Social Network Graph Link Prediction

MACHINE LEARNING
PROJECT REPORT

Submitted By:

Bhavna Nagar, 12
Parag Dudeja, 34

Submitted To:

Dr. Bharti Rana

Table of Contents

Acknowledgement

We would like to convey my heartfelt gratitude to Dr. Bharti Rana for her tremendous support and assistance in the completion of our project. We would also like to thank Dr. Bharti for providing us with this wonderful opportunity to work on a project with the topic Social Network Graph Link Prediction during this semester. The completion of the project would not have been possible without her help and insights.

Introduction

A r**ecommender system**, or a **recommendation system**, is a subclass of information filtering system that provides suggestions for items that are most pertinent to a particular user. Typically, the suggestions refer to various decision-making processes, such as what product to purchase, what music to listen to, what people to connect, or what online news to read. Recommender systems are particularly useful when an individual needs to choose an item from a potentially overwhelming number of items that a service may offer.

Recommendation systems are used in innumerable types of areas such as generation of playlists, music and video services like Jio Savaan, Wynk, Amazon Prime Music etc. and products recommendation for users in e-commerce applications and commercial applications.

Two main branches of recommender algorithms are often distinguished: **content-based recommender systems** [24] and **collaborative filtering models** [9].

Content-based recommender systems use content information of users and items, such as their respective occupation and genre, to predict the next purchase of a user or rating of an item.

Collaborative filtering models solve the matrix completion task by taking into account the collective interaction data to predict future ratings or purchases.

Social recommendation system is a system that recommends friends in social media applications such as Facebook, Twitter, Instagram etc.

In recent times social media is enjoying a great deal of success with a million of users visiting many sites like Facebook, Twitter etc. for social networking. The information that is very much interested by the users is suggested by the recommendation system by using information filtering techniques.

The Social Recommendation system recommends friends to the users based on the ratings given by the users with similar interests and topics. It recommends the friends to the users by knowing the followers and followee of a particular user.

Problem Statement

Social media platforms like facebook and instagram have millions of users. Finding missing links among the huge number of people is a difficult task. Objective of this is to calculate link features like number of followers, number of following, weight of edges and shortest path between source and destination node etc and predict link between two people in a social network.

We used facebook’s recruitment dataset present on kaggle.the dataset has edge list only i.e we were given source and destination node, each user was represented by some number. No data regarding the users was given. Given the two users we have to predict if there will be a link from one user(source) to another user(destination).

Dataset

Data contains two columns source and destination, each edge in graph

Data columns (total 2 columns):
- source_node int64
- destination_node int64

We have a DiGraph with 1862220 nodes and 9437519 edges

Related Work

The following research papers were studied extensively to understand the work done currently in the field of recommendation systems

1. Friend Recommendation using graph mining on social media, Kosaraju Naren Kumar (Student & Sathyabama University), Kanakamedala Vineela (Assistant Professor & NRI Institute of Technology)
2. Graph Convolutional Matrix Completion, Rianne van den Berg (University of Amsterdam), Thomas N. Kipf (University of Amsterdam), Max Welling(University of Amsterdam, CIFAR[1])

