UDACITY

< Return to Classroom

Recommendations with IBM

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Well Done. I really liked this submission. You ensured there is no hardcoding in any section of the notebook. You ensured the correct calculation of the total number of interactions based on the original dataframe in the `get_top_sorted_users` function. Awesome work here.

Now that you have completed the project, I think it is the right time to dive deeper into learning about putting such systems into production. The analysis that we performed in this notebook is only part of the equation. At the end of the day, the goal is to integrate this analysis as a feature in our main product. Integrating into the main product takes much more than just cool algorithms. There are a lot of things to consider. We have to think about creating the right system architecture based on the requirements. There is no better place to learn about using recommendation systems in production than Netflix. Netflix personalizes every aspect of a user's home page (even, the thumbnails of shows and movies are personalized according to the user). Please take a look at one of the earliest system architectures they used. You can also go through other articles in the series here.

All the best for your next project !!! Stay safe and keep learning.

Code Functionality & Readability

✓

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

Good Job !!! All files required for the review are submitted.

✓

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

Good Work !!! Although Udacity had supplied docstrings for most of the functions, you can see how these docstrings help us understand its interface's purpose and details. Documenting your code is very important as code is read multiple times over time and can help later understand the code. You can start incorporating these ideas into your projects too. Apart from docstrings, Python now has support for type hints that help make the input and output types clear to the user. You can read more about different ways of documenting your code in this excellent article.

Data Exploration

✓

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

Good Work here !!! However, it would be better to perform a little more visual exploratory analysis. The skeleton code provided is the bare minimum that we need to perform. However, we can explore the data further to enhance our understanding.

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.

Nicely Done !!! We get the top rank-based recommendations correctly.

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 a zero otherwise.

Well Done !!! The user-item matrix has been created successfully. The matrix passes all the tests.

✓

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

Well Done !!! The similarity is calculated correctly with the correct users.

✓

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

Well Done !!! You are adding only those articles as recommendations that are not already read by the user. Also, you used a set data structure that ensures there are no duplicate recommendations.

✓

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.

Nice Work !!! The function `get_top_sorted_users` is implemented correctly.

✓

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

You are correct that there are no interactions for new users in the system and hence, rank-based recommendations are a good choice to start with. This issue is known as the cold-start problem in the industry. Generally, most platforms combat this using some kind of onboarding flow for a new user where they ask the user to select a few items that they have liked or purchased in the past on any platform. Using knowledge-based recommenders or content-based recommenders are also viable options.

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.

Awesome Work !!! We can use the built-in SVD algorithm because there are no missing values. If there were missing values, we would have to use Funk-SVD as used in the lesson.

✓

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

You pass all the tests. The training and test sets are created perfectly.

✓

Perform assessment of the predicted vs. the actual values.

Well Done !!! The test accuracy decreases as the number of latent features increases. This pattern is different from the training set where the accuracy increased as the number of latent features increased.

✓

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.

Nice Work !!! **Accuracy is not the right metric to use here though. If you look at the dataset, most of the entries in the user-item matrix are zeros. More than 99% of the entries are zeros. So, a model which predicts all zeros will also have unusually high accuracy. However, such a model will be utterly useless. It would be better to use another metric here.** The current framework for model evaluation is also not at all robust because there are only 20 common users between the training and test set for which we can make predictions in the test set. All results are based on these 20 users, not a large group to take any conclusive action based on these numbers.

You should expand on A/B testing a little bit. How will you separate the user groups? Will it be based on userIDs, cookies, devices, IP addresses? How long will we run the experiment? What metrics will be tracked during this experiment?

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START