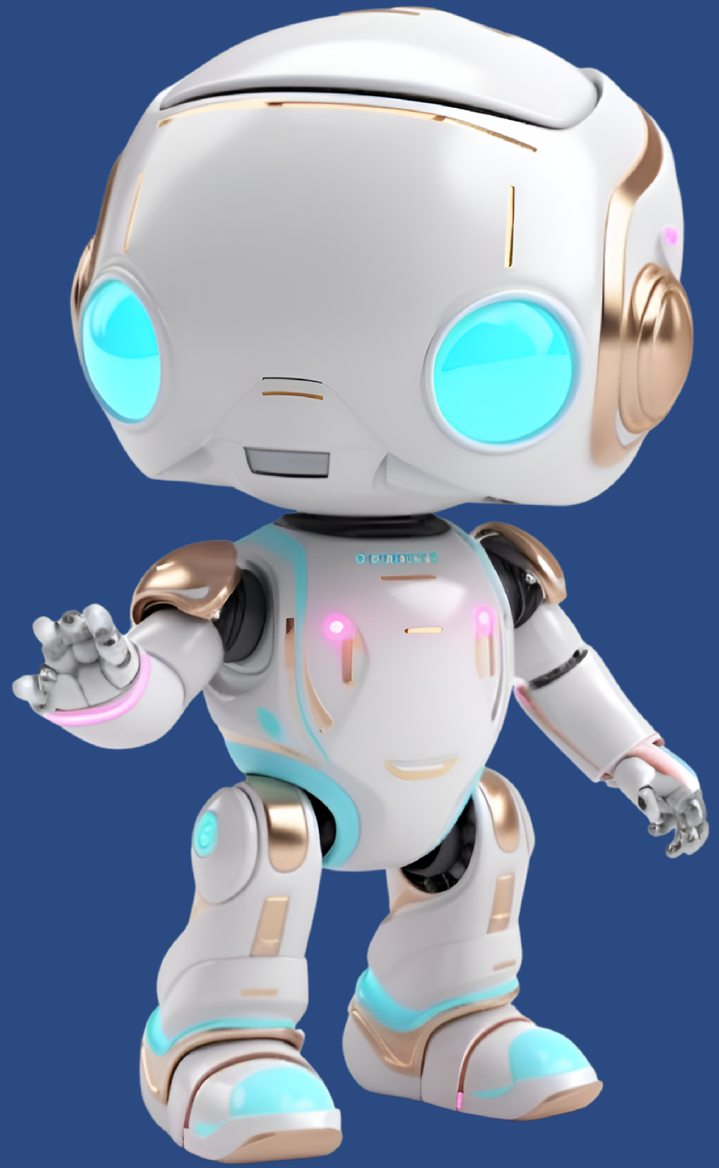
MOVIE RECOMMENDATION SYSTEM USING MACHINE LEARNING

