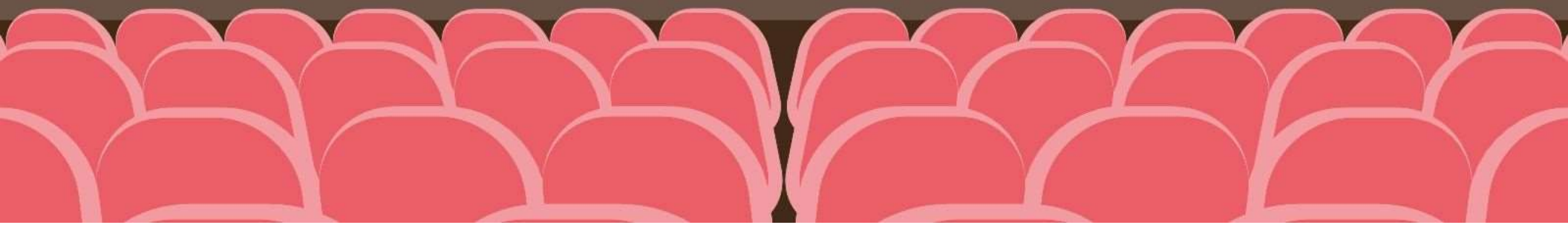
Movie Recommender System

Presented By:
Vedant Narendra Joshi

University ID:
801305907



The Problem Statement

Why this topic?

Due to overwhelming amount of options available on the internet, it is necessary to filter, organize and effectively distribute relevant information in order to address the issue of excess information that many internet users have been experiencing.

Recommender systems try to mitigate this problem of having huge options by filtering out the right content for them.

Similar to this, when we talk of entertainment, we think about movies. The internet has tens of thousands of films from various eras. Users sometimes struggle to decide what to watch, thus it is advantageous to develop a movie recommendation system based on user ratings, genres, preferences and other factors.

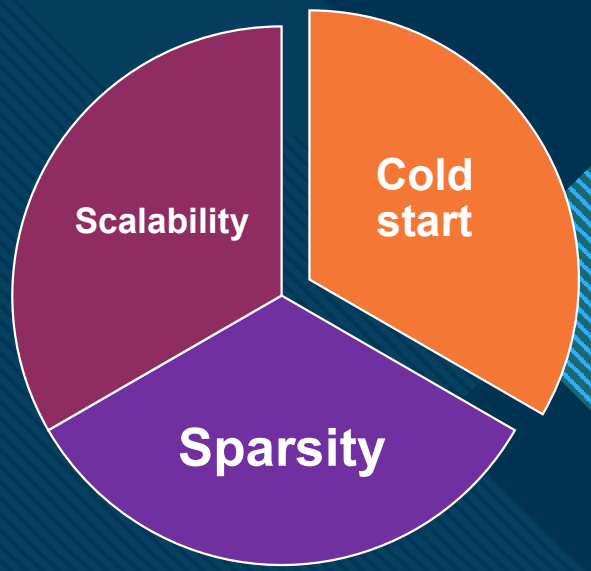
In this project, I'll explore several different algorithms on the MovieLens Dataset and analyze which one provides the most accurate recommendation.

The Problems and Challenges

- **Scalability**: In many of the environments in which these systems make recommendations, there are millions of users and products. Thus, large number of computation power is often necessary to calculate the recommendations.

- **Sparsity:** The number of movies watched on major OTT platforms is extremely large. The most active users will have rated a small subset of the overall database. Thus, even the most popular movies may have very few ratings.

- **Cold Start:** These systems often require a large amount of existing data on a user in order to make accurate recommendations



The Motivation

We frequently favor items that are comparable to other things we enjoy.

We also prefer to like items that people who are similar to us like.

These patterns can be utilized to create predictions about future developments.

These advances can be managed with the use of **recommendation systems.**

The Data [3]

Dataset Name: MovieLens 100K Dataset

Dataset Description: The dataset Consists of following features:-

1) 100,000 ratings (1-5) from 943 users on 1682 movies.

2) Each user has rated at least 20 movies.

