Assignment #3 – Recommender Systems

Due Date: Week #12 (8-April-2022 at 11:59 PM)

Purpose:
The purpose of this Lab assignment is to:

1. To practice building non-personalized recommender systems using association rules.
2. To practice building content-based recommender systems.

General Instructions:

Be sure to read the following general instructions carefully:

1. This assignment must be completed individually by all the students.
2. Only provide the requested screenshots and make sure to have a complete screenshot; partial screenshots will not earn any marks.
3. You will have to add all the analysis and screenshots in the Analysis report.
4. You will have to provide a **10-minute demonstration video for your solution** and upload the video together with the solution on **eCentennial** through the assignment link. See the "video recording instructions" at the end of this document.
5. In your max 10-minute demonstration video, you should explain your solution clearly, going over the main code blocks and the purpose of each method also demoing the execution of the code. YouTube links and links to google drive or any other media are not acceptable; the actual recording file must be submitted.
6. Any submission without an accompanying video will **lose 50%** of the grade.
7. Any submission without an accompanying Analysis report will **lose 50%** of the grade.

Assignment Pre-requisites:

1. Python
2. Apyori
3. Datasets attached to this assignment as indicated in the exercises

Assignment Exercises

Exercise #1: Acossiation rules (50%)

In this exercise, you will be building a small recommender app to recommend the top ingredients used in a certain cuisine type, for example, Italian, Greek cuisines…etc.
You will base your recommendations after mining recipe data posted on the web. The data has already been collected and available in the attached "recipes.Json" file to this assignment. You will be using the apriori algorithm to come up with the rules.

Exercise requirements:

1. Load the data and carry out some basic data analysis and exploration. Note the results of your analysis in your analysis report. At minimum, carry out the following:
    a. Note the total number of instances of recipes
    b. Note the number of cuisines available in the data.
    c. Create a table illustrating each cuisine type and number of recipes available in the file related to that cuisine.
2. Your app should receive a 'cuisine type' as input from the user, for example, 'Greek', 'Italian'…etc. If the 'cuisine type' is not available, then reply to the user "We don't have recommendations for XXX" where XXX is the inputted cuisine type. Then prompt the user to enter a different 'cuisine type'. (hint: use python input())
3. If the 'cuisine type' recipe data is available in the json file, then:
    a. Analyze all the ingredients available under the inputted "cuisine type" using the apriori algorithm, according to the following parameters:
        i. Set the support value to 100/total # of recipes for the selected cuisine'
        ii. Set the confidence value to 0.5
4. Present back to the user the following:
    a. The top group of ingredients that the algorithm calculates for the inputted cuisine type, i.e. the most frequent dataset. (hint: This would be stored in the first record of the RelationRecords returned from the algorithm)
    b. All rules with lift value greater than two.
5. Continue accepting input from the user and responding until the user enters an "exit" text.
6. Name your python script firstname_cusine_recommender.py and attach it to your submission. Make sure your script is callable from the command prompt without errors.

Exercise #2:Content based filter (50%)

Scenario: you have been given the amazon meta dataset for digital music. You have been asked to use this data to recommend top ten song titles to users who interact with your simple app. To achieve this, you will build a content-based filter that uses the songs' features.

Exercise requirements:

1. Load the dataset named "meta_Digital_Music.json.gz," attached to this assignment into a dataframe, name it *songs_firstname*. (Hint: on the dataset publisher webpage at http://jmcauley.ucsd.edu/data/amazon/links.html , there is a "sample code" that you can reuse to load the data; or you can build your own)
2. Data exploration:
    a. Carry out a thorough exploration and note the results into your analysis report. Make sure to check for empty data, not just null.
    b. Also, in your analysis report, suggest which columns you will take into consideration for the recommender system and why. (There are 19 columns in total)
    c. Based on the output of points a & b suggest any filtering steps, for example, if you need to drop any columns or filter out any rows and explain why. Write your explanation in the analysis report.
3. Feature engineering:
    a. Clean your data and prepare your feature space based on the results in point#2 above. You might combine columns, transform…etc. Make sure you follow all the recommendations you noted in your analysis report.
    b. If your feature space has text data, which most likely is the case:
        i. Pre-process the data and note the steps in your analysis report.
        ii. Create TF-IDF vectors for the textual description (or overview) of every song
    c. Compute the pairwise cosine similarity score of every song title.
    d. Store the recommendations into a separate file that your simple app will access.
4. Recommender function: Write the recommender function that takes in a song title as an argument and outputs the top ten 'song titles' most similar to it.
    a. Your app should receive a 'song title' as input from the user, for example, 'Long Legends', 'There can be miracles'…etc. if the 'song title' is not available, then reply to the user "*We don't have recommendations for XXX*" where XXX is the inputted song title. Then prompt the user to enter a different 'song title'. (hint: use python input())
    b. If the 'song title' is available, present the top-10 most similar song titles back to the user as a recommendation.
    c. Continue accepting input from the user and responding until the user enters an "exit" text.
5. Name your python script *"firstname_songs_recommender.py"* and attach it to your submission. Make sure your script is callable from the command prompt without any errors.