[1] Canadian Institute for Advanced Research

Friend Recommendation using graph mining on social media

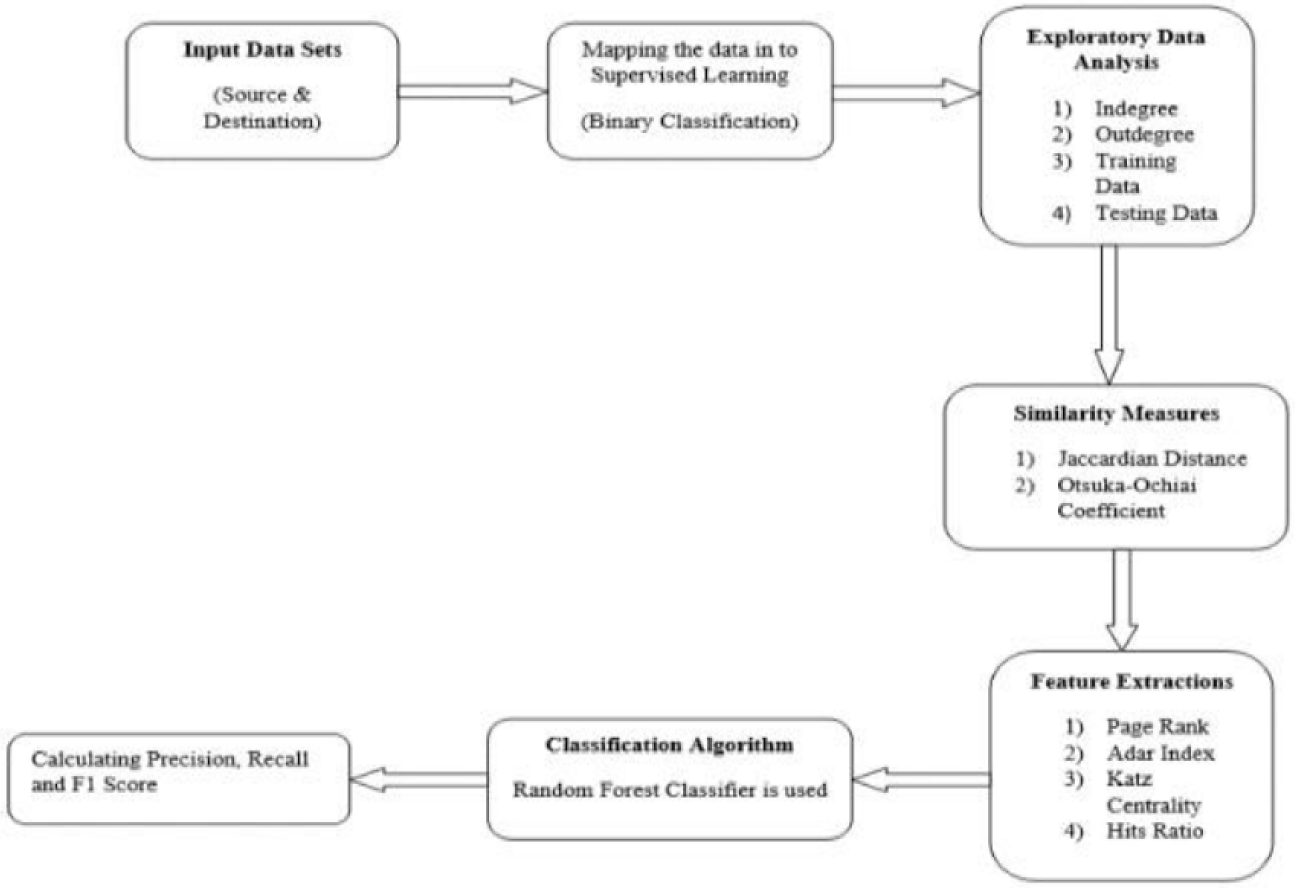
Abstract

“Recommendation systems are used for custom-made navigation by getting huge amounts of data particularly in the social media domain for recommending friends. A recommendation system act as a subclass for the information filtering system that pursue to predict the rating. The similarity measures that are calculated in this research are Jaccard distance and Otsuka-Ochiai coefficient. The feature extractions that are used in this paper are Adar index, PageRank, Katz centrality, Hits score.
The classification algorithm used is Random Forest Classifier.”

Highlights of the Paper

- Using a dataset of 9437519 nodes of both source and destination, in that entire volume of data 80% is used for training and 20% is used for testing for future predictions.
- In this model the classification algorithm used is Random Forest Classifier. Random forest algorithm is a tree based algorithm, so it will work well for dimensional data and nonlinear separable data. Accuracy score is calculated by the precision, recall and F1 score. For describing the performance of the test data in a classification model confusion matrix is used.
- The similarity measures that are calculated in this research are Jaccard distance and Otsuka-Ochiai coefficient. The feature extractions that are used in this paper are Adar index, PageRank, Katz centrality, Hits score.

Proposed Solution



Dataset Overview

- Dataset is taken from the Facebook's recruiting challenge on Kaggle and the dataset comprehends two columns such as Source and Destination
- The volume of the dataset is approximately 94 Lakhs, later on the data is fragmented into two types such as training data and testing data. 80% and 20% of the data is used for training and testing the data respectively
- Total no of nodes presents in the data: 1862220, Total no of edges presents in the data: 9437519

Mapping the problem into Supervised Learning problem

Generated training samples of good and bad links from given directed graph and for each link got some features like number of followers, is he followed back, page rank, katz score, adar index, some SVD features of adjacent matrix, some weight features etc. and trained machine learning model based on these features to predict link.

In-Degree – The number of edges directed into a vertex in a directed graph is called in-degree. In other words, it can be defined as the number of incoming nodes for a particular node. InDegree is used to find the number of followers for each user.

Out-Degree – The number of edges directed away from the vertex in a directed graph is called out-degree.In other words, it can be defined as the number of outgoing nodes for a particular node. Out-Degree is used to find the number of people each user is following.

- **Jaccard Distance** –Jaccard distance is nothing but a measure of similarity between two data nodes ranging from 0% to 100%. As the Jaccard distance increases then there is a high chance of an existing edge between the two nodes. It is defined as the size of the intersection of two sets to the size of the union of the two sets. It’s very sensitive towards small amounts of data and gives flawed results.

$$j = \frac{|X \cap Y|}{|X \cup Y|}$$

- **Cosine-Distance**(OTSUKA–OCHIAI COEFFICIENT)-Otsuka-Ochiai coefficient is nothing but an intersection of the no of elements to the square root of the no of elements in A multiplied by the number of elements in B.

$$K = \frac{|A \cap B|}{\sqrt{|A| \times |B|}}$$

- **Page Rank** – PageRank is an algorithm that was designed to rank the importance of web pages. Given a directed graph the PageRank algorithm will give each vertex (Ui) a score. The score represents the importance of the vertex in the directed graph. The Networkx library is used to compute the PageRank.

Checking For Same Weakly Connected Components

- If two users belong to the same weakly connected components that gives the higher probability or higher chance of a similar edge being present. Weakly connected component acts as a subgraph for the given directed graph. Weakly connected component acts as a strongly connected component in the case of an undirected graph.

- **Adar Index**– Adar Index Adar index or Adamic index is nothing but an inverted sum of degrees of common neighbors for two vertices. The Networkx library is used to compute the Adar index.

$$A(x,y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{log(|N(u)|)}$$

- **Katz Centrality** – Based on the centrality of its neighbors Katz Centrality computes the centrality of a node. It is an inductive reasoning of the eigenvector centrality. The Katz centrality for node i is Formula J) – Katz Centrality Where A is the adjacency matrix of the graph G with eigenvalues λ. Networkx library is used to compute the Katz centrality

$$x_i = \alpha \sum_j A_{ij} x_j + \beta$$

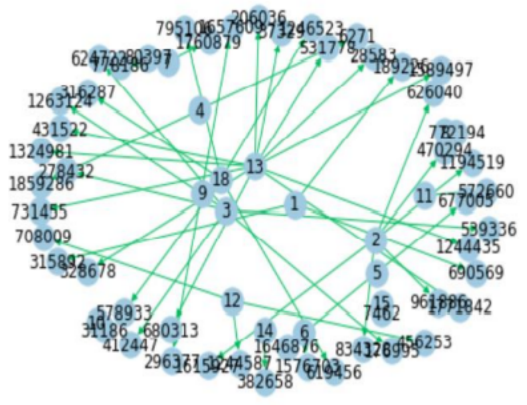
- **Hits Score** – The HITS(Hyper Induced Topic Search) algorithm computes two no for a node. Based on the incoming links authorities estimate the node value. Based on outgoing links, hubs estimates the node value. The Networkx library is used to compute the HITS score.

