Link Prediction in Facebook Networks

Group-13
Shubham Jain - AU1940315
Gaurav Bajaj - AU1940169
Homak Patel - AU1940042

Objective

- Objective-1: To find the most influential nodes in the network.
- Objective-2: Link Prediction in the network.
- One of the most important study issues in the subject of graphs and networks is link prediction.
- The goal of link prediction is to identify pairs of nodes that will either form or not form a link in the future.
- Predict formation of link between 2 unconnected nodes in the future.
    - How is it useful?
    - Predicting hidden links amongst terrorists
    - Find interactions between proteins
    - Friends Recommendation system
    - And many more...

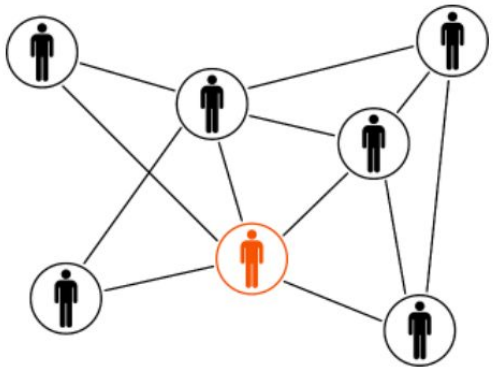


Image Source

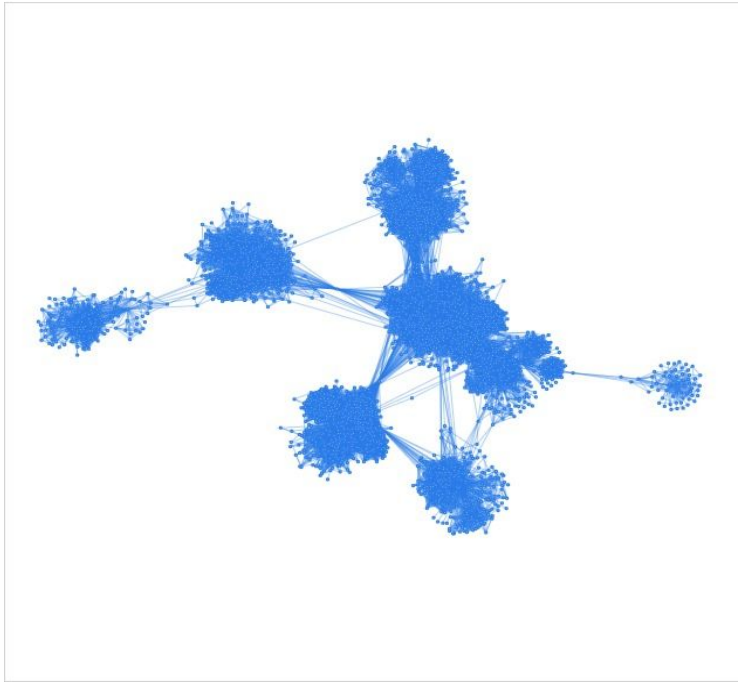
Dataset

- http://snap.stanford.edu/data/egonets-Facebook.html
- Nodes represent users.
- Undirected Edge between 2 users represents that they are friends with each other.
- 10 ego-networks, consisting of 193 circles and 4039 users.
- For our analysis we have taken entire dataset into consideration.