V.S.PAVITHRA-R180258
M.SOUMYA-R180919

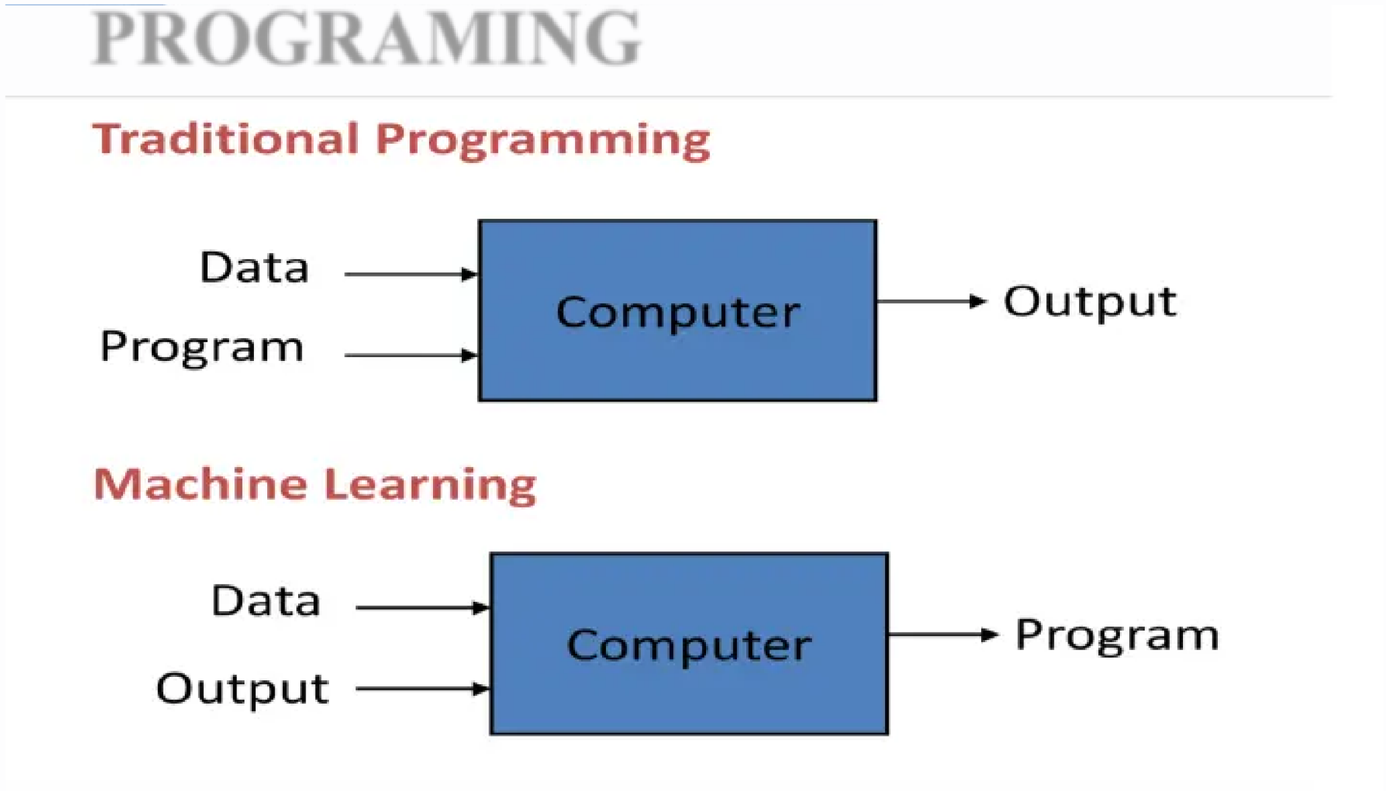
OUTLINE

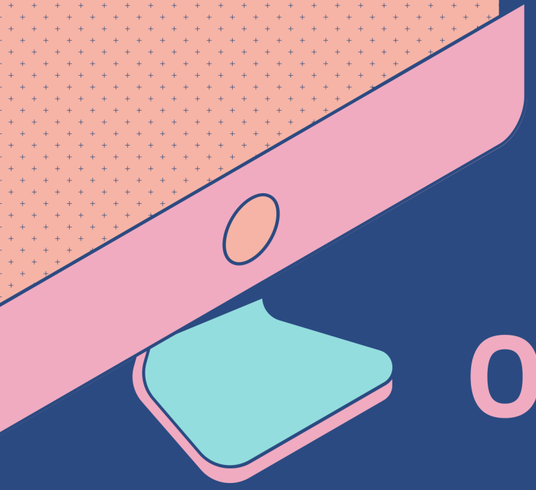
*INTRODUCTION
*OVERVIEW OF RECOMMENDATION SYSTEM
*REASON FOR SELECTING A TOPIC
*OBJECTIVE
*MOTIVATION
*PROJECT OVERVIEW
*USED TECHNOLOGIES
*DATA PREPROCESSING
*COUNT VECTORIZATION
*COSINE SIMILARITY
*CONTENT-BASED FILTERING
*RESULTS
*FUTURE WORK
*SNAPSHOTS
*CONCLUSION

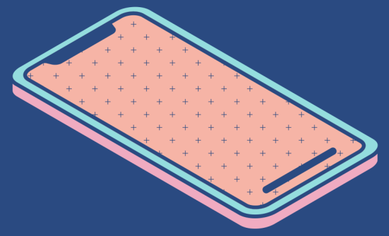


INTRODUCTION

- Welcome to our project presentation on "Movie Recommendation system using Machine Learning," where technology meets entertainment
- Machine Learning is the Art and Science of training computer without being explicitly programmed.

