UDACITY

Return to Classroom

Recommendations with IBM

REVIEW

CODE REVIEW

HISTORY

Requires Changes

1 specification requires changes

Excellent work on the submission. 👍👍

I want to congratulate you on submitting this project, which demonstrates your effort, commitment, and skill set. You obviously invested a lot of time and effort into finishing this project, and the outcomes are outstanding. The project shows your dedication to learning and developing as a Data Scientist, and I value your commitment to providing a high-quality solution.

There were a few instances, though, where some of the prerequisites were only partially completed.

In order for you to proceed without difficulty, I have given you feedback that is more thorough and includes the references I've attached. Take a close look at each.

Useful References:

- Class imbalance problem in classification [Article]
- Recommender Systems: An Introduction by Dietmar Jannach [Book]
- Solving Cold User problem for Recommendation system using Multi-Armed Bandit [Article]

Consider the areas that could use improvement as useful feedback for your upcoming projects as you consider them. Make the most of this chance to improve your abilities, broaden your knowledge, and keep aiming for perfection.

Keep in mind that obstacles and problems are a normal aspect of learning. We may overcome challenges and have even greater success in the future if we persevere and are dedicated to getting better.

I hope you have success. Looking forward to your next submission.
You can also ask questions on the Knowledge Portal to receive instant assistance from the mentors.

Have a great day! 🏆🏆

DON'T FORGET TO RATE MY WORK AS PROJECT REVIEWER! YOUR FEEDBACK IS VERY HELPFUL AND APPRECIATED.

Code Functionality & Readability

All the project code is contained in a Jupyter notebook or script. If you use a notebook, it demonstrates the successful execution and output of the code. All tests have passing remarks.

✅ All the project code is contained in a Jupyter notebook or script

Great job on meeting the requirement of having all the project code contained within a Jupyter notebook or script.

By consolidating the code in one place, it becomes more convenient for others to understand, reproduce, and build upon your work. The use of a Jupyter notebook or script also promotes better collaboration and sharing of your project with peers or stakeholders.

✅ If you use a notebook, it demonstrates successful execution and output of the code

Congratulations on fulfilling the requirement of demonstrating successful execution and output of the code within your notebook.

The successful execution and visible output in the notebook offer transparency and enable users to follow along and understand the steps involved. This demonstration is crucial for others to replicate your results, verify the accuracy of your code, and gain confidence in the effectiveness of your solution.

✅ All tests have passing remarks.

Well done on successfully meeting the requirement of having all tests accompanied by passing remarks.

Having passing remarks for all tests not only demonstrates your thoroughness in testing and quality assurance but also promotes confidence in the functionality and robustness of your solution. It enables others to rely on your code and build upon it with trust.

Useful references:

Python's assert: Debug and Test Your Code Like a Pro [Article]

Jupyter Notebook Shortcuts [Article]

Code is well documented and uses functions and classes as necessary. All functions include document strings. DRY principles are implemented.

✅ Code is well documented and uses functions and classes as necessary

Congratulations on fulfilling the requirement of having well-documented code.

Your code demonstrates a clear understanding of the importance of documentation. Well-documented code allows others to comprehend the purpose and functionality of each component, including functions and classes. It provides insights into input requirements, expected outputs, and any relevant details that facilitate the effective use and understanding of the codebase.

✅ All functions include document strings.

Excellent job on meeting the requirement of including docstrings for all functions.

The inclusion of docstrings demonstrates your commitment to providing clear explanations of each function's purpose, expected inputs, returned values, and any notable behavior. This documentation serves as a valuable reference for both yourself and other developers who interact with your code, making it easier to comprehend and work with.

✅ DRY principles are implemented.

You have successfully incorporated the DRY (Don't Repeat Yourself) principles in your code. 💯

Through your implementation of the DRY principles, you have effectively eliminated code duplication and reduced redundancy. This practice greatly enhances code maintainability and reduces the chances of introducing errors. By encapsulating common functionality within functions or classes, you promote code reuse, allowing you and others to leverage existing components in multiple parts of the codebase.

