Network Analysis and Link Prediction

UNIVERSITÀ DEGLI STUDI DI MILANO

LA STATALE

Deparment of Economics,Management
and Quantitative Methods
Data Science and Economics

Project of Social Network Analysis

Murat Aydın

December 2021

Abstract. This paper presents a network analysis using basic network analysis tools and algorithms on the social graph of Facebook TV pages where nodes represent the pages and the edges are mutual likes among them. The analysis is done in Python language using networkx library. We compute a bunch of features of the graph, like degree distribution, bridges, transitivity and assortativity. We then try to do link prediction using Logistic Regression and LightGBM algorithm which had %74 and %96 of test accuracy, respectively. The social network we use is fully connected, the "six degree of separation" is not confirmed. By studying the clustering coefficients, we show that while the whole network is relatively clustered well, the graph neighborhoods of users is way more sparse showing that the network shifts to be a information network rather than a social one. The community detection algorithms we used showed that the communities are based on cultural features, like where the TV shows are produced.

Facebook Graph. Figure 1 shows the network visualization obtained through Gephi. The layout algorithm is ForceAtlas2.

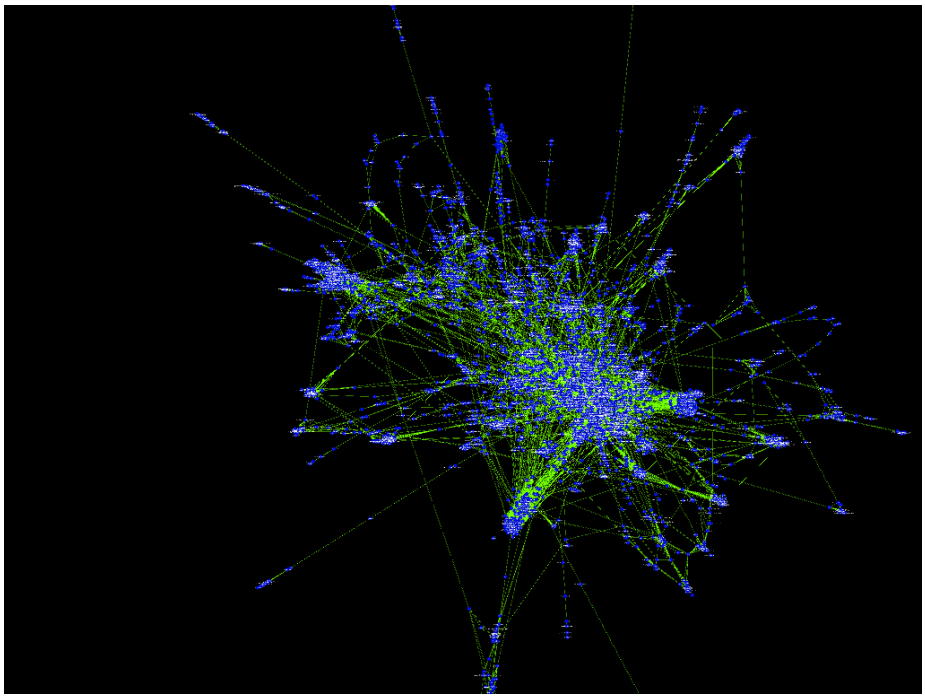


Figure 1: Network Visualization

We have an undirected network constisting of **3881** nodes and **17240** egdes, the density is around **0.00228** so it is not really a dense graph considering the maximum of number of edges that could be formed which is given by $N(N-1)/2$ for undirected networks where N is the number of nodes present in the graph.

Exploration of the Network. The degree of a node is the number of connections it has and their distribution is called scale-free. A scale-free network is a network whose degree distribution follows a power-law. Power law distributions follow a straight line(**linear dependence**) in log-log scale. When we plot the degree distribution, we avoid linear binning and the best way to plot them is to use ECDF and ECCDF. The ECCDF measures the number of users who have degree k or greater.

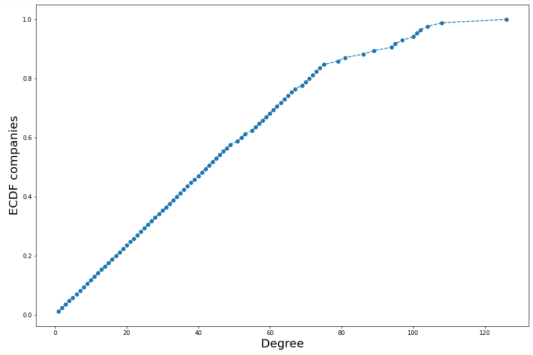


Figure 2: Python Output: Cumulative Distribution Function

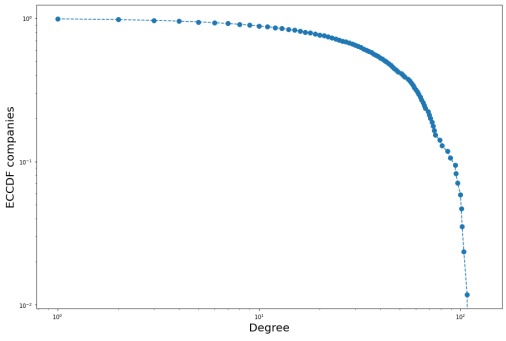


Figure 3: Empirical Complementary Cumulative Degree Distribution

In figure 2, Cumulative degree distribution is shown. As seen,

it follows almost a linear line in log-log scale showing that this is a scale-free or heavy-tailed or a pareto distribution.

