CMPE351 Assignment 3

James Sanii

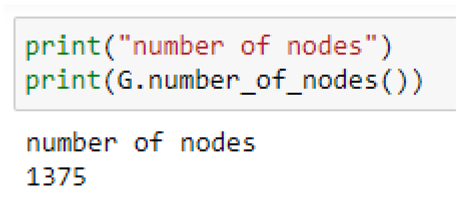
Part 1: Select data and create network.

For the assignment I decided to create a network where I treat papers as nodes and citations relationships and edges.

I had to select a subset of the papers to analyze since the dataset was to large to analyze the entire thing. I started by filtering by key word to find papers related to data analytics. The key words I chose to filter by are “Deep”, “Learning” and AI. The paper must have all 3 words in the abstract to be part of the subset. I then filtered by year to further reduce the dataset because I wanted the most recent papers and wanted to see how they interact with other recent papers. I ended up filtering out all papers that were from 2015 or before. So the final dataset contained papers with “Deep”, “Learning” and AI in their abstract and were published after 2015. I then used this data to create the Graph. Since the graph was not fully connected, I selected the largest subgraph to be the graph for this assignment. Larger graphs were tested but my kernel crashed when preparing the data for supervised learning.

Part 2:Network Analysis, reporting basic statistics

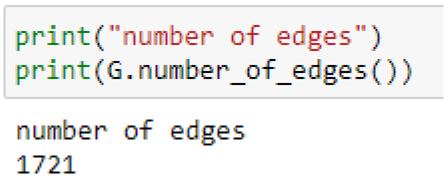
1) The network has 1375 nodes.



```
print("number of nodes")
print(G.number_of_nodes())
```

```
number of nodes
1375
```

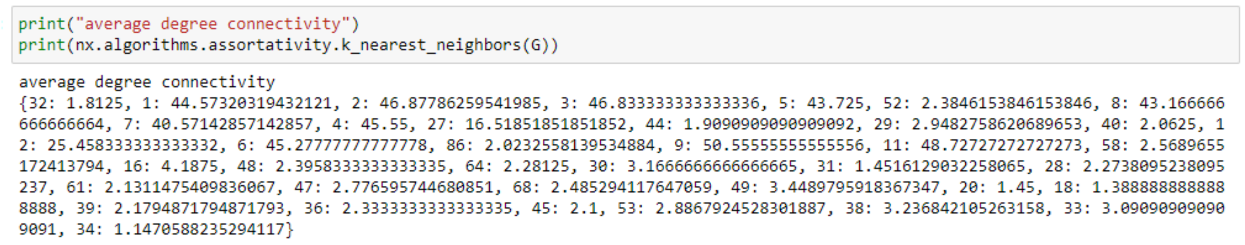
2) The network has 1721 edges.



```
print("number of edges")
print(G.number_of_edges())
```

```
number of edges
1721
```

3) Average degree connectivity results:

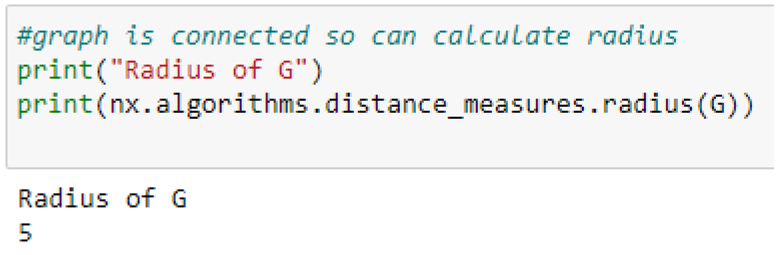


```
print("average degree connectivity")
print(nx.algorithms.assortativity.k_nearest_neighbors(G))
```

```
average degree connectivity
{32: 1.8125, 1: 44.57320319432121, 2: 46.87786259541985, 3: 46.833333333333336, 5: 43.725, 52: 2.3846153846153846, 8: 43.166666
666666664, 7: 40.57142857142857, 4: 45.55, 27: 16.51851851851852, 44: 1.9090909090909092, 29: 2.9482758620689653, 40: 2.0625, 1
2: 25.458333333333332, 6: 45.27777777777778, 86: 2.0232558139534884, 9: 50.55555555555556, 11: 48.72727272727273, 58: 2.5689655
172413794, 16: 4.1875, 48: 2.3958333333333335, 64: 2.28125, 30: 3.1666666666666665, 31: 1.4516129032258065, 28: 2.2738095238095
237, 61: 2.1311475409836067, 47: 2.776595744680851, 68: 2.485294117647059, 49: 3.4489795918367347, 20: 1.45, 18: 1.388888888888
8888, 39: 2.1794871794871793, 36: 2.3333333333333335, 45: 2.1, 53: 2.8867924528301887, 38: 3.236842105263158, 33: 3.09090909090
9091, 34: 1.1470588235294117}
```

