



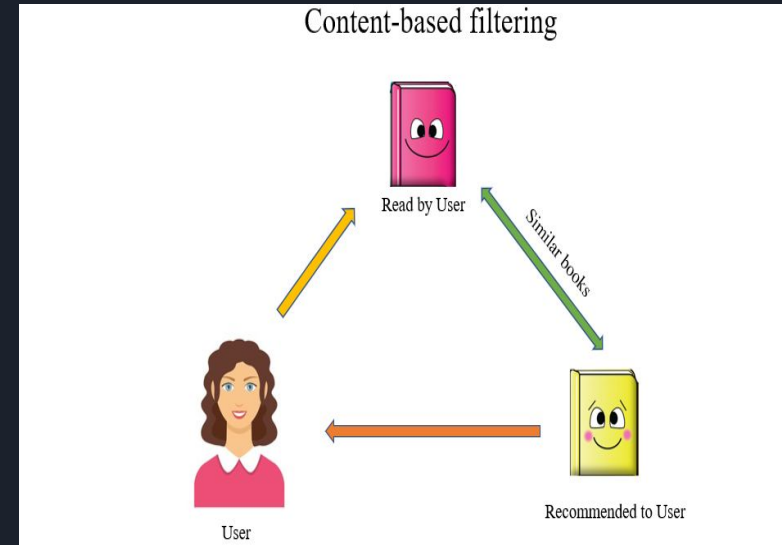
# Movie Recommendation System

## Mini Project

Netradeepak Chinchwadkar(IMT2020014)  
Aryan Bhatt (IMT2020020)  
Satvik Verma

# Recommendation System

- ❖ **Personalized Suggestions:** Recommendation systems analyze user preferences and behaviors to offer personalized suggestions, enhancing the user experience by presenting relevant content or products.
- ❖ **Increased Engagement:** By guiding users towards items or content they are likely to enjoy, recommendation systems can boost engagement metrics such as click-through rates, time spent on platform, and overall user satisfaction.
- ❖ **Improved Discoverability:** These systems help users discover new and diverse items or content within a vast selection, facilitating exploration and serendipitous discovery beyond their immediate interests.





# Dataset Used

We are using **Movie Lens 1M** for this project as dataset. It consist of:

- ❖ Movies.csv
- ❖ Ratings.csv
- ❖ Users.csv


## Movie.csv

ID		Title	Genres
0	1	Toy Story (1995)	Animation Children's Comedy
1	2	Jumanji (1995)	Adventure Children's Fantasy
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
3878	3948	Meet the Parents (2000)	Comedy
3879	3949	Requiem for a Dream (2000)	Drama
3880	3950	Tigerland (2000)	Drama
3881	3951	Two Family House (2000)	Drama
3882	3952	Contender, The (2000)	Drama Thriller

3883 rows × 3 columns

## Rating.csv

UserID	MovieID	Rating	Timestamp
1	1193	5	978300760
1	661	3	978302109
1	914	3	978301968
1	3408	4	978300275
1	2355	5	978824291
...	...	...	...
6040	1091	1	956716541
6040	1094	5	956704887
6040	562	5	956704746



```
# Number of users
```

```
print('The ratings dataset has', rating['UserID'].nunique(), 'unique users')
```

```
# Number of movies
```

```
print('The ratings dataset has', rating['MovieID'].nunique(), 'unique movies')
```

```
# Number of ratings
```

```
print('The ratings dataset has', rating['Rating'].nunique(), 'unique ratings')
```

```
# List of unique ratings
```

```
print('The unique ratings are', sorted(rating['Rating'].unique()))
```

✓ 0.0s

The ratings dataset has 6040 unique users

The ratings dataset has 3706 unique movies

The ratings dataset has 5 unique ratings

The unique ratings are [1, 2, 3, 4, 5]