Useful References:

Comments, Docstrings, and Type Hints in Python Code [Article]

How to DRY your Python code [Article]

Part I: Data Exploration

Correct values provided for variables (median_val,user_article_interactions,max_views_by_user,max_views,most_viewed_article_id,unique_articles,unique_users,total_articles) verified by correct output for the solution 1 dictionary test(t.sol_1_test(sol_1_dict)) at the end of Part 1

✅ *Explore the data to understand the number of users, articles, and information about the interactions that take place.*

Great Job!

You have successfully completed the dictionary associated with exploring the users, articles and information pertaining to user-article interaction. The inline test has a passing remark.

Through your data exploration efforts, you have successfully identified and extracted relevant information regarding the number of users. This includes determining the **total count of unique users** or any **other user-related metrics** that contribute to understanding the user base.

Furthermore, your examination of the interactions provides a deeper understanding of how users engage with the articles. This could involve **analyzing interaction patterns, calculating interaction frequencies, or identifying trends** in user-article interactions.

By exploring the data in this manner, you demonstrate a keen analytical mindset and an ability to extract meaningful insights from the dataset. This exploration serves as a foundation for subsequent analysis and decision-making processes, empowering you to make informed choices based on the data's characteristics.

Well done on meeting this requirement and employing best practices in data exploration!

Useful references:

Data Exploration in Python using NumPy, Matplotlib and Pandas [Article]

Exploratory Data Analysis Tutorial in Python [Article]

Part II: Create Rank Based Recommendations

Tests will ensure that your functions will correctly pull the top articles. The two functions should pull the top ids and the top names.

✅ Tests will ensure that your functions will correctly pull the top articles. The two functions should pull the top ids and the top names.

Your function passed all of the tests related to pulling the top 5, 10, and 20 articles in the dataset. Good work! 👍

The two functions `get_top_article_ids` and `get_top_articles` have demonstrated their ability to successfully retrieve the top articles based on popularity or interaction frequency. This is a key component in building a `Rank-based` recommendation system.

By leveraging these functions, you have established a foundation for generating recommendations that are driven by the overall popularity of articles. This approach can be particularly useful in scenarios where personalized user preferences are not available or not the primary focus.

Brief notes:

Merits of rank based recommendation system

- **Simplicity**: Rank-based recommendation systems are relatively easy to implement and understand. They rely on basic ranking algorithms and do not require complex user modeling or item profiling.
- **Cold Start Problem Mitigation**: Rank-based systems can be effective in addressing the cold start problem, where there is limited or no information about users' preferences or new items. By recommending popular items, these systems can

provide initial recommendations until more personalized data is available.
- **Scalability**: Rank-based systems are often highly scalable since they primarily rely on item popularity metrics. As a result, they can handle large datasets and high user traffic efficiently.

Demerits of rank based recommendation system

- **Lack of Personalization**: Rank-based systems do not consider individual user preferences, leading to non-personalized recommendations. The same popular items are recommended to all users, regardless of their specific interests or tastes.
- **Limited Diversity**: Rank-based systems tend to recommend popular items, which can result in a lack of diversity in the recommendations. Users may be exposed to the same set of popular items repeatedly, limiting their exposure to a wider range of content.

Example

- Google News: News filtered by trending and most popular news.
- YouTube: Trending videos.

Useful References:

Learning to rank for recommender systems [Research Paper]

The Limits of Popularity-Based Recommendations and the Role of Social Ties [Research Paper]

Part III: Collaborative Filtering

Create a matrix with users on the rows and articles on the columns. There should be a 1 if a user-article interacted with one another and zero otherwise.

✅ Create a matrix with users on the rows and articles on the columns.

You have successfully created a matrix that aligns users on the rows and articles on the columns.

This matrix representation is a crucial step in building a recommendation system that relies on user-item interactions. This matrix structure allows for efficient analysis and computation of various recommendation algorithms, such as Collaborative filtering or `matrix factorization`. It provides a solid foundation for further data processing and recommendation generation.

✅ If a user has interacted with an article, then place a 1 otherwise 0.

