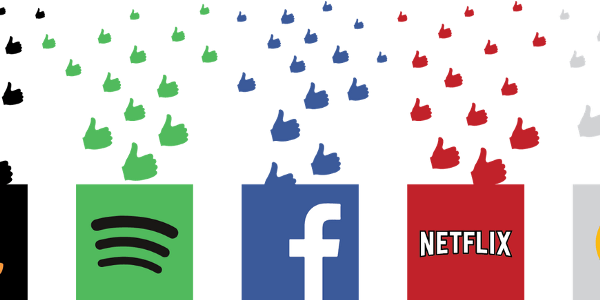
### **Building a Collaborative-Filtering Recommendation system for recommending classes to students**

**In this article, we will build a collaborative-filtering recommendation system that will use the ratings from similar users recommend them, various classes.**

A Recommendation System, in simpler terms, is basically a filtering system that aims to predict a rating or preference that a user would give to a particular item, which in our case are classes.

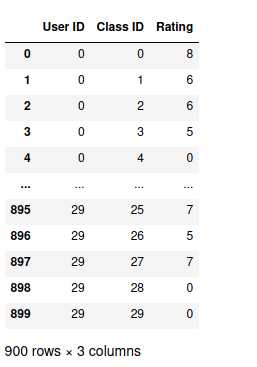
There are various types of algorithms used in building Recommendation systems such as Rank Based, Content-Based, Collaborative Filtering, Matrix Factorization, and many more. Content-Based and Collaborative Filtering are the most common ones among these. In this article, we will look into collaborative filtering to recommend classes to the students.

# Exploratory Data Analysis

First, let's create a dummy dataset.

Text

Description automatically generated

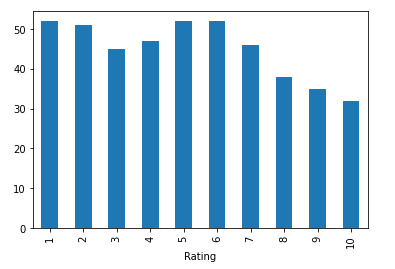


Let's first start by conducting EDA on our data to get a better understanding of what we are dealing with.

Here we have two separate dataframes. The first dataframe contains the rating a user gave to a class. The rating varies from 1–10 and 0 shows that the user hasn't attended that class. The other dataframe just contains the class ids that we have. We have 30 users and classes, thus a total of 900 rows in the dataframe.

Let's group the data by ratings and see the average ratings that the users give. We will use the following data for that.





# User-Based Collaborative Filtering

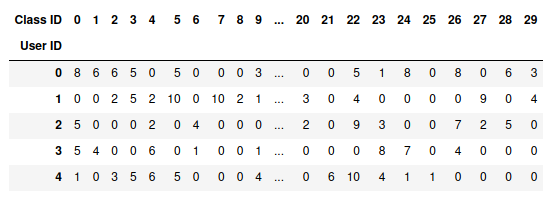
## There are different types of collaborative filtering algorithms, mainly item-based and user-based. In this article, we are going to be using user-based collaborative filtering.

## User-Based Collaborative filtering basically utilizes the assumption that generally, people with similar characteristics share similar interests. In our case, it would mean that if two students have highly rated similar classes in the past, we can recommend them classes that are highly rated by one student to the other as it makes sense that they will highly rate those classes in the future as well.

