Movies Recommendation System

There are three types of Recommendation system are as follow:

1. **Content Based Recommendation System**: It is a type of recommendation system that suggests items to users based on the features or attributes of the items and the user's past interactions with similar items. The primary goal is to recommend items that share characteristics with what the user has previously liked, interacted with, or shown interest in.
2. **Popularity Based Recommendation System**: It is a simple type of recommendation system that suggests items to users based on the overall popularity of those items among all users, without considering individual user preferences or behaviour. The main idea is to recommend items that are widely liked or used by the majority of users.
3. **Collaborative Based Recommendation System**: It is a type of recommendation system that makes suggestions to users based on the preferences and behaviours of other users. The core idea is that users who have had similar preferences in the past will likely enjoy similar items in the future.

**Steps of Projects:**

1. **Collecting the Dataset through this link :** [**https://drive.google.com/file/d/1cCkwiVv4mgfl20ntgY3n4yApcWqqZQe6/view**](https://drive.google.com/file/d/1cCkwiVv4mgfl20ntgY3n4yApcWqqZQe6/view)
2. **Preprocessing the Data :**

I. **Feature Engineering**: s

We eliminate the “Original Language” feature because the majority language of movies in English so we can remove that feature.

I can also remove these columns ['budget', 'homepage', 'original title', 'popularity', 'production companies', 'production countries', 'release date', 'revenue', 'runtime', 'spoken languages', 'status',

'tagline', 'vote average', 'vote count', 'director']

We Eliminate these features because it doesn’t contribute in model accuracy it increases the model complexity so we can Eliminate.

II. **Replacing Missing Values**:

In datasets there are total 4809 rows in which non null values are :

id 4809 non-null

title 4809 non-null

overview 4806 non-null

genres 4809 non-null

keywords 4809 non-null

crew 4809 non-null

cast 4809 non-null

Drop Null values Because the amount of null values are very less.

III. **Duplicate values**:

In dataset there are no duplicate values.

IV. **Extract the data from genres, keyword, cast and crew:**

In genres columns extracting movies categories (for E.g. Action, Romance, etc.).

In keyword column extracting keyword (for E.g. ”Bollywood”, “fall in love”, etc.) of movies.

In cast column extracting main role which play in movie (top 3 cast name included remaining are excluding).

In crew column extracting the Director name and rename the ‘crew’ column to ‘director’ column.

V. Convert datatype of “overview” column (str to list) because we need all columns in list datatype so we can easily concatenate.

VI. Remove the space from ‘genres', 'keywords', 'cast', 'director' to clear the confusion. for e.g. in cast column there is a name (Sam Worthington) and in crew column there is a name (Sam Mendes) so it may be confused in our model. To overcome this situation, we remove the spaces.

VIII. Create a new column (tags) in which concatenation of 'overview', 'genres', 'keywords', 'cast' 'director' columns. It will be help in our model for recommended.

IX. Create a new Dataframe(“new\_df”) in which store the columns 'id', 'title', 'tags' from movies Dataframe. Because I need only these columns to create our model remaining columns are unnecessary.

X. Convert the datatype of tags column (list to str) after converting the datatype we can easily passing in counter vectorization.

XI. In tags column some words are lowercase character and some words are uppercase character. so, we are going to converts all the words in lowercase character form.

Now Data Preprocessing is done.

Now we import NLTK library

* What is NLTK library?
* it's a natural language processing (NLP) technique used to reduce words to their root form. For example, the words **"running"**, **"runner"**, and **"ran"** can be reduced to the root word **"run"**.
* So in tags column similar kind of words are present and these words can generate different - different tokens. To overcome this situation we use nltk library.
* Why we use Stemming instead of Lemmatization?
* **Stemming**: Fast, but less accurate.
* **Lemmatization**: Converts words to their base dictionary form. More accurate, but slower.

Now we have 4806 rows of tag column and in this column there is a big paragraph of textual data and we cannot use textual data in our model so we are converting to counter vectorizer.

What is counter vectorizer ?

**Count Vectorizer** (short for **CountVectorizer**) is a text preprocessing technique used to convert a collection of text documents into numerical feature vectors.

It is part of the **Bag of Words (BoW)** model and is available in libraries like **Scikit-learn** (sklearn.feature\_extraction.text.CountVectorizer) in Python.

**🔍 What does it do?**

* **Builds a vocabulary** of known words.
* **Tokenizes** the text (splits into words).
* **Counts** how many times each word appears in each document.

**📊 Example:**

Let's say you have the following texts:

texts = ["I love machine learning", "Machine learning is fun"]

Using CountVectorizer, it will:

1. Build a vocabulary: ['fun', 'is', 'learning', 'love', 'machine']
2. Create a matrix:

| **fun is** | **learning** | **love** | **machine** |
| --- | --- | --- | --- |
| **0 0** | **1** | **1** | **1** |
| **1 1** | **1** | **0** | **1** |

After Creating the vector of each unique word of every row. We are calculating the Cosine Similarity instead of Euclidean distance because sometimes Euclidean distance has failed to calculate in higher dimension.

For more details about CountVectorizer u can check out at: [Scikit-learn CountVectorizer in NLP - Studytonight](https://www.studytonight.com/post/scikitlearn-countvectorizer-in-nlp)

What is Cosine Similarity?

**Cosine similarity** is a measure of similarity between two non-zero vectors. It calculates the **cosine of the angle** between them — the closer the angle is to 0°, the more similar the vectors are.

**📊 Output Range:**

* **1** → Vectors are exactly the same (0° between them).
* **0** → Vectors are orthogonal (90° apart, no similarity).
* **-1** → Vectors are completely opposite (180° apart).

In text analysis, cosine similarity usually ranges from **0 to 1** because word frequency vectors are non-negative.

**🧠 In Simple Terms:**

It measures **how similar two documents or sentences are**, based on their vector representations.

For more details u can check out at : [Understanding Cosine Similarity in Python with Scikit-Learn](https://memgraph.com/blog/cosine-similarity-python-scikit-learn)

Now we create a **recommend() function** which can take movie name as an argument and in that function we use a function which is sorted function. That **sort function** will sort the cosine similarity. First of all, we find the index of movie and this index of data are going to be sorted and we also use **enumerate function** because it give an index to all similarity score in tuple form and this index will help to **find out the find similar kind of movie name**.

After doing that we are store the similarity score and new\_df data into a pickle file.

**🔍** Why we are Serializing the data?

In ML, pickle is used to **save trained models and preprocessing steps**, so you can **reuse them without retraining** — saving time and enabling deployment.