3) Simple demographic info for the users (age, gender, occupation, zip)

Dataset Link:
https://grouplens.org/datasets/movielens/100k/

The Existing Approach [1]

- Data pre-processing becomes challenging due to the enormous amount of data that is available to us in the data.
- To mitigate this problem, The Singular Value decomposition approach for dimension reduction is introduced in approach that follows.
- For the classification of the movies into classes and for its further distribution after Recommendations, KMeans Clustering Method was used.
- Metrics like the standard deviation (SD), root mean square error (RMSE), mean absolute error (MAE), t-value, Dunn index, average similarity, and calculation time were used to evaluate the model.

The findings achieved using the above method are shown on the next slide.->

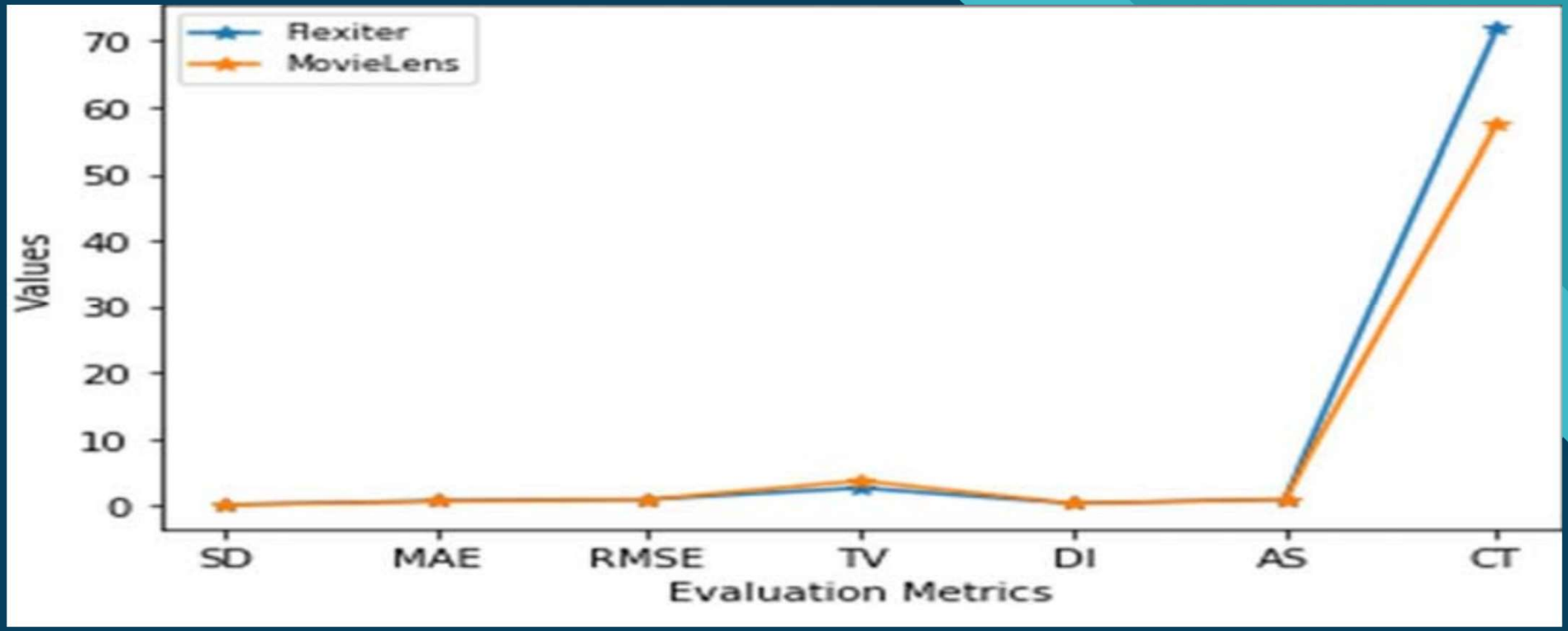
Continued Approach [1]

- Two Datasets, Flexiter and MovieLens were compared on the following Metrics and computation time was calculated.

