**Introduction**

Build two knn models to predict user ratings for anime, and anime recommendation model based on similarity between anime.

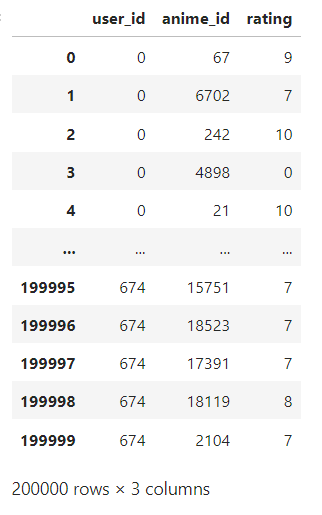
**1. Find, Select Data, and Data Processing (load, explore dataset…): (2 points)**

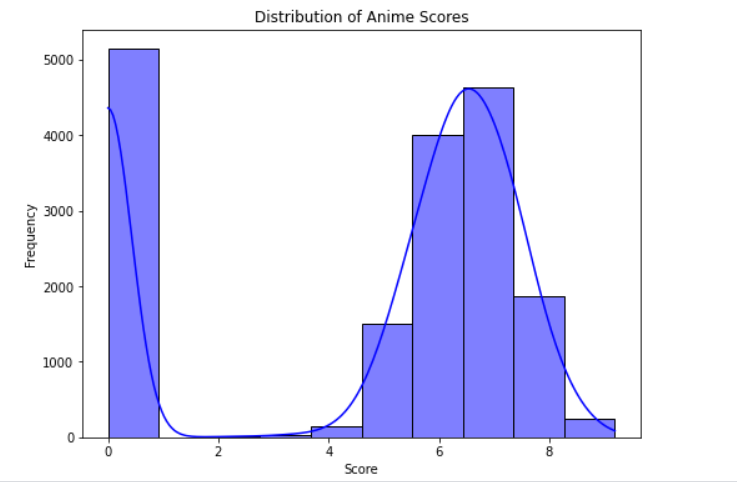
The datasets from [Anime Recommendation Database 2020](https://www.kaggle.com/datasets/hernan4444/anime-recommendation-database-2020/data?select=rating_complete.csv), are anime.csv and animelist.csv.

**animelist.csv** have the list of all animes register by the user with the respective score, watching status and numbers of episodes watched. This dataset contains 109 Million row, 17.562 different animes and 325.772 different users. The file have the following columns:

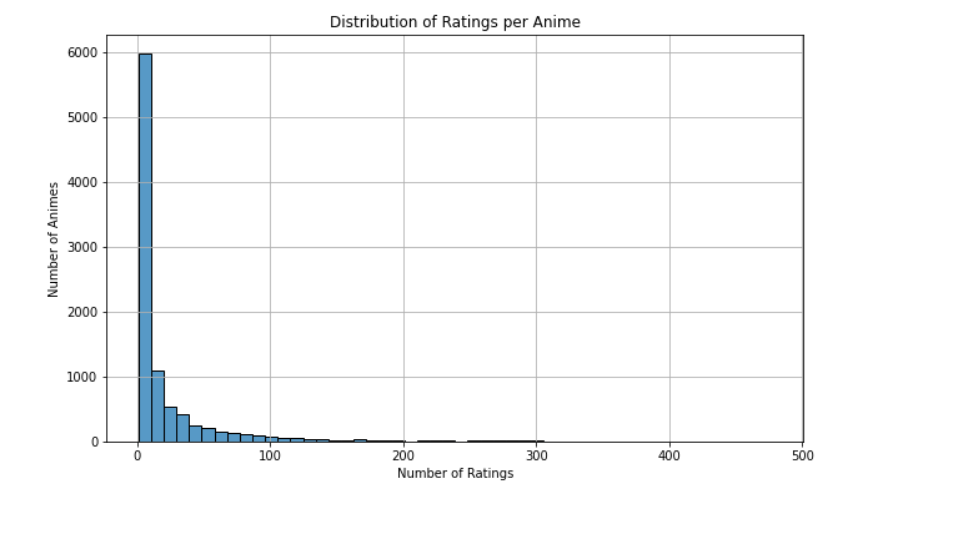
1. user\_id: non identifiable randomly generated user id.
2. anime\_id: MyAnimeList ID of the anime. (e.g. 1).
3. score: score between 1 to 10 given by the user. 0 if the user didn't assign a score. (e.g. 10)
4. watching\_status: state ID from this anime in the anime list of this user. (e.g. 2)
5. watched\_episodes: numbers of episodes watched by the user. (e.g. 24)

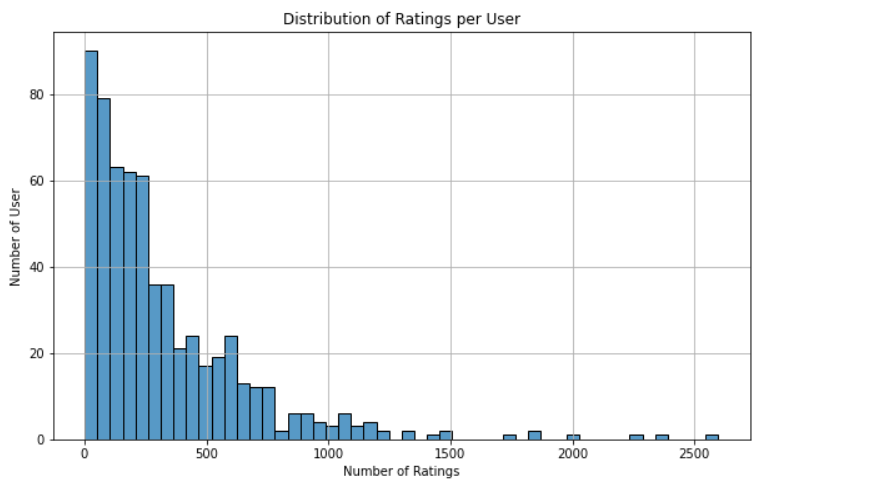
But due to the large number of samples and limited computer resources, we only take 200,000 samples and 3 columns user\_id, anime\_id and rating to train the model.



