**Graph Analysis:  
Friend Recommendation System**

**Members:**

Himani Thakkar (ht2191)

Hrituja Khatavkar (hk3219)

**Acknowledgement**

We take this opportunity to express our sincere and heartfelt gratitude to our professor, Prof. Gustavo Sandoval for imparting us the knowledge of Mining Massive Datasets and all the intricacies and workings of developing projects in the Data Mining world.

We would also like to acknowledge with much appreciation our teaching assistants, Hriditaa Dekate, Lenny Remache, and Xinru Li for their support and guidance.

We will surely keep this experience in mind as we move forward and begin working on similar projects in the industry.

**Table of Contents**

| 1. | Introduction ………………………………………………….………… | 3 |
| --- | --- | --- |
| 2  3. | Dataset …………………………………………………………………  Tools and Techniques ………………………………………………... | 4  5 |
| 4. | Implementation ……………………………………………………….. | 6 |
|  | 4.1 Exploratory Data Analysis …….………………………………… | 6 |
|  | 4.2 Prep Code ……………………………….……………………….. | 9 |
|  | 4.3 Featurization ……………..…....…………………………………. | 9 |
|  | 4.4 Model Building ……………………………………………….…... | 11 |
| 5. | Conclusion …………………………………………………………….. | 12 |
| 6. | Future Scope ………………………………………………………….. | 14 |
| 7. | References …………………………………………………………….. | 15 |

1. **Introduction**

Today, we see various social media platforms offer a feature that suggests new connections and recommends pages/people based on a user and its attributes. Having such a system in place, not only allows people to connect with one another easily but also facilitates businesses to maximize their social influence. In our project, we are trying to replicate this feature by performing graph analysis to predict missing connections as recommendations.

We must keep in mind that the network is evolving over time, new users are getting added to the network, and every day new connections are being made and broken. Based on the current network we want to be able to predict the upcoming changes in the network and make recommendations accordingly.

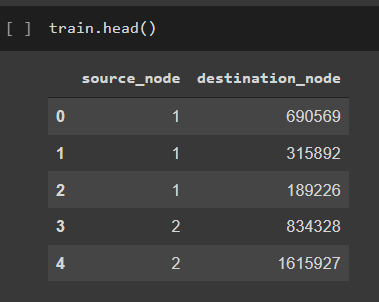
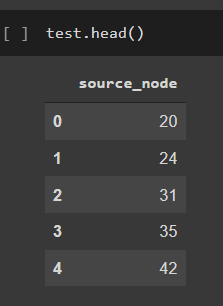
It has been observed that users are likely to connect with new users if there are some mutual connections already established. Our main goal, hence, is to recommend to users based on different features and analyze which features are most to least important for a link to be present and recommended.

1. **Dataset**

Our data is obtained from Facebook’s recruiting challenge on Kaggle. This dataset is a directed graph consisting of 1.86M nodes and 9.44M edges. We will be using train.csv and test.csv from the following link to complete our project.

<https://www.kaggle.com/c/FacebookRecruiting>

The dataset consists of two columns - source and destination. Each row in the dataset is an edge in the graph.

1. **Tools and Techniques**

To achieve our goal, we have used the following flow of processes.

****

Firstly we have performed extensive exploratory data analysis and manipulated the data in prep code before proceeding to the featurization and modeling.

We then use multiple parameters as features and input into our Random Forest classifier for modeling and predictions. We finally conclude with most ot least important features.