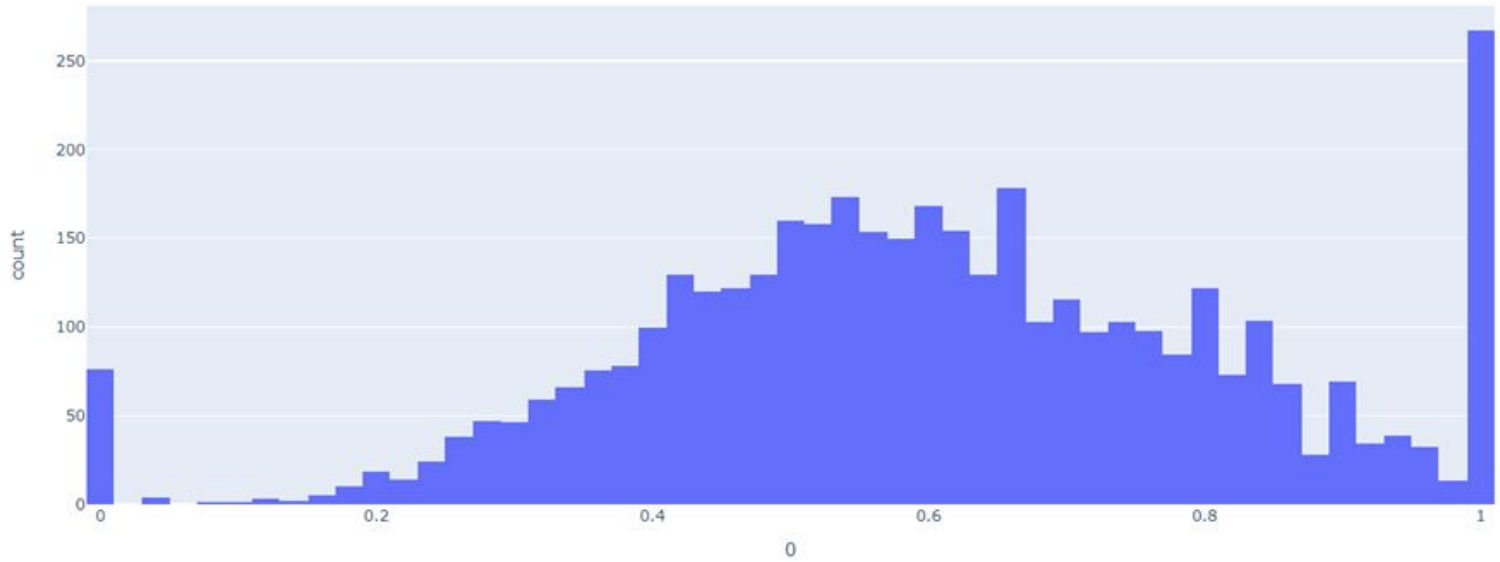
Statistics about Dataset

Nodes	4039
Edges	88234
Nodes in largest WCC	4039
Edges in largest WCC	88234
Nodes in strongest WCC	4039
Edges in strongest WCC	88234
Average Clustering Coefficient	0.6055

A Simple overview of the Network



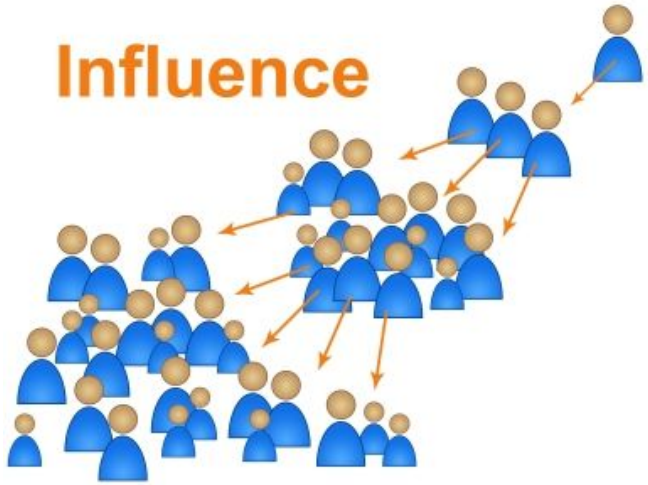
Clustering Coefficients



Problem

“Which are the most influential nodes in the dataset?”

- Influential nodes - These are the nodes with high centrality measures.
- Well, how does this matter anyway?
- Some practical applications: -
  - Rumour Containment
  - Viral Marketing
  - Virus Spreading



Approach

Building the Graph

- Used Networkx Library to build and visualize graphs.

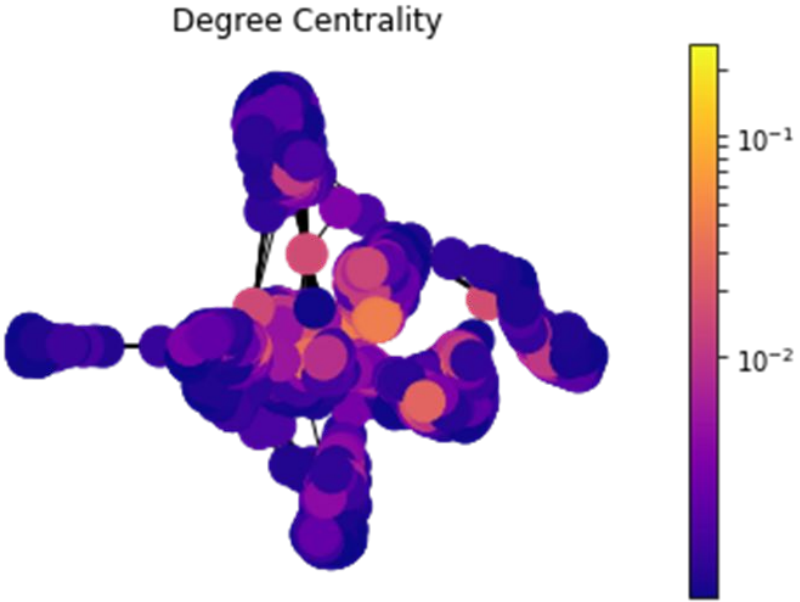
Computing Measures of Centrality

- Eigenvector Centrality
  - Relationship of a node with high-scoring nodes contribute more to its eigenvector score.
- Degree Centrality
  - number of connections (immediate neighbors) of a node.
- Closeness Centrality
  - Average of shortest path length from node to every other node.
- Betweenness Centrality
  - number of times a node lies on the number shortest path between two nodes

