This is movie recommender system project.

* yt link- https://www.youtube.com/watch?v=1xtrIEwY\_zY&list=PLKnIA16\_RmvY5eP91BGPa0vXUYmIdtfPQ&index=1&ab\_channel=CampusX
* datset - https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

This is an content based recommendation system.

Abstract - Basically creating tags for each movie by using the relevant information available for movies and then from the tags converting them to vectors and in later part calculating the the distance between the vectors and recommending the nearest five vectors as related or recommended movies.

From the link of the dataset we got the metadata of movies in two files. Imported both files and at first merged both files to create single dataframe.

Then deleted the columns/features we didn’t wanted and kept the remaining in this case here columns are ( 'movie\_id','title','overview','genres','keywords','cast','crew' ).

Then checked for some basic things like:

* If any row contains null values
* Drop the rows containing null values as it is very less
* Check for duplicates

Then performed some basic text processing

* Extracted the genres, keywords label, cast names from respective columns as it was dictionary in the dataframe
* Later created tags by adding all the text from each column other than ‘movie\_id’ and ‘title’ and placed that in new column ‘tags’
* Applied the lower function to lower all the text and joined each name /word in columns such as ‘keyword’ , ‘cast’, ‘crew’
* Then applied stemming to the tags such that it should not create problems in further steps

After the text processing applied text vectorizer.

* Here text vectorization is performed in such a way that at first added all the text of tags of every movie and found out the most frequent 5000 words. And using that created an 5000 dimensional vector.
* It works in such way that if for each dimension in a vector (basically a word is a vector) we count the number of same words present in the tag of the movies. Similarly for each dimension performed. And created vector for each movie.

Which means each movies has its own vector. And using the cosine distance not the Euclidean distance calculated distance between each vector with the other vector.

No at last finding out the top 5 recommendation by sorting distance in reverse to get the max similarity for them and picking the the 5 movies as our recommendation.