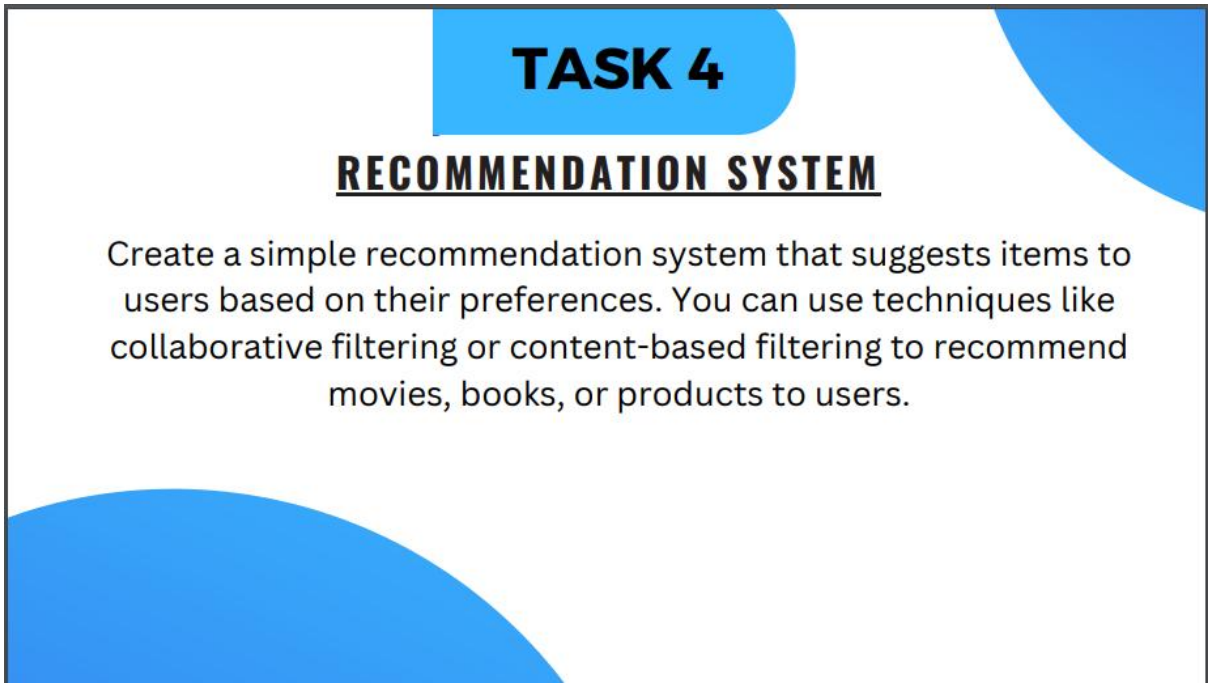
# Task Four -Recommendation system



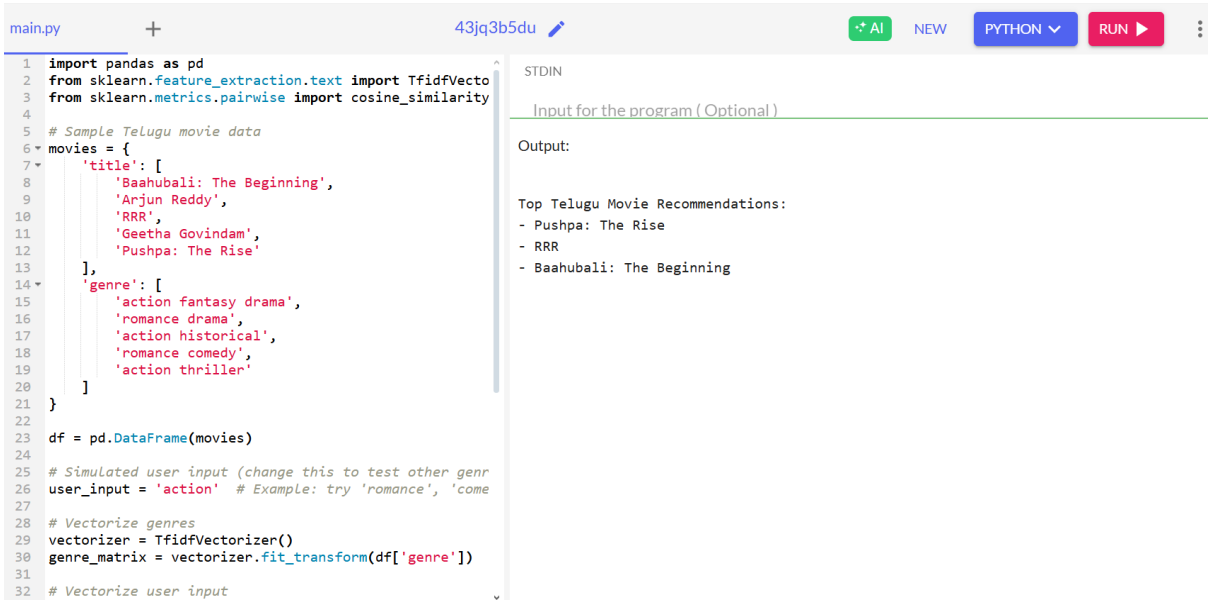

Source code

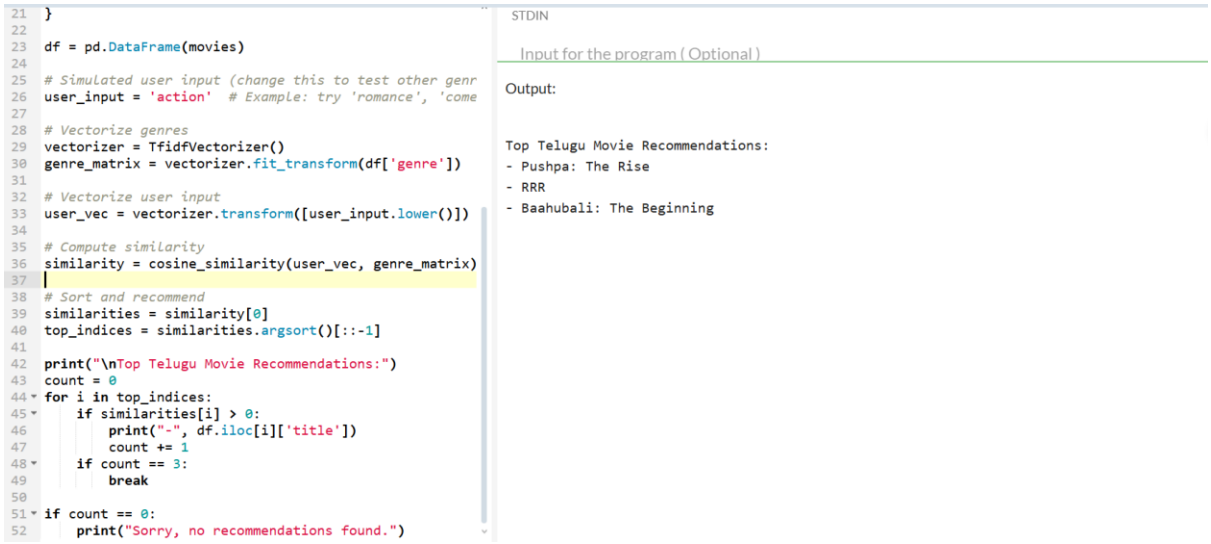
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity


# Sample Telugu movie data
movies = {
    'title': [
        'Baahubali: The Beginning',
        'Arjun Reddy',
        'RRR',
        'Geetha Govindam',
        'Pushpa: The Rise'
```

```
    ],
    'genre': [
        'action fantasy drama',
        'romance drama',
        'action historical',
        'romance comedy',
        'action thriller'
    ]
}

df = pd.DataFrame(movies)

# Simulated user input (change this to test other genres)
user input = 'action'  # Example: try 'romance', 'comedy', 'thriller'

# Vectorize genres
vectorizer = TfidfVectorizer()
genre_matrix = vectorizer.fit_transform(df['genre'])

# Vectorize user input
user_vec = vectorizer.transform([user_input.lower()])

# Compute similarity
similarity = cosine_similarity(user_vec, genre_matrix)

# Sort and recommend
similarities = similarity[0]
top_indices = similarities.argsort()[::-1]
```

```
print("\nTop Telugu Movie Recommendations:")
count = 0
for i in top_indices:
    if similarities[i] > 0:
        print("-", df.iloc[i]['title'])
        count += 1
    if count == 3:
        break

if count == 0:
    print("Sorry, no recommendations found.")
```

output





```
}

df = pd.DataFrame(movies)

