

ANALYSIS OF LINK PREDICTION IN TWITCH USING ML AND GRAPH THEORY ALGORITHMS

A PROJECT REPORT

Submitted by

MANOJ R (195002071)

SRUTHI G (195002117)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



Department of

Information Technology

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Rajiv Gandhi Salai (OMR), Kalavakkam – 603 110

MAY 2023

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this Report titled “**ANALYSIS OF LINK PREDICTION IN TWITCH USING ML AND GRAPH THEORY ALGORITHMS**” is the bonafide work of **MANOJ R (195002071)** and **SRUTHI G (195002117)** who carried out the work under my supervision.

Certified further that to the best of my knowledge the work reported herein does not form of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.Chandrabose Aravindan

Head of the Department

Department of Information Technology

SSN College of Engineering

Kalavakkam - 603 110

SIGNATURE

Dr.S.Karthika

SUPERVISOR

Associate Professor

Department of Information Technology

SSN College of Engineering

Kalavakkam - 603 110

Submitted for the project viva-voice Examination held on

EXTERNAL EXAMINER

INTERNAL EXAMINER

ABSTRACT

Link prediction in social networks is an interesting topic that has been studied extensively in recent times. The objective of link prediction is to estimate the probability of existence (or formation) of each of the non-existing links in the network in order to identify a set of missing or future links between the users. Its study is crucial to comprehending network evolution and its effects on individual nodes.

We attempted to analyse the social network by solving the link prediction problem for a dataset gathered from Twitch - an online video streaming platform. We are interested in solving the problem of personalized recommendations of streamers that a user can follow. This recommendation problem can be mapped to a binary classification problem wherein the two classes are - recommend or do not recommend with features extracted from the dataset.

We trained five classical machine learning models on the enhanced features extracted based on the study on graph theory to predict whether a link exists between any given pair of nodes and use this prediction for recommendations.

Upon extensive analysis and research, we observed very good accuracy for all the models, with Random Forest, XGBoost and

AdaBoost performing the best. Here are some reasons why the boosting algorithms and random forest might outperform other machine learning models :

1. Ensemble learning: Both boosting algorithms and random forests are examples of ensemble learning methods. They combine multiple weak learners (e.g., decision trees) to create a strong predictor. This can help to improve the accuracy and robustness of the model.
2. Non-linear relationships: Link prediction often involves non-linear relationships between the features and the target variable. Boosting algorithms and random forests are capable of modelling complex non-linear relationships, whereas some other models may struggle with this.
3. Feature importance: Both boosting algorithms and random forests are able to estimate the importance of each feature in the model. This can help to identify which features are most predictive of the target variable and can guide feature selection and engineering.

Overall, the combination of ensemble learning, non-linear modelling, feature importance estimation, and robustness to overfitting makes boosting algorithms and random forests strong candidates for link prediction tasks. However, it's worth noting that the performance of any machine learning model will depend on the specific dataset and task at hand, so it's always important to experiment with multiple models and compare their performance.

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. V.E Annamalai, Principal, SSN College of Engineering, Dr. Chandrabose Aravindan, Professor and Head of Department, Information Technology, for giving us the opportunity to carry out the project on the topic, “ANALYSIS OF LINK PREDICTION IN SOCIAL NETWORK” as our final year project. It was an enriching learning experience and helped us immensely in doing a lot of research.

It is a matter of pleasure to acknowledge the continuous support of our project guide, Dr.S.Karthika, during the project. Her innovative ideas, valuable inputs and keen suggestions kept us motivated even during the challenging parts of the project.

We sincerely thank our project panel members in the Department of Information Technology for providing us with helpful pointers throughout the course of the project.

We take this opportunity to thank all our staff members and friends for their help and assistance during this project work.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF TABLES	viii
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	xii
1	INTRODUCTION	1
	1.1 GENERAL	1
	1.2 NEED FOR THE STUDY	2
	1.3 MOTIVATION OF THE STUDY	2
	1.4 OBJECTIVES OF THE STUDY	3
2	REVIEW OF LITERATURE	4
	2.1 INTRODUCTION	4
	2.2 SUMMARY	6
3	SYSTEM DESIGN	7
	3.1 DATA COLLECTION AND PREPARATION	8
	3.2 FEATURE EXTRACTION	8
	3.3 MODEL BUILDING	8
4	TOOLS AND TECHNOLOGIES USED	11

	4.1	Google colab	11
	4.2	numpy	11
	4.3	pandas	11
	4.4	matplotlib	12
	4.5	networkx	12
	4.6	pickle	12
	4.7	sklearn	12
	4.8	xgboost	13
	4.9	seaborn	13
5		DATA COLLECTION AND PREPARATION	14
	5.1	DATA COLLECTION	14
	5.2	UNDERSTANDING THE DATA	14
	5.3	ASSUMPTIONS	17
6		FEATURE EXTRACTION	18
	6.1	PAGE RANK	18
	6.2	SHORTEST PATH	19
	6.3	FOLLOWS BACK	20
	6.4	FOLLOWER AND FOLLOWEE COUNT	20
	6.5	COMMON NEIGHBOUR	20
	6.6	PREFERENTIAL ATTACHMENT	21

7	MODEL BUILDING	22
7.1	ANALYSIS AND STANDARDISATION OF DATA	22
7.2	SPLITTING OF DATA	27
7.3	BUILDING ML MODELS	28
7.3.1	LOGISTIC REGRESSION	28
7.3.2	RANDOM FOREST	30
7.3.3	SUPPORT VECTOR MACHINE	33
7.3.4	ADABOOST	34
7.3.5	XGBOOST	36
7.4	PREDICTION	38
8	PERFORMANCE ANALYSIS	39
8.1	PERFORMANCE METRICS	39
8.1.1	ROC-AUC CURVE AND SCORE	39
8.1.2	CLASSIFICATION REPORT	42
8.1.2.1	PRECISION	42
8.1.2.2	RECALL	42
8.1.2.3	F1 SCORE	42
8.1.2.4	ACCURACY	43
8.1.3	CONFUSION MATRIX	43
8.2	RESULTS AND EVALUATION	46

9	CONCLUSION AND FUTURE WORKS	48
	REFERENCES	50

LIST OF TABLES

Table No.	Table Name	Page No.
8.1	AUC SCORE for ML Models	41

LIST OF FIGURES

Figure No.	Figure Name	Page No.
1.1	Workflow of link prediction problem	1
3.1	System Architecture	7
3.2	Comparing Logistic regression with Linear regression	9
5.1	Outline of the dataset	14
5.2	Visualization of a random sample of 500 edges from the dataset	15
5.3	Information about the graph before data preparation	15
5.4	In degree of the node	15
5.5	Out degree of the node	15
5.6	Present edges	16
5.7	Missing edges	16
5.8	Information about the graph after data preparation	17

6.1	Features extracted	21
7.1	Heatmap	22
7.2	Box plot and Histogram (after standardisation) of Page rank of source node	23
7.3	Box plot and Histogram (after standardisation) of Page rank of destination node	23
7.4	Box plot and Histogram (after standardisation) of Shortest path between the nodes	24
7.5	Box plot and Histogram (after standardisation) of Follows back feature	24
7.6	Box plot and Histogram (after standardisation) of followees of destination node	24
7.7	Box plot and Histogram (after standardisation) of followees of source node	25
7.8	Box plot and Histogram (after standardisation) of followers of destination node	25
7.9	Box plot and Histogram (after standardisation) of followers of source node	25
7.10	Box plot and Histogram (after standardisation) of common followees between the nodes	26
7.11	Box plot and Histogram (after standardisation) of common followers between the nodes	26
7.12	Box plot and Histogram (after standardisation) of Preferential attachment of source node	26
7.13	Box plot and Histogram (after standardisation) of Preferential attachment of destination node	27
7.14	Time taken to train model using logistic regression	29

7.15	Best parameters - logistic regression	29
7.16	Best estimators - logistic regression	30
7.17	Time taken to train model using random forest	31
7.18	Best parameters - random forest	31
7.19	Best estimators - random forest	31
7.20	Visualisation of random forest	32
7.21	Feature importance	32
7.22	Time taken to train model using SVM	34
7.23	Best parameters – SVM	34
7.24	Best estimators - SVM	34
7.25	Time taken to train model using adaboost	35
7.26	Best parameters – adaboost	36
7.27	Best estimators - adaboost	36
7.28	Time taken to train model using xgboost	37
7.29	Best parameters - xgboost	37
7.30	Best estimators - xgboost	37
8.1	ROC-AUC Curve for Logistic Regression	39

8.2	ROC-AUC Curve for SVM	40
8.3	ROC-AUC Curve for Random Forest	40
8.4	ROC-AUC Curve for AdaBoost	40
8.5	ROC-AUC Curve for XGBoost	41
8.6	Confusion Matrix	44
8.7	Confusion Matrix for Logistic Regression	44
8.8	Confusion Matrix for Random Forest	44
8.9	Confusion Matrix for Support Vector Machine	45
8.10	Confusion Matrix for AdaBoost	45
8.11	Confusion Matrix for XGBoost	45
8.12	ML Model Performance Metrics	46
9.1	Feature importance	48

LIST OF ABBREVIATIONS

ML	Machine Learning
SVM	Support Vector Machine
ROC	Receiver Operating Characteristic Curve
AUC	Area Under the ROC Curve
EDA	Exploratory Data Analysis

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Link prediction is the problem of predicting the formation of new edges on a given network. It is a fundamental problem that applies to networking in numerous contexts, including the Internet, the web, and online social networks [1]. The problem can be stated as follows - given a snapshot of a dynamic social network, represented as a directed graph of nodes and edges (where nodes represent users and edges represent follower - followee relations) is it possible to predict which new relations (links) are likely to be formed in the future.

