RECOMMENDATION SYSTEM FOR GAMERS

THE GAMERS

001
DAFRANE Safiya

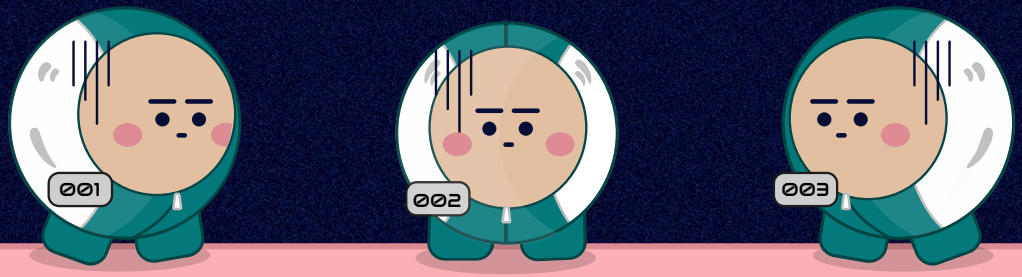
002
LARTIGUE Hugo

003
ROUGE-THOMAS Baptiste

004
ZARKAN Soumia

005
KNOUZI Mohamed-Amine

006
CARRIER Thomas



Problematic

Description ;

We are currently in the data team at Ubisoft. We are going to deal with a new problem today.

We have been asked to create a recommendation system in order to push customers to buy games that match their needs.
This system must be based on the identification of similar games to those chosen in order to allow customers to buy a game that matches them.

We start with a dataset without column name and without note. These data will have to be adapted to our problematic to be able to apply "cosine similarity".

This data could be used to train a game recommendation model that suggests games similar to those the user has previously enjoyed. Information about games purchased and played can also be used to predict which games the user is likely to purchase or play in the future, which could be useful for video game companies looking to market their products to new customers.

Description of the dataset

Here are the characteristics of our dataset

1. Data type: The dataset contain numeric data, text data.
1. Dataset size: use of df.head() to give us an overview of our data set. We see that we have 200 000 rows and 5 columns. The variables (columns):

This dataset represents the gaming activity of users identified by the example ID "151603712". Each row contains information about a specific game, including the name of the game, whether or not it was purchased, the total time played for that game, and a binary indicator of whether or not it was recommended.

The dataset contains 200,000 rows, so the dataset has many different games and that the dataset could provide interesting information about users' game preferences.

Process and Result

Here are the steps we have taken to answer our problem and the results obtained

1.
```
[ ]  cols = {'151603712':'game_id','The Elder Scrolls V Skyrim':'game_name','purchase':'status','1.0':'Hourplayed'}
     df.rename(columns = cols,inplace =True)
```

The rename() method with the use of cols and inplace=True allows to rename the columns of the DataFrame.
The first column is renamed from "151603712" to "game_id", the second column is renamed from "The Elder Scrolls V Skyrim" to "game_name", the third column is renamed from "purchase" to "status", and the fourth column is renamed from "1.0" to "Hourplayed".

2.
```
[ ]  df.drop(columns =['0'],inplace =True)
```

drop() with the use of columns and inplace=True allows to delete a specific column in a DataFrame. The argument columns=['0'] specifies that the column to delete has the name "0". This column is then deleted from the DataFrame.

3.
```
[ ]  # Droping a duplicated values:
     df.drop_duplicates(inplace=True)

[ ]  df.duplicated().sum()

     0
```

This allows you to see the number of duplicate rows and then delete them.

Process and Result

Converting Hours into Ratings

4.

```
# Collecting Id of player that had played a game for >/= than to 2 hours
df= df[(df['Hourplayed']>=2) & (df['status']=='play')]
```

```
df = df[df.groupby('game_name').game_id.transform(len)>=20]
```

These two lines of code perform filtering on a DataFrame df to collect the IDs of players who have played a game for more than 2 hours and who have played a given game at least 20 times.

The first line of code filters the df DataFrame using two conditions:

(df['Hourplayed']>=2 selects all rows where the 'Hourplayed' column has a value greater than or equal to 2.
(df['status']=='play') selects all rows where the 'status' column has the value 'play'.

