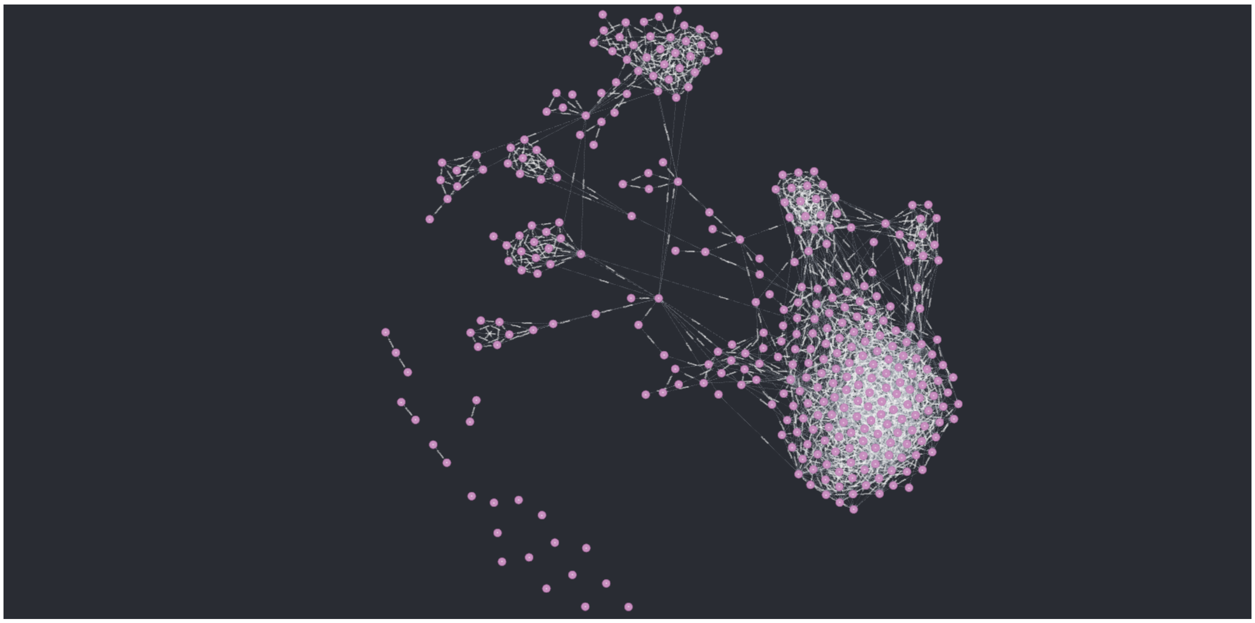
Part 1: Unveiling Patterns

Exploratory Data Analysis (EDA) is required for data understanding so that we can make modeling decisions. It also helps us in data preprocessing.

Visualizing the graph:



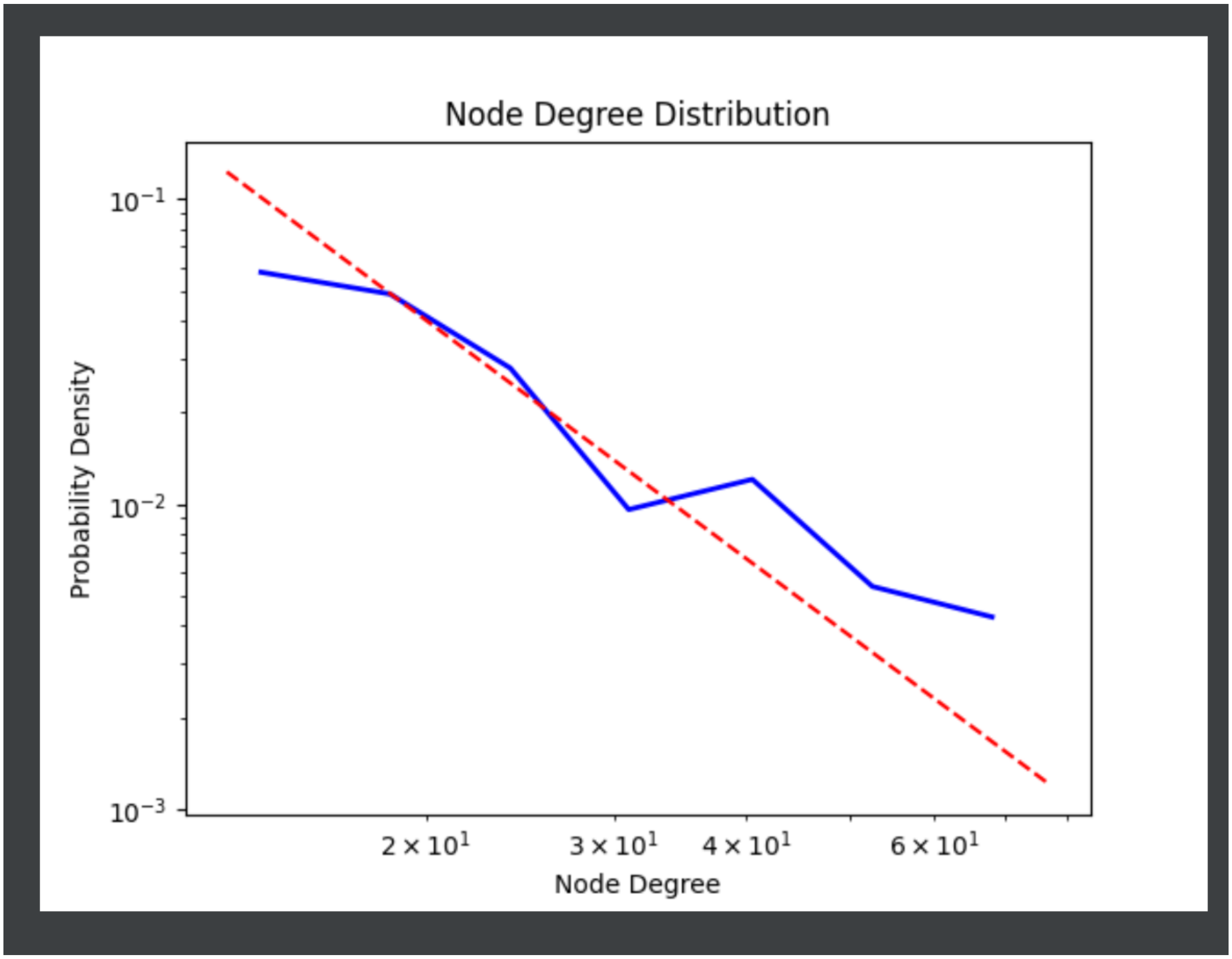
Finding the node degree and arranging them as a pandas data frame - by doing this we can see that there are certain nodes which have more importance as they have a high degree and then there are nodes with 1 or 0 degree, which are less important to our problem

Checking if graph exhibits power law distribution -

Calculating best minimal value for power law fit
Power-law alpha: 2.5997905949181854

This is to check if the graph is possibly a subgraph of a larger graph. If it does not follow then there may be a potential sample selection bias.

But in our case since alpha is close to 2, we can say that we follow a power law distribution and hence we are fine



Finding the density and average degree of graph -
We can find it by dividing the no of edges by no of nodes. The higher the density, the richer the graph is which gives us more possibility in feature learning i.e. more options on hyper parameter tuning.
In our case we get the following result which is pretty good and since the average node degree is not that high it means feature learning is limited to the connections.
The density of the graph is: 7
The average degree of the graph is: 14.518731988472622

Checking if graph is empty, self loops and weighted -
We can see that the graph is not empty and the relationships are not weighted, and also no self loops therefore feature learning will not have a significant impact.
The self loops in the graph are: 0

Part 2: Crafting Tomorrow's Chapters

Here the problem statement is to provide fresh co-authoring opportunities to a given author who approaches us.