	Flexiter	MovieLens
SD	0.13743	0.11453
MAE	0.73372	0.62896
RMSE	0.94876	0.92934
T-value	2.66674	3.74562
Dunn Index (4 Clusters)	0.34873	0.31945
Average Similarity (4 Clusters)	0.96	0.95
Computational Time in sec	71.89	57.43

Through Average Similarity the highest accuracy was obtained.
Following is the Graph plotted comparing the 2 Datasets.

Continued…



- The approach involving SVD and KMeans perform's well on the MovieLens Dataset than the Flexiter Movie Dataset.

The Existing Approach [2]

- Recommendation system is of 2 types namely, Collaborative Filtering and Content Based Filtering.
- In this research paper, the authors have used a Collaborative filtering Technique for their Recommendation system.
- Similar to everything, Data pre-processing and classification needs to be performed by any system.
- For this purpose, the authors have designed their data preparation and data analysis using Apache Mahout.

The findings achieved using the above method are shown on the next slide.->

Continued Approach[2]

- The Similarity of the movies was calculated using the Pearson Correlation Coefficient.

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

Figure.6 Pearson Correlation Coefficient formula

With the Apache Mahout framework, the MapReduce helps match recommendations to similar Movies.

The lord of the Rings: The Fellowship of t e Ring 12(01)	The lo rd of the Rings: The Two Towe rs 12(02)	0.999549
The Empire Strikes Back (1980)	Star Wars (1977)	0.999477 5
I diana Jones and the la st Crusade (19B9)	I diana Jones and t e Temple of Doom (1984)	0.999282 9
E.T. The Extr. -Terrestrl.1 (1982)	S .r W.n (1977)	0.9990453
The Godf.ther (1972)	The Godfat he r Part II (1974)	0.9989032
Pirates olthe Caribbean: The Curse of the Bl. ...	The Leag e of Extraordinary Gentlemen	0.99869
The M.ul. Reloaded (2003)	Bruce 101 hty (2003)	0.998663
Jeepers Creepers 2 12(03)	Fredd y vs. Jasee (2003)	0.99858
Harry Poner Andt e Cha berofSecrets 12(02)	The lord of the Rings: The Fellowship olthe	0.998 73
Signs 12(02)	Shre (2001)	0.998355
2 Fast H rlous (2003)	Bruce Al ighty 12(03)	0.998 32
H.rry Pott er Andt e Cha rnber of Secrets 12(02)	Shre (2001)	0.998276
The Texas Chains aw Massacre (2003)	Scary Movie 3 12(03)	0.998155
The Leag e of Extraordinary Gentle en (2003)	Bad Boys II (2003)	0.998154
Ice Age (2002)	S re 12(01)	0.99796
ill Bill Vol. 1 (2003)	Underworld (2003)	0.997915
Austin Powers In Goldmemb er (2002)	Signs 12(02)	0.997732
D. ddy D.y Care (2003)	Bruce Al ighty (2003)	0.997505
The Mu my (1999)	The Mu y Returns (2001)	0.997426
2 Fast 2 F rious (2003)	The Matrix Relo.ded (2003)	0.997' 3
IIIBill Vol. 1 12(03)	The Te.as Ch.ins. Massacre (2003)	0.997349
The F.st and the F rious 12(01)	XXX (2002)	0.996123
How to De.1 12(03)	B.d Boys 1112(03)	0.99 458
Down wit love (2003)	owtolose a Guy; 10 D. ys (2003)	0.992708
IThe Scorpion ing (2002)	The Mummy Returns (2001)	0.99107
Der alled (2002)	The Order{2(01)	0.9B559
Fr. n Sinatra - 3 P.ck(2002)	Great Music. ls - Vol. 2 (1951	0.952113
Biogr.p y: Bette D.vis (1926)	The Bette Davis Collection (1993)	0.952113
Someon e like Yo (1991)	Someon e like You Whereat e Heart 1512(02)	0.94 782
Be. uty a dthe Beast 12000 1	Be. uty and the Beast 119621	0.9 1396

Figure.8 Movies Similarities (Item Based)

Methodology - Part 1

Initially, we load the MovieLens Dataset using the pandas Library.