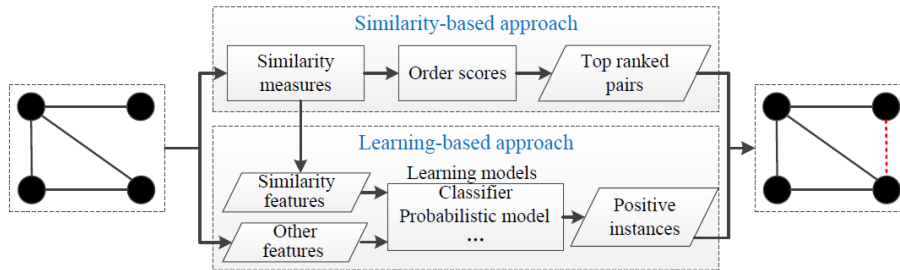


Figure 1.1 Workflow of link prediction problem

Numerous edges and vertices are added and/or removed throughout time as a consequence of changing individual connections. Consequently, social networks become very dynamic and complex [2]. Identifying the mechanisms by which they evolve is a fundamental question that is still not well understood, and it forms the motivation for our work here.

1.2 NEED FOR THE STUDY

From the link prediction research perspective, many supervised machines learning (ML) classification techniques may be used to address the link prediction problem. Multiple studies have shown that this method delivers good results; nevertheless, choosing the collection of features (variables) to train classifiers remains a key challenge. The issue of link prediction in dynamic or time-varying networks involves two key challenges: precision and efficiency.

Additionally, because similarity-based link prediction algorithms were first developed for static networks, it is crucial to choose which similarity-based approaches and machine learning algorithms to use when applying them to dynamic networks [3]. Extensive experiments with different machine learning algorithms are conducted to address this issue as well.

1.3 MOTIVATION OF THE STUDY

- As a technical problem, the efficacy of link prediction is generally not well understood. Today, link prediction algorithms are the basis for social recommendations in a wide range of social networks and applications, ranging from Facebook and Pinterest and Q&A sites like Quora. The success of these sites and the sheer volume of prior literature lead many to believe the problem is well addressed. Only evidence to the contrary comes from failed recommendations.

- Link prediction is frequently used in social networks to suggest friends to users. It has also been used to predict criminal associations.
- In biology, link prediction has been used to predict interactions between proteins in protein-protein interaction networks. Link prediction has also been used to infer interactions between drugs and targets using link prediction.
- Link can be used to improve marketing strategies, in particular viral marketing.

1.4 OBJECTIVES OF THE STUDY

- The objective is to analyse the link prediction in dynamic social networks using supervised machine learning algorithm and choosing the best model out of it.
- Another objective is to use graph theory algorithms [11] to analyse the link prediction in social network.

CHAPTER 2

REVIEW OF LITERATURE

2.1 INTRODUCTION

The base paper for analysing the link prediction problem is **The Link-Prediction Problem for Social Networks** by *Liben-Nowell* and *Kleinberg* [1]. This paper was a seminal work in this domain as it formalized the link prediction problem and established a fundamental concept that it is possible to infer future changes in a network based solely on features intrinsic to the network itself. This paper firmly established the idea that there is a notion of proximity that can be used to predict future links and the network contains certain latent information in the form of its topology which can be leveraged to infer future interactions. This concept is used to extract features using basic proximity measures of the network.

Link Prediction in Social Networks: the State-of-the-art, by *Wang Peng et al* [2], is one of the most cited survey papers which speaks about the idea behind link prediction, its applications and different ways to tackle it. The authors have given a general framework for solving the link prediction problem which involves similarity based prediction or learning based prediction. In similarity based prediction, every possible future link is assigned a score based on similarity between the nodes and higher score edges are likely to appear in future. In learning based approach, link prediction is seen as a binary classification problem. Furthermore, the paper also describes several metrics such as path based metrics or neighbour based metrics and their relevance in predicting links

in a given graph. Overall the paper is the most comprehensive and gives direction to solve the link prediction problem.

Review on Learning and Extracting Graph Features for Link Prediction [12], by *Ece C. Mutlu, Toktam Oghaz, Amirarsalan Rajabi and Ivan Garibay*. This work presents an extensive review of state-of-art methods and algorithms proposed on this subject and categorizes them into four main categories: similarity-based methods, probabilistic methods, relational models, and learning-based methods.

Another important paper that considered was **Network Growth and Link Prediction Through an Empirical Lens** by *Liu, et al* [3]. This paper is relatively recent and less cited but it proves the importance of using machine learning models as opposed to purely mathematical and statistical distance metrics for link prediction. The authors of this paper implemented 18 link prediction algorithms each categorized as ‘metric-based’ (using a single similarity or proximity metric) or ‘classification-based’ (using machine learning classifiers with multiple metrics as input features). The authors evaluated these algorithms on large detailed network traces obtained from Facebook, Youtube and RenRen. They firmly established that SVM consistently outperforms all metric based methods across all three networks. Based on their findings, only machine learning classifiers were implemented instead of single similarity metrics.

The method proposed in **Transitive Node Similarity for Link Prediction in Social Networks with Positive and Negative Links** [4], by *Panagiotis Symeonidis et al* , takes into account both local and global features of a graph. It states that the probability of existence of an edge

directly depends on length of pathway between the nodes and similarity between neighbours of the two nodes. The authors define a proximity function based on Jaccard Distance using which node similarity matrix and transitive similarity matrix is constructed. The authors describe how link prediction can be done based on the transitive similarity and how this concept can be extended to signed networks as well.

Grid search in hyperparameter optimization of machine learning models for prediction [13], by *Daniel Mesafint Belete, Manjaiah D H.*

This paper concentrated on the hyper parameter optimization using grid search methods on the dataset that contains binary class. The main objective of this paper is to generate the optimal parameter from the default parameter of each model which is used in the study using grid search methods and applying the optimal parameters to each model to test the effects of the methods for the prediction.

2.2 SUMMARY

In the Literature Survey above, numerous studies ranging from fundamental concept behind link prediction problem to the latest developments in the field have been cited. The section starts with studies on the advancements of machine learning. Following which multiple research articles and papers were referred to explore the involvement of various ML models in analyzing the link prediction in social network. This study will expand the scope of these findings by using various ML models and graph theory algorithms [11] on large skewed dataset.

CHAPTER 3

SYSTEM DESIGN

In this chapter, the overall design of the system is discussed. The workflow is introduced and is briefly explained [17]. The workflow consists of the following broad development phases:

- Data Collection and Preparation
- Feature Extraction
- Model Building

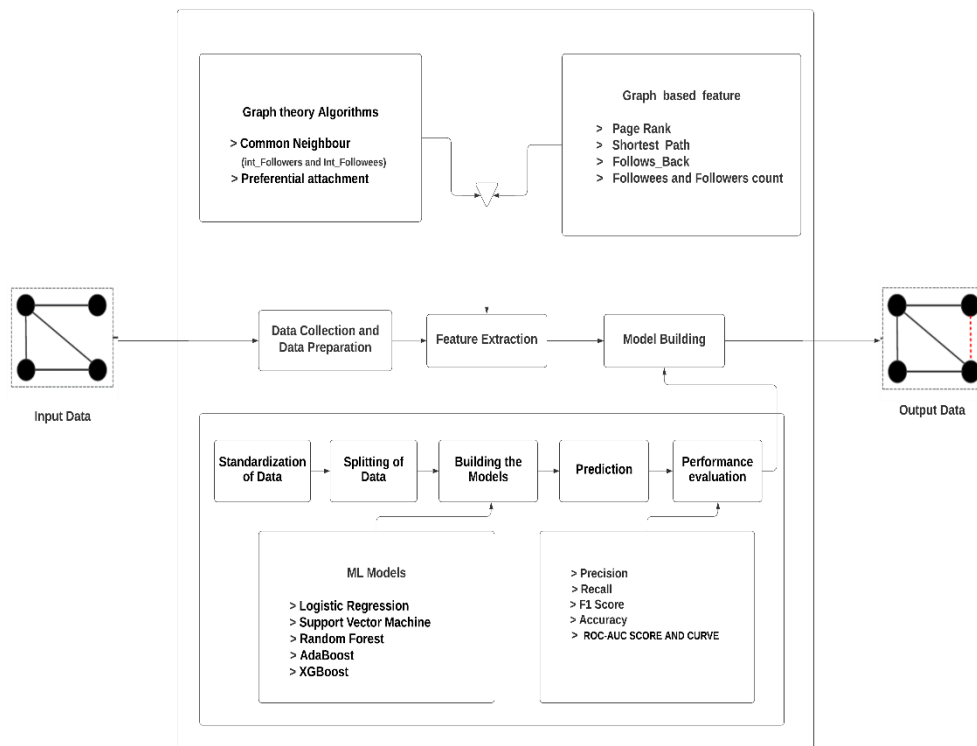


Figure 3.1 System Architecture

The workflow of the system illustrated in Figure 3.1 is as follows:

3.1 DATA COLLECTION AND PREPARATION

The dataset was obtained from the Stanford Large Network Dataset Collection of more than 50 large network datasets from tens of thousands of nodes and edges to tens of millions of nodes and edges [5]. The dataset contains details about from and to nodes. In this phase the dataset is converted into directed graph and then missing edges that equals the edges that are present were randomly selected in order to avoid the imbalance in dataset[4].