- **Weight Features** – An edge weight value was calculated between nodes in order to find the similarity of nodes. As the neighbor count goes up, edge weight decreases. Intuitively, consider one million people following a celebrity on a social network then chances are most of them never met each other or the celebrity. Whereas on the other hand, if a user has 30 contacts in his/her social network, the chances are higher that most of them know each other. As it is a directed graph, weighted in and weighted out are calculated separately.

$$W = \frac{1}{\sqrt{1 + |X|}}$$

- **Singular Value Decomposition** – For factorization of matrix into singular values and singular vectors SVD (Singular Value Decomposition) algorithm is used. SVD features for both source and destination nodes SVD is widely used in machine learning reduction techniques and matrix calculations

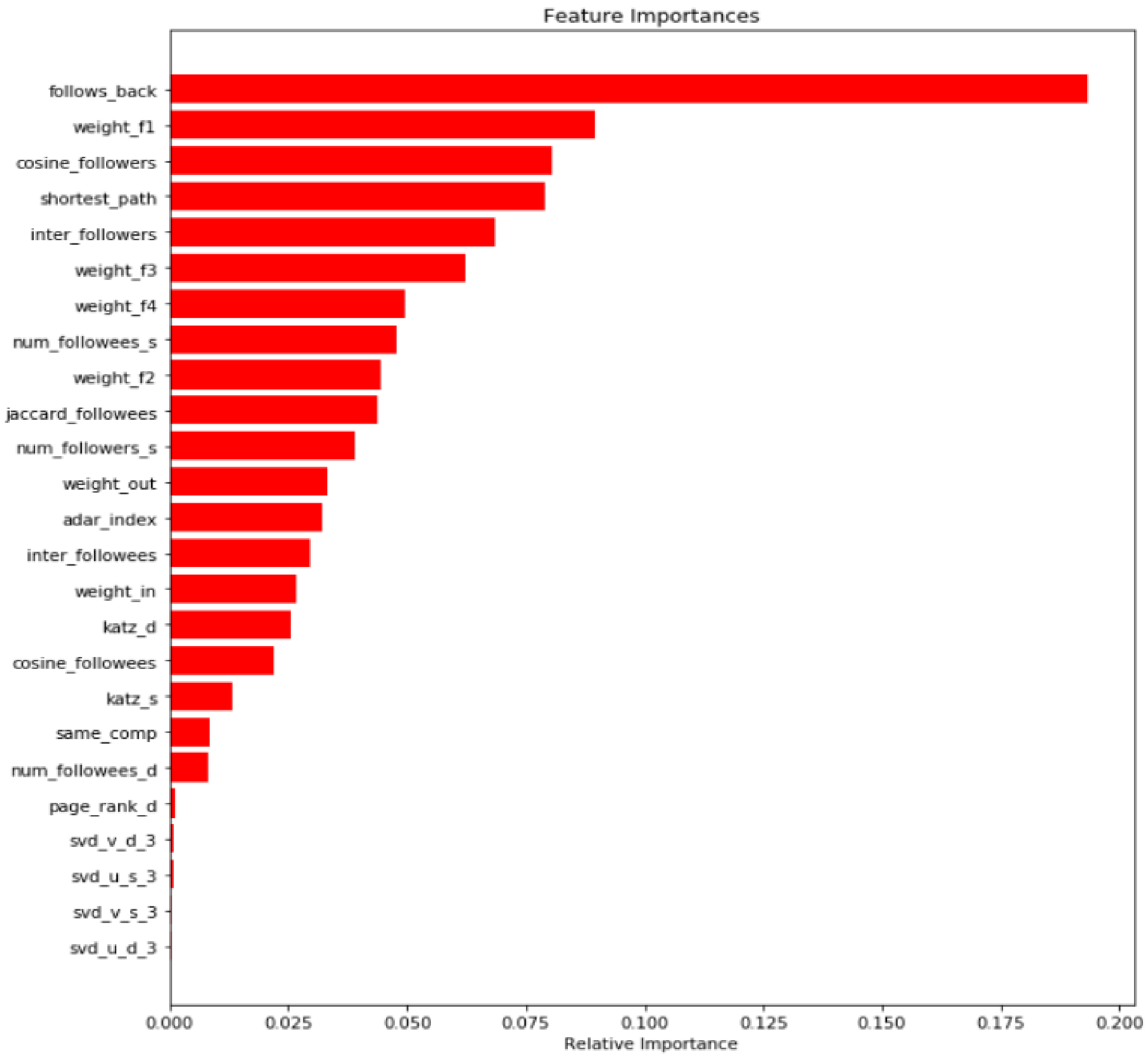
Experiment Results and Discussions



Subgraph-the directed graph obtained from the dataset. By this digraph number of edges and nodes are calculated.
The no of nodes in the subgraph are: 66
The no of edges in the subgraph are: 50
The average In-Degree of the above subgraph: 0.7576
The average Out-Degree of the above subgraph: 0.7576

Feature Extraction Graph

The below graph shows the differences and importance of the feature extractions that are calculated and obtained in a form of the bar graph. From the graph we can say that follows_back feature is the most important extraction compared to other features and SVD(Singular Value Decomposition) is the least preferred one.



Conclusion

The proposed system develops a friend recommendation system to suggest friends to the users using Random Forest Classifier. With the help of the proposed system this recommendation model is working with the accuracy of 89%. This is quite reasonable for the hardware and the volume of data that we have. Our data set consists of 94 lakhs nodes. Performance metrics for this model is obtained by calculating Precision, Recall and F1 score.The most important feature extraction that we calculated is follows_back. feature. For the better results and accuracy Preferential attachment, SVM classifier and Graph neural networks can be used. This can improve the performance of the model in future

Graph Convolutional Matrix Completion

Abstract

“We consider matrix completion for recommender systems from the point of view of link prediction on graphs. Interaction data such as movie ratings can be represented by a bipartite user-item graph with labeled edges denoting observed ratings. Building on recent progress in deep learning on graph-structured data, we propose a graph auto-encoder framework based on differentiable message passing on the bipartite interaction graph. Our model shows competitive performance on standard collaborative filtering benchmarks. In settings where complimentary feature information or structured data such as a social network is available, our framework outperforms recent state-of-the-art methods.”

Highlights of the paper

- Matrix completion as a link prediction problem on graphs
- The interaction data in collaborative filtering can be represented by a bipartite graph between user and item nodes, with observed ratings/purchases represented by links.
- Content information can naturally be included in this framework in the form of node features.
- Predicting ratings then reduces to predicting labeled links in the bipartite user-item graph.

Proposed Solution

- Graph convolutional matrix completion (GC-MC): a graph-based auto-encoder framework for matrix completion, which builds on recent progress in deep learning on graphs
- The benefit of formulating matrix completion as a link prediction task on a bipartite graph becomes especially apparent when recommender graphs are accompanied with structured external information such as social networks.