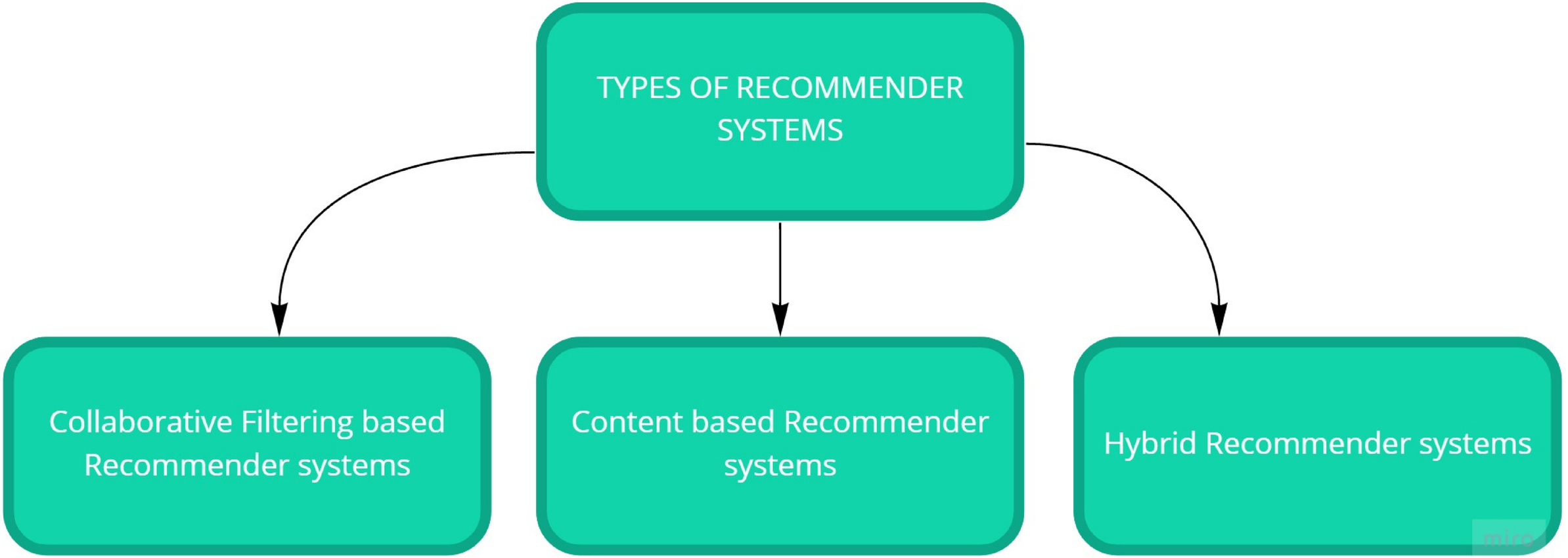




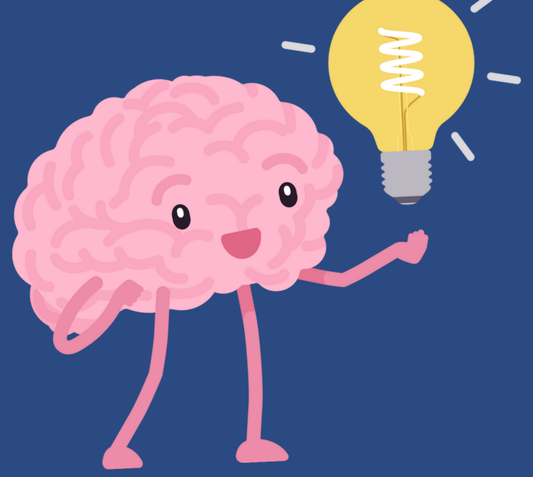


Overview of Recommendation System

- We tend to like things that are similar to other things,we tend to like things that similar people like .
- These Patterns can be used to make predictions to offer new things
- Recommendation Systems are a type of information filtering system hat enhances the quality of search results by showing items that are more relevant to the search item or are related to the users past searches.
- Recommendation systems have become very popular with the increasing availability of millions of product online
- Recommender systems are essential for web-based companies that offer a large selection of products. Amazon , Instagram, and Netflix all use recommender systems to help their online customers make sense of the large volume of individual items – books, films, electronics, whatever – found in their content catalogues.


TYPES OF RECOMMENDER SYSTEMS
Collaborative Filtering based Recommender systems
Content based Recommender systems
Hybrid Recommender systems

- Collaborative Filtering
  - Collaborative Filtering system maintains a database of many user’s ratings of a variety of items.
  - Makes use of the user data,ignoring content/item data.
- Content-based Filtering
  - Content-based filtering relies on assigning attributes to database objects so the algorithm knows something about each object.
  - These attributes depend primarily on the products, services, or content you're recommending.
  - Assigning attributes can be a monumental undertaking.
- Hybrid Filtering
  - Hybrid Recommender Systems are a combination of two or more recommender systems that work together to provide more accurate and diverse recommendations.