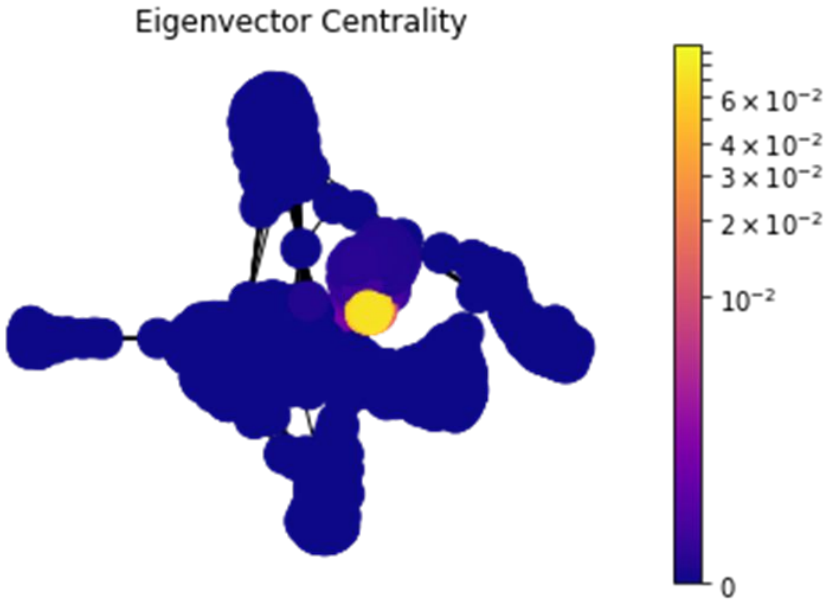
Rank Aggregation

- For this, we used Markov Chains Type-4.
- Also looked at other methods like Borda and Kemeny-Young.
- Also performed Community Detection in order to get an idea about different communities present in the network.
- Also, looked at major influencers in each community and maximum rank of amongst those influencers.

Results

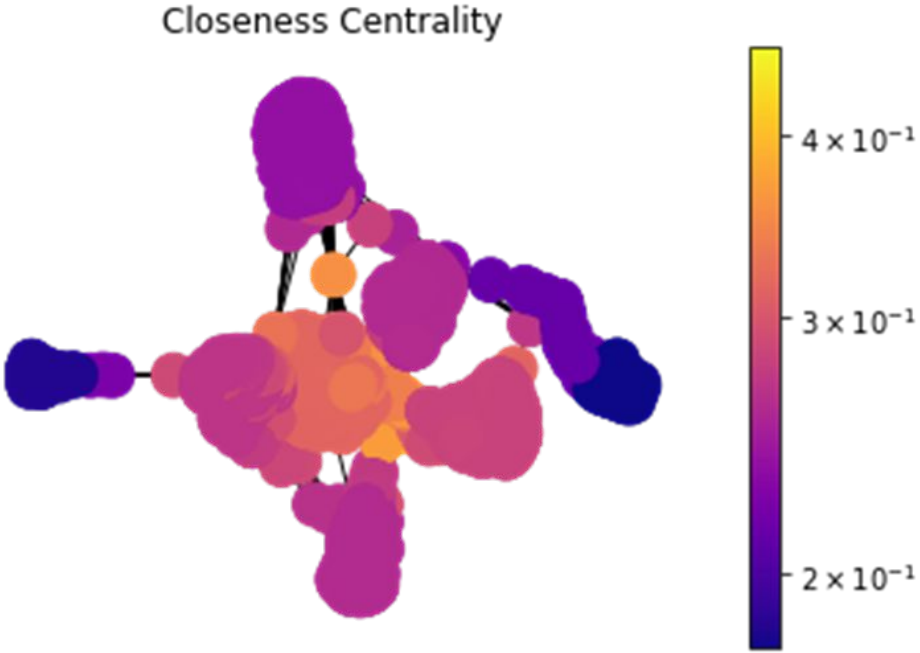


[Fig. Degree Centrality Viz.]

