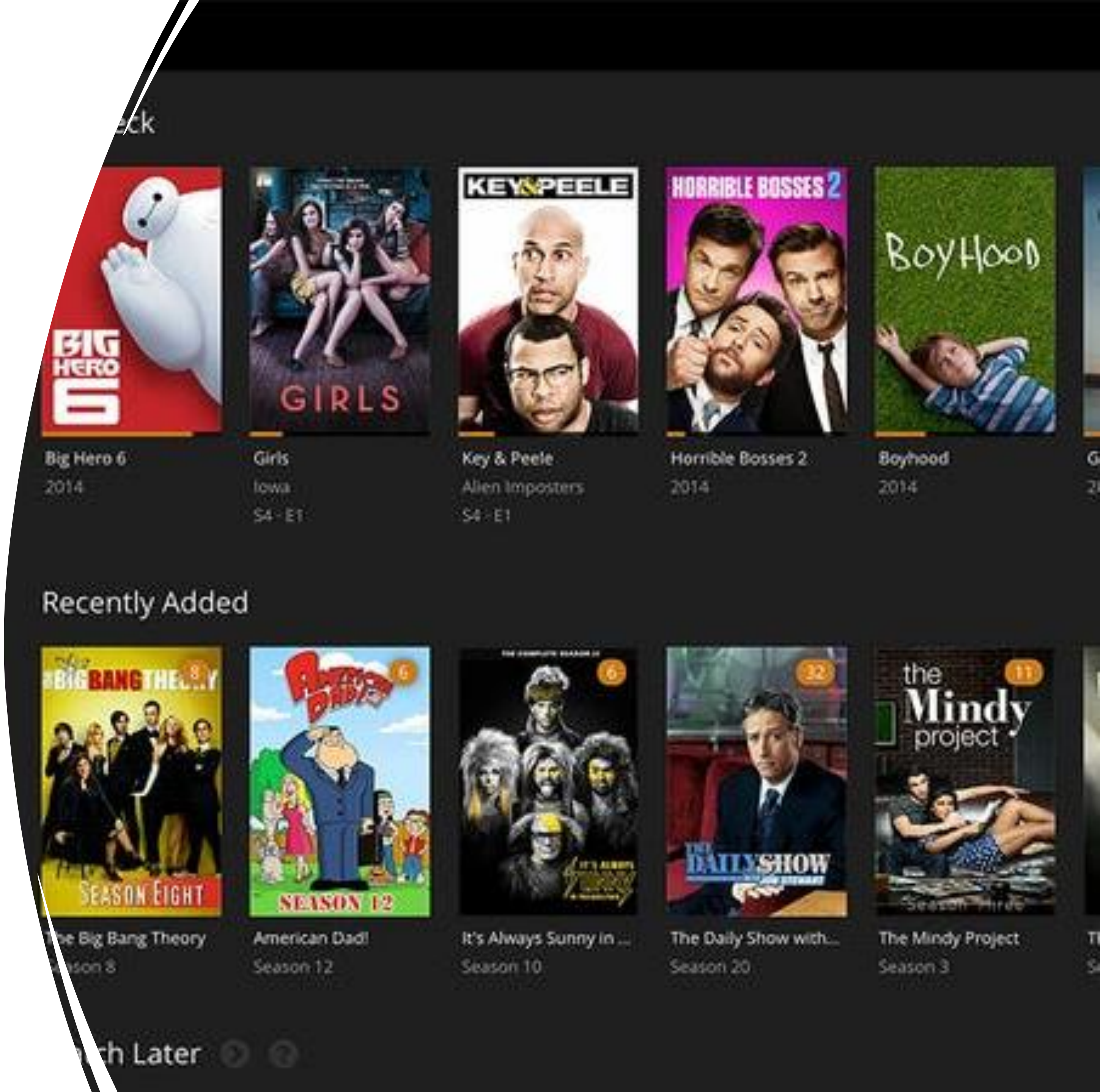
Movie Recommendation System

Submitted By:

- Priyanka
- Syed



Content

- What is a Recommendation System?
- Why Do We Need Recommender Systems?
- Problem Statement
- Project Objective
- Types of Recommender Systems
- Dataset
- Implementation
    - Data Cleaning
    - Data Pre-processing
    - Exploratory Data Analysis
    - Data Visualization
    - Model Building
    - Documentations and Final Report
    - Future Scope
    - Deployment of Project
- Building Recommendation System
- Any Questions???

What is a Recommendation System?

Recommendation System is a filtration program whose prime goal is to predict the “rating” or “preference” of a user towards a domain-specific item or item.

In our case, this domain-specific item is a movie

Therefore, the main focus of our recommendation system is to filter and predict only those movies which a user would prefer given some data about the user him or herself.

In layman’s terms, we can say that a **Recommendation System** is a tool designed to predict/filter the items as per the user’s behavior.

Why Do We Need Recommender Systems?

We now live in what some call the “era of abundance”. For any given product, there are sometimes thousands of options to choose from. Think of the examples above: streaming videos, social networking, online shopping; the list goes on. Recommender systems help to personalize a platform and help the user find something they like.

The easiest and simplest way to do this is to recommend the most popular items. However, to really enhance the user experience through personalized recommendations, we need dedicated recommender systems.

From a business standpoint, the more relevant products a user finds on the platform, the higher their engagement. This often results in increased revenue for the platform itself. Various sources say that as much as 35–40% of tech giants’ revenue comes from recommendations alone.

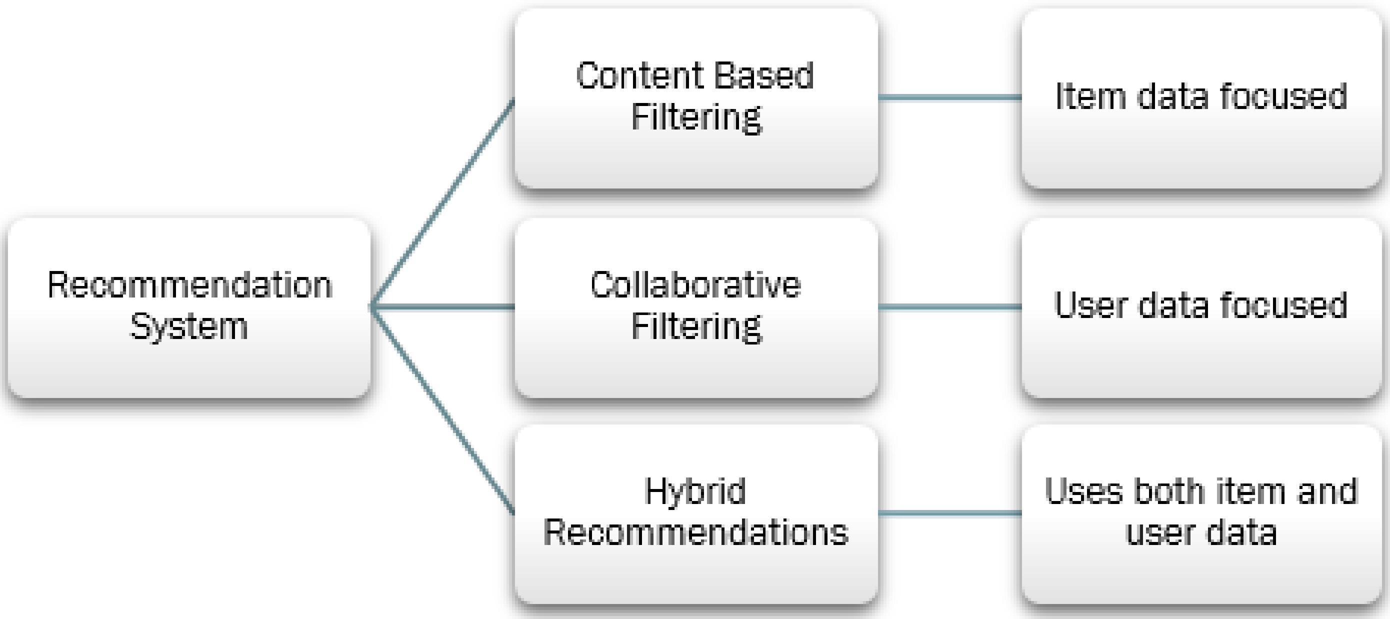
Problem Statement

- Using different techniques of Machine Learning, we need to build a Recommender System that recommends movies based on “ Cast, Genre, Reviews, TMDB/IMDB ratings”
- Using different types of recommendation techniques like:
    1. Popularity based recommender system
    2. Content based Recommender System
    3. Collaborative Recommender System

Project Objective

- To build a movie recommendation system with main focus to filter and predict personalized lists of useful and interesting content specific movies which a user would prefer based on some data provided about the user.

