
# MOVIE RECOMMENDER SYSTEM

PYTHON

Data Analytics and Machine Learning

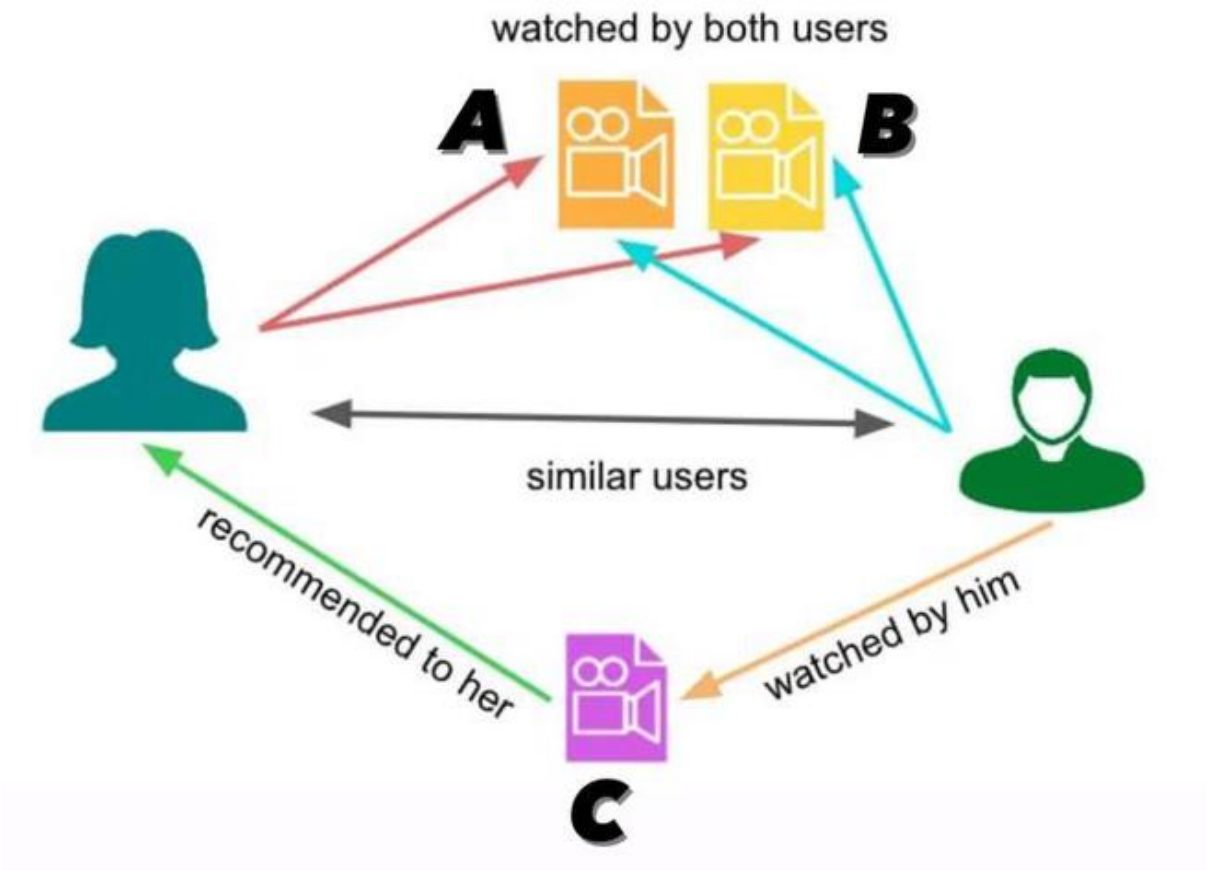
# Project Report

## I. Introduction

What is a recommender system?

A recommender system is an algorithm that aim to provide the most relevant information to user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in movie when you visit the movie in Netflix. You notice that some items are being recommend certain movies to you.

Below is a simple model how movie recommender system work.




Two users watch the same movie name A and B from Netflix. When this happens the similarity index of the two users computed. Depending on the score the system can

recommend movie C to other user because it detects that those two users are similar in term of the item they watch.

