


Article

A Comparative Study between Graph Database and Traditional Approach to forecast Coauthor Link Prediction based on Machine Learning Models

Mohammad Rezwanul Huq^{1,†} , Sanjeda Sara Jennifer^{2,†}, Shafiul Mahmud Partho^{3,†}, Fariha Fairuz^{4,†}

¹ Department of Computer Science and Engineering, East West University; mrhuq@ewubd.edu

² Department of Computer Science and Engineering, East West University; 2018-1-60-061@std.ewubd.edu

³ Department of Computer Science and Engineering, East West University; 2018-1-60-219@std.ewubd.edu

⁴ Department of Computer Science and Engineering, East West University; 2018-1-60-080@std.ewubd.edu

* Correspondence: mrhuq@ewubd.edu; Tel.: +880-1729-648-344

† These authors contributed equally to this work.

Abstract: In the modern world where research is taking a huge leap, the collaboration network between authors is also expanding, increasing the probability of different authors coming together to work on the same project, same research paper making them co-authors. In coauthorship, link prediction is used to anticipate new interactions between its members that are likely to occur in the future. Researchers have concentrated their efforts on studying and suggesting methods for providing effective reviews for authors who can collaborate on a scientific endeavor. In order to provide a precise link prediction, a graph database approach is proposed in this paper using nodes to determine most possible co-authors in future. In order to forecast the connections, we preprocessed the data set for the maximum relative contents. A supervised learning approach is used to execute the solution, which includes random forest classifier and logistic regression. The first findings of our technique reveal that the total of two author node's research collaboration indices has the greatest influence on the performance of supervised link prediction than that of the traditional approach, which stimulates us to conduct further study on employing such a forecast.

Keywords: coauthorship; coauthorship Network; Link Prediction; Graph Database; Nodes

1. Introduction

Coauthorship Network [1] is a type of social network of researchers collaborating with each other and cooperatively presenting their research outcome on a topic. Authors are expressed by nodes in the networks. The issue of anticipating the presence of a relationship between authors (nodes) in such a network is known as link prediction. Link prediction in the coauthorship network has been a prominent field of study. It is better to construct a similarity measure between two omnipresent authors and then use it to find the most potential coauthors in order to predict relationships between them in a coauthorship network(s). In this paper, the concept of co-author is presented as the act of participating in authorship or ownership of a journal, text, invention etcetera. The key objective of this study is to predict the coauthorship link, and also to compare relevant approaches.

Since, link prediction has an implicit structure of graph, a graph database (Neo4j) is implemented, as well as the usual machine learning methods are also used to get a clear and effective idea. A Graph Database (GDB) is a non-relational database which is doing exceptionally well in the present time working with abundant information, interconnected data, searching and providing efficient and effective outputs for specific requirements [2]. As the name suggests, a graph database stores data in the form of a graph, and displays the data in form of nodes, the relationship between the nodes in form of edges, which shows the link, or the relationship that exists between the data. Each node has its own properties,

labels, and similar nodes can be categorized in a group [3]. In this paper, the graph database features have been used to predict future co-author relationships. Authors and articles nodes have been created separately and were merged for further implementation using Cypher Query Language [4]. Moreover, several machine learning models, such as, Random Forest Classifier and Logistic Regression were implemented to get the prediction results using graph features.

The remaining part of the paper is structured out as follows: Section 2 discusses the background and related work done till date regarding this issue. In Section 3, the problem statement and proposed work are discussed in details. Section 4 shows the dataset characteristics and how they have been preprocessed beforehand for both the models, that is, Neo4j and traditional file, followed by Section 5, which presents the methodology for the given solution that includes feature characteristics for Neo4j, applied machine learning algorithms and model evaluation properties. Thenceforth, Section 6 describes the experimental setup and implementation part for the given problem statement with the traditional file and Neo4j approach. After that, the results of various experiments done over the above mentioned data sets and evaluates the better performance has been elaborated in Section 7, and finally, Section 8 concludes the paper with a vision and future scope on graph databases and data traditional approaches.

2. Background and Related Work

2.1. Graph Database: Neo4j

Neo4j is a database that is developed in Java [4] and can be used as an embedded or server database. The graph is made up of nodes (entities) that are connected to one another (depicted by relationships). Properties, which are key-value pairs, are used to store data in nodes and relationships. When opposed to other databases, Neo4j allows you to not only describe but also conveniently retrieve (traverse/navigate) related data.

2.2. Traditional File

Traditional python being the most widely used language leaves good scope for research and provides the opportunity to compare and contrast the performance of different methods that are being used.

2.3. Machine Learning Algorithm used for Traditional Approach

The machine learning algorithms applied in this work for predicting whether authors will collaborate in the future or not are Random Forest Classifier and Logistic Regression.

2.3.1. Random Forest

Random Forest is a classifier that combines a number of decision trees on different subsets of a dataset and averages the results to increase the predicted accuracy of the dataset [5]. To build a tree, the random forest classifier utilized in this study uses randomly selected characteristics or a combination of features at each node.

2.3.2. Logistic Regression

Logistic regression is a supervised Machine Learning algorithm and a statistical analysis method that is used for binary classification problems to predict a binary outcome [6]. This model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. This model or algorithm works better as predicting according to the data sets [5].

2.4. Related Works

Chuan et al. presented LDAcosin, a new metric for suggesting [7] writers to collaborate based on the content similarity of their articles. The more similar neighbors two nodes have, the higher the chances that they will be able to connect in the future. A coauthorship network, such as DBLP or ePrint arXiv, is considered, which denotes authors and linkages allude to collaboration between authors who have jointly published papers at a particular time. In the solution a few nodes are displayed as a three-dimensional vector (WCN, WAA, WJC) or a four-dimensional vector (WCN, WAA, WJC, LDAcosin) with labels 1 and 0. The suggested technique is based on estimating the comparable scores of author pairs for link prediction using content similarity measurements. It increases the degree of similarity between writers and considerably improves prediction accuracy. The key disadvantage and work gap of this approach is that, in comparison to other ways, the recommended methodology takes a long time to execute because it is in matrix form and requires sophisticated computations.

Kanakaris et al. [8] initially focused on closing the gap that exists between the link prediction works where unstructured textual data is not usually considered. Therefore, they extended previous future research collaboration by using a link prediction work and incorporated the existence of both structured and unstructured in one single knowledge graph, and observed how including the unstructured textual data via graph-of-docs affects the performance of the link prediction model. Afterward, for feature extraction, a range of graph measures and graph kernels were engaged. They applied feature combinations such as ALL, PM, WPM, AA_J (baseline), AA (baseline), p, J (baseline), AA_WPM, AA_P, and AA_PM to get accuracy, precision, and recall which are considered to be the main matrix for their performance evaluation. Finally, it was concluded that the involvement of both structural and textual data has greater chances to obviate overfitting and provides comparatively more accurate predictions.

Among the established work on link prediction, the studies closely relate to the work we demonstrate here are "Machine Learning within a Graph Database: A Case Study on Link Prediction for Scholarly Data" [9] and "On the utilization of structural and textual information of a scientific knowledge graph to discover future research collaborations: a link prediction perspective" [10].