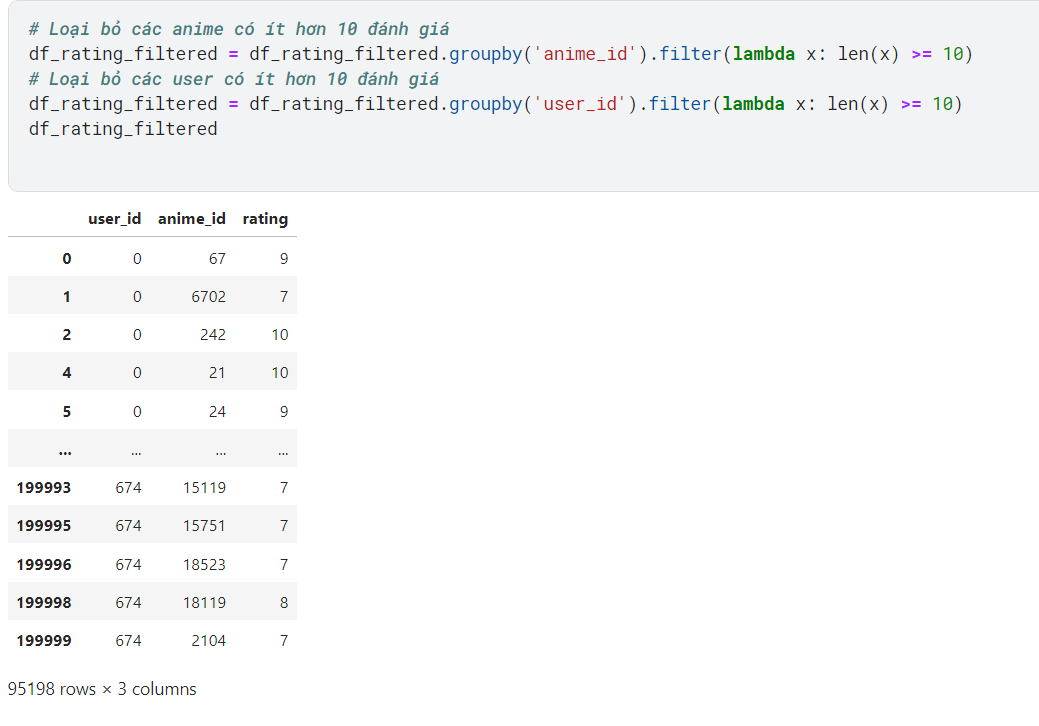
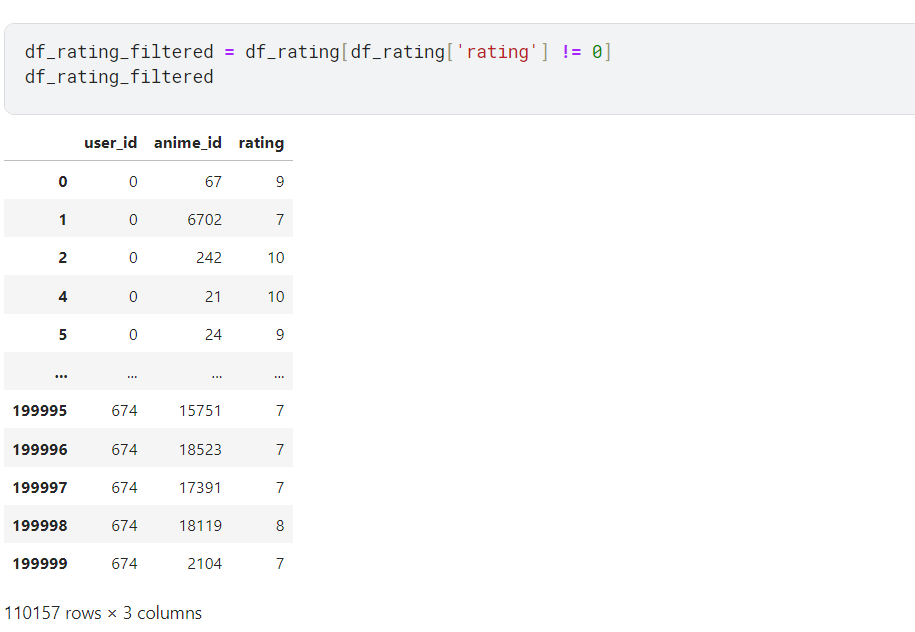


Many samples have 0 points rating, very unobjective, affecting the model.

A lot of anime have few ratings, lack of diversity, affecting the model.

And a lot of users have only a few anime ratings. This data needs to be processed.

Processing steps:



**anime.csv** contain general information of every anime (17.562 different anime) like genre, stats, studio, etc. This file have the following columns:

1. MAL\_ID: MyAnimelist ID of the anime. (e.g. 1)
2. Name: full name of the anime. (e.g. Cowboy Bebop)
3. Score: average score of the anime given from all users in MyAnimelist database. (e.g. 8.78)
4. Genres: comma separated list of genres for this anime. (e.g. Action, Adventure, Comedy, Drama, Sci-Fi, Space)
5. English name: full name in english of the anime. (e.g. Cowboy Bebop)
6. Japanese name: full name in japanses of the anime. (e.g. カウボーイビバップ)
7. Type: TV, movie, OVA, etc. (e.g. TV)
8. Episodes': number of chapters. (e.g. 26)
9. Aired: broadcast date. (e.g. Apr 3, 1998 to Apr 24, 1999)
10. Premiered: season premiere. (e.g. Spring 1998)
11. Producers: comma separated list of produducers (e.g. Bandai Visual)
12. Licensors: comma separated list of licensors (e.g. Funimation, Bandai Entertainment)
13. Studios: comma separated list of studios (e.g. Sunrise)
14. Source: Manga, Light novel, Book, etc. (e.g Original)
15. Duration: duration of the anime per episode (e.g 24 min. per ep.)
16. Rating: age rate (e.g. R - 17+ (violence & profanity))
17. Ranked: position based in the score. (e.g 28)
18. Popularity: position based in the the number of users who have added the anime to their list. (e.g 39)
19. Members: number of community members that are in this anime's "group". (e.g. 1251960)
20. Favorites: number of users who have the anime as "favorites". (e.g. 61,971)
21. Watching: number of users who are watching the anime. (e.g. 105808)
22. Completed: number of users who have complete the anime. (e.g. 718161)
23. On-Hold: number of users who have the anime on Hold. (e.g. 71513)
24. Dropped: number of users who have dropped the anime. (e.g. 26678)
25. Plan to Watch': number of users who plan to watch the anime. (e.g. 329800)
26. Score-10': number of users who scored 10. (e.g. 229170)
27. Score-9': number of users who scored 9. (e.g. 182126)
28. Score-8': number of users who scored 8. (e.g. 131625)
29. Score-7': number of users who scored 7. (e.g. 62330)
30. Score-6': number of users who scored 6. (e.g. 20688)
31. Score-5': number of users who scored 5. (e.g. 8904)
32. Score-4': number of users who scored 4. (e.g. 3184)
33. Score-3': number of users who scored 3. (e.g. 1357)
34. Score-2': number of users who scored 2. (e.g. 741)
35. Score-1': number of users who scored 1. (e.g. 1580)

But we only need to take the following columns to train the model.



Processing steps.

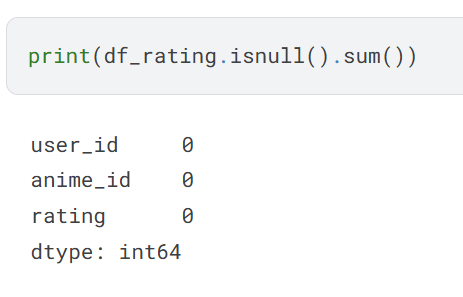


#For example, if a value in the “Genres” column is initially "Action,Adventure,Unknown", after applying the process\_multilabel function, this value will become ["Action", "Adventure"]. [] is not null.

