# Link Prediction in Social Networks

Atul Kumar

Supervised by Dr. Zamilur Rahman

Algoma University

Sault Ste. Marie, ON

## Contents

## 1 Introduction

Today, Social network has become an integral part of our daily lives, as it provides a platform to connect and collaborate with each other. More than half of the population on Earth uses social networks. Due to this rapid growth of social networks, link prediction has emerged as an important research topic in the field of network analysis.

Link prediction is the art of predicting the links between the nodes. This identifies patterns in the existing network and computes the likelihood of a future connection between two nodes. It involves calculating how similar the two nodes are, and based on that predicts the likelihood of a link. This link prediction has several other useful applications in the context of real world apart from just predicting the links between the two nodes, which we will see in the upcoming sections.

In this paper, our aim is to analyze the structure of social networks and compute the different kinds of scores between any nodes in the network. We'll begin by discussing some background of the problem statement. Then we will provide an overview of the methods that have been considered for calculating the similarity between any nodes in the social network. Once computed, we will then focus on the implementation of the Machine Learning models, where we will be training and testing our different models on the computed similarity scores. Finally, we will then analyze how effective our model is to predict future links in the social network by calculating several performance measures.

## 2 Background

Social networks can be represented as graphs, where users are assumed to be a node and a link/connection between the two users is as an edge. Considering the social networks as a graph will not only help visualize the patterns in the structure but can also apply the concepts from the graph theory that can help in visualizations of further common patterns in the graph. To name a few, some of those concepts are the degree of nodes, edge centrality, shortest path between two nodes, etc.

By now, we know that these social networks are mapped as a graph. It would be worthwhile to take a quick look at the graph in order to get a better understanding of the rest of the stuff.

A Graph G consists of two sets: a set V of vertices, or we can simply say nodes and a set E of edges that connect the vertices. Let's have a look a the graph:

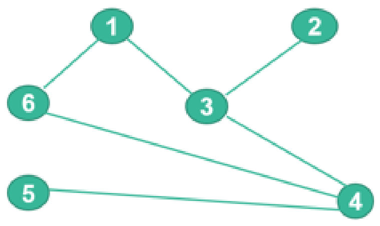


Figure 1: A graph with 6 vertices and 6 edges

As we can see, there are 6 vertices in the graph and 6 edges that connect those vertices together. It can be represented as:

$$V = \{1, 2, 3, 4, 5, 6\}$$
$$E = \{(1, 3), (1, 6), (2, 3), (3, 4), (4, 5), (4, 6)\}$$

It is worth noting that the edges in the above graph are represented in both directions, meaning the connection between any two nodes is bidirectional instead of unidirectional. On the social network, there are two types of connections between the users: a followers-Followees connection or a Mutual connection. For example, on LinkedIn, we can have two types of connections:

1. Two people can connect to each other, which represents a bi-directional link.
2. One can follow another user without the need for the followed user to follow them back, which represents a unidirectional link.

Now, let's take a look at the visual representation of this undirected and directed graph. Here, Figure 2 represents the undirected graph and Figure 3 represents the directed graph:

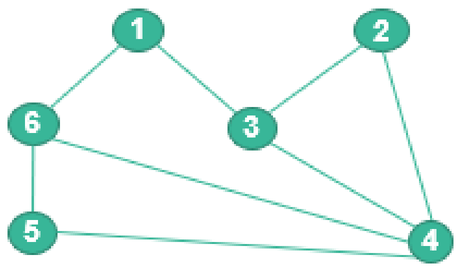


Figure 2: Undirected Graph

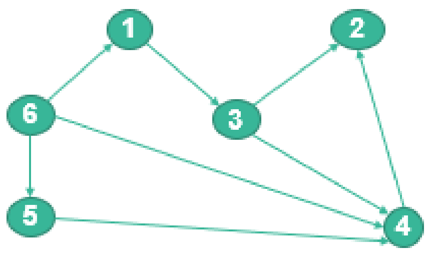


Figure 3: Directed Graph

In Figure 2 (i.e. Undirected Graph), all the connections/links between the users are in both directions, which simply means both are connected to each other. For example, user "1" is connected to user "3" and user "3" is also connected to user "1". However, in the case of Figure 3 (i.e. Directed Graph), we can see all of the links have a specific direction. For example, there exists a connection from user "1" to user "3", but not in opposite direction, so user "1" follows user "3" but user "3" doesn't follow user "1". Just a worth taking note, if there are edges in a directed graph, that go in either direction, then that graph can be treated as an undirected graph.

By now, we have an understanding of the graphs and how the social network can be mapped as graphs. For the rest of the sections, we will be using the terms graphs and social networks interchangeably. Now, let's see what the link prediction is and why is there a need to predict the link between the nodes.

As said in the introduction, link prediction involves predicting the likelihood of future connections between any two nodes in the graph. For calculating the likelihood of future connections, we calculate several similarity measures (which will be discussed in the next section), and depending on that we can conclude whether there is the possibility of an edge or not, between given two nodes, in the future. These similarity measures are solely calculated on the basis of the current network structure. In other words, the connections a user has currently can heavily influence their future connections on a social network. Let's have a look at the following graphs:

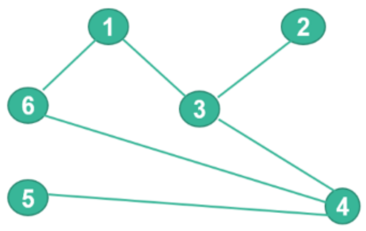


Figure 4: At time $t$

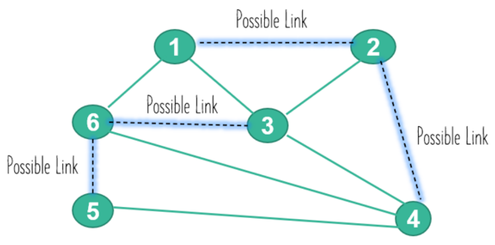


Figure 5: At time $t + n$

Above two graphs are showing a concise and informative representation of what actually the link prediction all about is. Let's assume, at time $t$, we have 6 users (represented by 6 nodes in the graph) and 6 connections(represented by edges) in the network shown in Figure 4. So, there's a connection between the following users at time $t$:

$$\{(1,3),(1,6),(2,3),(3,4),(4,5),(4,6)\}$$

Let's say at time $t$, we calculate those similarity measures and identified the future links as shown in Figure 5. So, this is representing the links that may appear at time $t + n$ in the given social network. Following are the possible future links:

$$\{(1,2),(5,6),(2,4)\}$$

Now knowing link prediction, we can imagine how important this is in the field of network analysis, as it helps to identify the potential connections in the network. This has several important applications in various fields. Following are some of them:

- Recommendation in Social Network

- Network Completion Problem
- Finding Experts and Identifying Collaboration Opportunities in Academic Social Networks
- Community Detection
- Anomaly Detection
- Event Detection

We will discuss that all in brief in one of the next sections. But, for now, let's focus on the methods that can be used for identifying the links in the social network.

## 3 Link Prediction Methods

According to N.N. Daud et Al [1], we can classify the methods of link prediction into four broad categories (as shown in Figure 6), These are as follows:

1. **Similarity:** Similarity method is one of the popular approaches in link prediction where we calculate how similar the two nodes are and determine the likelihood of a connection between them. Greater the similarity score of two nodes, the higher the chances of forming a link between nodes. For this, we calculate several similarity metrics such as Common Neighbor, Jaccard Coefficient, etc.

2. **Probabilistic:** It involves building a model that is based on the characteristics of the network structure, which aims to identify the underlying pattern of link formation in the given network. Then we calculate the probability value of each pair of nodes that is unconnected. Based on that probability value, we calculate the likelihood of a connection between two nodes. The greater the probability value of two nodes, the higher the chances of forming a link between nodes

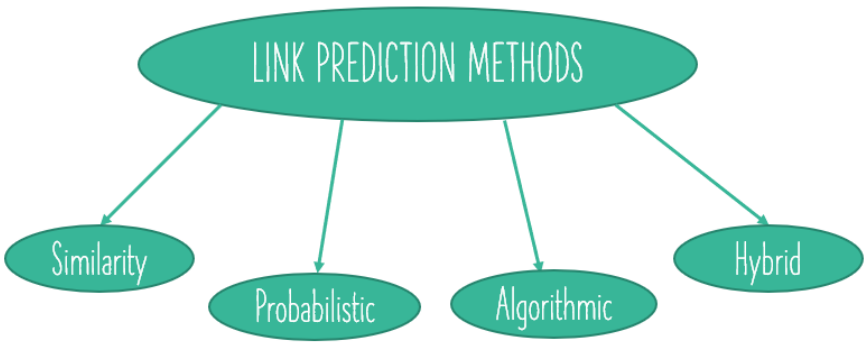


Figure 6: Link Prediction Methods

3. **Algorithmic:** This approach is widely used in predicting the links between the nodes. It involves analyzing the structure of the network and predicting the link based on the set of rules and procedures to identify the patterns in the network. One of the algorithmic methods used for predicting the links is using machine learning techniques.

4. **Hybrid:** Method is said to be hybrid if we combine two or more of the previously mentioned methods for enhancing the accuracy, robustness, etcetera.

Now, we have discussed all the methods that can be used for predicting the links between the nodes in the given social network. Now let's see the methods that we have used for purpose of this project. We have implemented a hybrid approach for predicting the links where we are combining the Similarity approach with the Algorithmic approach, in order to find fair accuracy scores. Let's take a deep dive into both of these approaches and the metrics that we have used for calculating scores.

## 3.1 Similarity

As discussed, the similarity approach involves calculating the similarity scores between the given two nodes based on several similarity metrics. These scores tell how similar those two nodes are and depending on the score, it predicts whether there should be a link exists between them or not. Similarity metrics can be divided into the following 3 broad categories [1]:

### 3.1.1 Local Indices

As the name says, these indices are local to the nodes and only take the immediate neighbors into account to find the similarity scores. That means, it requires minimal knowledge of the network structure to find the similarity, which also involves less computation since it just needs to look locally for each node in the network. It predicts the likelihood of a connection between two nodes based on the number of common connections that the given two nodes have in the current network. Following are the local similarity scores that have been considered for the purpose of this project (also shown in Figure 7):

1. **Jaccard Coefficient:** It calculates the number of common neighbours between two nodes with respect to the neighbours they both have in total.

$$JC(u,v) = \frac{\mid \Gamma(u) \cap \Gamma(v) \mid}{\mid \Gamma(u) \cup \Gamma(v) \mid}$$

2. **Common Neighbor:** It calculates the number of common neighbours between two nodes.

$$CN(u,v) = \mid \Gamma(u) \cap \Gamma(v) \mid$$

3. **Preferential Attachment Score [2] :** The node will be having a high chance to connect to that node which will be having high degree compared to nodes having a lower degree.

$$PA(u,v) = \mid \Gamma(u) \cdot \Gamma(v) \mid$$

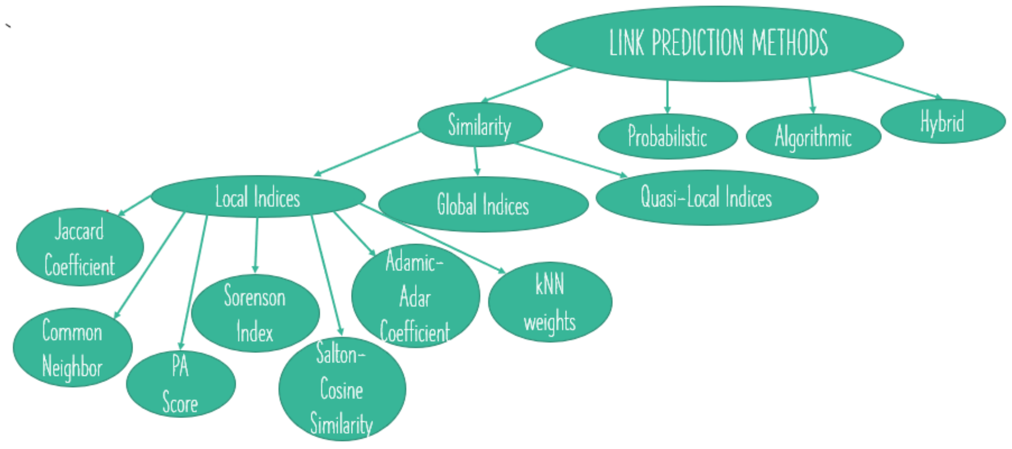


Figure 7: Local Indices Similarity Metrics

4. **Sorenson Index** [2] **:** It is used to compute the similarity score based on how the neighbouring nodes of u and v are overlapping.

$$SI(u,v) = \frac{\mid \Gamma(u) \cap \Gamma(v) \mid}{\mid \Gamma(u) + \Gamma(v) \mid}$$

5. **Salton-Cosine Similarity** [2] **:** It's a cosine metric (using the cosine of the angle between u and v) that is used to measure the similarity between two nodes.

$$SC(u,v) = \frac{\mid \Gamma(u) \cap \Gamma(v) \mid}{\sqrt{\mid \Gamma(u) \cdot \Gamma(v) \mid}}$$

6. **Adamic-Adar Coefficient** [3] **:** High score will be assigned to the node whose common neighbours will be having fewer degrees, as it will be shared by fewer nodes in the network and will be more specific in that context.

$$AA(u,v) = \Sigma_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |\Gamma(z)|}$$

7. **kNN weights** [4] **:** It shows that how important the k-closest neighbors are to

the current node for predicting the connection between itself and the new node.

$$kNNW_1(u,v) = w(u) + w(v)$$

$$kNNW_2(u,v) = w(u) \cdot w(v)$$

where,

$$w(v) = \frac{1}{\sqrt{1+ \mid \Gamma(v) \mid}}$$

### 3.1.2 Global Indices

One of the major drawbacks of local indices is that it just considers the immediate neighbourhood of the nodes. However, it solely may not be effective since it never gets to know how's the network structured beyond its immediate neighbors which also plays a major role in predicting the links. So, here global indices come in help. Unlike local indices which are just based on the common connections with the immediate neighbors, global indices consider the whole structure of the current network and calculate the similarity scores with respect to the whole structure, not just the immediate neighbors. Following are the global similarity scores that have been considered for the purpose of this project (also shown in Figure 8):

1. **Katz Index** [2] **:** This metric calculates the number of all the paths from node u to v and gives more weight to the path that has a shorter length relative to other paths from u to v.

$$Katz(u,v) = \sum_{l=1}^{\infty} \beta^l \cdot |path^l_{u,v}| = \beta A + \beta^2 A^2 + \beta^3 A^3 + \dots$$

where, $l$ is the path length

$\beta$ is damping factor

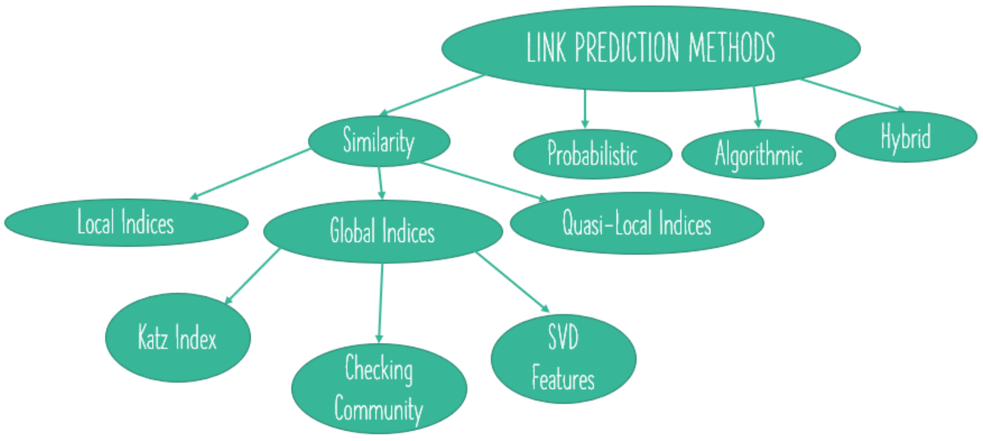


Figure 8: Global Indices Similarity Metrics

$|path^{l}_{u,v}|$ is the set of all the paths from $u$ to $v$

2. **Checking Community of Nodes:**

   - We are using the "greedy_modularity_communities" function from the NetworkX library [5] for detecting the communities in the social network.
   - Depending on how dense the network is, it will group the nodes together that will be from the same community.
   - Higher the modularity score (objective function to optimize the partition of nodes into communities), the higher the chances of nodes belonging to the same community.

3. **Singular Value Decomposition Features (SVD):**

   - We are using the "svds" function from the scipy library for representing the graph in a lower dimension.
   - It involves calculating the feature vector for each node.
   - Then, comparing these feature vectors of two nodes to measure the node similarity index.

### 3.1.3 Quasi-local Indices

As we saw in the global indices, we identify the patterns for each node in the network and recommend the potential links accordingly. However, identifying patterns in the network for each node can be time and space inefficient as the size of the network grows. So, we need some other approach that should work better than the local indices but not be as inefficient as global indices. Here comes the role of quasi-local indices.

Quasi-local indices use to have additional information likewise the global indices but compute the score on the basis of the nodes that are its immediate neighbors. So, the computational cost of calculating the similarity scores using the quasi-local Indices approach is not as high as that of global indices.

For purpose of this project, it involves calculating random-walk metrics which determine the importance of each node in the network. The idea behind the random walk is basically we start at an arbitrary node, and traverse the existing links (or edges) for $n$ number of iterations from that node, where $n$ is considered as a hyperparameter. On the basis of the nodes and edges it visits and traverses, it calculates the importance of the nodes. Here we are calculating two random-walk metrics (also shown in Figure 9):

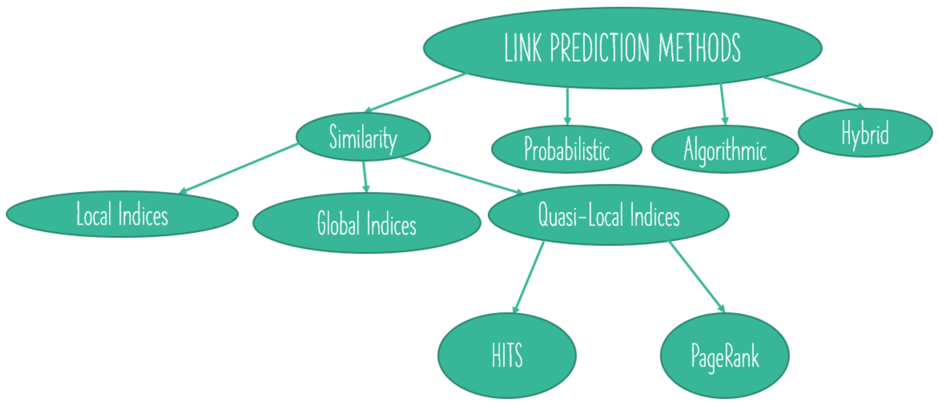


Figure 9: Quasi-local Indices Similarity Metrics

1. **Hyperlink-Induced Topic Search (HITS):** It calculates the importance or relevance of the nodes in the graph by looking at their connections. Some nodes in the graph can be more "important" than others as they may have connections that are more "important". The "important" term means here how significant or relevant the node is in the network. For example, the node occupying the central position (or which has a large number of connections) can be considered important in some contexts or some other metric can be used to represent the importance of the node. It requires the calculation of the following two different scores:
   (a) **Hub Score:** It determines the score of node importance by counting the number of nodes that it is linked to.
   (b) **Authority Score:** It determines the score of node importance by counting the number of nodes that are linked to it.
2. **PageRank:** As the name says, it involves ranking the nodes (or pages, if we consider a web page as a node) by their importance in the network (Note: again, the "important" term means the same as it means in HITS). It starts at some arbitrary node and keeps following the links (or edges) connected to the current node for $n$ number of iterations. We may arise the question that there can be several links attached to one particular node, so how would it decide which link to follow? Well, the chances of selecting the particular link will be dependent on how many links are there that are connected to the current node and also which one of them would be more important (nodes with a large number of connections tend to have higher importance). Finally, it will rank the pages, by assigning the scores, depending on how often the random walk algorithm visits the particular node.

Both of the above-given random-walk metrics are being computed using the NetworkX library.

## 3.2 Algorithmic

As discussed at the beginning of this section, the Algorithmic approach is widely used in predicting the links in the current network. As we know, while writing an algorithm, we define a set of rules and procedures to handle all the possible cases. Similarly, writing conventional algorithms for identifying patterns, relationships, etc. in the social network requires the same task. The algorithmic method that we have used here for predicting the links is using Machine Learning algorithms.

Let's take a look at the very general representation of the Machine Learning approach in Figure 10,

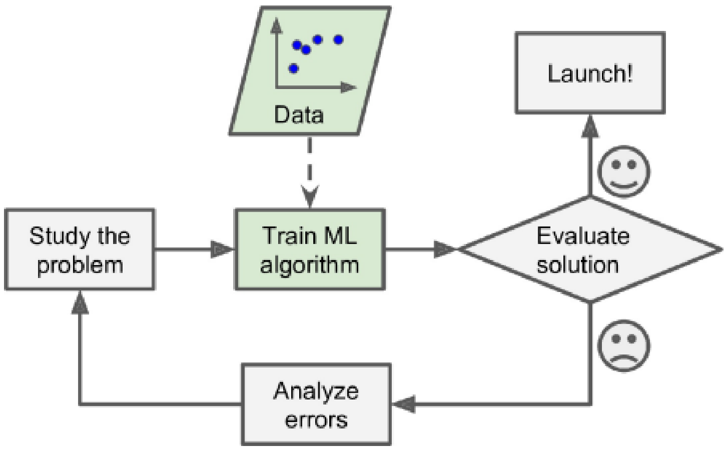


Figure 10: The Machine Learning approach [6]

It basically starts by studying the problem, where we analyze the problem statement. Once we have understood the goal of our problem, then we can start analyzing the data. Before feeding the data to the model, however, we preprocess the data as needed and make it suitable for ML models. After then, we train our model on that preprocessed data, which helps identify patterns, correlations, etc in the data. Once trained, we can evaluate our solution and test our data on the test set or analyze the errors, if they are any. Some of the key points, about the ML model, which is worth mentioning are as follows:

- Machine Learning approach is considered to be one of the primary approaches employed in conducting link prediction analysis.
- Since we are using the hybrid approach we can use those calculated similarity scores to train our model.
- The models will employ a supervised learning approach for a classification task.

Now, let's see the models that have been implemented for the purpose of this project.

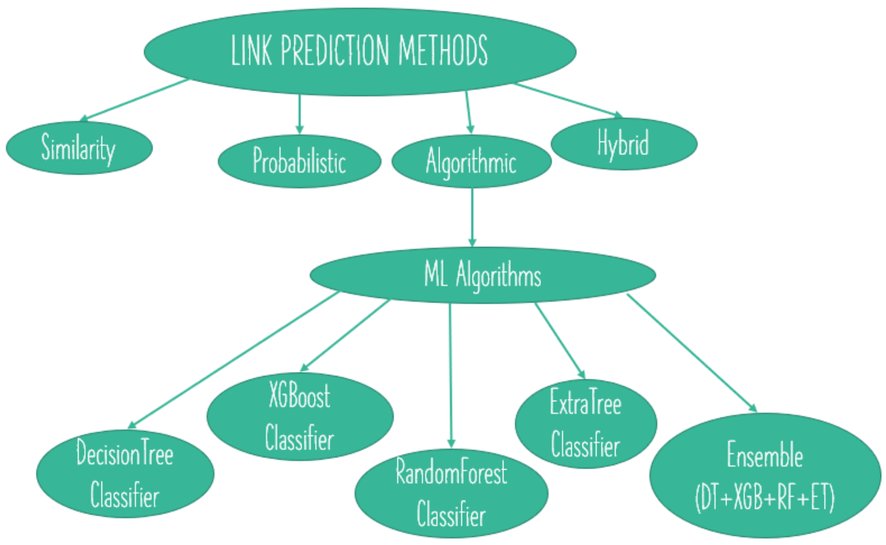


Figure 11: Machine Learning Algorithms

As we can see from Figure 11, we have implemented 5 different algorithms for predicting the links in the given dataset. Now, we will be taking a close look at each of them:

1. **DecisionTree Classifier:** This algorithm splits the data into subsets based on the input features. After generating the subsets, it selects the node which best splits the data with respect to the target variable and assigns it as a root node

and all its values become the branches of the root node. In the next step, again we choose the feature from the rest of the features that best splits the data and make its value as the branches of that node and we keep repeating the same step again and again until we use all of its features and build a tree. Once we have a tree, then we can make predictions by traversing the edges of the tree while taking a decision at each node.

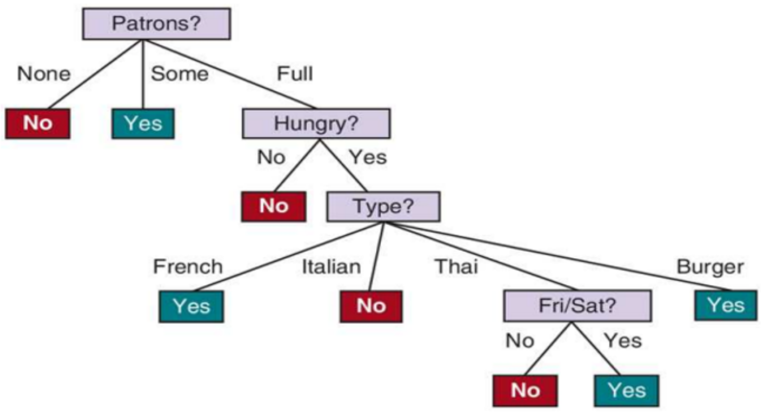
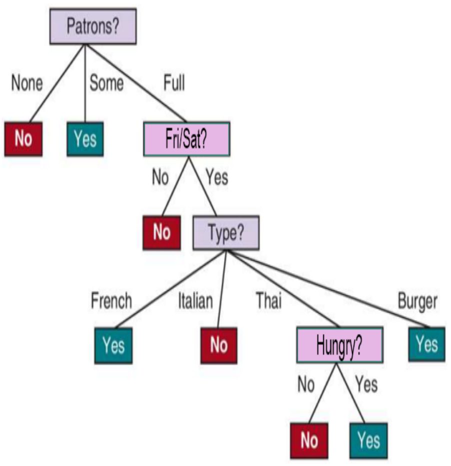


Figure 12: Decision Tree [7]

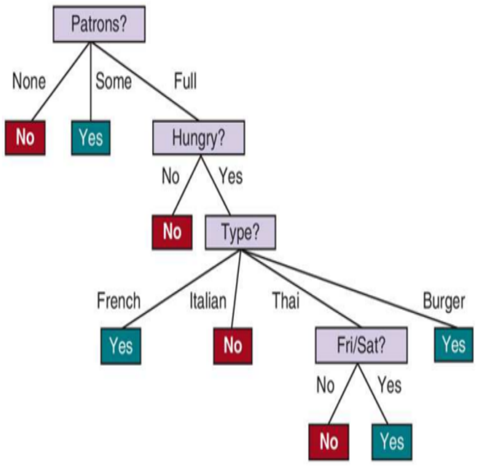
Figure 12 represents a DecisionTree where the nodes such as Patrons, Hungry, Type, and Fri/Sat are the input features for any arbitrary dataset and branches represent the values for the particular features. For example, “Patrons” has values None, Some, and Full. After making a decision at each node, it will take that edge, which has the same value which it decided and keep traversing the tree. Once it reaches the leaf node, it will show the output value.

2. **XGBoost Classifier:** It is also known as ExtremeGradientBoosting Classifier. This classifier also builds the tree as we have seen in the DecisionTree Classifier. But it involves building multiple decision trees. At each iteration, it builds a new tree where each of them aims to correct the errors made by the previous tree while

making the decision. For example, let's say at iteration 1, the classifier makes the wrong decision and picks the trivial feature and split the data from there. So, the next iteration will make sure, this error should be corrected. However, if it goes unseen in the next iteration as well, then it will be corrected in the $3^{rd}$ iteration and so on.



(a) Iteration: 1



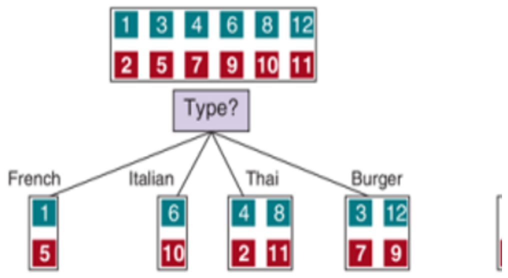
(b) Iteration: 2

Figure 13: Iterations [7]

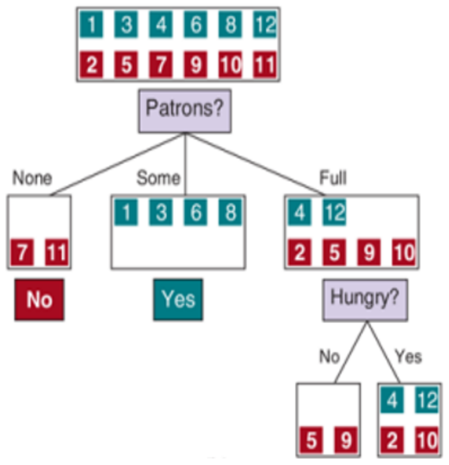
Let's suppose, we already know that the "Hungry?" feature helps in better splitting the data as compared to the "Fri/Sat?" feature in Figure 13 So now, As we can see in Figure 13a i.e. iteration 1, it picks up the "Fri/Sat?" feature first and "Hungry?" at the end which may not provide good accuracy at the end (because "Hungry?" is an important attribute as compared to "Fri/Sat?"), so it will go to the next iteration i.e. Figure 13b and will be correcting its decision that whatever it made in the previous iteration.

3. **RandomForest Classifier**: As the name says, "Forest", RandomForest is an ensemble learning technique which involves building multiple decision trees (making

a “Forest”), where each tree builds independently (i.e. without bearing any effect from other trees in the forest) using a subset of features that is randomly selected for each tree from the pool of given input features. For each of the trees, it selects the node which best splits the data, for that specific tree, concerning the target variable. Then, after getting the prediction from each of the trees, it aggregates the result and makes a final prediction.



(a) Tree: 1



(b) Tree: 2

Figure 14: Trees [7]

As we can see, the two decision trees that are independent of each other and a subset of features is randomly selected for each of the trees. For example, Figure 14a has assigned a “Type?” attribute, so the best split obviously is the “Type?” attribute. However, in Figure 14b, “Patrons?” and “Hungry?” attributes are assigned, from which “Patrons?” would be the best split as compared to the “Hungry?”.

4. **ExtraTree Classifier:** It is also an ensemble learning technique which is similar to the RandomForest Classifier, but just has a difference in the way the nodes are

selected for splitting the data for each particular tree in the forest.

Likewise RandomForest Classifier, it also involves building multiple decision trees (making a “Forest”), where each tree builds independently (i.e. without bearing any effect from other trees in the forest) using a subset of features that is randomly selected for each tree from the pool of given input features. However, for each of the trees, instead of searching for the feature that best splits the data, it selects the random feature and splits the data according to that.

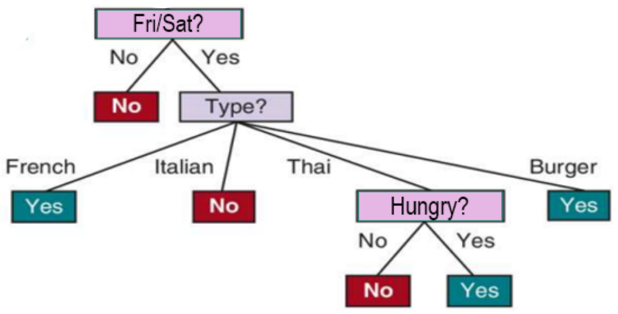


Figure 15: Tree [7]

As we have seen in the XGBoost Classifier, where we were splitting the data on the basis of “Fri/Sat?” attribute, we were going to the next iteration and correcting the output produced by iteration 1. However, after randomly splitting the features among the trees in ExtraTree Classifier, if we have one of the trees like Figure 15, then it will be considered a valid tree since it doesn’t search for the feature that best split, instead just split it randomly.

5. **Ensemble Classifier:** This model combines multiple individual models to make a more robust and stable model. In this, we combined all the above-given models with the aim of improving performance and accuracy.

### 3.3 Graphical Representation of our Project

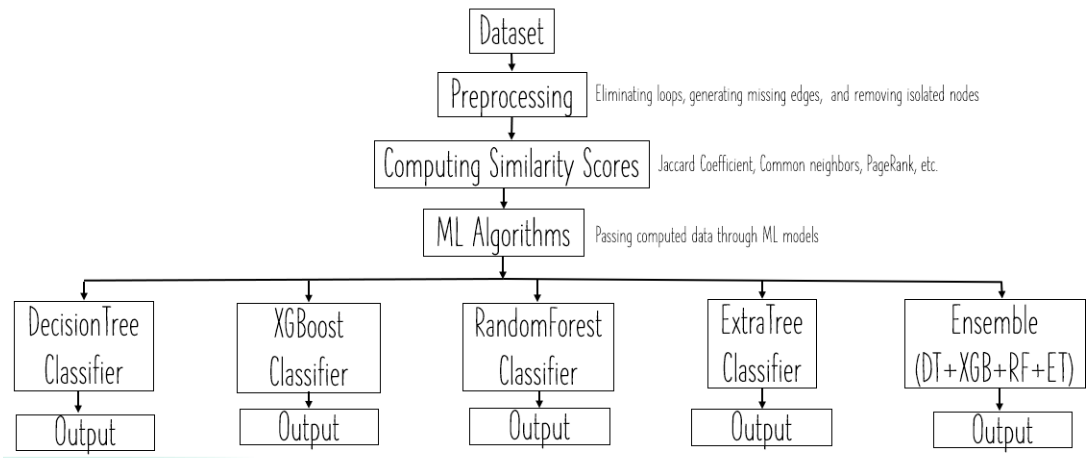

Dataset
Preprocessing
Eliminating loops, generating missing edges, and removing isolated nodes
Computing Similarity Scores
Jaccard Coefficient, Common neighbors, PageRank, etc.
ML Algorithms
Passing computed data through ML models
DecisionTree Classifier
XGBoost Classifier
RandomForest Classifier
ExtraTree Classifier
Ensemble (DT+XGB+RF+ET)
Output
Output
Output
Output
Output

Figure 16: Graphical Representation of our Project

As the saying goes, a picture is worth a thousand words. In this report, we have included a graphical representation of our approach, which help visually illustrate our idea within seconds.

The workflow of the experimentation can be summarized as follows. We are first bringing the dataset and preprocessing it (which will be discussed in the next section). Followed by the calculation of the similarity scores on the preprocessed data, as described in the earlier part of this section. After computing everything, we feed the same training data to all the ML models and evaluate their performance on the same test (unseen) data. Finally, we compare the performance measures (or output) of all the models to draw meaningful conclusions.

## 4 Datasets

We have used the following three datasets [8] for testing our approach:

1. Epinions (Directed Social Network): It's a who-trust-whom online social network of a consumer review site, where members of the site can decide whether to "trust"

each other.

- It has 75,879 nodes and 508,837 edges

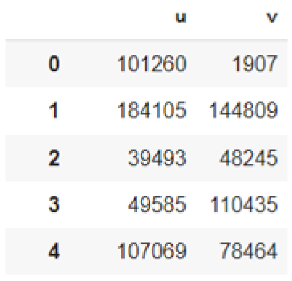
2. Facebook (Directed Social Network): The dataset consists of friend lists from Facebook. It was collected from survey participants using the Facebook app.

   - It has 75,879 nodes and 508,837 edges

3. Facebook (Undirected Social Network): The dataset consists of friend lists from Facebook. It was collected from survey participants using the Facebook app.

   - It has 3,959 nodes and 168,486 edges

## 4.1 Dataset Preprocessing



| | u | v |
|---|---|---|
| 0 | 101260 | 1907 |
| 1 | 184105 | 144809 |
| 2 | 39493 | 48245 |
| 3 | 49585 | 110435 |
| 4 | 107069 | 78464 |

Figure 17: Dataset Snippet

Figure 17 represents the snippet that how the data looks like when we import it first. Then we pass these datasets through the preprocessing steps, which are mentioned as follows:

1. Import the dataset from the file

2. Add a column to the imported data for marking the edge status ("1" for "present" and "0" for "absent"). Since, the edges that we are reading from the file are said to be existed in between the nodes, the edge status for all of them would be "1".

3. Eliminate the loops, if there are any. Since we can't be our own connections in the social network.

4. For the Binary class classification task, we must have another class to classify/predict to which the particular instance belongs. So, for that, we are generating the missing edges randomly. The idea behind generating the missing edges is if we find two nodes that are not connected by a direct edge and have no mutual connection, then we will consider that as a missing edge and mark the status of the edge, between those two nodes, to "0".

5. We are generating the same number of "missing" edges as that of "present" edges. However, the approach to generating the same number of missing edges is a bit different for each directed and undirected graph.

   - For a directed graph, we are running the for-loop the same number of times as that of a total number of present edges to generate a same number of missing edges.
   - For the undirected graph, we are running the for-loop half times of the total number of present edges to generate 50% of required missing edges. And, for the rest 50%, we switch the source node and destination node of the first 50% missing edges to make the links bi-directional.

6. After that we will split the dataset into two parts: training set (80% of whole dataset) and test set (20% of whole dataset), and make a separate graph of the training set using the NetworkX library.

7. Finally, we will calculate all those similarity functions for each and every instance in both the training and test set, which is also a bit different for each of the directed and undirected graphs.

- For a directed graph, we must take care of the edge direction. And for computing the similarity scores, we will be considering the out-degree and in-degree similarity scores, of each and every coefficient, separately.
- For an undirected graph, however, we can simply consider all the immediate neighbours (that are directly connected) of the node as a connection.

Note: We will be calculating those similarity scores, for both the training and test set, using the graph that we made in step-5 because it still contains 80% of the graph structure.

## 5 Experimental Results

This model approach was tested on three different datasets listed at the beginning of the previous section. Following are the experimental results for each of the datasets:

1. Epinions (Directed Social Network):

Table 1: Epinions (Directed Social Network)

| **Classifiers** | **EPINIONS (Directed Social Network): 75,879 nodes and 508,837 edges** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **TRAINING SET RESULTS** | | | | **TEST SET RESULTS** | | | |
| | **Accuracy** | **Precision** | **Recall** | **F1_Score** | **Accuracy** | **Precision** | **Recall** | **F1_Score** |
| DecisionTree Classifier | 94.83% | 99.95% | 99.90% | 99.93% | 89.02% | 94.16% | 83.20% | 88.34% |
| XGBoost Classifier | 96.48% | 96.16% | 96.96% | 96.56% | 88.92% | 95.42% | 81.77% | 88.07% |
| RandomForest Classifier | 96.21% | 99.85% | 99.97% | 99.91% | 90.33% | 95.73% | 84.42% | 89.72% |
| ExtraTree Classifier | 95.76% | 99.96% | 99.98% | 99.97% | 90.58% | 95.31% | 85.35% | 90.10% |
| Ensemble (XGB+RF+DT+ET) | 96.29% | 99.96% | 99.96% | 99.96% | 90.29% | 96.32% | 83.78% | 89.61% |

2. Facebook (Directed Social Network):

Table 2: Facebook (Directed Social Network)

| **Classifiers** | **Facebook (Directed Social Network): 4,039 nodes and 88,234 edges** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **TRAINING SET RESULTS** | | | | **TEST SET RESULTS** | | | |
| | **Accuracy** | **Precision** | **Recall** | **F1_Score** | **Accuracy** | **Precision** | **Recall** | **F1_Score** |
| DecisionTree Classifier | 96.07% | 100.00% | 99.99% | 100.00% | 95.94% | 95.91% | 95.98% | 95.94% |
| XGBoost Classifier | 97.50% | 98.25% | 97.47% | 97.86% | 97.37% | 97.90% | 96.82% | 97.36% |
| RandomForest Classifier | 97.21% | 100.00% | 99.99% | 99.99% | 97.32% | 97.69% | 96.93% | 97.31% |
| ExtraTree Classifier | 97.11% | 100.00% | 99.99% | 100.00% | 97.24% | 97.44% | 97.02% | 97.23% |
| Ensemble (XGB+RF+DT+ET) | 97.26% | 100.00% | 99.99% | 99.99% | 97.35% | 98.02% | 96.66% | 97.34% |

3. Facebook (Undirected Social Network):

Table 3: Facebook (Undirected Social Network)

| **Classifiers** | **Facebook (Undirected Social Network): 3,959 nodes and 168,486 edges** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **TRAINING SET RESULTS** | | | | **TEST SET RESULTS** | | | |
| | **Accuracy** | **Precision** | **Recall** | **F1_Score** | **Accuracy** | **Precision** | **Recall** | **F1_Score** |
| DecisionTree Classifier | 99.71% | 100.00% | 100.00% | 100.00% | 99.75% | 99.74% | 99.76% | 99.75% |
| XGBoost Classifier | 99.84% | 100.00% | 99.71% | 99.86% | 99.87% | 100.00% | 99.74% | 99.87% |
| RandomForest Classifier | 99.84% | 100.00% | 99.77% | 99.89% | 99.87% | 100.00% | 99.74% | 99.87% |
| ExtraTree Classifier | 99.83% | 100.00% | 100.00% | 100.00% | 99.87% | 100.00% | 99.74% | 99.87% |
| Ensemble (XGB+RF+DT+ET) | 99.84% | 100.00% | 99.78% | 99.89% | 99.87% | 100.00% | 99.74% | 99.87% |

## 6 Conclusions and Future Work

- All models perform exceptionally well on all datasets while training, achieving accuracies of 95% or higher.
- The models achieved close to 90% accuracy for the Epinions test set and above 95% accuracy on the test sets for the other two datasets, with one dataset reaching an exceptional 99% accuracy.
- All models demonstrate high precision rates approaching 100% across multiple datasets. However, the Epinions dataset shows lowest recall rate (close to 85%) among three datasets.

## References

[1] Nur Nasuha Daud, Siti Hafizah Ab Hamid, Muntadher Saadoon, Firdaus Sahran, and Nor Badrul Anuar. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications*, 166:102716, 2020.

[2] Wang Peng, Xu BaoWen, Wu YuRong, and Zhou XiaoYu. Link prediction in social networks: the state-of-the-art, 2014.

[3] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[4] Dima Kagan, Yuval Elovici, and Michael Fire. Generic anomalous vertices detection utilizing a link prediction algorithm, 2017.

[5] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[6] Aurélien Géron. *Hands-On Machine Learning witht Scikit-Learn, Keras, and TensorFlow.* O'Reilly, Canada, 2019.

[7] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach.* Pearson, USA, 2021.

[8] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.