

# Movie Recommendation Engine



Presented By:

Deep Patel (18BEE021)

Parth Patel (18BEE081)

Vivek Savaliya (18BEE104)

Prince Vaghasiya (18BEE120)

Presented to:

Prof. Usha Patel



# CONTENTS

## 1. INTRODUCTION

- Project Objectives
- Flow of the Project
- Need of Recommendation Engine

## 2. CONCEPT OF RECOMMENDATION ENGINE

## 3. TYPES OF RECOMMENDATION ENGINE

- Content Based Recommendation Engine
- Collaborative Filtering Recommendation Engine
- Matrix Factorization

## 4. ISSUES ASSOCIATED WITH RECOMMENDER SYSTEMS

## 5. CONCLUSION

## 6. REFERENCE

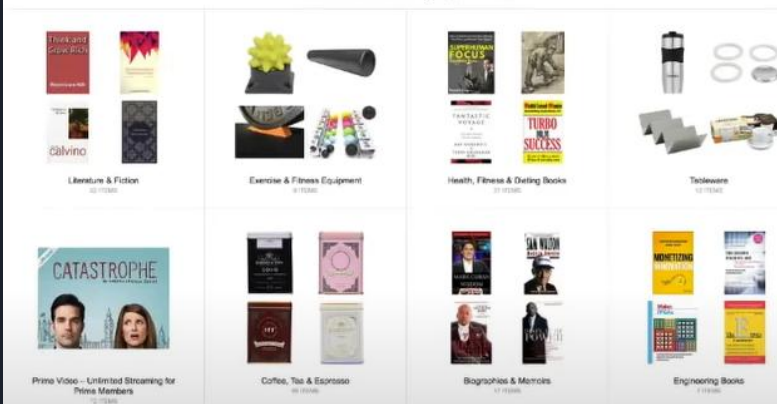
# INTRODUCTION

*“What movie should I watch this evening?”*

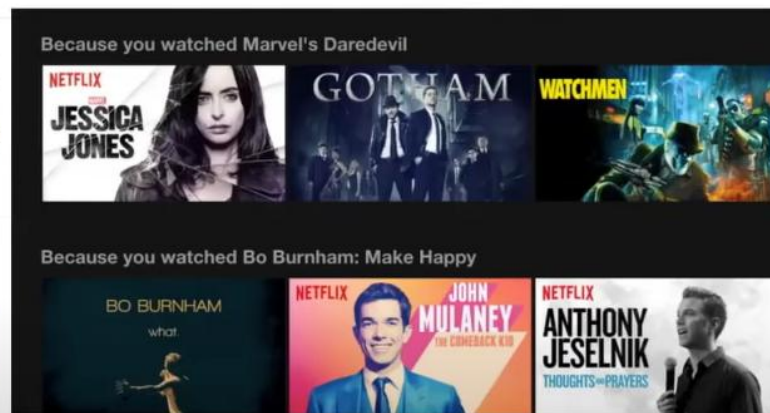
## What's Common?

### 1. Amazon

Recommended for you, Thomas



### 2. Netflix





# PROJECT OBJECTIVES

- Working on Algorithm for recommending movies.
- Working with a large datasets.
- Implement the processed data in the Algorithm.