3.2 FEATURE EXTRACTION

Feature extraction is the most important phase of the study. In this phase, the features are extracted from the dataset based on graph based features such as page rank, shortest path, follows back, follower & followee counts [12] and graph theory algorithms such as common neighbours and preferential attachment score [11].

3.3 MODEL BUILDING

In this phase, firstly the correlation between the features are studied using heatmap and exploratory data analysis is performed. Based on the data that is analyzed some of the features are dominating the others. To avoid this skewness standardization is performed since all the features will be given equal weightage while training a machine learning model. Here 70% of the data is used for training and the remaining 30% for testing the model. In this classification problem, five models namely, Logistic Regression, Random Forest, SVM, AdaBoost and XGBoost are trained.

Motivation For Using Classification ML Algorithms :

Classification in machine learning is a supervised ML technique. It predicts group relationships for data instances in the dataset. It is a recursive process of recognizing, understanding and grouping the data objects using labels into pre-defined categories. Classification algorithms are used in Machine Learning to predict the class label of a given data point. It allows machines to learn and predict new data points, even when no class labels are known.

- Linear regression is utilized for regression tasks, while logistic regression helps accomplish classification tasks.
- Linear regression is used to predict the continuous dependent variable using a given set of independent variables whereas Logistic Regression is used to predict the categorical dependent variable using a given set of independent variables.

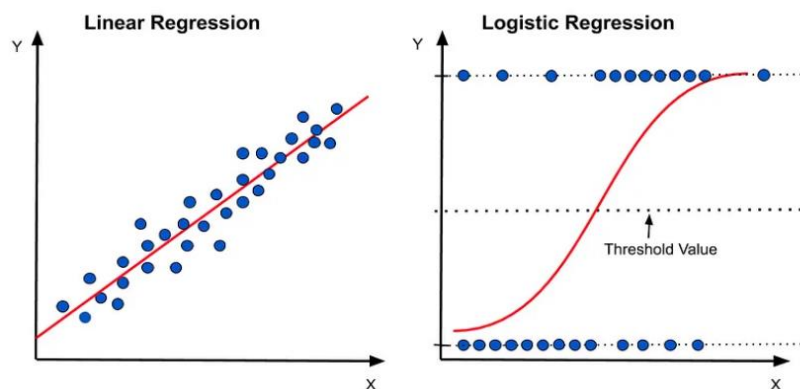


Figure 3.2 Comparing Logistic regression with Linear regression

In this phase, Grid Search [13] is used on each model in order to estimate the best parameters. Using Grid Search with cross validation, the model is trained on several possible combinations of parameters and use the validation accuracy to determine the best parameters. Then this best model is used to predict on the test data.

CHAPTER 4

TOOLS AND TECHNOLOGIES USED

4.1 Google Colab

Google Colaboratory ("Colab" for short) is a data analysis and machine learning tool that allows you to combine executable Python code and rich text along with charts, images, HTML, LaTeX and more into a single document stored in Google Drive. It connects to powerful Google Cloud Platform runtimes and enables you to easily share your work and collaborate with others.

4.2 numpy

NumPy is a python based external library that is imported to deal with data structures like arrays, sophisticated broadcasting functions, linear algebra, fourier transform, etc. NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows seamless and speedy integration with a wide variety of databases.

4.3 pandas

Pandas is a BSD - licensed library used to provide ready to use data structures and data analysis tools for Python. Pandas library is used to make data more interactive and flexible. The two major data structures supported by pandas are Series- a 1D data structure and a DataFrame - a 2D data structure. These data structures are used the most in various data analysis procedures.

4.4 matplotlib

Matplotlib is a visualization library supported by python. It can be used to create dynamic, static and interactive visualizations using python.

Matplotlib.pyplot is a collection of functions that makes matplotlib work like MATLAB.

4.5 networkx

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks [14]. With NetworkX one can load and store networks in standard and nonstandard data formats, generate many types of random and classic networks, analyze network structure, build network models, design new network algorithms, draw networks, and much more.

4.6 pickle

Pickling is a process by which data in object hierarchy is converted into a byte stream. Unpickling is a process by which a byte stream is converted into a data object. Pickle is an external library which implements binary protocols for serializing and deserializing data. One can easily save different variables into a pickle file and load them back in a different Python session, recovering the data exactly the way it was without having to edit the code.

4.7 sklearn

Scikit - learn is one of the eminent machine learning packages available in python. It is largely written in python and is built upon NumPy, SciPy and Matplotlib. Scikit Learn focuses more on data modeling. It contains supervised models like Regression and Unsupervised models like Clustering to which the data set can be subjected to. It also contains a

few predefined ensemble models which can be used to add up predictions of the individual model.

4.8 xgboost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable [15]. The xgboost package is an external set of library files that need to be imported to implement the XGBoost algorithm. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way.

4.9 seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. It aims to make visualization the central part of exploring and understanding data.

CHAPTER 5

DATA COLLECTION AND PREPARATION

5.1 DATA COLLECTION

Twitch is used widely by gamers to live-stream themselves while playing games. The nature of the platform is such that there are few popular gamers with many followers. The twitch dataset [5] was chosen as there has not been much link prediction work done on this previously.

5.2 UNDERSTANDING THE DATA

The twitch dataset contains details about from and to nodes. In this phase the dataset is converted into directed graph.

from	to
6194	255
6194	980
6194	2992
6194	2507
6194	986
6194	4003
0	82
15	343
15	4282

Figure 5.1 Outline of the dataset

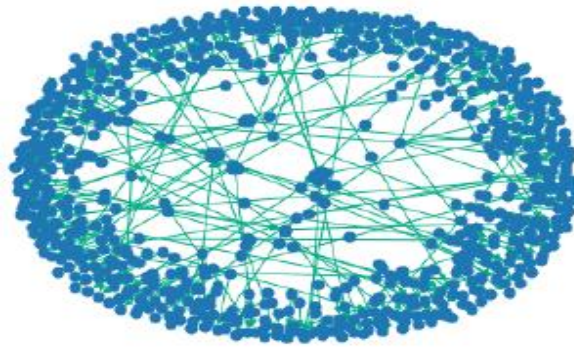


Figure 5.2 Visualization of a random sample of 500 edges from the dataset

The dataset was reasonably sized with 7126 nodes. The total number of possible edges in the network is 50,772,750 from which 35,324 are present in the network.

```
DiGraph with 7126 nodes and 35324 edges
```

Figure 5.3 Information about the graph before data preparation

```
Maximum indegree - Node : 88  Indegree : 465
Average indegree - 4.957058658433904
```

Figure 5.4 In degree of the node

```
Maximum out degree - Node : 54  Out degree : 540
Average out degree - 4.957058658433904
```

Figure 5.5 Out degree of the node

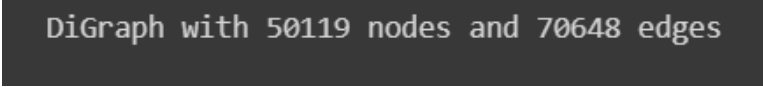
	Source	Destination	Class
0	6194	255	1
1	6194	980	1
2	6194	2992	1
3	6194	2507	1
4	6194	986	1

Figure 5.6 Present edges

Using all the missing edges from the graph would highly skew the dataset so 35,324 missing edges are randomly sampled. While sampling these missing edges, an edge is considered as missing if the distance between the source and destination was more than 2 as closely connected users are likely to be mutual friends even if an edge does not already exist. This is to ensure the model is able to properly distinguish between present and absent edges[4], thus improving its performance. This presence or absence of an edge is used as the target class variable for prediction.

	Source	Destination	Class
0	10847	50740	0
1	49613	19406	0
2	79713	11464	0
3	36081	36754	0
4	17670	39939	0

Figure 5.7 Missing edges



DiGraph with 50119 nodes and 70648 edges

Figure 5.8 Information about the graph after data preparation

5.3 ASSUMPTIONS

- Using all the missing edges from the graph would highly skew the dataset so the missing edges are randomly sampled to avoid skewed dataset
- While sampling these missing edges, a condition is added, i.e. to only consider an edge as missing if the distance between the source and destination was more than 2 as closely connected users are likely to be mutual friends even if an edge does not already exist.
- The edges that are present as 1 and the missing ones as 0. This presence or absence of an edge is used as the target class variable for prediction.
- Six-degree phenomenon is taken into account while calculating shortest path between 2 nodes [9].