In "Machine Learning within a Graph Database: A Case Study on Link Prediction for Scholarly Data" [9], the authors focused to incorporate ML models into Graph Databases. The principal cause was to create a singular platform wherein traditional storage and novel processing can be embedded. They have mainly portrayed the difference between the execution time of data splitting and feature calculation when performed inside and outside of the graph database. This study proposed to use the graph database to store information about features and assist with feature calculation.

In "On the Utilization of Structural and Textual Information of a Scientific Knowledge Graph to Discover Future Research Collaborations: A Link Prediction Perspective" [10], the prime target was to utilize the limitation of structured and unstructured data in graphs. To resolve the issue, both structured and unstructured textual data were represented in a scientific knowledge graph. The key features were extracted using graph measures and graph similarity techniques. Hence, future collaboration was determined from the ML model built with the help of extracted features.

Table 1 provides a brief overview of the related works stating the paper title, the key contributions and the certain limitations that their work faced .

Table 1. Summary of Contributions and Limitations of relevant papers.

| Paper Title | Key Contribution | Limitation |
|--|--|---|
| Machine Learning within a Graph Database: A Case Study on Link Prediction for Scholarly Data [9]. | Portrayed the time efficiency of feature calculation for in and outside of the graph database. | The main focus was time centric. |
| On the utilization of structural and textual information of a scientific knowledge graph to discover future research collaborations: a link prediction perspective [10]. | Integrating both structured and textual unstructured data in the graphs. | The relationship in the knowledge graph was not established using Neo4j. This could have solved the performance issue generated while generating nodes. |

3. Problem Statement and Proposed Work

3.1. Problem Statement

The modern world is full of connections and relations where a new phenomenon is found between every link. Books or articles written by famous authors, especially by collaborating with other well-known authors are what many publications seek.

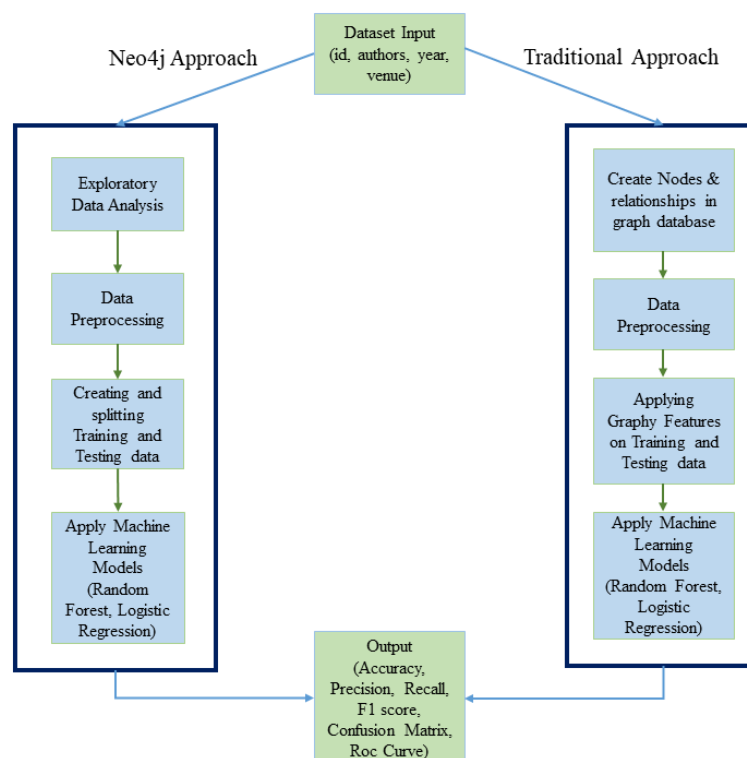
In this study, connections between authors collaborating in the future are predicted based on their previous collaboration records. So, a mathematical or reliable model can be formed for predicting the connection between the authors.

3.2. Proposed Work

The main goal is to predict the coauthor relationship that can take place in the future, and in order to detect the prediction value precisely, Neo4j and python is used.

In Neo4j, nodes and relationships are created between the attributes in order to assist in predicting coauthorship in the future. The neo4j approach of Figure 1 represents how the entire process is executed in a graph database (Neo4j). The dataset is preprocessed and split to make a fair training and testing data frame. After that machine learning models are used to make the prediction. Then, a comparison between the results is fairly observed, and is decided if our model is suitable for predicting links more accurately or not.

In the traditional approach, a relational method is developed for predicting coauthorship as shown in Figure 1. Lists and numpy arrays are used to store the relations and links. Normal python loops and functions are used to search and match the values. From these variable types, the values are predicted with those compared to graph databases.



1.png

Figure 1. Work Flow Diagram of Graph Database and Traditional Approach.

For both of our approaches, Random Forest Classifier and Logistic Regression Algorithms will be used to find out the accuracy of our model prediction. This work will help others to show how link prediction works efficiently and will be able to deal with more complex datasets in the future.

In this research, graph features in graph databases are created and simulated actual relationships and connections between authors for predicting links. Then in the traditional python method, data have been preprocessed normally using list, matrix, numpy array, and data frame and compared with that of the graph database.

4. Dataset

4.1. Dataset Characteristics

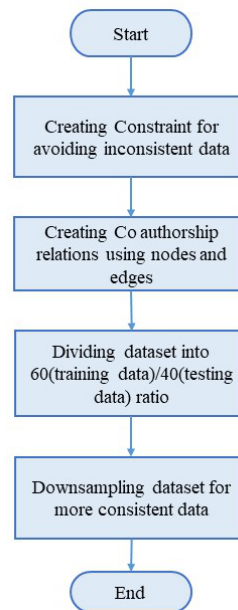
The dataset used here is the Citation Network Dataset [11], which has the details of published Articles for 60 consecutive years from 1958 to 2017. It contains information on authors who have collaborated in the past, the number of times their paper has been cited, references used, the title of the paper, venue, year of publication, id, and abstract.

4.2. Dataset Preprocessing

The current raw dataset needs to be preprocessed into a much cleaner, more understandable, and simpler form which would enhance the quality of the data, smoothen the dataset, give more accurate and consistent results and reduce redundancies. [12]. For the implementation in Neo4j, some constraints need to be created for ensuring data integrity [13]. On the other hand, in order to fit the dataset in Random Forest, the null values of the dataset must be taken care of as Random Forest does not support them. [14].

4.2.1. Preprocessing of Dataset for Neo4j

Preparing the dataset in a certain way is necessary before starting the work and sending them for training and testing. Figure 2 depicts the overall process of the implementation procedure for Neo4j. Initially, constraints are created on Article and Author using Cypher Query Language, which means that any repetition or inconsistency in the data will be eliminated. After that, the matching and merging operations will be done using Cypher Query Language once nodes and relationships have been created.