Basic Network Features Most of the nodes have a moderate number of connections, the maximum connections is 126, the minimum is 1. The median value is 5 while the mean is 8.8 with standard deviation being 12.56. The assortativity is 0.56.

In order to understand if our network is a random object or has a specific structure, we compare it with a random network modelled by $G(N, P)$ where N is the number of nodes and P is the probability that a link exists between two nodes where probability equals to the density of the real network.

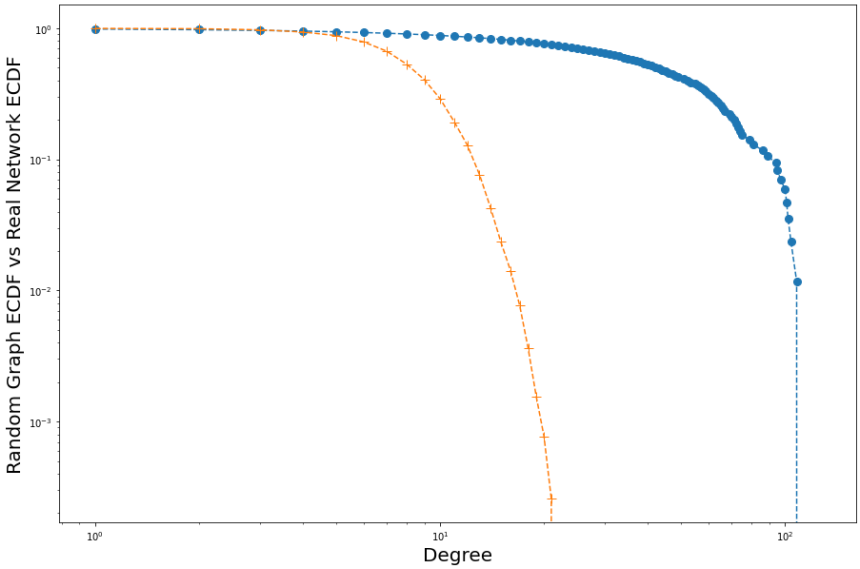


Figure 4: Random Network vs Real Network

Random Network Features. The random network characterized by the same number of nodes and links, 17240 and 3881 respectively has a maximum of 22 degrees and minimum 1. The mean degree is similar to that of real-network which is around 8.96 with a standard deviation 3.06 while the median is 9. The variance is lower in the random networks. Since they are modelled by Poission distriubtion which does not allow extreme values to live in, the

max and min value of degrees is way more lower compared to the real network. For example, the maximum degree in the real network is 126 but the probability that a node exists with a degree 126 is literally zero in random network.

Hubs. We have 39 hubs in our network at 99 percentile which corresponds to around 64 degrees. So nodes whose degree is over 64 are considered hubs in our network.

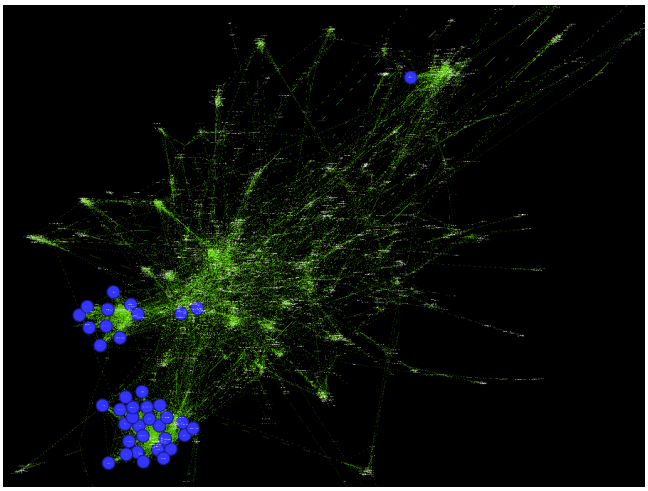


Figure 5: Hubs Gephi Visualization

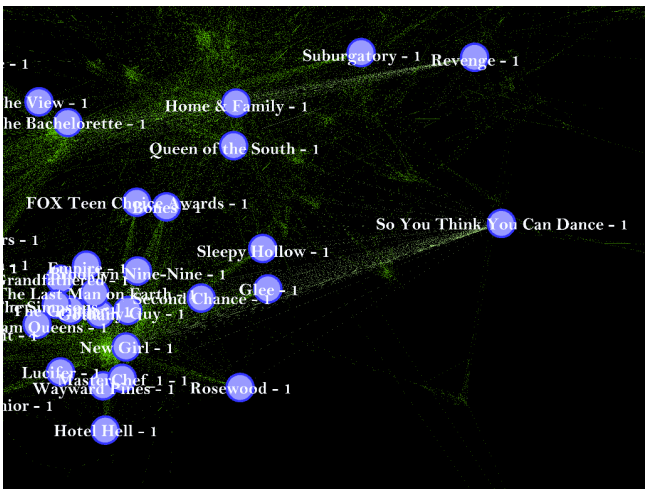


Figure 6: Hubs zoomed in

In figure 5, we provide a general picture of the network with hubs and in figure 6, we zoom in some of these hubs.

