1 ABOUT PROJECT

1.1 TASK

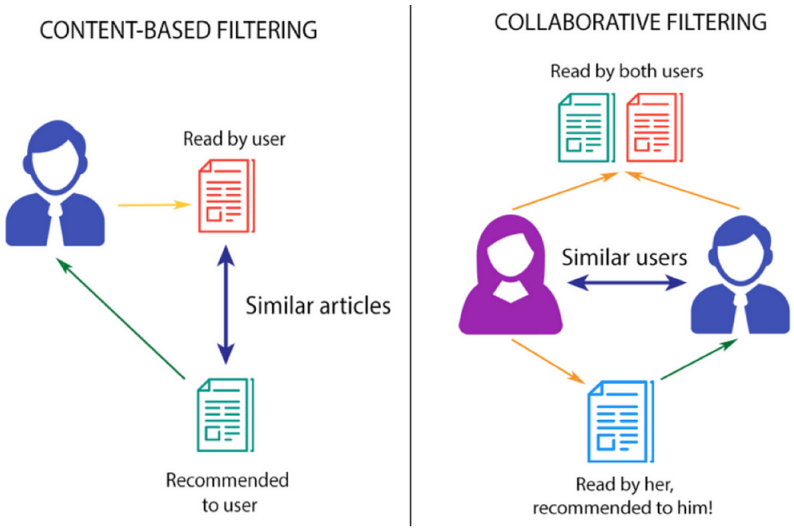
Movie Recommendation System: A recommendation system's purpose is to search for content that would be interesting to an individual. Recommendation systems are AI-based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their profile, search/browsing history, what other people with similar traits/demographics are watching, and how likely you are to watch those movies.

1.2 Datasets

Dataset Link:
`https://www.kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset`

1.3 Thought Process

- The project aims to develop a recommender system for this (MOVIE)
- Broadly classifying there are two types of recommender systems :



- **Content based Filtering :** In this the model tries to recommend the user based on his past queries, viewed movies, movies which he liked, browsing history and the suggest a movie which is similar to those features
- **Collaborative Filtering :** In this we try to find similar unrelated users and and as they are have same interest what ever a user views it will be recommended to the other user
- **Hybrid :** Content based Filtering + Collaborative Filtering
- So its decided that we will be using a hybrid model based on given task

1.4 Project Flow

In the dataset we have 4 csv files tags, ratings, movies and links

1. We will first visualize the data then pre-process it
2. Then train any selected model using this data
3. We are thinking of deploying this on a website
4. Finally test it !!!

2 Data Visualisation & Pre-Processing

2.1 CSV FILES

2.1.1 movies

movieId	title	genres
26871	My Father the Hero (1994)	Comedy|Romance
3190	Supernova (2000)	Adventure|Sci-Fi|Thriller
138208	The Walk (2015)	Adventure|Drama|Thriller
2730	Barry Lyndon (1975)	Drama|Romance|War
49647	Charlotte's Web (2006)	Children|Comedy|Drama|Fantasy

- we see that there are three features or columns in movies movieId, title and genres
- **movieId :** according to the description mentioned : Only movies with at least one rating or tag are included in the dataset. These movie ids are consistent with those used on the MovieLens web site (e.g., id ‘1‘ corresponds to the `<https://movielens.org/movies/1>`). Movie ids are consistent between ‘ratings.csv‘, ‘tags.csv‘, ‘movies.csv‘, and ‘links.csv‘ (i.e., the same id refers to the same movie across these four data files).
- So corresponding to each ID we have a url which directs us to a page where info about that movie is present
- So there are a total of 9742 movies with ID’s ranging from (1 - 193609)
- **Title :** It includes the name of the movie along with the year of airing in ()
- **genres :** It has multiple genres (pipe-separated list)

- According to documentation the available genre were Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, (no genres listed)
- I would like to split the column title into two columns title and year
- Also for easy working I have converted the pipe-separated to a list
- Finally there is no missing data

movieId	genres	Title	Year
1016	[Children, Comedy]	Shaggy Dog, The	1959
8447	[Sci-Fi]	This Island Earth	1955
62511	[Comedy, Drama]	Synecdoche, New York	2008
3769	[Action]	Thunderbolt and Lightfoot	1974
96471	[Crime, Drama, Mystery]	Prime Suspect 3	1993

2.1.2 tags

userId	movieId	tag	timestamp
567	170945	Suspenseful	1525286501
474	3536	priest	1137181376
477	1089	neo-noir	1242494879
62	184471	video game adaptation	1528024898
573	2116	classic	1186588944

- we see that there are four features or columns in tags userId, movieId, tag and timestamp
- **userID :** according to the description MovieLens users were selected at random for inclusion. Their ids have been anonymized. User ids are consistent between 'ratings.csv' and 'tags.csv' (i.e., the same id refers to the same user across the two files).
- **tag :** according to the description (Comment) Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is determined by each user.
- **timestamp :** according to the description Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.
- The timestamp column seems redundant so we can just drop it
- Finally there is no missing data

2.1.3 ratings

userId	movieId	rating	timestamp
479	842	2	1039362294
97	3717	4	1043382867
414	529	4	961517293
68	4681	3	1269123495
603	1201	5	963177205

- we see that there are four features or columns in ratings userId, movieId, rating and timestamp
- **rating :** according to the description Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).
- timestamp column seems redundant so we can just drop it
- Finally there is no missing data

2.1.4 links

movieId	imdbId	tmdbId
172637	219263	74535
127180	2044056	125336
82041	1316536	46420
4613	97637	10345
82848	11541	23282