Matrix completion as link prediction in bipartite graphs

- Rating matrix M of shape $N_u \times N_v$, where N_u is the number of users and N_v is the number of items. Entries M_{ij} in this matrix encode either an observed rating (user i rated item j) from a set of discrete possible rating values, or the fact that the rating is unobserved (encoded by the value 0)

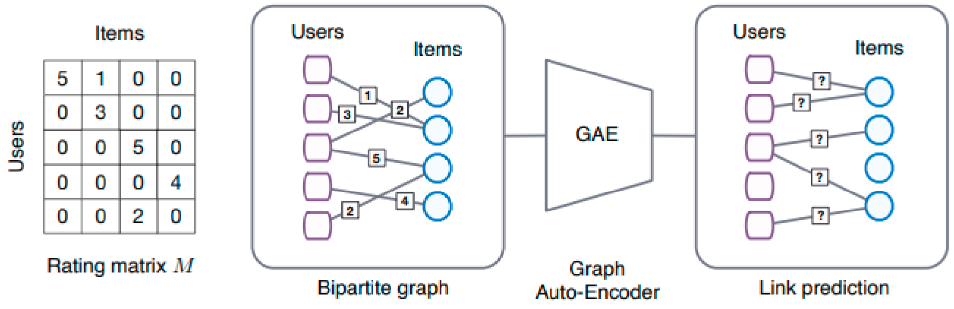


Figure 1: *Left*: Rating matrix M with entries that correspond to user-item interactions (ratings between 1-5) or missing observations (0). *Right*: User-item interaction graph with bipartite structure. Edges correspond to interaction events, numbers on edges denote the rating a user has given to a particular item. The matrix completion task (i.e. predictions for unobserved interactions) can be cast as a link prediction problem and modeled using an end-to-end trainable graph auto-encoder.

- The task of matrix completion or recommendation can be seen as predicting the value of unobserved entries in M.
- Previous graph-based approaches for recommender systems typically employ a multistage pipeline, consisting of a graph

feature extraction model and a link prediction model, all of which are trained separately.

- Recent results, however, have shown that results can often be significantly improved by modeling graph-structured data with end-to-end learning techniques and specifically with graph auto-encoders for unsupervised learning and link prediction.

Graph Auto-encoder

- Graph autoencoders are comprised of
- A graph encoder model Z = f(X, A), which take as input an $N \times D$ feature matrix X and a graph adjacency matrix A, and produce an $N \times E$ node embedding matrix $Z = [z^T_1, \dots, z^T_N]^T$
- A pairwise decoder model $\check{A} = g(Z)$, which takes pairs of node embeddings (z_i, z_j) and predicts respective entries $\check{A}_{ij}$ in the adjacency matrix.

Graph convolutional encoder

We can assign a specific transformation for each rating level, resulting in edge type specific messages $\mu_j \rightarrow i,r$ from items j to users i of the following form

$$\mu_{j\rightarrow i,r} = \frac{1}{c_{ij}} W_r x_j \, . \qquad (1)$$

and by subsequently accumulating them into a single vector representation

$$h_i = \sigma\left[\operatorname{accum}\left(\sum_{j\in\mathcal{N}_{i,1}} \mu_{j\rightarrow i,1}, \dots, \sum_{j\in\mathcal{N}_{i,R}} \mu_{j\rightarrow i,R}\right)\right], \qquad (2)$$

To arrive at the final embedding of user node i, we transform the intermediate output h_i as follows

$$u_i = \sigma(W h_i)\,. \tag{3}$$

We will refer to (2) as a graph convolution layer and to (3) as a dense layer.

Bilinear Decoder

For reconstructing links in the bipartite interaction graph we consider a bilinear decoder, and treat each rating level as a separate class. Indicating the reconstructed rating between user i and item j with Mˇij , the decoder produces a probability distribution over possible rating levels through a bilinear operation followed by the application of a softmax function:

$$p(\check{M}_{ij} = r) = \frac{e^{u_i^T Q_r v_j}}{\sum_{s \in R} e^{u_i^T Q_s v_j}}\,, \tag{4}$$

with Qr a trainable parameter matrix of shape E × E, and E the dimensionality of hidden user (item) representations ui (vj). The predicted rating is computed as

$$\check{M}_{ij} = g(u_i, v_j) = \mathbb{E}_{p(\check{M}_{ij}=r)}[r] = \sum_{r \in R} r\, p(\check{M}_{ij} = r)\,. \tag{5}$$

Putting it all together

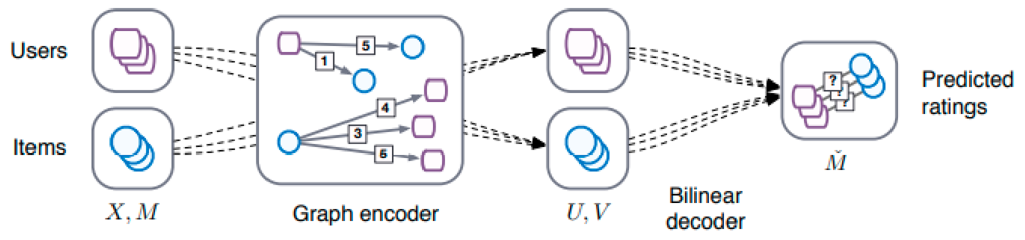


Figure 2: Schematic of a forward-pass through the GC-MC model, which is comprised of a graph convolutional encoder $[U, V] = f(X, M_1, \ldots, M_R)$ that passes and transforms messages from user to item nodes, and vice versa, followed by a bilinear decoder model that predicts entries of the (reconstructed) rating matrix $\check{M} = g(U, V)$, based on pairs of user and item embeddings.

Model Training

- **Loss function:** During model training, we minimize the following negative log likelihood of the predicted ratings $\check{M}_{ij}$:

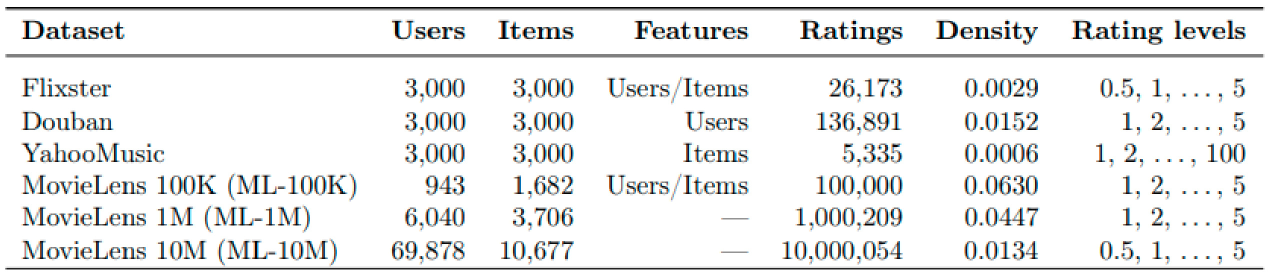
$$\mathcal{L} = -\sum_{i,j;\Omega_{ij}=1} \sum_{r=1}^{R} I[r = M_{ij}] \log p(\check{M}_{ij} = r) \,, \qquad (6)$$

  with I[k = l] = 1 when k = l and zero otherwise. The matrix $\Omega \in \{0, 1\}^{N_u \times N_i}$ serves as a mask for unobserved ratings, such that ones occur for elements corresponding to observed ratings in M, and zeros for unobserved ratings.