Bridges. An Edge between A and B is a bridge if, when deleted, it would make A and B lie in 2 different components. An Edge is a local bridge if its endpoints have no friends in common– if deleting the edge would increase the distance of the endpoints to a value more than 2. All the bridges are weak ties through which we shrink the diameter of our network meaning that we reduce the number of steps needed to visit all part of network. Bridges help us to spread the information through all the network, without bridges we could not reach all part of the network and without local bridges we would spend much more time to spread the information.

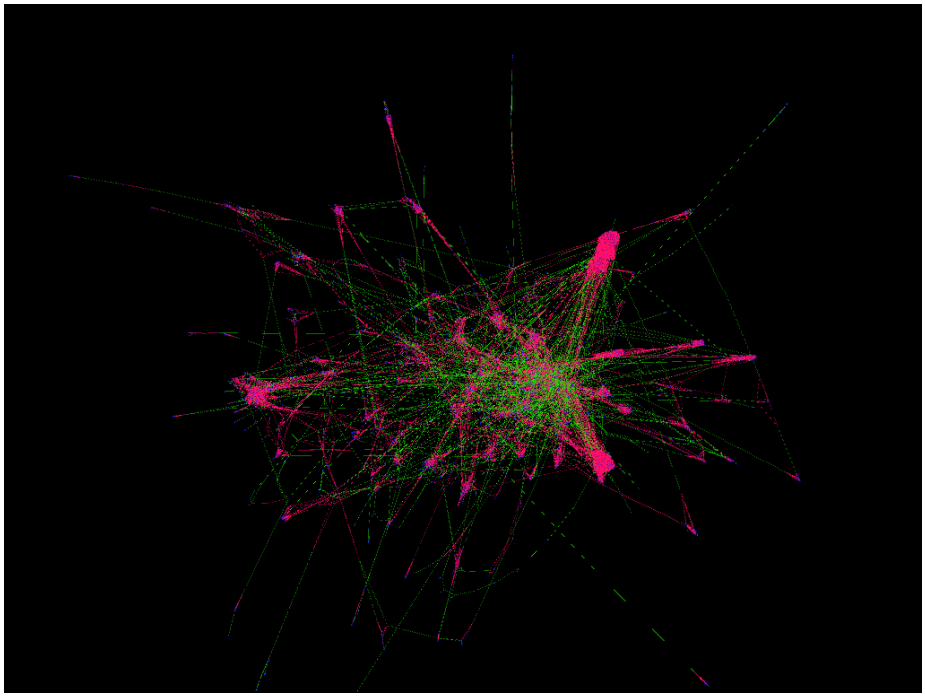


Figure 7: Local Bridges Gephi Visualization

Here in our network, we have 721 bridges meaning that deleting those 721 edges would create 721 different components in the network. However since they are a small fraction of the whole dataset,

visualizing them is a little complicated. We instead plot the local bridges in figure 7 where the green edges represent the local bridges while the pinkish ones represent normal nodes.

Transitivity. Perfect transitivity in Social networks occurs only when each component is a fully connected graph or clique where all the possible links should exist. This is not the case in real social networks and so the concept of perfect transitivity is useless since it never occurs. We instead study the partial transitivity which is the probability to have a link between two nodes on the basis of having a common friend. Global clustering coefficient is a way to measure such a transitivity in the network. The GCC coefficient of our network is **0.59** so we can say that our network is fairly transitive.

