UDACITY

‹ Return to "Data Scientist Nanodegree" in the classroom

Recommendations with IBM

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Keen Learner,
Congratulations on completing this project.🎉
The work shows a good understanding of the underlying concepts covered in the lessons. Keep up the hard work and all the best moving forward.

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.

Well done! All the tests passed.👍

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

Your code is well documented and has good use of DRY principles. Nice work here. Lots of good coding practices.

##Extra Tips
Check out the following:

- Docstrings vs Comments
  Google Style Python Docstrings

  Best of the Best Practices" (BOBP) guide to developing in Python

Data Exploration

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

You have successfully completed the dictionary associated with exploring the user-item matrix. Excellent job!

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.

Excellent implementation of the `get_top_articles` function, passing all the tests related to pulling the top 5, 10, and 20 articles in the dataset.

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.

Nice work setting up the `user-item` matrix. You have passed all of the tests, and you are ready to take on the rest of the collaborative filtering tasks!

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

Well done implementing this section of the code to get similar users, passing all the tests. ✓

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

Excellent job completing `get_article_names()`, `get_user_articles()` and `user_user_recs()` functions for user-user collaborative filtering based 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 job here!
You did very well by using the `get_top_sorted_users` function to get user most similar to user 1 and user 131.
👏

Tips

Here is another approach:

```
user1_most_sim = get_top_sorted_users(1).iloc[0]['neighbour_id']
user131_10th_sim = get_top_sorted_users(131).iloc[9]['neighbour_id']
```

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

Nice job! That's right! For new users, we should recommend using the most popular items using the `get_top_article_ids` function.

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.

Nice try here.

Remark

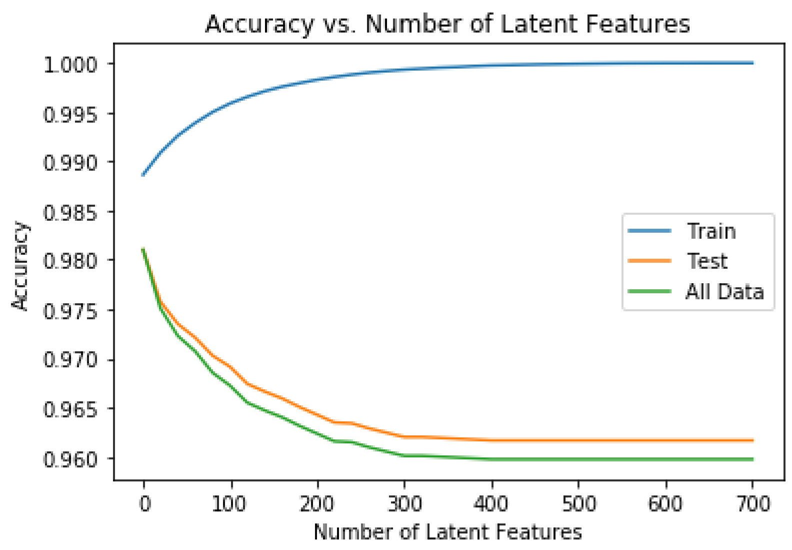
Here, we can use SVD because there aren't any missing values(NaNs) in the original `user-item` matrix. This was not true of our user-item matrix used throughout the lesson, that's why FunkSVD was used. So basically, we use SVD when there are no missing values in our data.

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

Well done! You accurately split the training and testing segments, successfully identifying users that are in both sets. Tests passed!👍