2.png

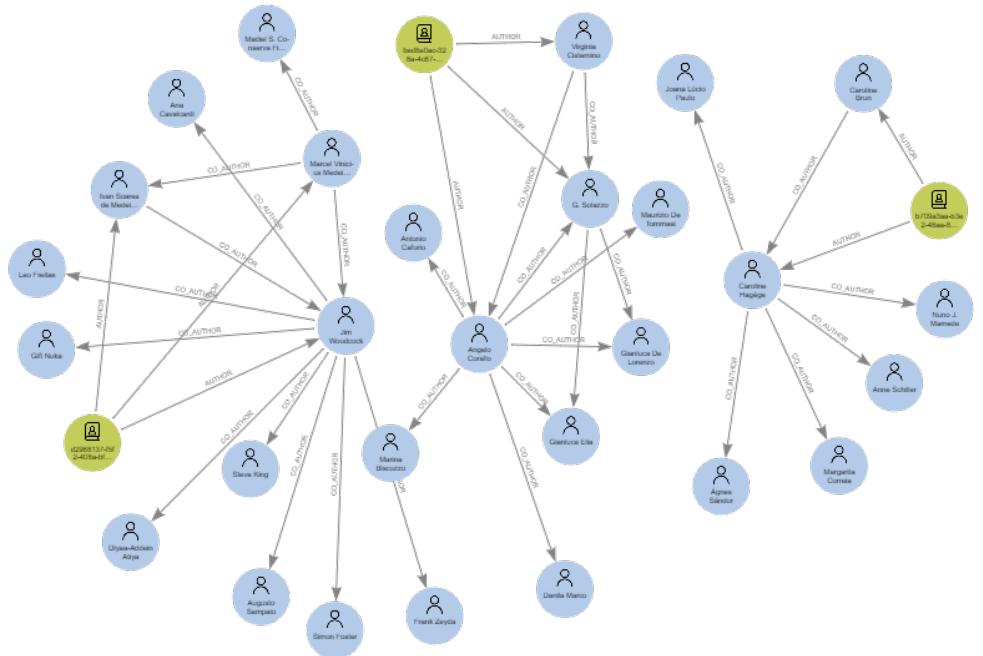
Figure 2. Work Flow Diagram of Preprocessing in Neo4j for Implementation.

The relationship between authors and articles, as well as some representations of those relationship in which authors have worked on the same paper and how they are connected as graphs has been exhibited in Figure 3, 4, 5 and 6. Also, Figure 7 portrays a high level view of our dataset.



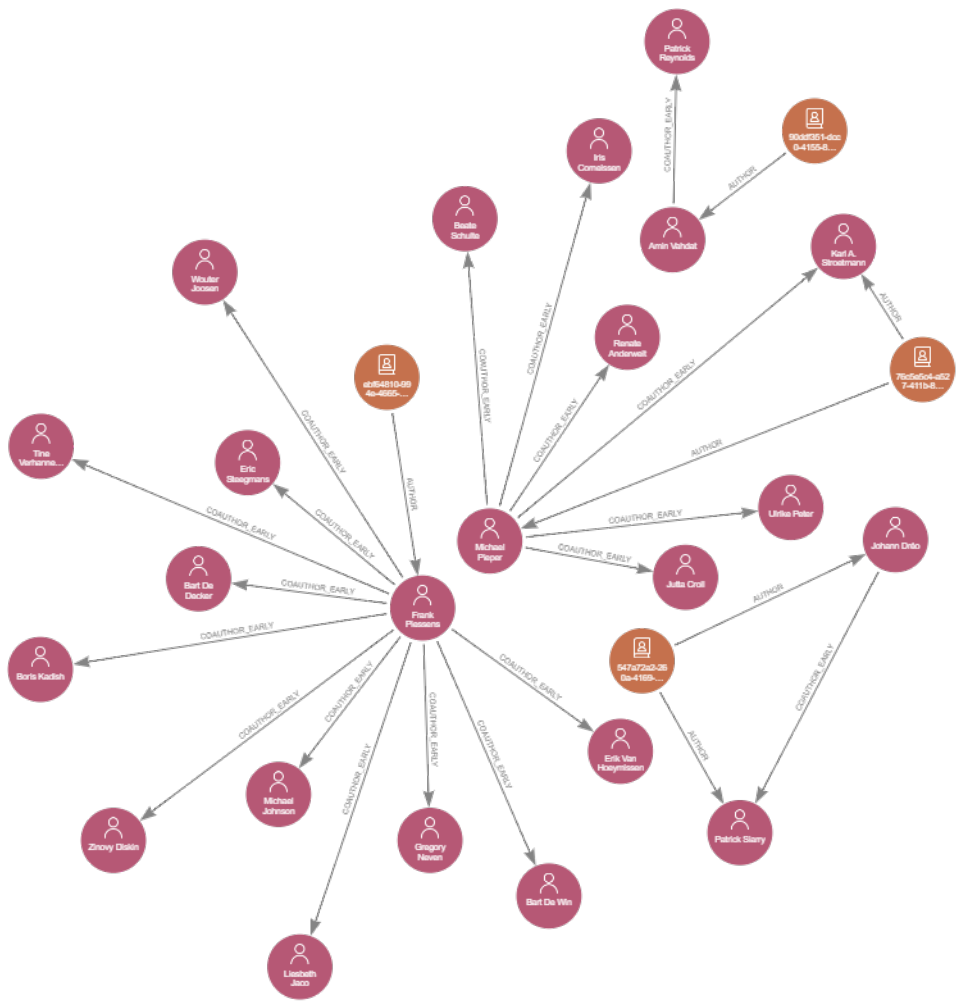
3.png

Figure 3. Relationship between Author and Article.



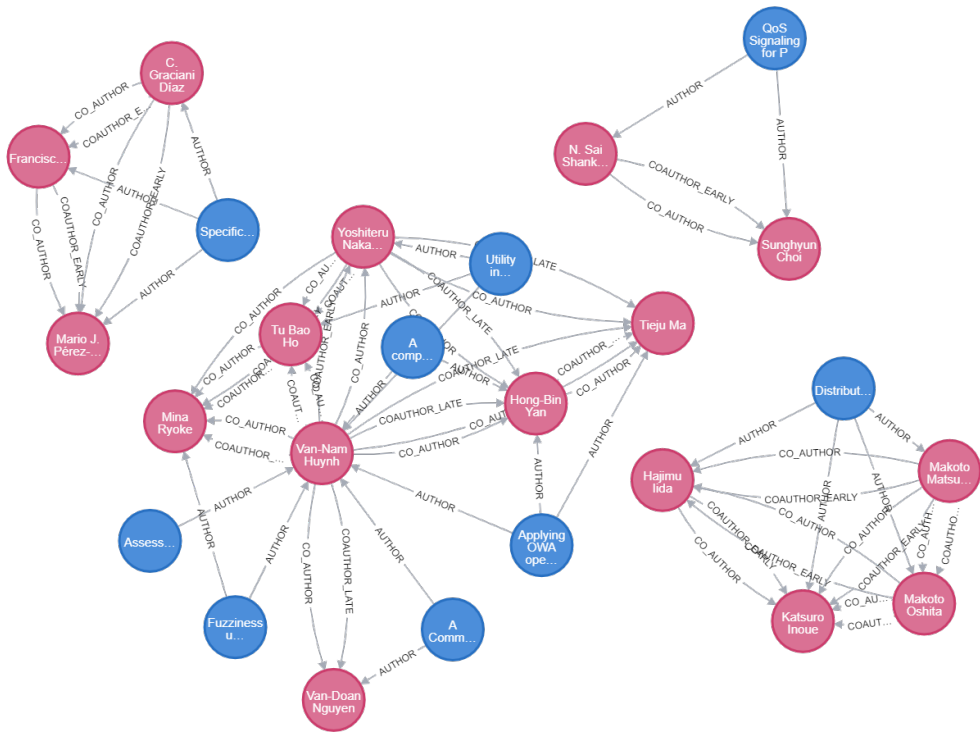
4.png

Figure 4. Coauthor Graph (Relationship between Authors through articles).