# II. Data collection and Data preprocessing

## Data Collection

In order to analyze different machine learning technique. We have collected the dataset movie from Kaggle source name top10K-TMDB-movies. The dataset we collect for all variable have such as id, title, genre, original_language, overview, popularity, release_date, vote_average, and vote_count.

*Dataset top10K-TMDB-movies from Kaggle*

| | id | title | genre | original_language | overview | popularity | release_date | vote_average | vote_count |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 278 | The Shawshank Redemption | Drama,Crime | en | Framed in the 1940s for the double murder of h... | 94.075 | 1994-09-23 | 8.7 | 21862 |
| 1 | 19404 | Dilwale Dulhania Le Jayenge | Comedy,Drama,Romance | hi | Raj is a rich, carefree, happy-go-lucky second... | 25.408 | 1995-10-19 | 8.7 | 3731 |
| 2 | 238 | The Godfather | Drama,Crime | en | Spanning the years 1945 to 1955, a chronicle o... | 90.585 | 1972-03-14 | 8.7 | 16280 |
| 3 | 424 | Schindler's List | Drama,History,War | en | The true story of how businessman Oskar Schind... | 44.761 | 1993-12-15 | 8.6 | 12959 |
| 4 | 240 | The Godfather: Part II | Drama,Crime | en | In the continuing saga of the Corleone crime f... | 57.749 | 1974-12-20 | 8.6 | 9811 |

## Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into a usable format. The dataset contains some missing values. In order to train the model on this dataset, we used the following preprocessing techniques

## Data Cleaning

For the dataset we have check the missing value and we consider to fill it with mean/ median or drop the raw dataset if we can fill it with the mean/ median. Next we check the duplicate value and if we see duplication the data we can drop it directly. Handling the incorrect and inconsistent data (eg: data has capitalization or white space). Check

the variable types it is correct types or not. If we see the error with the types we can convert to correct format. After that identify and correct the outliers has the low or high value that confirm to rest the pattern in data and the technique solve the outlier sorting dataset which is the easy way spot the outlier and it fail for largest dataset, visualization using boxplot and histogram and scatter can detect the outlier dataset.

# III. Methodology of Recommendation System

### Content-Based Filtering:

A recommendation system technique called Content-Based Filtering (CBF) makes suggestions to users based on the characteristics of the objects they have already engaged with. To provide individualized recommendations, the system examines item qualities (genre, keywords, descriptions, or numerical values) and contrasts them with a user's prior selections.

Content-based filtering only considers the individual user and the inherent qualities of the objects, as opposed to collaborative filtering, which depends on user interactions and evaluations from numerous users.

## Agile Methodology:

### 1. Collecting the data set:

Collecting all dataset the required data from Kaggle web site in this project.

### 2. Data Analysis:

Make sure that the collected data set are correct and analyzing the data in the csv file

### 3. Algorithms:

In our project we have two algorithms, one is CountVectorizer and other is Consine similarity are used to build the machine learning recommendation model.

### 4. Training and Testing the model:

Once the implementation of the algorithm is completed. We have to train the model to get the result. We have tested it several times the model is recommend different set of moves to different users.

### 5. Improvements in the project:

In the later we can implement different algorithms and methods for better recommendation.

## IV. IMPLEMENTATION

We are going to use the top10K-TMDB-movies dataset to build a simple content-base filtering recommender system. The first we need to do is import necessary *library such as.*

*import pandas as pd*

*import numpy as nb*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*from wordcloud import WordCloud*

Next, we load in the data set using pandas *read_csv()* with specify the csv data set name. we define the data set name *df*.

*df = pd.read_csv('top10K-TMDB-movies.csv')*

Now let´s check the head of the data to see data we are deal in with

*df.head()*

we check the number of the rows and columns for the data set we work with

*df.shape*

let´s look at what each columns represents:

- Id - the movie ID on the website.
- Title – the movie name.

- Genre – the movie genre.
- Original_Laguage – original language in which the movie is released.
- Overview – summary of the movie.
- Popularity – movie popularity.
- Release_date – movie release date.
- Vote_average – movie vote average.
- Vote_count – movie vote count.

