MY WORKS

Kaan Karakaş 18070001042

1-DataAnalyze.ipynb

Firstly, I read the netflix.titles data and made the necessary analysis for this data. As a result of these analyzes, I obtained the following results and made inferences by examining the results. The number of movies in the dataset is more than twice the number of TV shows.The number of movies is around 6000 and the number of TV shows is around 2500. I can see that Docuseries in 2017, British TV shows in 2018, crime TV shows in 2019 are in vogue and over-published.TV comedies are on the rise in 2020.There are more children's shows in 2021 and 1993. According to the data, the countries with the highest number of broadcasts are United States, United Kingdom, Japan, South Korea, respectively. United States has widened the gap. (Also, these results are written in the DataAnalyze). Then I found the null values in the data and dropped them. As far as I remember I had 4500 data left but for use case 2, the association rules were taking a lot of time and not giving good results. The reason for this is that if there are too many movies while creating fake data, users watch unrelated movies. There are so many movie options. I decided to reduce the data. We have to 2000. Finally, I saved the cleaned data as new_netflix_data.csv.

2-MockData.ipynb

I decided to use the Faker library for fake data and imported it. For use case 2, I had 400 different people watch at least 25 and at most 30 movies. In this way, association learnin will give better results. First, I randomly determined how many movies the next user watched and randomly selected movies from netflix data. I have a total of 12000 fake user data and saved them as fake_user_data.csv.

3-UseCase1.ipynb

Firstly, I read the fake user data and netfix data. I added the 'genres' column to the fake user data and saved the genres from the netflix data according to the name of the movies. In this way, while working on fake user data, I will be able to see what kind of movies the users are watching. I also added the Day Since Watched column. In this way, I would be able to see how many days ago the users were upset with the movie and clustering accordingly. After these operations, I have 11948 data and 9 columns. I ran the K-means algorithm with reference to the Age and Day since Watched columns. I don't need to encoder because both columns contain integer values. Using the Elbow method, I found the best number of clusters for this data, namely 4. Clusterları şu şekilde isimlendirdim.

0: "Young Person who watched a long time ago",

1: "Old Person who watched recently",

2: "Old Person who watched a long time ago",

3: "Young Person who watched recently"

Then I wrote the recommend movie function. The main purpose of this function was to suggest people from the same cluster and with the same movie taste. Of course, I was careful that he did not propose himself. I wrote a test code to see if it works. In this test code, a random username is selected from the data and sends it to the recommend function. Finally, I saved the model using pickle. The name of the pickle file is k_means_model.pkl.

4-UseCase2.ipynb

Firstly, I read the fake user data and netfix data. I prepare the data for association learning by grouping the user-watched titles by username. In this way, I have combined each user in one line. The purpose of my application of Association rules is to reveal the films that are frequently watched together and to create suggestions accordingly. That's why I combined the movies the user watched in one line. Movies are separated by commas. I applied TransactionEncoder to convert transaction list to encoded format. This was necessary so that we could look at similarity ratios. Then I applied the apriori method by making the min support 0.005. I tried 0.0001 first but it was taking too much time. When I increased the number, the association rules were less. After creating association results, I wrote a function called generate recommendation. This function takes the movie name as a parameter and sends 3 recommended movies using the association result. Then I wrote a test code. This code picks a random movie name from netflix data and sends it to the recomend function. Finally, I saved the model using pickle. The name of the pickle file is association_model.pkl.

5-Flask And PythonAnyWhere

I moved the files, datasets and models that I run in Google Colab into visual studio code. I created 3 html files for web and put them in HtmlForm. Index.html is for the first page and the user will choose a username or movie on that page. According to him, k_means_model or association_model will work. If the user chooses a name and presses the button, it will go to k_means.html and the recommended movies will be shown. If he selects the movie and presses the button, he will go to association.html and suggestions will be shown. I read the datasets for index.html in app.py. In this way, the user will be able to select existing usernames or movies. Movies and usernames are sorted alphabetically for ease of use. if the user chooses a name, he goes to the k_means_recommend function in app.py and there the model is read and 5 movie recommendations are sent to k_means.html. This function is similar to the get_recommend_movie function in UseCase1. Also get_user_genres is put in app.py for use. If the user chooses a movie name, he goes to the association_recommend function in app.py, where the model is read and 4 movie recommendations are sent to association.html. This function is similar to the generate_recommendations function in UseCase2. Finally, the domain was created using pythonAnyWhere. The link is below.

http://kaankarakas35.pythonanywhere.com/