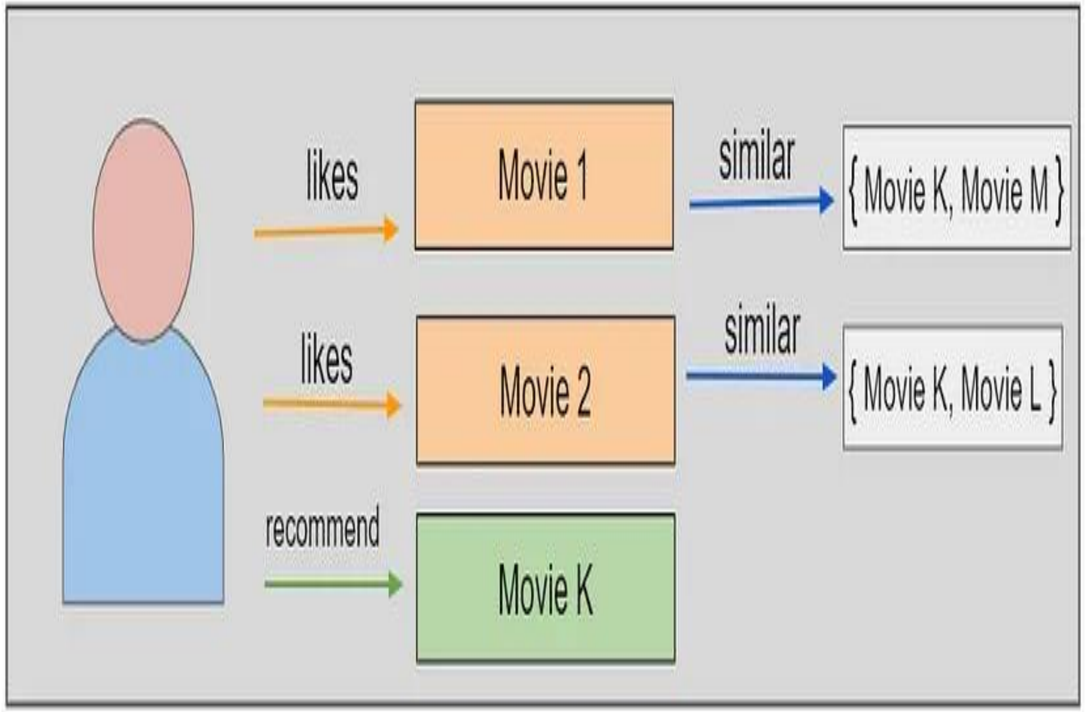
Types of Recommender Systems



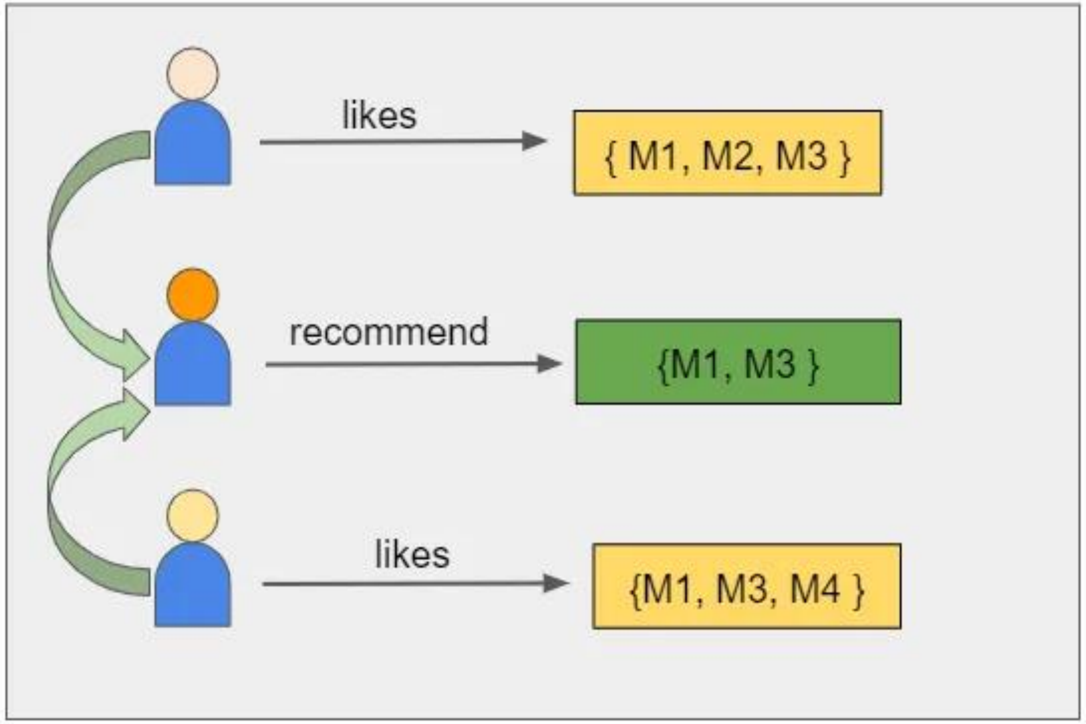
- **Content-Based Filtering:** This approach recommends movies based on the similarity of their content. It involves analyzing movie attributes such as genre, actors, director, plot keywords, and ratings. By comparing these attributes with a user's preferences, the system recommends movies that have similar characteristics to the ones they have liked in the past.
- **Collaborative Filtering:** Collaborative filtering recommends movies based on the behavior and preferences of similar users. There are two types of collaborative filtering:
  a) **User-Based Collaborative Filtering:** This method finds users who have similar movie preferences to the target user and recommends movies they have enjoyed but the target user has not seen.
  b) **Item-Based Collaborative Filtering:** This method identifies similar movies based on user ratings and recommends items that are similar to the movies the target user has already rated or watched.
- **Hybrid Approaches:** Hybrid recommender systems combine multiple techniques to provide more accurate recommendations. For example, you can combine content-based filtering and collaborative filtering to leverage both movie attributes and user preferences.

Types of Recommender Systems

Content-Based Movie Recommendation Systems



Collaborative Filtering Movie Recommendation System



Dataset

The Movies Dataset

These files contain metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

This dataset consists of the following files:

- **movies_metadata.csv:** The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
- **keywords.csv:** Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.
- **credits.csv:** Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.
- **links.csv:** The file that contains the TMDB and IMDB IDs of all the movies featured in the Full MovieLens dataset.

Implementation

- Data Collection and Preprocessing
- Exploratory Data Analysis
- Data Visualization
- Model Building
  1. Implementation of Machine Learning Algorithms (KNN, SVM)
  2. Implementation of TF-IDF(NLP Technique)
- Documentations and Final Report
- Future Scope
- Deployment of Project

Building Recommendation Systems

Popularity-Based Recommendation

vote_count, vote_average

Content-Based Recommendation

Movies Overview, Description, Taglines: Cosine Similarity

Movies, Cast, Crew,Genre and Keywords: Count Vectorizer

Collaborative-Based Recommendation

(SVD)

Data Collection

- https://www.kaggle.com/rounakbanik/movie-recommender-systems/data

Dataset

The Movies Dataset

These files contain metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

This dataset consists of the following files:

- **movies_metadata.csv:** The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
- **keywords.csv:** Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.
- **credits.csv:** Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.
- **links.csv:** The file that contains the TMDB and IMDB IDs of all the movies featured in the Full MovieLens dataset.
- **links_small.csv:** Contains the TMDB and IMDB IDs of a small subset of 9,000 movies of the Full Dataset.
- **ratings_small.csv:** The subset of 100,000 ratings from 700 users on 9,000 movies.