You have passed all of the tests associated with user-item matrix.

You have successfully completed the very first step of `collaborative filtering`. By correctly implementing the code, you have generated a user-item matrix that reflects user interactions with articles.

The matrix accurately represents whether a user has interacted with an article or not, assigning a value of 1 for interactions and 0 for non-interactions. This matrix serves as the foundation for collaborative filtering, allowing you to analyze user-item relationships and make personalized recommendations.

With the successful completion of this step, you are now ready to move on to the subsequent steps of collaborative filtering.

Brief Notes:

Here's how the user-item matrix is used in collaborative filtering:

1. **Memory-based collaborative filtering**:
    - User-based collaborative filtering: Similarity between users is calculated based on their interaction patterns (e.g., using cosine similarity or Pearson correlation). Users who have similar preferences are considered as neighbors.
    - Item-based collaborative filtering: Similarity between items is calculated based on user interactions. Items that are frequently interacted with by the same users are considered similar.
    - Predictions are made by aggregating the preferences of similar users or items. For example, for a user-item pair, the ratings of similar users for the item are weighted and combined to predict the rating for that user-item pair.
2. **Model-based collaborative filtering**:
    - Machine learning or matrix factorization techniques are used to create a model that captures the underlying patterns and latent factors in the user-item matrix.
    - The model is trained on the observed interactions to learn the relationships between users and items.
    - Once the model is trained, it can predict the preferences or ratings for user-item pairs that have not been observed in the user-item matrix.

Find similar users needed for user-user collaborative filtering model. Write a function that finds similar users.

✅ Find similar users needed for user-user collaborative filtering model. Write a function that finds similar users.

Great job on implementing the `find_similar_users` function!

It successfully returns similar users as expected, which is an essential step in user-user based collaborative filtering.

The function plays a crucial role in identifying users who share similar tastes in reading articles. By computing the similarity between users based on their article interactions, you have established a powerful mechanism for discovering users with similar tastes and preferences.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Well done on understanding and implementing the backbone of user-user based collaborative filtering! This function sets the foundation for building a robust recommendation system.

Useful References:

Performance Comparison of Different Similarity Measures for Collaborative Filtering Technique [Research Paper]

Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison [Research Paper]

Make recommendations using user-user based collaborative filtering. Complete the functions needed to make recommendations for each user.

✅ Make recommendations using user-user based collaborative filtering.

Nice work implementing the `user_user_recs` function.

The function effectively utilizes user-user collaborative filtering to generate recommendations for a given user. It takes into account the similarity between users based on their article interactions and incorporates the logic of choosing users with the most total article interactions before those with fewer interactions.

✅ Complete the functions needed to make recommendations for each user.

Awesome work 👏

By utilizing supporting functions such as `get_article_names` and `get_user_articles`, the code increases modularity and promotes code reuse. This enhances the maintainability and readability of the overall implementation.

Brief Notes:

Merits of User-User based collaborative filtering

- Easy to Understand and Implement: User-based collaborative filtering is relatively easy to understand and implement compared to other recommendation algorithms. It relies on the similarity between users based on their past interactions, making it intuitive to grasp.
- Serendipity and Diversity: User-based collaborative filtering can provide serendipitous recommendations by identifying users with similar tastes but who have explored different items. It can introduce users to new and diverse items that they may not have discovered otherwise.

Demerits of User-User based collaborative filtering

- Scalability with Large User Base: As the number of users grows, the computational complexity of calculating user similarities increases. This can impact the scalability of user-based collaborative filtering, making it less efficient for systems with a massive user base.
- Cold Start for New Items: Similar to the cold start problem for new users, user-based collaborative filtering faces challenges when new items are introduced. Since there is limited or no historical data for new items, it becomes difficult to identify similar users and provide accurate recommendations.

Useful References:

Hybrid User-Item Based Collaborative Filtering [Research Paper]

Improve your original method of using collaborative filtering to make recommendations by ranking the collaborative filtering results by users who have the most article interactions first and then by articles with the most interactions.