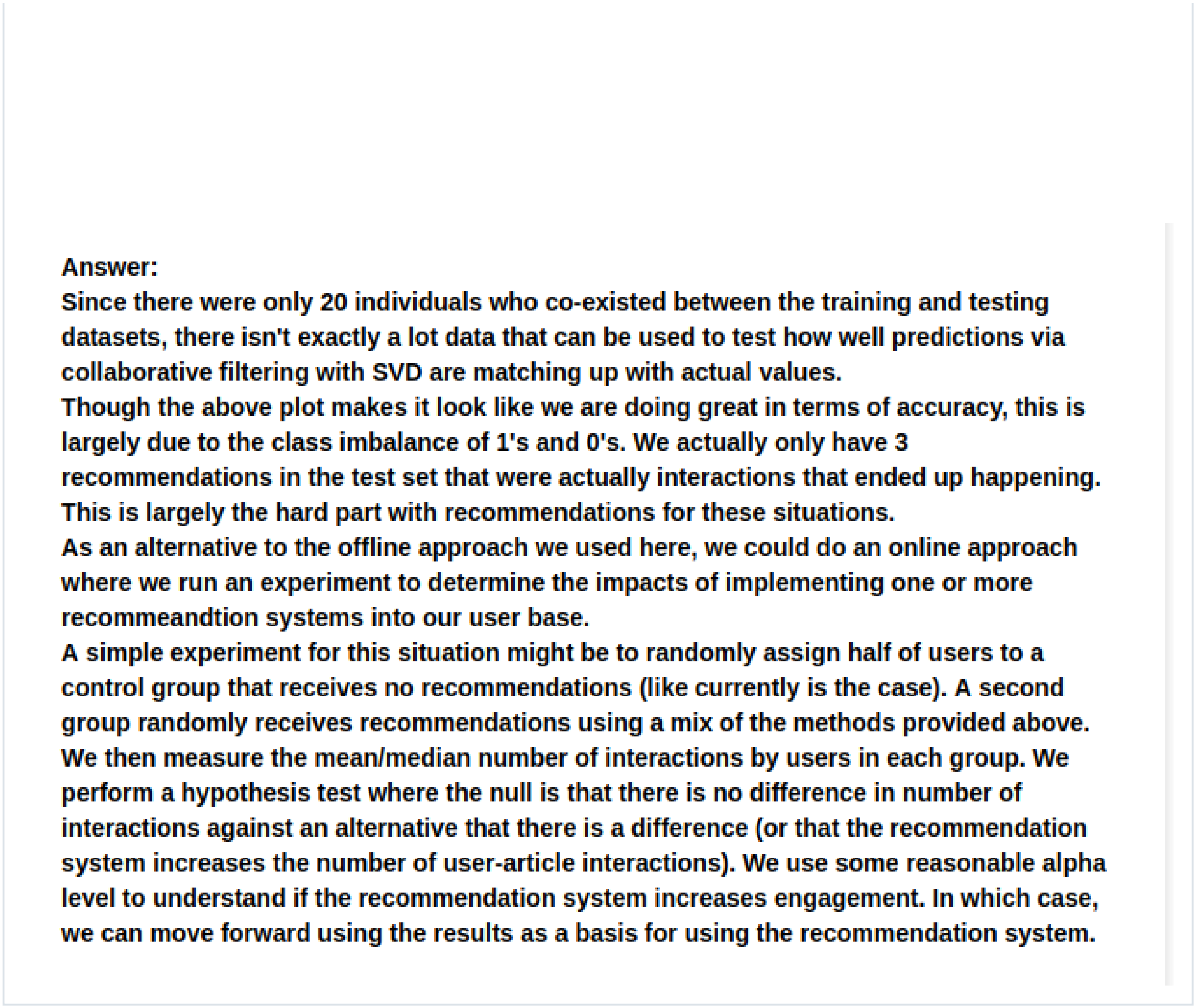
Perform assessment of the predicted vs. the actual values.

Excellent visualization of the accuracies vs latent features.👏



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.

A good and elaborate discussion here. You did well by applying knowledge from A/B testing here. This shows a good understanding of the project. Keep it up!



Answer:
Since there were only 20 individuals who co-existed between the training and testing datasets, there isn't exactly a lot data that can be used to test how well predictions via collaborative filtering with SVD are matching up with actual values.
Though the above plot makes it look like we are doing great in terms of accuracy, this is largely due to the class imbalance of 1's and 0's. We actually only have 3 recommendations in the test set that were actually interactions that ended up happening. This is largely the hard part with recommendations for these situations.
As an alternative to the offline approach we used here, we could do an online approach where we run an experiment to determine the impacts of implementing one or more recommeandtion systems into our user base.
A simple experiment for this situation might be to randomly assign half of users to a control group that receives no recommendations (like currently is the case). A second group randomly receives recommendations using a mix of the methods provided above. We then measure the mean/median number of interactions by users in each group. We perform a hypothesis test where the null is that there is no difference in number of interactions against an alternative that there is a difference (or that the recommendation system increases the number of user-article interactions). We use some reasonable alpha level to understand if the recommendation system increases engagement. In which case, we can move forward using the results as a basis for using the recommendation system.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review