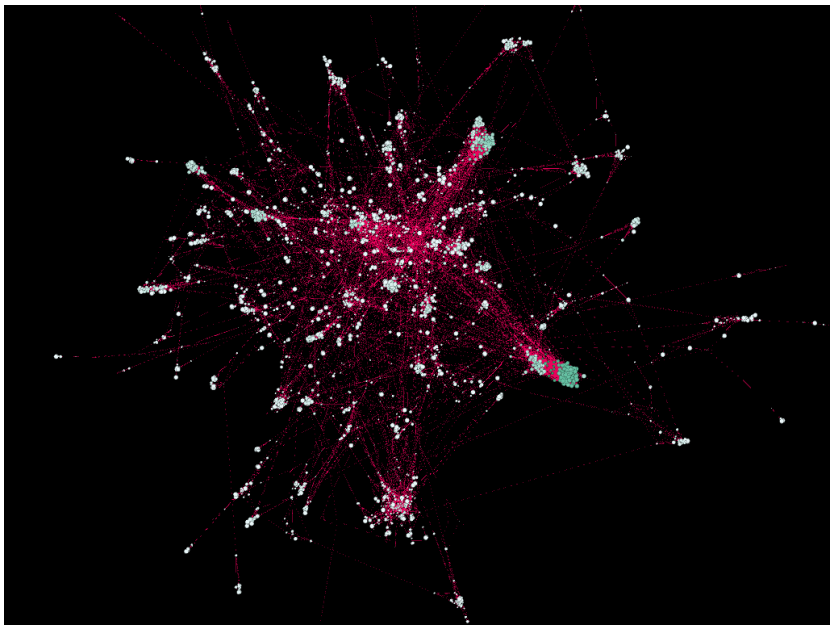


Figure 8: Local Clustering Coefficient Gephi Visualization

When we want to see the transitivity node by node, we calculate Local Clustering Coefficient. High LLC score means that the node having that score is surrounded by connections which are connected

to themselves as well meaning a very cohesive group. The average LCC in our network is 0.374. This scored is obtained through the sum of all single LCC scores divided by total number of nodes. So given these two scores, we see that the total network is more transitive than the networks of the single nodes. High values of average LCC might mean that the platfrom has more like a social aim while the low values implies that the platfrom is shifting to be an information network. In fact, the social network we have is about TV pages so it is not really like a facebook friendship network. So the people following these TV show pages are not friends with themselves, they do not follow each other in Facebook. They are there to get an information about the TV show. In figure 8, we see the LCC scores of single nodes.

Centrality Measures. These measures tell us how important a node within the network. In real world interaction, we consider people with many connections to be more important. This concept is modelled by the degree centrality.

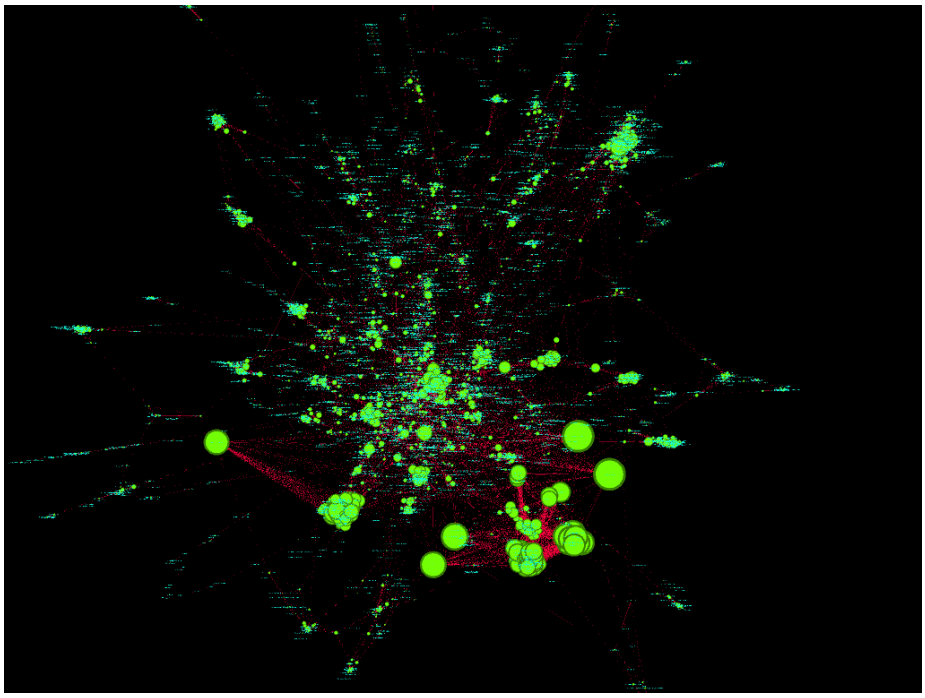


Figure 9: Degree Centrality Gephi Visualization

```
10 most important nodes for Degree Centrality:
('Home & Family', 0.032474226804123714)
('Queen of the South', 0.032474226804123714)
('So You Think You Can Dance', 0.027835051546391754)
('Glee', 0.026804123711340208)
('New Girl', 0.026288659793814433)
('Family Guy', 0.026030927835051548)
('The Simpsons', 0.025773195876288662)
('Dancing with the Stars', 0.025773195876288662)
('MasterChef_1', 0.025)
('Bones', 0.025)
```

Figure 10: Python output: 10 most important nodes by Degree Centrality

The degree centrality is a very naive approach. Sometimes having the most friends does not make one the most important person in the network. We sometimes care about who has the highest functionality in the network in terms of information spread? **Betweenness Centrality** gives us an idea about this. While the degree central-

ity measures how well-connected a node is, betweenness centrality shows us the extent that a node has control over the information flow in the network. How much information passes through a node in a given network? This question is answered by betweenness centrality.