Reason for selecting the topic

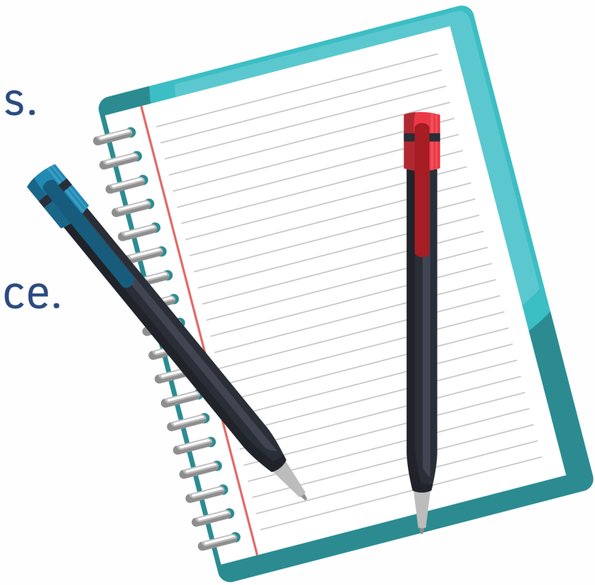
The majority of people watch movies in today's culture,but they are only allowed to watch one before they feel confused about what to watch next.What if there was a system that could comprehend you and provide recommendations for you based on your interests? Recommendation systems are there to help with it.

Customers frequently check at the product recommendations from their most recent browsing.Customer satisfaction is the most crucial factor,and the recommendation system has been helping with that for years.

User specific recommendations are provided by recommender systems,which also assist users in making informed choices during online transactions.Sales are increased,the web surfing experience is changed.Customers are retained,and the shopping experience is improved.

OBJECTIVE

- To create a personalized and efficient movie recommendation system for users based on their preferences and content similarities
- Utilize content-based filtering to recommend movies tailored to individual user tastes.
- Implement count vectorization and cosine similarity to analyze movie descriptions and calculate similarity scores.
- Showcase the capability of the system to provide accurate and relevant movie suggestions.
- Enhance the user experience by offering a seamless and intuitive recommendation interface.
- Improved user engagement and satisfaction through relevant content recommendations.



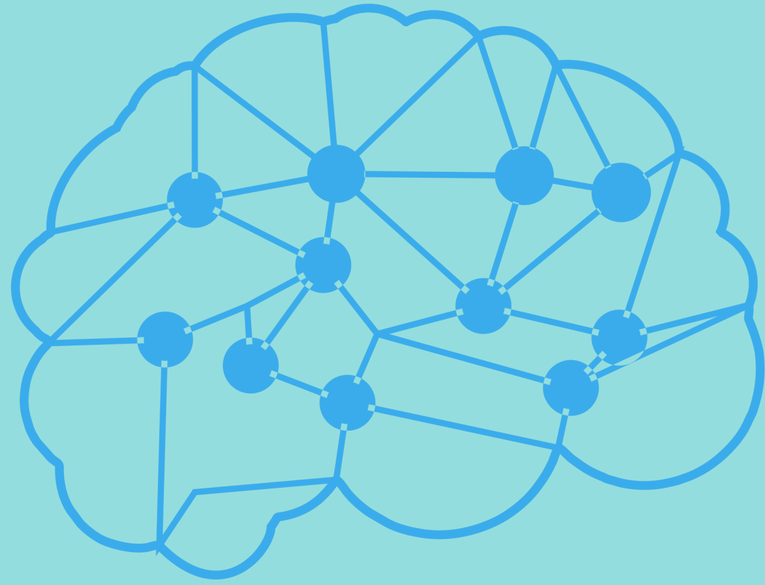
Motivation

Netflix and Spotify ,for example ,rely heavily on the effectiveness of their recommendation engines to drive their growth and success.I feel that recommendation systems would have a significant influence on application access and usage,and it is critical that the user maintains interest in the application.We need to display people trending information or stuff that hey could enjoy in order to keep them interested in our app.This is precisely why i set out to create this method. The decision to adopt content-based filtering for our movie recommendation system is,Content-based filtering offers transparency in recommendation generation by explicitly relying on features of items and user preferences.Users can understand why a particular movie is recommended to them, as it is based on the content similarities between their preferred movies and the recommended one.content-based methods are less dependent on extensive user interaction data .This makes content-based filtering particularly useful in scenarios where user-item interaction data may be sparse or difficult to obtain.