Naming and Submission Rules:

1. You must name your submission according to the following rule: **YourFullname_COMP262_assignmentnumber**.Example: **AdamPerjouski_COMP262_assignment1**
2. Upload the submission file on e-Centennial using the Assignment link(s).
3. In total you should submit the following:
    a. One demonstration video
    b. One python script for exercise #1
    c. One python script for exercise #2
    d. One analysis report covering both exercises. Make sure you write your name and student Id in the analysis report.
4. Put all the above files in a single folder, compress it, and submit the compressed file.

Rubric (applies to each exercise)

Evaluation criteria	**Not acceptable**	**Below Average**	**Average**	**Competent**	**Excellent**
	0% - 24%	**25%-49%**	**50-69%**	**70%-83%**	**84%-100%**
Requirements in exercises 50%	Missing all requirements required	Some requirements are implemented.	Majority of requirements are implemented but some are malfunctioning.	Majority of requirements implemented.	All requirements are implemented Correctly.
Instruction/ Code Documentation on python script 5%	No comments explaining code. Missing screenshots	Minor comments are implemented.	Some code is correctly commented.	Majority of code is correctly commented.	All code is correctly commented.
Written analysis Content 15%	Missed all the key ideas; very shallow.	Shows some thinking and reasoning but most ideas are underdeveloped.	Indicates thinking and reasoning applied with original thought on a few ideas.	Indicates original thinking and develops ideas with sufficient and firm evidence.	Indicates synthesis of ideas, in-depth analysis and evidences original thought and support for the topic.
Written analysis report format and organization 5%	Writing lacks logical organization. It shows no coherence and ideas lack unity. Serious errors. No transitions.	Writing lacks logical organization. It shows some coherence but ideas lack unity. Serious errors.	Writing is coherent and logically organized. Some points remain misplaced.	Writing is coherent and logically organized with transitions used between ideas and paragraphs to create coherence. Overall unity of ideas is	Writing shows high degree of attention to logic and reasoning of all points. Unity clearly leads the reader to the conclusion.

	Format is very messy.	Format needs attention, some major errors.	Format is neat but has some assembly errors.	present. Format is neat and correctly assembled.	Format is neat and correctly assembled with professional look.
Demonstration Video 25%	Very weak no mention of the code changes. Execution of code not demonstrated.	Some parts of the code changes presented. Execution of code partially demonstrated.	All code changes presented but without explanation why. Code demonstrated.	All code changes presented with explanation, exceeding time limit. Code demonstrated.	A comprehensive view of all code changes presented with explanation, within time limit. Code demonstrated.

Demonstration Video Recording

Please record a short video (max 10 minutes) to explain/demonstrate your assignment solution. You may use the Windows 10 Game bar to do the recording:

1. Press the Windows key + G at the same time to open the Game Bar dialog.
2. Check the "Yes, this is a game" checkbox to load the Game Bar.
3. Click on the Start Recording button (or Win + Alt + R) to begin capturing the video.
4. Stop the recording by clicking on the red recording bar that will be on the top right of the program window.

(If it disappears on you, press Win + G again to bring the Game Bar back.)

You'll find your recorded video (MP4 file), under the Videos folder in a subfolder called Captures.

Or

You can use any other video recording package freely available.

References:

i. Dataset recipies : Recipe Ingredients Dataset | Kaggle
ii. http://jmcauley.ucsd.edu/data/amazon/links.html