A Study of Applying Graph Database and Traditional Approach to forecast Co-author link Prediction based on Machine Learning Models.

A thesis to be submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

by

Samia Sultana	Sanjeda Sara Jennifer	Fariha Fairuz	Shafiul Mahmud Partho
ID: 2017-2-60-122	ID: 2018-1-60-061	ID:2018-1-60-080	ID:2018-1-60-219

Under the supervision of

Dr. Mohammad Rezwanul Huq

DEPT. OF COMPUTER SCIENCE AND ENGINEERING

EAST WEST UNIVERSITY

DHAKA, BANGLADESH

February 11, 2022

TITLE: A Study of Applying Graph Database and Traditional Approach to forecast Co-author link Prediction based on Machine Learning Models.

Abstract: In co-authorship, 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 our project using nodes to determine most possible co-authors in future. In order to forecast the connections, we pre-processed the dataset 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, which stimulates us to conduct further study on employing such a forecast.

Keywords: Co-Authorship, Link Prediction, Graph Database (Neo4j), Nodes, ROC, ML.

1. Introduction

The issue of anticipating the presence of a relationship between nodes in a network is known as link prediction. In online social networks, link prediction is used to anticipate new interactions between members that are likely to occur in the future. Researchers' collaborations frequently have an interdependent effect. It's tough to know which possible cooperation should be approached ahead of time. Link prediction in the co-authorship network has been one of the key objectives in link prediction studies thus far. It is better to construct a similarity measure between two omnipresent writers and then use it to find the most potential co-authors in order to deliver exact predictions of relationships between them in a co-authorship network(s). Understanding the other approaches for link prediction, a graph database (neo4j) is implemented in our project as well as the usual machine learning methods to get a clear and effective idea. To handle the challenge of anticipating the presence of a relationship between nodes in a graph, the graph database technique is used.

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[1].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[2].

In our project, 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. Moreover, several machine learning models such as, Random Forest Classifier and Logistic Regression were implemented to get the prediction results using graphy features. These model implementations show that the accuracy result of the prediction considerably varies with each other. By evaluating the predicted results with the real collaborations, we can estimate their validity.

The remaining part of the paper is structured out as follows: Section 2 discusses the related work. In section 3, we described the problem statement and proposed work. Section 4 shows the dataset characteristics and preprocessing for the models in both neo4j and traditional file. Section 5 presents the methodology for the given solution which includes feature characteristics for Neo4j, applied machine learning algorithms and model evaluation properties. Section 6 describes the implementation part for the given problem statement with the traditional file and neo4j approach. Section 7 discusses the results of various experiments done over the above mentioned datasets and evaluates the better performance. Finally, Section 8 concludes the paper with a vision of future scope on graph databases and datasets.

2. Related Works

Chuan et al. presented LDAcosin, a new metric for suggesting[3] writers to collaborate based on the content similarity of their articles. The more similar neighbors two nodes have, the more probable they will be able to connect in the future. A co-authorship network, such as DBLP or ePrint arXiv, is considered, in which denotes authors and linkages allude to collaboration between authors who have jointly published papers at a specific 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. In this circumstance, our proposed graph database technique is a far superior solution. We can manage the generated nodes and provide a better performance with much less duration by utilizing the graphy-features. Also through the SVM method the area under the ROC curve(AUC) shows a poor score between (0.60-0.67). As a result, referencing the limitations of the paper, other parameters must be considered in the new metric to enhance AUC values.

Sara Cohen and Netanel Cohen-Tzemach proposed[4] combining relational databases (MySQL), data storage systems (Redis), and graph data storage systems to implement link prediction (neo4j). Furthermore, through the visualization of common neighbors, jaccard coefficients, adamic-adar, graph distance, katz measure, and rooted pagerank, 1000 nodes were randomly selected from the network to provide the top 100 prospective neighbors to determine the co-authorship. The background logic of the paper was, link prediction is a typical social network function that has been thoroughly investigated, and hence effective metrics of interest for link prediction have already been defined. Also and more critically, the computational complexity of link prediction metrics varies greatly, for example, they may analyze only a small area of a node for which new associations should be anticipated, or they may execute random walks throughout the whole social network graph.The node distances were generated by using the cypher query language. The accessible cypher query language was used in Neo4j implementations, which does not appear to be well optimized for recursive calculations. Thus this refers to the drawback of this solution and also the jaccard coefficients and adamic-adar matices were not implemented fully in the Neo4j platform. This implementation gap was fulfilled in our work. However, the overall procedure was unsuitable for hybrid systems and had some drawbacks for end users, such as a graph representation (cypher query-like).

