MOVIE RECOMMENDATION

COMPARING TEXT & MEDIA ADAPTATIONS

MOVIE RECOMMENDATION

A movie recommendation system is a technology that suggests movies to users based on their preferences, viewing history, or behavior. Here's a basic breakdown of how they work:

Movie recommendation systems are pivotal in today's entertainment landscape due to their ability to personalize user experiences by suggesting tailored movie options, facilitating content discovery amidst a sea of choices, and driving user engagement through curated selections. offering a competitive edge in the market, and continually adapting to evolving preferences through advancements in technology, ensuring ongoing relevance and superior service.

IMPORTING LIBRARIES FOR MOVIE RECOMMENDATION SYSTEM

Tkinter

Tkinter is a Python library for creating graphical user interfaces (GUIs). It is used to create the user interface for the movie recommendation system, allowing users to interact with the system and view recommendations.

Sklearn.feature_extraction.text

Sklearn.feature_extraction.text is a Python library for text preprocessing and feature extraction. It is used to extract relevant information from movie reviews and ratings, which is used to train the recommendation system.

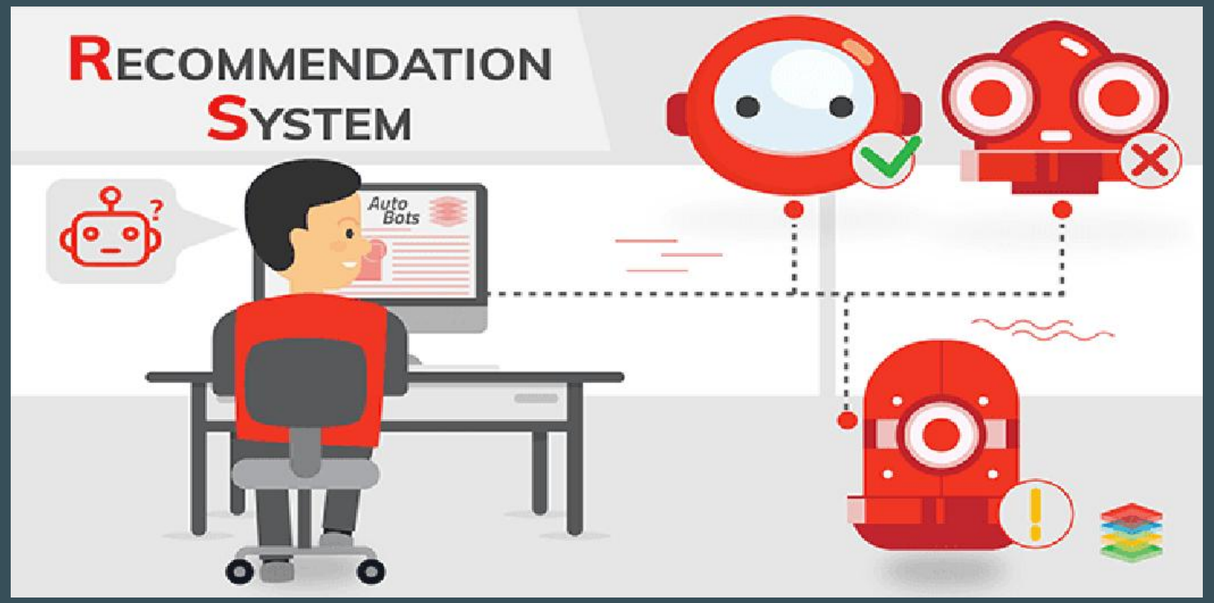
MultinomialNB

MultinomialNB is a Python library for multinomial logistic regression. It is used to train the recommendation system to predict which movies a user is likely to enjoy based on their past viewing history and preferences.

Pandas

Pandas is a Python library for data manipulation and analysis. It is used to load and preprocess the movie review and rating data, as well as to create the recommendation system's training and testing datasets.

