Recommend books to users based on their past preferences (collaborative filtering or content-based filtering).

EX.NO : 10

DATE : / /2025

To write a program to develop a book recommendation system using collaborative filtering and content-based filtering based on user reading history.

AIM:

To write a program to detect plagiarism between two documents using Cosine Similarity (TF-IDF) and Jaccard Coefficient (shingle/ n-gram overlap), and to decide if documents are likely plagiarized based on similarity thresholds.

ALGORITHM:

Step 1: Start
Step 2: Import necessary libraries.
Step 3: Load and preprocess news dataset.
Step 4: Clean articles (lowercase, remove punctuation, stopwords).
Step 5: Convert text to TF■IDF vectors.
Step 6: Split into train and test sets.
Step 7: Train Logistic Regression model.
Step 8: Evaluate accuracy and classification report.
Step 9: Test with a custom input article.

PROGRAM:

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

# Sample book dataset
books = pd.DataFrame({
    'title': [
        "Data Science Handbook",
        "Deep Learning Guide",
        "Python Programming",
        "Machine Learning with Python",
        "Artificial Intelligence Basics"
    ],
    'author': ["A", "B", "C", "D", "E"],
    'genre': ["DS", "ML", "Programming", "ML", "AI"],
    'description': [
        "Guide to data science techniques and tools.",
        "Deep learning concepts and implementation.",
        "Core python programming concepts.",
        "Machine learning algorithms using python.",
        "Basics of artificial intelligence and concepts."
    ]
})

# Create combined content
books['content'] = books['author'] + " " + books['genre'] + " " +
books['description']

# TF-IDF Vectorization
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(books['content'])

# Similarity scores
similarity = cosine_similarity(tfidf_matrix)

def recommend_book(book_title):
    idx = books[books['title'] == book_title].index[0]
    scores = list(enumerate(similarity[idx]))
    scores = sorted(scores, key=lambda x: x[1], reverse=True)
```

```
    top_books = [books.iloc[i[0]]['title'] for i in scores[1:4]]
    return top_books

print("Content-Based Recommendations:")
print(recommend_book("Python Programming"))
```

RESULT:

Thus, a program has been successfully executed.