Libraries Used

Libraries Used

1. Programming Language
   - Python
2. Data Cleaning and Pre-processing
   - NumPy
   - Pandas
   - Ast
   - Datetime
3. Data Visualization
   - Matplotlib
   - Seaborn
4. Feature Extraction
   - Nltk (NLP)
   - Sklearn (CountVectorizer, Cosine_similarity)
   - Surprise (Scikit)

Data Cleaning and Pre-processing

- For pre-processing, connect the data access from various files.
- Perform data cleaning tasks to handle missing values, duplicates, removing special characters, data type conversion and inconsistencies.
- Apply text preprocessing techniques like lowercasing, tokenization, and removing stop words.
- Transform the dataset into a format that recommendation engines can understand.

Snippet Before Pre-processing Data

Out[39]:

	cast	crew	id	actors	director	keywords	keywords_type
0	[{'cast_id': 14, 'character': 'Woody (voice)',...	[{'credit_id': '52fe4284c3a36847f8024f49', 'de...	862	[Tom Hanks, Tim Allen, Don Rickles]	John Lasseter	[{'id': 931, 'name': 'jealousy'}, {'id': 4290,...	[jealousy, toy, boy, friendship, friends, riva...
1	[{'cast_id': 1, 'character': 'Alan Parrish', '...	[{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de...	8844	[Robin Williams, Jonathan Hyde, Kirsten Dunst]	Joe Johnston	[{'id': 10090, 'name': 'board game'}, {'id': 1...	[board game, disappearance, based on children'...
2	[{'cast_id': 2, 'character': 'Max Goldman', 'c...	[{'credit_id': '52fe466a9251416c75077a89', 'de...	15602	[Walter Matthau, Jack Lemmon, Ann-Margret]	Howard Deutch	[{'id': 1495, 'name': 'fishing'}, {'id': 12392...	[fishing, best friend, duringcreditsstinger, o...

Snippet After Pre-processing Data

Out[60]:

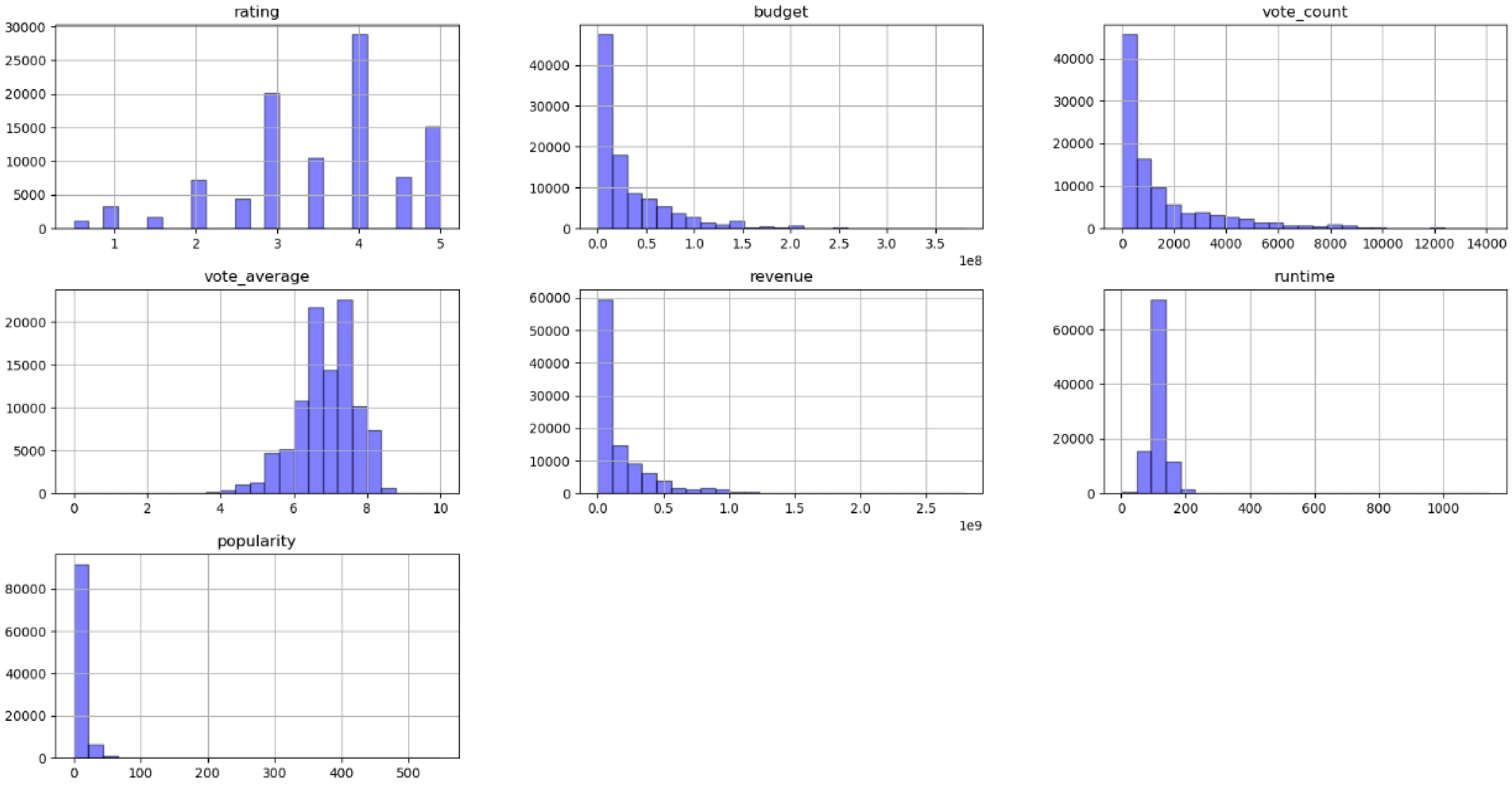
	id	movieId	userId	title	rating	actors	director	genres_category	original_language	languages_category	...	production_house	popularity	runtime	
0	862	1	7	Toy Story	3.0	[Tom Hanks, Tim Allen, Don Rickles]	John Lasseter	[Animation, Comedy, Family]	en	[English]	...	[Pixar Animation Studios]	21.946943	81.0	37:
1	862	1	9	Toy Story	4.0	[Tom Hanks, Tim Allen, Don Rickles]	John Lasseter	[Animation, Comedy, Family]	en	[English]	...	[Pixar Animation Studios]	21.946943	81.0	37:
2	862	1	13	Toy Story	5.0	[Tom Hanks, Tim Allen, Don Rickles]	John Lasseter	[Animation, Comedy, Family]	en	[English]	...	[Pixar Animation Studios]	21.946943	81.0	37:

3 rows × 24 columns

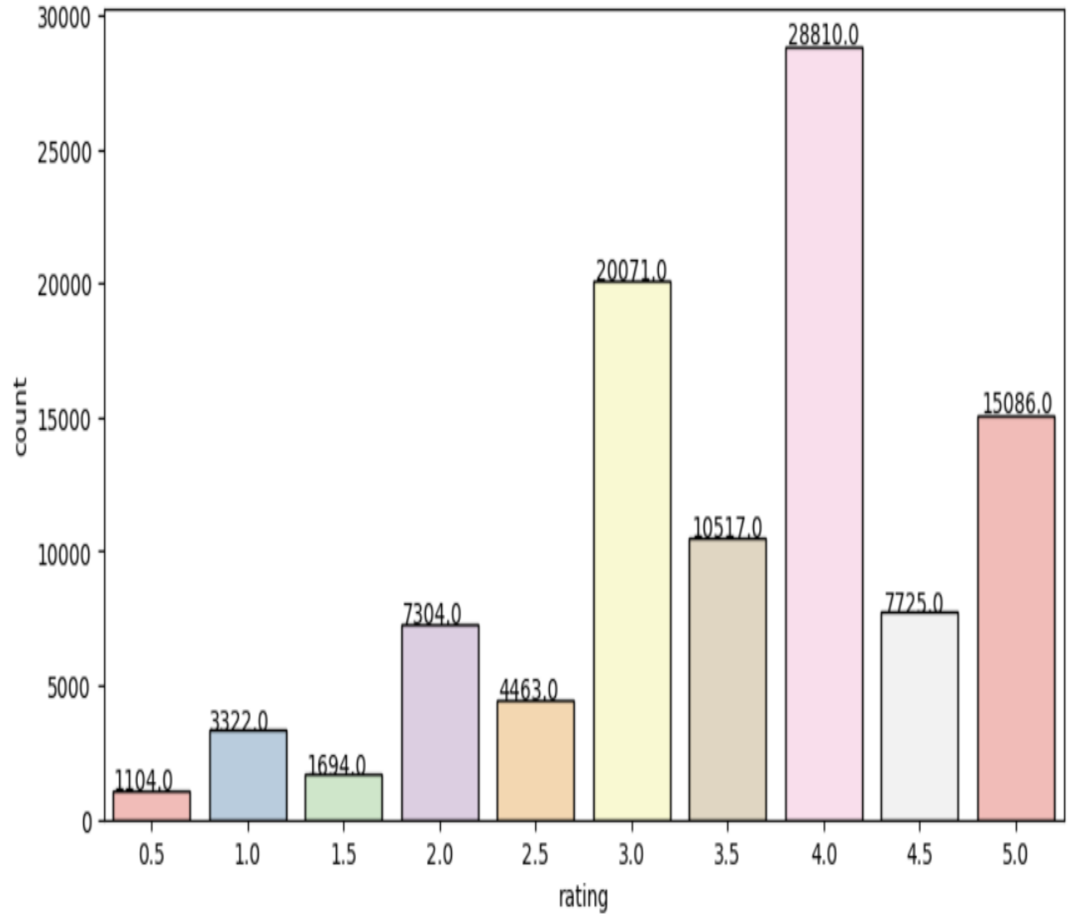
Exploratory Data Analysis (EDA)

- All the features of the new Data frame were analysed and visualized using visualization techniques.
- Python libraries like MATPLOTLIB and SEABORN was used for this purpose.
- Univariate, Bivariate and Multivariate analysis was done to understand the data and to draw meaningful insights from the data.

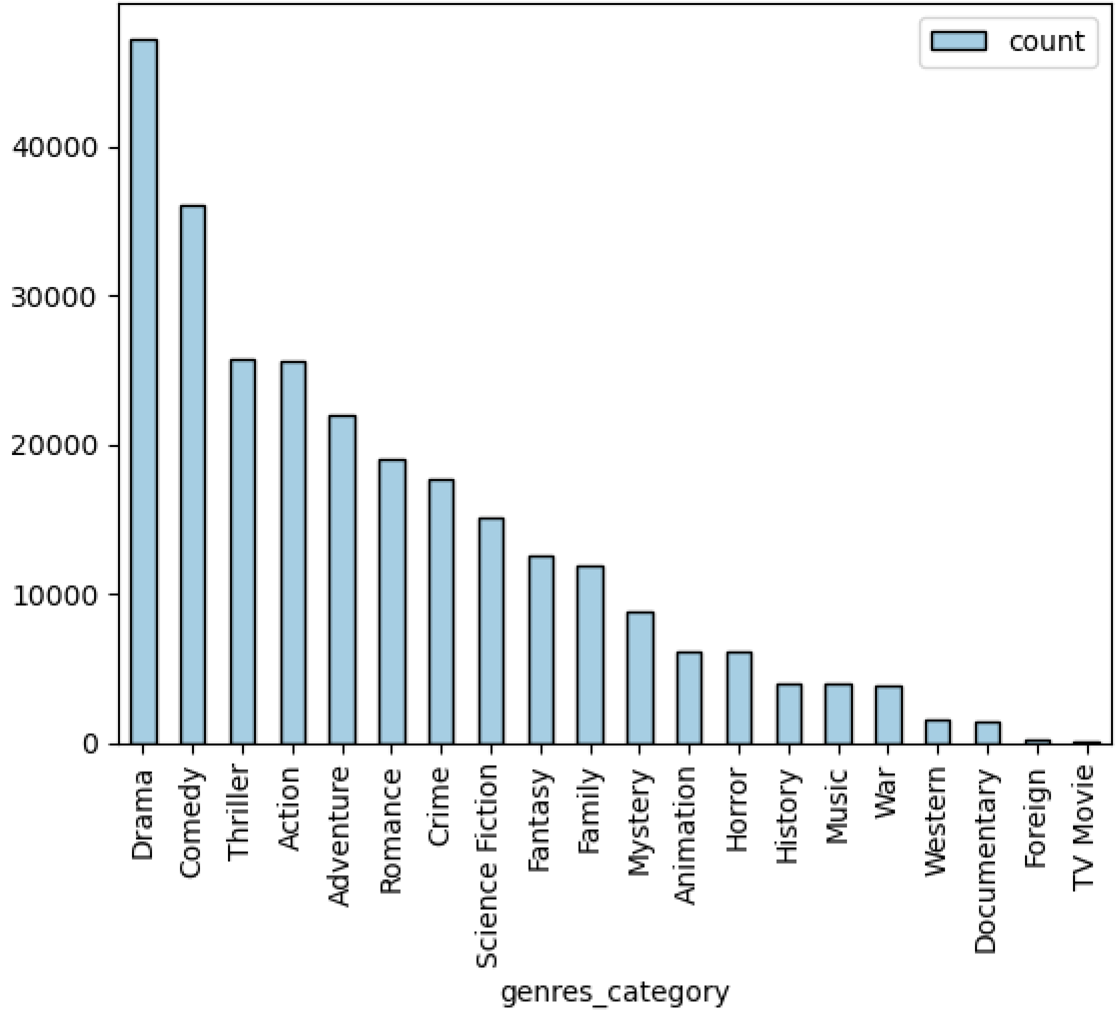
In [71]:
```
final_data.hist(column=['rating', 'budget', 'vote_count', 'vote_average', 'revenue', 'runtime', 'popularity'], bins = 25, color =
plt.show()
```