DATABASE IN MOVIE RECOMMENDATION



The code reads the movie database from a CSV file using pandas
The below give code lets yoyr acces the data base
data = pd.read_csv('movie_metadata (1).csv')
This code imports the pandas library and reads the CSV file 'movie_dataset.csv' into a pandas DataFrame named 'df'.

Databases are the backbone of movie recommendation systems, storing information about movies, user preferences, ratings, and more. They enable the system to analyze vast amounts of data, understand user behaviors, and generate personalized recommendations. Without databases, recommendation systems would lack the necessary information to make accurate and relevant movie suggestions to users based on their tastes and preferences.

Separate Features and Labels

To perform movie recommendation, the movie data needs to be separated into features and labels. This separation allows us to train a machine learning model to predict movie recommendations based on the given features.

Movie Data

Movie	Genre	Rating	Year	Label
Movie A	Action	4.5	2019	Recommended
Movie B	Comedy	3.8	2018	Not Recommended
Movie C	Drama	4.2	2020	Recommended
Movie D	Horror	3.9	2017	Not Recommended
Movie E	Romance	4.1	2019	Recommended

CONVERT TEXT TO NUMERICAL FEATURES

This code snippet represents a common preprocessing step in machine learning when working with movie data or any text-based information for building a recommendation or prediction model.

```
CountVectorizer
CountVectorizer is used to convert text data
into numerical features.
X = data[['Released_year', 'director_name',
'genres', 'imdb_score']]
y = data['movie_title']
vectorizer = CountVectorizer()
X_transformed =
vectorizer.fit_transform(X.astype(str).apply('
'.join, axis=1))
```

CONVERT TEXT TO NUMERICAL FEATURES

1. Separating Data into Features and Labels:
    - X represents the features, which include attributes such as the released year, director name, genres, and IMDb score.
    - y represents the labels, which in this case, seems to be the movie titles.
2. Using CountVectorizer to Convert Text to Numerical Features:
    - CountVectorizer is a tool often used in natural language processing (NLP) to convert text data into numerical features that machine learning algorithms can understand.
    - The fit_transform() function from CountVectorizer is applied to the combined string representation of the features (Released_year, director_name, genres, imdb_score). It concatenates these columns into a single string for each row.
- This concatenated string is then transformed into a numerical representation using the fit_transform() method. It converts the text data into a matrix where each row corresponds to a movie and each column represents a unique word or combination of words (depending on the parameters used in CountVectorizer).

Train a Naive Bayes Classifier

Naive Bayes classifier

```
model = MultinomialNB()

model.fit(X_transformed, y)
```