# Neighbourhood based Collaborative Filtering

## ❖ **Introduction to Neighborhood-Based Collaborative Filtering (CF):**

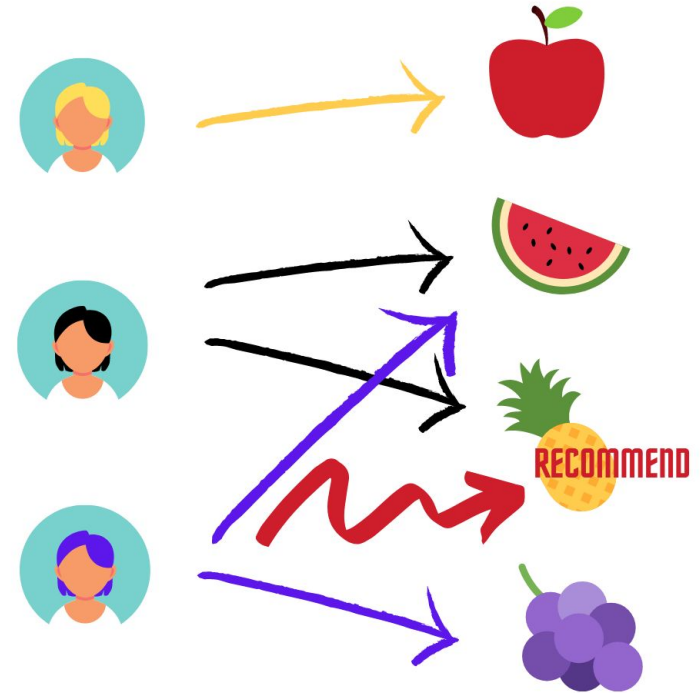
- Neighborhood-based CF is a popular approach in recommendation systems that relies on similarities between users or items.
- Instead of analyzing all users or items, it focuses on a subset, or "neighborhood," that are most similar to the target user or item.
- This method leverages the idea that users with similar preferences tend to like similar items, and vice versa.


## ❖ **Types of Neighborhood-Based CF:**

- User-Based CF: This method identifies similar users based on their past interactions and recommends items liked by those similar users.
- Item-Based CF: Here, similarities between items are calculated based on user interactions, and items with high similarity to those already liked by the user are recommended.
- These approaches can be further refined using different similarity metrics such as Pearson correlation, cosine similarity, or Jaccard similarity.

# User-User based Filtering

- ❖ **User Similarity Calculation:** User-user based collaborative filtering computes the similarity between users by analyzing their past interactions with items. This is typically done using metrics such as cosine similarity or **Pearson correlation** coefficient, quantifying the likeness of preferences between users.
- ❖ **Neighborhood Selection:** Once user similarities are determined, the system selects a subset of users, often referred to as the "neighborhood," who exhibit the highest similarity to the target user. This neighborhood serves as the basis for generating recommendations, focusing on users whose preferences align closely with the target user's.
- ❖ **Recommendation Generation:** Recommendations are generated by aggregating the preferences of users within the selected neighborhood. Items that have been positively rated by similar users but not yet consumed by the target user are suggested, aiming to provide personalized recommendations that reflect the user's tastes and preferences.





```
topUsersRating=topUsers.merge(rating, left_on='UserID', right_on='UserID', how='inner')
topUsersRating.head()
```

2] ✓ 0.0s

	similarityIndex	UserID	MovieID	Rating
0	1.0	352	2987	4
1	1.0	352	1179	4
2	1.0	352	647	4
3	1.0	352	648	1
4	1.0	352	2120	4

```
pearsonDF = pd.DataFrame.from_dict(pearsonCorDict, orient='index')
pearsonDF.columns = ['similarityIndex']
pearsonDF['UserID'] = pearsonDF.index
pearsonDF.index = range(len(pearsonDF))

topUsers=pearsonDF.sort_values(by='similarityIndex', ascending=False)[0:100]
topUsers.head()
```

1] ✓ 0.0s

	similarityIndex	UserID
87	1.000000	352
81	1.000000	245
58	0.980196	4808
63	0.980196	5788
12	0.980196	869