For successful completion of this project, we have used multiple python libraries, some of which along with their uses are stated below:

* NetworkX: For analysis and building the graph from the dataset we have used the NetworkX library. Using this library we have visualized the data and also manipulated it according to the requirements of the problem statement.
* Pandas: To manipulate and work with massive data
* MatplotLib: To help us visualize data and perform Exploratory Data Analysis
* Pickle: To maintain the program state across sessions

1. **Implementation**

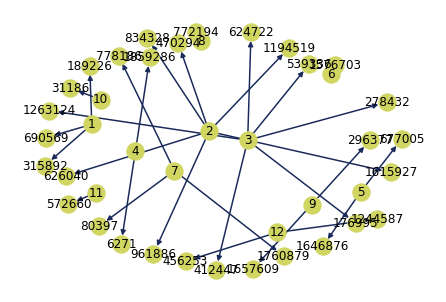
**4.1 Exploratory Data Analysis**

In this step, we try and find out as much about the data as possible.

1. Visualization

Initially, we tried to understand the input data and it’s distribution visually as this helps us to interpret the data better and have a better understanding. We thus sampled our dataset for first 30 rows and created the below graph for better visualization.

As we can see, this is a directed graph with nodes representing users and edges representing connections in an ordered way. An edge from a source node (user 1) to an destination node (user 2) represents, that the source node follows destination node. (i.e. user 1 follows user 2)



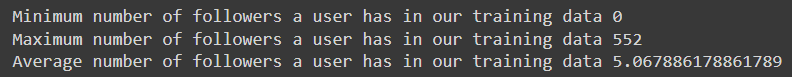
1. Number of vertexes

Our entire dataset contains 1862220 unique nodes (users).

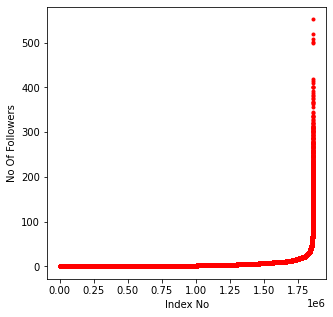


1. Indegree

In-degree of a vertex is the number of edges coming to the vertex. This tells us about the number of followers a user has

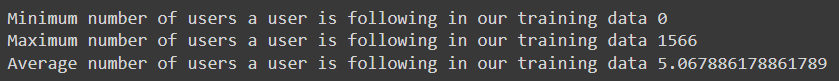


The following graph helps us visualize the distribution of data. As we can see that there is only 1 user with maximum number of followers, i.e. 552 and very few users have near maximum value. This can indicate that such users are popular, or maybe influencers / celebrities.

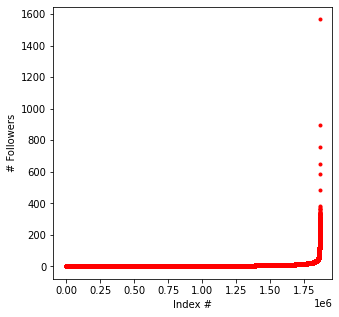


1. Outdegree

Out-degree of a vertex is the number edges which are coming out from the vertex. This tells us about the number of users a particular user follows, i.e. number of followee a user has.

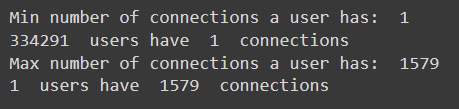


The following graph helps us visualize the distribution of data. As we can see that there is only 1 user with maximum number of followees, i.e. 1566 and very few users have near maximum value. This can indicate that such users follow a lot of people.



1. Graph connections

Here we are trying to find out how well connected the graph is. As we can see from the below output screenshot, the graph contains 334291 users that have only 1 connections.



To understand more about the connectivity of the graph, we try and find out the number of weakly connected components in the graph. A weakly connected component is a subgraph that is unreachable from other nodes/vertices of a graph or subgraph. It is important to know about how many such subgraphs we have as it can play a vital role in recommending people across and within such subgraphs.

As seen in the below screenshot, our graph contains 45558 such weakly connected subgraphs out of which 32195 are subgraphs with only 2 nodes.



**4.2 PrepCode**

After exploring the data and its limit, we now perform 2 important steps before we can proceed with featuring and modeling.