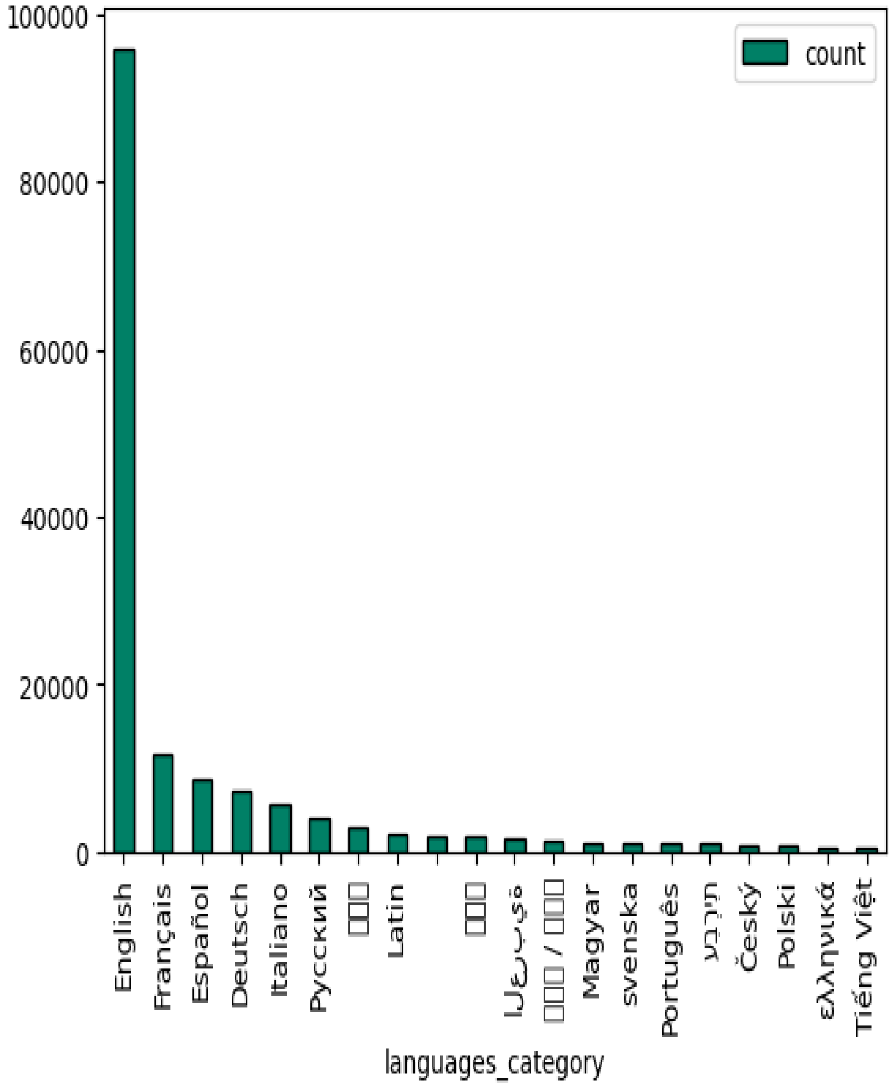
Based on the graph above, we can conclude Most movies got 4 Stars



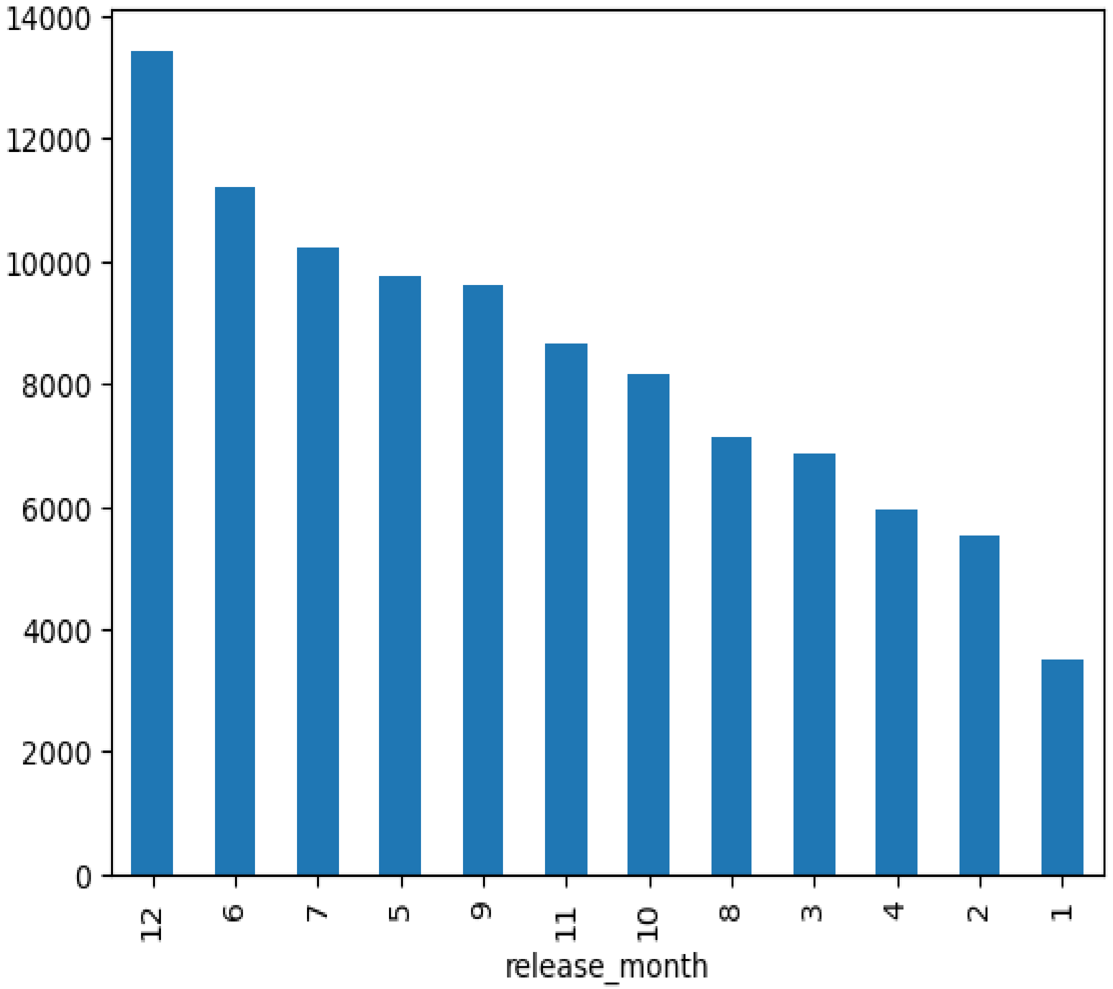
Based on the graph above, we can conclude Most movies were released based on the Drama Genre



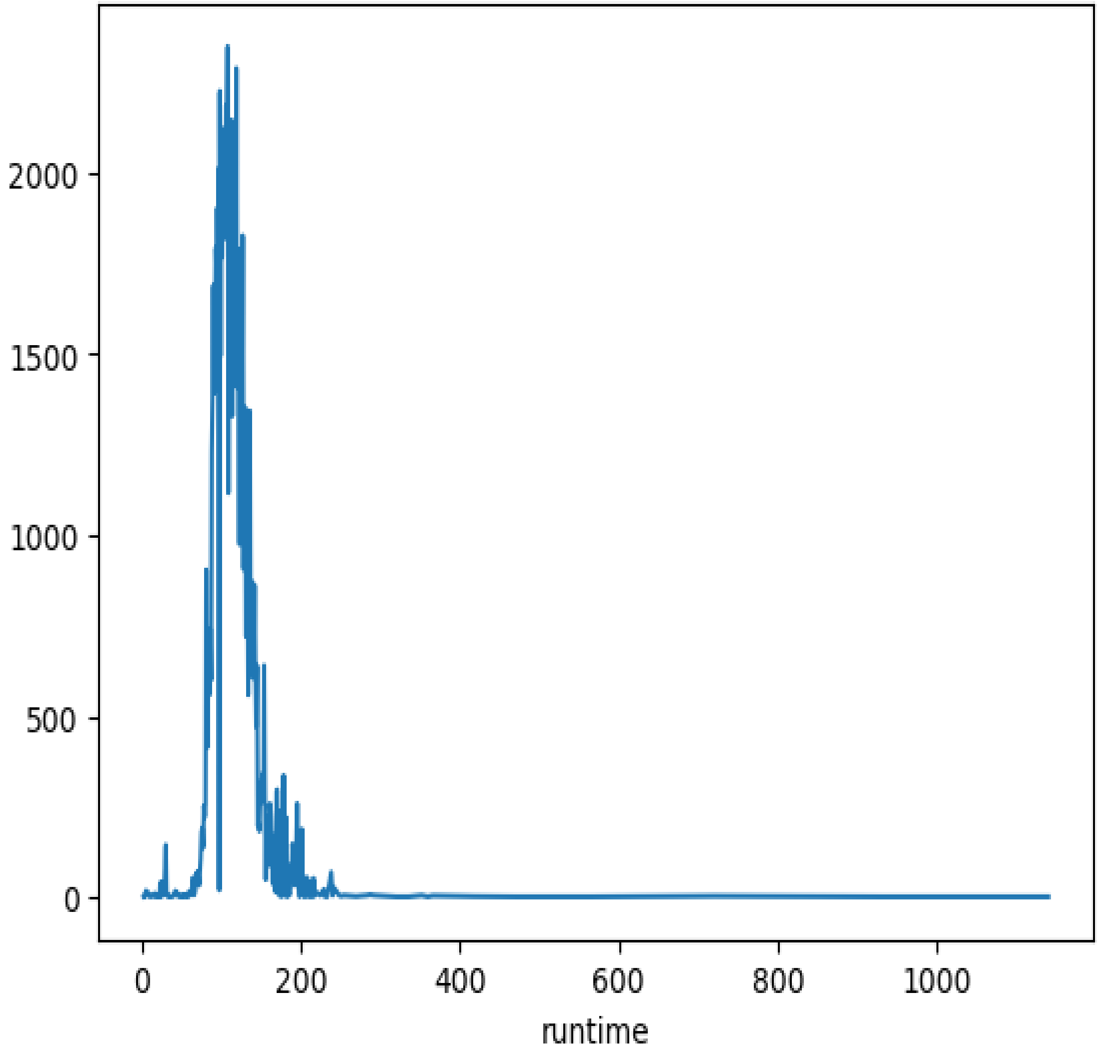
Based on the graph above, we can conclude Most movies were made in English Language