Tingli Wang and Guoqiong Liao reviewed[5] a Review of Link Prediction in Social Networks. Many methods for implementing the link prediction were introduced. The paper presents the notion of link prediction and highlights the most recent link prediction research and methodologies. Methods Based on Similarity Measures, such as Common Neighbor (CN), Jaccard’s Coefficient (JC), Adamic/Adar (AA), Preferential Attachment (PA) etc. Methods Based on Probabilistic Model which Wang et al. [4] introduced a local probabilistic model for link prediction that models the local neighborhood comprising two nodes using an undirected graphical model called Markov Random Fields. They develop a local model rather than a global model to lower the expense of big scale networks. The technique is divided into two stages: first, determining which nodes will be included in the local model, and then using frequent non-derivable itemsets to identify the structure of the graphical model as well as learning the model's parameters. For heterogeneous models the paper examined that there are two common approaches to addressing link prediction for heterogeneous networks, according to available research. To begin, consider all types of links equally and investigate each type of connection independently, ignoring any link type correlation. These assessments aided in the development of the implementation for our suggested technique by clarifying the principles of link prediction in our project.

3. Problem Statement and Proposed Work

3.1 Introduction

In this section, we will discuss our problem and the reason we tend to work with the link prediction as well as the importance of the problems and how it should concern in the future world. Also we will also propose some algorithm to find out the link and we will also describe the goal and objectives of our work.

3.2 Problem Statement

Modern world is full of connections. Many relationships and connections are occurring every time. People communicate through links and networks. Nowadays many phenomena are conducted through links and connections. Our virtual life is nothing but some link and relations. Many criminal activities as well as detective activities are followed through link. This is becoming a huge issue of our era and everyone is finding a link between every phenomenon and person.

A paper can be written by many authors at many different times. Some publications always want to publish famous authors' books or articles. They may want more production and publication, so they might search for a famous author and ask for a collaboration with another famous author. Also, some may want to find a collaboration according to previous collaboration. Others might want to predict whether an author could collaborate in the future or not. This is a small concern in front of other important issues.

With this psychology, we wanted to predict whether any two authors would collaborate in future or not according to their previous collaborations. So, a mathematical or reliable model can be formed for predicting the connection between the authors.

3.3. Proposed Work

Our main goal is to predict the coauthor relationship that can take place in the future. To detect the prediction value precisely we are going to implement this section with the help of neo4j and python.

In neo4j we will create nodes and relationships between the attributes which are going to help us predict future co authorship. We will create few relationships for the purpose of making our work easier. We will preprocess the dataset and split the data to make a fair training and testing dataframe. After that we will use machine learning models to find out if our prediction is right or wrong. Then we will fairly compare the results and decide if our model is suitable for predicting links more accurately.

In the traditional approach, we want to develop a relational method for predicting co-authorships. We will handle null values as well as unnecessary columns. Not much like a graph database, we want to create a matrix that can show the relationship between any two authors with the respect of time. Lists, 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, we will split them into train and test for learning our ML models and see how it predicts values with those compared to graph databases.

For both of our approach we are going to use Random Forest Classifier and Logistic Regression Algorithms to find out the accuracy of our model prediction. Our work on this project will help others to show how link prediction works efficiently and how it will be able to conduct in further future to work for more realistic datasets.

3.4 Conclusion

In this research, we will create graph features in graph databases and simulate actual relationships and connections between authors for predicting links. Then in the traditional python method, we will preprocess our data normally using list, matrix, numpy array, dataframe and compare our results with the results of graph database.

4. Dataset

4.1 Introduction

This section introduces the dataset, how the dataset has been extracted, pre-processed and used for the purpose of Future Co-Author Link Prediction.

4.2 Dataset Characteristics

The dataset that has been used here is the Citation Network Dataset[6], which contains information about published Articles of 60 consecutive years from 1958 to 2017. The dataset contains information of Authors who have collaborated and written a paper together, number of times their paper has been cited (n_citation), references used, title of the paper, venue, year of publication, id and abstract.

