CO2106 – Data Analytics

CW2 – Web scraping, model building and Evaluation
School of Computing & Mathematical Sciences
University of Leicester

Assessment Information

Assessment Number	2
Contribution to overall mark	70%
Submission Deadline	27 Mar 2026 at 12:00 pm

Assessed Learning Outcomes

This first assignment aims at testing your ability to:

- Collect data from the web and write maintainable Python code for that purpose using web scraping and BeautifulSoup.
- Carry out data pre-processing and visualization.
- Build up and evaluate a data-driven recommender system and a predictive model.

AI Policy

You should not use any AI tools to create any of the gradable content/component in this assignment. However, you may use AI tools to help you check for grammatical and spelling errors. Any failure to comply with this rule will be considered as a case of plagiarism, and you will be referred to the plagiarism officer for the appropriate sanctions.

How to submit

For this assignment, you need to submit the followings:

1. The Python source code written in order to complete the tasks set in the paper. You should submit the Python code files, say emt12_solution.py or solution_emt12.ipynb file for your solution to the given problem. You should

structure your code accordingly. Whenever you decide to write a utility function (not specified in this assignment paper), comment on its purpose.

2. A signed coursework cover which covers your completed and signed declaration on Plagiarism and Collusion.

Please put your source code and the signed coursework cover into a zip file CW2_YouremailID.zip (e.g., CW2_emt12.zip) and then submit your assignment through the module's Blackboard site by the deadline. Note that to submit, you need to click on the Coursework link on Blackboard and then upload your zipped file.

Problem Statement

Part 1: Web scrapping using BeautifulSoup **[Total 10 Marks]**

Consider the problem of collecting data consisting of cooking recipes from a website such as the BBC. Consider for example the cooking recipe from the following URL:

https://www.bbc.co.uk/food/recipes/easiest_ever_banana_cake_42108

Implement the function collect_page_data that takes as input a BBC recipe URL and returns a pandas dataframe with the following columns:

columns=['title', 'total_time', 'image', 'ingredients', 'rating_val', 'rating_count',
'category', 'cuisine', 'diet', 'vegan', 'vegetarian', 'url'].

You should also generate a csv file storing the contents of the dataframe. It is clear that, in this case, your dataframe will contain one row of data; nonetheless you can see how to collect data from a dynamic website. Test your function with appropriate inputs. Full marks will be given depending on correctness and maintainability of your code.

Part 2 (Guided): Building Up a recommender engine **[Total 40 marks]**

Consider the list of Books dataset from the Amazons website. It contains two files:

1. The information of the books with the *bookId*, *title, Author, SubGenre, Height* and *Publisher*

2. The review ratings of the books.

The CSV file for the details of the books books_new.csv and the review ratings file *ratings.csv* are available from the CO2106 Data Analytics Blackboard site.

The books_new.csv file contains features such as *bookId*, *title, Author, SubGenre, Height* and *Publisher.* The ratings.csv contains features such as *userID, bookId, reviewRating.*

Objective: Your aim is to develop a basic recommender engine that enables us to suggest book recommendations for a given user's preference profile.

1. Load the contents of the two files books_new.csv and ratings.csv. Combine the contents of both of these files as a single dataframe. Identify the categorical and numerical features of the resulting dataset. Identify and treat the missing values in the resulting data frame.
   Show the summary statistics. **[10 marks]**
2. Calculate the average ratings for each of the books and show the 10 highest. Compute a 95% confidence interval for the average ratings using the Bootstrapping method by creating 1000 samples of size 100. **[8 marks]**
3. Include in an appropriate dataframe an extra column called column rating count to record the number of ratings along with the average rating. Comment on any relationship that exists between the average rating and the rating count? Suggest a *threshold, if any,* for the number of ratings under which the rating can be considered as not significant? **[8 marks]**
4. Assuming that a user would like a book if and only its rating is greater or equal to 3.6, transform the column rating of the dataset into a binary format with
   a. 1 representing 'like',
   b. -1 representing 'dislike'

   Consider the following selected features:

   ```
   features = ['Title','Author','Genre','SubGenre',
   'Publisher']
   ```

   a) Add to your dataframe a column `combined_features`, which combines all the contents of the features in the given list features as a single string wherein each feature's contents is separated from the other by one space string. Using the class `CountVectorizer` and the function `cosine_similarity`, compute the cosine similarity matrix of the books from the dataframe formed by the `combined_features`. **[6 marks]**

   b) Consider the book 'Orientalism'. Relying on the vector space method, use a matrix-vector product to show the first 10 book recommendations for a user who has liked that particular book. Show the titles of these recommendations along with their similarities. **[8 marks]**

Part 3 (Open-ended): Building up and to evaluate a recommender engine [Total 50 marks]

Consider the entire dataset resulting from the combination of both files `books_new.csv` and `ratings.csv`. You need to treat categorical and numerical variables as such and not proceed as we did in Part 2.

Objective: Your aim is to develop **efficient and accurate** predictive models that enable us to:

- Predict if a user likes a given book based on their taste and the books description.
- Suggest book recommendations for a given user's preference profile.