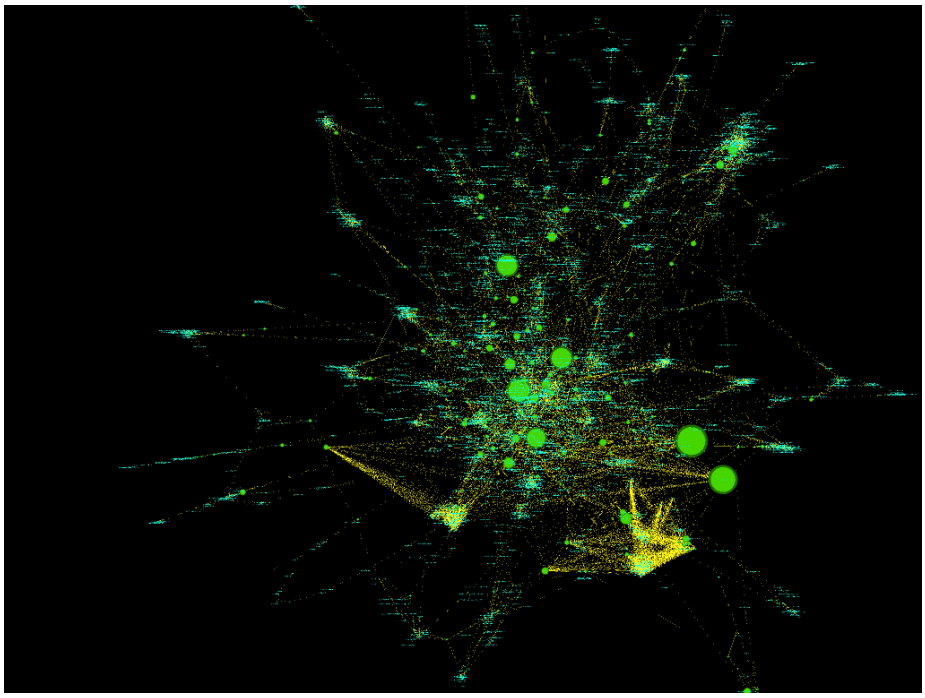


Figure 11: Betweenness Centrality Gephi Visualization

```
10 most important nodes for Betweennes Centrality:
('Queen of the South', 0.10558344153782458)
('Home & Family', 0.09361490900487743)
('The Tonight Show Starring Jimmy Fallon', 0.0807568339812378)
('The Voice', 0.07444201515082309)
('The Voice Global', 0.07435318455633814)
('Access', 0.06875519475475324)
('American Idol', 0.04015925066199617)
('Downton Abbey', 0.038981168075068386)
('The List', 0.03796034657286787)
('tagesschau', 0.03435903314134657)
```

Figure 12: Python output: 10 most important nodes by Betweennes Centrality

As we can see, the number of nodes having a high BC values is lower than that of nodes having high DC values since BC is looking for something way more specific than DC. If we look at figure 10 and 12, we see that there is a small overlap between most important

nodes given by BC and DC. In particular,**Queen of the South and Home & Family** is in the both list. These are very important nodes in the sense that if there is any rumour in the network, they are the ones which are very likely to know about it.

EigenVector Centrality on the other hand is with us to deepen the concept of Degree Centrality that is too naive. It is an extension of DC and it tells us that not all connects are equivalently important, some of them should be more important than others.

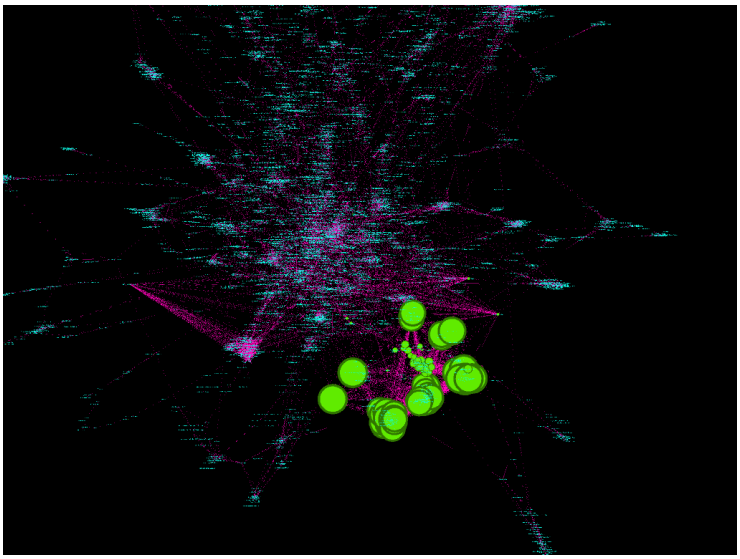


Figure 13: EigenVector Centrality Gephi Visualization

```
10 most important nodes for Eigenvector Centrality:
('So You Think You Can Dance', 0.1385223057268764)
('New Girl', 0.1384828196082673)
('Bones', 0.13726421537552602)
('MasterChef_1', 0.13690301639674096)
('Family Guy', 0.13682999194136913)
("Bob's Burgers", 0.13667617567220133)
('The Simpsons', 0.13647646937162283)
('Brooklyn Nine-Nine', 0.13641567701984297)
('Glee', 0.13617883937519476)
("Hell's Kitchen", 0.13587525865254133)
```

Figure 14: Python output: 10 most important nodes by EigenVector Centrality

There is no overlap between EVC and BC but the result of EVC and DC is quite similar. These 10 nodes have the most important connections in the network.