CHAPTER 6

FEATURE EXTRACTION

In this chapter the features are extracted from the dataset based on graph based features [12] and graph theory algorithms [11]. Here 70,648 edges is used to extract various features such as –

- Page Rank
- Shortest Path
- Follows Back
- Follower & Followee counts
- Inter Followers & Followee counts (Common neighbours)
- Preferential Attachment score

6.1 PAGE RANK

Standardisation PageRank is the algorithm used by Google Search to rank websites in their search engine results. It works by counting the number and quality of links to a page to determine a rough estimate of how important the website is [6]. The underlying assumption is that more important websites are likely to receive more links from other websites. Accordingly, the page ranks are calculated for both source and destination nodes of each edge and these formed two of the features.

$$PR(P_i) = \frac{(d)}{n} + (1-d) \times \sum_{l_{j,i} \in E} PR(P_j) / \text{Outdegree}(P_j)$$

d - damping factor (0.1~0.15)

n - |page set|

Equation (6.1) Page Rank formula

6.2 SHORTEST PATH

The shortest path between two nodes is simply the path with the least number of intermediate nodes [4]. In this feature if a direct link already exists between two nodes, it is first deleted and then the shortest path is calculated between them. The intuition behind this feature is that nodes which are close to each other have shorter path lengths indicating that they are likely to be good recommendation candidates. Here six degree phenomenon is used as mentioned in the assumptions[9].

$$SP(x,y) = \min (|\text{path}(x,y)|)$$

$|\text{path}(x,y)|$ - all possible distances between x and y node

$$SP(x,y) = \begin{cases} SP(x,y) , & \text{if } SP(x,y) \leq 6 \\ -1 & , \text{otherwise} \end{cases}$$

Equation (6.2) Shortest path formula

6.3 FOLLOWS BACK

This feature simply indicates whether a reversely directed edge exists in the network for each existing edge, i.e. whether a user follows back one of his followees.

$$FB(x,y) = \begin{cases} 1, & \text{if edge exist from y to x} \\ 0, & \text{otherwise} \end{cases}$$

Equation (6.3) Follows back formula

6.4 FOLLOWER AND FOLLOWEE COUNT

These features are the number of followers and followees of source and destination nodes. The intuition here is that popular streamers have a large number of followers and are good choices for recommendation candidates.

Follower_count(x) = No. of incoming links for node x

Followee_count(x) = No. of outgoing links for node x

Equation (6.4) Follower and followee count formula

6.5 COMMON NEIGHBOUR

Common Neighbours is a similarity measure used in link prediction that captures the notion that a common friend may introduce two strangers to each other [16]. It tells us about the total number of common friends a pair of nodes have. The common-neighbours predictor computes the number of common neighbours between two

nodes and entities with more neighbours in common are more likely to have a link

$$CN(x, y) = |N(x) \cap N(y)|$$

$N(z)$ – neighbours of z node

Equation (6.5) Common neighbour formula

6.6 PREFERENTIAL ATTACHMENT

Preferential attachment [18] means that the more connected a node is, the more likely it is to receive new links. A value of 0 indicates that two nodes are not close, while higher values indicate that nodes are closer.

$$PA(x, y) = |N(x)| * |N(y)|$$

$N(u)$ - set of nodes adjacent to u

Equation (6.6) Preferential attachment formula

After all the necessary features are extracted using graph based features and algorithms from the given dataset, the data is illustrated in Figure 6.1

Class	Page_Rank_Src	Page_Rank_Dst	Shortest_Path	Follows_Back	Followers_Src	Followees_Src	Followers_Dst	Followees_Dst	Int_Followers	Int_Followees	Preattach_Src	Preattach_Dst
1	0.000013	0.000021	-1	0	0	3	0	4	0	0	3	4
1	0.000013	0.000035	-1	0	0	3	0	15	0	1	3	15
1	0.000013	0.000014	4	0	0	3	3	0	0	0	3	3
1	0.000013	0.000019	3	0	0	3	8	14	0	0	3	22
1	0.000013	0.000014	-1	0	0	3	0	19	0	0	3	19
...
0	0.000019	0.000027	-1	0	0	0	0	0	0	0	0	0
0	0.000010	0.000012	-1	0	0	1	0	0	0	0	1	0
0	0.000019	0.000021	-1	0	1	1	0	0	0	0	2	0
0	0.000010	0.000019	-1	0	0	0	0	0	0	0	0	0
0	0.000027	0.000032	-1	0	18	7	0	0	0	0	25	0

Figure 6.1 Features extracted

CHAPTER 7

MODEL BUILDING

In this chapter, the following steps are executed –

- Analysis and Standardisation of data
- Splitting of data
- Building ML models
- Prediction

7.1 ANALYSIS AND STANDARDISATION OF DATA

For analysing the features that are extracted, heatmap is plotted. A heatmap of the features uses colours to indicate the strength and direction of the correlation between each pair of features. Correlation is a measure of how much two features vary together.

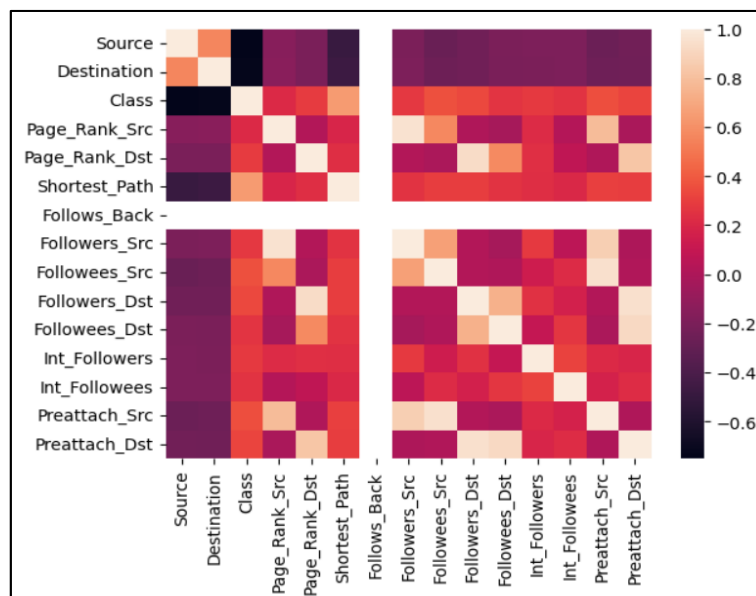


Figure 7.1 Heatmap

From Figure 7.1, features that are related to each other and features are independent are identified.

Exploratory data analysis is a method used to analyse and investigate data sets and summarize their main characteristics, often using data visualization methods. EDA is an important step in any data analysis, as it can help to look at data before making any assumptions.

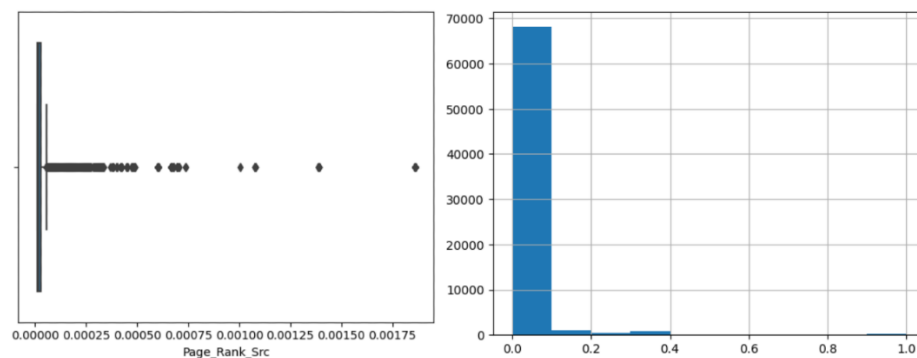


Figure 7.2 Box plot and Histogram (after standardisation) of Page rank of source node

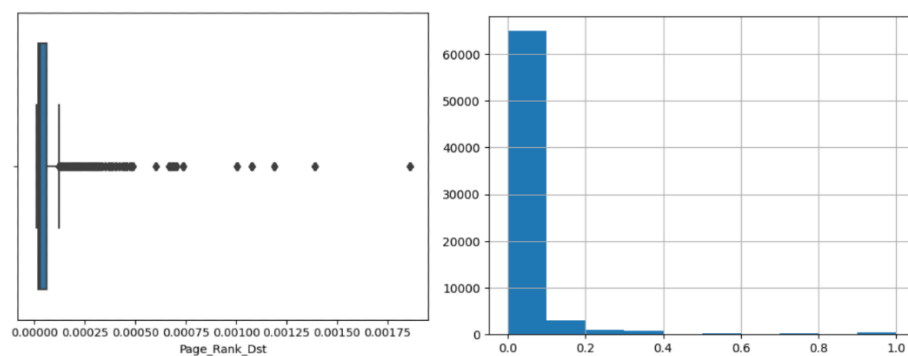


Figure 7.3 Box plot and Histogram (after standardisation) of Page rank of destination node

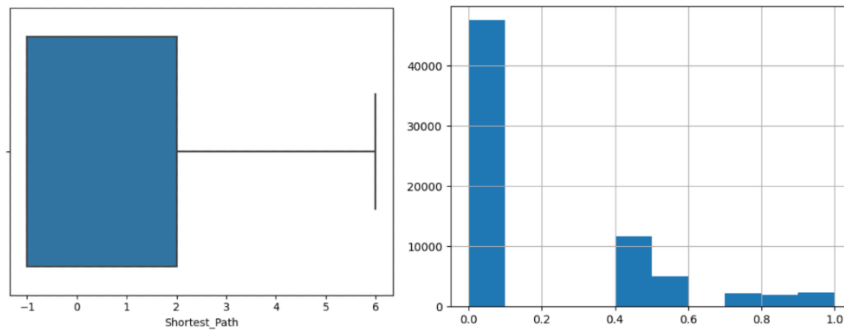


Figure 7.4 Box plot and Histogram (after standardisation) of Shortest path between the nodes