These include finding the missing edges of the graph and splitting the training dataset into train nad test (validation).

i] Finding missing edges: Here we manipulate the input data to create a table such that each node represents a non-edge or a missing edge in the graph. A missing edge means that there is no direct connection between the source and destination node. By doing this we are able to create a sample set of all the solutions we may have and can recommend.

ii] Splitting the training dataset.

We also split our dataset into train and test (validation) set with an 80-20 split, meaning 80% of input dataset is now are actual training dataset and 20% of the remaining data is now our testing/validation data. By performing this step, we get a chance to test our classifier with the actual value and we can also avoid/test overfitting.

**4.3 Featurization**

In this step, we find out multiple features that may predict if a link between the two nodes could be present or not. These features determine how likely is user 1 to know and follow user 2.

Following is the list of features we have calculated and analysed.

* Cosine Distance

Cosine distance is a metric used to measure how similar the documents are irrespective of their size. Cosine distance can be thought of as a complement of cosine similarity. Similarity increases when distance between two points decreases and vice versa. Here we are calculating the cosine distances for followers and followees of all the node pairs.

* Page Rank

PageRank is a popular algorithm that Google uses to search and rank pages on the web to provide relevant results in their search engine results page. PageRank works by counting the number and quality of links to a page (nodes in this case) to determine a rough estimate of how important the website (node) is. Consider a user A to be very important, i.e A has a lot of connections. B and C are A’s friends. The probability that C would be B's friend is higher than of D who has less connections.

Here, we want to calculate the PageRank around a each node pair (source and destination) in the training set.

* Jaccard Distance

The jaccard Index measures the similarity between finite sets, and is defined as the size of intersection divided by the union of the sample sets. The Jaccard distance which measures the similarity between sample sets is complimentary to jaccard index and is obtained by subtracting the jaccard index by 1. We will be calculating the jaccard distance between the followers and followees of each node pair in the training set.

* Follows Back?

To check if a person follows the other person back or not.

* HITS Score

Hyper-link induced topic search (HITS) identifies good authorities and hubs for a topic by assigning two numbers to a node : an authority and a hub weight. Authorities estimate the node value based on the incoming links. Hubs estimates the node value based on outgoing links.\\

Here, in this context, it is used for understanding the importance of the node based on incoming link (hub) and outgoing link (authority)

* Shortest Path

Shortest path is the path between two nodes such that the sum of their weights is minimum. We are calculating the shortest path between every node pair in our data.

* Checking for same community

Each weakly connected subgraph is a community. As discussed in exploration there might be some nodes that are disconnected from other nodes. Hence this becomes an important feature for our model. Here we are checking if node u and v belong to the same community (subgraph) or not.

We then map all these features for our training and test data and save them for our next step - model building.

**4.4 Model Building**

We now build and train our machine learning model using the Random Forest Classifier. It consists of a large number of individual decision trees that operate as an ensemble.

Each individual tree in the random forest spits out a class prediction the class with the most votes becomes our model’s prediction.

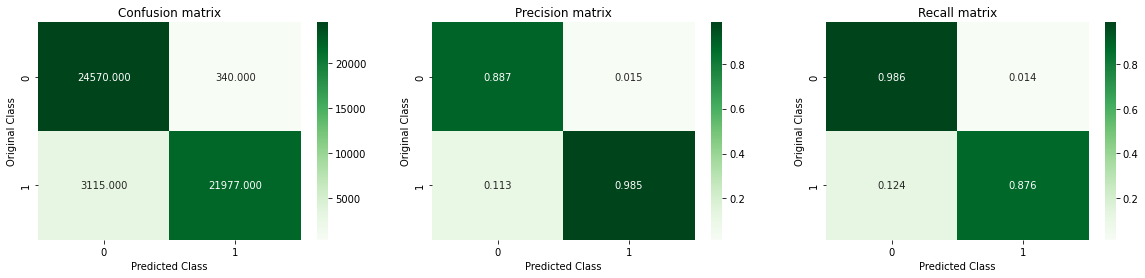
Why Random Forest?