Average degree connectivity gives the average neighbor degree for all nodes with the same degree. This measures how connected nodes with certain degrees are. Nodes with a degree of 9 have the highest average degree connectivity which means those nodes on average are connected to other nodes with the highest degree.

4) Radius of the Network

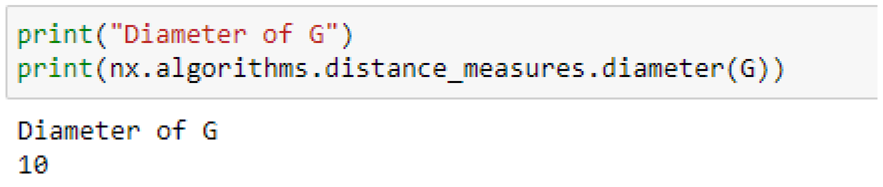


```
#graph is connected so can calculate radius
print("Radius of G")
print(nx.algorithms.distance_measures.radius(G))
```

```
Radius of G
5
```

Minimum distance among all the maximum distances between a vertex to all other vertices is 5 edges.

5) Diameter of the network

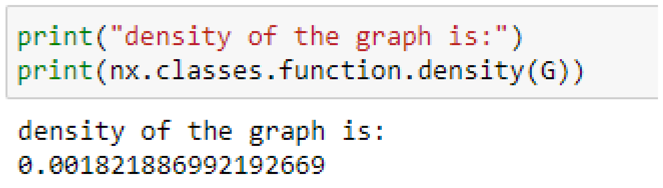


```
print("Diameter of G")
print(nx.algorithms.distance_measures.diameter(G))
```

```
Diameter of G
10
```

This means that the maximum distance between a pair of vertices in 10 edges. So the maximum distance from one node to another is 10 edges.

6) Density of the network

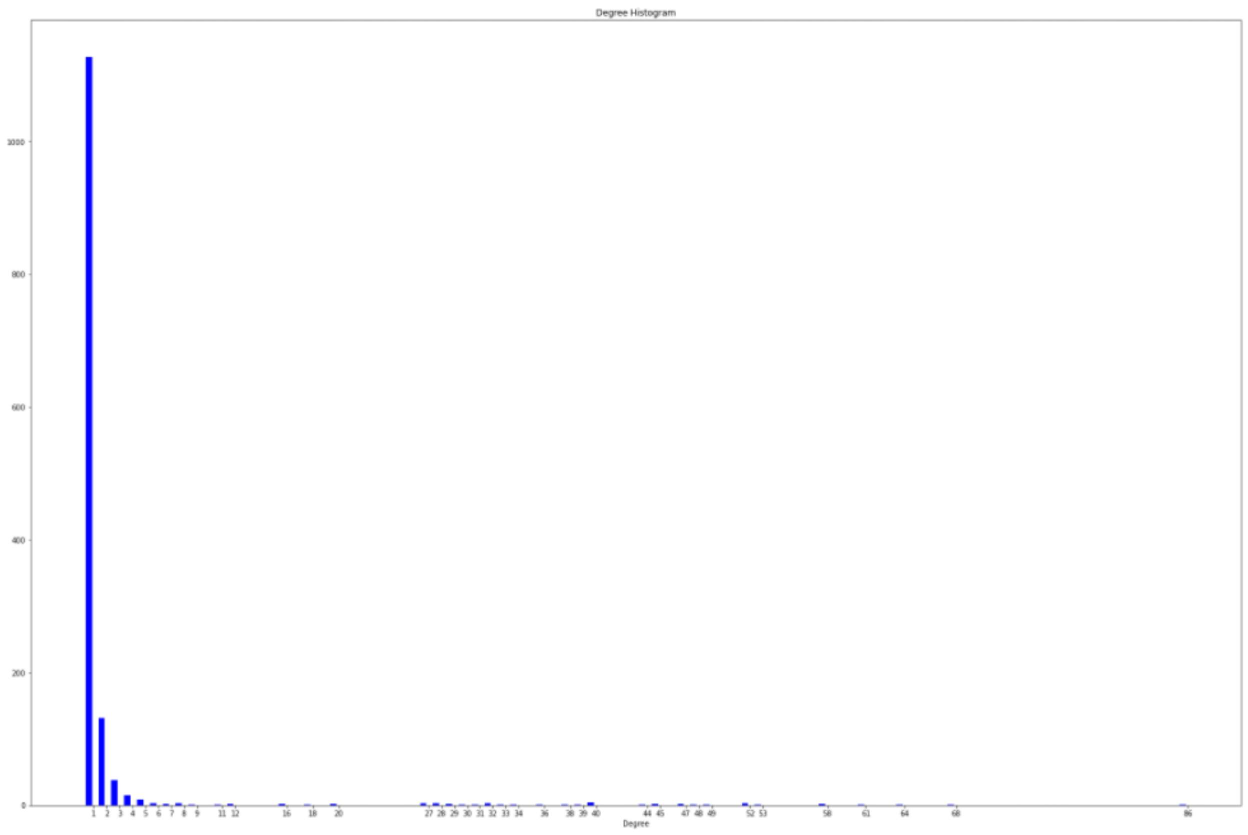


```
print("density of the graph is:")
print(nx.classes.function.density(G))
```

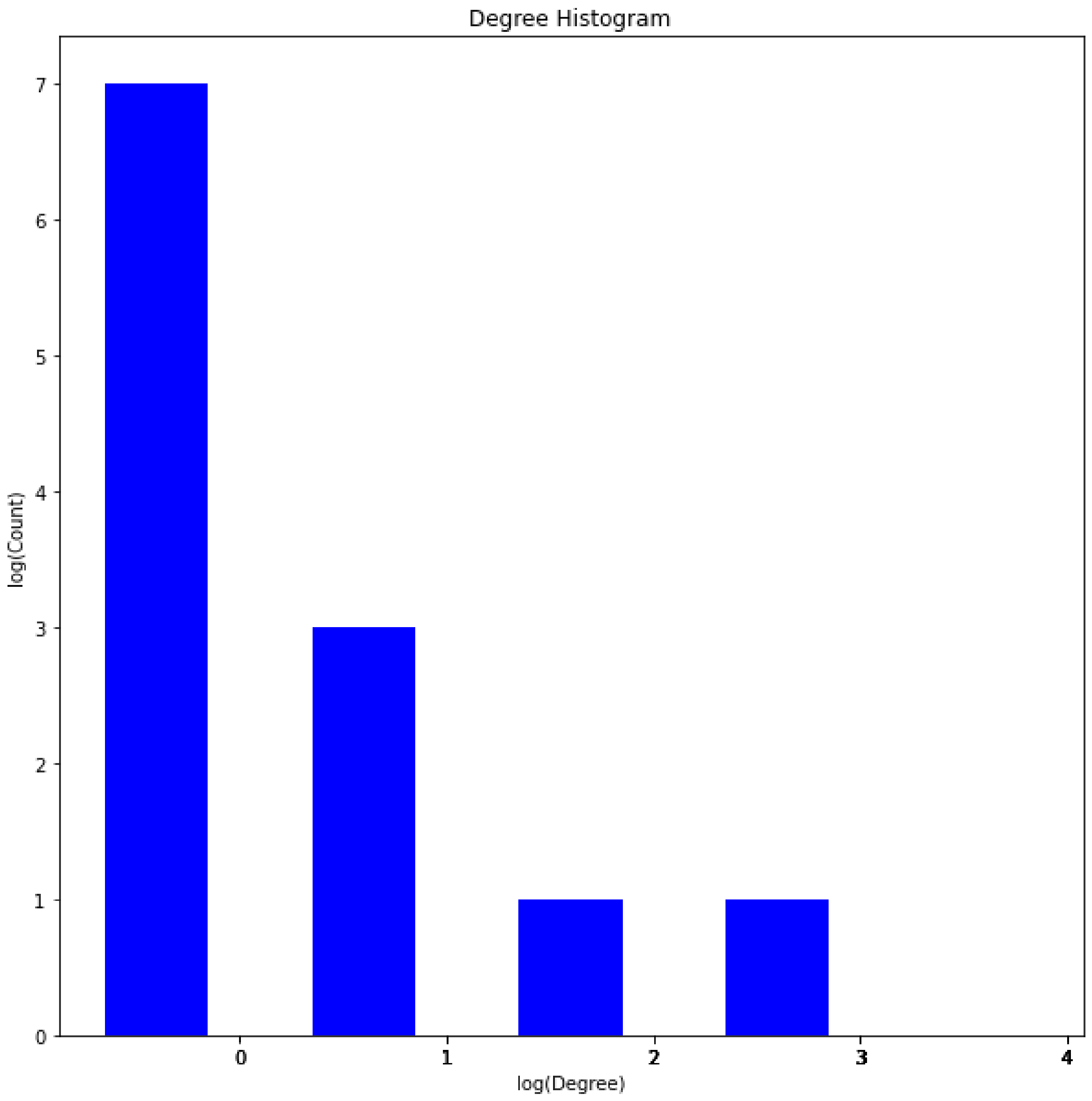
```
density of the graph is:
0.001821886992192669
```

Density of the network describes how many of the potential connections are connected. This graphs density value is very small meaning that most possible connections are not part of the graph. This means there are very few edges compared to the maximum possible amount of edges.

Degree Histogram



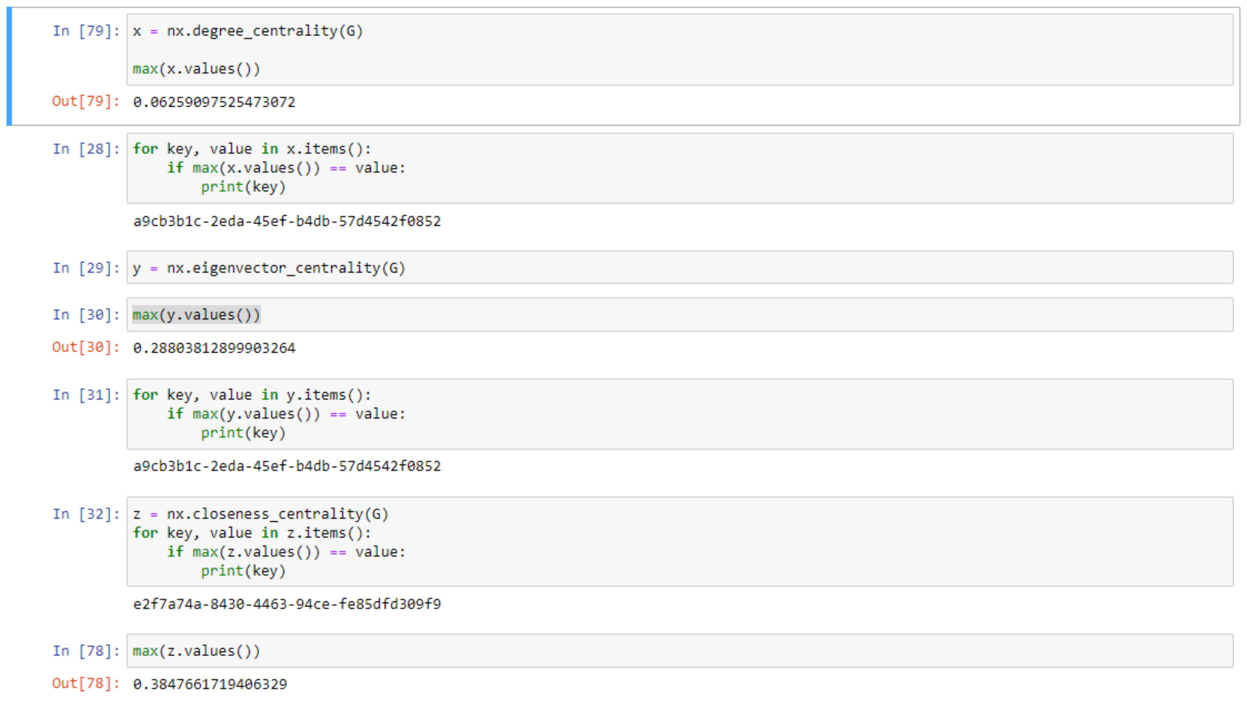
Since it was hard to read, I used a log-log scale for this next histogram.



Looking at both histograms I can see that the majority of nodes have a degree of 1, then 2, then 3, then 4 and after that almost all other values are between 0 and 10 nodes with the given degree centrality.

Part 3: Node centrality analysis

For node centrality analysis I calculated degree centrality, eigenvector centrality and closeness centrality. Degree centrality was used to figure out the most connected individual, this node represents the node that is cited the most or cites others the most. This node is found by using the max function to find the node with the highest Degree Centrality value. Eigenvector centrality was used to find the node with the largest influence over the entire network. I believe the node with the maximum eigenvector centrality value represents the most import source in the network, meaning many other papers are influenced by this piece of work. Closeness centrality is best for finding individuals who are at the best place to influence the entire network. The node with the highest closeness value is the nodes that would affect the field the most if changes or a retraction of the paper was done. The node id, that are most impactful for each of these metrics are shown below from the python output.



```
In [79]: x = nx.degree_centrality(G)

         max(x.values())

Out[79]: 0.06259097525473072

In [28]: for key, value in x.items():
             if max(x.values()) == value:
                 print(key)

         a9cb3b1c-2eda-45ef-b4db-57d4542f0852

In [29]: y = nx.eigenvector_centrality(G)

In [30]: max(y.values())

Out[30]: 0.28803812899903264

In [31]: for key, value in y.items():
             if max(y.values()) == value:
                 print(key)

         a9cb3b1c-2eda-45ef-b4db-57d4542f0852

In [32]: z = nx.closeness_centrality(G)
         for key, value in z.items():
             if max(z.values()) == value:
                 print(key)

         e2f7a74a-8430-4463-94ce-fe85dfd309f9

In [78]: max(z.values())

Out[78]: 0.3847661719406329
```

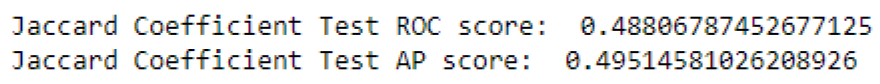
Part 4: Link prediction

The unsupervised method I decided to use is Jaccard Coefficients. I used logistic Regression for supervised learning.

The classification-based metric I decided to use ROC score. Mean Average precision is the Rank based evaluation metric that I decided to use.

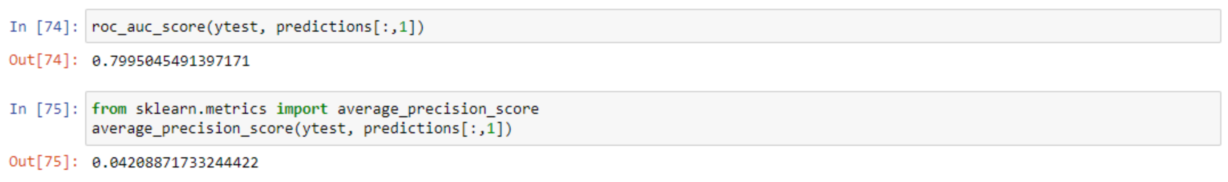
Jaccard Coefficient results are as followed:

Since Jaccard coefficient ROC score is around 0.5 it suggests that the model has no real predictive power. The average precision value is being around 0.495 suggests that is ranking is relatively good.



```
Jaccard Coefficient Test ROC score:  0.48806787452677125
Jaccard Coefficient Test AP score:  0.49514581026208926
```

The following are the results of the Supervised learning.



```
In [74]: roc_auc_score(ytest, predictions[:,1])

Out[74]: 0.7995045491397171

In [75]: from sklearn.metrics import average_precision_score
         average_precision_score(ytest, predictions[:,1])

Out[75]: 0.04208871733244422
```

The ROC score is around 0.8 which means it is a decent predictor. But the average precision score is very low. This means that the algorithm is good a predicting links overall but is bad at predicting the minority positive class.

Overall looking at the metrics, Supervised learning performed better at predicting potential links whereas unsupervised learning was better at ranking the links in what links are most likely to have a link.