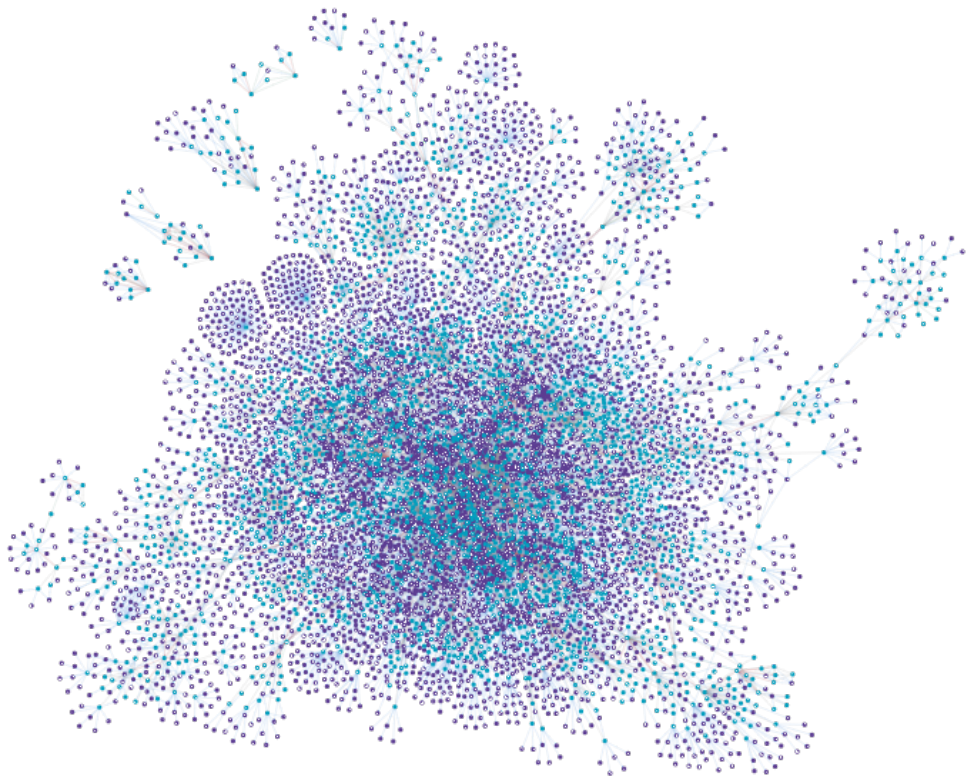


5.png

Figure 5. Coauthor Early Graph (Collaboration before 2006).



6.png
 Figure 6. Final Collaboration Graph with all Relationships.



7.png
 Figure 7. Aerial View of the Entire Graph.

The collaboration information of authors who have worked on the same paper is stored in "coauthor_collaboration". The dataset is sorted by year in order to do the splitting for training and testing purposes. Since the highest number of articles were published in 2006,

the dataset has been divided into a 60-40 ratio. The collaboration that happened before the year 2006, that is, the data from 1958-2005 have been stored on COAUTHOR_EARLY_Pairs for training the model, and the collaboration that happened from the year 2006, that is, the data from 2006-2017 have been stored in COAUTHOR_LATE_ as shown in Table 2 with a purpose of testing the model.

Table 2. Coauthor Pairs.

| Coauthor_Early_Pairs | Coauthor_Late_Pairs |
|----------------------|---------------------|
| 81096 | 74128 |

After acquiring the training and testing dataset, the pair of authors who have collaborated are stored and their relationships are checked to set the label. The label of the authors who have worked together will be set to 1. Additionally, we need to find some authors who have not collaborated so that we can train and test with the label value 0. The graph contains more authors who have not collaborated.

Therefore, there will be more author pairs who have not collaborated than the author who has. So, we will take the advantage of Graph Database for each training and testing set for both coauthor_early and coauthor_late relationships. From the graph, we will first check who satisfies the coauthor pair and then check the nearby edges to find out the author pairs who have not worked together on a paper yet and set their label to 0. Still, the number of author pairs whose label is 0 will appear enormous times surpassing the coauthor pairs (coauthor_early and coauthor_late in both cases). Hence, we need to balance our training and testing datasets. Now for the final step of data preprocessing, the dataset is down-sampled making it ready to be fit into the model for training and testing.

4.2.2. Preprocessing of Dataset for Traditional File Implementation

In order to work in the conventional python approach, the four JSON files (dblp-ref-0.json, dblp-ref-1.json, dblp-ref-2.json, dblp-ref-3.json) have been converted into four CSV files and then merged into a single CSV file named 'dblp-ref- small.csv'. It has 51956 tuples and 8 columns. In order to deal with the attributes, it is important to know whether null values are present or not, and what the data type of the attributes is, which is exactly what Table 3 characterized.

Table 3. Datatype and Null checking of each Attribute.

| Column Name | Data Type | No. of Null Values |
|-------------|-----------|--------------------|
| authors | object | 0 |
| n_citation | int64 | 0 |
| references | object | 13430 |
| title | object | 0 |
| venue | object | 0 |
| year | int64 | 0 |
| id | object | 0 |
| abstract | object | 7642 |

In Table 3, the 'authors' column defines the author collaboration for a specific paper and the column is in string format. The 'id' column is unique. There are only four unique venues where the articles were published. The 'year' column indicates when the article was published. The null value adaption in preprocessing poses a challenge.