- The Dataset is divided into 2 parts, namely, the movie Data and the user ratings.

The dataset is not separated initially, so we first separate the data so we can use it for further process.

- The dataset was not separated, hence for this part, I separated the files using pd.read function and encoding techniques.

Since the dataset is big, I did a lot of data visualization to view any discrepancies and ambiguities.

- Graphs using matplotlib were plotted for the visualization.

Methodology – Part 2

- The project was run without clearing the multiple entries in the dataset.
- I cleaned the data by removing the multiple entries and created a refined Dataset.
- This gave a clearer picture in the analysis of the data.

Looking at length of original items_dataset and length of unique combination of rows in items_dataset after removing movie id column

```
len(items_dataset.groupby(by=column_names2[1:])),len(items_dataset)
```

```
(1664, 1682)
```

We can see there are 18 extra movie id's for already mapped movie title and the same duplicate movie id is assigned to the user in the user-item dataset.

Methodology – Part 2, continued…

Example of newly created Dataset by merging.

```
merged_dataset = pd.merge(dataset, movie_dataset, how='inner', on='movie id')
merged_dataset.head()
```

	user id	movie id	rating	timestamp	movie title
0	196	242	3	881250949	Kolya (1996)
1	63	242	3	875747190	Kolya (1996)
2	226	242	5	883888671	Kolya (1996)
3	154	242	3	879138235	Kolya (1996)
4	306	242	5	876503793	Kolya (1996)

A dataset is created from the existing merged dataset by grouping the unique user id and movie title combination and the ratings by a user to the same movie in different instances (timestamps) are averaged and stored in the new dataset.

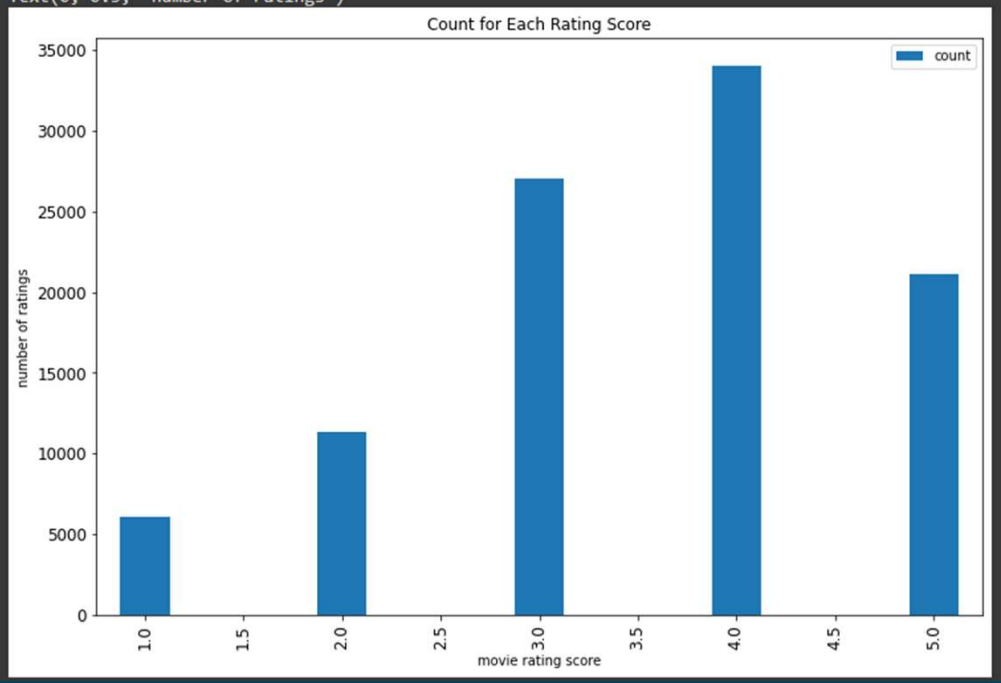
Methodology – Part 3, Exploratory Data Analysis

After Data Cleaning, we have plotted the user ratings against the number of ratings.

```
ax = rating_count_df.reset_index().rename(columns={'index': 'rating score'}).plot('rating','count',
    figsize=(12, 8),
    title='Count for Each Rating Score',
    fontsize=12)

ax.set_xlabel("movie rating score")
ax.set_ylabel("number of ratings")
```

```
Text(0, 0.5, 'number of ratings')
```



- **We can see that number of 1.5, 2.5, 3.5, 4.5 ratings by the users are comparatively negligible.**

Methodology - Part 3, Exploratory Data Analysis, continued...

