1 Introduction

1.1 Problem:

Project 1 is a graph based machine learning problem, which is a real-world social network scenario. Example, if B is a common friend of A and C, then what is the probability of A knowing C (existence of an edge between A and C). Where the edge can be real or fake. The dataset we have consisted of more than 2 million nodes, with 7 million positive edges and approximately 14 million negative edges.

1.2 Importance of link prediction:

Link prediction can be used to detect suspicious groups on a network (terrorist network). A vector space model can be converted to a graph model to utilise the availability of numerous similarity measures. Commercially this technique can be used for identifying user group for target marketing.

1.3 Environment Setup:

Phase one (experiment and exploration) was done using Macintosh, Windows and Ubuntu. Final results were obtained using Ubuntu virtual machine on NectarCloud.

Environment Configuration	Data set files	Main Libraries
Ubuntu 16.04 LTS(Xenial) amd64	Test Set : test-public.txt	NumPy, Scipy
Disc space: 30GB	Prediction format sample.csv	Pandas, SciKit
RAM: 6 GB, Virtual CPU's : 2	Data Set : train.txt	itertoold, random

Table 1: Resources

2 The Experiment

Link prediction paper[1] provided us with an idea of using NetworkX library to create a graph. The result was not a success, as the process of graph creation was killed in Nectar, due to memory limitation. This failure lead to identifications of nodes with 100,000 edges and above. As a result we removed few nodes, one of which had 700,000 edges and few of them had edges more than 100,000 in the count.

2.1 Sampling

We switched to Json representation of nodes, using nodes as keys and edges as values.

Characteristics of positive sample	Characteristic of Negative Sample
Edge between 2 node, common neighbours between nodes	Node should not be a source
	Degree of node should be small
20% of test-public.txt nodes included in sample	Degree of source to that node should me small

Table 2: Identification of positive and negative sample

As we see above that negative edges are double to positive edges, we made sure that the training sample has an almost equal distribution of negative and positive edges, thus balancing class distribution.

[1] https://www.researchgate.net/publication/220876141_Link_Prediction_in_Social_Networks_Using_Computationally_Efficient_Topologi

2.2 Feature Engineering

Knowledge from lectures provided us with ideas to deal with this project. First, to convert the graph to vector format using nod2vec and then apply matrix representation for solving the problem. Second, generate feature values and learn a model, followed by prediction. We used feature extraction with the utilization of training model concepts.

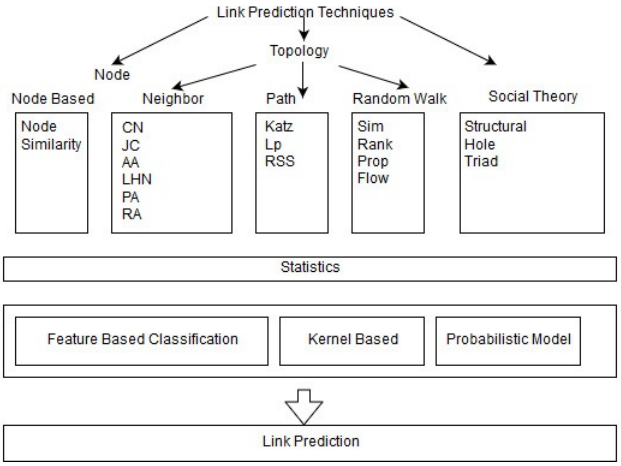

Figure 1: Link Prediction Model

Feature	Accuracy	
1	Jaccard Coefficient	0.82
2	Hub Depressed	0.81
3	Sorensen Index	0.77
4	Adamic Adar	0.79
5	Salton Cosine Smilarity	0.84
6	Leicht-Holme-NErman	0.5
7	Preferrential Attachment	0.5
8	Resource Allocation	0.77
9	Salton Cosine 2	0.85
10	Salton Cosine 3	0.85
11	Common Neighbours	0.52

Table 3: Features with Accuracy

Let x,y be two node, N(x) are nodes adjacent to x then

$$JaccardCoefficient = \frac{|x \cap y|}{|x \cup y|} \qquad Adamic/Adar = \sum_{u \epsilon N(x) \cap N(y)} \frac{1}{log|N(u)|} \tag{1}$$

Our solution is based on topology approach, where we identify the neighbors of a node. In real life we can think about it as people tend to make friends with people who are close to them. The features which we utilized have been mentioned in Table 3, along with the representation of their accuracy. An example of such feature is Jaccard Coefficient, and this measure captures the notion that proportion of friends of A who are also friends of B is a good measure of similarity between A and B. Formula and definition for all the other features can be found here [Wan+14]

2.3 Machine Learning: Prediction Modelling

The algorithms we selected including Multi-layer Perceptron, Random Forest, KNN, Naive Bayes and decision tree. The setup for modelling is implemented on NectarCloud, where we used Anaconda and used its base virtual environment. The coding was done using python3 with all the required libraries configured in the environment.

