Artificial Intelligence
PROJECT

Movie Recommendation System

By:
- Aditya Sethi
- Akash Sunda
- Chetan Verma

Mentor: Prof. Punam Bedi

Problem Statement:

The sheer volume of movies and TV shows available on various streaming platforms makes it increasingly challenging for viewers to discover content that aligns with their personal preferences. As a result, there is a growing need for effective movie recommendation systems that can help users discover new films and enhance their viewing experience.

Objective:

- ✓ Scalability and Efficiency
- ✓ Personalization and Serendipity
- ✓ Ethical Considerations

Recommendation Engine:

- Recommendation system/engine is a computational system that uses data, user preferences, and item characteristics.
- It provides tailored suggestions or recommendations to users.
- Recommendations aligned with individual interests, preferences, and needs help users discover items, products, content, or services.
- Widely used in various domains (e-commerce, content platforms, music streaming, news websites).
- Aims to enhance user experience and increase engagement.
- Facilitates decision-making by presenting options of interest.
- Primary goal is to improve user satisfaction and retention.

SBI Debit Card | Offers refresh every alternate day

SBI card | Offers refresh every day

Activate your card for online transactions ▸

prime Shop & get rewards worth ₹1,200

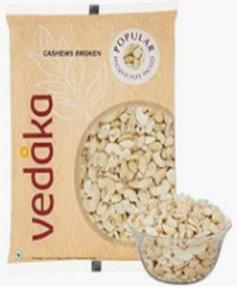
5% Unlimited cashback* Amazon Pay ICICI Bank credit card

View all offers ▸

Recommended deals for you

₹499 and under Great Indian Festival

BSB Home® Premium Super Soft Cloudy Printed Mink Single Bed Blanket for All Season, Ultrasoft & Cozy Single Ply Blanket 152 x 228...



32% off Great Indian Festival

Amazon Brand - Vedaka Popular Cashews - Broken, 200g

30% claimed

61% off Great Indian Festival

Amazon Brand - Presto Disinfectant Toilet Cleaner - 1 L (Pack of 4)

49% claimed

₹399 and under Great Indian Festival

Best Deal

35% off Great Indian Festival

Dabur Meswak Complete Oral Care Toothpaste - 500g (2 x 200g + 1 x100g) | Tooth Decay Prevention, S...

10% claimed

52% off Great Indian Fe

Odonil Gel Pocket Mix (Assorted pack of 3 ne Infused with Essential

83% claimed

Types of Recommendation Systems

Recommendation systems typically employ various filtering techniques such as:

- ✓ Content-based Filtering
- ✓ Collaborative Filtering
- ✓ Hybrid Filtering

Content-Based Filtering

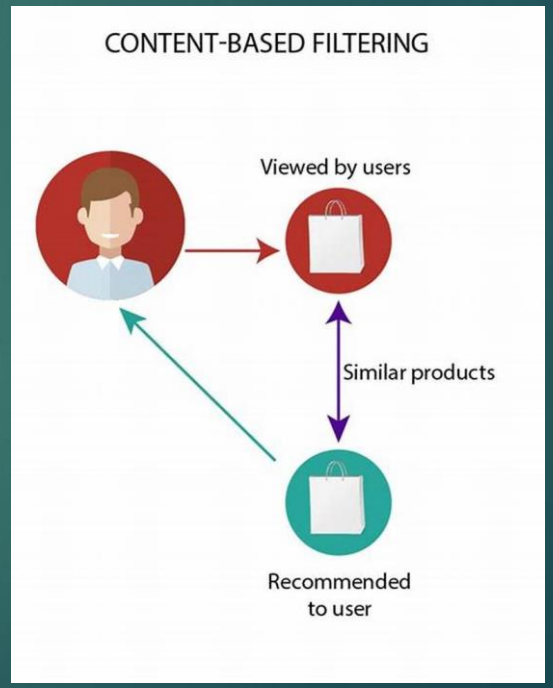
Content-based filtering is a recommendation system technique that suggests items (such as movies, articles, or products) to users based on the characteristics and features of the items and a profile of the user's preferences.