I have approached this problem with 2 solutions both of which are a recommendation system but one uses collaborative filtering and another one uses link prediction.

Link prediction directly uses node feature vectors but performs less better when compared to collaborative filtering approach in which we have to embed the feature vectors manually.
In both the models, the performance can be improved further with hyperparameter tuning, using different optimizers or adding more layers to the GNN

Also as the problem statement is to give personalized co-author recommendations based on an author and his features, collaborative filtering excels in this area. But the other approach can also be used as well

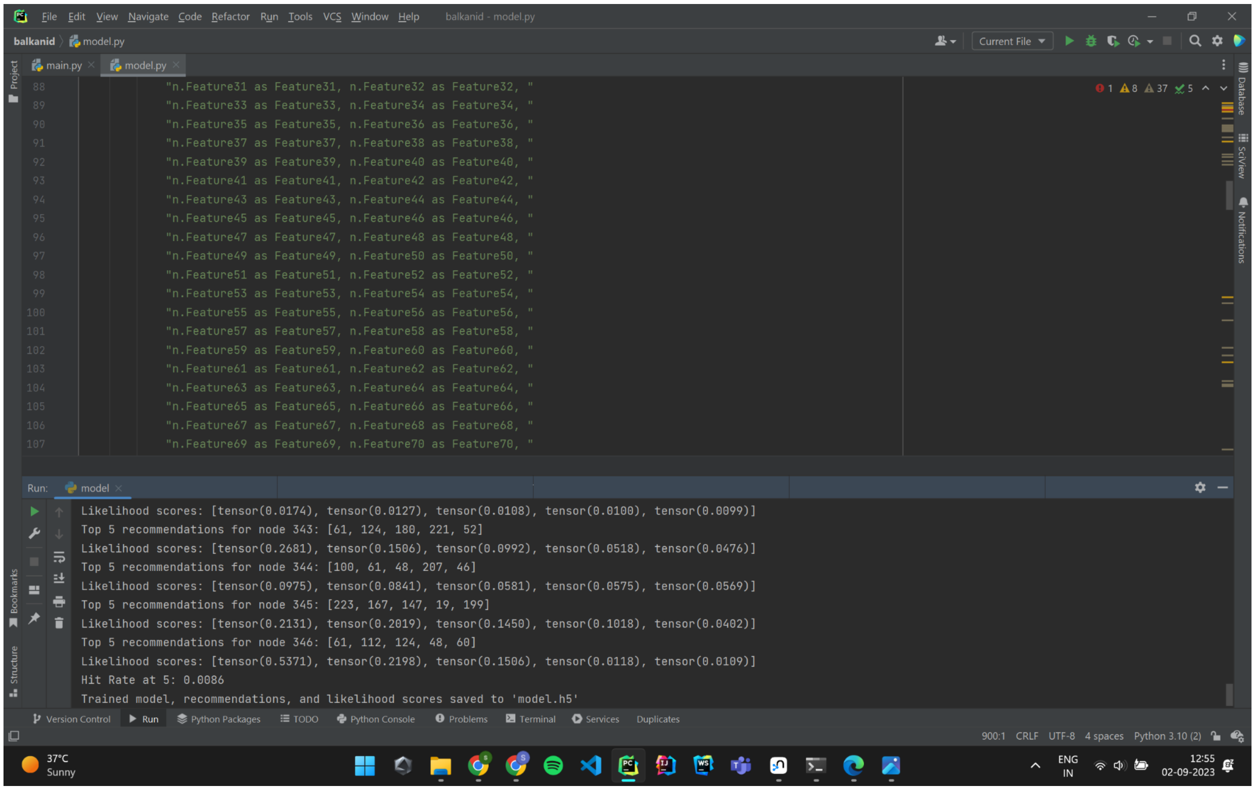
For both the models, I will provide results for nodes with id 56, 100, 343. So that I can include as much variation as possible.

Results for model.py (Approach using Collaborative filtering) -

Top 5 recommendations for node 56: [112, 61, 127, 207, 3]
Likelihood scores: [tensor(0.1807), tensor(0.1032), tensor(0.0527), tensor(0.0491), tensor(0.0466)

Top 5 recommendations for node 100: [148, 221, 42, 40, 44]
Likelihood scores: [tensor(0.2303), tensor(0.1122), tensor(0.0815), tensor(0.0597), tensor(0.0517)]

Top 5 recommendations for node 343: [61, 124, 180, 221, 52]
Likelihood scores: [tensor(0.2681), tensor(0.1506), tensor(0.0992), tensor(0.0518), tensor(0.0476)]

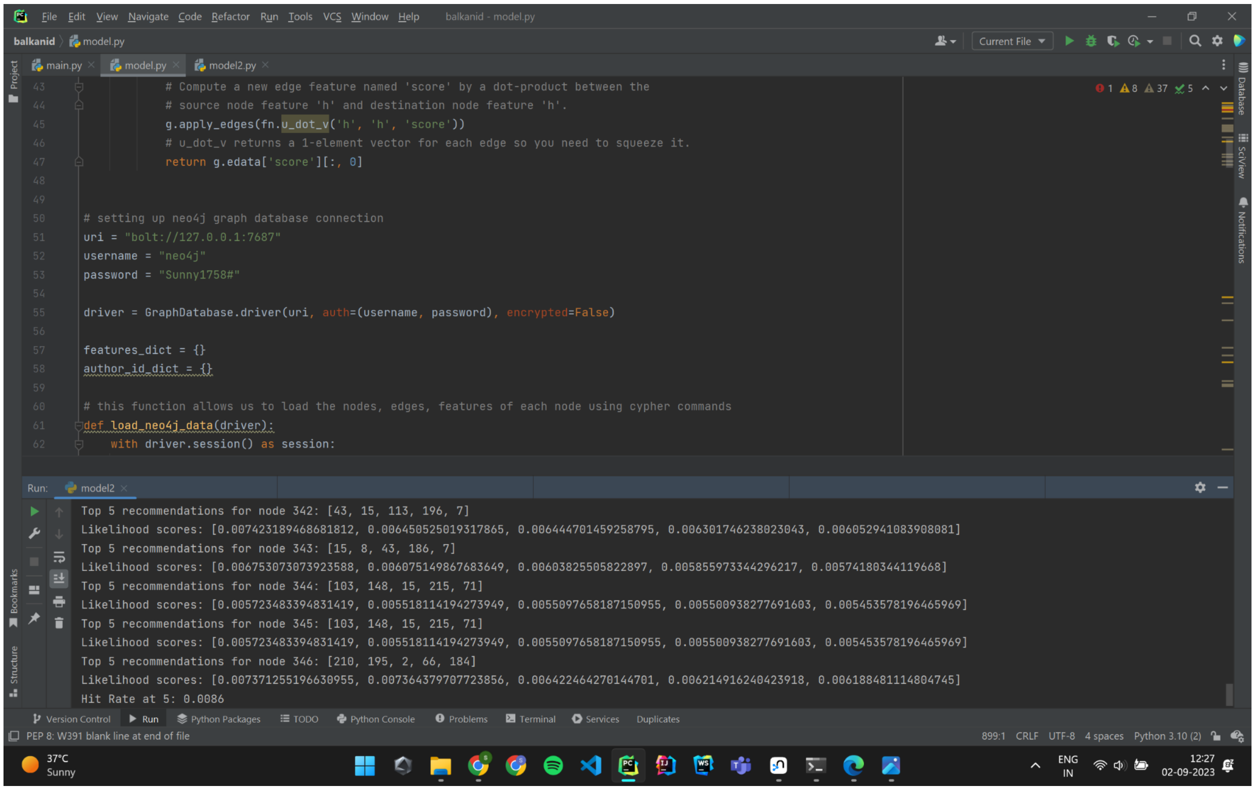


Results for model2.py (Approach using Link prediction) -

Top 5 recommendations for node 56: [20, 140, 180, 183, 167]
Likelihood scores: [0.010310431942343712, 0.007679259404540062, 0.007418027613312006, 0.007402912247925997, 0.007058937568217516]

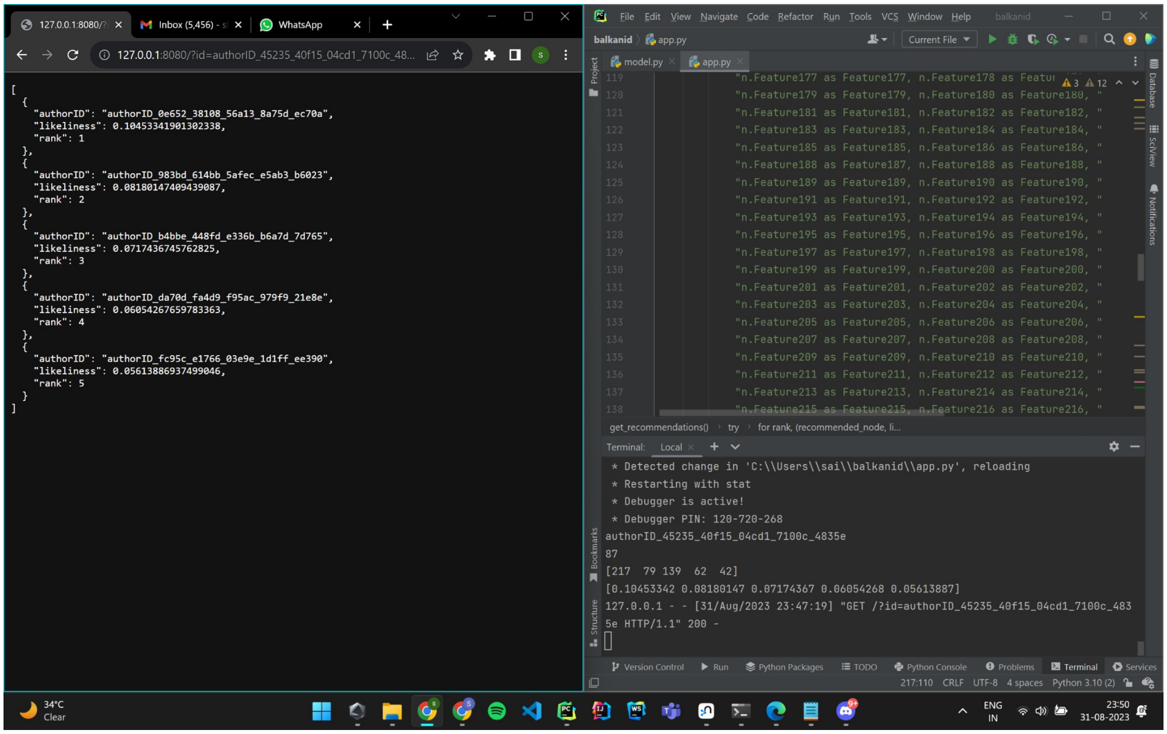
Top 5 recommendations for node 100: [204, 60, 160, 208, 1]
Likelihood scores: [0.008443913422524929, 0.008035272359848022, 0.007700406480580568, 0.007544406224042177, 0.007389116566628218]

Top 5 recommendations for node 343: [15, 8, 43, 186, 7]
Likelihood scores: [0.006753073073923588, 0.006075149867683649, 0.006038255505822897, 0.005855973344296217, 0.00574180344119668]

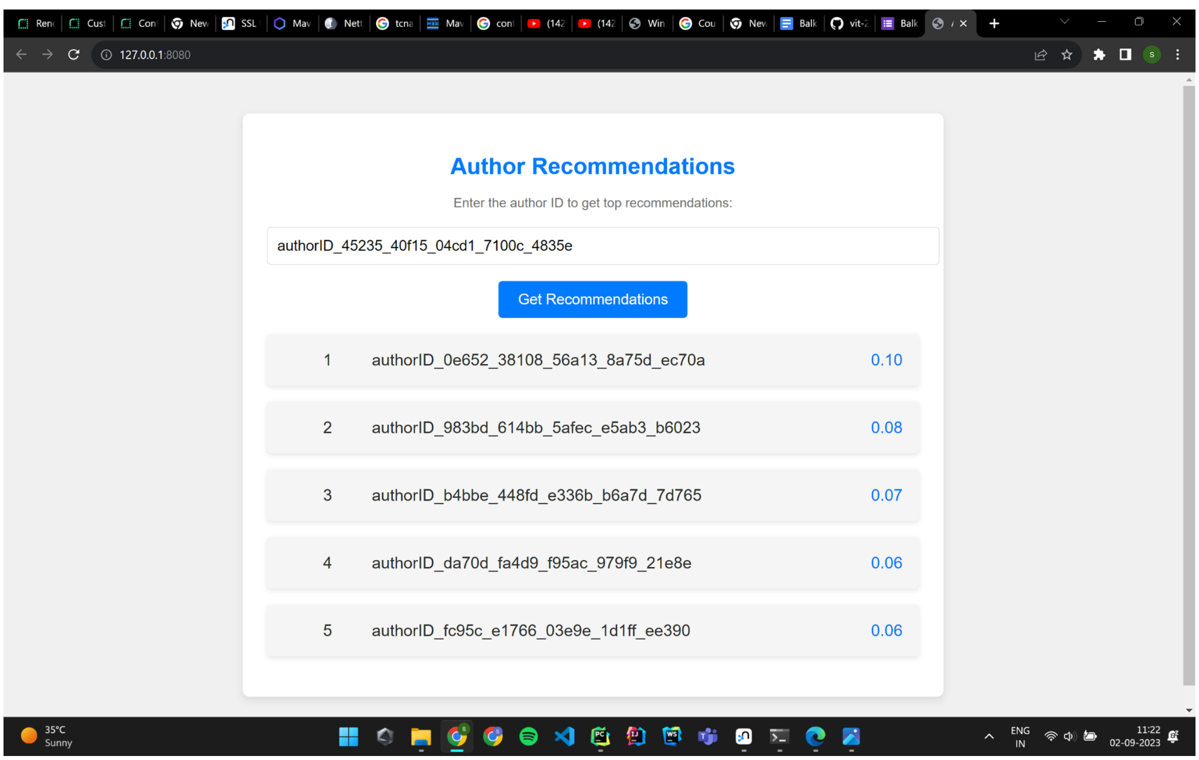


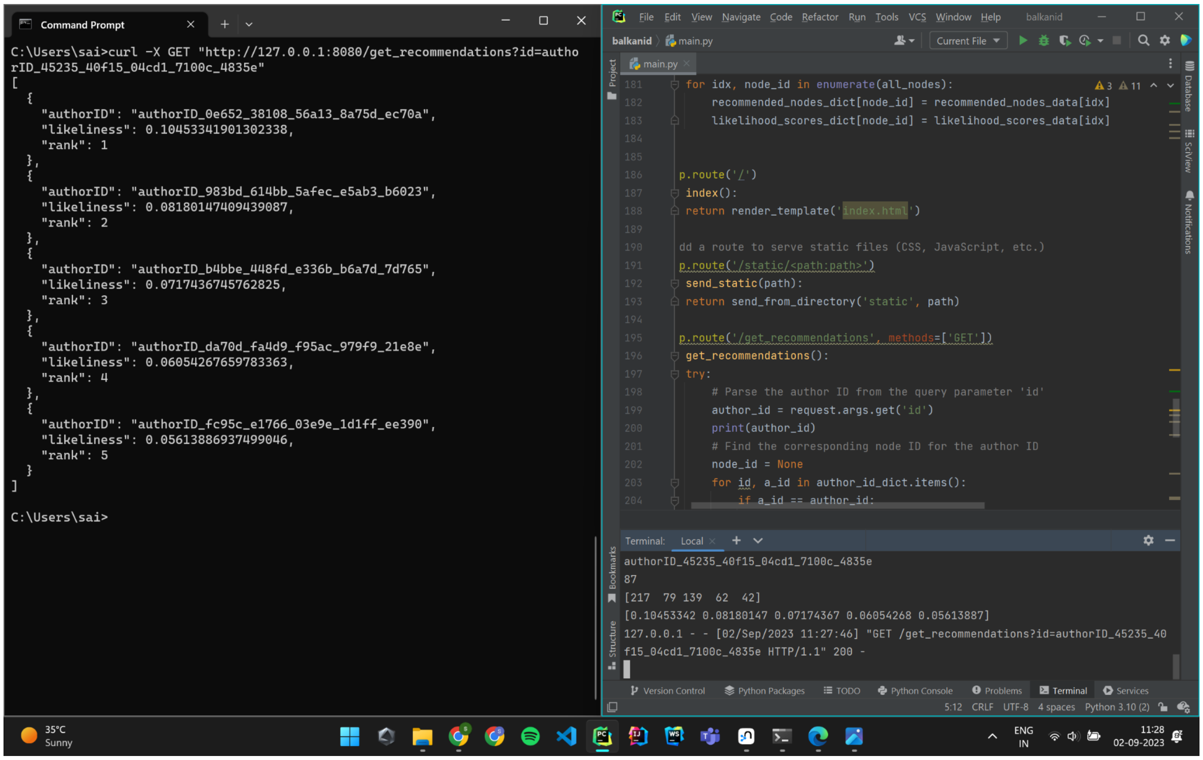
Part 3: Cloud Chronicles

Created a flask application and used h5py module to store the model outputs (collaborative filtering as it performs better). The flask application takes the id parameter from GET Request, finds the nodeid for the given authorid and finds the recommended coauthor (nodes) and likelihood scores using the .h5 file.
All of the results mentioned are using model.py (collaborative filtering approach)
These results are stored in the JSON format response as mentioned in the file



After getting the response in the required format, then I included a index.html file as well style.css to improve the front end of the application.





After the successful deployment of the flask app on the local host, I tried the following cloud deployment websites to deploy my flask application in. They are pythonanywhere, render and google cloud.

But in all of the above mentioned deployment websites, I was getting the following error -

Sep 1 07:46:33 PM neo4j.exceptions.ServiceUnavailable: Couldn't connect to 127.0.0.1:7687 (resolved to ()): Sep 1 07:46:33 PM Failed to establish connection to ResolvedIPv4Address(('127.0.0.1', 7687)) (reason [Errno 111] Connection refused) - how to solve this error, that I am getting because neo4j database is not getting accessed.

I went through stack overflow, neo4j documentation, github and the only solution I was able to find was I needed to setup SSL certification and then use HTTPS instead of BOLT, as the deployment website was using HTTPS protocol.

https://neo4j.com/docs/operations-manual/current/security/ssl-framework/#ssl-certificates

I tried setting up the certificates and SSL over HTTPS by following the above documentation, But when I did follow all the steps mentioned above, I was getting the DBMS cannot be setup error in neo4j Desktop and could not find any solutions for it.
Hence I was unable to setup the application of Cloud, in the given time frame. I had to use neo4j Desktop, instead of it's web version or sandbox, as only the desktop version had accepted the .dump file format which was provided to us in the dataset.

I would have resolved this issue, if I had more time, but I gave my best trying to debug it and solve it. The other solution, I think of as now, is to save the made dgl graph use it for getting nodes, edges and feature vectors, rather than importing neo4j and using load_neo4j_data