✅ In get_top_sorted_users, sort the neighbors_df by the similarity

The top similar users are retrieved using the dot product. Well done 👏

This approach of using the dot product as a similarity measure allows for efficient computation and provides valuable insights into user-user similarities. A higher dot product indicates a higher degree of similarity between users in terms of the articles they have interacted with.

✅ In get_top_sorted_users, sort the neighbors_df by the number of interactions where highest of each is higher in the dataframe

Great job on sorting the neighbors_df by the number of interactions in descending order!

This ensures that the users with the highest number of interactions are listed first, making the recommendations more reliable. Well done! 👍

✅ In user_user_recs_part2, choose the `users` that have the most total article interactions before choosing those with fewer article interactions.

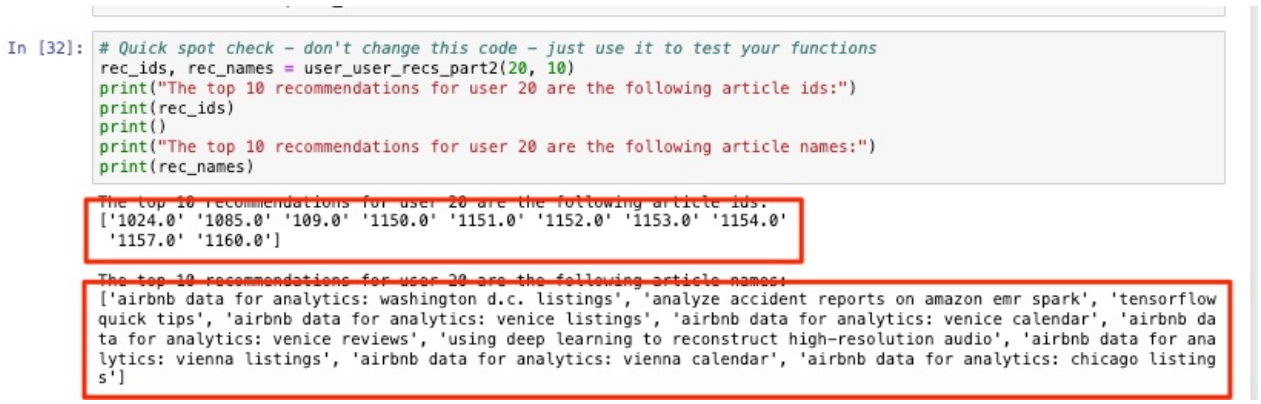
Excellent work on selecting the users with the most total article interactions before those with fewer interactions in the user_user_recs_part2 function!

This approach prioritizes users who have a higher level of engagement and can potentially provide more accurate recommendations. Keep up the good work! 👍

❌ In user_user_recs_part2, choose articles with the articles with the most total interactions before choosing those with fewer total interactions.

You are arbitrarily selecting the articles to add to the final record list of recommendations.

This approach can lead to inconsistent and unreliable recommendations for users. **Let's say we want to return 10 articles to the user and we received 20 articles to recommend from the very first similar user. Which 10 articles will you recommend?** You are arbitrarily selecting 10 articles from the list of recommendations.



```
In [32]: # Quick spot check - don't change this code - just use it to test your functions
         rec_ids, rec_names = user_user_recs_part2(20, 10)
         print("The top 10 recommendations for user 20 are the following article ids:")
         print(rec_ids)
         print()
         print("The top 10 recommendations for user 20 are the following article names:")
         print(rec_names)

The top 10 recommendations for user 20 are the following article ids:
['1024.0' '1085.0' '109.0' '1150.0' '1151.0' '1152.0' '1153.0' '1154.0'
 '1157.0' '1160.0']

The top 10 recommendations for user 20 are the following article names:
['airbnb data for analytics: washington d.c. listings', 'analyze accident reports on amazon emr spark', 'tensorflow
quick tips', 'airbnb data for analytics: venice listings', 'airbnb data for analytics: venice calendar', 'airbnb da
ta for analytics: venice reviews', 'using deep learning to reconstruct high-resolution audio', 'airbnb data for ana
lytics: vienna listings', 'airbnb data for analytics: vienna calendar', 'airbnb data for analytics: chicago listing
s']
```

To address this issue:

It is recommended to incorporate a more systematic approach for selecting the articles to be added to the final list of recommendations. One approach could be to prioritize articles based on the number of interactions they have received from users. By considering the popularity or engagement level of articles, you can ensure that the most relevant and widely appreciated articles are recommended to users.