- **Node dropout** and **mini-batching** is used in training the model

Experiments

Model was evaluated on a number of common collaborative filtering benchmark datasets: MovieLens2 (100K, 1M, and 10M), Flixster, Douban, and YahooMusic. The datasets consist of user ratings for items (such as movies) and optionally incorporate additional user/item information in the form of features. For Flixster, Douban, and YahooMusic we use preprocessed subsets of these datasets provided by [22] 3 . These datasets contain sub-graphs of 3000 users and 3000 items and their respective user-user and item-item interaction graphs (if available).



Dataset	Users	Items	Features	Ratings	Density	Rating levels
Flixster	3,000	3,000	Users/Items	26,173	0.0029	0.5, 1, . . . , 5
Douban	3,000	3,000	Users	136,891	0.0152	1, 2, . . . , 5
YahooMusic	3,000	3,000	Items	5,335	0.0006	1, 2, . . . , 100
MovieLens 100K (ML-100K)	943	1,682	Users/Items	100,000	0.0630	1, 2, . . . , 5
MovieLens 1M (ML-1M)	6,040	3,706	—	1,000,209	0.0447	1, 2, . . . , 5
MovieLens 10M (ML-10M)	69,878	10,677	—	10,000,054	0.0134	0.5, 1, . . . , 5

Table 1: Number of users, items and ratings for each of the MovieLens datasets used in our experiments. We further indicate rating density and rating levels.

Summary of Results

Model	ML-100K + Feat
MC [3]	0.973
IMC [11, 31]	1.653
GMC [12]	0.996
GRALS [25]	0.945
sRGCNN [22]	0.929
GC-MC (Ours)	0.910
GC-MC+Feat	**0.905**

Table 2: RMSE scores[6] for the MovieLens 100K task with side information on a canonical 80/20 training/test set split. Side information is either presented as a nearest-neighbor graph in user/item feature space or as raw feature vectors. Baseline numbers are taken from [22].

Model	Flixster	Douban	YahooMusic
GRALS	1.313/1.245	0.833	38.0
sRGCNN	1.179/0.926	0.801	22.4
GC-MC	**0.941/0.917**	**0.734**	**20.5**

Table 3: Average RMSE test set scores for 5 runs on Flixster, Douban, and YahooMusic, all of which include side information in the form of user and/or item graphs. We replicate the benchmark setting as in [22]. For Flixster, we show results for both user/item graphs (right number) and user graph only (left number). Baseline numbers are taken from [22].

Model	ML-1M	ML-10M
PMF [20]	0.883	–
I-RBM [26]	0.854	0.825
BiasMF [16]	0.845	0.803
NNMF [7]	0.843	–
LLORMA-Local [17]	0.833	0.782
I-AUTOREC [27]	0.831	0.782
CF-NADE [32]	**0.829**	**0.771**
GC-MC (Ours)	0.832	0.777

Table 4: Comparison of average test RMSE scores on five 90/10 training/test set splits (as in [32]) without the use of side information. Baseline scores are taken from [32]. For CF-NADE, we report the best-performing model variant.

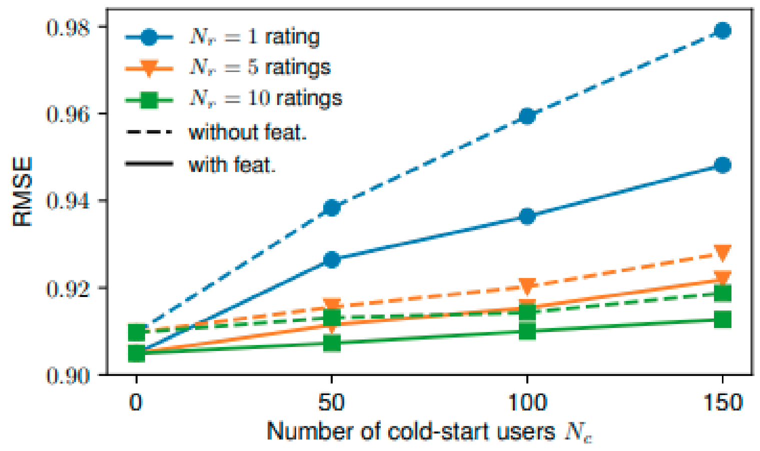


Figure 3: Cold-start analysis for ML-100K. Test set RMSE (average over 5 runs with random initialization) for various settings, where only a small number of ratings N_r is kept for a certain number of cold-start users N_c during training. Standard error is below 0.001 and therefore not shown. Dashed and solid lines denote experiments without and with side information, respectively.

Conclusions

- In this work, the authors have introduced graph convolutional matrix completion (GC-MC): a graph auto-encoder framework for the matrix completion task in recommender systems.
- The encoder contains a graph convolution layer that constructs user and item embeddings through message passing on the bipartite user-item interaction graph.
- Combined with a bilinear decoder, new ratings are predicted in the form of labeled edges.
- Proposed model outperforms recent related methods by a large margin, as demonstrated on a number of benchmark datasets with feature- and graph-based side information.
- In future work, the authors wish to extend this model to large-scale multi-modal data (comprised of text, images, and other graph-based information), such as present in many realistic recommendation platforms.
- To address scalability, it is necessary to develop efficient approximate schemes, such as subsampling local neighborhoods