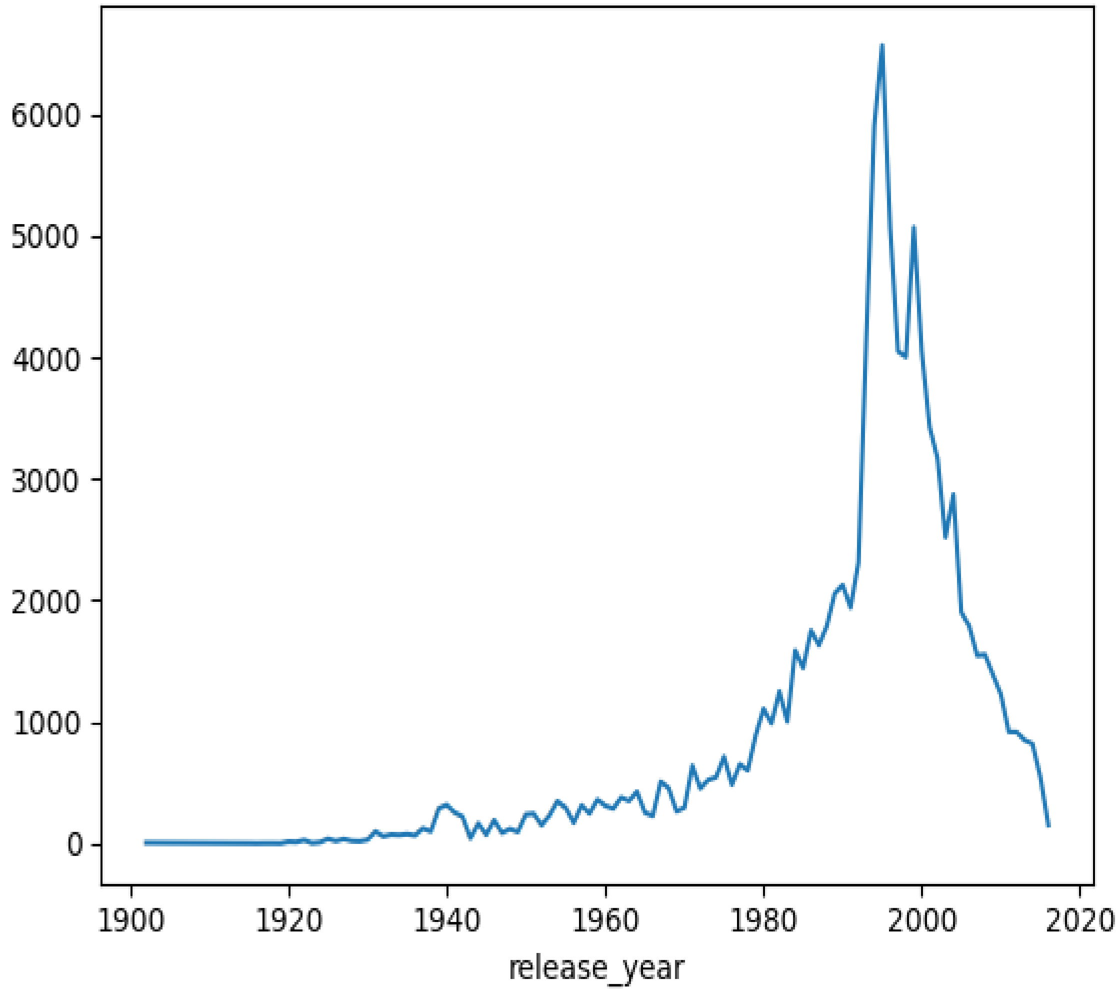
Based on the graph above, we can conclude Most movies were released in the month of December



Based on the graph above, we can conclude Most movies were made of duration between 90-200 minutes



Based on the graph above, we can conclude Most movies were released in the year 2000



Pre-processed Final Dataset For Model Training

```
In [90]: final_movie_data = final_movie_data.drop_duplicates()
         final_movie_data.head(3)
```

Out[90]:

	id	movieId	title	actors	director	genres_category	original_language	languages_category	overview	tagline	...	production_house
0	862	1	Toy Story	Tom Hanks,Tim Allen,Don Rickles	John Lasseter	Animation Comedy Family	en	English	Led by Woody, Andy's toys live happily in his ...		...	Pixar Animation Studios
247	8844	2	Jumanji	Robin Williams,Jonathan Hyde,Kirsten Dunst	Joe Johnston	Adventure Fantasy Family	en	English Français	When siblings Judy and Peter discover an encha...	Roll the dice and unleash the excitement!	...	TriStar Pictures Teitler Film Interscope Commu...
354	15602	3	Grumpier Old Men	Walter Matthau,Jack Lemmon,Ann-Margret	Howard Deutch	Romance Comedy	en	English	A family wedding reignites the ancient feud be...	Still Yelling. Still Fighting. Still Ready for...	...	Warner Bros. Lancaster Gate

3 rows × 22 columns

Popularity Based Recommendation System

In [97]:
```
movie_score = movie_popularity.sort_values('score', ascending = False)
movie_score[['title', 'vote_count', 'vote_average', 'score', 'popularity']].head()
```