2.3.1 Random Forest

The AUC of RF is 0.85. We selected Random Forest as our modeling algorithm because it is used for both regression and classification. Also because our data set is huge by creating a large number of tree, i.e. a huge forest with a random distribution, we can improve the accuracy of prediction. Lastly, it can be used on pairs of data. Implementation of Random Forest was inspired from here [2]
Cross Validation of model using 10-folds technique and 5 features (Jaccard, SaltonCosine, AdamicAdar/HD/HP):
Correctly Classified Instances:95%, Mean Absolute Error:0.0803, Root Mean Square Error:0.2026,

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.933	0.027	0.977	0.933	0.954	0.903	0.979	0.987	positive
	0.973	0.067	0.923	0.973	0.947	0.903	0.979	0.9657	negative
Avg.	0.951	0.045	0.952	0.951	0.951	0.903	0.979	0.977	

[2] https://machinelearningmastery.com/implement-random-forest-scratch-python/

Parameter Setting :- Bag Size: 90 | Number of Iterations: 120 | Seed Value : 2
Input Set for training: 10000 positive edge samples, 10000 negative edge sample and feature 1,2,4,5,7,8 from Table 3. Result was 85.063 percent accuracy in prediction of fake and real edges.

2.3.2 Multi Layer Perceptron

The AUC of MLP is 0.82. Because it provides flexibility or having multiple layers, which helps in getting better accuracy with more number of features.
Cross Validation of Model using 10 folds:

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.894	0.047	0.958	0.894	0.925	0.843	0.961	0.976	positive
	0.953	0.106	0.882	0.953	0.916	0.843	0.961	0.919	negative
Avg.	0.920	0.074	0.923	0.920	0.921	0.843	0.961	0.950	

2.3.3 Naive Bayes

The final AUC of NB we get 0.75. We find the reason that NB is less sensitive to missing data and the algorithm is relatively simple. Since the original train set does not contain any fake edges, so the way to generate negative samples will affect the prediction.

2.3.4 KNN

The AUC of KNN we get 0.82. The key to determining the performance of the KNN algorithm is to find a suitable K value. However, it is difficult to find a suitable K value according to the features we selected. Only AA and RA have a clear boundary. For Jaca, Salton, HD, HP and LHN, the boundaries between classes are blurred. The KNN algorithm is less suitable because of the feature selection.

2.3.5 Decision Tree

The AUC of the DT is 0.80. The challenge we facing here is how to finding the best depth for a DecisionTree. We have the over fitting problem: high score on training data but low score on validation.

3 Challenges Faced in the experiment

NetworkX makes life easier for graph based problems, but it is practically impossible to store 20 millions edge information in a variable, we tried cloud and other techniques to make graph using NetowrkX but it didn't work out. On other hand with the approach we took, of writing data as key value pairs to file is also less efficient, unless implemented using compression and multi threading approach.

4 What could have been done better.

Given enough time we could apply Split Finding Algorithm for providing default direction to sprase data on tree. Also we can use compression and multi threading to process huge data set available. One factor that could have been easily applied was sorting data samples on basis of feature values.

5 Conclusion

To conclude the report, we faced a high volume dataset and got a good understanding that there is a lot more to machine learning then just training a model using available algorithms. By being a part of

this activity we now have a clear understanding of how we should plan a course of action provided a machine learning problem. We have learned how to validate out generated models, how to implement a machine learning algorithm using python platform. That said this assignment was an interesting assignment, because now we have an understanding how Facebook friend suggestion might be working or how corp orates identify their target segment for marketing a product.

References

[Wan+14] Peng Wang et al. "Link Prediction in Social Networks: the State-of-the-Art". In: *CoRR* abs/1411.5118 (2014). arXiv: 1411.5118. URL: http://arxiv.org/abs/1411.5118.