# NEED OF RECOMMENDATION ENGINE

- Customer Satisfaction
- Providing Reports
- Revenue Maximization

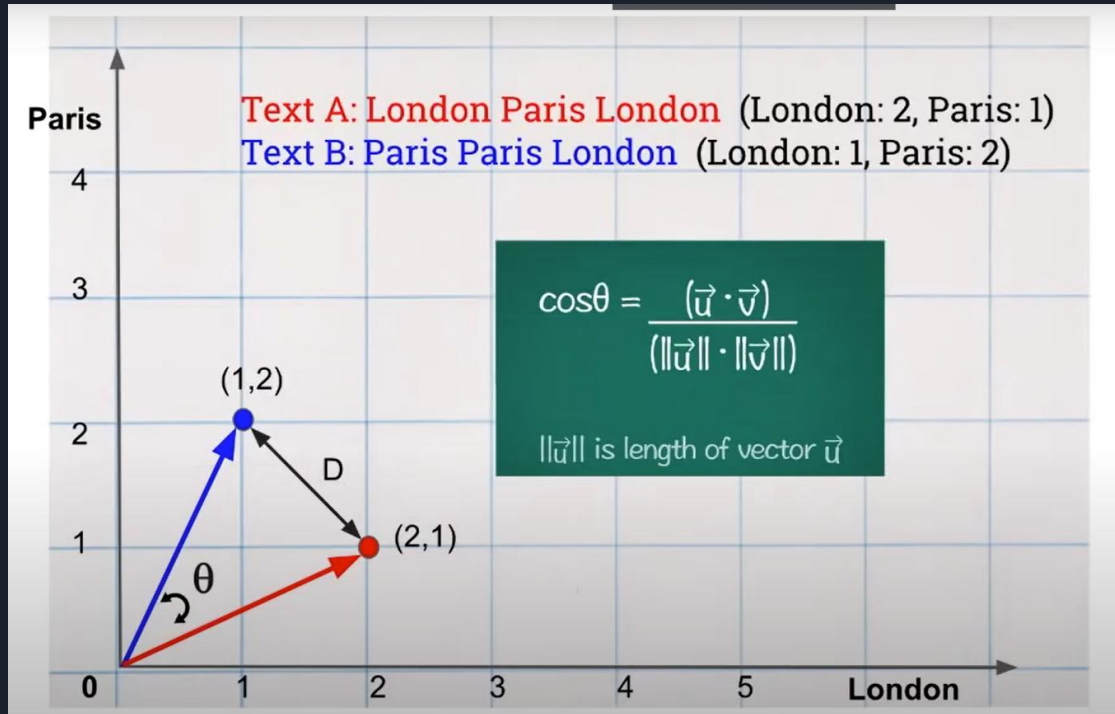


Users



Items

# CONCEPT OF SIMILARITY FOR RECOMMENDATION





# Continue...

```
In [1]: > import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

```
In [2]: > text = ["London Paris London", "Paris Paris London"]
```

```
In [3]: > cv = CountVectorizer()
count_matrix = cv.fit_transform(text)
```

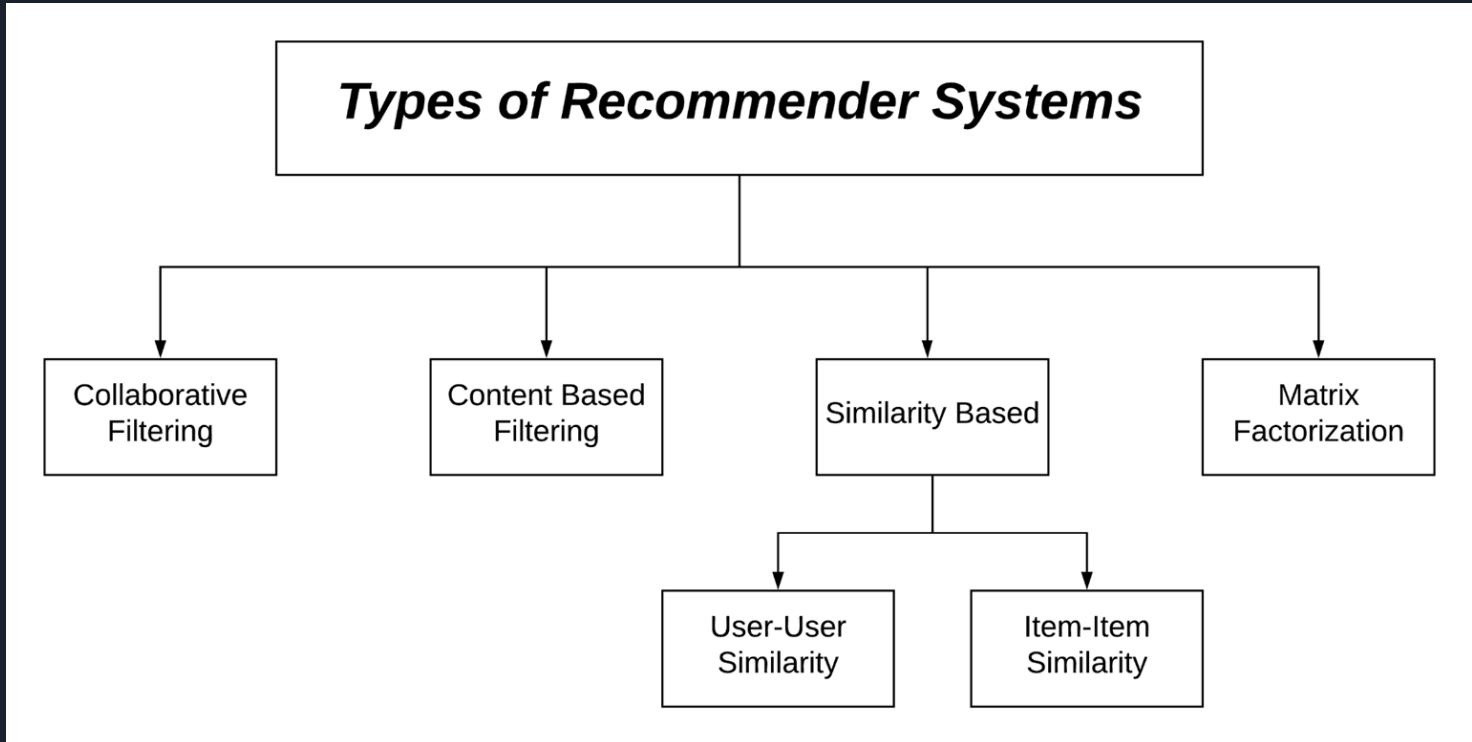
```
In [4]: > print(cv.get_feature_names())
print(count_matrix.toarray())
```

```
['london', 'paris']
[[2 1]
 [1 2]]
```

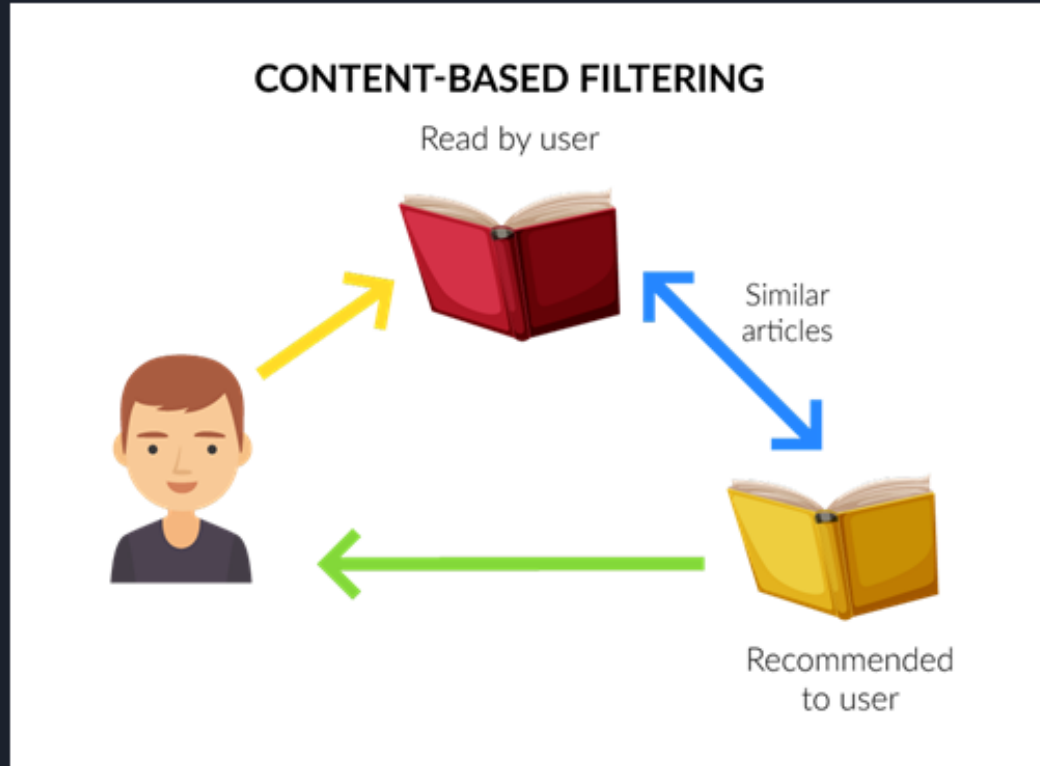
```
In [5]: > similarity_scores = cosine_similarity(count_matrix)
print(similarity_scores)
```

```
[[1.  0.8]
 [0.8 1.  ]]
```

# TYPES OF RECOMMENDATION ENGINE

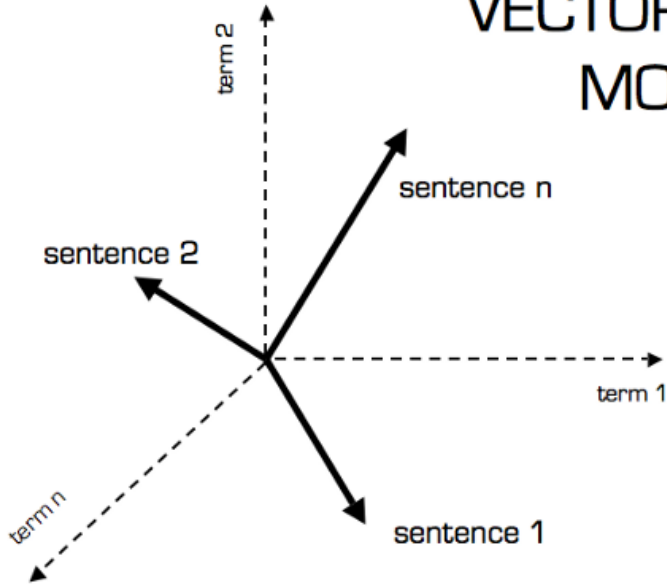


# CONTENT BASED RECOMMENDATION ENGINE

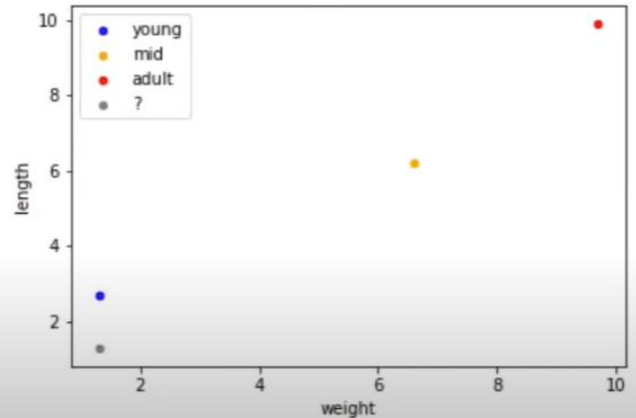
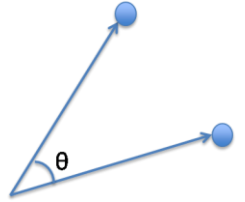


# CONCEPT OF CONTENT BASED RECOMMENDATION ENGINE

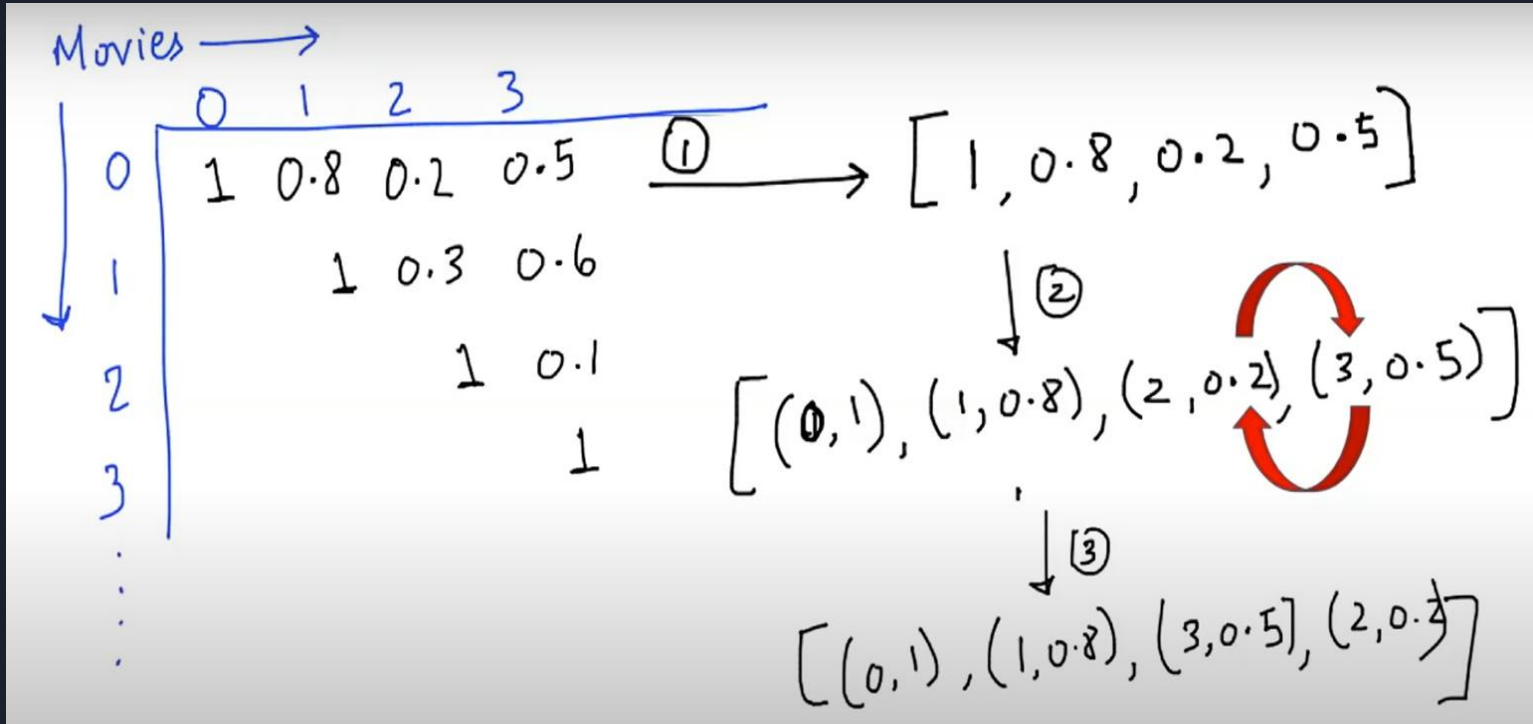
## VECTOR SPACE MODEL



$$\text{sim}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$



Continue...



# IMPLEMENTING CONTENT BASED RECOMMENDATION ENGINE

```
✓ [17] movie_user_likes = "Interstellar"
      0s
      movie_index = get_index_from_title(movie_user_likes)
      similar_movies = list(enumerate(cosine_sim[movie_index])) #accessing the row corresponding to given movie to find all the similarity scores for
```

```
✓ [18] sorted_similar_movies = sorted(similar_movies, key=lambda x: x[1], reverse=True)[1:]
      0s
```

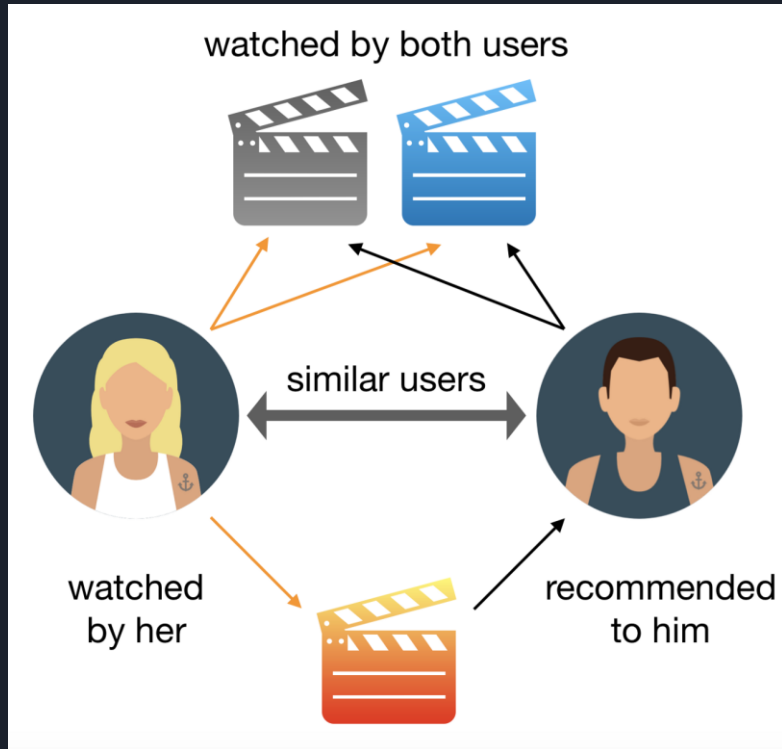
```
✓ [19] i=0
      0s
      print("Top 5 similar movies to "+movie_user_likes+" are:\n")
      for element in sorted_similar_movies:
          print(get_title_from_index(element[0]))
          i=i+1
          if i>5:
              break
```

Top 5 similar movies to Interstellar are:

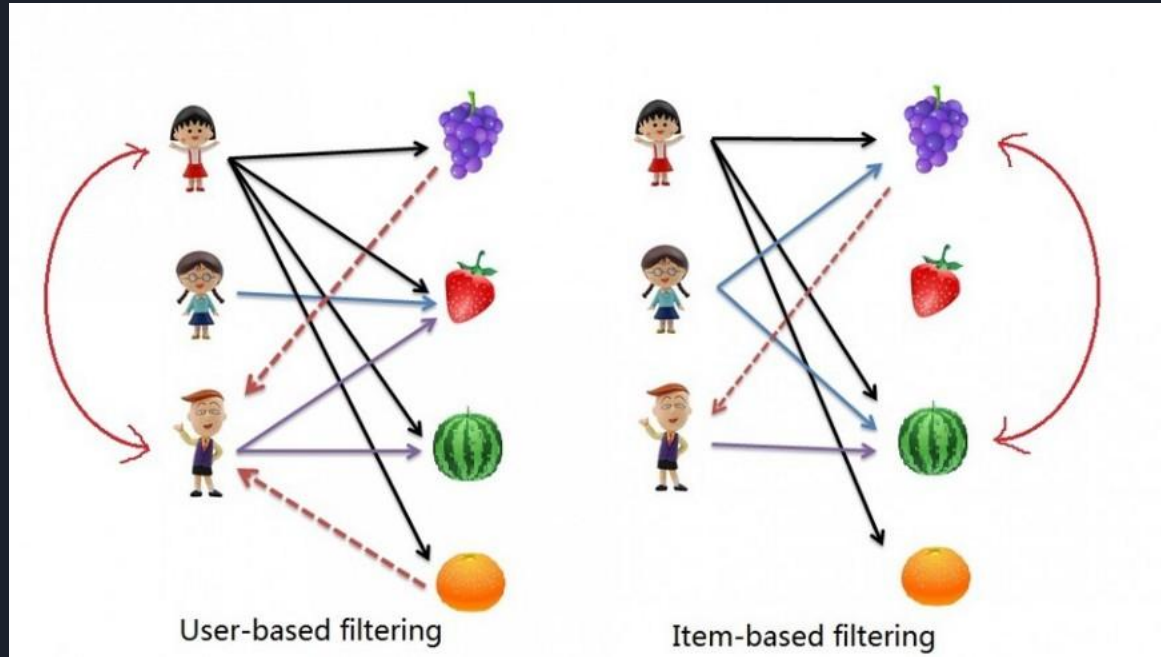
The Matrix Revolutions  
Midnight Special  
The Matrix Reloaded  
The Martian  
The Terminator  
Armageddon

✓ 0s completed at 10:23 PM

# COLLABORATIVE FILTERING RECOMMENDATION ENGINE



# TYPES OF COLLABORATIVE FILTERING RECOMMENDATION ENGINE





# Cont...

	User 1	User 2	User 3
Movie 1	5	2	4
Movie 2	3	4	3
Movie 3	1	4	1
Movie 4	2		??
Movie 5	5	2	??

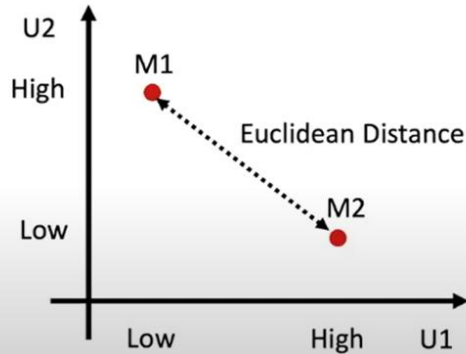
User-User Collaborative Filtering

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	5	3	1	2	5
User 2	2	4	4		2
User 3	4	3	1	??	??

Item-Item Collaborative Filtering

# CONCEPT OF COLLABORATIVE FILTERING RECOMMENDATION ENGINE

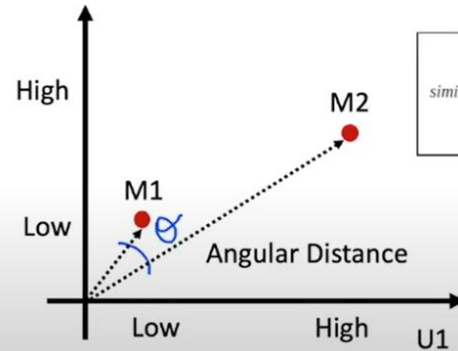
## Quantifying the Similarity



Dissimilar Users

Option 1: Cosine Distance

Option 2: Pearson Correlation



Similar Users

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Continue...

		users												sim(1,m)
		1	2	3	4	5	6	7	8	9	10	11	12	
movies	1	1		3		?	5			5		4		1.00
	2			5	4			4			2	1	3	-0.18
	<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
	4		2	4		5			4			2		-0.10
	5			4	3	4	2					2	5	-0.31
	<u>6</u>	1		3		3			2			4		<u>0.59</u>

# IMPLEMENTING COLLABORATIVE FILTERING BASED RECOMMENDATION ENGINE

```
✓ 0s ▶ action_lover = [("Amazing Spider-Man, The (2012)",5),("Mission: Impossible III (2006)",4),("Toy Story 3 (2010)",2),("2 Fast 2 Furious (Fast and Furious 2, The) (2003)",4)]
similar_movies = pd.DataFrame()
for movie,rating in action_lover:
    similar_movies = similar_movies.append(get_similar(movie,rating),ignore_index = True)