**2. Implementation: Use Surprise library: (4 points)**

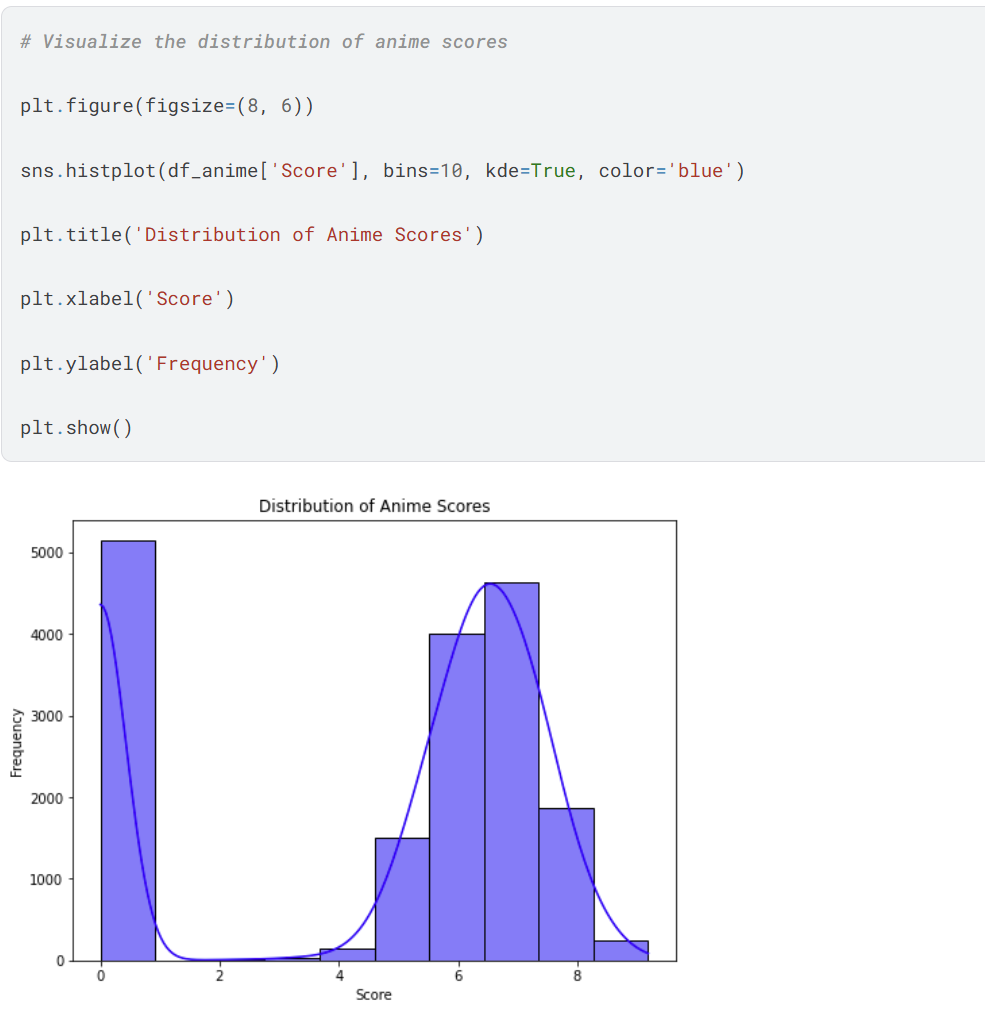
**Recommend Rating**

# Prints out the number of missing values ​​for each column in the DataFrame

****

This result shows that the df\_rating DataFrame has absolutely no missing values ​​in any column, meaning your data is complete and ready for analytics or machine learning models without having to process the values lack.

# Visualize the distribution of anime scores:

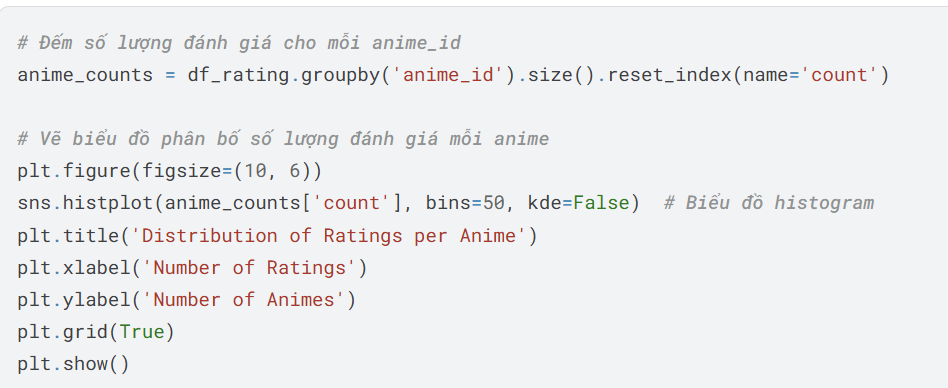


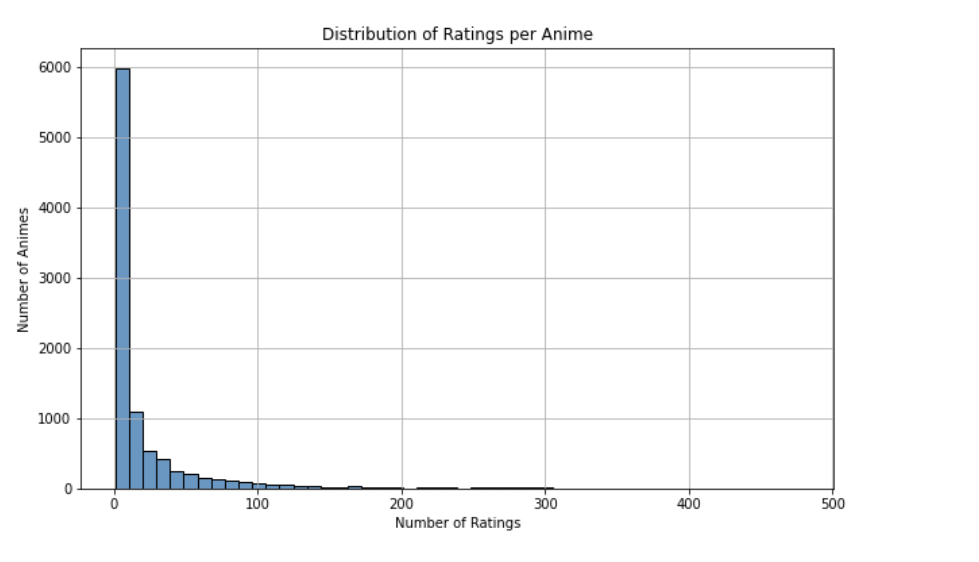
Score 0: There is a peak at score 0, indicating a high frequency of the lowest scores.