1. Write a function, *vec_space_method,* which takes in a book and returns the 10 most similar books to the given one. Do this using a suitable matrix-vector product in the Vector Space Method. You must use matrix-vector product as instructed and justify the efficiency of your method (you can add comments into your submitted Python file). **[10 marks]**

2. Write a function, knn_similarity, which takes in a book and returns the 10 most similar books to the given one. Do this by using the KNN algorithm and present the output in a readable manner. **[10 marks]**
3. Consider the following test set composed of 4 users:
   - User 1 likes 'Fundamentals of Wavelets'
   - User 2 likes 'Orientalism'
   - User 3 likes 'How to Think Like Sherlock Holmes'
   - User 4 likes 'Data Scientists at Work'

   Using this test set, evaluate both recommender systems you have built up in Part 3 and in terms of **coverage** and **personalisation.** You can add comments on this evaluation into your submitted Python code file. **[20 marks]**.

4. Write a function, *predict_like*, which predicts whether a user would like that book or not. **Your solution must use any appropriate algorithm covered in this module; any solution outside the algorithms covered in this module will be awarded zero mark**. Investigate how accurate are your predictions. **[10 marks]**

Marking Group Work

Normally, a group will be given the same mark unless some members made little or no contributions. Any group can be called for an interview during marking. All group members **must attend**, explain their contributions, and defend the work submitted.

Marking Criteria

For each question, full marks will be awarded depending on **correctness through testing**, quality of the **code**, **efficiency, graphs** where there is visualisation, **comments/justifications** where needed; see the **marking rubric** below for more details.

Criteria	**Fail (<30%)**	**Compensatible Pass to Pass (30-49%)**	**Class 2.2 to 2.1 (50-69%)**	**Class 1 (70-85%)**	**Class 1 (86-100%)**
Part 1 – Web scrapping using BeautifulSoup **10** possible points	0 – 2.5 points Attempt that shows up to some understanding	3 – 4.5 points An attempt that works but with some issues	5 – 6.5 points + Correct function and tested using at least 3 recipe pages.	7– 8.5 points + well-structured and commented code	9 – 10 points + robustness e.g. uses of exceptions handling
Part 2 – (Guided): Building up a recommender engine **Q1** **10** possible points	0 – 2.5 points Attempt that shows up to some understanding e.g., Load the data	3 – 4.5 points Load the data + identify numerical/categorical variables and missing values	5 – 6.5 points + treat missing values	7 – 8.5 points + summary statistics	9 – 10 points + show the 10 highest recipes
Part 2 – (Guided): Building up a recommender engine **Q2** **8** possible points	0 – 2.5 points Attempt that shows up to some understanding	2.5 – 4.5 points Calculate average ratings for each book	4.5 – 5.5 points + Show the 10 highest ratings + presentation	5.5 – 6.5 points + Implement Bootstrapping to compute the 95% confidence interval of the mean	6.5 – 8 points +Structure & Comment of the code

Part 2 – (Guided): Building up a recommender engine **Q3** **8** possible points	0 – 2.5 points Attempt that shows up to some understanding	2.5 – 4.5 points Basic correct graph	4.5 – 5.5 points Neat graph with self-explanatory information	5.5 – 6.5 points + commentary on the graph	6.5 – 8 points + suggestion of a threshold and justification

Part 2 – (Guided): Building up a recommender engine **Q4.a** **6** possible points	0 – 1.5 points Attempt that shows up to some understanding	1.5 – 2.5 points Correct but not tested	2.5 – 3.5 points Tested and Correct processing of the data	3.5 – 5 points + structured + presentation	5 - 6 points + commented
Part 2 – (Guided): Building up a recommender engine **Q4.b** **8** possible points	0 – 2.5 points Attempt that shows up to some understanding	2.5 – 4.5 points Correct and tested but not as specified.	4.5 – 5.5 points A Correct as specified and tested	5.5 – 6.5 points + well-structured and re-usable code	6.5 – 8 points + Commented code + presentation
Part 3 – (Open-ended): Building up and to evaluate a recommender engine **Q1**	0 – 2.5 points Attempt that shows up to some	3 – 4.5 points A partial answer that	5 – 6.5 points Tested, correct and as specified	7 – 8.5 points +efficiency	9 – 10 points +Commented +well-structured and

	understanding	works			commented
10 possible points					
Part 3 – (Open-ended): Building up and to evaluate a recommender engine **Q2** **10** possible points	0 – 2.5 points Attempt that shows up to some understanding	3 – 4.5 points Partial answer that works	5 – 6.5 points Tested, correct and as specified	7 – 8.5 points +efficiency	9 – 10 points +Commented
Part 3 – (Open-ended): Building up and to evaluate a recommender engine **Q3** **20** possible points	0 – 5 points Attempt that shows up to some understanding	6 – 9 points Evaluation against both metrics but partial	10 – 14 points Evaluation against both metrics but one metric is partial	15 – 17 points Complete evaluation against both metrics	18 – 20 points Well commented and presented comparison of the recommender engines

Part 3 – (Open-ended): Building up and to evaluate a recommender engine **Q4** **10** possible points	0 – 2.5 points	3 – 4.5 points Basic model (as specified) that works without evaluation	5 – 6.5 points Model (as specified) that is well justified and evaluated	7 – 8.5 points + Well-structured and commented	9 – 10 points + High predictive accuracy