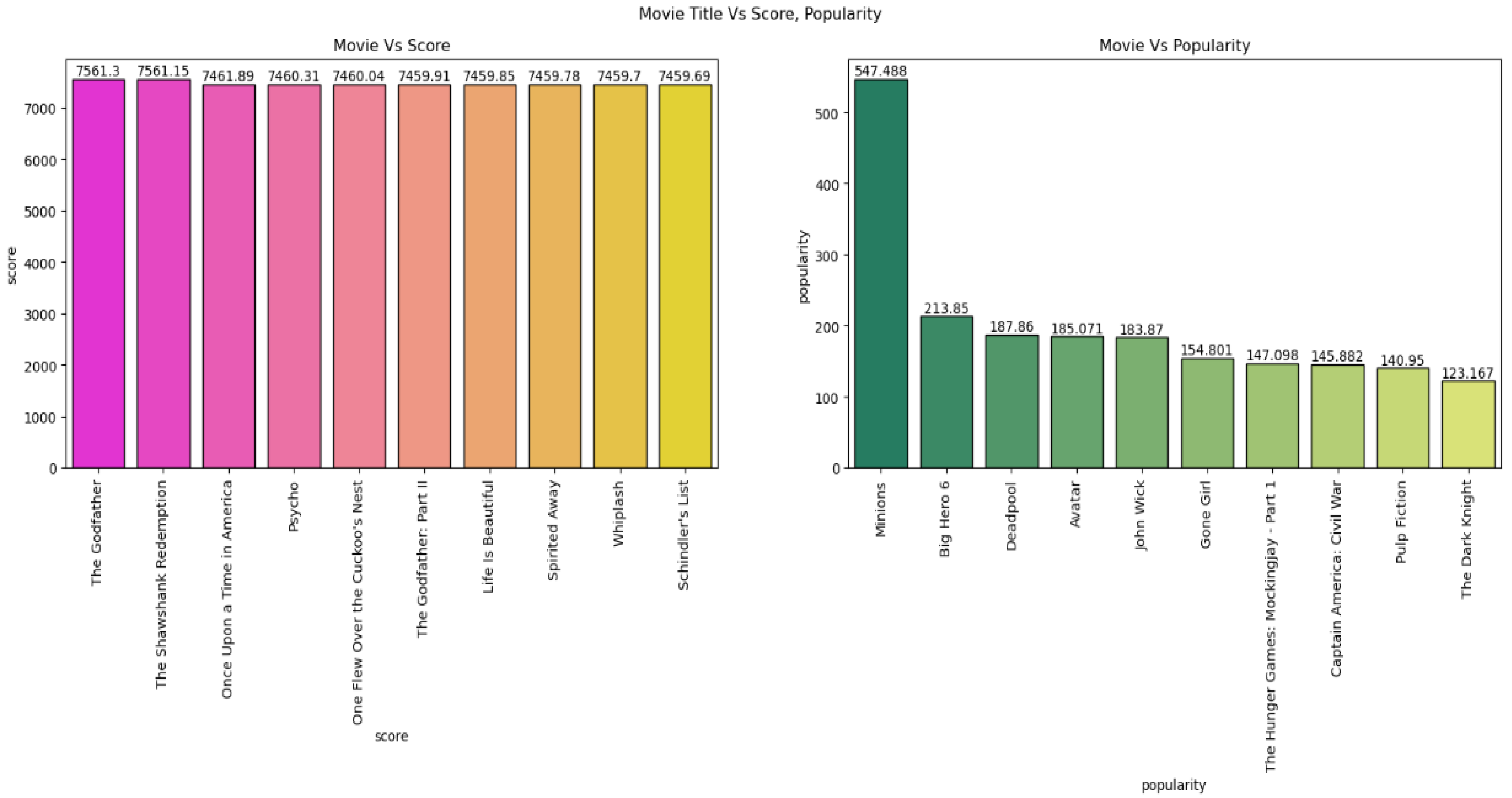
Out[97]:

	title	vote_count	vote_average	score	popularity
22016	The Godfather	6024.0	8.5	7561.299804	41.109264
9528	The Shawshank Redemption	8358.0	8.5	7561.149874	51.645403
30830	Once Upon a Time in America	1104.0	8.3	7461.892494	32.182851
30246	Psycho	2405.0	8.3	7460.307716	36.826309
28136	One Flew Over the Cuckoo's Nest	3001.0	8.3	7460.040636	35.529554

In [98]:
```
movie_popular = movie_popularity.sort_values('popularity', ascending = False)
movie_popular[['title', 'vote_count', 'vote_average', 'score', 'popularity']].head()
```

Out[98]:

	title	vote_count	vote_average	score	popularity
99788	Minions	4729.0	6.4	6492.546829	547.488298
99158	Big Hero 6	6289.0	7.8	7204.977181	213.849907
99511	Deadpool	11444.0	7.4	7001.145524	187.860492
94146	Avatar	12114.0	7.2	6899.329893	185.070892
99116	John Wick	5499.0	7.0	6797.851063	183.870374


Movie Title Vs Score, Popularity
Movie Vs Score
score
7561.3
7561.15
7461.89
7460.31
7460.04
7459.91
7459.85
7459.78
7459.7
7459.69
The Godfather
The Shawshank Redemption
Once Upon a Time in America
Psycho
One Flew Over the Cuckoo's Nest
The Godfather: Part II
Life Is Beautiful
Spirited Away
Whiplash
Schindler's List
Movie Vs Popularity
popularity
547.488
213.85
187.86
185.071
183.87
154.801
147.098
145.882
140.95
123.167
Minions
Big Hero 6
Deadpool
Avatar
John Wick
Gone Girl
The Hunger Games: Mockingjay - Part 1
Captain America: Civil War
Pulp Fiction
The Dark Knight

Why We Choose Content Based Filtering?

- Content-based filtering enables personalized recommendations based on individual user preferences.
- It can handle the cold-start problem by leveraging item attributes when there is limited or no user data. Recommendations are independent of user behavior or ratings, making it useful for new users
- Content-based filtering offers transparency and explain ability in its recommendations, as they are based on item attributes. The use of item attributes makes it easier to explain why certain items are recommended to users.
- It can provide diverse recommendations by considering various item attributes, allowing users to explore a wider range of items.
- Content-based filtering is less influenced by trends or popularity bias because it focuses on the intrinsic attributes and characteristics of items rather than relying on user behavior or ratings., reducing the spillover effect.

Content Based Recommendation System

In [106]:
```
movie_content_based['tags'] = movie_content_based['languages_category'] + ' ' + movie_content_based['keywords_type'] + movie_cont
movie_content_based = movie_content_based.drop(['id', 'actors', 'director', 'budget', 'original_language', 'popularity', 'revenue
movie_content_based.head(2)
```

Out[106]:

	movieId	title	tags
0	1	Toy Story	English jealousy toy boy friendship friends ri...
247	2	Jumanji	English Français boardgame disappearance based...

Text Vectorization Process

- In this process, we have created an additional column in our dataset called as “Tags”. It would contain all the necessary and important keywords related to the particular movie.
- Then, in next step we combined all the tags and choose the top 5000 most occurring words as the column names for the matrix
- So, further for every movie we have tags and we plotted the same in vector form of 5000x5000 matrix, where in the columns would be as discussed above and in the rows there would be the movie names
- Also, in order to choose only the unique repeating words(excluding a ,an the, words) we have used a filter called “PorterStemmer()”

```python
import nltk
```

```python
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
```

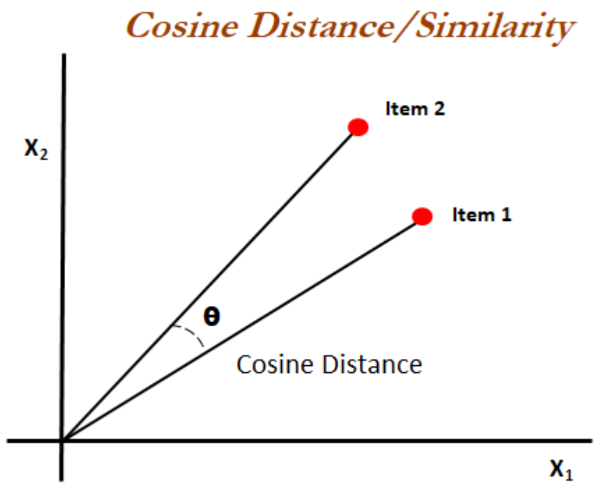
```python
def stem(text):
    y=[]
    for i in text.split():
        y.append(ps.stem(i))
    return " ".join(y)
```

```python
new['tags']=new['tags'].apply(stem)
```

Similarity Score Calculation

Similarity Calculation:

- Compute the similarity between movies using a suitable similarity metric, such as cosine similarity, Jaccard similarity, or Euclidean distance.
- Calculate the similarity scores between movies based on their feature vectors, allowing you to quantify how similar or related they are to each other.
- We go with cosine similarity




```
[194] from sklearn.feature_extraction.text import  CountVectorizer
      vector = CountVectorizer(max_features=5000 , stop_words='english')
      vector_matrix = vector.fit_transform(new_df4['summary']).toarray()
```

```
[ ] from sklearn.metrics.pairwise import cosine_similarity
    similarity_scores= cosine_similarity(vector_matrix, vector_matrix)
```

```
[ ] from sklearn.metrics.pairwise import linear_kernel
    cosine_sim = linear_kernel(vector_matrix, vector_matrix)
```

Output

```
In [118]: get_recommendations('Minions').head(10)

Out[118]: 8285          Despicable Me 2
          7208                 Year One
          1822            One Tough Cop
          7783                   Cars 2
          1648      The Return of Jafar
          8624           Doctor Strange
          8444           The Lego Movie
          7799               The Smurfs
          8417    Death of a Superhero
          7258                  G-Force
          Name: title, dtype: object
```

Collaborative Based Recommendation System

Item-based collaborative filtering

The concept in this case is to find similar movies instead of similar users and then recommending similar movies.

In [120]:
```
ratings_small.head(3)
```

Out[120]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182

In [121]:
```
reader = Reader()

data = Dataset.load_from_df(ratings_small[['userId', 'movieId', 'rating']], reader)
# data.split(n_folds=5)

# Load the dataset (download it if needed)
# data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm
algo = SVD()

# Run 5-fold cross-validation and then print results
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).
```

Hybrid Based Recommendation System

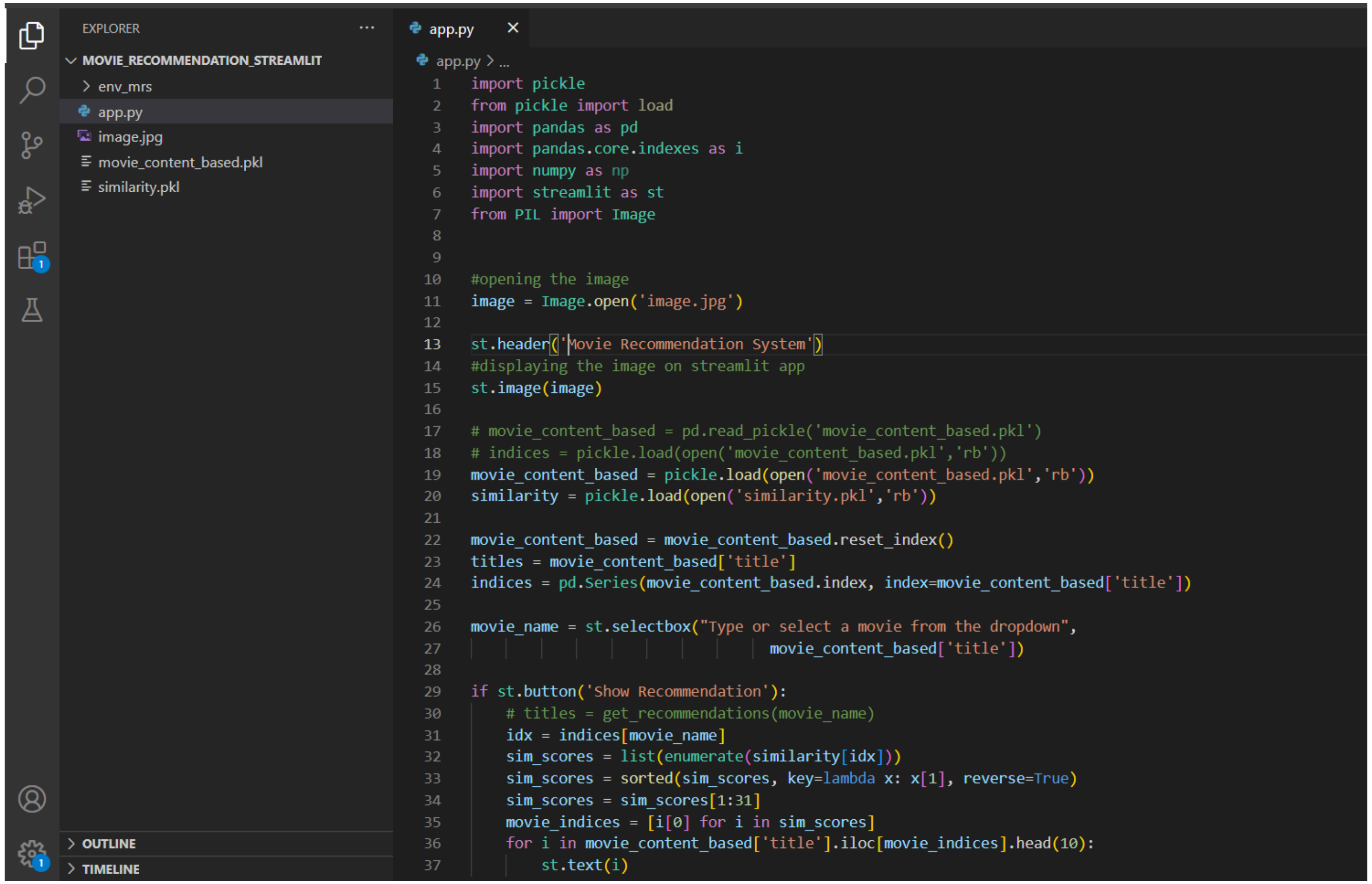
In [129]:
```
hybrid_recommendation(1, 'Avatar')
```

Out[129]:

	title	vote_count	vote_average	release_year	id	est
63382	Predator	2129.0	7.3	1987	106	3.038834
95986	X-Men: First Class	5252.0	7.1	2011	49538	3.036595
31966	The Day the Earth Stood Still	323.0	7.3	1951	828	2.959586
64884	Alien Nation	81.0	5.9	1988	10128	2.881361
96212	Rise of the Planet of the Apes	4452.0	7.0	2011	61791	2.812945
95194	Wall Street: Money Never Sleeps	504.0	5.8	2010	33909	2.760961
99180	The Book of Life	778.0	7.3	2014	228326	2.758056
20885	Independence Day	3334.0	6.7	1996	602	2.755978
93054	Dragonball Evolution	475.0	2.9	2009	14164	2.739797
99781	Independence Day: Resurgence	2550.0	4.9	2016	47933	2.738253

Model Deployment

- Deploy code using vs code.



- Deploy the movie recommender system on Streamlit showing top 10 recommended movies.

Movie Recommendation System

Type or select a movie from the dropdown

toy story

Show Recommendation

Type or select a movie from the dropdown

toy story

Show Recommendation

```
toy story 2

toy story 3

the 40 year old virgin

what's up, tiger lily?

toy story of terror!

man on the moon

burke & hare

factory girl

beverly hills chihuahua

white men can't jump
```

Challenges Faced In The Project

- Some columns contained mixed datatypes with categorical and datetime where the actual datatype is int64.It was handled by deleting the rows with datetime and was typecasted to integer.

- While running visualizations some of the plots were not supported. They were handled by changing the plots which had support.

- While doing content-based filtering, due to large number of rows in the features column error was thrown which indicated lack of memory. Hence, we have used overview feature after doing all necessary preprocessing to allocate memory to run within the limits available.

- While running the applications for some input values key value error and other types of errors have been raised. Initially it was a challenge to figure it out but later understood that some of the movie titles were same maybe some movies released with similar movie names in different years. All the duplicated values are dropped by which error was resolved.

Future Scope

- Collect regular user feedback to improve the accuracy and relevance of the recommendation system.
- Give recommendation with rating and review of movie
- To provide up-to-date recommendations, the system should be updated on a regular basis with new movie releases, user preferences, and evolving algorithms. Monitor system performance and make necessary optimisations.
- Deep learning techniques can be used to improve the accuracy of predictions of our model.
- These techniques can be integrated with similar apps like YouTube, Spotify etc. to generate recommendations.

Any Questions???

Thank You!!!