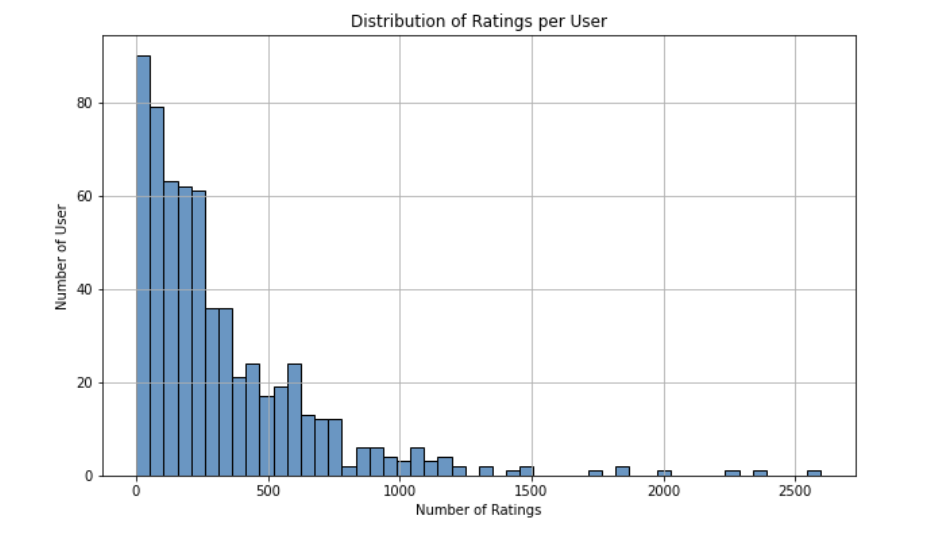
Score range from 5 to 8: There is a significant concentration of scores in this range, with another peak around the score of 6, indicating that many anime receive average scores.

Fewer anime receive scores higher or lower than this: The number of anime decreases as the score approaches 10 or falls below 5.

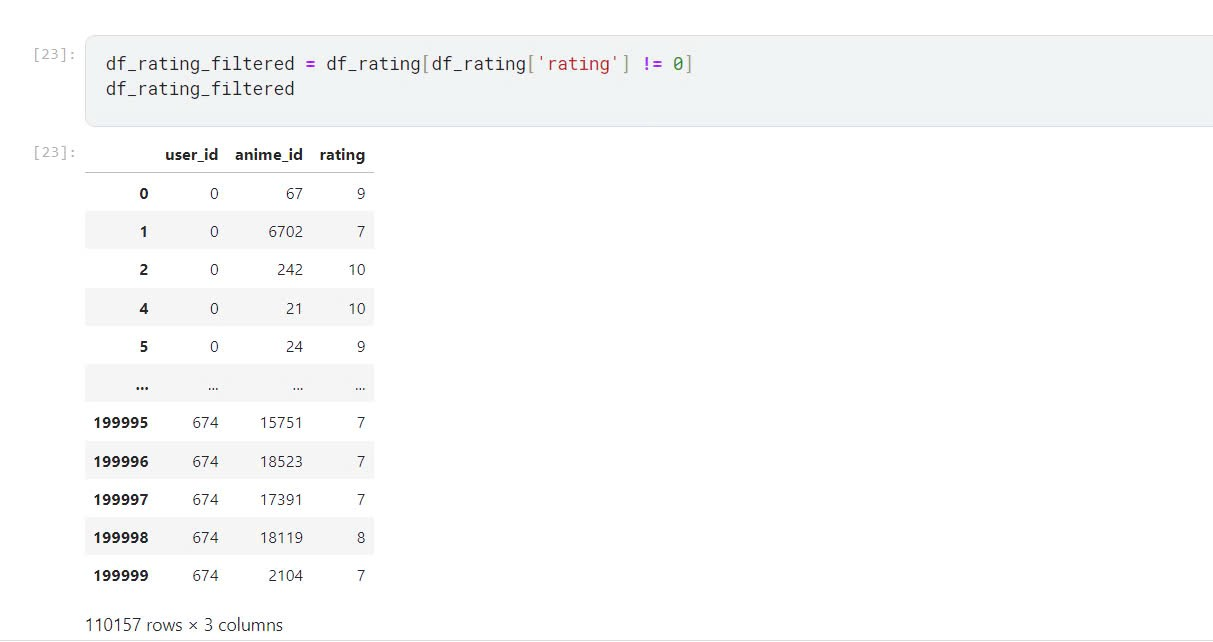
# Count the number of ratings for each anime

****

****

****

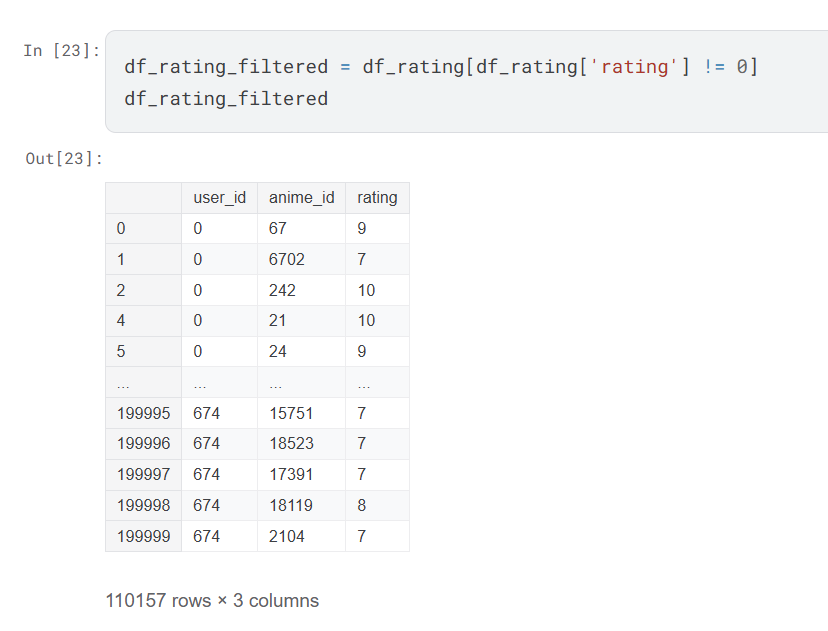
There are 2 charts, 1 for user\_id, 1 for anime\_id, we can see that there are many anime series with few ratings, and also many users with only a few ratings.



Remove rating =0, number of items is reduced by half.

