV. Cosign Similarity

- Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.

- The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity

- Mathematically, **Cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space.**

- In this context, the two vectors I am talking about are arrays containing the word counts of two documents.

- As a similarity metric, how does cosine similarity differ from the number of common words?

- When plotted on a multi-dimensional space, where each dimension corresponds to a word in the document, the cosine similarity captures the orientation (the angle) of the documents and not the magnitude. If you want the magnitude, compute the Euclidean distance instead.

- How to calculate cosine similarity

$$similarity(A, B) = \cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$$

- Example of cosine similarity matrix

```
Array ([[ 1.        ,  0.40824829,  0.40824829],
       [ 0.40824829,  1.        ,  0.33333333],
       [ 0.40824829,  0.33333333,  1.        ]])
```

- To find similarity between movie, we calculate similarity of vectors and then based on similarity matrix, we will fetch most similar movie based on user interest.

## V. Recommend movies

- Now we have everything to recommend movies based on user interest.

- When user enter name of movie to recommend similar movies, python code fetch index of that movie from data frame.
- Based on index, we will retrieve similarity score of this movie from similarity matrix.
- After getting similarity score of movies, we will sort all similarity score based on user movies.
- Then, we will fetch index of most similar movies from similarity matric.
- At the end, based on index, we will search movies name from data set and print movies name.

- Visual representation of movie recommendation system.


Figure 16: UI of movie recommendation system.

# References


Acilar, A. M., & Arslan, A. (2009). A collaborative filtering method based on artificial immune network. *Expert systems with applications*, *36*(4), 8324-8332.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, *17*(6), 734-749

Celma, Ò., & Serra, X. (2008). FOAFing the music: Bridging the semantic gap in music recommendation. *Journal of web semantics*, *6*(4), 250-256.

Chen, L. S., Hsu, F. H., Chen, M. C., & Hsu, Y. C. (2008). Developing recommender systems with the consideration of product profitability for sellers. *Information sciences*, *178*(4), 1032-1048.

Jalali, M., Mustapha, N., Sulaiman, M. N., & Mamat, A. (2010). WebPUM: a web-based recommendation system to predict user future movements. *Expert systems with applications*, *37*(9), 6201-6212.

Kim, H. N., El-Saddik, A., & Jo, G. S. (2011). Collaborative error-reflected models for cold-start recommender systems. *Decision support systems*, *51*(3), 519-531.

Min, S. H., & Han, I. (2005). Detection of the customer time-variant pattern for improving recommender systems. *Expert systems with applications*, *28*(2), 189-199

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*. doi:10.1155/2009/421425

Yu, K., Schwaighofer, A., Tresp, V., Xu, X., & Kriegel, H. P. (2004). Probabilistic memory-based collaborative filtering. *IEEE transactions on knowledge and data engineering*, *16*(1), 56-69.