**Group Members:**

1. **Pavan Jignesh Pandya (SU ID: - 340197894)**
2. **Bruno De Paula Luiz (SU ID: - 326264210)**

**Final Project: Gaming Recommendation System**

**Introduction:**

The main purpose of building this project was to make a good recommendation system for the relatable games a gamer/user should play. Everyone has their perspective on different games. In general, most people like only specific games, and they play the same game recurringly because they can’t find a different game with a similar environment. So, the idea of this project is to build a recommender system that helps users find different games with a similar environment to which they are used.

We built the recommender system 2 times for this project. In the first attempt I made a simple recommendation system with the help of my teammate Bruno. We basically applied the basics which we learned in the class. I have Mainly learned about 2 types of classical recommendation algorithms. They were as follows:

1. Collaborative filtering

* The Collaborative filtering is a process of selecting information or patterns using techniques involving collaboration among multiple agents, viewpoints, data source, etc.
* In Collaborative filtering, we don’t need to have additional information about the users or content of items

1. Content-based filtering

* In Content-based filtering, we need to find the similarity between the user and all of the existing items is the core of this type of recommender systems.

While Searching on the Internet I also found a new way to incorporate the recommendation system, that was the Hybrid recommendation system

1. Hybrid Recommendation Systems

In hybrid recommendation systems, products are recommended using both content-based and collaborative filtering simultaneously to suggest a broader range of products to customers.

We are going to use Content-based filtering in this project.

For the gaming recommendation system, we initially used the collaborative filtering specifically used the item-based filtering. The recommendation system that we are currently trying to build here is dependent mainly upon the following kinds of information:

1. User-item interactions: Information that defines user-item relationship.
2. Characteristic information: It is the information that defines the profile of a product or a user.

For the Collaborative filtering, I used the cosine similarity matrix to find the similarity between the different games. The general formula I used for the cosine similarity is given below.

Text

Description automatically generated with low confidence

I specifically used the Item-based similarity matrix for finding the games which are similar to one another.

A picture containing text

Description automatically generated

**Using Steam Dataset**

I gather all my initial data from all the steam official website. My data included a large amount of unwanted information so, I cleaned my initial dataset and stored the new dataset as clean\_dataset.csv. Along with these I needed User reviews and User behavior data also, I found all of those data from steam. Steam I a very popular video game distribution service center. It is currently the best digital game distribution center in the world.

In our Case we created a gaming recommendation using collaborative filtering after clean and preprocessing the data, As you can see I have submitted 2 codes in my submission. The first was created by me(Pavan) and it was very simple recommender system. Later, with the help of my teammate Bruno we created a new recommender system with is more advanced than the initial one, It is having a good UI which ask the user for which game they need the recommendation for and then there are various options available Like, from which year you need the recommendation and We added a dropdown menu it shows all the related games stored inside the database from which the user will select their game which needed to be recommended, There is also a rating bar where the minimum rating is set, this rating is the rating given by the steam to all of its games I think it is the most accurate game rating predictor so I used that and inched the same in my dataset and lastly we added drop box indicating the number of recommendation the user will need. Explanation of both of the implementations is given below as follows: -

**Implementation 1:**

This is the Implementation of the first code which I submitted after cleaning the dataset, I removed all those games that had less than 1000 positive ratings

