BUILDING AN E- COMMERCE RECOMMNDATION ENGINE USING PYTHON:

Course code :AL3391

Course Title: Artificial Intelligence

Submitted by:

1. **BRINDA SRI M(953622243017)**
2. **MAHA LAKSHMI K(953622243052)**
3. **RAMYA R(953622243078)**

TITLE

BUILDING AN E-COMMERCE RECOMMENDATION SYSTEM USING PYTHON:

AIM:

To build an E-commerce recommendation system engine using python .

OBJECTIVE:

An ecommerce recommendation engine aims to create a more personalized and enjoyable shopping experience for users, driving engagement, increasing conversions, and ultimately building a stronger relationship between the users and the ecommerce platform. The specific objectives may vary based on the goals and priorities of the ecommerce business implementing the recommendation system.

ABOUT THE PROJECT:

Improving User Engagement: By recommending products that are relevant to users' preferences and needs, the recommendation engine aims to keep users actively engaged on the ecommerce platform.

Increasing Conversions: Personalized recommendations can lead to higher conversion rates as users are more likely to purchase products that align with their interests. This, in turn, positively impacts the revenue of the ecommerce platform.

Enhancing User Experience: A good recommendation engine contributes to an improved overall user experience by helping users discover new products they might be interested in. This can be especially valuable in large ecommerce catalogs where users may have difficulty finding products on their own.

Increasing Customer Loyalty: Personalized recommendations can foster a sense of

connection between users and the ecommerce platform. Users who consistently receive relevant suggestions are more likely to become repeat customers.

Optimizing Product Discovery: Ecommerce platforms often have vast catalogs, and users may not be aware of all the products available. The recommendation engine aids in surfacing products that users might have otherwise missed, promoting a broader exploration of the inventory.

Adapting to User Preferences: Recommendation engines leverage user behavior and historical data to adapt and improve their suggestions over time. As users interact more with the platform, the recommendations become increasingly tailored to their evolving preferences.

ALGORITHM:

Import Libraries:

Import the necessary libraries for data manipulation, machine learning, and collaborative filtering.

Prepare Data:

Create sample data for user-item interactions and product information.

Convert Data to DataFrames:

Convert the data to Pandas DataFrames for easier manipulation.

Collaborative Filtering:

a. **Define the Reader Object:** Specify the rating scale for collaborative filtering.
b. **Load the Dataset:** Load the user-item rating data using the Surprise library.
c. **Split the Dataset:** Divide the dataset into training and testing sets.
d. **Use KNNBasic Algorithm for Collaborative Filtering:** Specify the collaborative filtering algorithm (KNN) and its options.
e. **Train the Model:** Train the collaborative filtering model with the training set

Content-Based Filtering:

a. **Create a TF-IDF Vectorizer for Item Descriptions :** Tokenize and vectorize product descriptions using TF-IDF.
b. **Compute Cosine Similarity Between Items Based on Descriptions:** Calculate cosine similarity between products based on TF-IDF vectors.

6. Get Collaborative Recommendations for a User:

Provide a user ID and get collaborative filtering recommendations for that user**. Get**

7. Content-Based Recommendations for a Product:

Specify a product ID and get content-based recommendations for similar products.

PROGRAM:

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from surprise import Dataset, Reader, KNNBasic
from surprise.model_selection import train_test_split

data = [
    ('User1', 'Product1', 4),
    ('User1', 'Product2', 5),
    ('User2', 'Product1', 3),
    ('User2', 'Product3', 4),
    ('User3', 'Product2', 5),
```

```
    ('User3', 'Product3', 4),
]

product_data = [
    ('Product1', 'Description of Product1'),
    ('Product2', 'Description of Product2'),
    ('Product3', 'Description of Product3'),
]

df = pd.DataFrame(data, columns=['user', 'item', 'rating'])
product_df = pd.DataFrame(product_data, columns=['item', 'description'])

reader = Reader(rating_scale=(1, 5))

dataset = Dataset.load_from_df(df[['user', 'item', 'rating']], reader)

trainset, testset = train_test_split(dataset, test_size=0.2)

sim_options = {
    'name': 'cosine',
    'user_based': True,
}
model = KNNBasic(sim_options=sim_options)

model.fit(trainset)

tfidf_vectorizer = TfidfVectorizer(stop_words='english')
```

```
tfidf_matrix = tfidf_vectorizer.fit_transform(product_df['description'])


cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)


def get_collaborative_recommendations(user_id, model, n=5):
    unrated_items = df.loc[df['user'] == user_id]['item'].unique()
    user_predictions = [(item, model.predict(user_id, item).est) for item in unrated_items]
    user_predictions.sort(key=lambda x: x[1], reverse=True)
    return [item for item, _ in user_predictions[:n]]


def get_content_based_recommendations(item_id, cosine_sim_matrix, n=5):
    item_index = product_df.index[product_df['item'] == item_id].tolist()[0]
    similar_items = list(enumerate(cosine_sim_matrix[item_index]))
    similar_items = sorted(similar_items, key=lambda x: x[1], reverse=True)[1:n+1]
    similar_item_indices = [i for i, _ in similar_items]
    return list(product_df['item'].iloc[similar_item_indices])



user_id = 'User1'
collaborative_recommendations = get_collaborative_recommendations(user_id, model, n=3)
print(f"Collaborative Filtering Recommendations for {user_id}:
{collaborative_recommendations}")



item_id = 'Product1'
content_based_recommendations = get_content_based_recommendations(item_id,
cosine_sim_matrix=cosine_sim, n=3)
```

```
print(f"Content-Based Filtering Recommendations for {item_id}:
{content_based_recommendations}")
```

OUTPUT:

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
Collaborative Filtering Recommendations for User1: ['Product2', 'Product1']
Content-Based Filtering Recommendations for Product1: ['Product2', 'Product3']
```

CONCLUSION:

The provided ecommerce recommendation system program employs a combination of collaborative filtering and content-based filtering techniques to deliver personalized product recommendations. The program uses the Surprise library to implement collaborative filtering with the K-Nearest Neighbors (KNN) algorithm.Collaborative filtering relies on user-item interactions to identify similar users and make recommendations based on their preferences. The KNN model is trained on user-item ratings, and predictions are generated for unrated items. Content-based filtering is implemented using TF-IDF vectors and cosine similarity. Product descriptions are vectorized using TF-IDF, and the cosine similarity matrix is

calculated to find similar products based on content.Content-based recommendations are made by identifying products with similar descriptions

The ecommerce recommendation system aims to enhance user experience by offering personalized product suggestions. Collaborative filtering leverages user behavior, while content-based filtering focuses on product features. The combination of both techniques provides a more comprehensive and accurate recommendation systemThe program is designed to be adaptable, and additional data or features can be incorporated to further improve recommendation accuracy.

The provided ecommerce recommendation system program serves as a starting point for building a personalized recommendation engine. It demonstrates the integration of collaborative and content-based filtering techniques, providing a foundation for more advanced and tailored recommendation systems in ecommerce applications.