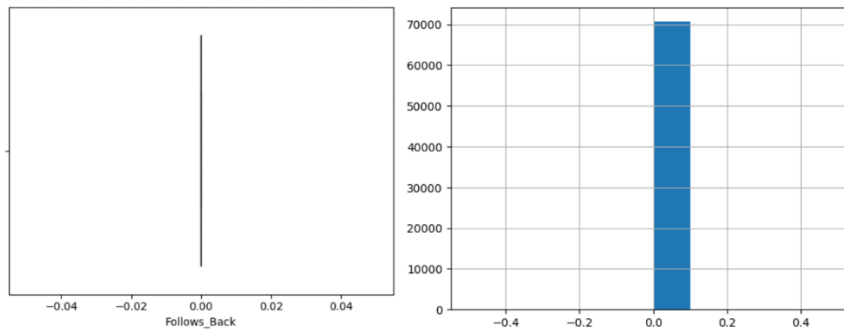


Figure 7.5 Box plot and Histogram (after standardisation) of Follows back feature

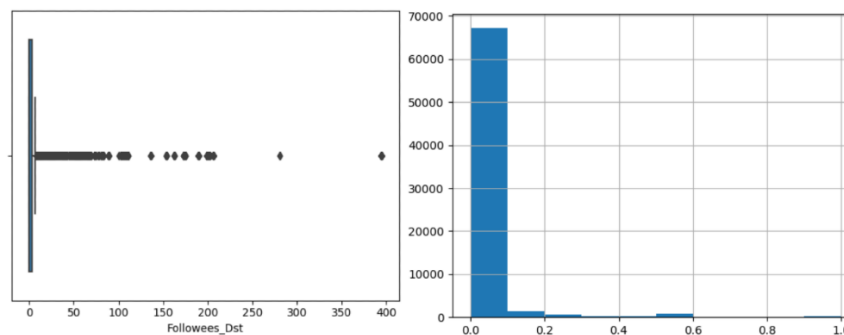


Figure 7.6 Box plot and Histogram (after standardisation) of followees of destination node

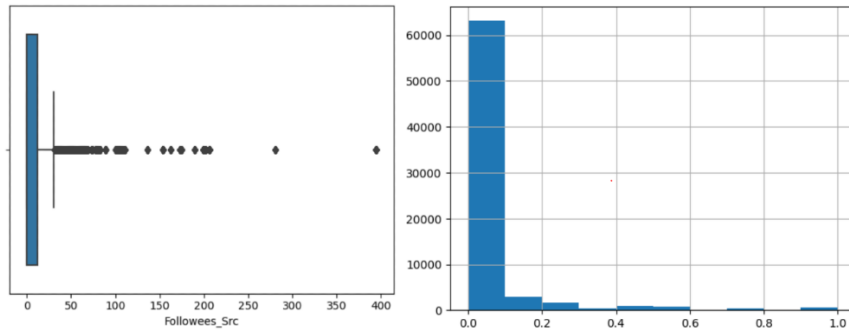


Figure 7.7 Box plot and Histogram (after standardisation) of followers of source node

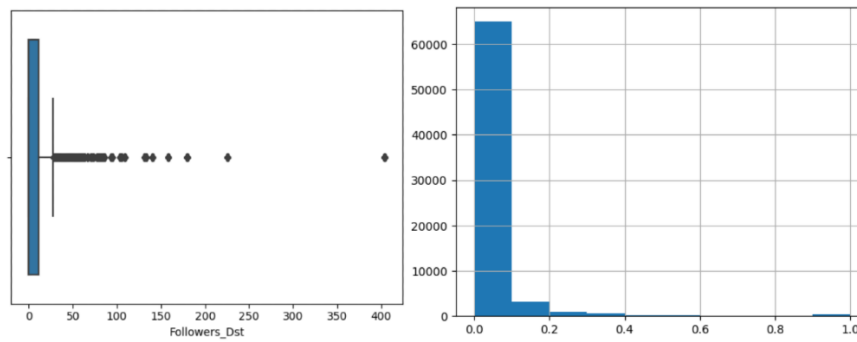


Figure 7.8 Box plot and Histogram (after standardisation) of followers of destination node

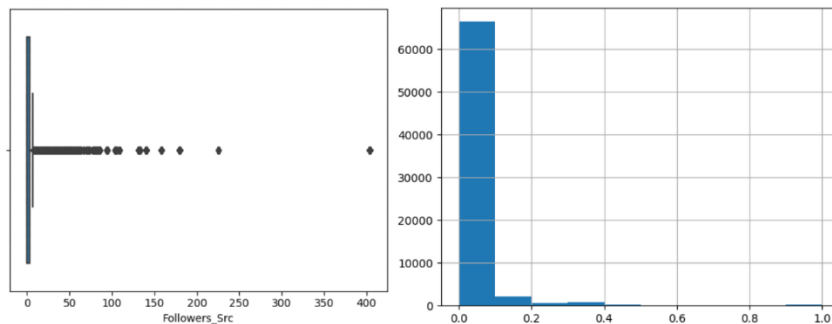


Figure 7.9 Box plot and Histogram (after standardisation) of followers of source node

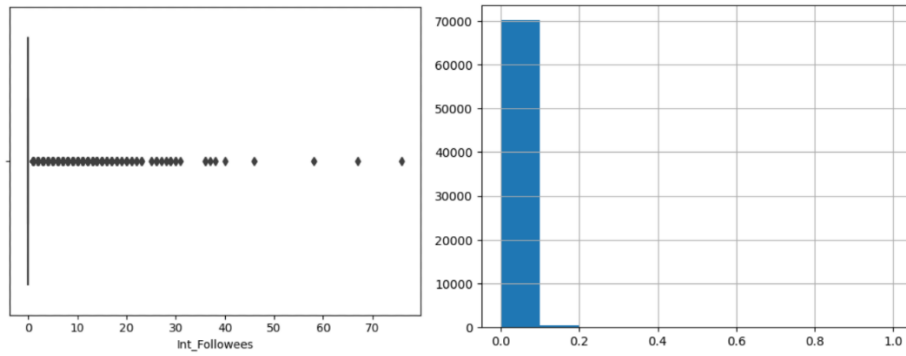


Figure 7.10 Box plot and Histogram (after standardisation) of common followers between the nodes

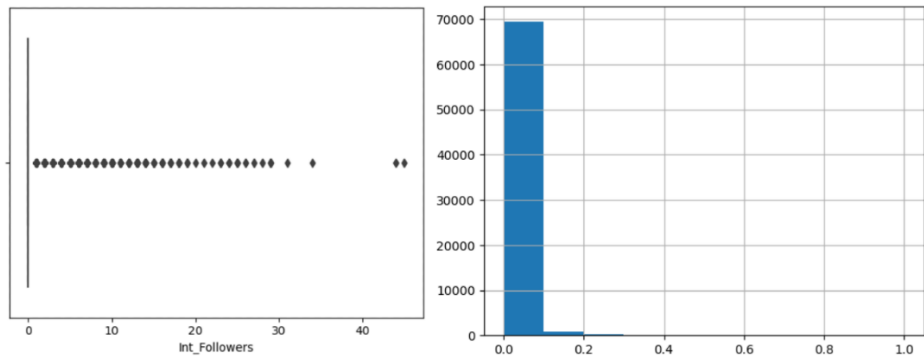


Figure 7.11 Box plot and Histogram (after standardisation) of common followers between the nodes

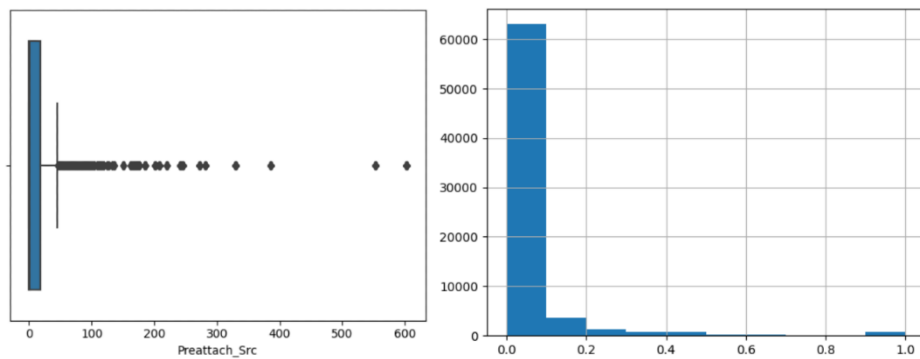


Figure 7.12 Box plot and Histogram (after standardisation) of Preferential attachment of source node

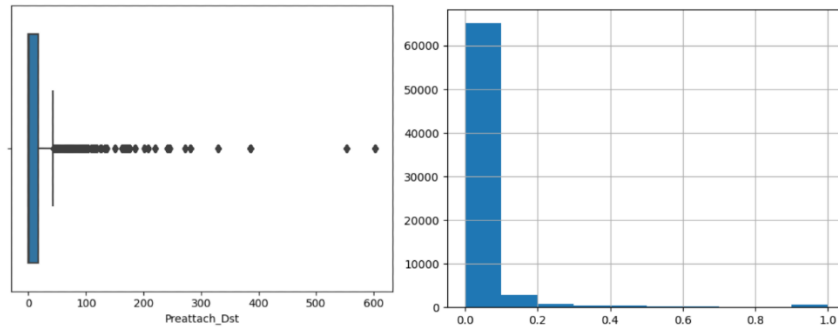


Figure 7.13 Box plot and Histogram (after standardisation) of Preferential attachment of destination node

Here for all the features that are extracted, EDA (Box plot) is performed. Based on the results observed from EDA outliers have been identified for all the features. Thus standardization of data is implemented for all the features. Standardisation is required to bring the data to the same scale. In case the data is not standardised, we get skewed results as one or more features might dominate the others. For example, features like number_of_followers have a wider distribution than page_rank because of which number_of_followers will dominate over page_rank. However, after standardisation, all values lie between 0 to 1 and will be given equal weightage while training a machine learning model.