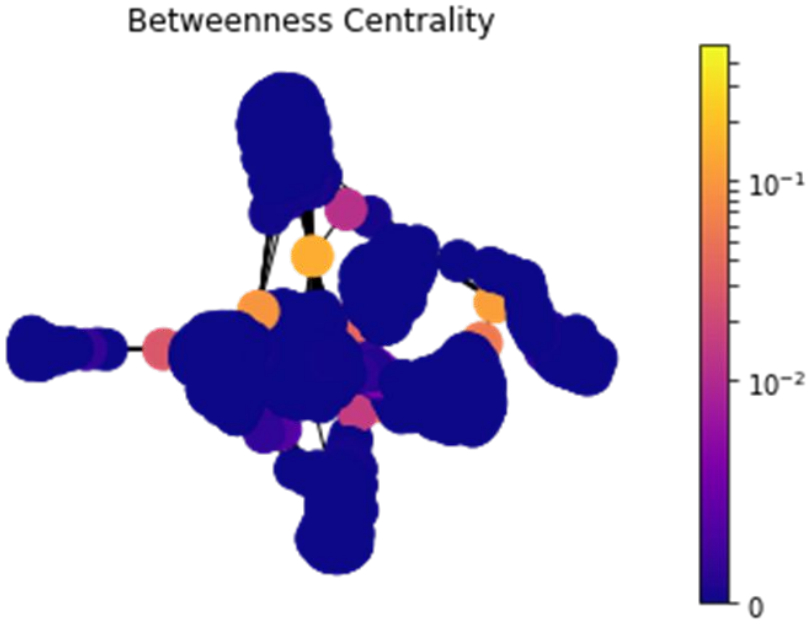


[Fig. Eigenvector Centrality Viz.]

Results

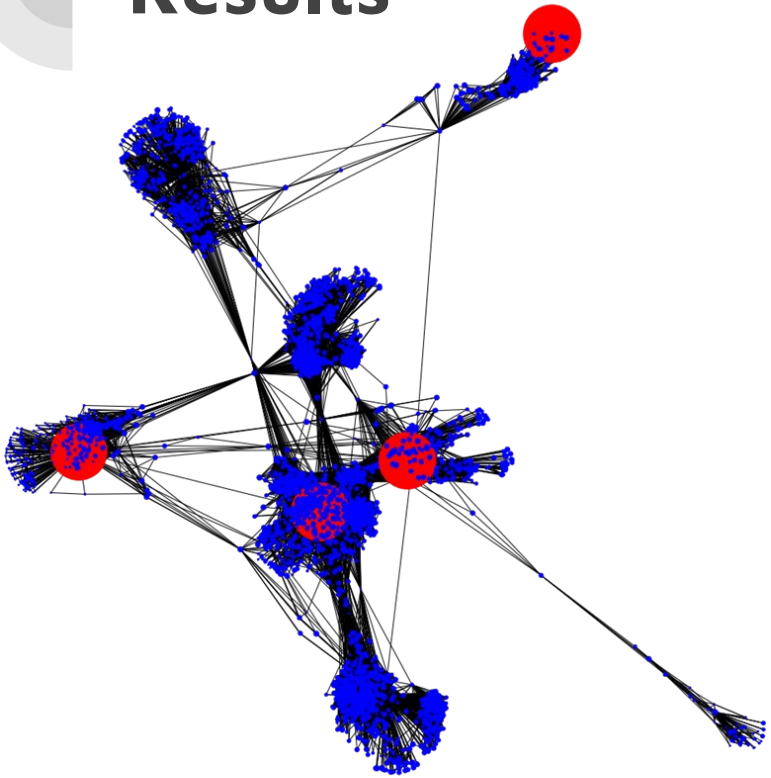


[Fig. Closeness Centrality Viz.]

