Raphael Ku
SI 671
Homework 2
Username (Kaggle): Raphael Ku
uniqname: rapku

Part I: Link prediction

Libraries used	Networkx, random
Approach	Due to memory limitations on my system, I implemented reservoir sampling, allowing me to run through the network file and extract 10% of the edges there with uniform probability (3.09M edges) Afterwards, a subgraph was produced using the number of edges sampled, and the Jaccard coefficient was calculated between each node and any node 2 steps away (neighbor of neighbors). Reservoir sampling, subgraph generation, and similarity measurement was done 10 times. The top 150,000 node pairs were kept for one iteration and were part of node pairs evaluated for the next iteration. For the 150,000 candidates, a separate machine that can load the whole network file was used, and the Jaccard coefficient for each of the node pairs were done. The top 50,000 were chosen from there.
Other methods used	The same procedure was done with specific filters to reduce the amount of data to keep in memory. One such method was only pairing nodes to neighbors of neighbors with degree of 30 and above. On testing, this led to node pairs with Jaccard coefficients of less than 0.2, even in their own subgraph, so this was discarded.

Part II: Attribute prediction

Libraries used	Networkx, collections
Approach	Using Jaccard coefficient to weight ego network (radius=2) nodes, combination with highest cumulative weight is chosen. This method was chosen, due to the intuition that nodes that share common neighbors with a target node should provide more accurate information about a target node. Thus, choosing the most suggested combination of attributes by the most similar nodes should be a good heuristic for selection.
Other methods used	Used a naïve method that just predicts the most common combination of attributes from its ego network (radius 2). General performance is around the same as Jaccard coefficient, with marginal improvement in score made by Jaccard coefficient cumulative weights. However, the naïve method is much more computationally efficient. Attempted to use Sklearn's Random Forest implementation with MultiOutputClassifier. While able to build the Y-train samples, couldn't identify good features that were computationally efficient to build for training dataset.

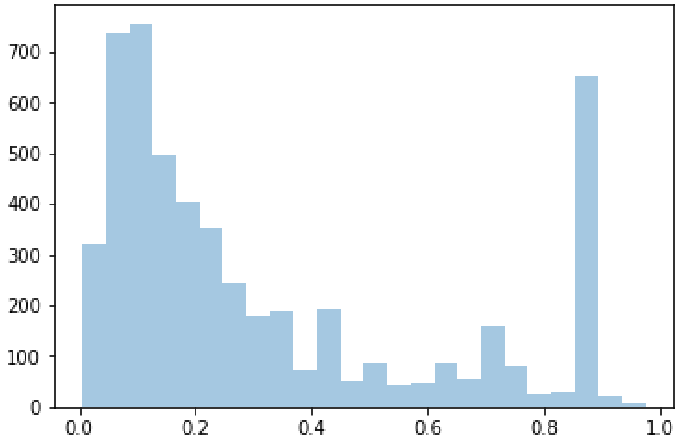
Part III: **Analysis**

Homophily:

The attribute prediction methods used (Choosing most frequent/highest cumulative Jaccard coefficient) relied on homophily being present, in that the direct neighbors of a node will most likely be like the target node. This resulted in a 0.83531 F-1 score in the public dataset.

One potential metric for similarity between attributes and nodes is to use a modified cosine similarity measure, where the vectors in use are the weights for each combination of attributes found in the neighbors of a node.

Using that metric, the following results were found for the subgraph of node 4559 compared to all neighbors of 4559. For 4559 and its neighbors, there seems to be 2 distinct groups, which may indicate homophily in high similarity pairs, and distinct communities on lower similarity pairs.



Error analysis

AttrType	Error rate	# of errors (Dev)
T0	1.95%	12,909
T1	35.83%	237,462
T8	1.88%	12,472