Recommendation Engine

**Problem Statement: -**

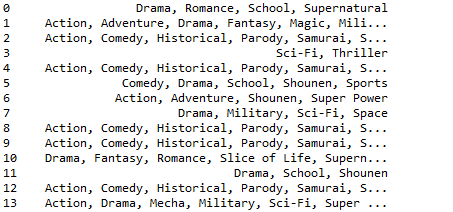
The Entertainment Company, which is a startup online movie watching platform, wants to improvise its collection of movies and showcase those that are highly rated, and recommend those movies to its customer by their movie watching footprints. For this the company has collected its data and shared it with you to provide some analytical insights and also to come up with a Recommendation Algorithm so that it can automate its process for effective recommendations based on Users Interest and behavior patterns.

step1:

Load the Entertainment.csv file to python using pandas. Using head function view the first few rows of the data.

Data contains 51 rows and 4 columns .

The category column is



step2:

checking for na values in game column. There are no na values in game column

step3:

given the reviews of rating between -9 to 9 is to be considered so remove rows which are not of that values. later Tdidf vectoriser is applied to category column

term frequency- inverse document frequency is a numerical statistic that is intended to reflect how important a word is to document in a collection or corpus. So Tfidf vectoriser is used to find important games and stop words are also removed.

from sklearn.feature\_extraction.text, we import TfidfVectorizer

step4:

Preparing the Tfidf matrix by fitting and transforming to Category column of given data.

Transform a count matrix to a normalized tf or tf-idf representation

tfidf\_matrix.shape # (51,34)

step5:

With the above matrix we need to find the similarity score. There are several metrics for this such as the euclidean, the Pearson and the cosine similarity scores. For now we will be using cosine similarity matrix. A numeric quantity to represent the similarity between movies

Cosine similarity - metric is independent of magnitude and easy to calculate

cosine(x,y)= (x.y⊺)/(||x||.||y||)

from sklearn.metrics.pairwise module linear\_kernel is imported for calculating cosine similarities

creating a mapping of entertainment title to index number

thus "Toy Story (1995)" is mapped to 0 index, like all titles are mapped.

step6:

to get recommendations of movies based on their top scores, a get\_recommendations() function is defined which takes movie title and top number as arguments.

with that we can find the score of any movie