Graphical user interface, text, application, email

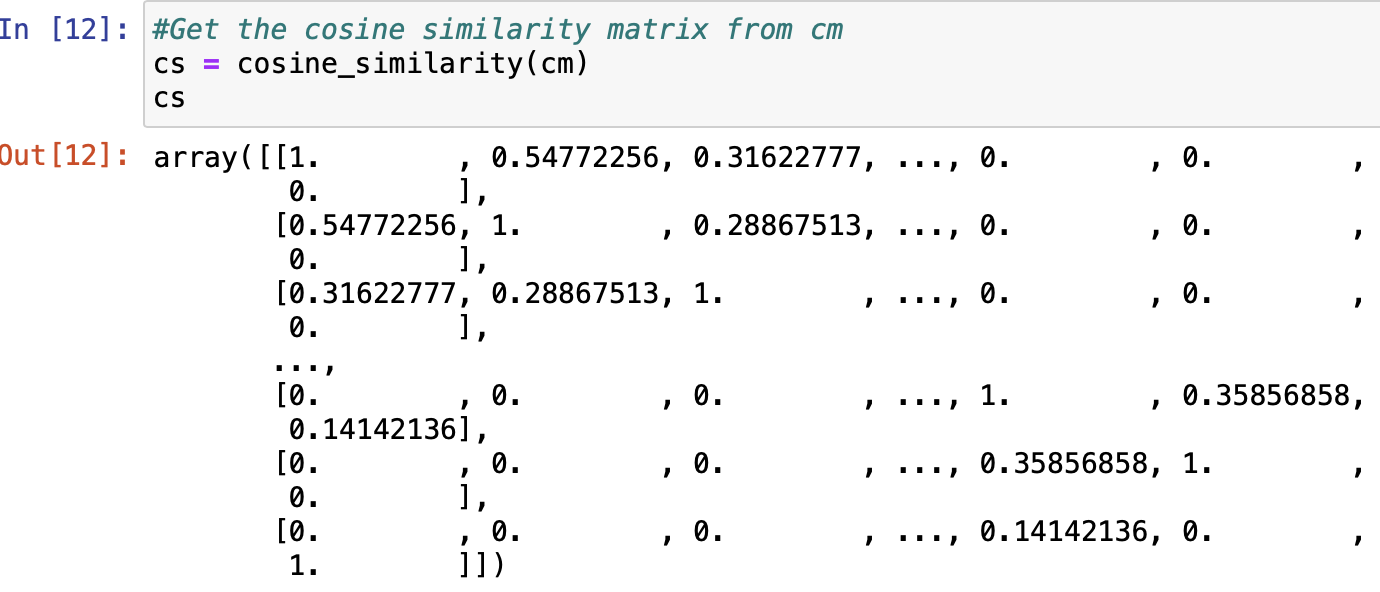
Description automatically generated

After that I reordered the index of the data set, after that I listed out each company and the games they produced. Later than that I checked if there were any missing columns. Now for the Implementation I combined the values of all the important columns into a single string.

Now, I needed to create a new column that would hold the strings, After creating that column I converted the string column into a matrix from with the help of Count Vectorizer. From the matrix that I created I got the cosine similarity matrix and later I printed the cosine similarity matrix to check the dimensions.

Graphical user interface, text, application, email

Description automatically generated



Now I added a variable name to ‘title’ where we would write the name of the game of which we would like to recommendation. After getting the input I would generate the list of every game that has similarity to my game. After that I would sort that list. And in the end, I just printed out the top 10 games who have the most similarity to my game.

Graphical user interface, application

Description automatically generated

This was the initially recommendation system which I made, And later with the help of Bruno I made another recommendation system whose implementation is shown below

**Graphical user interface, text, application, email

Description automatically generated**

**Implementation 2**

So, the newer recommendation system which Bruno and I made together, the initial steps were same, we cleaned and preprocessed the dataset, we checked and removed all those columns which were null and not required currently in out project.

A new step we added is to extract the year in which all games were released, now we created a score column, which kept the count of all the positive and negative scores, and later it just did the total of their scores and return the average in the end. From this we found the total rating of a game and its total score which all the users gave in average. Later I calculated the minimum number ratings that would be required to be in the chart, because if there is just one user rating for a game and that user gave that game full rating it should not be considered because there is not sufficient data gathered on that game.

Graphical user interface, text, application, email

Description automatically generated

Among the games that were qualified, I created a new column called weighted average for them, Then I calculated the weighted average of every game with the help of ‘score’ and ‘total\_rating’ and passing them inside a function. Later I printed out the Top 15 games of all time (overall top 15 games).

With the help of this function, Now I have the weighted score of every game and the non-important games have been discarded.

Text

Description automatically generated with low confidence

We need to make the ‘tfidf vectorizer’ understand difference between games separated by ‘-’ and all the new games for this we need to do the required transformation on our dataset. The screenshot of the same is given below.

Text

Description automatically generated

Now, I created an object for ‘Tfidf Vectorizer’ which contained all the stop words. Later I converted the genre tags which I generated above and converted them into matrix form.

Text

Description automatically generated

Now I created the cosine similarity matrix with the help of TFIDF matrix generated above. After then I created a new function called ‘matching score’ which I used the fuzz ratio to calculate the similarity between the two games.

Now I created a bunch of new functions each of the function sort out the data set in a different order all index are in a specific order for each of the function. The screenshot of the code is given below.

Graphical user interface, application

Description automatically generated

Now Bruno and I created 2 new functions for finding the closest title to which the user title matched, and then the user must select from the drop-down menu.

The closest title function only returned one value and the score of similarity between the entered input and the one inside out data set. So, I created a new function that returns the 10 closest names to which the user entered so even if there was some error in entering the name by the user, the program would give enough options to choose from the correct one.

The drop-down menu contains the names of all the games related to the title the user wrote, and the user have the select the game for which he is looking for inside the drop-down menu.

Graphical user interface, text, application, email

Description automatically generated

Lastly I used the gradio library to predict the similarity ratings and for a good User Interface, With the help of this library and all the index I generated above I created this user interface and generated the required results whose screenshots are attached below.

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

THANK YOU EVERYONE.