7.2 SPLITTING OF DATA

In order to test the model on unseen data, it is necessary to split the data and this should be done randomly to avoid bias in the training or the testing phase. Random split ensures that the model is trained on edges which belong to both the classes (1 and 0). Here 70% of the data are used for training and the remaining 30% for testing the model.

7.3 BUILDING ML MODELS

7.3.1 LOGISTIC REGRESSION

Logistic Regression is one of the oldest classification algorithms. It is used when the value to predict is categorical in nature, commonly when the data has binary output. A hypothesis function is drawn to fit to the binary dataset. Logistic function uses a sigmoid function which is a 'S' shaped curve. A threshold is used to classify the test set records. In general, the threshold is set to 0.5.

Algorithm:

1. Initialize the logistic regression model with random weights
2. Split the data into training and testing sets
3. For each epoch:
 - a. Shuffle the training set
 - b. For each training example:
 - i. Compute the predicted output using the logistic regression formula
 - ii. Compute the error between the predicted output and the true output
 - iii. Update the weights using gradient descent
 - c. Compute the accuracy on the training and testing sets
4. Output the final weights and accuracy metrics

Here is the logistic regression formula:

$$y = 1 / (1 + \exp(-z))$$

where y is the predicted output (between 0 and 1), and z is the input to the sigmoid function, which is the linear combination of the features and their corresponding weights:

$$z = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

where w_0 is the bias term, and x_i are the features.

The weights are updated using gradient descent:

$$w_j = w_j - \text{learning_rate} * (\text{error} * x_i + \text{regularization_term} * w_j)$$

where j is the index of the weight, learning_rate is the step size for the gradient descent, error is the difference between the predicted and true output, $\text{regularization_term}$ is a penalty term to prevent overfitting, and x_i is the j -th feature.

```
Time taken to train model : 45.91 seconds
```

Figure 7.14 Time taken to train model using logistic regression

```
{'C': 3593.813663804626, 'penalty': 'l2'}
```

Figure 7.15 Best parameters - logistic regression

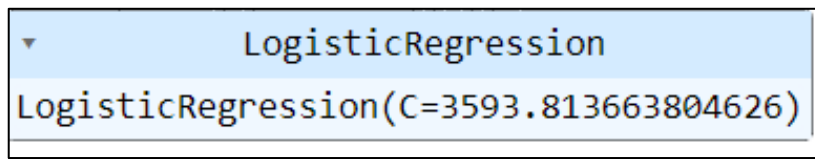


Figure 7.16 Best estimators - logistic regression

7.3.2 RANDOM FOREST

Random Forest Classifier is one of the bagging models. Here, while selecting the records for different models, row and column sampling with replacement happens, which means both the features and the records are sampled to data subsets. These subsets are then sent to multiple decision tree models for training. Majority vote of learners present in the random forest is considered as the final prediction on a test set. In random forest classifier the base learner is the decision tree. A decision tree with [7] intense depth has low bias and high variance which leads to overfitting of models. So, the final model is not relying on single models prediction instead multiple learners prediction is used. This reduces the variance, and every decision tree learns properly from the subset given to it.

Algorithm:

1. Input the training set of nodes and edges and the number of trees to grow in the random forest.
2. For each tree to be grown in the random forest:
 - a. Randomly select a subset of nodes and edges from the training set with replacement.
 - b. Construct a decision tree using the selected subset of nodes and edges.

- c. Use the decision tree to predict the presence or absence of links between the remaining nodes.
3. Output the random forest model, which consists of the collection of decision trees.
4. Input the test set of nodes and edges.
5. For each pair of nodes in the test set:
 - a. Use each decision tree [8] in the random forest to predict the presence or absence of a link between the nodes.
 - b. Aggregate the predictions across all decision trees to obtain a final prediction for the pair of nodes.
6. Output the predictions for all pairs of nodes in the test set.

```
Time taken to train model : 1502.98 seconds
```

Figure 7.17 Time taken to train model using random forest

```
{'max_depth': 9, 'min_samples_leaf': 21, 'min_samples_split': 120, 'n_estimators': 100}
```

Figure 7.18 Best parameters - random forest

```
RandomForestClassifier
RandomForestClassifier(max_depth=9, min_samples_leaf=21, min_samples_split=120)
```

Figure 7.19 Best estimators - random forest

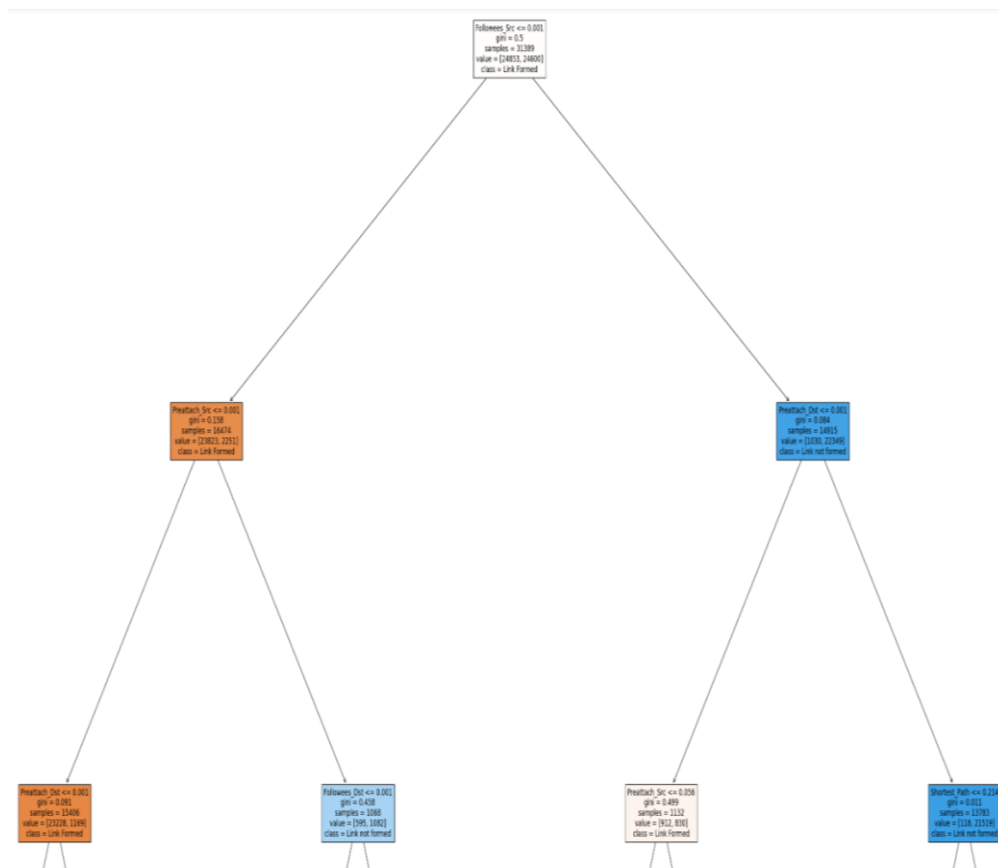


Figure 7.20 Visualisation of random forest

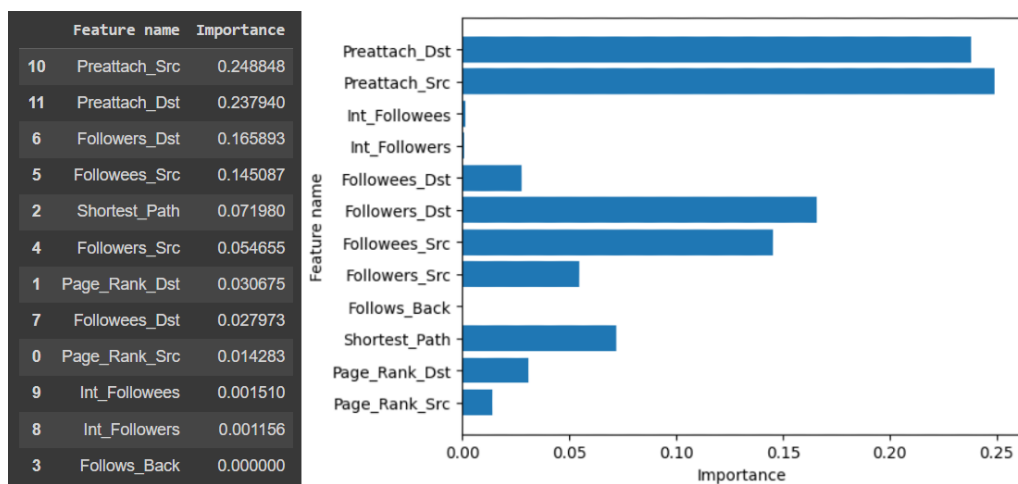


Figure 7.21 Feature importance

7.3.3 SUPPORT VECTOR MACHINE

SVM [3] is a simple classification algorithm. It strives to find the best line which helps to separate classes in space. In this model, the space is 2-Dimensional and the best line is called a hyperplane. The model looks at the very extreme case which is very close to the boundary and uses it to construct its analysis. These extreme cases form the support vectors which are the closest points of both the classes to the hyperplane. For the test records, when the point lies on the negative side of the plane, it is considered as one class and if it lies on the positive side of the hyperplane it is considered as another class. ‘Sigmoid’ type of kernel is used in the presented model.

Algorithm:

1. Input the training set of nodes and edges and their respective labels indicating whether a link exists or not.
2. Pre-process the input data by transforming each node into a feature vector, where each feature represents some characteristic of the node.
3. Train the SVM model using the training set of labelled nodes and edges, where the goal is to find a hyperplane that maximally separates the positive and negative labelled data points.
4. Input the test set of nodes and edges.
5. For each pair of nodes in the test set:
 - a. Transform each node into a feature vector using the same pre-processing as in step2.
 - b. Use the trained SVM model to predict the label for the pair of nodes.
6. Output the predictions for all pairs of nodes in the test set.

```
Time taken to train model : 16043.98 seconds
```

Figure 7.22 Time taken to train model using SVM

```
{'C': 1, 'kernel': 'rbf'}
```

Figure 7.23 Best parameters – SVM

```
▼ SVC  
SVC(C=1, probability=True)
```

Figure 7.24 Best estimators - SVM

7.3.4 ADABOOST

Adaboost is one of the earliest boosting algorithms. Here, every observation is given an initial weight $1/n$, where n is the total number of observations [10]. Initially, a weak model, mostly a decision tree is trained. Using this model, predictions are done. If the model predicts the given observation incorrectly, the weight of the observation is increased and the opposite is true for correct prediction.

Next, a new model is trained where incorrectly predicted observations are given more weight. All the above steps are repeated till the model predicts perfectly. Adaboost works well on highly imbalanced datasets.

Algorithm:

1. Input: a set of training examples, each consisting of a pair of nodes (u,v) and a binary label y (1 if there is a link between u and v , 0 otherwise).

2. Initialize the weight vector w for each training example as $1/n$, where n is the total number of examples.
3. For $t = 1$ to T , do the following:
 - a. Train a weak classifier h_t using the current weight vector w .
 - b. Calculate the error rate ϵ_t of h_t on the training set, weighted by the current weight vector w .
 - c. Calculate the weight α_t of h_t as $\alpha_t = \log((1 - \epsilon_t)/\epsilon_t)$.
 - d. Update the weight vector w as follows:
 - i. For each correctly classified example (u,v) , multiply its weight by $\exp(-\alpha_t)$.
 - ii. For each incorrectly classified example (u,v) , multiply its weight by $\exp(\alpha_t)$.
 - iii. Normalize the weight vector w so that it sums to 1.
4. Output the final strong classifier H as a weighted combination of the T weak classifiers, where the weight of each classifier is proportional to its α_t value.
5. To make a prediction for a pair of nodes (u,v) , compute the score as $H(u,v) = \sum_{t=1}^T \alpha_t h_t(u,v)$. If the score is above a threshold, predict that there is a link between u and v , otherwise predict no link.

```
Time taken to train model : 2669.09 seconds
```

Figure 7.25 Time taken to train model using adaboost

```
{'learning_rate': 1.0, 'n_estimators': 50}
```

Figure 7.26 Best parameters – adaboost

```
▼ AdaBoostClassifier  
AdaBoostClassifier()
```

Figure 7.27 Best estimators – adaboost

7.3.5 XGBOOST

XGBoost is designed to work with large and complicated datasets. It has become the most used model in many competitions. The base algorithm for XGBoost is the gradient boosting algorithm, which is sequential, where the base model calculates the average of the predictions and residual is calculated from that for every record which is sent to the first decision tree. The residual of the first decision tree is captured and given as input to the next decision tree. Each tree in sequential learning has a learning rate associated with it in order to reduce overfitting of the model. In XGBoost information gain is used to develop a tree.

Algorithm:

1. Initialize the model parameters, including the number of iterations, learning rate, maximum tree depth, minimum child weight, subsampling rate, and column subsampling rate.
2. Then split the data into training and validation sets, and convert the data into the DMatrix format required by XGBoost.

3. Then set the XGBoost parameters, including the objective function and the optimization algorithm.
4. Train the model on the training data, and use the validation set to monitor the model's performance and prevent overfitting using early stopping.
5. Finally, make predictions on the test data using the trained model.

```
Fitting 3 folds for each of 405 candidates, totalling 1215 fits
[13:19:44] WARNING: ../src/learner.cc:767:
Parameters: { "silent" } are not used.

Time taken to train model : 13764.85 seconds
```

Figure 7.28 Time taken to train model using xgboost

```
{'colsample_bytree': 1.0, 'gamma': 0.5, 'max_depth': 5, 'min_child_weight': 5, 'subsample': 0.6}
```

Figure 7.29 Best parameters - xgboost

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=1.0, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=0.5, gpu_id=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=0.02, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=5, max_leaves=None,
               min_child_weight=5, missing=nan, monotone_constraints=None,
               n_estimators=600, n_jobs=None, nthread=1, num_parallel_tree=None,
               predictor=None, ...)
```

Figure 7.30 Best estimators – xgboost

7.4 PREDICTION

Once the models are built, 30% of the data is used to test the model. In this phase the features are fed as input and the model's prediction are compared to the ground truth.

CHAPTER 8

PERFORMANCE ANALYSIS

8.1 PERFORMANCE METRICS

8.1.1 ROC-AUC CURVE AND SCORE

The ROC (Receiver Operating Characteristic) curve is a graphical representation of the performance of a binary classifier system as its discrimination threshold is varied¹. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two groups (Edge present /(Edge Absent). AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

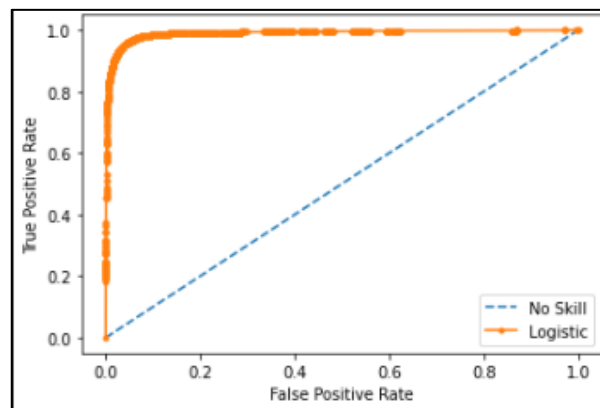


Figure 8.1 ROC-AUC Curve for Logistic Regression

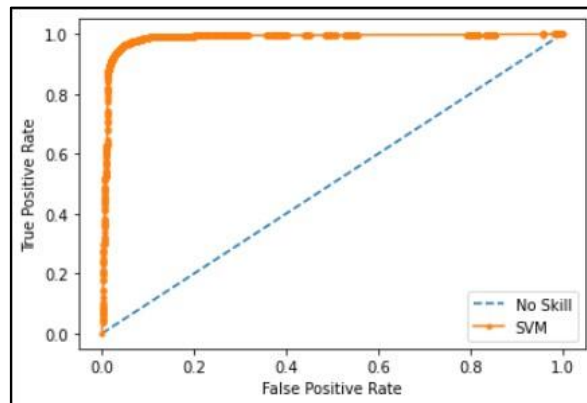


Figure 8.2 ROC-AUC Curve for SVM

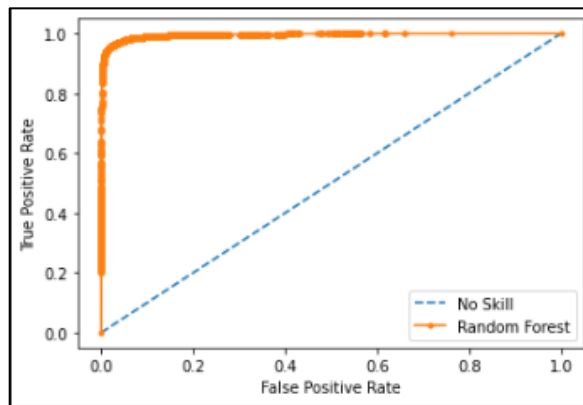


Figure 8.3 ROC-AUC Curve for Random Forest

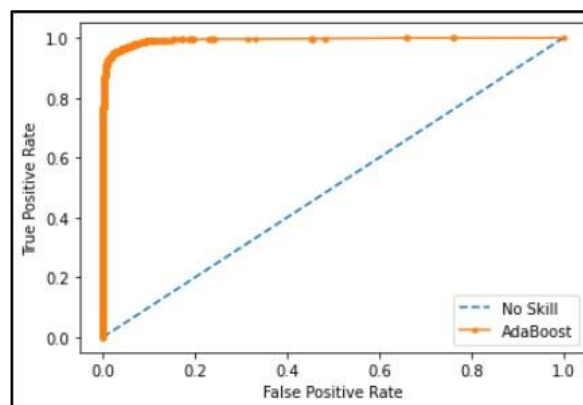


Figure 8.4 ROC-AUC Curve for AdaBoost

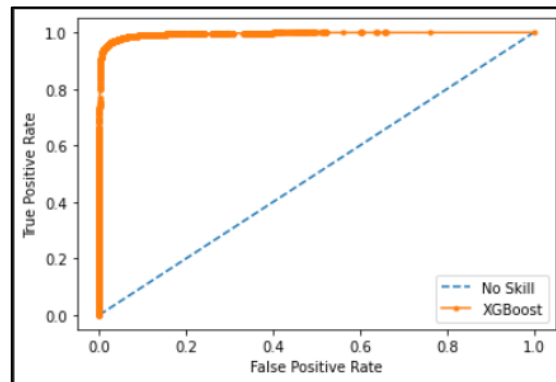


Figure 8.5 ROC-AUC Curve for XGBoost

Given below are the AUC score for all the ML models:

Table 8.1 AUC SCORE for Machine Learning Models

Machine Learning Model	AUC SCORE
Logistic Regression	0.989
Support Vector Machine	0.985
Random Forest	0.995
AdaBoost	0.994
XGBoost	0.995

The most ideal AUC is 1 and AUC close to this is observed in all the models.

8.1.2 CLASSIFICATION REPORT

8.1.2.1 PRECISION

It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Equation (8.1) Precision Formula

8.1.2.2 RECALL

It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Equation (8.2) Recall Formula

8.1.2.3 F1 SCORE

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances)

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

Equation (8.3) F1 Score Formula

8.1.2.4 ACCURACY

Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{\text{Number of Correct predictions}}{\text{Total number of predictions made}}$$

Equation (8.4) Accuracy Formula

8.1.3 CONFUSION MATRIX

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm.

- true positives: the cases in which we predicted yes and the actual output was also yes.
- true negatives: the cases in which we predicted no and the actual output was no.
- false positives: the cases in which we predicted yes and the actual output was no.
- false negatives: the cases in which we predicted no and the actual output was yes.

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Figure 8.6 Confusion Matrix