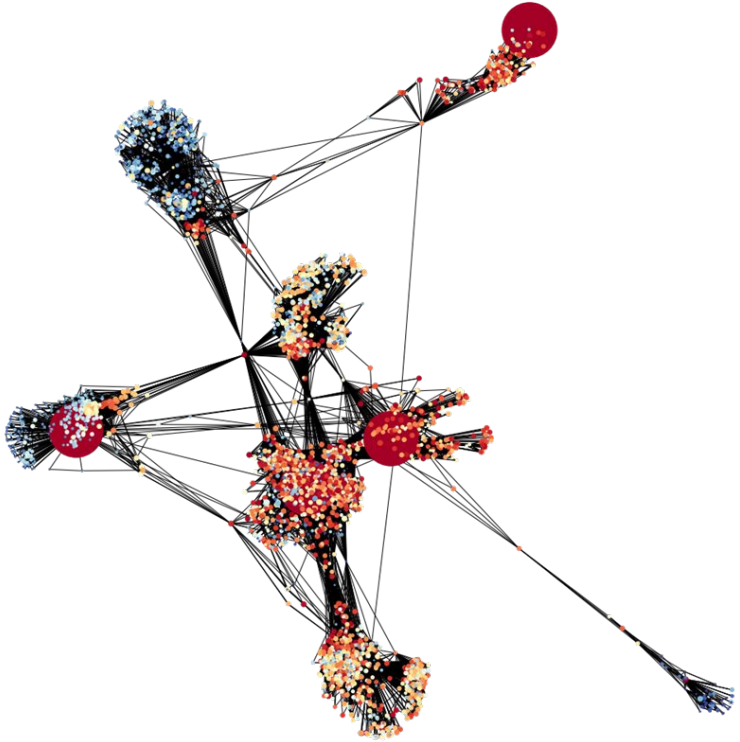


[Fig. Betweenness Centrality Viz.]

Results

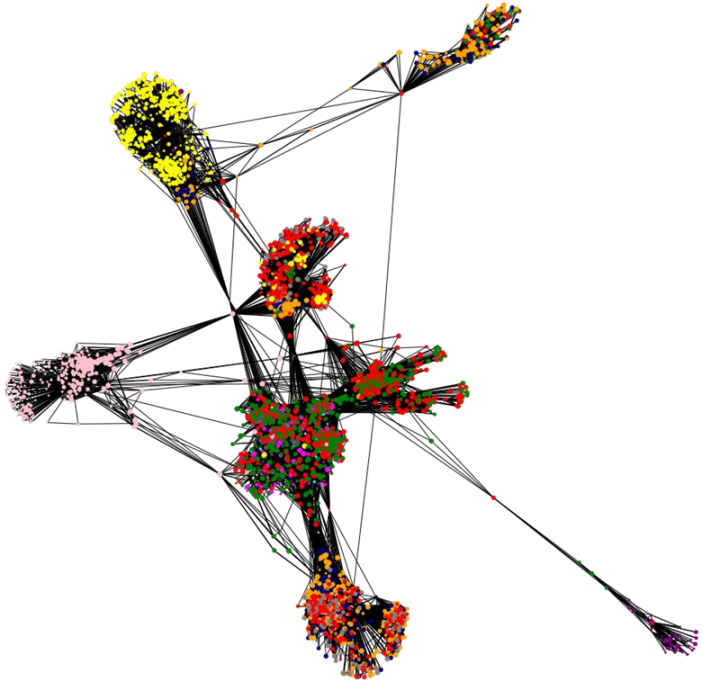


[Fig. Top 4 influential nodes Viz. (in red)]

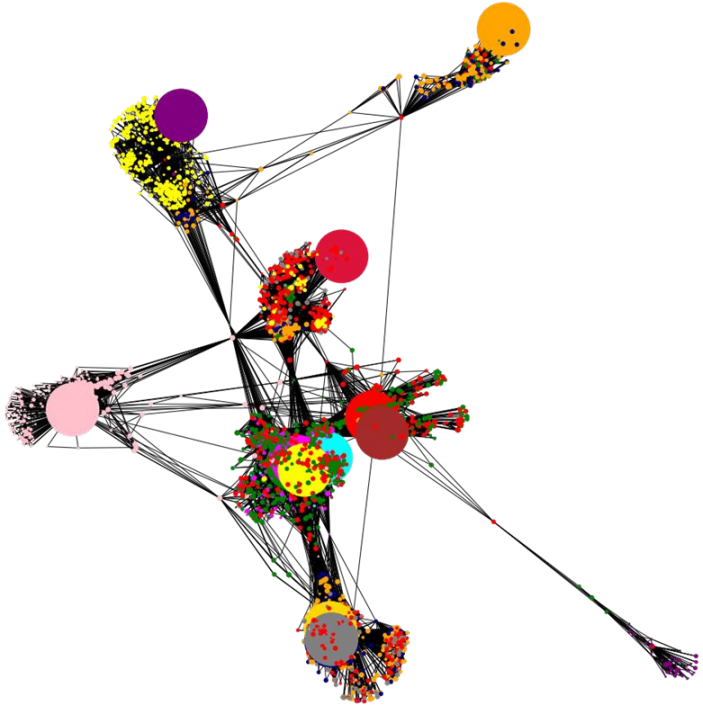


[Fig. Node Rankings with top 4 ranked nodes Viz.]

Results



[Fig. Different Communities in the Dataset Viz.]



[Fig. Influencing nodes per communityViz.]