Ratings for the movies not seen by a user is by default considered as 0.
We added that to the existing dataset and observed the following graphs.

```
[17] total_count = num_items * num_users
     zero_count = total_count-refined_dataset.shape[0]
     zero_count

     1469459
```

```
# append counts of zero rating to df_ratings_cnt
rating_count_df = rating_count_df.append(
    pd.DataFrame({'count': zero_count}, index=[0.0]),
    verify_integrity=True,
).sort_index()
rating_count_df
```

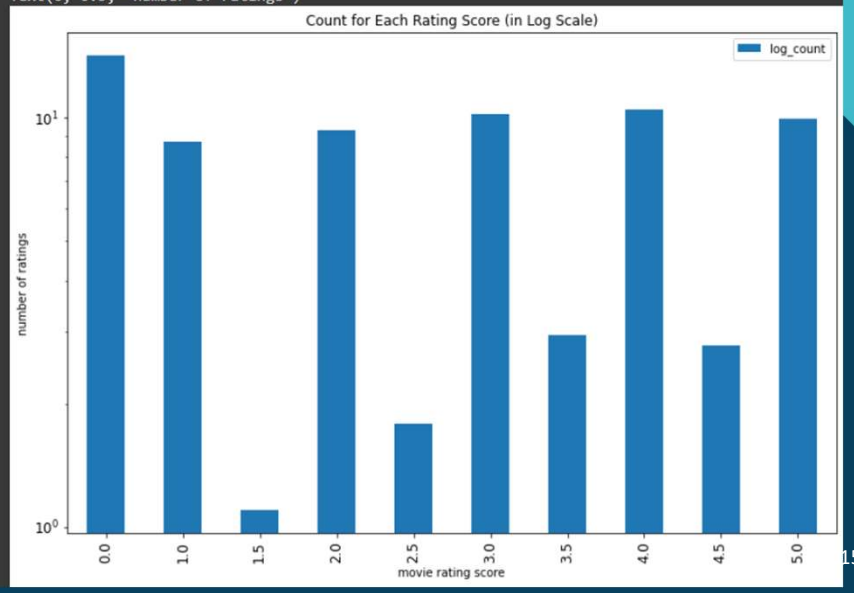
	count
0.0	1469459
1.0	6083
1.5	3
2.0	11334
2.5	6
3.0	27060
3.5	19
4.0	34042
4.5	16
5.0	21130

As the number of 0 ratings is tremendously huge. It creates a sparsity in our dataset and hence a lot of computation power is required.
To mitigate this problem, we log transform the dataset and plot the graph.

```
ax = rating_count_df.plot('rating score', 'log_count', 'bar', figsize=(12, 8),
    title='Count for Each Rating Score (in Log Scale)',
    logy=True,
    fontsize=12,)

ax.set_xlabel("movie rating score")
ax.set_ylabel("number of ratings")
```

```
Text(0, 0.5, 'number of ratings')
```



Methodology - Part 4, Creating a KNN Model.

After the Data Cleaning & Exploratory Data Analysis, Its time to create a KNN model using a Cosine similarity metric for making recommendations. We convert the dataset into a SciPy sparse matrix to reduce the computation effort.

```
[26] # transform matrix to scipy sparse matrix
     user_to_movie_sparse_df = csr_matrix(user_to_movie_df.values)
     user_to_movie_sparse_df
```

```
<943x1664 sparse matrix of type '<class 'numpy.float64'>'
        with 99693 stored elements in Compressed Sparse Row format>
```

Fitting K-Nearest Neighbours model to the scipy sparse matrix:

```
[27] knn_model = NearestNeighbors(metric='cosine', algorithm='brute')
     knn_model.fit(user_to_movie_sparse_df)
```

```
NearestNeighbors(algorithm='brute', metric='cosine')
```