PROJECT OVERVIEW

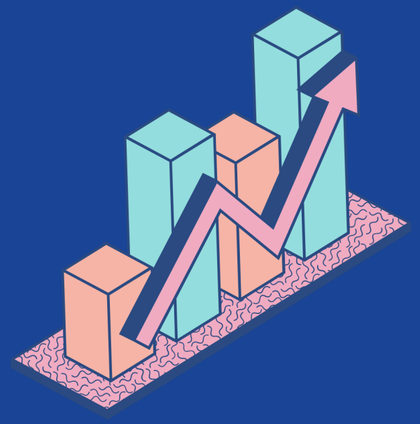
The purpose of this project is to develop a movie recommendation system using Python and machine learning algorithms, specifically cosine similarity. The system aims to provide personalized movie recommendations to users based on their preferences. The cosine similarity algorithm will be used to measure the similarity between movies and users, allowing for effective recommendation generation. The project involves data collection and pre-processing, where a dataset of movie ratings and user information will be gathered. Feature extraction techniques will then be applied to extract relevant information from the dataset, such as genre, director, and actors. The cosine similarity algorithm will be implemented to compute the similarity scores between movies and users based on their shared features.Results and analysis will be presented to showcase the effectiveness of the system in providing accurate and personalized recommendation.The implementation of this system has the potential to enhance the movie watching experience and facilitate movie discovery for users In conclusion, this project aims to develop a movie recommendation system using Python and the cosine similarity algorithm, providing users with personalized movie suggestions based on their preferences.

Used Technologies:



- Programming Languages
    - Python
- Machine Learning Libraries
    - Scikit-learn
    - Pandas
    - Numpys
- Development Environment
    - Jupyter Notebook
- MachineLearning Model
    - Cosine-Similarity

Data Preprocessing



Dataset Used:

A Kaggle dataset which was scraped from wikipedia and contains plot summary of movies.Here i use two datasets one is movies and another is credits.The movies dataset contains 5000 movies wit feature like title,director,genres,languagee etc...And the credits dataset contains cast,crew features.

Numpy & Pandas

- We employed the Pandas library to load the dataset into a DataFrame, allowing for easy exploration and manipulation of the data.

Handling Missing values

- Checked for missing values in each column to assess the completeness of the dataset.
- Evaluated the impact of missing values on the overall dataset and applied appropriate imputation techniques.

Removing Duplicates

- Used Pandas to identify and remove duplicate entries based on unique identifiers such as movie titles.
- Eliminating duplicates ensures data integrity and prevents bias in subsequent analyses.