# Simulated user input (change this to test other genr
user_input = 'action'  # Example: try 'romance', 'come

# Vectorize genres
vectorizer = TfidfVectorizer()
genre_matrix = vectorizer.fit_transform(df['genre'])

# Vectorize user input
user_vec = vectorizer.transform([user_input.lower()])

# Compute similarity
similarity = cosine_similarity(user_vec, genre_matrix)

# Sort and recommend
similarities = similarity[0]
top_indices = similarities.argsort()[::-1]

print("\nTop Telugu Movie Recommendations:")
count = 0
for i in top_indices:
    if similarities[i] > 0:
        print("-", df.iloc[i]['title'])
        count += 1
    if count == 3:
        break

if count == 0:
    print("Sorry, no recommendations found.")
```

STDIN

Input for the program ( Optional )

Output:

```
Top Telugu Movie Recommendations:
- Pushpa: The Rise
- RRR
- Baahubali: The Beginning
```

Explanation

**1. Import necessary libraries**

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

- pandas: Used to create and handle the movie data as a DataFrame.
- TfidfVectorizer: Converts text (genres) into TF-IDF vectors.
- cosine_similarity: Measures similarity between the user's input and movie genres.

**2. Sample Telugu movie data**

movies = {

'title': [

'Baahubali: The Beginning',

'Arjun Reddy',

'RRR',

'Geetha Govindam',

'Pushpa: The Rise'

],

'genre': [

```
        'action fantasy drama',
        'romance drama',
        'action historical',
        'romance comedy',
        'action thriller'
    ]
}
```

- movies: A dictionary storing movie titles and their genres (in short text descriptions).

**3. Convert dictionary to DataFrame**

```
df = pd.DataFrame(movies)
```

- Creates a **DataFrame** df for easier handling:

| index | title | genre |
|---|---|---|
| 0 | Baahubali: The Beginning | action fantasy drama |
| 1 | Arjun Reddy | romance drama |
| 2 | RRR | action historical |
| 3 | Geetha Govindam | romance comedy |
| 4 | Pushpa: The Rise | action thriller |

**4. User input for genre preference**

```
user_input = 'action'
```

- This simulates a user saying they like **"action"** movies.
- You can change this to other genres like 'romance', 'comedy', 'thriller', etc., to get different results.

**5. Convert genres into vectors using TF-IDF**

```
vectorizer = TfidfVectorizer()
genre_matrix = vectorizer.fit_transform(df['genre'])
```

- TfidfVectorizer turns each genre into a numerical vector based on the importance of each word.
- genre_matrix is a matrix where each row represents a movie and each column a unique term from all genres.

**6. Vectorize the user input**

user_vec = vectorizer.transform([user_input.lower()])

- Converts the user's input ('action') into the same vector format.
- Ensures fair comparison with movie genre vectors.

**7. Compute cosine similarity**

similarity = cosine_similarity(user_vec, genre_matrix)

- Calculates **cosine similarity** between the user's vector and each movie's genre vector.
- Returns a 1x5 array of similarity scores (one for each movie).

**8. Sort similarities in descending order**

similarities = similarity[0]

top_indices = similarities.argsort()[::-1]

- similarity[0]: Gets the 1D array of similarity scores.
- argsort()[::-1]: Sorts indices of movies from most similar to least.

**9. Print top recommendations**

```
print("\nTop Telugu Movie Recommendations:")
count = 0
for i in top_indices:
    if similarities[i] > 0:
        print("-", df.iloc[i]['title'])
        count += 1
    if count == 3:
```

break

- Loops through sorted indices and prints top 3 movies **with non-zero similarity**.
- df.iloc[i]['title']: Fetches the movie title using the index.

**10. Handle no recommendations**

if count == 0:

print("Sorry, no recommendations found.")

- If no movie has a similarity score above 0, informs the user.

**Expected Output for user_input = 'action'**

TF-IDF will match movies containing the word "action" in their genre.

So the **output will be**:

Top Telugu Movie Recommendations:

- Baahubali: The Beginning

- RRR

- Pushpa: The Rise

All of these movies include "action" in their genre description.