- we see that there are four features or columns in links userId, movieId, imdbId and tmdbId
- **imdbId :** according to the description imdbId is an identifier for movies used by `<http://www.imdb.com>`. E.g., the movie Toy Story has the link `<http://www.imdb.com/title/tt0114709/>`.
- **tmdbId :** according to the description tmdbId is an identifier for movies used by `<https://www.themoviedb.org>`. E.g., the movie Toy Story has the link `<https://www.themoviedb.org/movie/862>`.
- Finally there is no missing data

2.2 WEB SCRAPING

Although we had information about a particular movie like it's title, year of airing, the ratings and tags were not given for all movies and so I decided to fetch more data like Director, Top crew which must include hero and heroine and an overview or summary which has useful keywords

2.2.1 METHOD 1 : WIKIPEDIA

- I first tried to scrape data from WIKIPEDIA using request beautiful-Soup . Although the request was granted the problem occured when the movie title was ambiguous.
- For example movie with title heat was giving the idea of physics energy form. Hence this method was a failure

2.2.2 METHOD 2 : tmdbId

- Then I tried to scrape data from tmdb website via the id mentioned in the links
- Although here request was accepted the source page was outdated or restricted its information I was not able to get anything useful. Hence method was a failure.

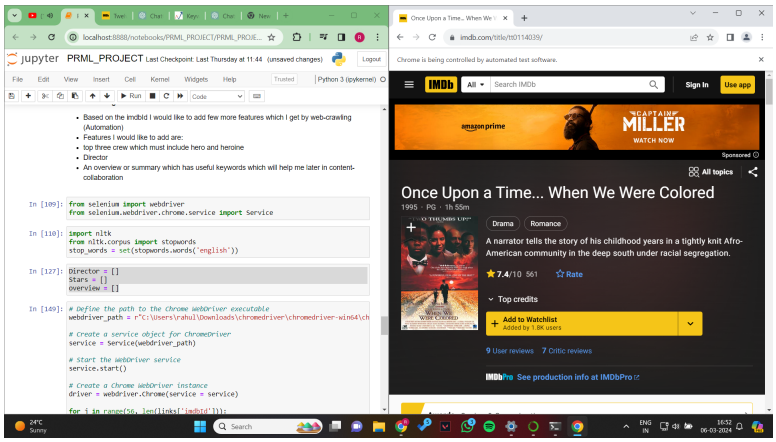
2.2.3 METHOD 2 : tmdbId

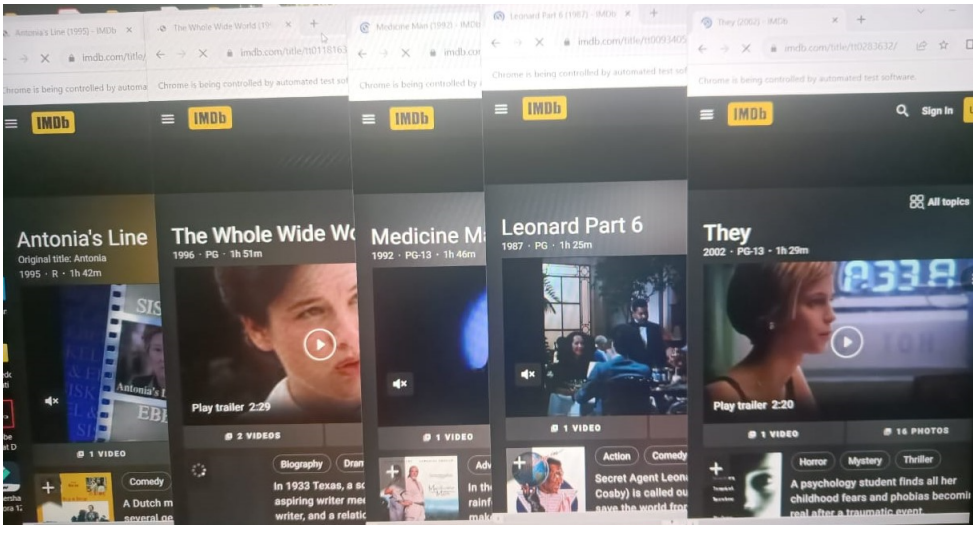
- Then I tried to scrape data from imdb website via the id mentioned in the links
- Here the request itself was forbidden hence another failure

2.2.4 METHOD 3 : SELENIUM

- Then I researched about other possible tools which could be used for scraping and came across selenium web driver and gave it a shot on imdb and was a success

- How did I scrape information about 10,000 movies ?
- Although selenium was working but it is relatively way slower than BeautifulSoup because it gets all the data on source page and then scrapes it accordingly
- So I initiated 10 web-drivers simultaneously and gave each one a 1000 movies each and left the process to run all night. It was not an easy task and took its toll on laptop to the extent its internal fan was running at high speed till it ended.





2.3 stopwords

- Stop words are a set of commonly used words in a language. Examples of stop words in English are “a,” “the,” “is,” “are,” etc. Stop words is commonly used in Natural Language Processing (NLP) to eliminate words that are so widely used that they carry very little useful information.
- During my scraping of overview/summary I have made sure to remove stopwords from it

2.4 Merging

- For content based we will only be requiring movies and links along with the scraped data
- So first add three new columns Director, Stars and Keywords in links dataframe
- Later just concatenate links and movies but drop one of the moviesId as they are arranged in same order and make a new final df dataframe
- Finally converting it into a CSV file it looks like below