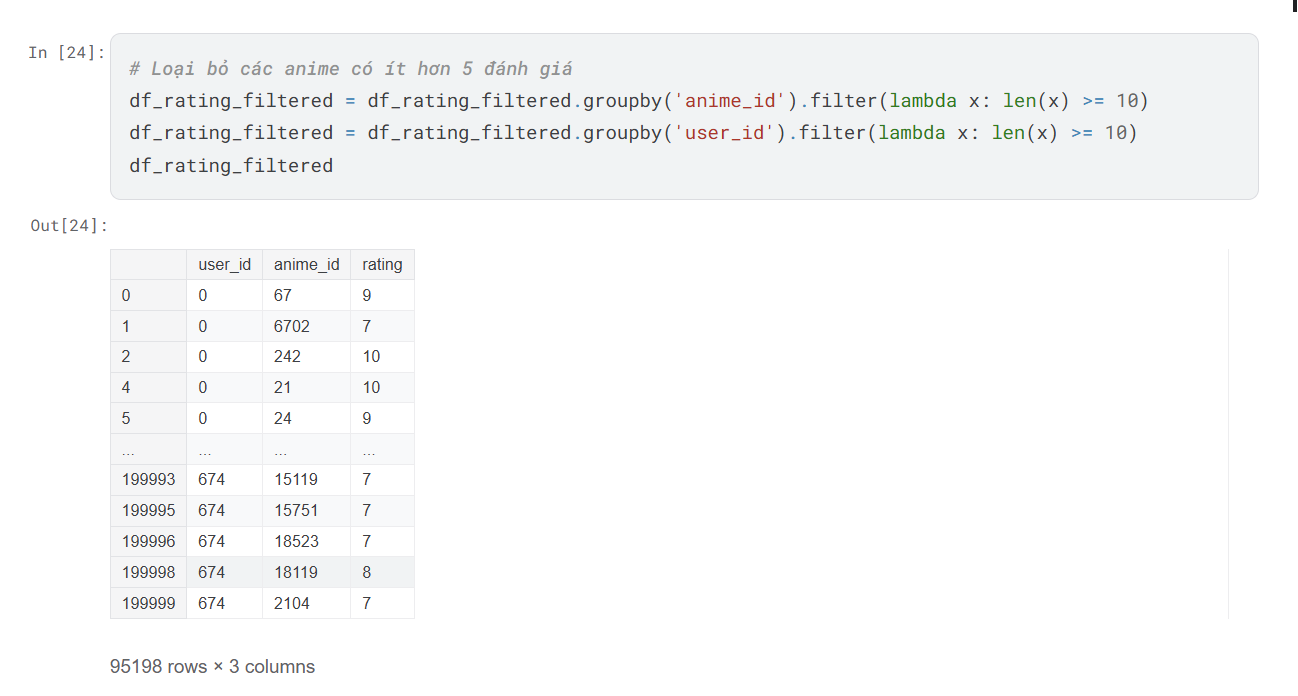


Number of goods continues to decrease.

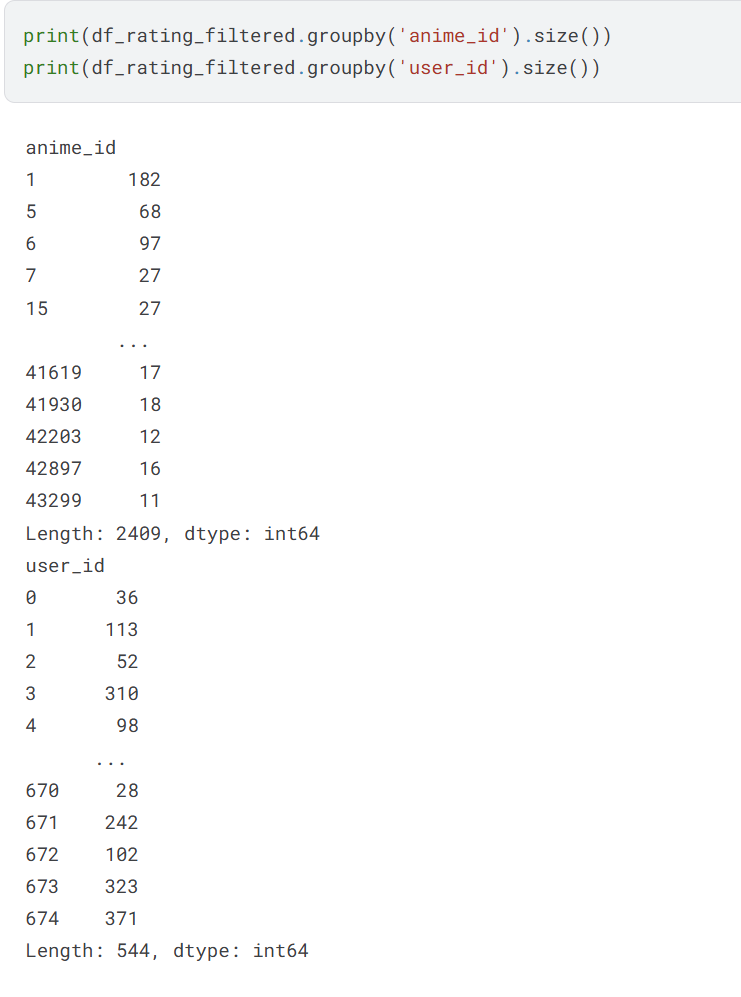
# Remove reviews with a value of 0.  


This is useful in scenarios where you want to analyze or work with actual ratings while excluding those that are not meaningful (like 0), which might be due to users not having rated or providing invalid ratings.

# Filter a DataFrame named df\_rating\_filtered to exclude anime with less than 10 ratings and users with less than 10 ratings



# Performs grouping and counting of reviews for each anime and each user in the DataFrame



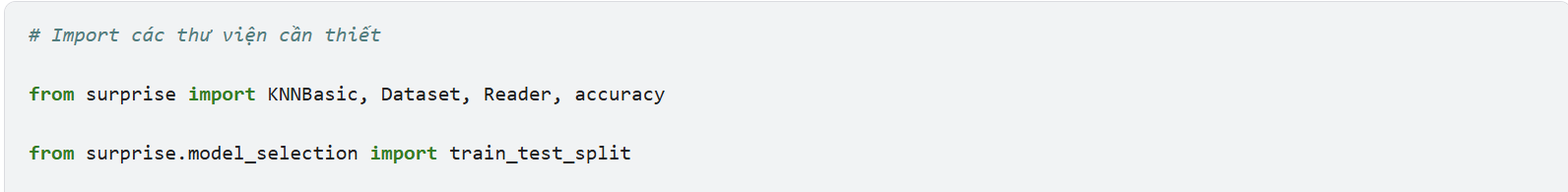
**Suggested steps:**

*Surprise* is a Python sci-kit library for recommender systems. It is simple and comprehensive to build and test different recommendation algorithms.

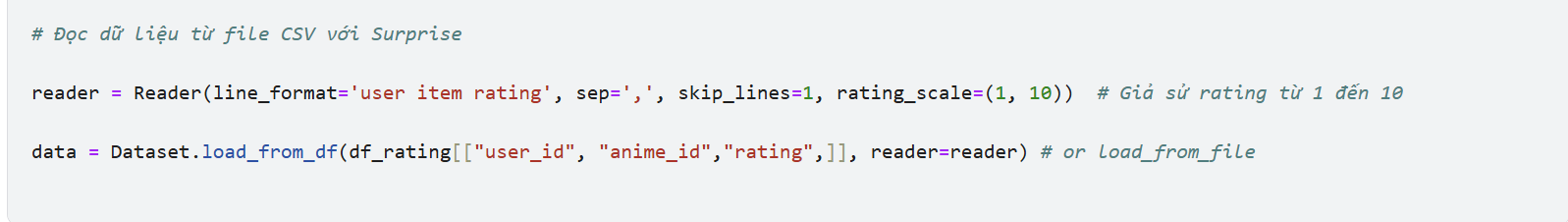
First, let's install it:

****

Now we import required classes and methods

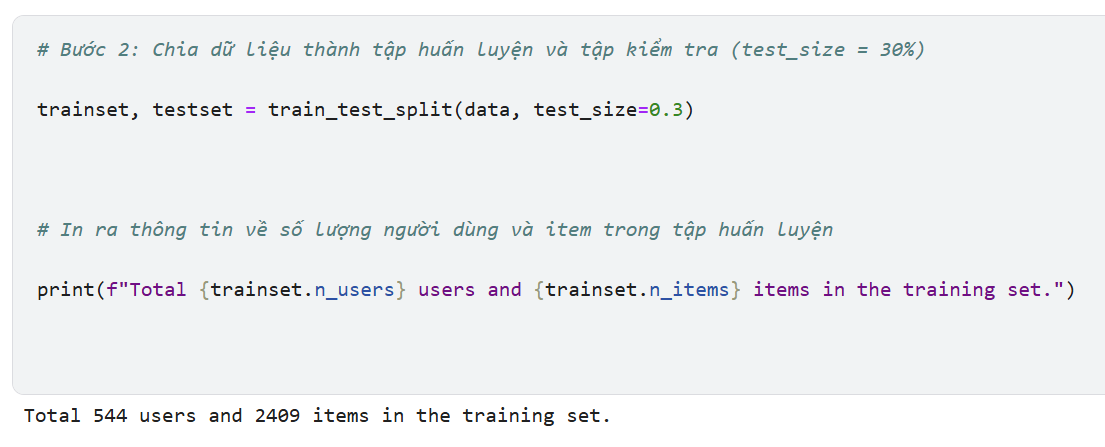
****

*# Read the course rating dataset with columns user item rating*



We split it into trainset and testset:

then check how many users and items we can use to fit a KNN model:

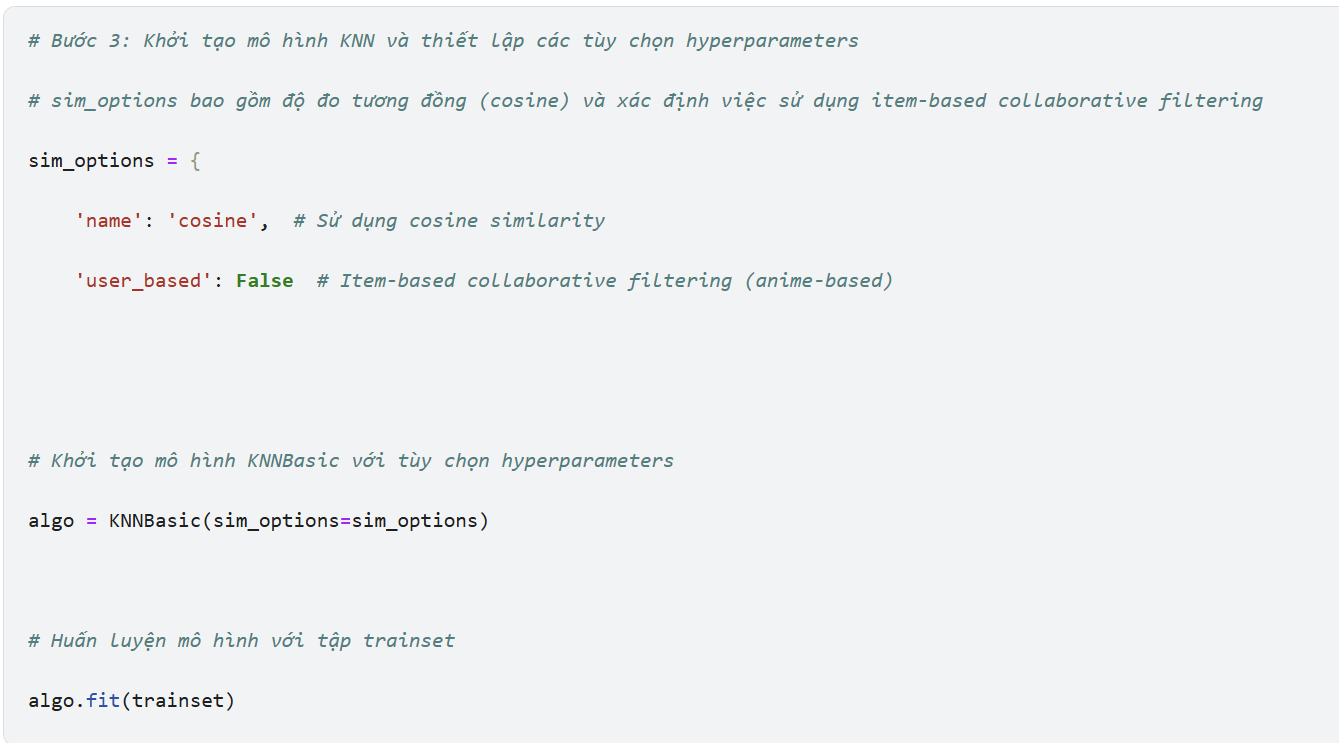


**TASK: Perform KNN-based collaborative filtering on the user-item interaction matrix**

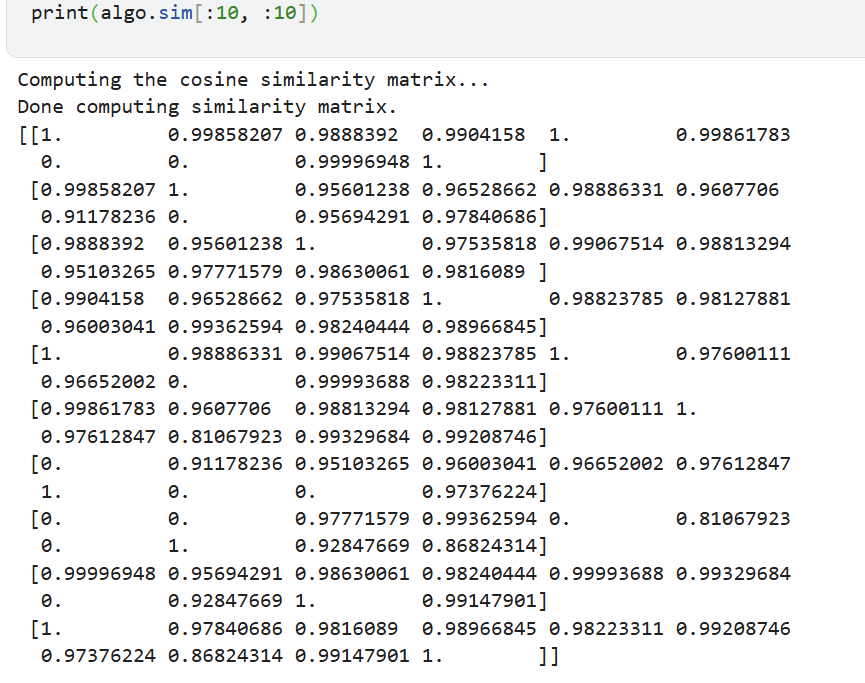
*TODO: Fit the KNN-based collaborative filtering model using the trainset and evaluate the results using the testset:*