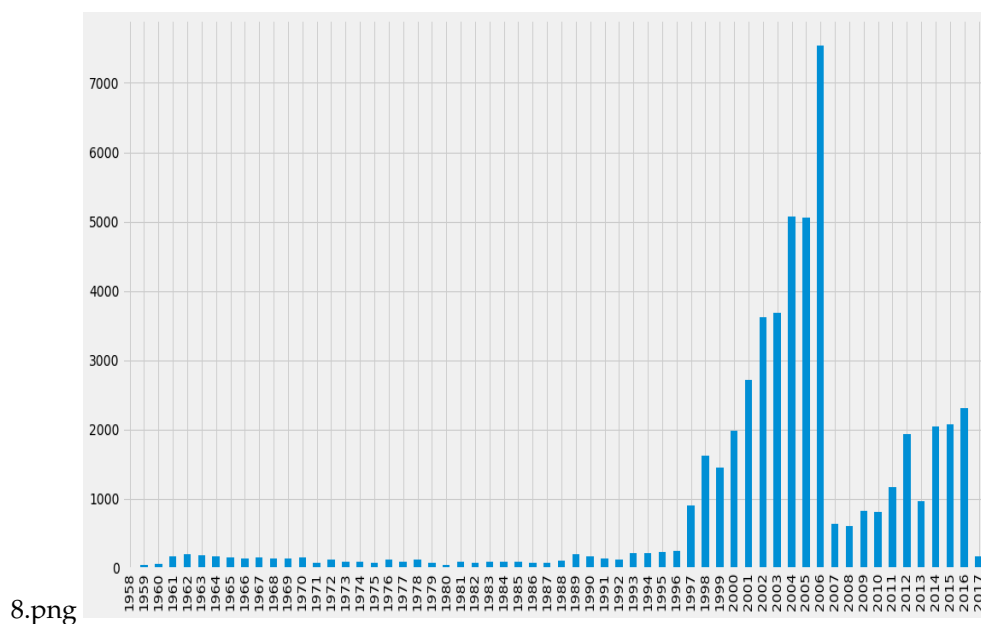


Figure 8. Total Number of Articles Published for each year from 1958 to 2017

According to Figure 8, maximum collaboration occurred in the year 2006. There are 46752 unique author collaborations, and 50% data is till 2005. Prior to 2006, there is 21059 published papers, and in 2006 and subsequent years, the number of papers published is 30897. We do not have the coauthor pairs for each year, but the purpose is to predict the coauthor link through the years. So primarily, we need to sort our data by year. Most of the author collaboration and article publications were in 2006 which is 7536. Now, for creating the dataset 'venue', 'n_citations', 'references', 'title', 'id', and 'abstract' columns are not needed for preprocessing, and with that, the problem of null value adaption is solved.

Figure 9 introduces the total flow of work for the preprocessing of traditional file implementation. On one hand, we need to create an array or list that contains author collaborations with the respective year that can be accessed with list indexing easily. It is required for the 'authors' and 'year' column to be stored in a NumPy array. Next, all the unnecessary characters such as space, third bracket, quotation marks, etcetera are replaced. After that, we split the names with commas and put them in a list. On the other hand, we need another list for keeping respective collaboration years. The indexing maps and puts collaboration authors in one list and their collaboration year in another list. Afterward, through the same mapping, we append this list. The list of authors is then called a collaboration list with its respective year alongside. It is a list of lists. In every nested list, the last position has the year and the rest of it is collaboration. Now we will find the unique author name and make a list. In order to do that, a unique author collaboration needs to be created. Since it is a string list and the name of the author is not accessible, all the unnecessary characters need to be removed once more. All the names that can be repeated are put into a long string, and the names of authors are in comma separation. After the splitting is done, we find 132462 author names.

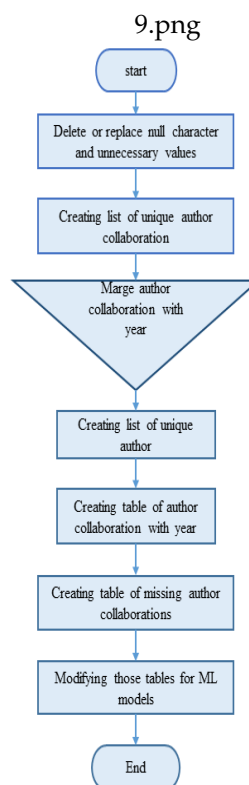


Figure 9. Flow Chart of Traditional File Implementation Preprocessing.

We found 80290 unique authors and put them on a list. The index value of each author will be known as the author's unique id. Now, we have to create a 2D matrix with 4 columns named 'author_1', 'author_2', 'collab_label' and 'year'. At first, we will store the author index value, and then find collaboration for the specific author. Next, we will store his or her coauthor with their collaboration year and put 1 in as collab_label. For further execution, the collaboration will be kept in a separate CSV file. So, 'author_collab_JN_unique.csv' is the final CSV file that we can directly fit into an ML model. However, we got no zero values in this file, which indicates non-collaboration between authors. Hence, we have to store some zero values in another file namely the non-collaboration array list. There are many authors who did not collaborate with each other, so, we need to go through the 1...2 hop.

To make that list, we go 1...2 hop from the coauthor index of its author and check whether the author collaborated within those ranges or not. If it appears that the author did collaborate from its initial coauthors 1...2 hop range, it will not store the value because it is already stored in the collaboration file. Otherwise, it will store the author index with its non-coauthor index alongside the collab_label as '0' and null string for the year. This way, the 'author_non_collab_JN_unique.csv' file for non-collaboration is created followed by the files which are being fit into ML models.

5. Methodology

5.1. Feature Characteristics for Neo4j

A simple Graphy Feature Model was created to find out whether two authors will have a future collaboration based on the features extracted from Common Neighbors, Preferential Attachment, and Total Number of Neighbors.

5.1.1. Common Neighbors

The term "Common Neighbors" refers to two strangers who share a mutual friend and are more likely to be introduced than those who do not [15]. In this paper, the number of potential triangles between two writers is calculated using common neighbors. This expresses the possibility of two authors who share coauthors being introduced and collaborating in the future.

$$CN(x, y) = | N(x) \cap N(y) | \quad (1)$$

Here, $N(x)$ is the set of nodes adjacent to node x , and $N(y)$ is the set of nodes adjacent to node y . A value of 0 indicates that two nodes are not close, while higher values indicate nodes are closer.

5.1.2. Preferential Attachment

Preferential attachment states that new authors are more likely to attach to older authors who already have a network of contacts [16]. It assigns a score to each pair of authors based on the number of coauthors they have.

$$PA(x, y) = | N(x) * N(y) | \quad (2)$$

PA value of 0 indicates that two nodes are not close, while higher values indicate that nodes are closer.

5.1.3. Total Neighbors

Total Neighbors calculates a node's proximity based on the number of unique neighbors it has [17]. The premise behind it is that the more linked a node is, the more probable it is to get new linkages.

$$ToN(x, y) = | N(x) \cup N(y) | \quad (3)$$

where $N(x)$ is the set of nodes adjacent to x , and $N(y)$ is the set of nodes adjacent to y . A value of 0 indicates that two nodes are not close, while higher values indicate nodes are closer.

5.2. Model Evaluation Properties

5.2.1. Precision

It measures how many correct positive forecasts have been made [18]. Precision calculates the minority class's accuracy. The result value is between 0-1.

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

5.2.2. Recall

Recall is a statistic that measures how many correct positive predictions were produced out of all possible positive predictions [18]. The value is between 0-1.