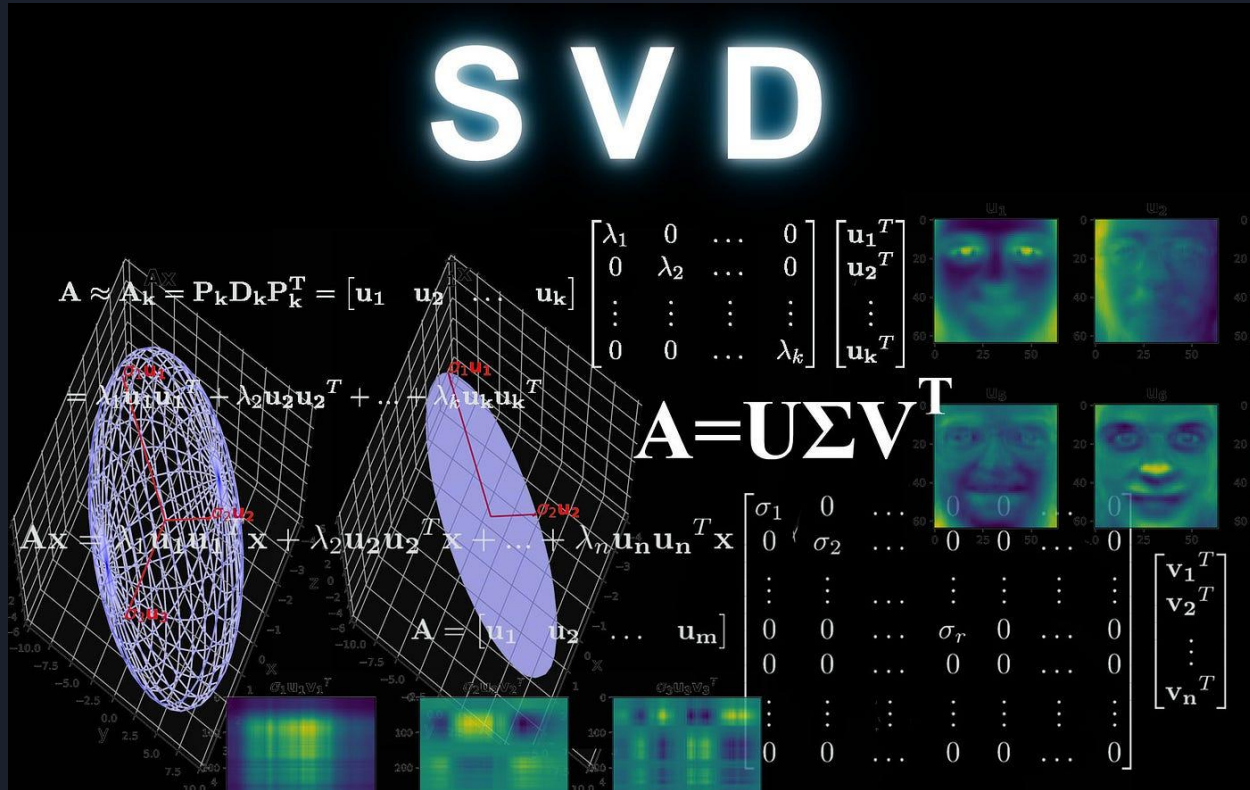
```
recommendation_df = recommendation_df.sort_values(by='weighted average recommendation score',  
ascending=False)  
recommendation_df.head(5)
```

✓ 0.0s

	weighted average recommendation score	MovieID
MovieID		
3658	138.058971	3658
3659	138.058971	3659
3879	21.795175	3879
2585	21.144750	2585
1493	17.192635	1493