Methodology

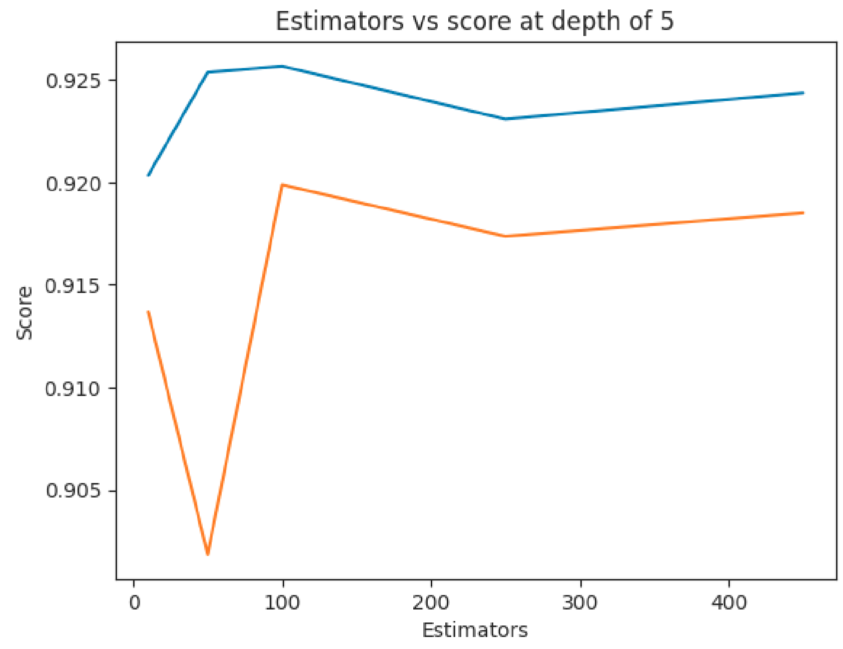
- Posing problem as classification problem
- Generated training samples of good and bad links from given directed graph
- From a given directed graph and for each link got some features like no of followers, is he followed back, page rank, katz score, adar index, some SVD features of adj matrix, some weight features etc. and trained machine learning model based on these features to predict link.

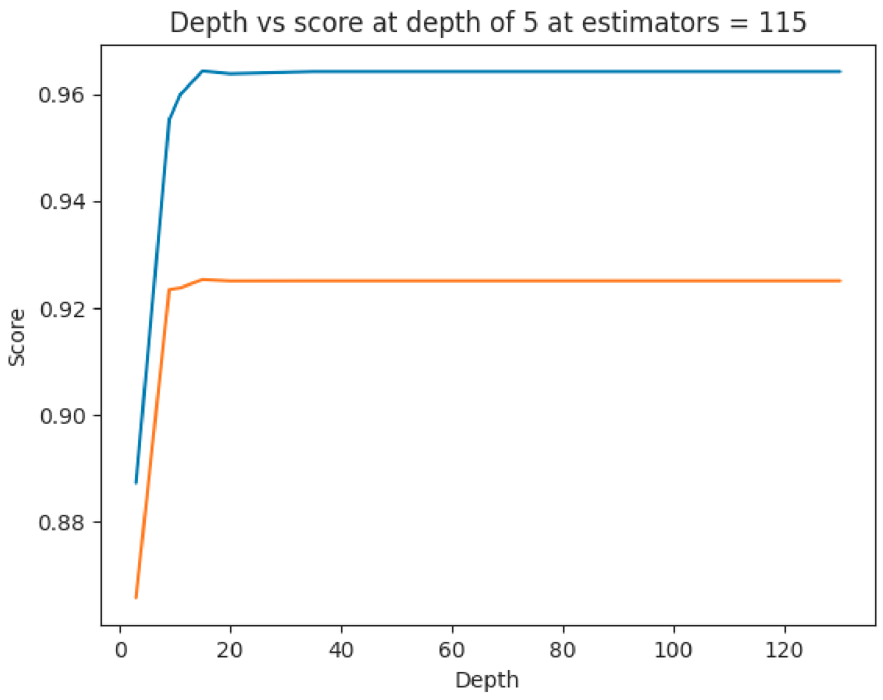
Novelty

- Only nodes with shortest distance greater than 2 considered while generating training sample
- Ensemble methods such as XGBoost and CatBoost used, which give significant improvements in the results