* It gives a higher accuracy through cross validation.
* Random forest classifier can handle the missing values and maintain the accuracy of a large data.
* If there are more trees, it doesn’t allow over-fitting trees in the model.
* It has the ability to work on a large data set with higher dimensionality.
* For our project, since the data is huge, random forest was the best choice

1. **Conclusion**

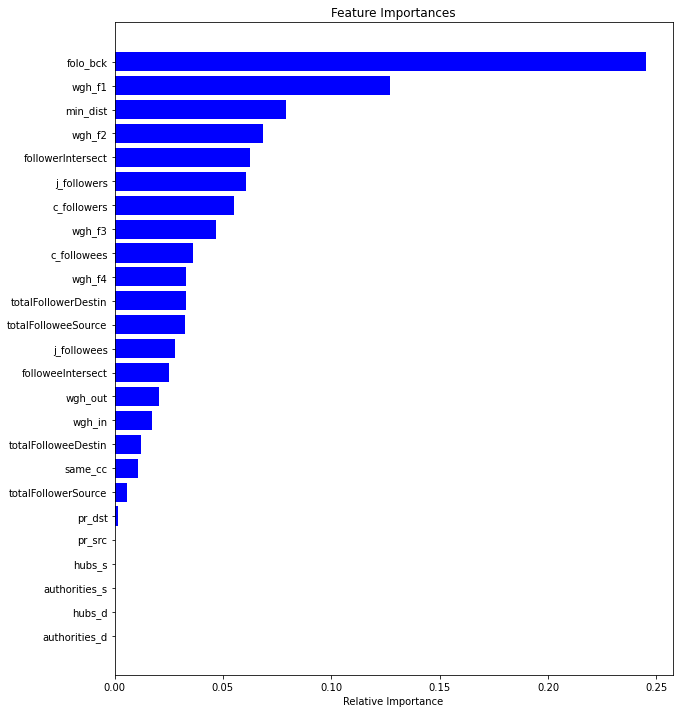
We now plot the confusion matrix for our test data predictions to analyse model output.

A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known.

****

We see that our model classify 24570 positive points as truly positive points and 21977 negative points as truly negative in our test data. Some points are misclassified by our model which means our model is not overfitted. We have also demonstrated Precision and Recall matrx in a heat map form for better visualization.

We then also plot a graph which shows the importance of each feature in predicting the edge.

****

As we can see that follow back is the most important feature for predicting an edge.

1. **Future Scope**

* This system can be extended to find the nth connection like in LinkedIn and hence even further improve it’s prediction.
* Metadata of the nodes for example demographics of a user can also be used in improving the results
* Further link analysis algorithms like the The SALSA Algorithm and Randomized HITS can also be used to compare the result.

1. **References**
2. <https://medium.com/analytics-vidhya/evaluating-a-random-forest-model-9d165595ad56#:~:text=F1%20score%20is%20a%20little,low%2C%20F1%20will%20be%20low>.
3. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
4. <https://stackoverflow.com/questions/9402255/drawing-a-huge-graph-with-networkx-and-matplotlib>
5. <https://vgnshiyer.medium.com/link-prediction-in-a-social-network-df230c3d85e6>
6. <https://medium.com/analytics-vidhya/facebook-recommendation-system-case-study-8dfc3ff5ddcc>
7. <https://medium.com/@gorerohan15/link-prediction-in-social-networks-599e6d9bed9b>
8. <https://matplotlib.org/3.5.0/tutorials/introductory/pyplot.html>
9. <https://networkx.guide/algorithms/components/weakly-connected-components/>
10. <https://realpython.com/train-test-split-python-data/>
11. <https://medium.datadriveninvestor.com/cosine-similarity-cosine-distance-6571387f9bf8>
12. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
13. <https://www.log2base2.com/data-structures/graph/degree-of-each-vertex-in-the-graph.html>