Using the describe or info commands we can get a brief description of our dataset. This is important in order to enable us to understand the dataset we are working with

*df.describe()*

*df.describle(include=[object])*

now we jumping in to data cleaning we check the missing values first and duplication values.

*df.isna().sum()*

*df.duplicated().any()*

we have checked the missing values and duplication values, consider to drop it because if we keep it effect the model and accuracy of the machine learning model. Otherwise, we not fill it because the data set have more enough.

*df = df.dropna()*

*df = df.drop_duplicates()*

we plot the seaborn to see the data set if still to have missing value or not.

*Import seaborn as sns*

*sns.headmap(df.isna())*

we use numerical value such as id, popularity, vote_average and vote_count as defined *df_corr()* with *corr()* and plot correlation matrix to see how variables are related to each other

*df_corr = df[['id','popularity','vote_average','vote_count']]*

*df_corr.corr()*

*corr = df.select_dtypes(include=['float','int']).corr()*

*plt.figure(figsize=(7,7))*

*sns.heatmap(corr, annot=True)*

we use the pair plot to understand the best set of features to explain a relationship between two variables or to form the most separated clusters

*sns.pairplot(data=df, diag_kind="kde", palette="husl");*

we plot the world cloud that is visual tool to displaying the most frequently used words in a text or set of texts in genre represent the types of the movie in data set.

*from wordcloud import WordCloud*

*# Ensure all entries in 'genre' are strings and handle NaNs*

*df['genre'] = df['genre'].astype(str).fillna('')*

*text = ', '.join(df['genre'])*

*wordcloud = WordCloud(width=800, height=400, background_color='black').generate(text)*

*plt.figure(figsize=(10, 5))*

*plt.imshow(wordcloud, interpolation='bilinear')*

*plt.axis('off')  # Hide the axes*

*plt.show()*

And the last point visualization technique we use bar char to see the highest popularity movie in seaborn.

*top10_popularity = df.nlargest(10,'popularity')[['id', 'title', 'genre', 'original_language', 'overview', 'popularity','release_date', 'vote_average', 'vote_count']].set_index('title')*

*top10_popularity*

*sns.barplot(x='popularity', y=top10_popularity.index, data = top10_popularity);*

*plt.legend(bbox_to_anchor=(1.05,1), loc = 2);*

*plt.title("Top 10 of popularity movies");*

*plt.show();*

For data preparation we use variables selection we chosen some variables such as is, title, overview, and genre, and we combined two variable title and overview as new variable called tags.

*df = df[['id','title','overview','genre' ]]*

*df['tag'] = df['overview'] + df['genre']*

*df.head(1)*

and after we get new variable called tags that combined overview, and genre. So we consider to drop those and define the new data frame called df_up.

*df_up = df.drop(columns=['overview', 'genre'])*

*df_up.head(1)*

Now jumping into the algorithms we apply we dataset we use CountVectorizer algorithms that work by transforming a collection of text document into matrix of token count. We use CountVectorizer algorithm with package Scikit-learn in python.

*from sklearn.feature_extraction.text import CountVectorizer*

*cv = CountVectorizer(max_features=9985, stop_words='english')*

*vector = cv.fit_transform(df_up['tag'].values.astype('U')).toarray()*

*x = cv.transform(df_up['tag'])*

After we transformed the text document to matrix next step we have used the Cosine Similarity algorithms to compute the similarity score between the movies vectors.

*from sklearn.metrics.pairwise import cosine_similarity*

*simmilarity = cosine_similarity(vector)*

Next step we already apply CountVectorizer algorithms and Cosine Similarity algorithms, we use ID of the move to access the tags with similarity to generate the new title move with the similar range values. Example we use the index 2 name movie is The Godfather.

```
df_up['title'].iloc[2]
```

```
df_up[df_up['title'] == "The Godfather"]
```

we use for loop to generate the similarity of the index 2 movie name is The Godfather. The loop show the similarity values.

```
dis = sorted(list(enumerate(simmilarity[2])), reverse=True,  key=lambda vector:vector[1])
for i in dis[:10]:
  print(i)
```

we have created the function to generate the movie recommendation with loop the result. It show as name of the movies with 5 similars.

```
def recommander(movies):
  index = df_up[df_up['title'] == movies].index[0]
  distance = sorted(list(enumerate(simmilarity[index])), reverse=True, key=lambda vector:vector[1])
  for i in distance[0:5]:
    print(df_up.iloc[i[0]].title)
recommander('The Lion King')
```