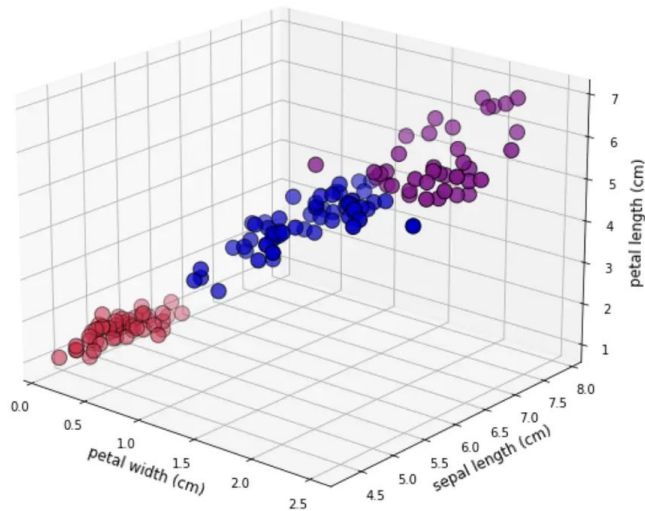


# Singular Value Decomposition

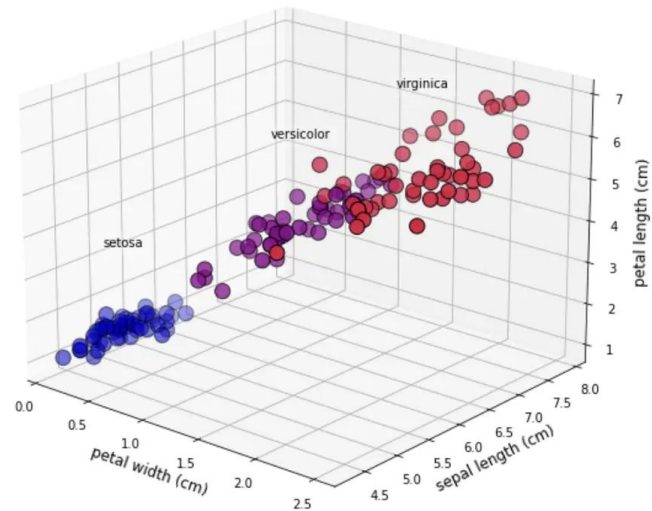


# K-Means Clustering

K-Means Clusters for the Iris Dataset



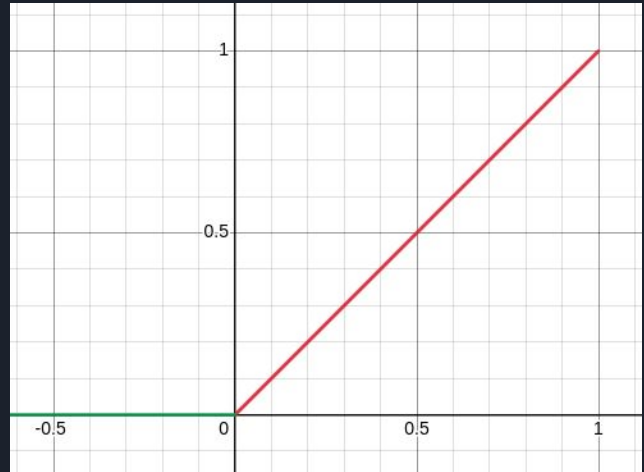
Actual Labels for the Iris Dataset



# Preprocessing for K-Means

The following are listed down as novelties in our approach towards preprocessing over what is expected and standard:

1. The first digit of zip code and occupation is one-hot encoded to make user similarity more evident
2. The year of release is incorporated into the genre by the broad scale of decade of release
3. The User x Genre\_Rating matrix is calculated by getting the time weighted average of all reviews for that genre
4. Instead of replacing NaN values by global mean, they are placed a two-fold group mean



# K-Means Algorithm

- After a series of tests to check best possible number of clusters used and number of eigenvalues dropped the following were decided (Number of clusters=26 Removed values=10)
- Best genres are calculated by K-Means but we need to recommend movies
- The following formula was used to map genre-wise rating to movie-wise rating
- The movie's affinity = movie's rating \* (sum of ratings of genres for that movie)
- The 5 movies with maximum affinity for the user are recommended

Number of Clusters	Number of values dropped			
	NaN	5	10	15
5	0.03563436635			
7	0.03563903802		0.038872708	
8	0.03567797821		0.038283075	
9	0.03555737115	0.039139524	0.03756817	
10	0.03558929697		0.036655848	
11	0.0356578263		0.035163748	
12	0.03573313102		0.035071203	
13	0.03562840068		0.037767607	
14	0.03560674841		0.035606748	
15	0.03568096218		0.036309271	0.035087772
16			0.0368589	0.034750942
17			0.037604071	0.037604071
18			0.034052085	0.035380416
19			0.033802979	0.033802979
20			0.033983736	0.032615908
21			0.032401519	0.034159006
22			0.032288519	
23			0.037026563	
24			0.031468201	
25	0.03579637205	0.032860516	0.030551421	
26	0.03574630994	0.031505366	0.030356796	0.033836082
27			0.033271834	
35	0.03589513893			