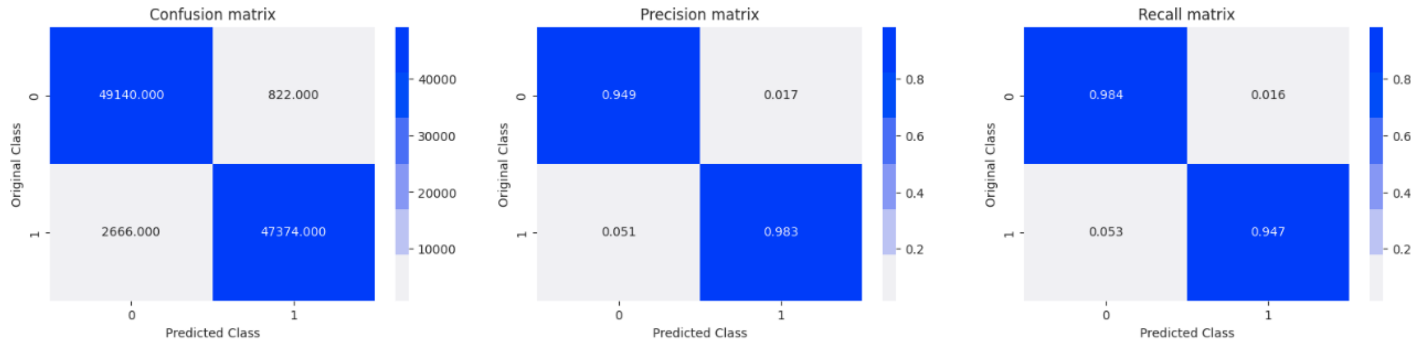
Experimental Results

Random Forest Classifier

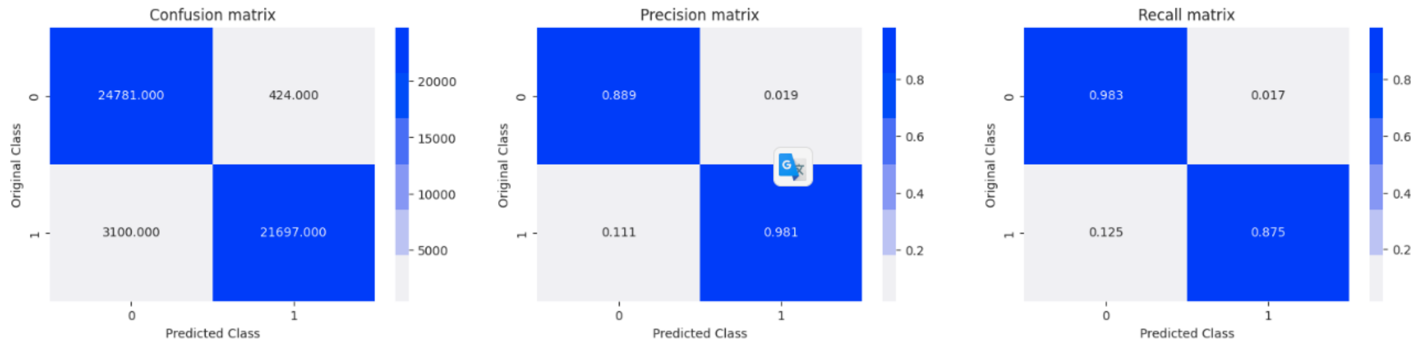




```
Train confusion_matrix
```



```
Test confusion_matrix
```

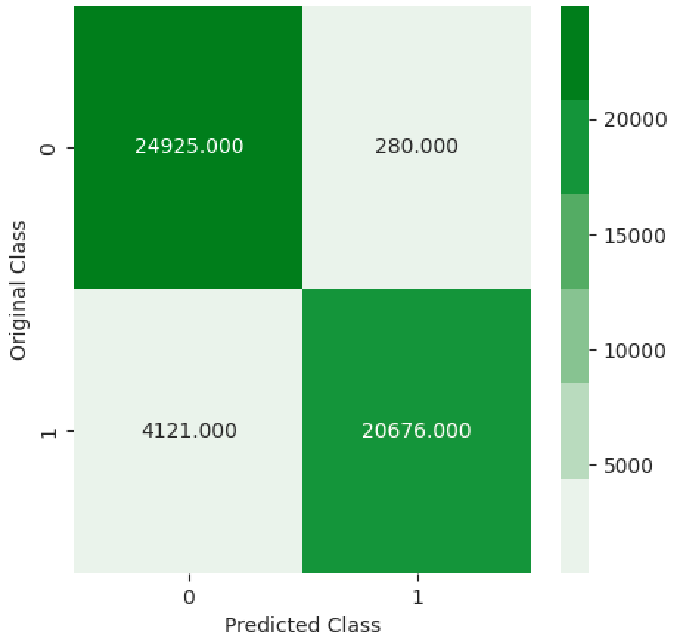


Train f1 score: 0.964
Test f1 score: 0.929

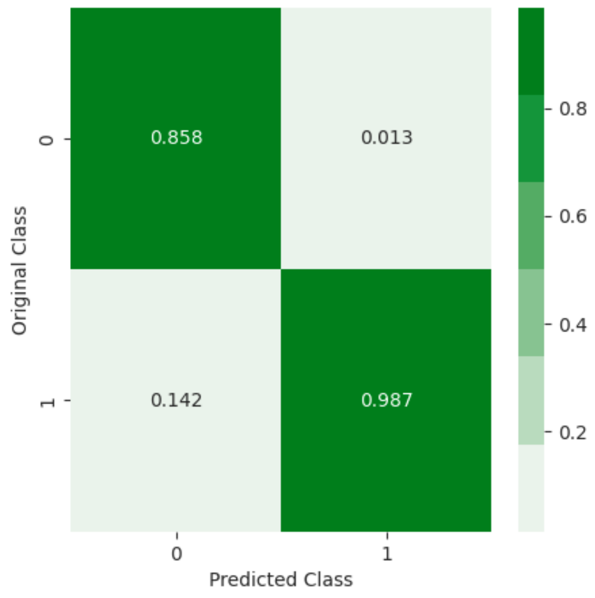
XGBoost

Train log loss: 0.021593514320677812
Test log loss: is 0.35836026786772435
Error rate: 8.801647934082636