*## WRITE YOUR CODE HERE:*

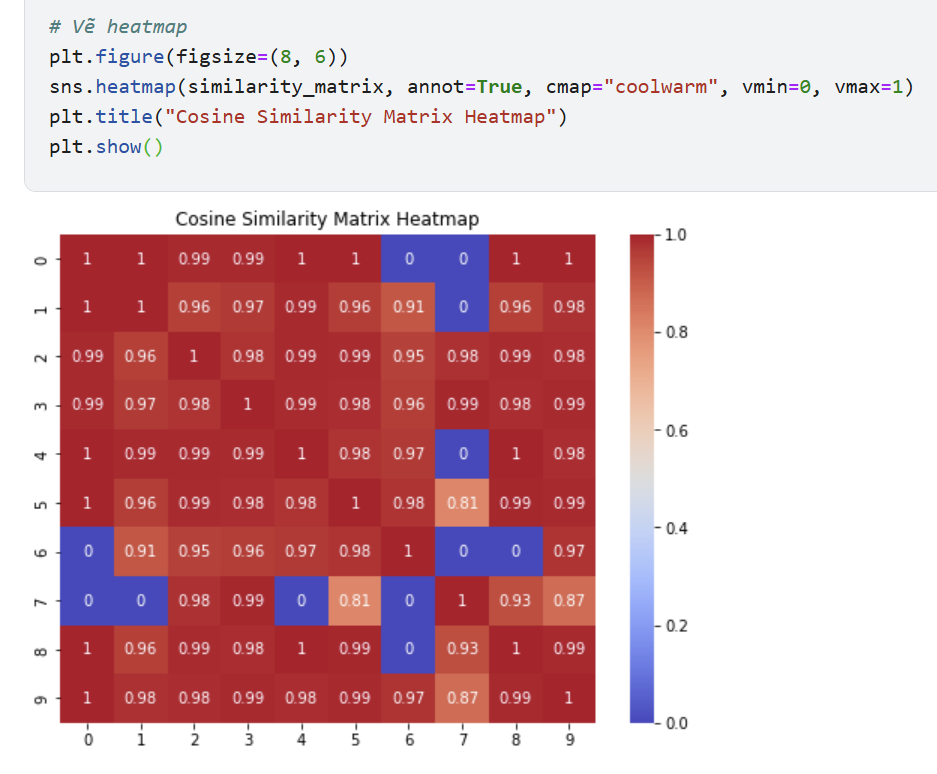
*# - Define a KNNBasic() model*

**

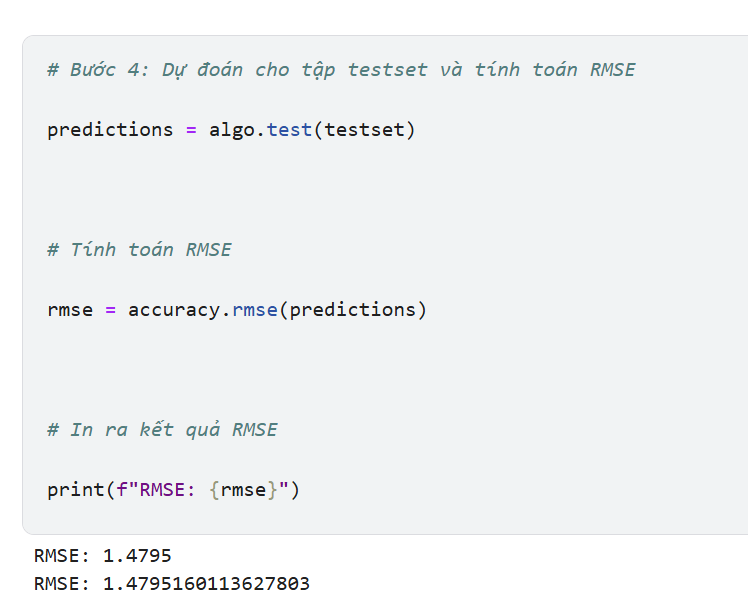
*# Create a similarity matrix*

**

*# Create a heatmap chart*

**

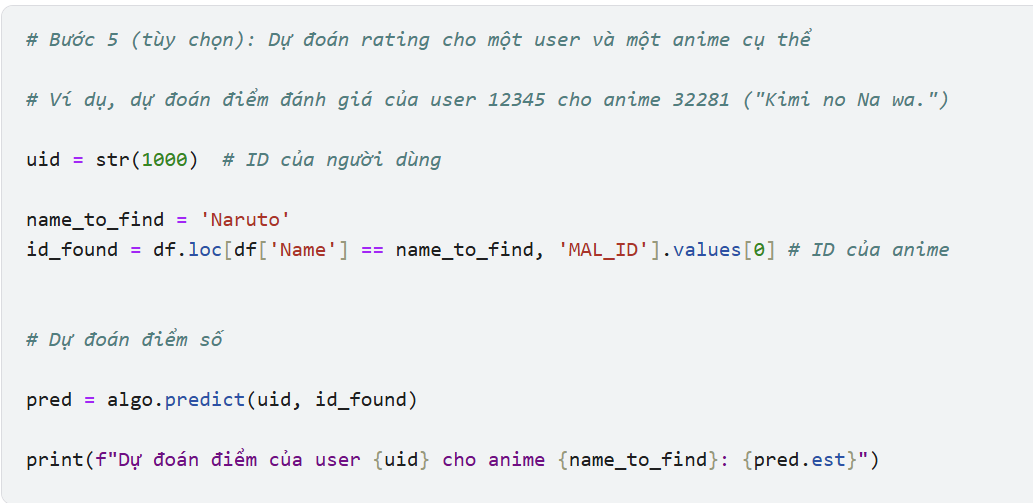
*# - Then compute RMSE*

**

To learn more detailed usages about *Surprise* library, visit its website from [here](https://surprise.readthedocs.io/en/stable/getting_started.html?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMML321ENSkillsNetwork32585014-2022-01-01)

# Predict ratings for a specific user and a specific anime:

# For example, predict user 12345's rating for anime 32281 ("Kimi no Na wa.")



# Save the KNN Surprise model:



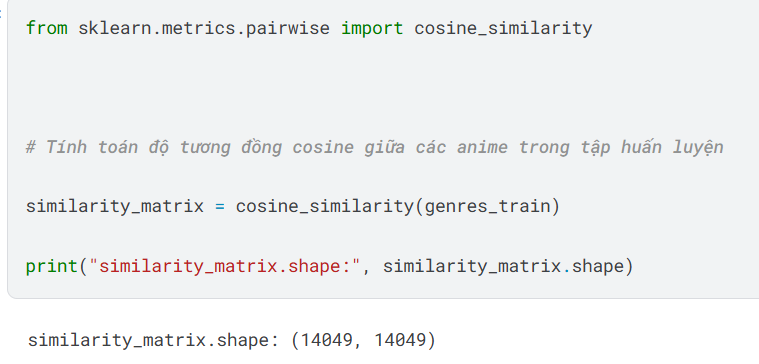
**3. Implementation: Use numpy, pandas, and sklearn. (3 points)**

**Recommend Anime**

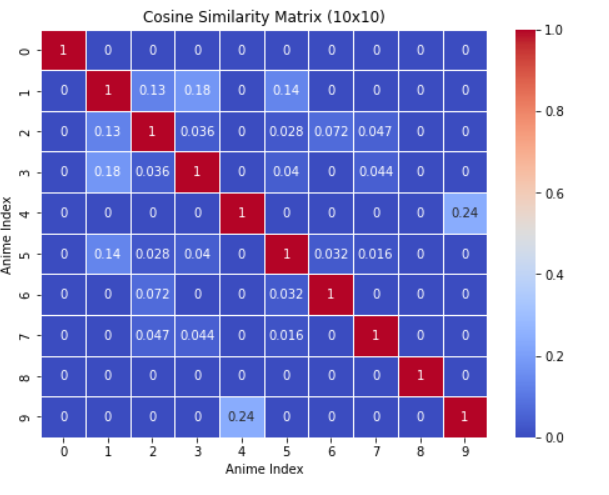
#Use the anime\_metadata dataset as data for the model. Split it into 2 sets, the train set and the test set based on the genre column because the genre column has similarities with each other, for example, many anime have the same genre such as action, horror, romance... And the test set is 20%, the train set is 80%