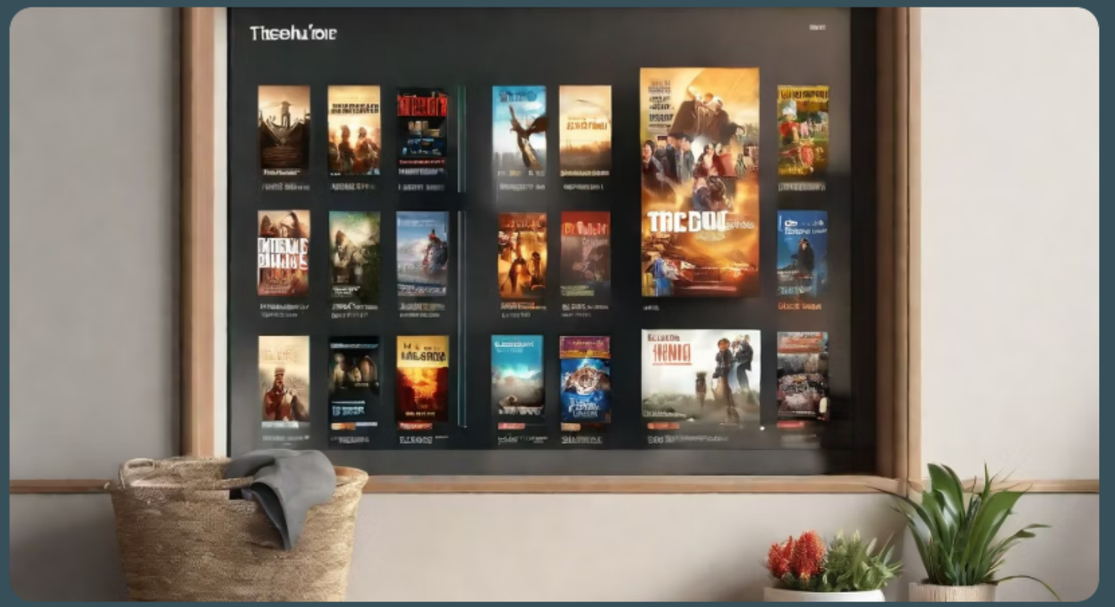
Step 1: Transform Features and Labels

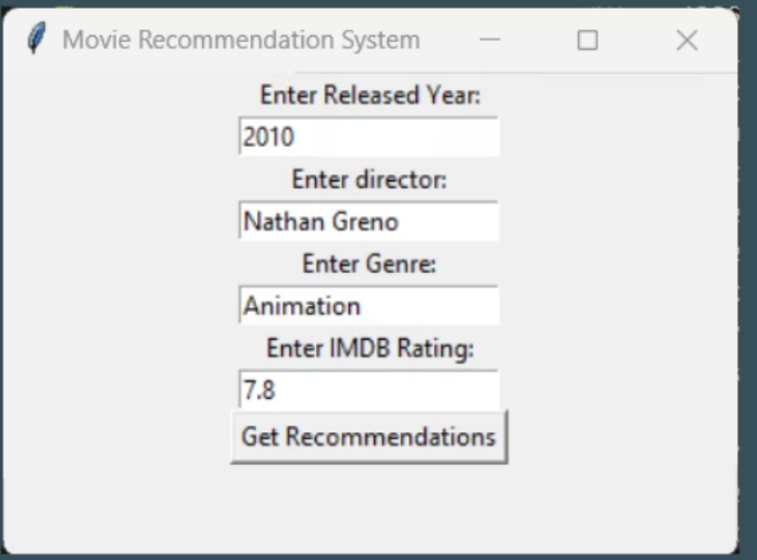
Before training the Naive Bayes classifier, we need to transform the features and labels into a suitable format. This may involve converting categorical variables into numerical representations or normalizing numerical features.

Step 2: Train the Classifier

Once the features and labels are transformed, we can proceed to train the Naive Bayes classifier. This involves fitting the classifier to the training data and learning the underlying probability distribution of the features given the labels.

Create a Tkinter Window





Tkinter Window for Movie Recommendation System

To create a Tkinter window for the movie recommendation system, follow these steps:

1. Import the Tkinter module.
2. Create a new Tkinter window object.
3. Add widgets to the window, such as labels, buttons, and entry fields.
4. Define functions to handle user interactions, such as button clicks.

Entry Fields for Movie Features

IMDB Rating

It is a rating given to a movie by common people, critics, and movie lovers

Genre

Select the genre of the movie from a drop-down list.

Director

Enter the name of the movie director.

Release Year

Enter the year the movie was released.

Recommend Movies Function

Function Definition

The recommend_movies function takes user preferences as input and returns a list of recommended movies.

Input Parameters

- user_preferences: A dictionary containing user preferences such as genre, rating, and release year.