similar_movies.head(10)
similar_movies.sum().sort_values(ascending=False).head(20)
```

Amazing Spider-Man, The (2012)	3.233134
Mission: Impossible III (2006)	2.874798
2 Fast 2 Furious (Fast and the Furious 2, The) (2003)	2.701477
Over the Hedge (2006)	2.229721
Crank (2006)	2.176259
Mission: Impossible - Ghost Protocol (2011)	2.159666
Hancock (2008)	2.156098
The Amazing Spider-Man 2 (2014)	2.153677
Hellboy (2004)	2.137518
Snakes on a Plane (2006)	2.137396
Jumper (2008)	2.129716
Chronicles of Riddick, The (2004)	2.121689
Tron: Legacy (2010)	2.111843
Fantastic Four (2005)	2.083022
X-Men: The Last Stand (2006)	2.077530
Wreck-It Ralph (2012)	2.067907
Kung Fu Hustle (Gong fu) (2004)	2.067457
Godzilla (2014)	2.061653
Incredible Hulk, The (2008)	2.050104
Quantum of Solace (2008)	2.016189

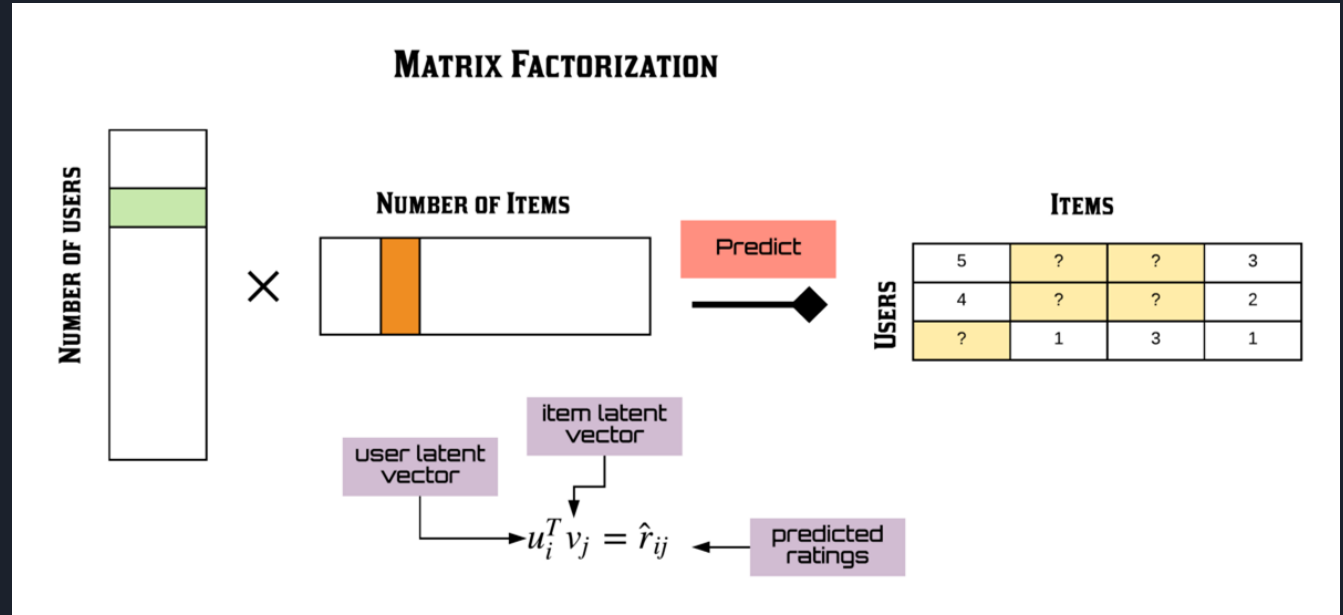
dtype: float64

✓ 0s completed at 1:09 AM

01:09 10.11.21

# MATRIX FACTORIZATION

- Need of Matrix Factorization
- Reasons to Reduce Dimensions
- Advantages of Reducing Dimensions



# CONCEPT OF MATRIX FACTORIZATION

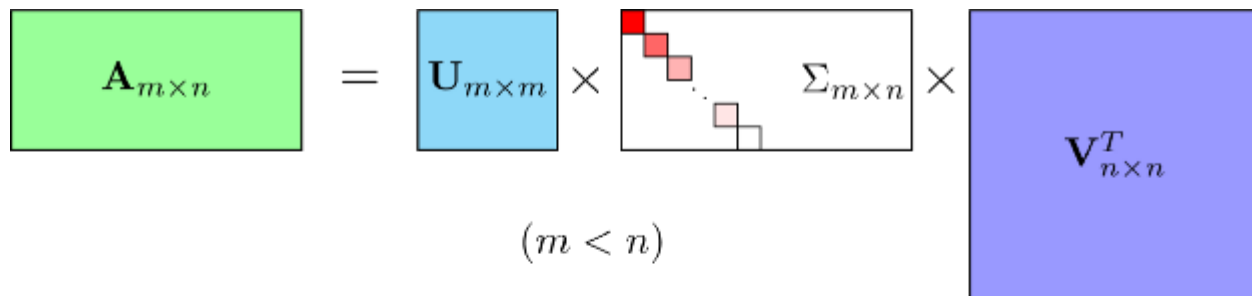


Diagram illustrating the concept of matrix factorization for the case where  $m < n$ . The matrix  $\mathbf{A}_{m \times n}$  (green box) is equal to the product of three matrices:  $\mathbf{U}_{m \times m}$  (blue box),  $\Sigma_{m \times n}$  (white box with a diagonal of red and pink squares), and  $\mathbf{V}_{n \times n}^T$  (purple box). The condition  $(m < n)$  is noted below the  $\Sigma$  matrix.

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \times \Sigma_{m \times n} \times \mathbf{V}_{n \times n}^T$$

$(m < n)$

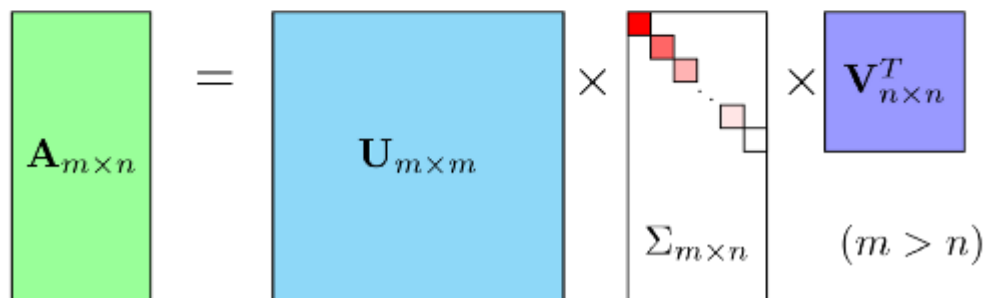


Diagram illustrating the concept of matrix factorization for the case where  $m > n$ . The matrix  $\mathbf{A}_{m \times n}$  (green box) is equal to the product of three matrices:  $\mathbf{U}_{m \times m}$  (blue box),  $\Sigma_{m \times n}$  (white box with a diagonal of red and pink squares), and  $\mathbf{V}_{n \times n}^T$  (purple box). The condition  $(m > n)$  is noted below the  $\Sigma$  matrix.

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \times \Sigma_{m \times n} \times \mathbf{V}_{n \times n}^T$$

$(m > n)$

# IMPLEMENTING MATRIX FACTORIZATION BASED RECOMMENDATION ENGINE