## Creating User Item Matrix

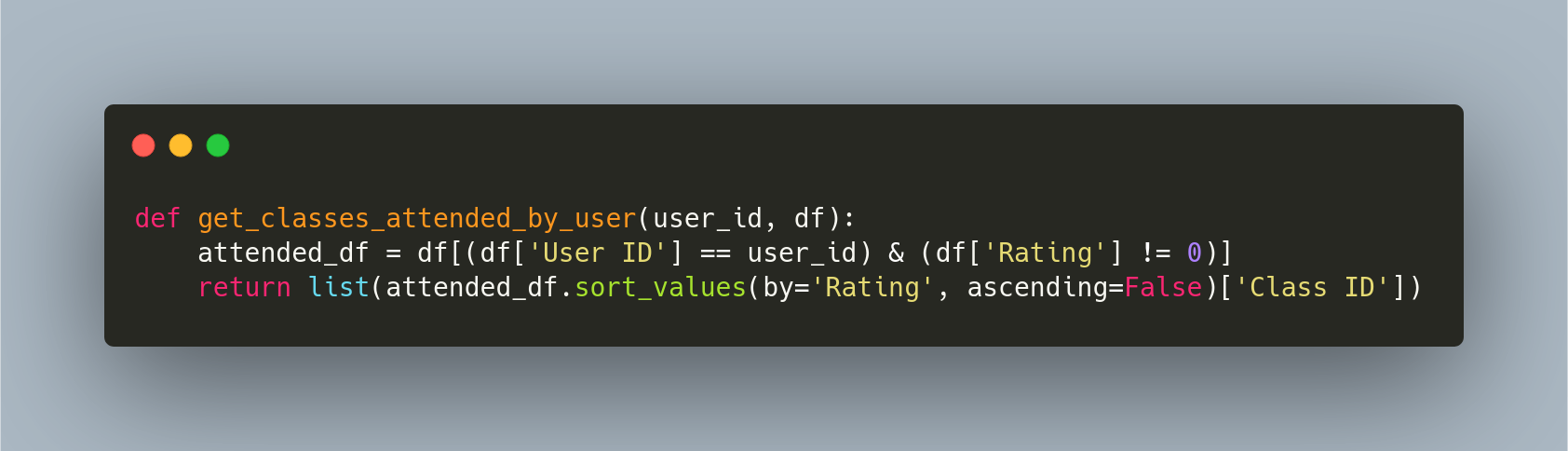
For this, we will first create a user-item matrix (in our case student-class) where the rows and columns will be the student and class ids respectively. If a student has given a class a rating, we will place **‘the rating’** there and **‘0’** if they have not.





## Get Classes attended by Users

Now, we will write a function that will return the classes attended by a user and sort them by the ratings. We do this because after getting similar users, we will recommend the classes attended by the most similar user with the highest rating to that particular user.



## Getting Similar Users

Now we will work on getting users similar to a particular user. This is done by comparing the ratings given by the two users to a class attended by both of them. We will calculate the total ratings and return the users with the highest total ratings.



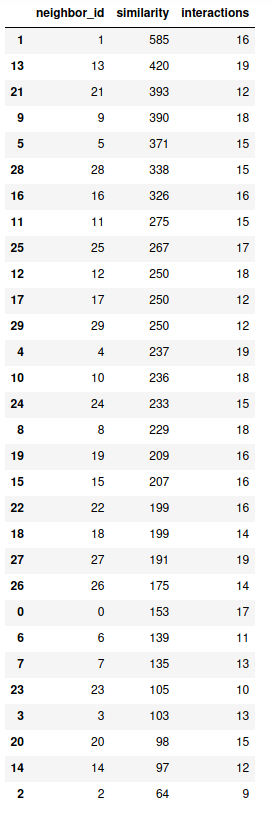
There’s a lot going on in this function. Let’s break it down a bit:

1. It starts with computing the dot product of the user with user\_id with all other users. A higher dot product means that the users gave similar high ratings to the same classes.
2. user\_interactions contain data regarding the number of classes each user has attended.
3. We merge the similar\_users with user\_interactions and sort it first by the similarity and if it's the same, then use the one with more interactions.

This is what we get when we run this function for user\_id 1:

Graphical user interface, application

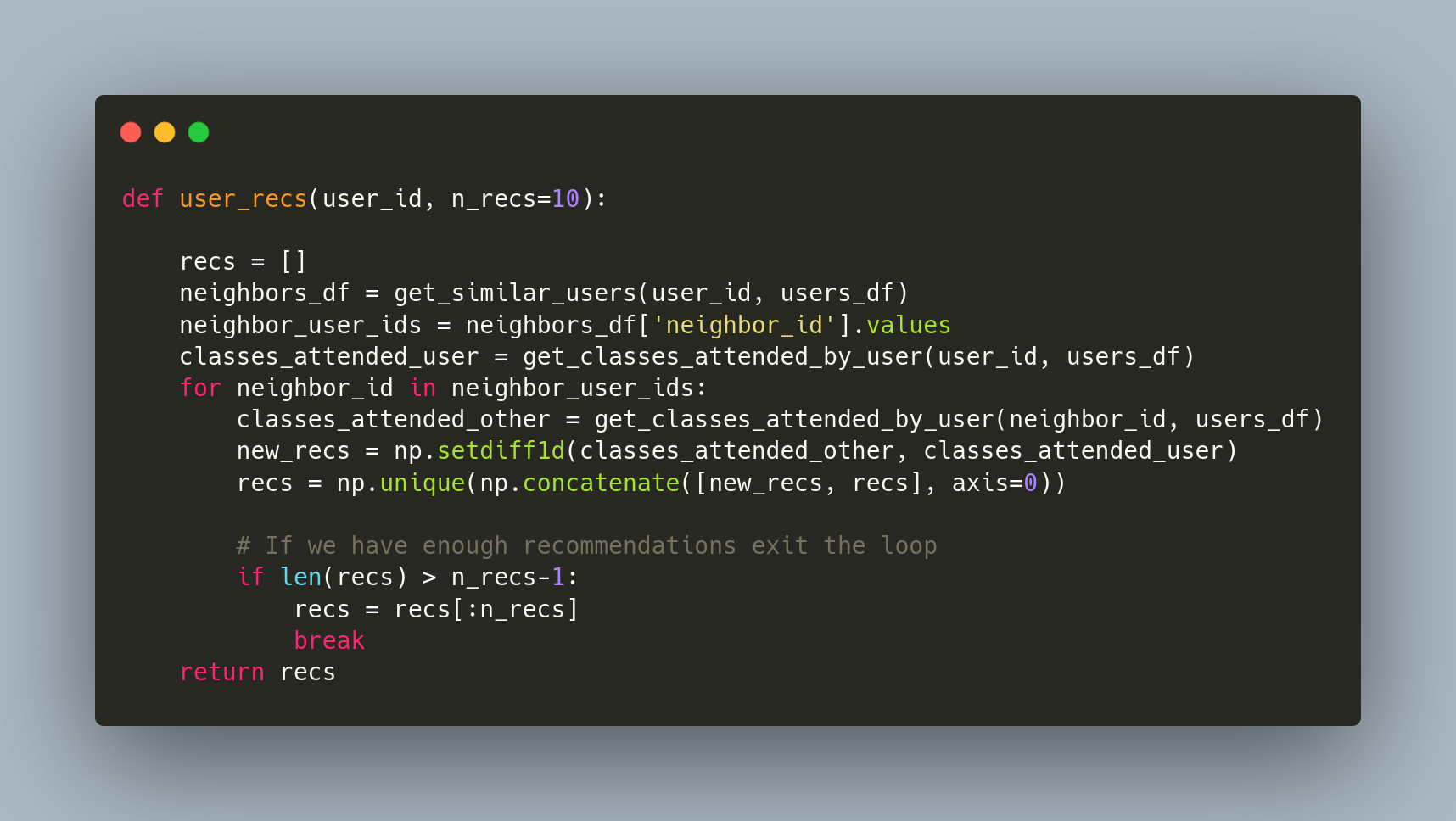
Description automatically generated



## 

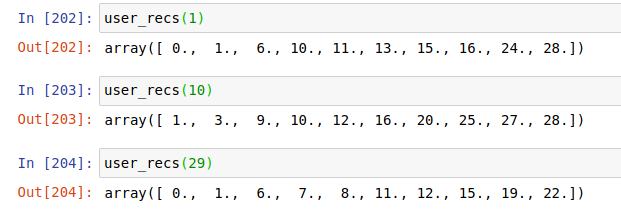
## Providing Recommendations

Finally, we will combine all of this to recommend classes to users.



The function executes as follows:

1. We get the most similar users based on ratings on classes using the function get\_similar\_users
2. We get the classes attended by the current user.
3. We iterate through the ids of similar users and append the classes not yet attended by the current user to the total recommendations.
4. If we reach the number of recommendations we want, we exit the loop

Let’s test this function:

As we can see, the function successfully recommends the top 10 classes the user is most likely to give a higher rating to.

# Conclusion

In this article, we looked into collaborative-filtering based recommendation systems:

1. We saw the importance of creating a user-item matrix and using the dot product of ratings as a means to calculate the similarity between different users
2. We built a recommendation system using the information we had regarding the ratings users give to various classes they had attended
3. We calculated the similar users and recommended the highest-rated classes not yet attended by the current user.