**

## - Calculate the similarity between two users using their rating history (the row vectors of the interaction matrix)

**

## - Build a similarity matrix for each pair of users with the training dataset

**

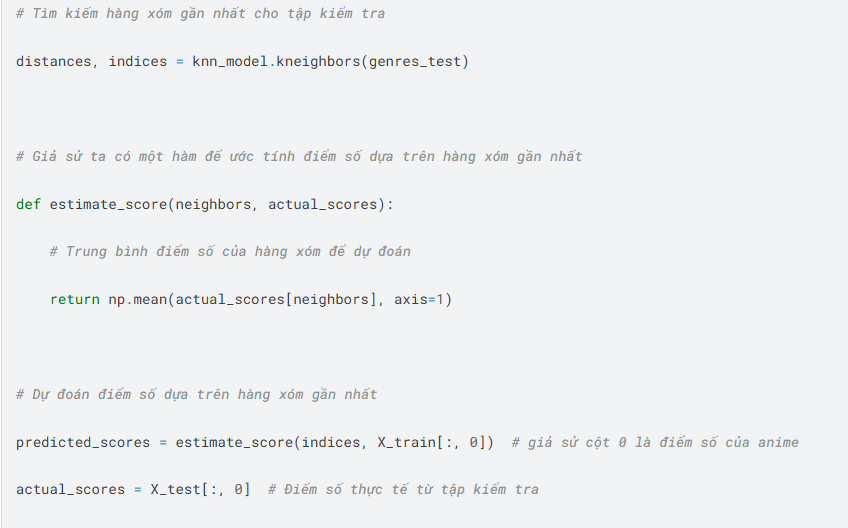
## - For each user, find its k nearest neighbors in the sim matrix

use KNN to find the closest anime based on the similarity matrix, here I give k = 5, which means the 5 closest neighbor anime for each test episode based on cosine similarity and then take this genres\_train episode to train it to learn which are the closest neighbors in the vector space

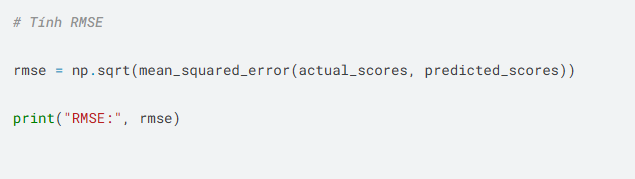
**

## - For each rating in the test dataset, estimate its rating using the KNN collaborative filtering equations shown before

Next is to find the nearest neighbors for the test set, indices[i] will return the indices of the k nearest animes in the training set, which can be used to look up information about those animes (name, genre, rating, etc.) in the training set. distances[i] will indicate the similarity between anime i in the test set and its nearest neighbors in the training set.

**

## - Calculate RMSE for the entire test dataset





## - Visualize data

## Points that deviate far from the red line:

- Many green points are far from the red line, especially the set of points with Actual Scores = 0 but have many different predicted values.

This shows that the model predicts low scores for some anime while their actual scores may be different.

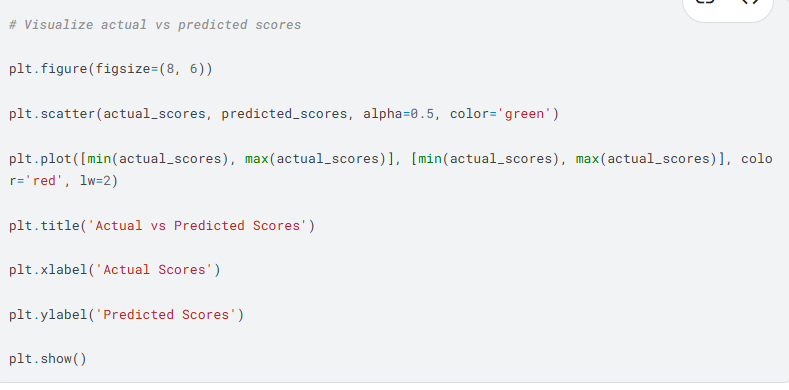
#Highly scattered points:

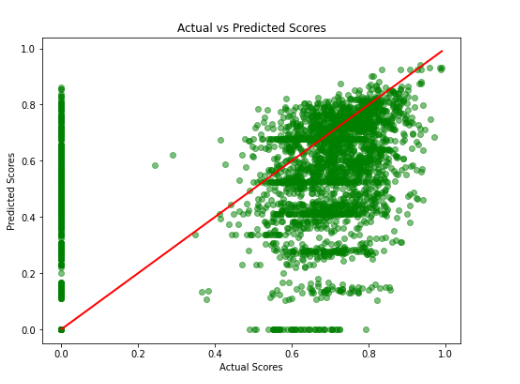
- The green points are widely scattered, especially in the range of actual scores from 0.4 to 0.8, indicating that the model does not predict completely accurately.

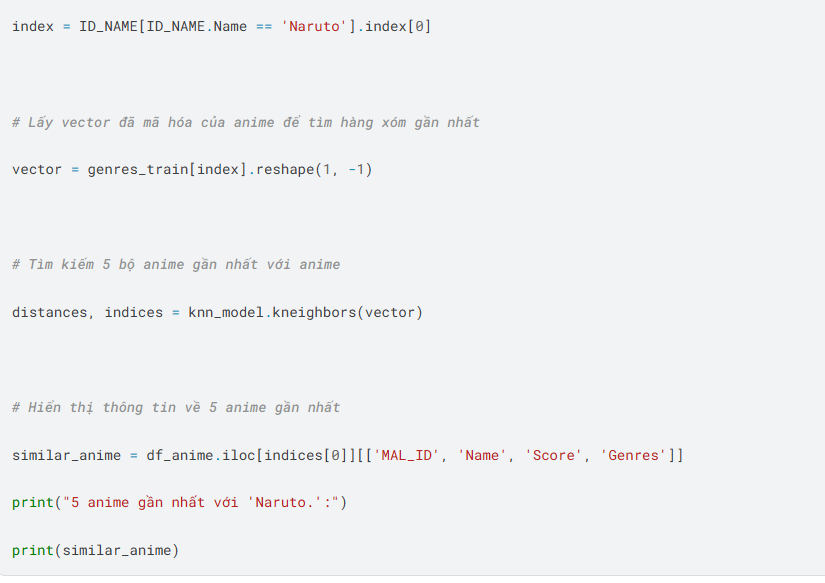
However, some points are still close to the baseline, indicating that there are correct predictions.

#Overall trend:

- The overall trend shows that as the actual scores increase, the predicted scores also increase, although the accuracy is not high.

**

**

****

**4. Code interview (1 point)**