The last model we have work with python pickle to save the machine learning algorithm in disk for later use. Because next last step we will deployment the model in web framework.

```
import pickle
pickle.dump(df_up, open('movie_list.pkl', 'wb'))
pickle.dump(simmilarity,open('simmilarity.pkl', 'wb'))
```

*pickle.load(open('movie_list.pkl','rb'))*

## V. Result

Since our project is movie recommendation system on can develop a move recommendation system by using either content-based filter.

In our project we have deploy the model in web framework by using the streamlit that is open-source python library designed to simplify the process of building interactive web applications for machine learning project.

Screen shot the result:

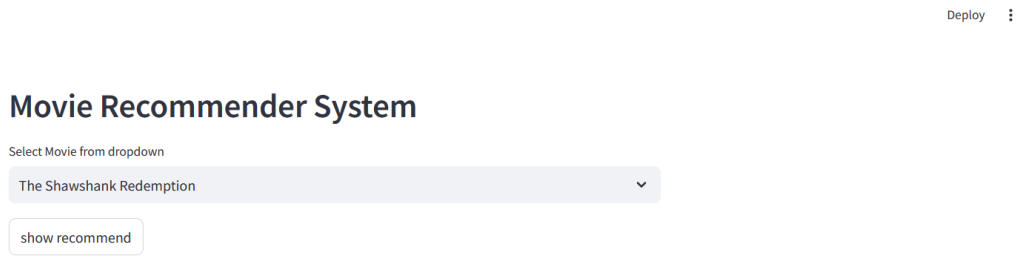



Fig 1: The original web pages

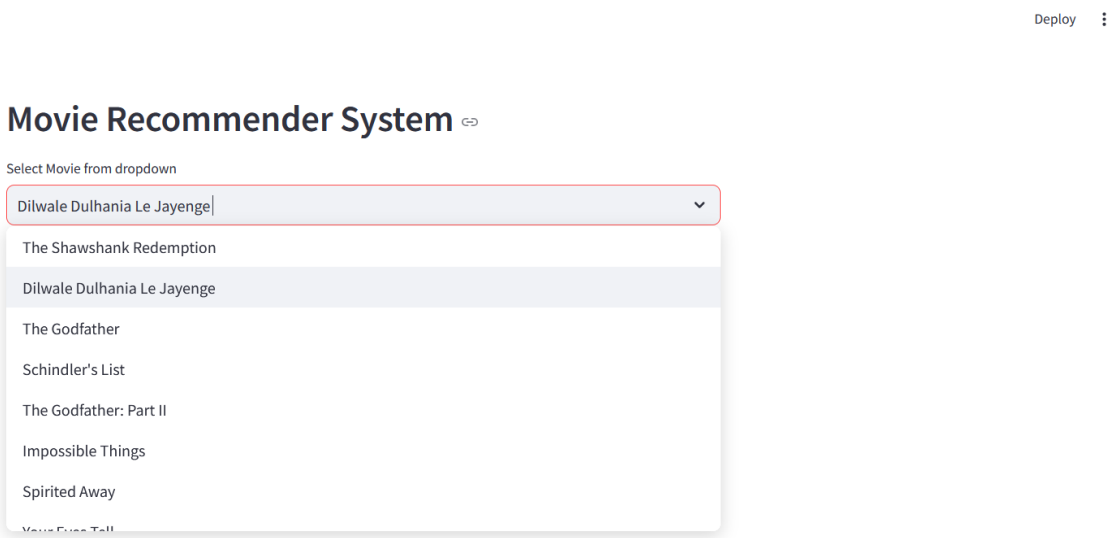


Fig 2: Display of list of recommended movies

User can search the movie or press the name of movie

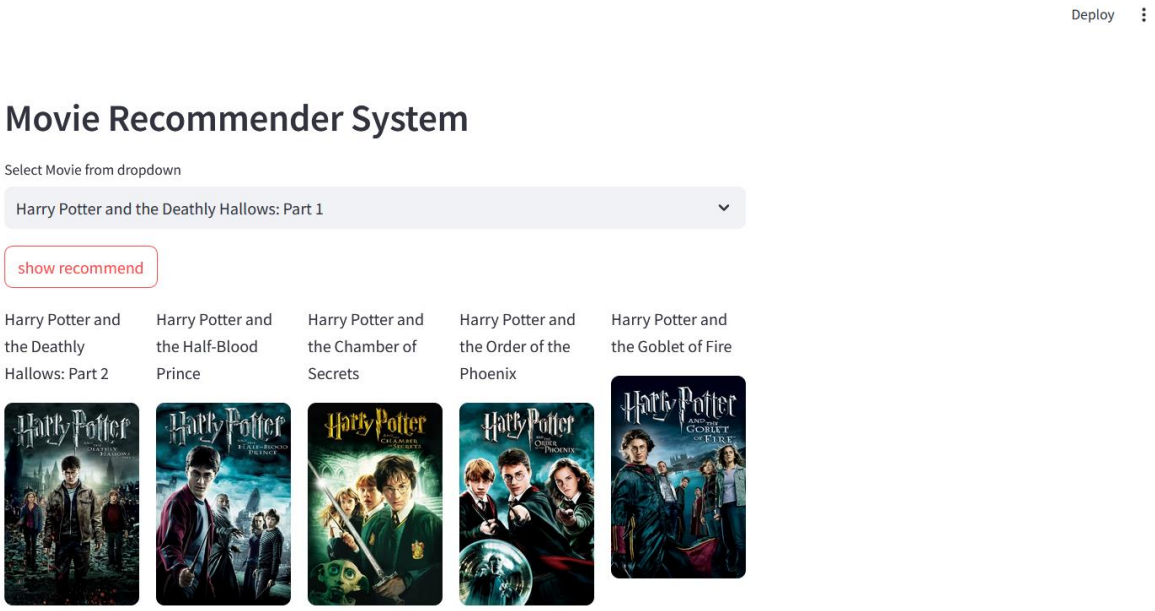



Fig 3: Display the result with name move recommended movie

## VI. Conclusion

In this project, we developed a movie recommendation system utilizing content-based filtering techniques, specifically through the application of machine learning algorithms such as CountVectorizer and cosine similarity. By combining movie titles and overviews into a single feature set, we created a robust basis for generating recommendations based on textual content.