Community Detection. We are sometimes interested in the communities hidden and embedded in the network. They are not easy to find at one glance since the network structures, in general, are highly complicated objects and even the best visualization techniques do not allow human eye to catch them. We will use **Greedy, Louvain and LPA** algorithms to detect the communities if any. After presenting all of them, we will also compare their performances by means of modularity.

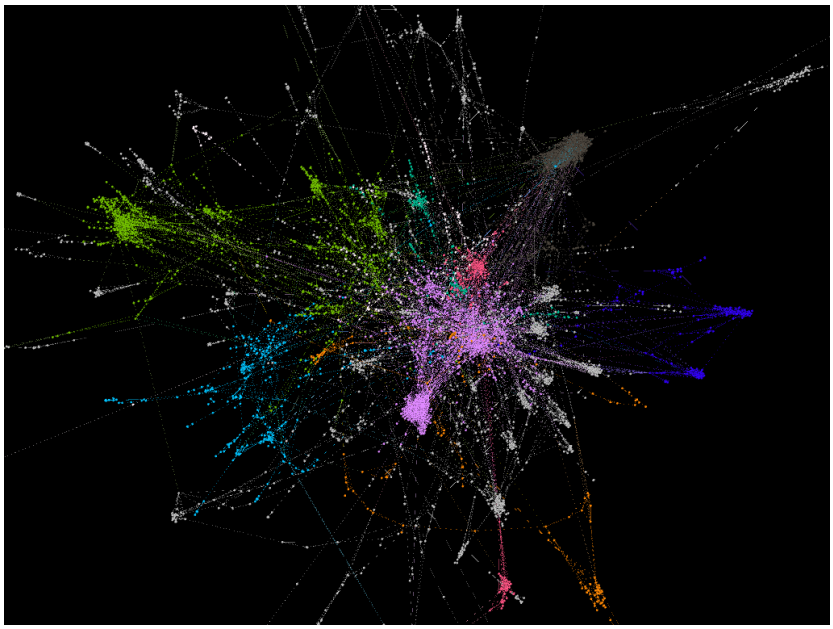


Figure 15: Greedy Partition Gephi Visualization

In figure 15, we show the communities obtained by the Greedy Partition. With this algorithm, we obtained 58 communities in the network. First ten communities have nodes higher than 100. We will not go deeper in the analysis of the features of these TV shows, however our preliminary examination showed that there is no a particular sensible relationship between them in terms of the type of a Tv Programme but rather, the relationship between them is cultural and geographical. For example, The Purple Group is American made shows show in figure 16, The light blue group is French made shows shown in figure 17 while The Green group is Germany originated shows shown in figure 18.

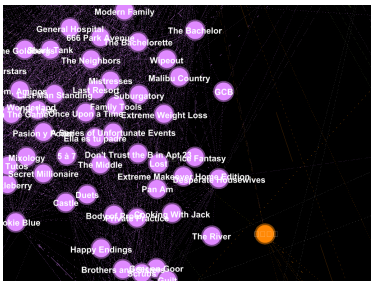


Figure 16: Greedy Partition American Community Gephi Visualization

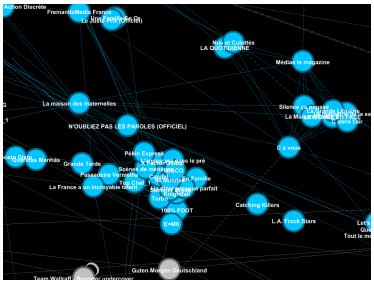


Figure 17: Greedy Partition French Community Gephi Visualization

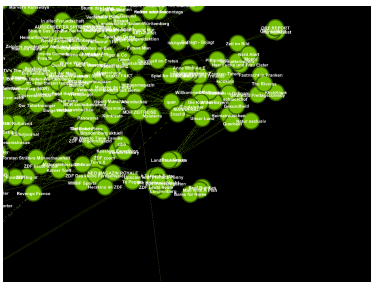


Figure 18: Greedy Partition German Community Gephi Visualization

The other communities can be examined and visualized in the same fashion taking into account that the communities are not formed by the features of the TV shows but rather on cultural and racial basis.

Louvain Algorithm. Louvain is another community detection algortihm which generally performs better than the Greedy Approach.