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

5.2.3. F1-Score

The F1-score is mostly used to compare the results of two different classifiers [18]. When the F1 score is 1, the model is deemed perfect, but when it is 0, the model is considered a complete failure.

$$F1Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (6)$$

5.2.4. Confusion Matrix

An $N \times N$ matrix is used to evaluate the performance of a classification model, where N is the number of target classes [19]. The matrix compares the actual goal values to the machine learning model's predictions. The target variable has two values: Positive or Negative. The columns represent the actual values of the target variable. The rows represent the predicted values of the target variable.

5.2.5. ROC Curve

The Receiver Operating Characteristics Curve, or ROC curve, is a statistic for evaluating the performance of a classifier model [20]. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test. The findings of the area under the ROC curve (AUC) were deemed excellent for AUC values of 0.9-1, good for AUC values of 0.8-0.9, acceptable for AUC values of 0.7-0.8, bad for AUC values of 0.6-0.7, and failed for AUC values of 0.5-0.

6. Experimental Setup

6.1. Environment

For the Graph Database approach, we have used Neo4j Desktop and Neo4j Sandbox to run our Cypher Query and connected the Neo4j server with Visual Studio Code and Jupyter Notebook of Anaconda so that we could run Cypher Query in python shell. We have used the 'Python 3.8.8 64-bit' kernel. The device Configuration is Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz, RAM 12.0 GB (11.9 GB usable), 64-bit operating system, x64-based processor, Windows 10.

For the Traditional Approach, we have used Visual Studio Code and Jupyter Notebook of Anaconda to run our python codes. For this, we have used the 'Python 3.10.0 64-bit' kernel. The device Configuration is Intel(R) Core(TM) i3-7100U CPU @ 3.90GHz, RAM 12.0 GB (11.9 GB usable), 64-bit operating system, x64-based processor, Windows 10 Pro.

6.2. Neo4j Implementation

One of the main reasons for implementing the coauthor prediction in Neo4j is because of the interface and facilities that it provides. Creating nodes, properties, relationships, and finding the nearest neighbors, is comparatively less complex in Neo4j. It gives an overall snapshot of the whole scenario and the visualization makes the prediction seem more reasonable and understandable.

The Graph Algorithm Library Supports a total of 6 link prediction algorithms and we are going to apply 3 algorithms for the purpose of our prediction [21]. After preprocessing the dataset, the node pairs and their relationship type from the training data frame COAUTHOR_EARLY and testing data frame which contains COAUTHOR_LATE relationships are sent to the user-defined function graphy_features. The "UNWIND" feature of Neo4j takes this huge collection of node pairs and returns common neighbor (cn), preferential

attachment (pa), and total neighbor (tn) in one single query which is required for our prediction purpose [22].

6.2.1. Random Forest Implementation

The result of common neighbor, preferential attachment, and total neighbor for both train and test datasets with different parameters to observe the output for Random Forest Classifier is given in Table 4:

Table 4. Different Parameter of Training and Testing Dataset for Random Forest.

| Training Dataset | | | | | | Testing Dataset | | | | | |
|------------------|--------|-------|-----|------|------|-----------------|-------|-------|-----|-------|------|
| node1 | node2 | label | cn | pa | tn | node1 | node2 | label | cn | pa | tn |
| 27545 | 75873 | 0 | 0.0 | 96.0 | 20.0 | 118412 | 53349 | 0 | 1.0 | 10.0 | 6.0 |
| 28106 | 77951 | 0 | 0.0 | 15.0 | 8.0 | 81239 | 71969 | 0 | 0.0 | 76.0 | 23.0 |
| 50936 | 115653 | 1 | 5.0 | 84.0 | 15.0 | 100859 | 70321 | 0 | 1.0 | 27.0 | 11.0 |
| 11785 | 11788 | 1 | 1.0 | 12.0 | 6.0 | 70055 | 71439 | 1 | 3.0 | 414.0 | 52.0 |
| 73129 | 11932 | 0 | 0.0 | 3.0 | 4.0 | 16783 | 29439 | 1 | 5.0 | 160.0 | 23.0 |

The training of the dataset is done with respect to cn, pa, and tn, and is fit into the Random Forest classifier model. After the training has been completed, the same process is followed with the test data set COAUTHOR_LATE for prediction.

6.2.2. Logistic Regression Implementation

The implementation of the Logistic Regression Algorithm follows an approach similar to the Random Forest Algorithm. The result of common neighbor, preferential attachment, and total neighbor using different parameters of both training and testing data set is shown in Table 5:

Table 5. Different Parameter of Training and Testing Data set for Logistic Regression.

| Training Dataset | | | | | | Testing Dataset | | | | | |
|------------------|--------|-------|-----|------|------|-----------------|--------|-------|-----|------|------|
| node1 | node2 | label | cn | pa | tn | node1 | node2 | label | cn | pa | tn |
| 15570 | 43542 | 0 | 0.0 | 68.0 | 21.0 | 13058 | 120540 | 1 | 2.0 | 24.0 | 9.0 |
| 52852 | 100706 | 0 | 1.0 | 20.0 | 8.0 | 16172 | 43949 | 1 | 1.0 | 42.0 | 12.0 |
| 10959 | 10961 | 0 | 1.0 | 4.0 | 3.0 | 16650 | 16651 | 1 | 2.0 | 81.0 | 28.0 |
| 25562 | 25563 | 1 | 3.0 | 16.0 | 5.0 | 23590 | 71500 | 0 | 1.0 | 63.0 | 15.0 |
| 72820 | 72821 | 1 | 1.0 | 4.0 | 3.0 | 31072 | 31072 | 0 | 0.0 | 9.0 | 3.0 |

The training of the dataset is done with respect to cn, pa, and tn, and is fit into the Logistic Regression model. After the training has been completed, the same process is followed with the test dataset COAUTHOR_LATE for prediction.

6.2.3. Traditional File Implementation

For our model to learn the data, we have replaced null values with zeroes in the non-collaboration dataset in the 'year' column. Subsequently, we have sorted the collaboration data frame with respect to the year. In Neo4j, we have split the data by the following: the year after 2006 is for testing data and all the rest is for training data. The test data is from 2006 to 2017, and the train data range from 1958 to 2005. The total number of rows in the main dataset is 341038, and the train and test datasets are 175650 and 165388 respectively. Here, the splitting percentage is 52% training and 48% testing. If we split our non-collaboration data by 67.23% train size and then merge it with collaboration train and test dataset, then the total train-test split will become 60% and 40% according to our graph database approach and ML will not overfeed. As for the non-collaboration data, we

have split it randomly. After that, we need to merge all the train data of collaboration and non-collaboration into one and do the same for test data. Lastly, the files are simply loaded and fit into the machine learning algorithm: random forest and logistic regression. As the datasets are huge, in Table 6 only 5 rows are shown for better understanding.