Feature Extraction

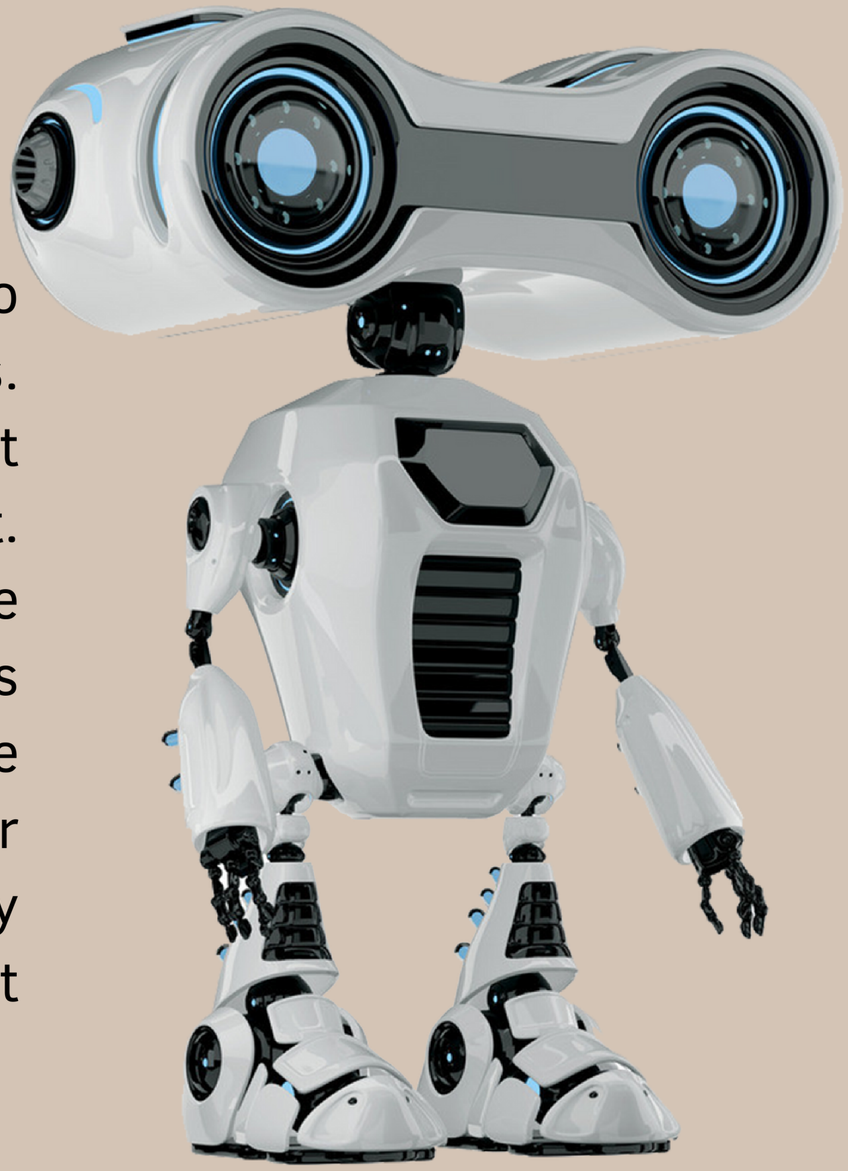
- Extracted limited elements from features containing lists using custom Python functions.
- Utilized Python functions to efficiently extract relevant elements from list-based features.
- Reducing the dimensionality of list features to a manageable number of elements for improved computational efficiency.

Text Data Processing

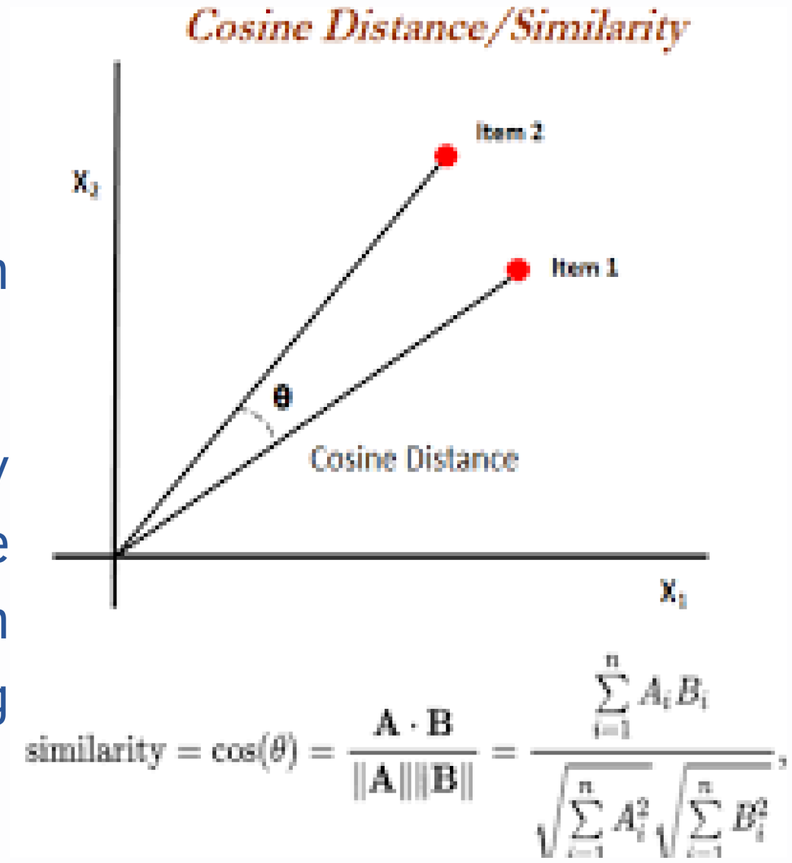
- Preprocessed and cleaned movie descriptions using Pandas and custom Python functions, including removing special characters, converting to lowercase, and handling stopwords.
- Employed tokenization techniques to break down descriptions into individual words.
- The dataset is now cleaned, free of duplicates, and ready for further analysis using content-based filtering techniques.