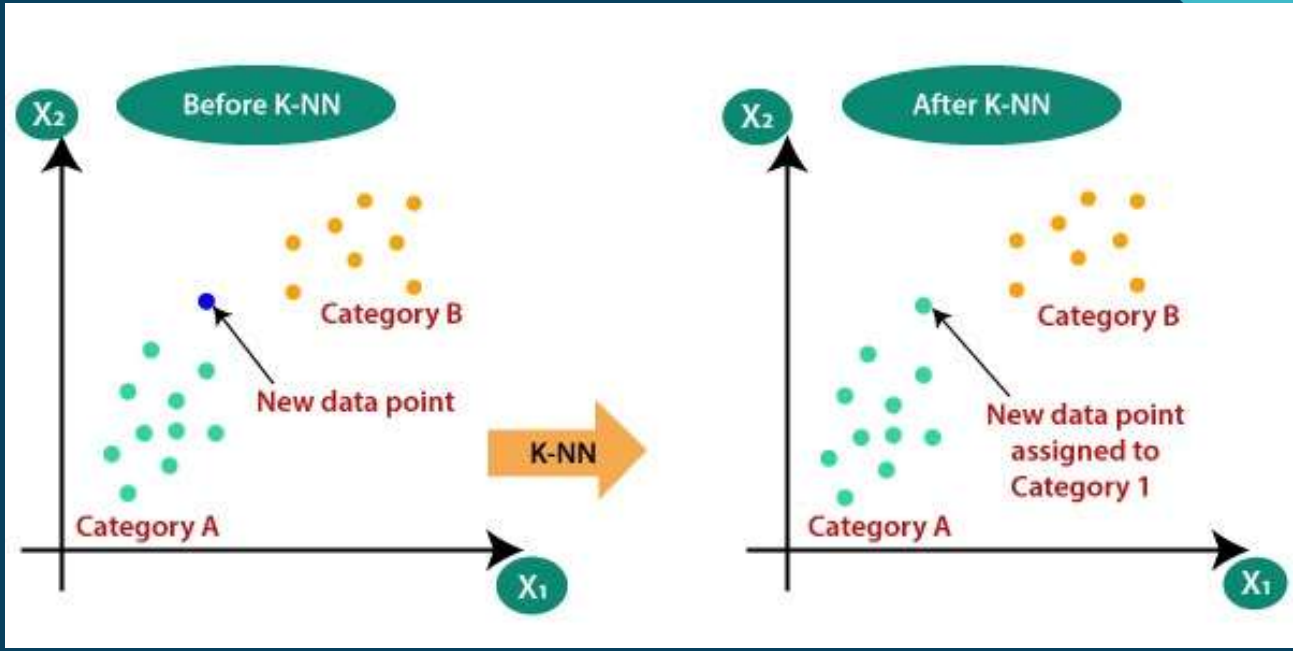
```
## function to find top n similar users of the given input user
def get_similar_users(user, n = 5):
  ## input to this function is the user and number of top similar users you want.

  knn_input = np.asarray([user_to_movie_df.values[user-1]])  #.reshape(1,-1)
  # knn_input = user_to_movie_df.iloc[0,:].values.reshape(1,-1)
  distances, indices = knn_model.kneighbors(knn_input, n_neighbors=n+1)

  print("Top",n,"users who are very much similar to the User-",user, "are: ")
  print(" ")
  for i in range(1,len(distances[0])):
    print(i,". User:", indices[0][i]+1, "separated by distance of",distances[0][i])
  return indices.flatten()[1:] + 1, distances.flatten()[1:]
```