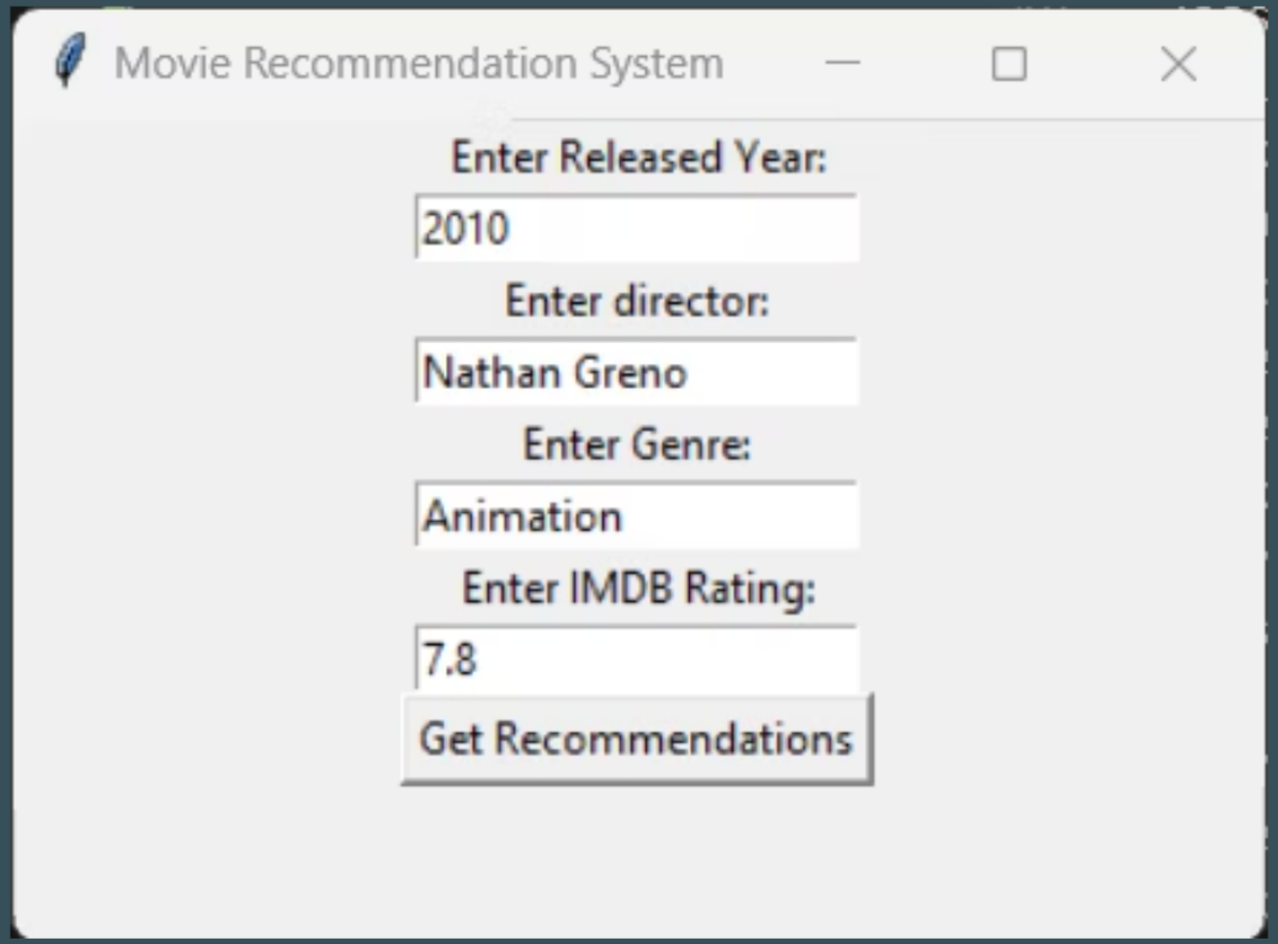
Output

The function returns a list of recommended movies based on the user preferences.

Algorithm

The function uses a recommendation algorithm that analyzes the user preferences and compares them to the characteristics of available movies.

OUTPUT


Movie Recommendation System
Enter Released Year:
2010
Enter director:
Nathan Greno
Enter Genre:
Animation
Enter IMDB Rating:
7.8
Get Recommendations

OUTPUT