Count Vectorization

Count Vectorization is a text preprocessing technique used to convert a collection of text documents into numerical vectors. It represents each document as a vector, where each element corresponds to the count of a specific word in the document. This process involves building a vocabulary from the entire corpus of documents and counting the occurrences of words for each document. The result is a matrix known as the Document-Term Matrix (DTM), which serves as input for machine learning models. Count Vectorization is commonly used in natural language processing tasks such as text classification, sentiment analysis, and information retrieval.



Cosine Similarity



Cosine Similarity is a metric used to measure the similarity between two vectors by calculating the cosine of the angelbetween them.

In our movie recommendation system, we employ Cosine Similarity to quantify the similarity between movies based on their feature vectors. These feature vectors are derived from count vectorization of movie descriptions.Enhances the user experience by suggesting movies that align closely with the user's preferences.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

How It Works:

- Each movie is represented as a vector in a high-dimensional space, where each dimension corresponds to a unique word.
- Cosine Similarity calculates the cosine of the angle between two movie vectors, resulting in a score between -1 (completely dissimilar) and 1 (completely similar).
- A higher cosine similarity score indicates a greater similarity between two movies.

Content-Based Filtering

Content-Based Filtering is a recommendation system technique that relies on the inherent characteristics of items and the preferences expressed by a user. In the context of our movie recommendation system, content-based filtering involves analyzing and leveraging features associated with each movie to provide personalized recommendations. This approach is rooted in the idea that if a user has shown interest in certain movies, they are likely to enjoy movies with similar attributes.

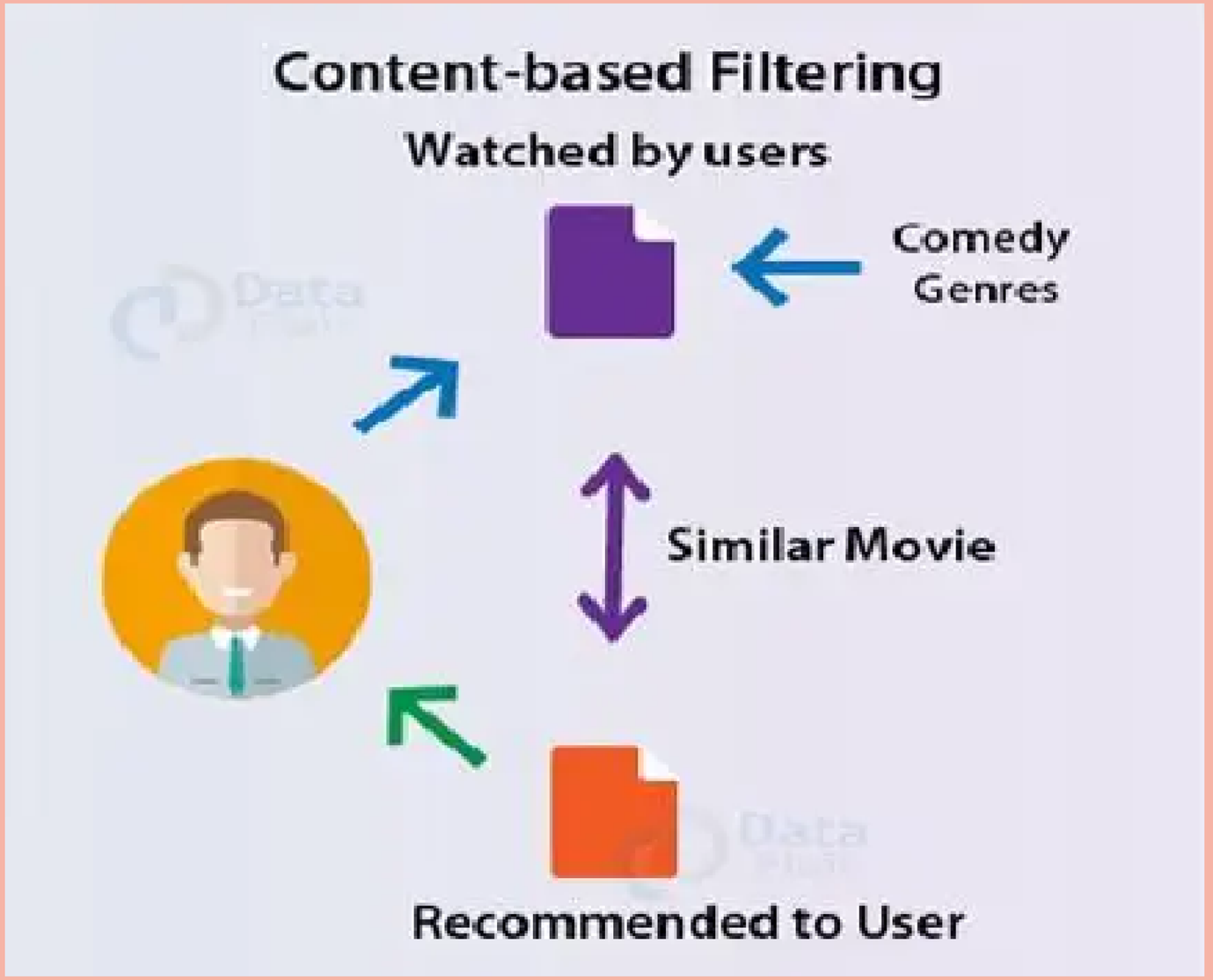
The process begins with the extraction and representation of relevant features from the movie dataset. In our case, we utilize count vectorization to convert textual information, such as movie descriptions, into a numerical format. This numerical representation allows us to quantify the content of each movie

