# Topic: Recommendation Engine

**Instructions:**

Please share your answers filled in-line in the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Vishvash C Batch ID:** 23012024

**Topic: Recommender Engine**

**Guidelines:**

**1. An assignment submission is considered complete only when the correct and executable code(s) is submitted along with the documentation explaining the method and results. Failing to submit either of those will be considered an invalid submission and will not be considered a correct submission.**

**2. Ensure that you submit your assignments correctly and in full. Resubmission is not allowed.**

**3. Post the submission you can evaluate your work by referring to the keys provided. (will be available only post the submission).**

**Hints:**

1. **Business Problem**
   1. **What is the business objective?**
   2. **Are there any constraints?**
2. **Work on each feature of the dataset to create a data dictionary as displayed in the image below:**



1. **Data Pre-processing**

**2.1 Data Cleaning and Data Mining.**

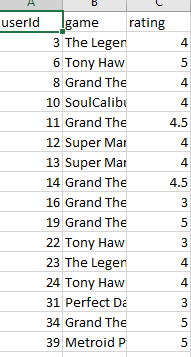
1. **Exploratory Data Analysis (EDA):**
   1. **Summary.**
   2. **Univariate analysis.**
   3. **Bivariate analysis.**
2. **Model Building**
   1. **Build the Recommender Engine model on the given data sets.**
3. **Write about the benefits/impact of the solution - in what way does the business (client) benefit from the solution provided?**

**Problem Statement: -**

Q1) Build a recommender system with the given data using UBCF.

This dataset is related to the video gaming industry and a survey was conducted to build a

recommendation engine so that the store can improve the sales of its gaming DVDs. A snapshot of the dataset is given below. Build a Recommendation Engine and suggest top-selling DVDs to the store customers.



**Questions to Trigger Your thoughts:**

1. **What are Recommendation Systems?**

Recommendation systems are algorithms and techniques used to predict and suggest items or content that a user may be interested in, based on their preferences and past interactions.

1. **How are Knowledge-based Recommender Systems different from Collaborative and Content-based Recommender Systems?**

Knowledge-based recommender systems rely on explicit knowledge about the items and user preferences. Collaborative systems use past interactions and preferences of similar users, while content-based systems analyze the characteristics of items and user preferences.

1. **What is the difference between Collaborative and Content-based Recommender Systems?**

Collaborative systems recommend items based on the preferences and behavior of similar users, while content-based systems recommend items based on the features and attributes of the items and the user's past interactions with similar items.

1. **How does the surprise library work in the recommendation engine?**

The Surprise library is a Python scikit for building and evaluating recommendation systems. It provides various collaborative filtering algorithms and evaluation metrics to assess the performance of these algorithms.

1. **What are the three main types of recommendation engines?**

The three main types of recommendation engines are Collaborative Filtering, Content-based Filtering, and Hybrid Filtering.

1. **What is the logic behind the recommendation engine?**

The logic behind a recommendation engine is to analyze user preferences or item attributes and use this information to predict or suggest items that the user might like or find useful.

1. **What are the benefits of recommendation engines?**

Benefits of recommendation engines include improved user experience, increased user engagement, higher conversion rates, better customer satisfaction, and increased sales or revenue.

1. **What is NLP usage in recommendation engines?**

NLP (Natural Language Processing) can be used in recommendation engines to analyze text data such as user reviews, product descriptions, or user queries to understand user preferences and improve the quality of recommendations.

1. **What is a Model-Based Collaborative approach?**

A model-based collaborative approach involves building a statistical model based on the user-item interactions to make predictions about user preferences for items that they haven't interacted with yet.

1. **Which are used for filtering in a Recommendation Engine?**

Filtering methods used in recommendation engines include Collaborative Filtering, Content-based Filtering, and Hybrid Filtering.

1. **For an eCommerce website which one is explicit data?**

Explicit data for an eCommerce website could include user ratings, reviews, purchase history, and explicit preferences provided by the user.

1. **How would you create a Recommender System for Text Inputs?**

For text inputs, a recommender system could use techniques like Natural Language Processing (NLP) to analyze text data such as user reviews or item descriptions, and then recommend items based on the similarity of text features or sentiments.

1. **What are the different methods that you can collect User Data for the Recommendation Process?**

Different methods for collecting user data for the recommendation process include explicit feedback (ratings, reviews, preferences provided by the user), implicit feedback (browsing history, purchase history, clicks), and demographic information.