Different Approaches to Link Prediction

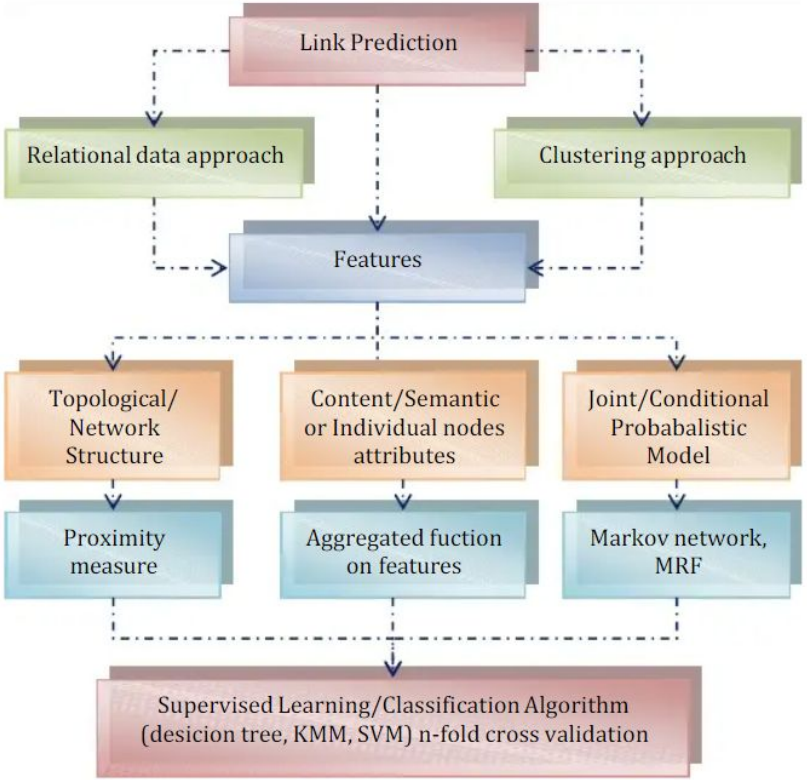
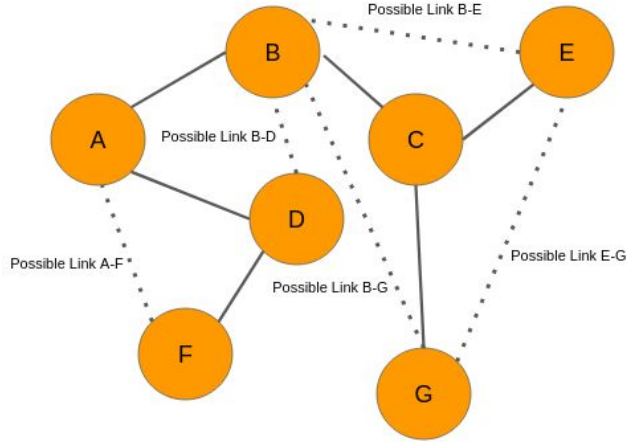


Image Source

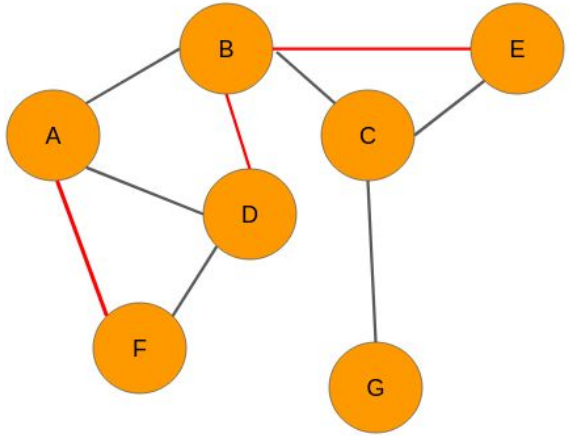
Strategy to Solve Link Prediction Problem

- Let's look at a dummy graph to better comprehend this concept. A 7-node graph is shown below, with the disconnected node-pairs AF, BD, BE, BG, and EG:



Graph at time t

- Let's imagine we analysed the data and came up with the graph below. A few new links (in red) have been established:



Graph at time t+n

Strategy to Solve Link Prediction Problem

- Our goal is to anticipate whether a link exists between any two unconnected nodes.
- We can extract the following node pairs with no linkages between them from the network at time t:
  - A-F
  - B-D
  - B-E
  - B-G
  - E-G
- The next stage for us is to design features for each pair of nodes. There are various approaches for extracting features from network nodes.
- Assume we utilise one of these strategies to create features for each of these couples. However, we are still unsure of the target variable.
- Examine the graph at time t+n. We can observe that the network has three new linkages for the pairs A-F, B-D, and BE, respectively. As a result, we'll give each of them a value of 1. Because there are no linkages between the nodes, the node pairings B-G and E-G will be assigned a value of 0.
- Now that we have the target variable, we can build a machine learning model using this data to perform link prediction.
- So, this is how we need to use social graphs at two different instances of time to extract the target variable, i.e., the presence of a link between a node pair.