Content-Based Filtering

Cosine Similarity, a key component of content-based filtering, is then applied to measure the similarity between the feature vectors of different movies. This similarity score helps identify movies that share common attributes or themes. A higher similarity score indicates a stronger resemblance between two movies.The content-based filtering algorithm uses this similarity information to recommend movies that align with a user's preferences. By considering the content of previously liked or rated movies, the system suggests new movies with similar content, offering a personalized and tailored viewing experience. This method is particularly useful in addressing the "cold start" problem, where limited user interaction data is available.

content-based filtering enhances the accuracy and relevance of recommendations by focusing on the content attributes of items. It provides a transparent and interpretable approach to suggesting movies, directly aligning with the goal of delivering a personalized and satisfying user experience in our movie recommendation system

Content-Based Filtering



RESULTS

Our system demonstrated its capability to provide accurate movie suggestions based on user preferences. By focusing on content features like movie descriptions and keywords, we successfully created a personalized experience for users seeking movie recommendations.
Through the use of count vectorization and cosine similarity, we gained valuable insights into the content similarities between movies. This allowed us to understand how well our system identified and recommended movies with similar themes and characteristics.

In [107]:
```
def recommend(movie):
    movie_index=new_df[new_df['title'] == movie].index[0]
    distances=similarity[movie_index]
    movies_list=sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]

    for i in movies_list:
        print(new_df.iloc[i[0]].title)
```

In [109]:
```
recommend('Batman Begins')
```

```
The Dark Knight
Batman
Batman
The Dark Knight Rises
10th & Wolf
```

Future Work

The development of a web interface stands as the next significant milestone for our movie recommendation system. While the core recommendation algorithm has been successfully implemented and validated in a Jupyter Notebook environment, transitioning to a user-friendly web interface opens up new avenues for accessibility and user engagement.The primary focus will be on creating an intuitive and visually appealing web interface that seamlessly integrates with our existing recommendation system Users will have the opportunity to interact with the recommendation engine effortlessly, providing input and receiving personalized movie suggestions Implement a user-friendly input mechanism allowing users to express their preferences easily. This may include options for entering keywords, selecting genres, or any other relevant criteria for movie recommendations..

Snapshots

[40]:
```
import numpy as np
import pandas as pd
```

[41]:
```
movies=pd.read_csv('tmdb_5000_movies.csv')
credits=pd.read_csv('tmdb_5000_credits.csv')
```

[42]:
```
movies.head(1)
```

t[42]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	produ
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"is "nam

[43]:
```
credits.head(1)
```

Out[43]:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

In [44]:
```
movies=movies.merge(credits,on='title')
```

In [45]:
```
movies.head(1)
```

Out[45]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	...	runtime
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	...	162.0

1 rows × 23 columns

Conclusion

As we finish up our project, we're excited about the movie recommendation system we've built. It's like a helpful friend suggesting movies based on what you like. We focused on understanding movie descriptions and used that info to make recommendations.

We started by getting the movie details, cleaning them up, and picking out important things. Then, we turned those details into numbers to see how similar movies are to each other. This way, we could suggest movies with similar vibes.

In Future we will make a simple website where you can easily tell us what kind of movies you like.It's all about making your movie-searching experience more fun and personal.

we faced some challenges along the way, like making sure we have good data and figuring out how to handle complicated movie information. But we're always learning and improving.