Here is the expected result:

```
The top 10 recommendations for user 20 are the following article ids:
[1330.0, 1427.0, 1364.0, 1170.0, 1162.0, 1304.0, 1351.0, 1160.0, 1354.0, 1368.0]

The top 10 recommendations for user 20 are the following article names:
['insights from new york car accident reports', 'use xgboost, scikit-learn & ibm watson machine learning apis', 'predicting chu
rn with the spss random tree algorithm', 'apache spark lab, part 1: basic concepts', 'analyze energy consumption in buildings',
'gosales transactions for logistic regression model', 'model bike sharing data with spss', 'analyze accident reports on amazon
emr spark', 'movie recommender system with spark machine learning', 'putting a human face on machine learning']
```

Note: Order does not matter.

You can achieve this by utilizing functions or methods that allow you to sort articles based on their popularity or interaction count, and then selecting the top-rated or most interacted articles to be added to the recommendation list.

Sample Code:

```
# Grouping articles by interaction count
article_interactions = df.groupby('article_id').count()['user_id']

for each top neighbor:
    # List of article id values present in user, but not in input user_id
        new_recs = np.setdiff1d(get_user_articles(neighbor[i]), get_user_articles(user_id), assume
_unique=True)

        # Sorting recommendations based on article popularity/interaction count
        recs_to_add = article_interactions.loc[new_recs].sort_values(ascending=False)

    ...
```

NOTE: THIS IS A SAMPLE CODE. YOU HAVE TO WORK AROUND IT TO MAKE IT WORKABLE.

This will enhance the quality and relevance of the recommendations provided to users, resulting in a more satisfying user experience.

If you encounter any challenges or require guidance during the refactoring process, I encourage you to reach out to the mentors at the Knowledge Hub. They are available to provide assistance and help you overcome any obstacles you may face.

Provide recommendations for new users, which will not be able to receive recommendations using our user-user based collaborative filtering method.

✅ If we were given a new user, which of the above functions would you be able to use to make recommendations? Explain

That's right!

For new users, we should recommend the most popular articles using the `get_top_articles_ids` function. This approach helps address the cold start problem by suggesting articles that have high overall popularity and engagement.

You correctly identified this idea and implemented it by utilizing the `get_top_articles_ids` function to pass the inline test. Good work 👍

✅ Can you think of a better way we might make recommendations? Explain

Awesome!

You proposed a better recommendation system which is a Content-based recommendation system. Content-based recommendation systems are a great approach to address the cold start problem as they do not heavily rely on user data. Instead, they focus on the characteristics and attributes of the items being recommended. By analyzing the content of the items and matching them to user preferences, these systems can provide personalized recommendations even for new users.

✅ Using your existing functions, provide the top 10 recommended articles you would provide for the new user.

You returned the correct set of articles and passed the test. 💯

By utilizing your existing function, you were able to generate the set of articles that are most suitable for a new user. Your recommendation aligns with the expected results, demonstrating your understanding of the system and its functionality.

Useful References:

Solving Cold User problem for Recommendation system using Multi-Armed Bandit [Article]

Part V: Matrix Factorization

Perform SVD on user-item matrix. Provides U, Sigma, and V-transpose matrices, as well as an explanation of why this technique works in this case.

✅ Perform SVD on user-item matrix. Provides U, Sigma, and V-transpose matrices

Great job on performing SVD on the user-item matrix and obtaining the U, Sigma, and V-transpose matrices.

Your implementation successfully utilizes the `numpy.linalg.svd` function to dyuecompose the matrix. This technique is suitable for the given scenario because it allows us to factorize the matrix and capture latent factors or dimensions that contribute to the user-item interactions.