Features	Link (Target Variable)
Features of A-F pair	1
Features of B-D pair	1
Features of B-E pair	1
Features of B-G pair	0
Features of E-G pair	0

Approach For Link Prediction (ML)

Building the Graph

- Used Networkx Library to build and visualize graphs.

Extracting the Features

- 5 features
  - **Jaccard Coefficient**
  - **Resource Allocation Index**
  - **Adamic Adar Index**
  - **Preferential Attachment**
  - **Common Neighbor centrality**

Model Training and Testing

- Implemented 3 Classical ML models.
  - SVM
  - Logistic Regression
  - ANN
- We have splitted the dataset into 4:1 ratio for training and testing data.
- Accuracy:
  - SVM : 64.06%
  - Logistic Regression:65.95%
  - ANN: 65.72%

Approach For Link Prediction (General Link prediction Algorithms)

Building the Graph

- Used Networkx Library to build and visualize graphs.

Using General link prediction algorithms

- Link prediction algorithms
  - **Jaccard Coefficient**
  - **Resource Allocation Index**
  - **Adamic Adar Index**
  - **Preferential Attachment**
  - **Common Neighbor centrality**
- For ego-network graph of node-0 all the above-mentioned link predictions algorithms are used.These algorithms returns u,v and p. The u and v are the nodes and p is the probability of link prediction between them.
- A threshold value is taken for link prediction i.e. p>= threshold value are only taken as future links in this case.
- Link prediction using Jaccard coefficient has also been implemented for a graph formed by ego-networks of node-0 and node-107.

Plotting graphs depicting link prediction

- Adding the predicted edges in the graph.
- Differentiating predicted edges from the existing edges by using different colors for nodes and edges (Blue for existing links and yellow for predicted link formations).
- Plotting the new graph created using pyvis python library.

Results (ML Approach)

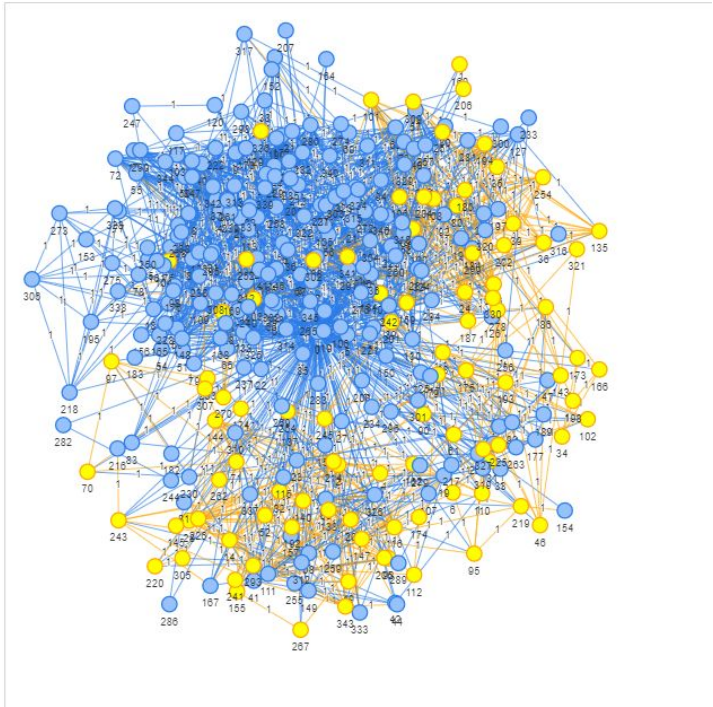
```
Accuracy of SVM Model: 0.640661938534279
```

```
Accuracy of Logistics Regression Model: 0.6595744680851063
```

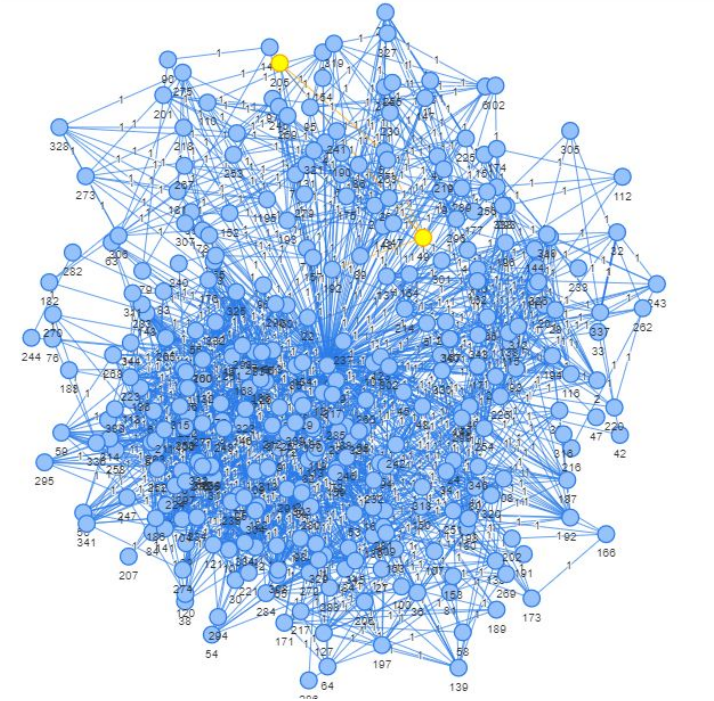
```
ANN Model Accuracy: 0.657210401891252 9
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/
  ConvergenceWarning,
```

[Fig. Link Prediction using ML models (SVM, LR, ANN)]

Results (General Link prediction Algorithms)

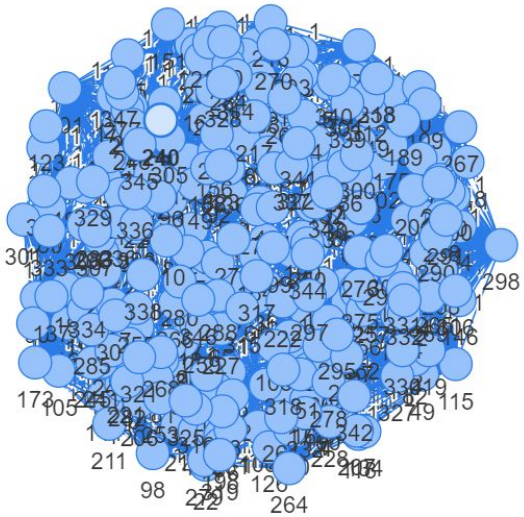


[Fig. Link Prediction using Jaccard Coefficient for Ego-Network of node 0 at threshold value=0.5]

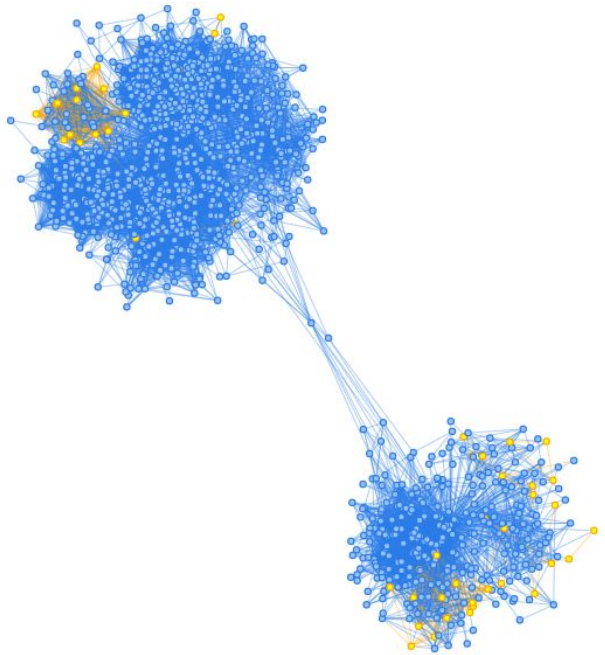


[Fig. Link Prediction using Jaccard Coefficient for Ego-Network of node-0 for threshold value=0.8]

Results (General Link prediction Algorithms)



[Fig. Link Prediction using other general link prediction algorithms at threshold value=0.5 and 0.8]



[Fig. Link Prediction using Jaccard Coefficient for graph made from the ego networks of node 0 and node-107 at threshold value=0.7]

References

- https://networkx.org/documentation/stable/reference/algorithms/link_prediction.html
- https://www.computerscijournal.org/vol6no2/composite-and-mutual-link-prediction-using-svm-in-social-networks-2/
- https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/resource-allocation/
- https://www.analyticsvidhya.com/blog/2020/01/link-prediction-how-to-predict-your-future-connections-on-facebook/
- https://aksakalli.github.io/2017/07/17/network-centrality-measures-and-their-visualization.html
- https://medium.com/social-media-theories-ethics-and-analytics/facebook-graph-analysis-using-networkx-1e0741138a37
- greedy_modularity_communities — NetworkX 2.8 documentation
- https://pyvis.readthedocs.io/en/latest/documentation.html
- https://pypi.org/project/mc4/
- https://pyvis.readthedocs.io/en/latest/documentation.html