MAVERICKS | KUNAL WASNIK 22393 | SUBHASIS PANDA 22992 | ASHHAR ZAMAN 22881 |

1 Problem Statement and Objectives

This report outlines our approach and findings concerning link prediction in a subset of the Twitter network, comprising approximately 4.8 million nodes and 24 million edges. In this context, each node signifies a unique Twitter user identified by a randomly assigned ID, while each edge represents a "following" relationship between two users. The dataset includes a test set consisting of 2,000 edges, with an equal distribution of genuine and fabricated connections. The genuine edges represent authentic connections within the Twitter network that were excluded from the training dataset. Conversely, the fake edges are artificially generated and do not correspond to real connections. The primary project objective was to accurately classify each test edge as either genuine or fake.

2 Methods and Experiments

To accomplish our project's objectives, we initiated the process by constructing a directed graph from the provided training data network. This critical step was efficiently executed using the built-in functions of the NetworkX library in Python. Given the constraint of limited time, we employed intelligent sampling techniques to identify the most suitable positive and negative node pairs. These selected pairs were then subjected to detailed analysis, allowing us to extract pertinent features necessary for our model.
In our approach, we adopted various supervised learning models available in the scikit-learn (sklearn) library. These models included K-Nearest Neighbors, Logistic Regression, Decision Trees, and Support Vector Machines (SVM). To enhance the predictive accuracy of our base models, we employed ensemble methods, specifically XGBoost, Stacking, and Bagging.
Our ultimate goal in this project was to provide probability estimates for the formation of links in the future. This entailed leveraging the insights and patterns extracted from the training data to make informed predictions about node connections in subsequent instances.

3 Features Description

Let $G = [V, E]$ be the graph that represents the topological structure of a general social network. Links in the graph are denoted by $e = (u, v) \in E$, where $u, v \in V$.

Features	Formulas	Description
Jaccard's Coefficient	$(\mid \Gamma(u) \cap \Gamma(v) \mid)/(\mid \Gamma(u) \cup \Gamma(v) \mid)$	Common neighbors normalized by total neighbors
Preferential Attachment	$\mid \Gamma(u) \mid\mid \Gamma(v) \mid$	New connections determined by size of current neighborhoods
Page Rank	$PR(u) = \Sigma PR(v)/L(v)$	Random walk with resets in G
Common Followers	$\mid \Gamma(u) \cap \Gamma(v) \mid$	Number of common neighbors
Cosine	$\mid \Gamma(u) \cap \Gamma(v)/(\Gamma(u)\Gamma(v))$	Common followers normalized by pref. attachment
Transitive Friends	$\mid \Gamma_{out}(u)\Gamma_{in}(v) \mid$	Calculate the number of transitive friends of u, v and v, u
Bot Follower	B(u)=$\Gamma_o ut(u)/\Gamma_i n(u)$	Describes if the source node has characteristics of a bot
Celebrity feature	C(u)=$\Gamma_{in}(u)/\Gamma_{out}(u)$	Describes if the source node has characteristics of a celebrity
Shortest path length	N/A	All shortest paths between $\Gamma(u) and \Gamma(v)$

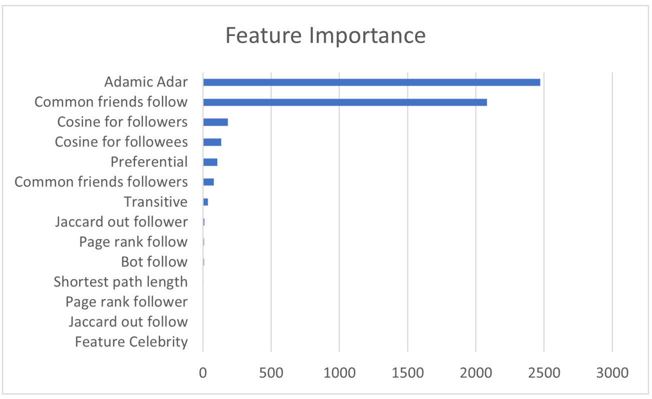
4 Final Approach

The graph of the given training data of non empty nodes are made using the library of networkx for the direct calculation and access of features function in the library that we can further extract for the calculation of other features.

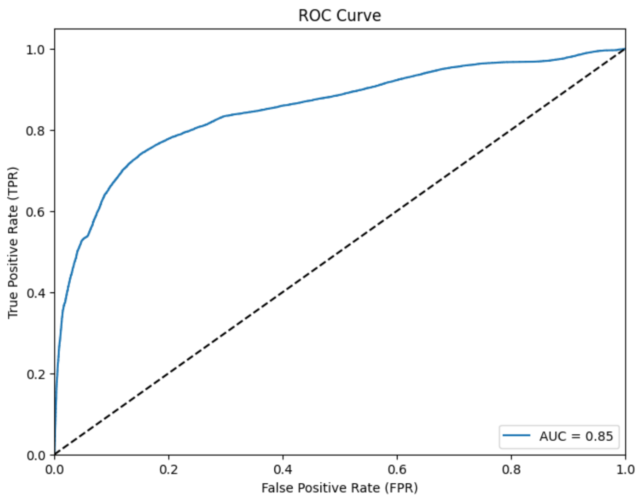
Positive links are created by randomly taking the key node and choosing any node randomly in the same row which gave us 500,000 of positive pairs. The negative sampling is done in two ways : (i) Taking the pairs from only the key values. (ii) Taking random pairs of nodes from the training data. The former performed better as the key value gave more defined features for the model to learn the negative pairs and hence we used the same.
All the above features are extracted by using inbuilt module of the networkx library such as outedges and inedges module. The features are added to the dataframe with the output as 1 and 0 for linked and non linked nodes respectively for the training of model.
The features and output (1 and 0 for linked and non linked respectively) are trained by using the Logistic regression model by splitting the data for training and cross validation.
The test data is read and the features are added for the test data pairs using the previously generated graph. Probability prediction is done using the trained model which is our final prediction.