1. **What are the different types of Memory-Based Collaborative approaches?**

Different types of memory-based collaborative approaches include User-Based Collaborative Filtering and Item-Based Collaborative Filtering.

Top of Form

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Feature | Description | Type | Relevance |
| userId | User ID | Nominal | Relevant |
| game | Game title | Nominal | Relevant |
| rating | Rating given by user | Quantitative | Relevant |

**Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Mar 28 14:48:08 2024

@author: Lenovo

"""

'''

CRISP-ML(Q) process model describes six phases:

# - Business and Data Understanding

# - Data Preparation

# - Model Building

# - Model Evaluation and Hyperparameter Tuning

# - Model Deployment

# - Monitoring and Maintenance

# Business Problem: Build a recommender system with the given data using UBCF.

This dataset is related to the video gaming industry and a survey was conducted to build a

recommendation engine so that the store can improve the sales of its gaming DVDs. A snapshot of the dataset is given below. Build a Recommendation Engine and suggest top-selling DVDs to the store customers.

# Objective(s): Maximize game effecient recommendation

# Constraint(s): Minimize the user's game selection time

Success Criteria:

a. Business: Increase the Number of games purchase by 10% to 15%

b. ML:

c. Economic: Additional revenue of $100K to $120K

Data Collection:

Dimension: 5000 rows and 3 columns

Name of Feature | Description | Type | Relevance

userId | User ID | Nominal | Relevant

game | Game title | Nominal | Relevant

rating | User ratings | Quantitative | Relevant

'''

# Importing all required libraries, modules

import pandas as pd

from sklearn.metrics.pairwise import cosine\_similarity

import joblib

games = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Recommender\_System/Assignments/Dataset/Datasets\_Recommendation Engine/game.csv", encoding = 'utf8')

# Database Connection

from sqlalchemy import create\_engine, text

engine = create\_engine("mysql+pymysql://{user}:{pw}@localhost/{db}".format(user = "root", pw = "1234", db = "recommend\_db"))

# Upload the Table into Database

games.to\_sql('games', con = engine, if\_exists = 'replace', chunksize = 1000, index = False)

# Read the Table (data) from MySQL database

sql = 'select \* from games'

data = pd.read\_sql\_query(text(sql), con = engine.connect())

# Create DataFrame

df = pd.DataFrame(data)

# Create user-item matrix

user\_item\_matrix = df.pivot\_table(index='userId', columns='game', values='rating', fill\_value=0)

# Calculate cosine similarity between users

cosine\_similar = cosine\_similarity(user\_item\_matrix, user\_item\_matrix)

# Save the Pipeline for tfidf matrix

joblib.dump(cosine\_similar, 'matrix')

# Load the saved model for processing

cosine\_sim = joblib.load("matrix")

# Function to get similar users

def get\_similar\_users(user\_id, k):

sim\_users = cosine\_sim[user\_id - 1] # User IDs start from 1

similar\_users = sorted(list(enumerate(sim\_users, 1)), key=lambda x: x[1], reverse=True)

similar\_users = [user[0] for user in similar\_users if (user[0] != user\_id) & (user[1] != 0 )]

return similar\_users

# Function to recommend DVDs to a user

def recommend\_dvds(user\_id, k):

if user\_id not in user\_item\_matrix.index:

print("User ID not found in the dataset.")

return []

similar\_users = get\_similar\_users(user\_id, k)

# user\_ratings = user\_item\_matrix.iloc[user\_id - 1]

recommendations = []

for sim\_user in similar\_users:

sim\_user\_ratings = user\_item\_matrix.iloc[sim\_user - 1]

unrated\_dvds = sim\_user\_ratings[sim\_user\_ratings == 0].index

sim\_user\_sim = cosine\_sim[user\_id - 1][sim\_user - 1] # Similarity between users

weighted\_ratings = sim\_user\_ratings \* sim\_user\_sim

# recommendations.extend(weighted\_ratings[exclude = unrated\_dvds].sort\_values(ascending=False).index)

recommendations.extend(weighted\_ratings.drop(index=unrated\_dvds).sort\_values(ascending=False).index)

return list(set(recommendations[:k]))

# Example usage

user\_id = 2341 # Change user ID as per requirement

k = 5

recommended\_dvds = recommend\_dvds(user\_id, k)

recommended\_dvds

**Output:**

(for different input)

user\_id = 14

k = 2

recommended\_dvds = recommend\_dvds(user\_id, k)

recommended\_dvds

Out[12]: ['Super Puzzle Fighter II Turbo HD Remix', 'Grand Theft Auto V']