Louvain algorithm detected 45 communities, slightly lower than

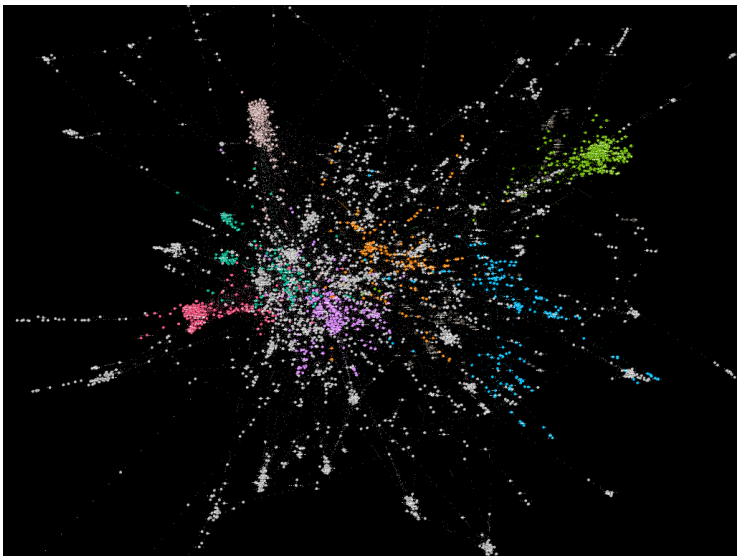


Figure 19: Louvain Gephi Visualization

that of Greedy Approach. However the logic behind these clusterings is the same. There is now also a cluster of a big Canadian-American TV company, A & E, showed in figure 20.

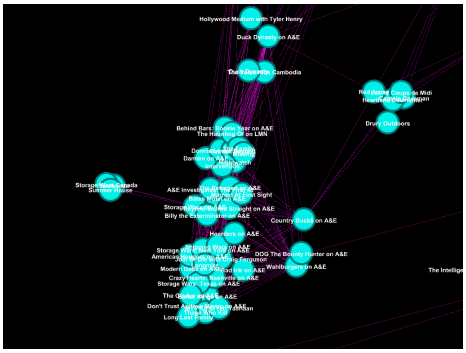


Figure 20: community of A & E company Gephi Visualization

LPA Algorithm. This algorithm detected 406 communities which is way higher than the first two algorithms. In figure 21, we show the partition given by LPA and in figure 22, we present the performances of the algorithms.

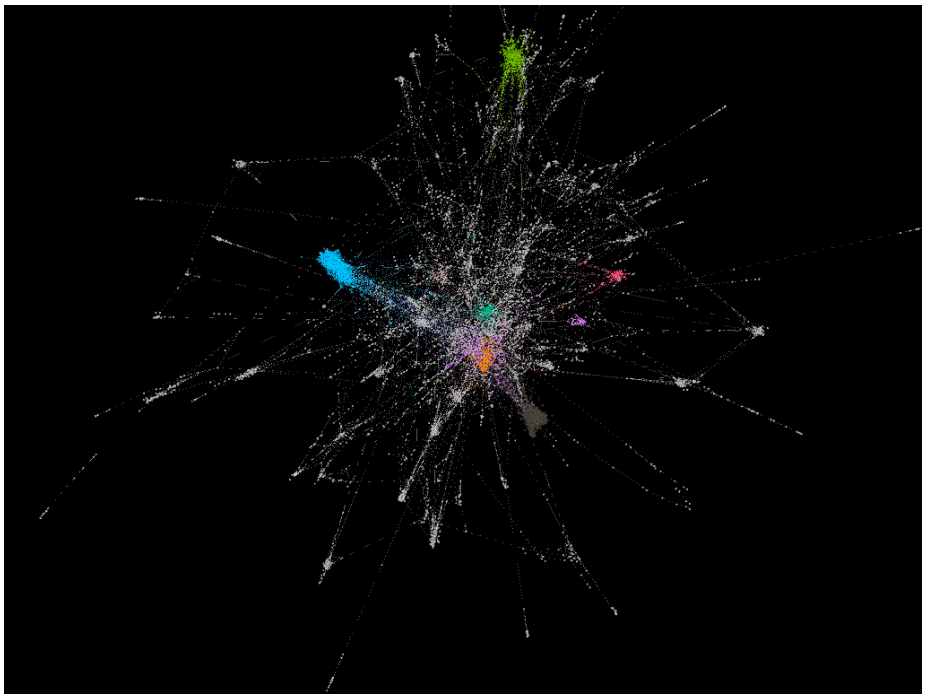


Figure 21: LPA partition Gephi Visualization

```
Greedy

Coverage 0.9314965197215778
Modularity 0.8271650837635454
Performance 0.9337122168003251
------------------------------------------------------------

Louvain library

Coverage 0.9327146171693735
Modularity 0.8707276655218265
Performance 0.9694825703865249
------------------------------------------------------------

LPA

Coverage 0.859338747099768
Modularity 0.8129397653436413
Performance 0.9922922139845985
------------------------------------------------------------
```

Figure 22: Python Output: Performance of Algorithms

The performance of a detection algorithm is evaluated by the modularity score. The highest modularity is reached by the **Lou-**

vain Algorithm.

Link Prediction. Link prediction is one of the most important research topics in the field of graphs and networks. The objective of link prediction is to identify pairs of nodes that will either form a link or not in the future. To do the prediction, we firstly should create a training set from the graph at hand. The technique we will apply is the following :

1) Remove some of the edges at random that do not lead to isolated nodes. So at the end of this operation, the graph should remain connected as before.