```
predictions = recommend_movies(preds, 1310, movies, ratings, 20)

predictions
```

	movie_id	title	genres
1618	1674	Witness (1985)	Drama Romance Thriller
1880	1961	Rain Man (1988)	Drama
1187	1210	Star Wars: Episode VI - Return of the Jedi (1983)	Action Adventure Romance Sci-Fi War
1216	1242	Glory (1989)	Action Drama War
1202	1225	Amadeus (1984)	Drama
1273	1302	Field of Dreams (1989)	Drama
1220	1246	Dead Poets Society (1989)	Drama
1881	1962	Driving Miss Daisy (1989)	Drama
1877	1957	Chariots of Fire (1981)	Drama
1938	2020	Dangerous Liaisons (1988)	Drama Romance
1233	1259	Stand by Me (1986)	Adventure Comedy Drama
3011	3098	Natural, The (1984)	Drama
2112	2194	Untouchables, The (1987)	Action Crime Drama
1876	1956	Ordinary People (1980)	Drama
1268	1296	Room with a View, A (1986)	Drama Romance
2267	2352	Big Chill, The (1983)	Comedy Drama
1278	1307	When Harry Met Sally... (1989)	Comedy Romance
1165	1186	Sex, Lies, and Videotape (1989)	Drama
1199	1222	Full Metal Jacket (1987)	Action Drama War
2833	2919	Year of Living Dangerously (1982)	Drama Romance



# ISSUES ASSOCIATED WITH RECOMMENDER SYSTEMS

- Handling Unknown Users/Items (Cold Start Problem)
- Data Sparsity
- Scalability
- Dynamic Updates



# CONCLUSION

- Recommendation Engine is for sure a companion and advisor to help us make the right choices by providing us tailored options and creating a personalized experience for us.
- It is beyond any doubt that recommendation engines are getting popular and critical in the new age of things.





# REFERENCES

- <https://grouplens.org/datasets/movielens/>
- <https://www.kaggle.com/tmdb/tmdb-movie-metadata>
- <https://unsplash.com/s/photos/movie>

***Thank you!***