The use of CountVectorizer allowed us to effectively convert text data into a numerical format, enabling the computation of similarity scores between movies. By applying cosine similarity, we were able to measure the degree of similarity between different movie vectors, facilitating the identification of movies that share common characteristics.

The results of our system demonstrate the effectiveness of content-based filtering in providing personalized movie recommendations. Users are likely to discover films aligned with their preferences, enhancing their overall experience on the platform. While this approach has proven successful, it also highlights the potential for further improvements, such as integrating collaborative filtering techniques to incorporate user behavior and ratings, thereby expanding the system's capability to recommend diverse content.

Overall, this project illustrates the power of machine learning in the realm of recommendation systems and sets the stage for future enhancements that could lead to more sophisticated and user-centric solutions.

## VII. References


[1] KNOWLEDGE DOCTOR. (2023, April 2). *NETFLIX Movie Recommender System using Machine learning* [Video]. YouTube. https://www.youtube.com/watch?v=kuC38ZCcbZI

[2] GouravSharmaAmbah. (2022, December 21). *final report.pdf* [Slide show]. SlideShare. https://www.slideshare.net/slideshow/final-reportpdf-254979130/254979130

[3] ASTRO CODER. (2024, July 13). *Movie Recommendation System | Python Machine Learning Project tutorial for Beginners* [Video]. YouTube. https://www.youtube.com/watch?v=i-B_I2DGIAI

[4] GeeksforGeeks. (2024, September 10). *Python | Implementation of Movie Recommender System*. GeeksforGeeks. https://www.geeksforgeeks.org/python-implementation-of-movie-recommender-system/