Table 6. Training and Testing Dataset for Traditional Approach.

| Training Dataset | | | | Testing Dataset | | | |
|------------------|----------|----------|-------|-----------------|----------|----------|-------|
| Serial no. | author_1 | author_2 | label | Serial no. | author_1 | author_2 | label |
| 12 | 3 | 4 | 1 | 222550 | 48616 | 48614 | 1 |
| 58 | 16 | 15 | 1 | 219266 | 47736 | 7101 | 1 |
| 30 | 9 | 10 | 1 | 213701 | 46179 | 49387 | 1 |
| 300552 | 59750 | 12178 | 0 | 169221 | 30950 | 37965 | 0 |

7. Result and Analysis

7.1. Analysis of Proposed Algorithm

We have implemented the Random Forest Classifier and Logistic Regression for predicting binary class labels for 0 and 1.

For Neo4j, in the Random Forest Classifier, the parameters we have used are `n_estimators` (which means the number of trees in the forest, we have worked with value 30), `max_depth` (maximum depth of the tree). We have set the value of `max_depth` as 10 and `random_state` as 0. It means that when a tree is being built, there should be no randomness of the bootstrapping of the samples [23]. In logistic regression, we have used two parameters: `solver='saga'`, and `max_iter=50`. Here, `solver` means a solving algorithm for optimizing the problem. 'Saga' is a solving method and `max_iter` means the maximum number of iterations taken for the solvers to converge. Here, it will take 50 iterations to converge [24].

In the traditional python scikit learn approach, we used random forest but different parameters and values. The `n_estimators` is set to 100, which means there should be 100 trees in the forest. The `n_jobs` parameter means the number of jobs to run in parallel. Here the value of `n_jobs` is set as 4 [23]. We also used a logistic regression machine learning model, and the parameter is `solver='newton-cg'` [24].

7.2. Comparative Analysis of the Performance of both Algorithms

7.2.1. Neo4j

In Neo4j, we have implemented both random forest classifier and logistic regression. In the first part, the training and testing data set were created for 2 to 3 hops of the existing coauthorship early and late graphs. In the next part of training and testing, datasets were created for 3 to 4 hops nearly edges between the coauthorship early and late graph.

7.2.2. Traditional Approach

We have also implemented both the random forest classifier and logistic regression algorithm for better comparison. In the preprocessing, we split the collaboration data into two parts. From 1958-2005 is set for training data and 2006-2017 is set for test data. Also, non-collaboration data which is created using 1 to 2 hop is split and merged with collaboration testing and training data. It is then fit into both the machine learning models.

7.3. Evaluating Models

7.3.1. Neo4j Analysis

In Neo4j, random forest classifiers performed better than logistic regression. We can see that through the following tables and graphs.

Precision, Recall and F1-score

A model that produces a higher precision score generates fewer false positives. A model that produces a higher recall score generates fewer false negatives. A model that produces a higher F-1 score indicates more perfect precision and recall.

Table 7. Precision, Recall, F-1 Score for Random Forest Classifier and Logistic Regression in Neo4j.

| | Random Forest | | | Logistic Regression | | |
|----------|---------------|--------|----------|---------------------|--------|----------|
| | Precision | Recall | F1-Score | Precision | Recall | F1-Score |
| 2..3 hop | 0.93 | 0.90 | 0.91 | 0.91 | 0.79 | 0.85 |
| 3..4 hop | 0.97 | 0.91 | 0.94 | 0.96 | 0.81 | 0.88 |

As it appears in Table 7, in both of the cases, 3..4 hop gives better results in terms of precision, recall, and F1-score.

Confusion Matrix

Table 8 presents the output of the confusion matrix for both 2..3 hops and 3..4 hops.

Table 8. Confusion Matrix for Random Forest Classifier and Logistic Regression.

| | Random Forest Classifier | | | | Logistic Regression | | | |
|----------|--------------------------|-------|------|------|---------------------|-------|-------|------|
| | TN | TP | FN | FP | TN | TP | FN | FP |
| 2..3 hop | 68730 | 66694 | 7434 | 5398 | 68136 | 58846 | 15282 | 5992 |
| 3..4 hop | 71913 | 67675 | 6453 | 2215 | 71856 | 60298 | 13830 | 2272 |

ROC Curve

From the ROC curve value given in Table 9, we can conclude that for 3..4 hops, the model is performing better for both Random Forest and Logistic Regression in Graph Database (Neo4j).

Table 9. ROC Curve values for Random Forest Classifier and Logistic Regression.

| | Random Forest Classifier | Logistic Regression |
|----------|--------------------------|---------------------|
| 2..3 hop | 0.97 | 0.93 |
| 3..4 hop | 0.98 | 0.95 |

For the given values, our model is also predicting that 3..4 hop performs better.

7.3.2. Traditional Approach Analysis

For the Traditional python approach, the random forest classifier performed slightly better than logistic regression. We can see that from the tables and graphs.

Precision, Recall, F-1 Score

Here both algorithms kind of gave the same result. However, the recall value is slightly better in the case of Random Forest Classifier which does not really make any sufficient changes.

Confusion Matrix

Representation of the Confusion Matrix for both models is applied in the traditional approach.

ROC Curve

For the traditional approach, neither Random Forest Classifier nor Logistic Regression gave any satisfactory result as as shown in the ROC curve values in Table 10.

Table 10. Precision, Recall, F-1 Score, Confusion Matrix and ROC Curve values of Random Forest Classifier and Logistic Regression for traditional approach.

| | Precision | Recall | F1 Score | TN | TP | FN | FP | ROC Values |
|---------------------|-----------|--------|----------|-------|------|-------|--------|------------|
| Random Forest | 0.05 | 0.21 | 0.08 | 98475 | 8562 | 32847 | 156826 | 0.28 |
| Logistic Regression | 0.05 | 0.20 | 0.08 | 98215 | 8288 | 33107 | 157100 | 0.29 |

7.3.3. Classification Accuracy

Table 11 gives a clear picture of the accuracy of both models when implemented in the two approaches: Neo4j and Traditional File. For 3...4 hops, both Random Forest Classifier and Logistic Regression gave better accuracy. As for the traditional file, the accuracy is shown alongside that of 2...3 hop and 3...4 hop in Table 11 for better comparison.

Table 11. Accuracy of Random Forest and Logistic Regression for neo4j and traditional file.

| | 2..3 hops | 3..4 hops | Traditional File |
|---------------------|-----------|-----------|------------------|
| Random Forest | 0.91 | 0.94 | 0.36 |
| Logistic Regression | 0.86 | 0.89 | 0.36 |

We have performed Random Forest Classifier and Logistic Regression for both Graph Database and Traditional Approach. In Neo4j, Random Forest Classifier gave better accuracy than the Logistic Regression algorithm. In the traditional procedure, none of the algorithms provided satisfactory accuracy. However, in the traditional approach, Random Forest Classifier worked slightly better than Logistic Regression.

8. Conclusion and Future Work

In conclusion, we can clearly state that in the case of coauthor link prediction, the Graph Database performed far better than the traditional approach. It was easier to create nodes from the existing dataset and create relationships between them and implement different methods within a short time. In Neo4j, the training and testing datasets that were created for 3 to 4 hops gave better accuracy than the training and testing datasets for 2 to 3 hops. This was because the 3 to 4 hops model was trained with more accurate data. In the link prediction mechanism, the Graph Database works more efficiently and fast.

So, for link prediction in the Traditional Approach, we cannot create links like a graph (node and edges) which was a great barrier for us to implement the connection between authors conveniently. We had to undergo lengthy preprocessing to find out the collaborations and unique authors. The most challenging part was to find the coauthor pair list with year as well as to find some non-collaboration pairs which was much easier in the Neo4j graph database, even though it took a while to find out all the relationships and uniqueness.

It is evidently proved that Graph Databases will always perform better when it comes to link prediction. Social network analysis, citation network analysis, the outbreak of any disease, page ranking, mapping etcetera involve link prediction. To acquire satisfactory and more accurate results, the Graph Database should be implemented vastly. We have worked with a relatively small dataset or nodes and relationships, but graph databases can handle large amounts of data.

We have only used the 'authors' and 'year' columns for predicting the link between any two authors. If we had considered 'venue' for creating connections, it could have given much more accurate results to predict the link between authors.

Abbreviations

| | |
|------|--|
| MDPI | Multidisciplinary Digital Publishing Institute |
| GDB | Graph Database |
| JSON | JavaScript Object Notation |
| CSV | Comma Separated Values |
| ML | Machine Learning |
| CN | Common Neighbour |
| PA | Preferential Attachment |
| TN | Total Neighbour |
| TP | True Positive |
| FP | False Positive |
| FN | False Negative |
| TN | True Negative |
| ROC | Receiver Operating Characteristic |
| AUC | Area Under the ROC Curve |

References

1. Savić, M.; Ivanović, M.; Jain, L.C. Co-authorship networks: An introduction. In *Complex Networks in Software, Knowledge, and Social Systems*; Springer International Publishing AG: Cham, 2019; Vol. 148, pp. 179–192. https://doi.org/10.1007/978-3-319-91196-0_5.
2. Why Graph Databases? The Advantages of Using a Graph Database. <https://neo4j.com/why-graph-databases>, accessed on 25 February 2022.
3. Miller, J.J. Graph database applications and concepts with Neo4j. In Proceedings of the Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA; Georgia Southern University, , 23 March 2013; Vol. 2324, p. 36.
4. Vukotic, A.; Watt, N.; Abedrabbo, T.; Fox, D.; Partner, J. *Neo4j in Action*; Manning, 2015.
5. Liu, Y.; Wang, Y.; Zhang, J. New machine learning algorithm: Random forest. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Liu, B.; Ma, M.; Chang, J., Eds.; Basic Teaching Department, Tangshan College, Springer, Berlin, Heidelberg: Berlin, Heidelberg, 2012; Vol. 7473 LNCS, pp. 246–252. https://doi.org/10.1007/978-3-642-34062-8_32.
6. Lawton, G. Logistic Regression. <https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression>, accessed on 06 February 2022.
7. Chuan, P.M.; Son, L.H.; Ali, M.; Khang, T.D.; Huong, L.T.; Dey, N. Link prediction in co-authorship networks based on hybrid content similarity metric. *Applied Intelligence* **01 August 2018**, *48*, 2470–2486. <https://doi.org/10.1007/s10489-017-1086-x>.
8. Kanakaris, N.; Giarelis, N.; Siachos, I.; Karacapilidis, N. for Predicting Future Research Collaborations. *Entropy* **25 May 2021**, *23*, 2–18. <https://doi.org/10.3390/e23060664>.
9. Sobhgol, S.; Durand, G.; Rauchhaupt, L.; Saake, G. Machine Learning within a Graph Database: A Case Study on Link Prediction for Scholarly Data. In Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS. INSTICC, SciTePress, 30 January 2021, Vol. 01, pp. 159–166. <https://doi.org/10.5220/0010381901590166>.

10. Giarelis, N.; Kanakaris, N.; Karacapilidis, N. On the Utilization of Structural and Textual Information of a Scientific Knowledge Graph to Discover Future Research Collaborations: A Link Prediction Perspective. In Proceedings of the Discovery Science; Appice, A.; ; Tsoumakas, G.; ; Manolopoulos, Y.; ; Matwin, S., Eds.; Springer International Publishing: Cham, 15 October 2020; Vol. 12323 LNAI, pp. 437–450. https://doi.org/10.1007/978-3-030-61527-7_29.
11. Dataset of Co-author Link Prediction. <https://github.com/mneedham/link-prediction/tree/master/data>, accessed on 08 November 2021.
12. Anunaya, S. Data Preprocessing in Data Mining -A Hands On Guide. <https://www.analyticsvidhya.com/blog/2021/08/data-preprocessing-in-data-mining-a-hands-on-guide>, accessed on 05 February 2022.
13. Constraints. <https://neo4j.com/docs/cypher-manual/current/constraints>, accessed on 07 February 2022.
14. Data Preprocessing in Python. <https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python>, accessed on 07 February 2022.
15. Yang, Z.; Hu, R.; Zhang, R. An improved link prediction algorithm based on common neighbors index with community membership information. In Proceedings of the 7th IEEE International Conference on Software Engineering and Service Sciences, ICSESS; IEEE: Beijing, China, 26-28 August 2016; pp. 90–93. <https://doi.org/10.1109/ICSESS.2016.7883022>.
16. Ruj, S.; Pal, A. Preferential attachment model with degree bound and its application to key predistribution in WSN. In Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications, AINA; Institute of Electrical and Electronics Engineers Inc.: Crans-Montana, Switzerland, 23 May 2016; pp. 677–683, [1604.00590]. <https://doi.org/10.1109/AINA.2016.141>.
17. Qiu, B.; Wang, J.; Liu, Y.; Xu, Z. Neighbor sum distinguishing total colorings of graphs with bounded maximum degree and maximum average degree. In Proceedings of the IEEE International Conference on Computational Science and Engineering and IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, CSE and EUC 2017; Institute of Electrical and Electronics Engineers Inc.: Guangzhou, China, 10 August 2017; Vol. 1, pp. 898–901. <https://doi.org/10.1109/CSE-EUC.2017.182>.
18. Accuracy, Precision, Recall and F1 Score: Interpretation of Performance Measures. <https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures>, accessed on 09 February 2022.
19. N B, H. Confusion Matrix, Accuracy, Precision, Recall, F1 Score. <https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd>, accessed on 08 February 2022.
20. Bhandari, A. AUC-ROC Curve in Machine Learning Clearly Explained. <https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning>, accessed on 09 February 2022.
21. Needham, M. Link Prediction with Neo4j Part 2: Predicting co-authors using scikit-learn. <https://towardsdatascience.com/link-prediction-with-neo4j-part-2-predicting-co-authors-using-scikit-learn-78b42356b44c>, accessed on 09 November 2021.
22. Needham, M.; Hodler, A.E. *Mark Needham Graph Algorithms Practical Examples in Apache Spark and Neo4j*, illustrate ed.; O'Reilly Media, Inc.: 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2019; p. 231.
23. Random Forest Classifier Parameter. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>, accessed on 09 February 2022.
24. Logistic Regression Parameter. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, accessed on 09 February 2022.