4.1 Performance

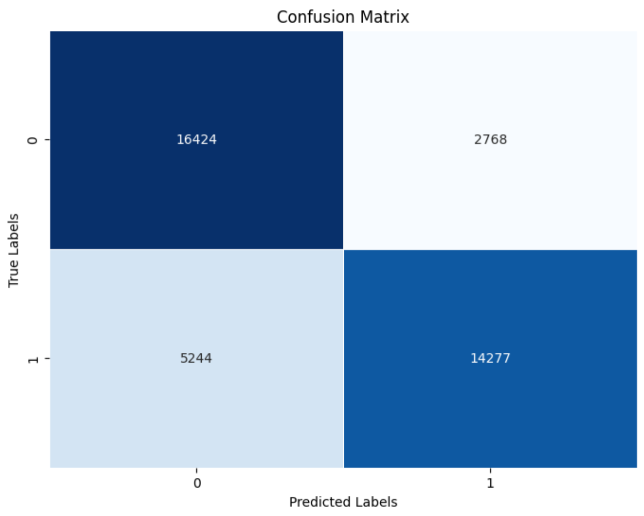
We tried solving the problem by using multiple model algorithms,however the best result was obtained when the logistic regression model algorithm was used. Logistic regression is a simple and interpretable model that can handle sparse and high-dimensional data. In this approach,graph was obtained using networkx library modules, a total sample of 200k (100k positive samples,and 100k negative samples) were taken from the graph,and all the features mentioned above were extracted.Then the model was trained on all these data sets, keeping 80% of the whole data set as training samples, and the rest as test-data. After training and testing the model predictions were made for the given test data of 2000 samples. The final AUC that we get in the kaggle competition is 87.632 %.



(a) Feature Importance



(b) ROC Curve



(c) Confusion Matrix

Figure 1: Analysis of Prediction of trained model

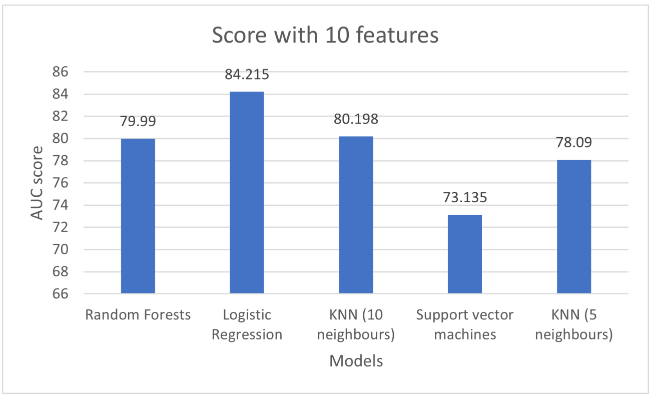
4.2 Advantages and Disadvantages of Final approach

The model used in the final approach(logistic model) is computationally more efficient than others. It helped in producing probability estimates and also confidences of each output. The accuracy of the model has no effect after increasing the magnitude of samples from 200k to 1 million edges.

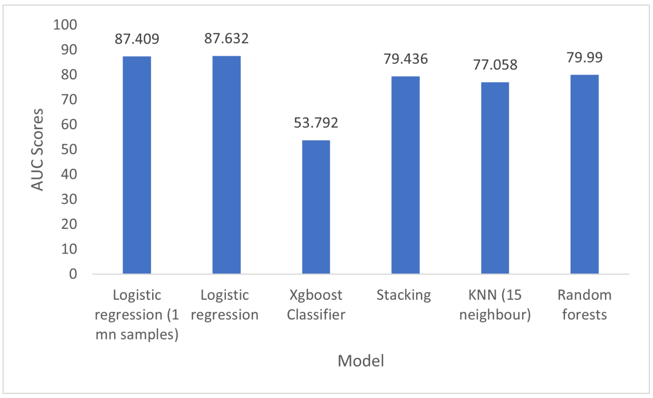
5 Results and Discussions

According to the final results derived,logistic regression model worked better than the rest models,when trained on 500k positive samples and 500k negative samples. Logistic model gave an accuracy of 80.686% on test and train data and a score of 86.184% for ROC (receiver operating characteristic).

5.1 Performance of alternative approaches



(a) AUC score with 10 features



(b) AUC score with 14 features

Figure 2: Effect of number of features for model training

5.2 Challenges

We tried different ways of scaling the features before feeding them to the classifier to train it better. Optimal sample size for training the model was a concern. Due to the large number of nodes in the training set it was difficult to learn meaningful node embeddings and network features for all nodes. Also code run time for creating the samples was high (around 5-6 hours) for large samples (for 1 million samples).

6 Conclusion

In this project, we learned how to apply machine learning techniques to a real-world problem involving social network analysis. we gained experience in working with large-scale network data, generating node embeddings, extracting network features, and building and evaluating a probability estimating classifier.

7 References

- Fire, Michael, et al. "Link prediction in social networks using computationally efficient topological features." 2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing. IEEE, 2011.

- Cukierski, William, Benjamin Hamner, and Bo Yang. "Graph-based features for supervised link prediction." The 2011 International joint conference on neural networks. IEEE, 2011.

- Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, “Exploring network structure, dynamics, and function using NetworkX”, in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

- Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.