The resulting U matrix represents user characteristics, the Sigma matrix contains the singular values representing the importance of each dimension, and the V-transpose matrix represents item characteristics. Well done on completing this task successfully!

✅ Explain why this is different than in the lesson

You're absolutely correct!

In this case, we can confidently use SVD because the original user-item matrix does not have any missing values. This is a key difference from the user-item matrix used in the lesson, where missing values were present. Since SVD requires a complete matrix, the absence of missing values in the current scenario allows us to directly apply the SVD technique. Your understanding of this distinction is commendable. Well done!

Useful References:

- An introduction to SVD and its widely used applications [Article]
- How Does the Funk SVD Algorithm work in Recommendation Engines? [Article]

Split the user-item matrix into training and testing segments. Identify the users in the test set that are also in the training.

✅ Split the user-item matrix into training and testing segments.

Great job on splitting the user-item matrix into training and testing segments!

This step is crucial in evaluating the performance of recommendation systems. By separating the data into training and testing sets, you ensure that the model is trained on one portion and tested on another, providing an unbiased assessment of its predictive capabilities. Well done on successfully implementing this step!

✅ Identify the users in the test set that are also in the training.

Great job identifying the users in the test set who are also present in the training set.

This step is critical, especially when dealing with the cold start problem. By identifying these common users, you ensure that you have sufficient data to evaluate the performance of your recommendation system for users who have already interacted with items in the training set. This helps address the cold start problem and provides a fair assessment of your system's effectiveness.

Useful References:

Recommender Systems: An Introduction by Dietmar Jannach [Book]

Perform assessment of the predicted vs. the actual values.

✅ Perform assessment of the predicted vs. the actual values.

Your analysis of predicted v/s the actual values is done correctly. 💯

You have created a nice visual of the training and testing accuracy values with varying numbers of latent features. This visual representation enhances the understanding of the model's behavior and helps in identifying the optimal number of latent features for achieving the best accuracy.

Suggestions:

You can additionally calculate precision, recall, f1 score, and observe the distribution of prediction on the training set and testing set, to get some insights into the overfitting issue.

- **F1 Score**: The F1 score is a combination of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall at the same time. The F1 score is useful when we want to find a balance between precision and recall. It is calculated as the harmonic mean of precision and recall.
- **Precision**: Precision tells us how many of the positive predictions made by the model are actually correct. In other words, it measures the accuracy of the model when it predicts something as positive. A high precision means that the model is making fewer false positive errors.
- **Recall**: Recall tells us how many of the actual positive instances in the data were correctly predicted as positive by the model. It measures the model's ability to find all the positive instances. A high recall means that the model is making fewer false negative errors.

Useful References:

Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on? [Article]

Provide a discussion about the results, as well as a method by which you could test how well your recommendation engine is working in practice.

✅ Comment on the results that you found in the previous section.

Nice write-up! 👍

You have provided a comprehensive and thoughtful analysis of the results obtained. Your explanation of the challenges faced in the analysis demonstrates a clear understanding of the problems associated with class imbalance and its impact on the performance of the recommendation engine. Overall, your comment provides valuable insights and showcases your ability to critically evaluate the results.

✅ Discuss a method by which you could test how well your recommendation engine is working in practice.

Indeed!

Incorporating concepts of A/B testing is an excellent approach to assess and enhance the performance of the recommendation engine.

By dividing users into a control group and an experiment group, we can compare the effectiveness of the previous version (control) with the current version (experiment). This allows us to gather valuable opinions and feedback from real users, enabling us to make data-driven improvements.

Analyzing the statistics and user behavior of the control and experiment groups will provide insights into the impact of the updated recommendation engine. Metrics such as click-through rates, conversion rates, user engagement, and satisfaction can be compared to evaluate the effectiveness of the new recommendations.

A well-designed A/B test ensures that any observed differences in user behavior can be attributed to the changes made in the recommendation engine. It provides a reliable method to measure the impact of the system in a real-world setting and guide further iterations for improvement.

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH