**PROJECT EXPLANATIONS**

CAN BATURLAR

19070001022

**ANALYZE DATA CASE**

I analyze the Netflix titles data in 19070001022.ipynb notebook. These are my findings:

* In total, there are more movies than tv shows.
* In each year, movies and tv shows are increased. Also, last year (2020), there were more tv shows than movies.
* Draw bar chart for 10 shows in each year. I thought “show” is movies and tv shows. So I drew them both in the chart. After that, I show those 10 shows’ titles each year.
* To visualize 10 shows in each genre, I find unique genres and put them into a list. Some shows had more than 1 genre. I drop those rows for the chart because I want to get shows that have 1 unique genre. After that, I show those 10 shows’ titles in each genre.
* In each country, I visualize number of shows. But there were many countries, so I also visualize top 10 countries with the most shows. Numbers 1 and 2 are obvious, the United States and India. Number 10 is Mexico.

After my analysis, I prepared and cleaned data for creating mock viewer data. I dropped columns show\_id and description. After that, there were some null values. First, I fill null values with “Unknown”. But then, when I narrowed the show data for use case 2, I dropped null rows.

Finally, I got the first 2448 rows (the last show title is “Eric ldle's What About Dick?

” Because after my problem in the use case 2, I narrow the data in Excel and the last show become this show. I wanted to show how I narrow the data in Python too.) and saved the new data as “cleaned\_netflix\_data”. Then, I checked the viewers data I created using Mockaroo with ProfileReport.

**MY DATA-MANAGE CODE (data\_manage.ipynb)**

When I created the mock viewer data using the Mockaroo site, it created some wrong rows because of encoding issue. Also, when I created the data, I had 10000 unique usernames. But data had to be 1000 unique usernames with more than 10000 rows. So, in this notebook, I rearranged my mock data. Here are the steps:

* Get viewers and cleaned versions of the Netflix data.
* Get the first 1000 unique usernames from the viewers data.
* This function is for fixing the viewers data to have only 1000 unique users by changing the usernames, ages, and country of the user if the user’s username is different from the first 1000 usernames.
* This function is for creating 20000 more rows in the viewers data. I needed this function to have better results in use case 2.
* Last function is to fix the titles if the title is not in the Netflix data. I needed this function because the first data I created by Mockaroo had encoding issues in the titles. Also, I narrowed the Netflix data to around 2000 rows to have better results in use case 2. But my mock data was already created. So, I used this function to change deleted titles with randomly picked remaining titles.

**USE-CASE 1**

In use case 1, the job was recommending movies based on user (mock viewers data) clustering. I used K-means clustering to label and name my clusters as we did in previous homework. I tried many features for clustering. Those are the features I tried:

* Make date\_watched column datatime and scale it using MinMaxScaler.
* Create a new column named “days\_ago”. This is the number of days between date\_watched and today’s date.
* Encode country\_of\_user column with LabelEncoder.

These are in my code as commented out. I also tried those in the below but removed them from my code:

* Get ratings from the shows data and merge it with the viewers data to have rating for each title in viewers data. Finally, encode ratings column with LabelEncoder. (This did not make sense in clustering)
* Get genres from the shows data and merge it with viewers data to have genres for each title in viewers data. Finally, encode genres column with One-hot encoder. (This was not a good approach for clustering either.)

In the end, I realize that job was working on the viewer data. Meaningful features for clustering were age and percentage\_watched columns. So, these are my final steps for use case 1:

* Get viewers data.
* Get genres from the shows data and merge it with viewers data to have genres for each title in viewers data.
* Create variable named “model\_data” to work on it for clustering.
* Scale age and percentage\_watched columns using StandardScaler.
* Created find\_best\_clusters and generate\_elbow\_plot functions. I copy paste those functions from my Assignment 5 – Clustering.
* Draw the plot for the scaled\_data to analyze elbow method. (Optimal number of clusters is 4)
* Create a new column named “cluster” in the model data to see which user is in which cluster.
* Visualize the clusters and name them.
* Created get\_recommendations function. It returns 10 movies for the provided username. First, gets the cluster of the provided user. Then, gets the shows that the user already watched. After that, gets the users in the same cluster with the same genres watched with the provided user, and shows that they watched. Finally, removes shows that the user already watched from the other users’ shows and returns 10 shows. (In this way, If the user watched only action movies, the model recommends action movies for him.)
* Saved the model as a pickle file named “model\_data”.

**USE-CASE 2**

In use case 2, the job was recommending movies based on a selected movie by using association rule learning. My approach was to get users watched or opened movies in a list to create an apriori data structure. Here are the steps:

* Get viewers and cleaned versions of the Netflix data.
* Create a new dataframe called “user\_show\_df” and the columns are username and show\_title. Then for each unique username in the viewers data, get the list of shows that the user is watched or opened in a list to set it into user\_show\_df dataframe’s show\_title column.
* Get the all show\_title column data in the user\_show\_df dataframe as list to encode it using TransactionEncoder. This creates us a apriori data structure.
* Get frequent datasets using apriori function. I took the threshold(min\_support) very low because when I increase it, found associations decreasing. If I had real users data rather than mock data, I could easily increase the threshold to make the model more accurate. But mock data makes it harder to find associations.
* Get rules by the frequent datasets and sort the rules by the lift.
* Create a function to get recommendations based on the provided show. I took the function from “Week 6 – Association Rule Learnning” lesson. “antecedents” is the movie and “consequents” is the movie recommended for the movie in the antecedents.
* Finally test the model with the randomly picked title from the Netflix data.
* Save the model as a pickle file named “association\_model”.

Simple HTML forms for use case 1 and 2, I deployed my flask app in PythonAnywhere, Here is the link:

<http://canngos.pythonanywhere.com/>