5.

```
df['game_id'] = df['game_id'].astype(str)
```

```
average = df.groupby(['game_name'],as_index = False).Hourplayed.mean()
```

The first line of code is used to convert the 'game_id' column to a string (str).
The second line of code is used to group the DataFrame data by game ('game_name'). The mean() method is then used to calculate the average hours played for each game group.

The result is stored in a new DataFrame called 'average', which contains two columns: 'game_name' (which contains the game name) and 'Hourplayed' (which contains the average hours played for each game).

6.

```
df = df.merge(average,on = 'game_name')
```

```
# keeping all important columns (game_id,game_name,rating ) and drop all other columns
df.drop(columns = [ 'status', 'Hourplayed', 'avg_hourplayed'],inplace =True )
```

We merge two data frames with the "game name" key. Then we keep only the columns we want to analyse.

Parsing the data

The code used seeks to calculate the cosine similarity between each pair of games in a DataFrame using the scikit-learn library and store the results in a 'cosine_sim' matrix.
This allows us to recommend similar games to a user based on their game preferences.

We therefore seek to create a cosine similarity between games, based on stored video game data.
Then we transform the textual data into a numerical format that can be used to compute the cosine similarity between games.

Then, the cosine similarity matrix is calculated using the cosine_similarity() function of sklearn.metrics.pairwise. This function calculates the cosine similarity between each pair of games in the 'count_matrix' and stores the result in the 'cosine_sim' matrix.

In summary, the above code aims to compute cosine similarity between games and users from video game data stored in a pandas DataFrame using the scikit-learn library and other libraries. The results are stored in separate pandas DataFrames ('df_game' for similarity between games and 'df_user' for similarity between users), which can be used for recommending customized games for a specific user or for recommendation of new games similar to a specific game, based on the cosine similarity between games and users.

Our recommendations

1.
```python
def similar_games(game):
    count = 1
    print('Similar games to {} are : \n'.format(game))
    for item in df_game .sort_values(ascending=False,by = game).index[1:6]:
        print('N.{} = {}'.format(count,item))
        count+=1
```

This Python code aims to find the 5 most similar games to a given game.
So this function scans a pandas DataFrame to find the 5 most similar games to a given game, then displays their name with a number from 1 to 5.

```
similar_games('Fallout 4')

Similar games to Fallout 4 are :

N.1 = Endless Legend
N.2 = Middle-earth Shadow of Mordor
N.3 = The Wolf Among Us
N.4 = Divinity Original Sin
N.5 = Papers, Please
```

The first game recommended for "Fallout 4" is "Endless Legend". This means that, according to the similarity criteria defined in the df_game dataframe, "Endless Legend" is considered the most similar game to "Fallout 4".

In this case, "Fallout 4" and "Endless Legend" probably share similar characteristics, such as the post-apocalyptic theme, the RPG elements, or perhaps aspects of city management or technology research. This is why "Endless Legend" has been recommended as the most similar game to "Fallout 4".

Our recommendations

2.

```
similar_games('The Elder Scrolls V Skyrim')

def similar_games(game):
    count = 1
    print('Similar games to {} are : \n'.format(game))
    for item in df_game .sort_values(ascending=False,by = game).index[1:6]:
        print('N.{} = {}'.format(count,item))
        count+=1
```

This feature recommends the 5 most similar games to "The Elder Scrolls V Skyrim" based on users' gaming preferences. This can help players discover new games they might like, based on the games they have already enjoyed.

The 5 most similar games to "The Elder Scrolls V Skyrim", according to the data in the df_game dataframe, are:

- Age of Mythology Extended Edition
- Metro Last Light
- Starbound
- Deus Ex Human Revolution - Director's Cut
- Metro 2033

This means that, according to the similarity criteria defined in the df_game dataframe, these games are the closest to "The Elder Scrolls V Skyrim" in terms of gameplay characteristics, such as theme, style, game mechanics, etc.

These results can help gamers who liked "The Elder Scrolls V Skyrim" to discover new games that they might also like according to their gaming preferences.