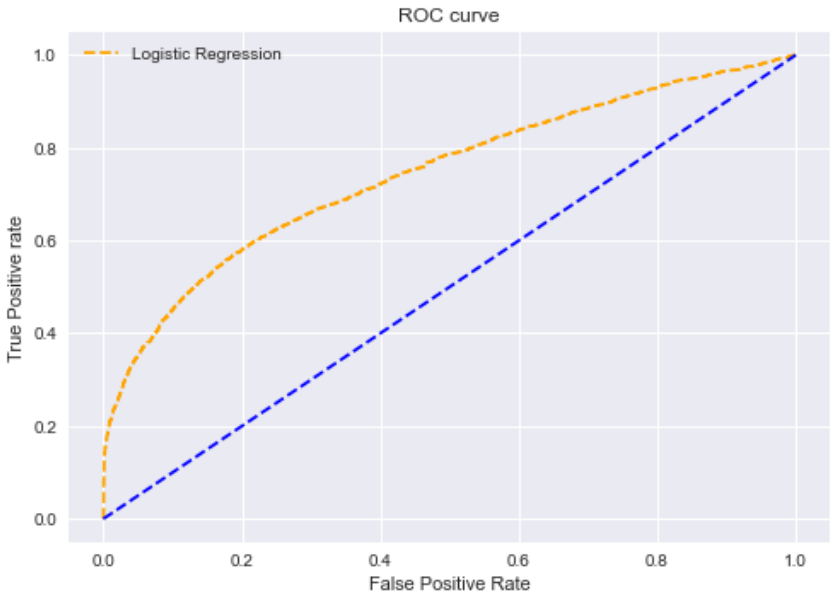
2) Removed labels will be labelled as 1 and the unconnected pairs as 0. The reasoning behind this method is that we already know that the removed edges exist at time t in the graph G. By removing edges, we try to replicate the graph at time t-1 knowing the current situation of the graph at time t. By labelling them as positive samples, we create a fake training set hoping that this random removings would be able to replicate the graph at time t-1. The idea behind labelling the current unconnected nodes as 0 is a way to tell the algorithm that such nodes are behaving as a negative example because we could not observe any connection between them at time t. There might be thousands of possible combinations of nodes that are not connected. To do this computation efficiently, we can use the nodes having shortest path lower than 2 (so that we reduce the number of unconnected pairs, otherwise the training set will be highly imbalanced) that do not lead to an isolated graph.

3)After creating our training set, we need a way to extract information from the nodes, namely the features. Features here refer to the features in classical machine learning, the feature space like age,education,weight etc. Such features are not given us by the graph but rather are obtained through an algorithm called **Node2Vect.**

4) After the third step, we are ready. We will have a dataset consisting of a bunch of features given by the algorithm Node2Vec for each node and their respective classes, positive or negative. And eventually, with this dataset, we feed any desired learning algorithm

to do the prediction.

Result. In figure 23, we present the accuracy of Logistic regression along with the ROC curve.




```
predictions = lr.predict_proba(xtest)
roc_auc_score(ytest, predictions[:,1])
```

```
0.7435256004788587
```

Figure 23: Python Output: ROC curve and Logistic Regression Accuracy

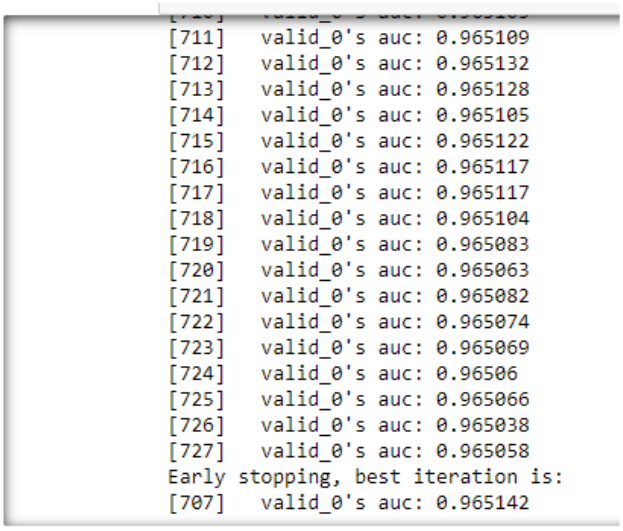


Figure 24: Python Output: Light Gradient Machine Boosting Accuracy

In figure 24, we present the result of LGBM algorithm that reached an impressive accuracy of %96.

Conclusion

We tried to analyse Facebook TV Pages network graph. We saw that average local clustering coefficient was low while the global clustering coefficient was high, relatively. This indicates that the network shifts to an information network than a social one. We used a bunch of community detection algorithms. They detected the communities not based on the features of TV shows but rather on cultural basis. The communities were separated based on which country they are produced. The best detection algorithm turned out be the Louvain Algorithm, this is shown by the modularity metric in figure 21. We then did a link prediction and reached an impressive accuracy of %96 by LGBM algorithm.