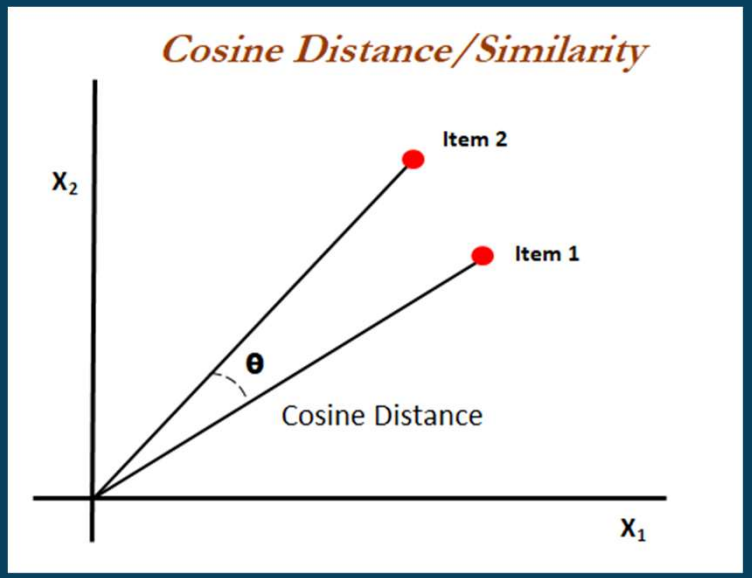
The Algorithm. -> KNN (K Nearest Neighbors)

K Nearest Neighbor is a simple algorithm that maintains all available examples and classifies fresh data or cases based on a similarity metric. It is usually used to classify a data point based on how its neighbors are categorized.



The Mathematics Used. -> Cosine Similarity

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



- **A . B** = product (dot) of the vectors 'A' and 'B'.
- **||A||** and **||B||** = length of the two vectors 'A' and 'B'.
- **||A|| * ||B||** = cross product of the two vectors 'A' and 'B'.

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

Methodology - Part 5, Fine tuning the Model.

The model had a drawback that it recommends movies which are already seen by the given input user.
To mitigate this problem, a function was built to clear out movies which were already rated.

```
def filtered_movie_recommendations(n):

  first_zero_index = np.where(mean_rating_list == 0)[0][-1]
  sortd_index = np.argsort(mean_rating_list)[::-1]
  sortd_index = sortd_index[:list(sortd_index).index(first_zero_index)]
  n = min(len(sortd_index),n)
  movies_watched = list(refined_dataset[refined_dataset['user id'] == user_id]['movie title'])
  filtered_movie_list = list(movies_list[sortd_index])
  count = 0
  final_movie_list = []
  for i in filtered_movie_list:
    if i not in movies_watched:
      count+=1
      final_movie_list.append(i)
    if count == n:
      break
  if count == 0:
    print("There are no movies left which are not seen by the input users and seen by similar users. May be increasing the number of similar users who are to be conside
  else:
    pprint(final_movie_list)
```

Results.

- After all the required changes have been made, we create a function which contains all the other function and call it to make the recommendations.
- Following is the final output of our project.

```
spell_correction()
```

```
Enter the Movie name: Mission: Impossible (1996)
Enter Number of movie recommendations needed: 5
Top 5 movies which are very much similar to the Movie- Mission: Impossible (1996) are:

Independence Day (ID4) (1996)
Rock, The (1996)
Twister (1996)
Eraser (1996)
Broken Arrow (1996)
```

Conclusion.

- The Movie Recommender system was successfully implemented.
- The KNN Algorithm worked very well with the Cosine Similarity Metric.
- For more accurate Recommendations, we will need more ratings and less sparse data.
- Huge amount of sparse data can hamper the model generalization.
- To mitigate this problem, a log transform worked the best on the given dataset.

Future of the System.

- When we don't have enough ratings for a movie or when a user's rating for a movie is very high or low, the cosine similarity computation fails. Other approaches, such as modified cosine similarity, can be used to compute similarity as an upgrade on this project.

- Neural Networks and Deep Learning have been popular in a variety of disciplines in recent years, and it appears that they can also be used to solve recommendation system challenges.

References.

[1] Mayur Rahul et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1022 012100," Movie Recommender System using Single Value Decomposition and K means Clustering"

[2] Wu, Ching-Seh & Garg, Deepti & Bhandary, Unnathi. (2018). Movie Recommendation System Using Collaborative Filtering. 11-15. 10.1109/ICSESS.2018.8663822.

[3] https://grouplens.org/datasets/movielens/100k/

THANK YOU!