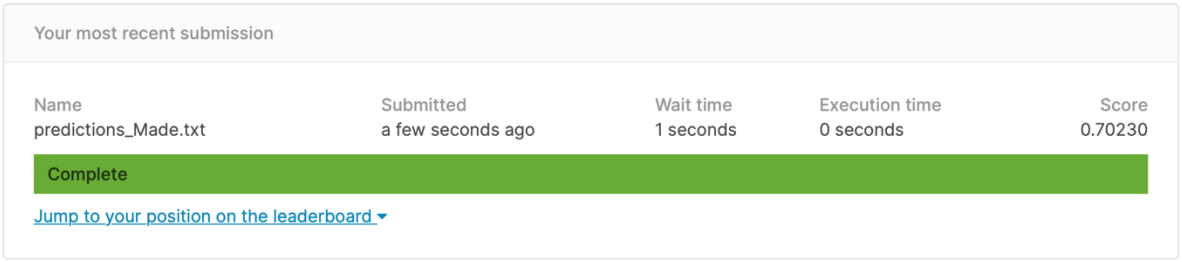
**Cook Prediction Task** Will Powers – A59008589 – UserName: wpowers96



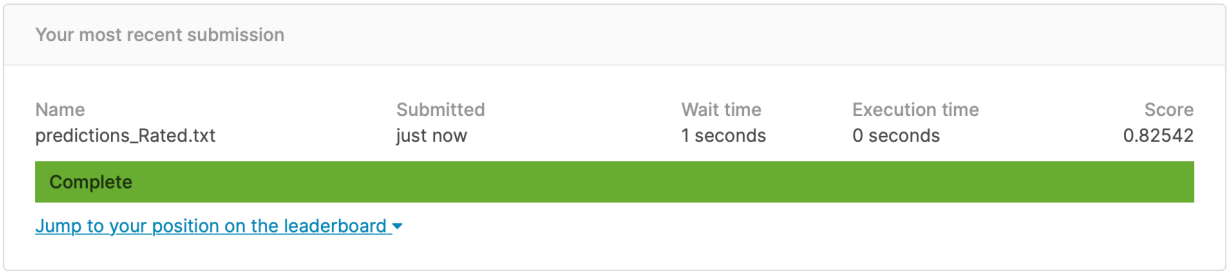

The goal of the cook prediction task was to predict whether a user would make a recipe. After considering many models and attempting ensembling methods, the best model that I could create was an optimized version of the baseline, attaining **accuracy of 0.70230.**

First, I would need to create a training dataset to do the tuning on. I collected the user-rating pairs from the training interactions and for each randomly sampled a recipe that the user had not interacted with and added those to the user-rating pairs. I created a 'didCook' array to indicate the presence of an interaction for each pair. I shuffled both these arrays in tandem. Then I split the first 90% of each into training data and the last 10% for validation.

To predict that a user-item pair would lead to an interaction the model I created would predict so if the item was among the top sequential items that contributed to X% of all interactions. I tuned the value for X and got 64% to optimize the categorization accuracy on the training set. The validation set roughly agreed.

After uploading to Kaggle I continued to tune the value of X on the public dataset to 70%. This felt valid for 2 reasons: despite my tuning on the training set, the value of the validation set agreed. So it was my belief the model accuracy on the private data would be similar to the public one. Further, my training data only used items that had been seen before. It might then be more realistic to real-world conditions tune on the public data.

## Rating Prediction Task -




After tuning an SVD model with surprise for a while, I compromised yet again and decided to optimize certain aspects of the baseline code to achieve the tasks of predicting the rating a user would give a certain recipe.

For a certain user-recipe pair, if the user and recipe had rated/been rated before. We would return a weighted average of the user's average rating and the recipe's average rating. The weights were tuned during training and the optimal weight to give the user average rating is .84 and the weight for the average recipe rating is .16. If only the user had rated before, it's average would be returned. If only the recipe had been weighted before, the recipe average would be returned. If neither had rated/been rated, then the global average rating would be returned.

Test and validation sets were used with ratio of 90/10. Test and validation set data was removed from the data used to calculate averages, otherwise the training models overfit.

Extra-tuning of the average weights on the public data on Kaggle did not improve upon the training optimal weight values. That is a good sign that the test was designed well to simulate randomized data conditions.