Pros:

- ✓ Personalization
- ✓ Transparency
- ✓ Diversity

Cons:

- ✓ Resource Intensive
- ✓ Limited Discovery
- ✓ Dependence on Item Description



Collaborative Filtering

Collaborative filtering is a recommendation system technique that suggests items (such as movies, products, or content) to users based on the preferences, behaviors, or interactions of other users.

This method relies on the idea that users who have shown similar preferences or behavior in the past are likely to have similar preferences for future items.

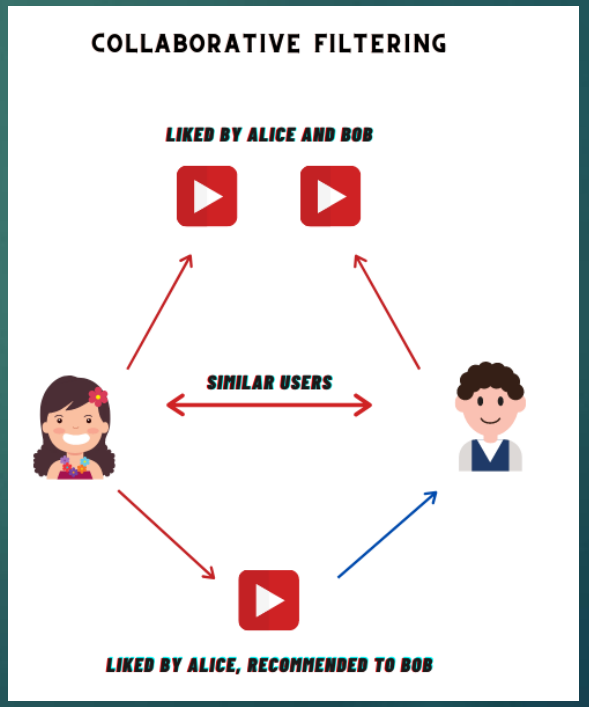
Collaborative filtering is divided into two main types: user-based (user-user) and item-based(item-item).

Pros:

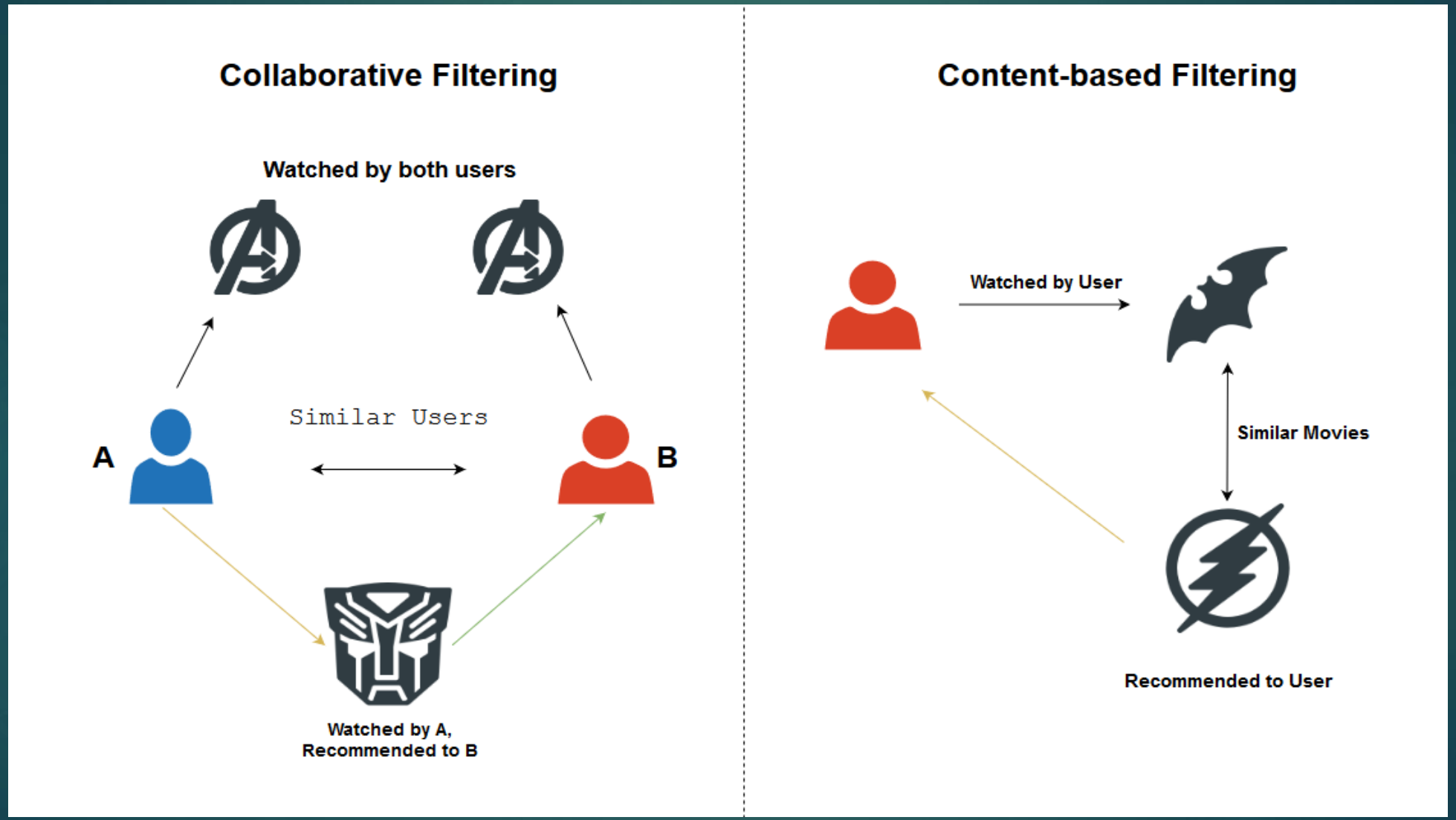
- ✓ Effective Personalization
- ✓ Serendipity
- ✓ New User Integration

Cons:

- ✓ Resource Intensive
- ✓ Popularity Bias
- ✓ Privacy Concerns



Collaborative Vs Content-Based Filtering



NETFLIX

Home | TV Shows | Movies | New & Popular | My List | Browse by Languages

Popular on Netflix

Because you watched Gumraah

Because you watched Cobra Kai

Dataset used:

We will be using dataset from the MovieLens website, which has been collected over some period (till 09/2018) and comprises of 100,000 ratings from 600 users on 9000 movies having following attributes:

User Id, Movie Id, Movie Title, Rating, Genre, imdbId.

Example:

Input:

User inputs any movie name like Iron Man.

Output:

User will get recommendation for similar movies like Iron Man 2, Iron Man 3, Captain America.

User-User Collaborative Filtering

Finds the similarity score between users. Based on this similarity score, it then picks out the most similar users and recommends products which these similar users have liked or bought previously.

- ✓ Quite time consuming as it involves calculating the similarity for each user and then calculating the prediction for each similarity score.
- ✓ Useful when the number of users is less.
- ✓ Offers highly personalized recommendations by considering the preferences of similar users.
- ✓ Sparsity: Vulnerable to sparsity issues, especially in large-scale systems where users may not have interacted with a significant portion of items.
- ✓ Data Quality: Highly dependent on the quality of user ratings and the availability of sufficient historical data.
- ✓ Cold start for new users: Faces challenges in providing accurate recommendations for new users who have not yet provided sufficient ratings or interactions.

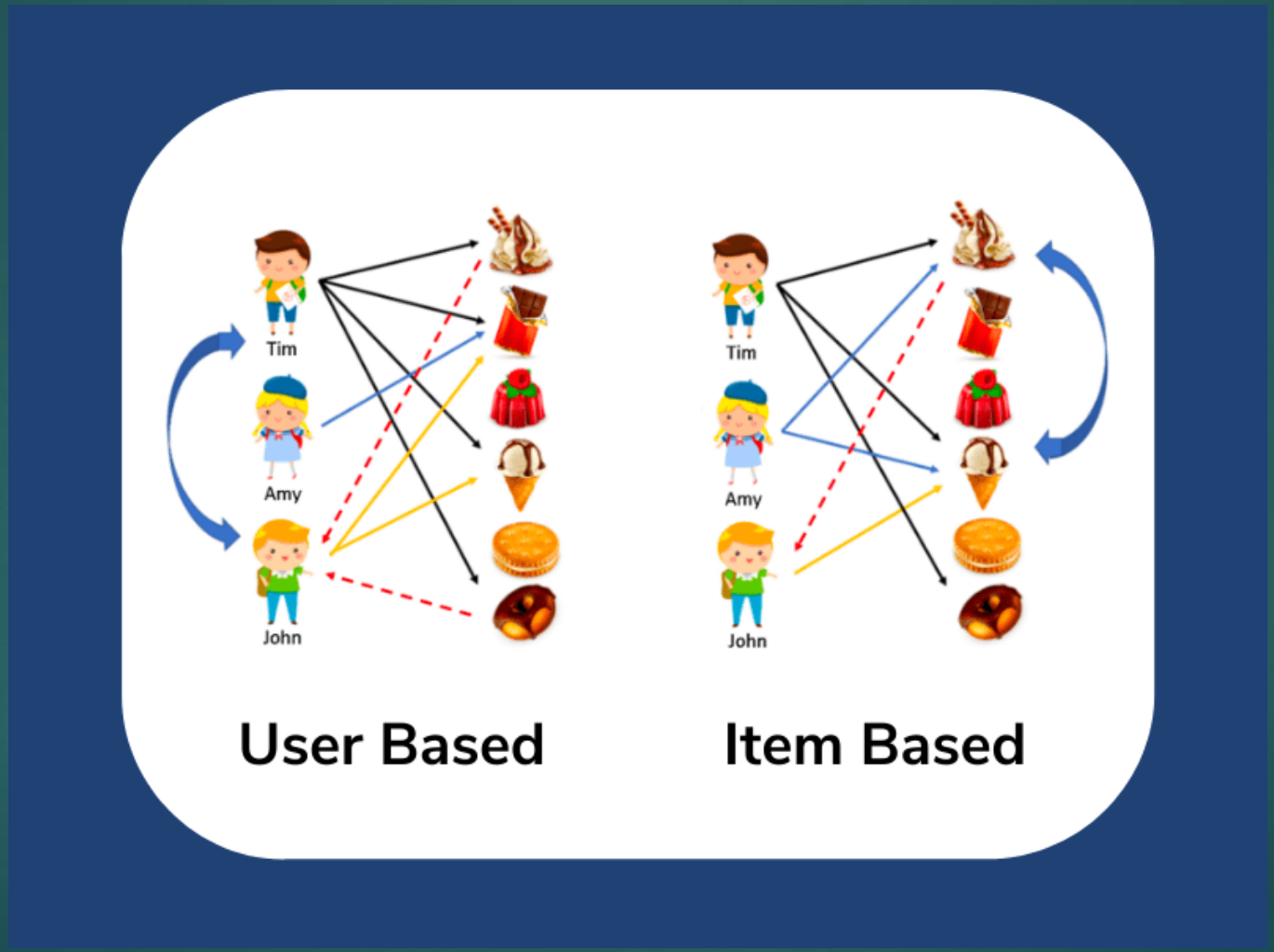
Item-Item Collaborative Filtering

Compute the similarity between each pair of items.

Finds the similarity between each movie pair and based on that, we will recommend similar movies which are liked by the users in the past.

- ✓ Efficiency: Can be computationally more efficient than user-user collaborative filtering, especially as the number of items is often less than the number of users.
- ✓ User Independence: Recommendations can be generated without relying on explicit user information, making it more resilient to new user cold start problems.
- ✓ Scalability: Tends to scale better with a large number of items, as the computation of item similarities is often less intensive than user similarities.
- ✓ Quality of Item Descriptions: The effectiveness of item-item collaborative filtering is highly dependent on the quality and relevance of item descriptions or features.
- ✓ Cold Start for New Items: Faces challenges in providing accurate recommendations for new items with limited or no historical interaction data.

User-User Vs Item-Item Filtering



Similarity Measures

- Quantify likeness between two data sets
- Higher when objects are more alike
- Often falls in the range [0,1]

Some Measures:

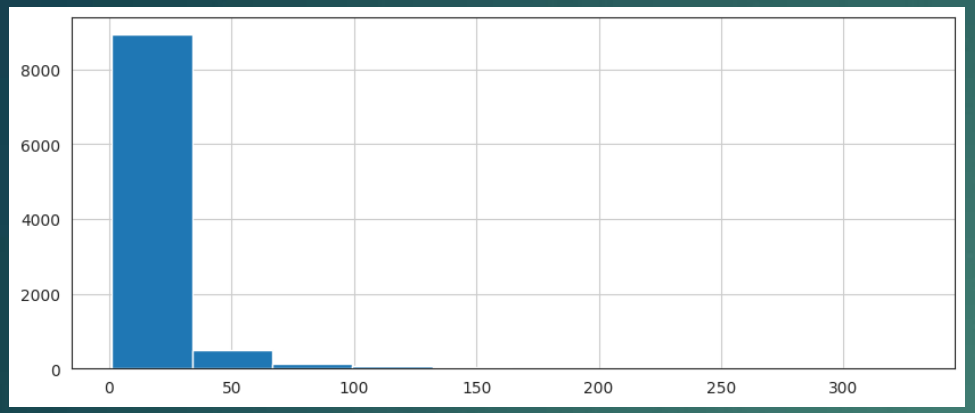
- **Cosine Similarity**
- **Pearson Correlation Coefficient**
- Jaccard Similarity
- Spearman Rank Correlation

Cosine Similarity

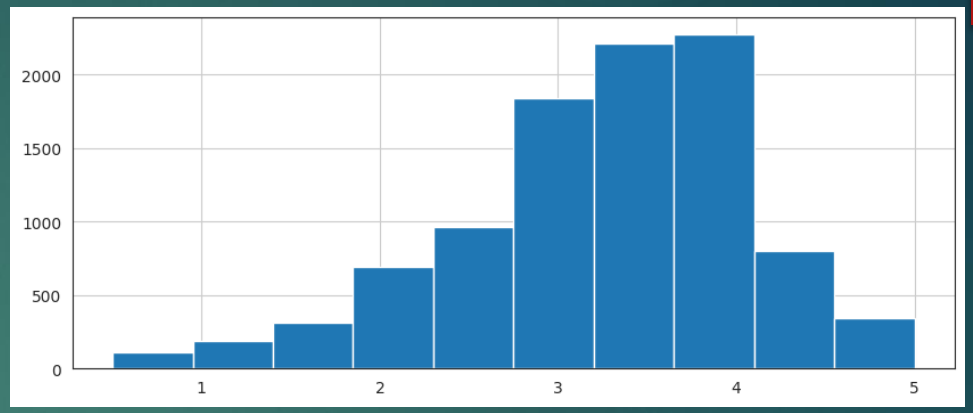
- Measures the cosine of the angle between two vectors.
- Ranges from -1 to 1, where 1 means the vectors are identical, 0 means the vectors are orthogonal, and -1 means they are diametrically opposed.
- Widely used in text mining, collaborative filtering, and other recommendation systems.
- Handling Sparsity: Performs well in situations with sparse data, such as user-item interactions, where many entries are missing.

- Formula: $\text{cosine\_similarity}(A,B) = (A \cdot B)/(\|A\| \cdot \|B\|)$

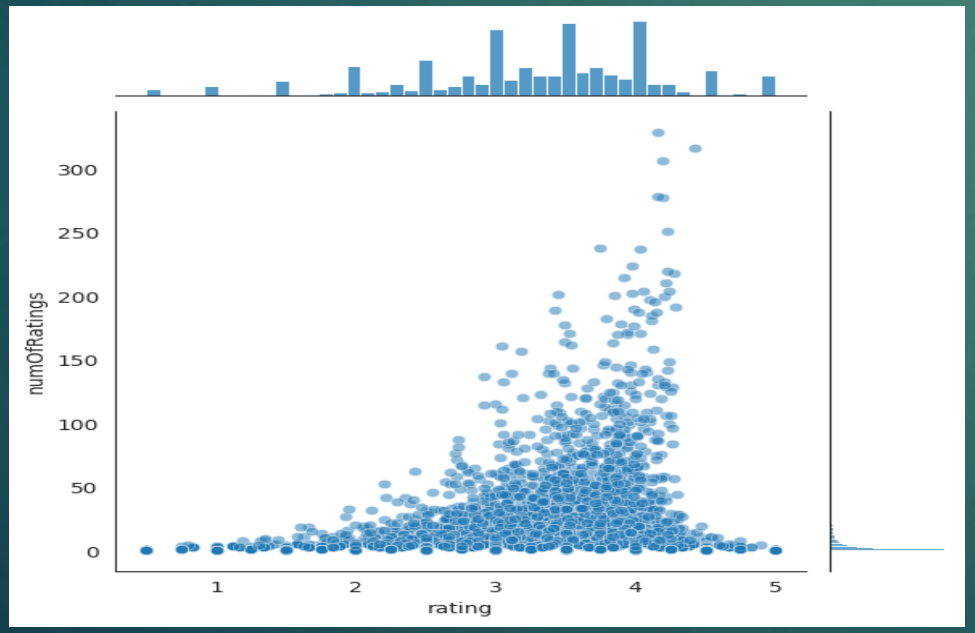
Data Visualization

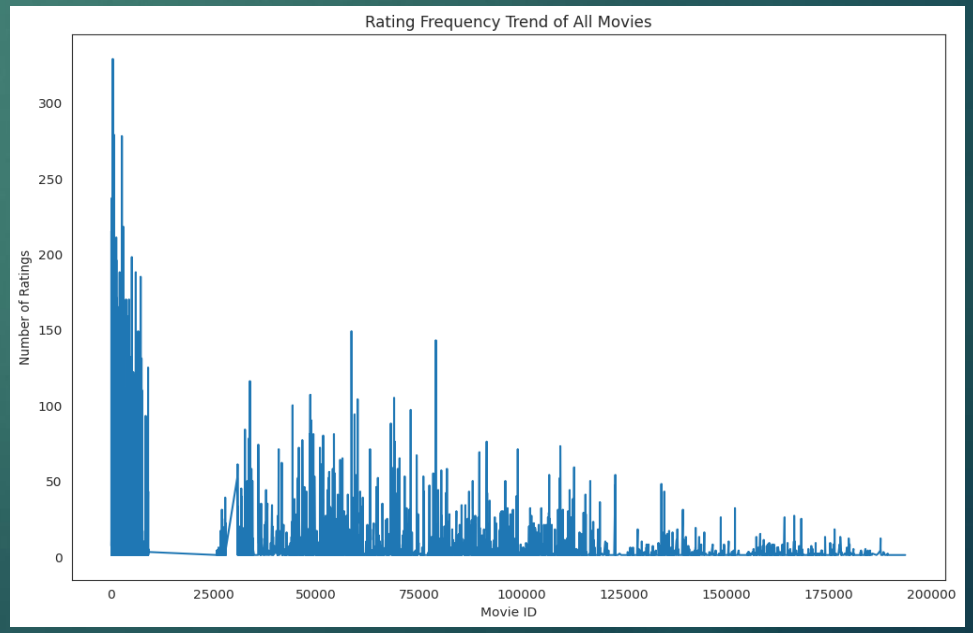


x-axis represents the number of ratings a movie has
y-axis represents the frequency (or count) of movies with a specific number



x-axis represents the possible ratings
y-axis represents the frequency (or count) of movies or items with a specific rating.





Cosine Similarity using KNNBasic

```
df1 = df[['userId', 'movieId', 'rating']]
```

```
df1.head()
```

	userId	movieId	rating
0	1	1	4.0
1	5	1	4.0
2	7	1	4.5
3	15	1	2.5
4	17	1	4.5

```
df1.isnull().sum()
```

```
userId     0
movieId    0
rating     0
dtype: int64
```

```
reader = Reader(line_format='user item rating', sep=',', rating_scale=(1, 5))
```

```
data = Dataset.load_from_df(df1[['userId', 'movieId', 'rating']], reader)
```

```
trainset, testset = train_test_split(data, test_size=0.25)
```

```
sim_options = {
    'name': 'cosine',
    'user_based': False
}

model = KNNBasic(sim_options=sim_options)
model.fit(trainset)
```

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
<surprise.prediction_algorithms.knns.KNNBasic at 0x7efc5507edd0>
```

```python
def get_top_similar_movies(movie_name, N=5):
    movie_id = movies.loc[movies['title'] == movie_name, 'movieId'].values[0]
    movie_inner_id = model.trainset.to_inner_iid(movie_id)

    raw_neighbor_inner_ids = model.get_neighbors(movie_inner_id, k=N)

    top_similar_movies = [model.trainset.to_raw_iid(inner_id) for inner_id in raw_neighbor_inner_ids]
    top_similar_movie_names = [movies.loc[movies['movieId'] == movie_id, 'title'].values[0] for movie_id in top_similar_movies]

    return top_similar_movie_names
```

```python
movie_name = ('Iron Man (2008)')
top_similar_movies = get_top_similar_movies(movie_name, N=5)

print(f"Top 5 movies similar to '{movie_name}':")
for i, similar_movie in enumerate(top_similar_movies, 1):
    print(f"{i}. {similar_movie}")
```

```
Top 5 recommended movies for Iron Man (2008):
Dark Knight, The (2008)
WALL·E (2008)
Avengers, The (2012)
Iron Man 2 (2010)
Avatar (2009)
```

Model Evaluation

Evaluated accuracy using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE)

RMSE: refers to the square root of the mean of the squared errors.

RMSE = $((\sum i=1 \text{ to } n(yi-y\hat{}i)^2)/n)^{0.5}$

where:

n is the number of observations,
yi is the actual value of the target variable for observation i,
y^i is the predicted value of the target variable for observation i.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}}$$

Mean Absolute Error (MAE): It is calculated as the average absolute differences between the predicted and actual values

MAE = $1/n\sum$ i=1 to n $|Actual_i - Predicted_i|$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |Actual_i - Predicted_i|$$

where:

n is the total number of predictions.

$Actual_i$ is the actual rating for the i-th prediction.

$Predicted_i$ is the predicted rating for the i-th prediction.

```
predictions = model.test(testset)


rmse = accuracy.rmse(predictions)
print(f'RMSE: {rmse}')

mae = accuracy.mae(predictions)
print(f'MAE: {mae}')
```

```
RMSE: 0.9816
RMSE: 0.9815837602600298
MAE:  0.7651
MAE: 0.765058403010988
```

Pearson Correlation Coefficient

- Correlation measures the linear relationship between two variables
- Ranges from -1 to 1, where 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 indicates no linear correlation
- Widely used in various domains, including finance, biology, and collaborative filtering
- Handling Sparsity: Can be sensitive to sparsity and may not perform well when there are many missing values in the data
- Formula: $\text{pearson\_correlation}(A,B) = (\text{cov}(A,B))/(\sigma A \cdot \sigma B)$

THANK YOU!