```
[[10321  894]
 [ 199  9781]]
```

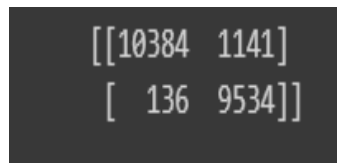
Figure 8.7 Confusion Matrix for Logistic Regression

Thus accuracy for logistic regression is calculated using the formula shown in Equation (8.4) and it is observed to be 94.56%

```
[[10132  447]
 [ 388 10228]]
```

Figure 8.8 Confusion Matrix for Random Forest

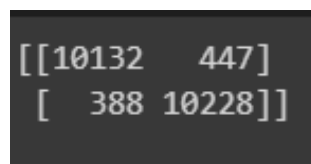
Thus accuracy for random forest is calculated using the formula shown in Equation (8.4) and it is observed to be 96.86%



[[10384	1141]
[136	9534]]

Figure 8.9 Confusion Matrix for SVM

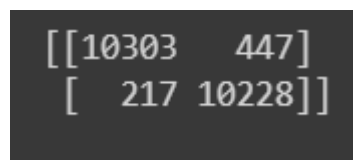
Thus accuracy for SVM is calculated using the formula shown in Equation (8.4) and it is observed to be 93.91%



[[10132	447]
[388	10228]]

Figure 8.10 Confusion Matrix for AdaBoost

Thus accuracy for AdaBoost is calculated using the formula shown in Equation (8.4) and it is observed to be 96.88%



[[10303	447]
[217	10228]]

Figure 8.11 Confusion Matrix for XGBoost

Thus accuracy for XGBoost is calculated using the formula shown in Equation (8.4) and it is observed to be 96.86%

8.2 RESULTS AND EVALUATION

The following figure shows various classification metrics for each of the machine learning models.

	Logistic Regression	Random Forest	Support Vector Machine	AdaBoost	XGBoost
Precision	94.72%	97.01%	94.40%	97.01%	97.04%
Recall	94.80%	96.90%	94.13%	96.98%	96.98%
F1 Score	94.50%	97.00%	94.00%	97.00%	97.00%
Accuracy	94.56%	96.86%	93.91%	96.88%	96.86%
AUC Score	98.9%	99.5%	98.5%	99.4%	99.5%

Figure 8.12 ML Model Performance Metrics

In Logistic Regression, the AUC score of 0.989 is obtained. Since this model considers each feature as an independent feature, results are seem to be good.

In SVM model the AUC score of 0.985 is obtained. The model is able to learn both the presence and missing edges with an accuracy of 96.86%. That might be because of the presence sufficient records of both present and missing edges and thus the model is able to learn significant differences out of them.

In case of Random Forest, Adaboost and XGBoost models almost same AUC score of 0.995 is obtained. This is because these models works on

boosting algorithm and it gives more importance to the weaker class. That is why these models are able to classify the presence and absence of an edge with the higher accuracy.

Thus very good accuracy for all the models are observed, with Random Forest, XGBoost and AdaBoost performing the best.

CHAPTER 9

CONCLUSION AND FUTURE WORKS

CONCLUSION

In this study, a way of detecting whether a link will be formed in the future in a social network graph is presented. Such a prediction has several applications like prediction of a disease outbreak, suggesting alternate route or recommendations on websites like Netflix/Amazon and so on.

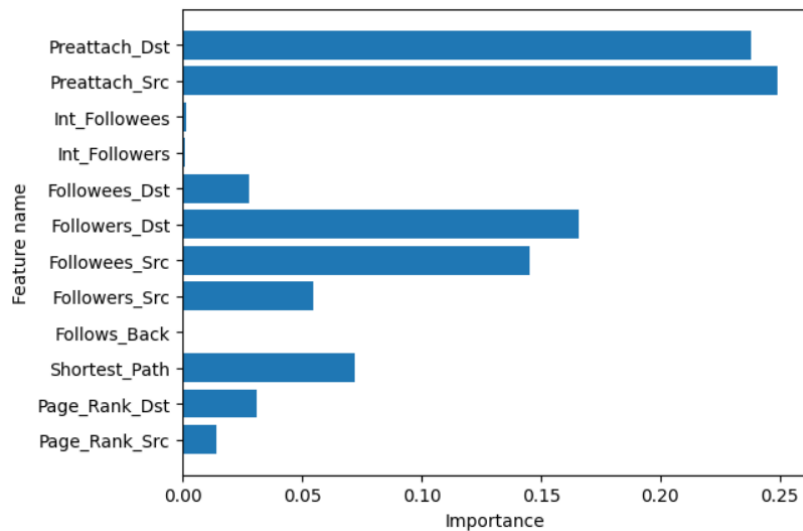


Figure 9.1 Feature importance

It is evident from the above visualization that not all features have high contribution in the prediction. The contribution can depend on the application of link prediction and it is necessary to explore more features and focus on the most relevant ones.

FUTURE SCOPE OF THE PROJECT

Link prediction can be used for various applications; here some typical applications are addressed.

- Recommendation in social networks
- Predict interactions between proteins
- Friends suggestion
- Predict criminal associations
- Marketing strategies

CHALLENGES AND FUTURE WORKS

Although numerous efforts have been made in link prediction, there are still many potential future challenges, and some new open problems require further study. Here some possible challenges on the link prediction problem are addressed.

- Disappearing link prediction
- Link prediction under dynamic nodes
- Overcoming imbalance
- Incorporating social theories
- Link prediction in heterogeneous social networks
- Fair evaluation and benchmark datasets

REFERENCES

1. D. Liben-Nowell and J. Kleinberg (May 2007), ‘The link-prediction problem for social networks’, vol. 58, no. 7, pp. 1019–1031.
2. W. P. X. B. W. Y. Z. XiaoYu (2015), ‘Link prediction in social networks : the state-of-the-art’, vol. 58, no. 1, pp. 1–38.
3. Q. Liu, S. Tang, X. Zhang, X. Zhao, B. Zhao, and H. Zheng (2016), ‘Network Growth and Link Prediction Through an Empirical Lens’, pp. 1–15.
4. P. Symeonidis, E. Tiakas, and Y. Manolopoulos (2010), ‘Transitive node similarity for link prediction in social networks with positive and negative links’, pp.183–190.
5. J.Leskovec, ‘Stanford Large Network Dataset Collection’.
<http://snap.stanford.edu/data/twitch-social-networks.html>
6. ‘Page Rank Algorithm and Implementation’ 2017.
<https://www.geeksforgeeks.org/page-rank-algorithm-implementation/>
7. <https://towardsdatascience.com/4-ways-to-visualize-individual-decision-trees-in-a-random-forest-7a9beda1d1b7>
8. <https://www.codementor.io/@mgalarny/visualizing-decision-trees-with-python-scikit-learn-graphviz-matplotlib-154mszcto7>
9. <https://www.techtarget.com/whatis/definition/six-degrees-of-separation>
10. <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-adaboost-algorithm-with-python-implementation/>
11. <http://be.amazd.com/link-prediction/>

12. Ece C. Mutlu, Toktam Oghaz, Amirarsalan Rajabi and Ivan Garibay
'Review on Learning and Extracting Graph Features for Link Prediction'
13. Daniel Mesafint Belete, Manjaiah D H 'Grid search in hyperparameter optimization of machine learning models for prediction'
14. <https://networkx.org/documentation/networkx-2.4/>
15. <https://xgboost.readthedocs.io/en/stable/>
16. <https://neo4j.com/docs/graph-data-science/current/algorithms/linkprediction/#:~:text=Link%20prediction%20algorithms%20help%20determine,predictions%20about%20relationships%20between%20nodes.>
17. <https://vgnshiyer.medium.com/link-prediction-in-a-social-network-df230c3d85e6>
18. https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.link_prediction.preferential_attachment.html