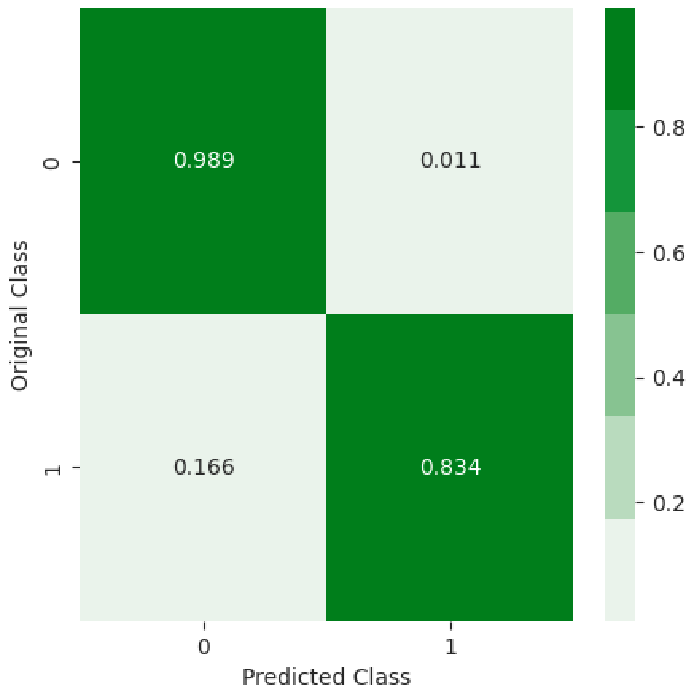
Confusion Matrix



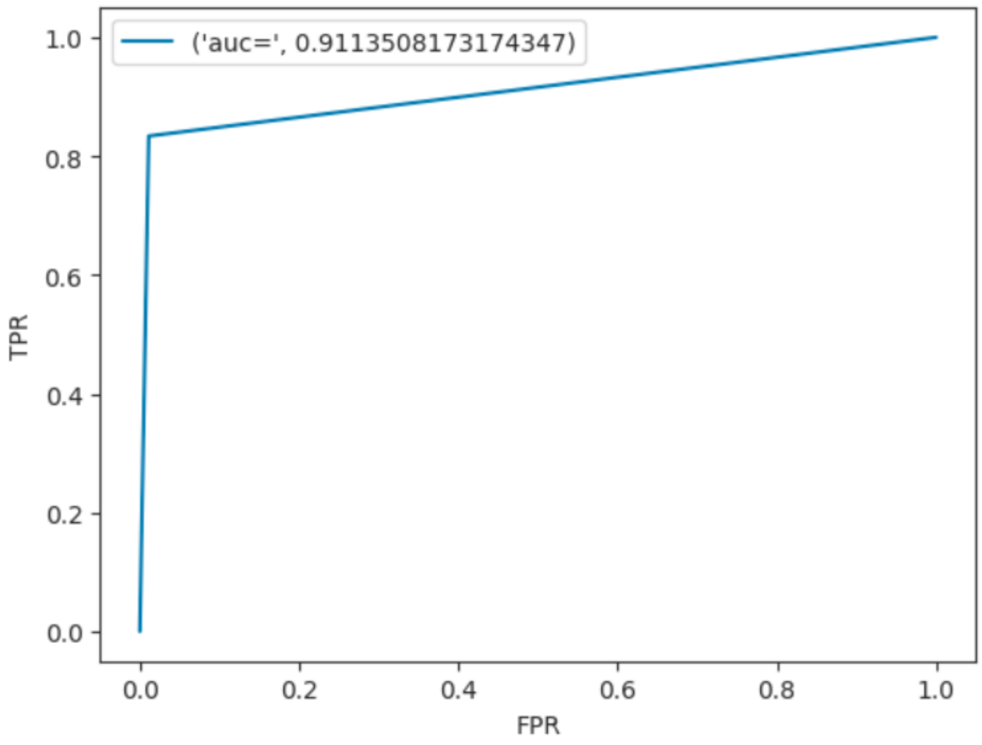
Precision Matrix

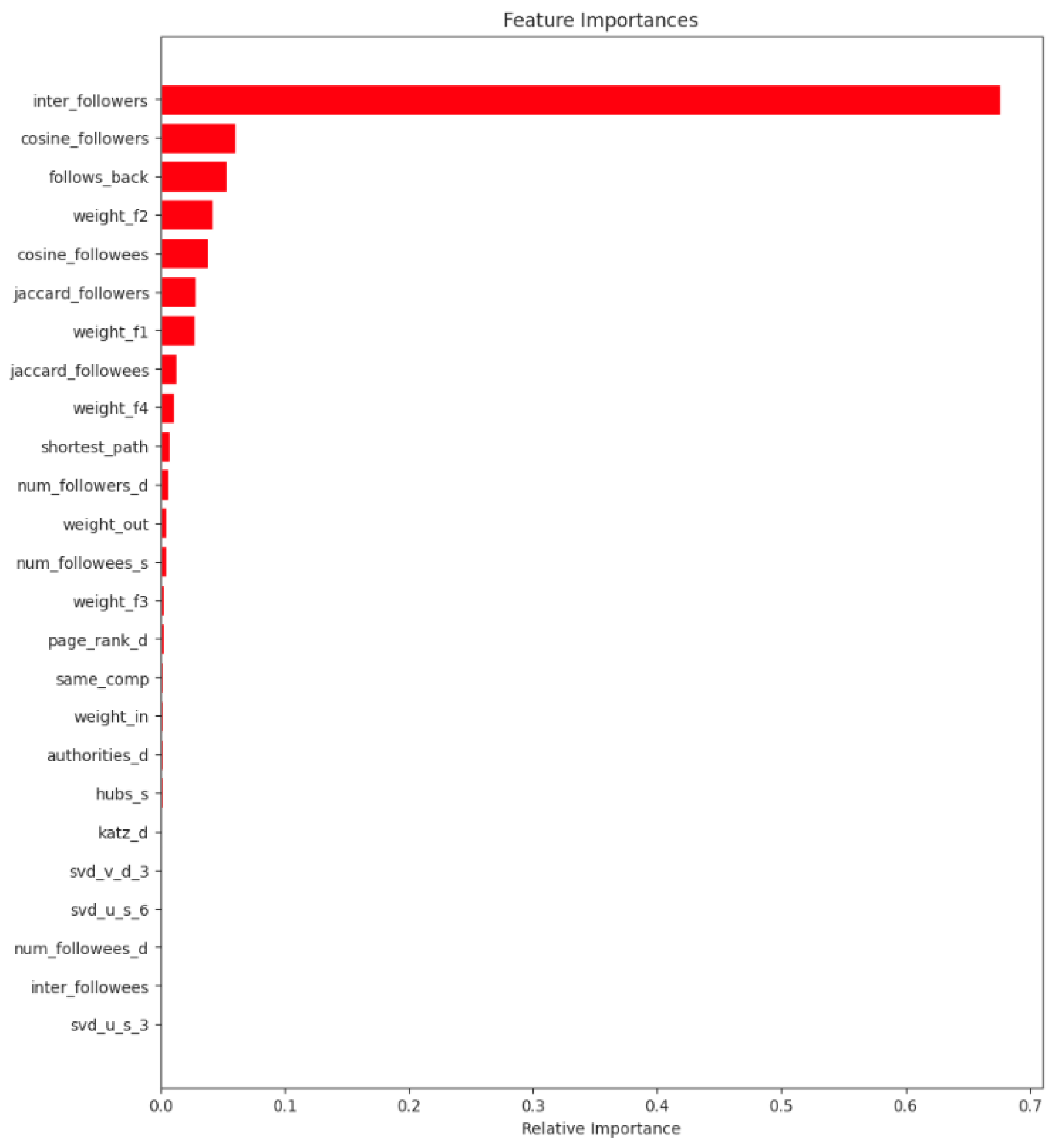


Recall Matrix



AUC-ROC Graph





Train f1 score: 0.993
Test f1 score: 0.904

<u>CatBoost</u>
Best test error = 0.2399055714
For depth=10, learning_rate=0.1

Conclusion and Future Work

We can see the results after taking nodes with shortest distance greater than 2, combined with the usage of ensemble methods such as XGBoost and CatBoost got better results, around 91.9% accuracy over the 89% accuracy of the paper studied. This is quite reasonable for the hardware and the volume of data that we have.

For better results, SVM classifier and Graph neural networks can be used. This can improve the performance of the model in future

Bibliography

1. IJETMS-SE-016.pdf
2. https://arxiv.org/pdf/1706.02263v2.pdf
3. https://www.cs.cornell.edu/home/kleinber/link-pred.pdf
4. https://www3.nd.edu/~dial/publications/lichtenwalter2010new.pdf
5. https://www.youtube.com/watch?v=2M77Hgy17cg
6. https://en.wikipedia.org/wiki/PageRank