4.3 Preprocessing

Before starting the work, the dataset needs to go through pre-processing. The current raw dataset needs to be pre-processed into a much cleaner, understandable and simpler form which would enhance the quality of the data, smoothen the dataset, give more accurate and consistent results and reduce redundancies[7]. For the implementation in Neo4j, some constraints need to be created for ensuring data integrity[8].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 NULL values[9].

4.3.1 Preprocessing of Dataset for Neo4j

Before we start working with the data and send them for training and testing, we need to prepare them in a certain way. Initially, constraints are created on Article and Author using Cypher Query Language, which means that there will not be any repeatedness and inconsistencies in the data. After that, when nodes and relationships have been created, and the matching and merging operations are done using Cypher Query Language, we now have a visual representation of the graph of which authors have worked on the same paper and how they are connected.

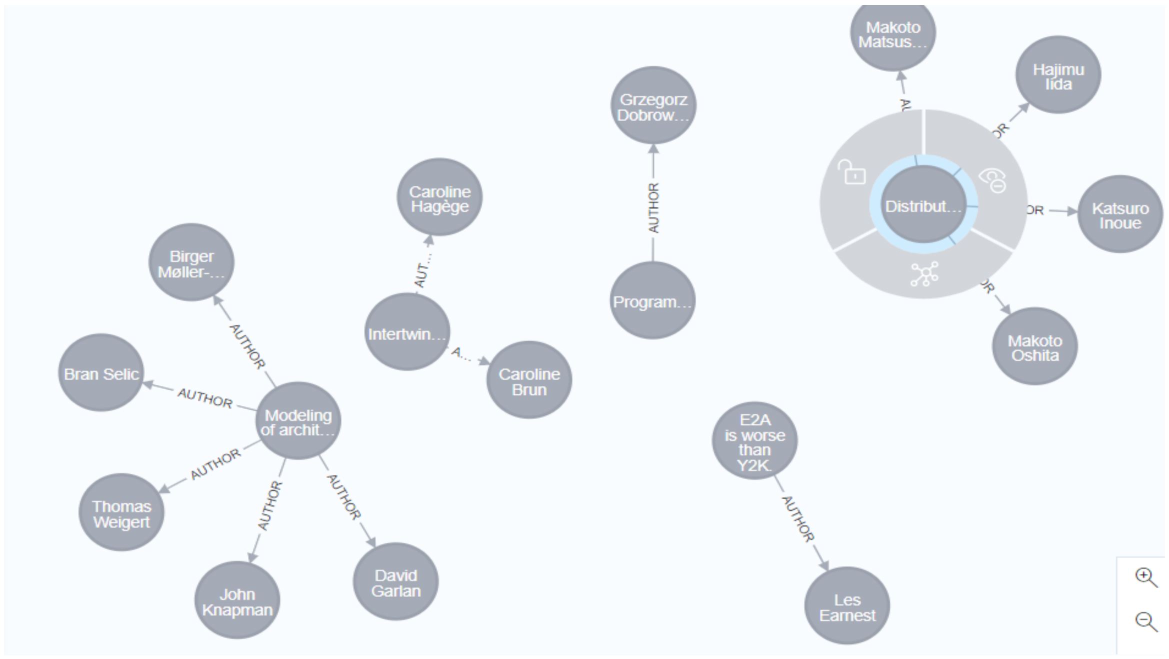


Figure 01: Relationship between Authors and Article

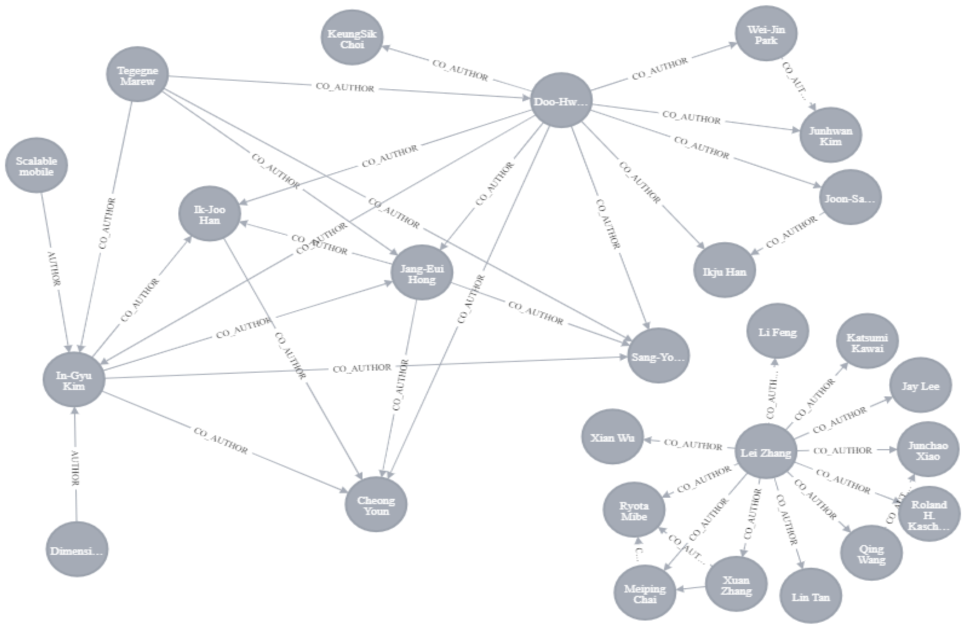


Figure 02: Coauthor Graph

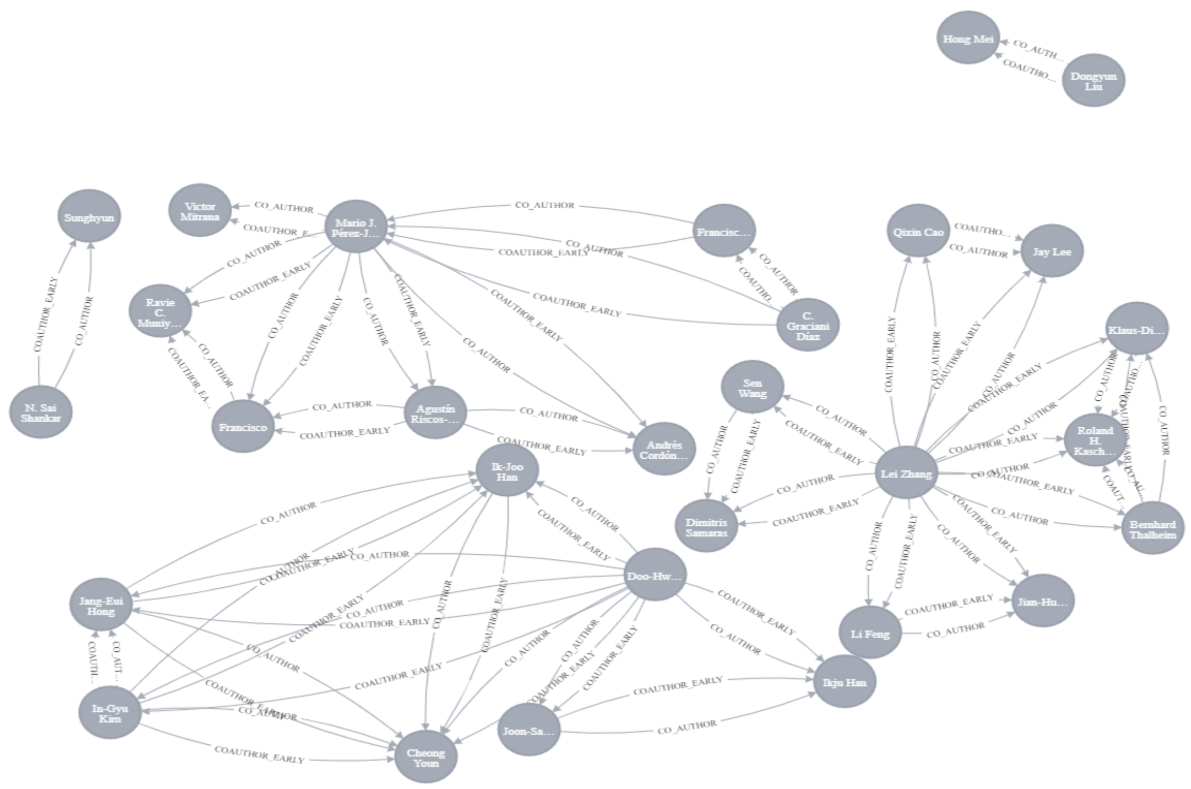


Figure 03: Coauthor Early Graph (collaboration before 2006)

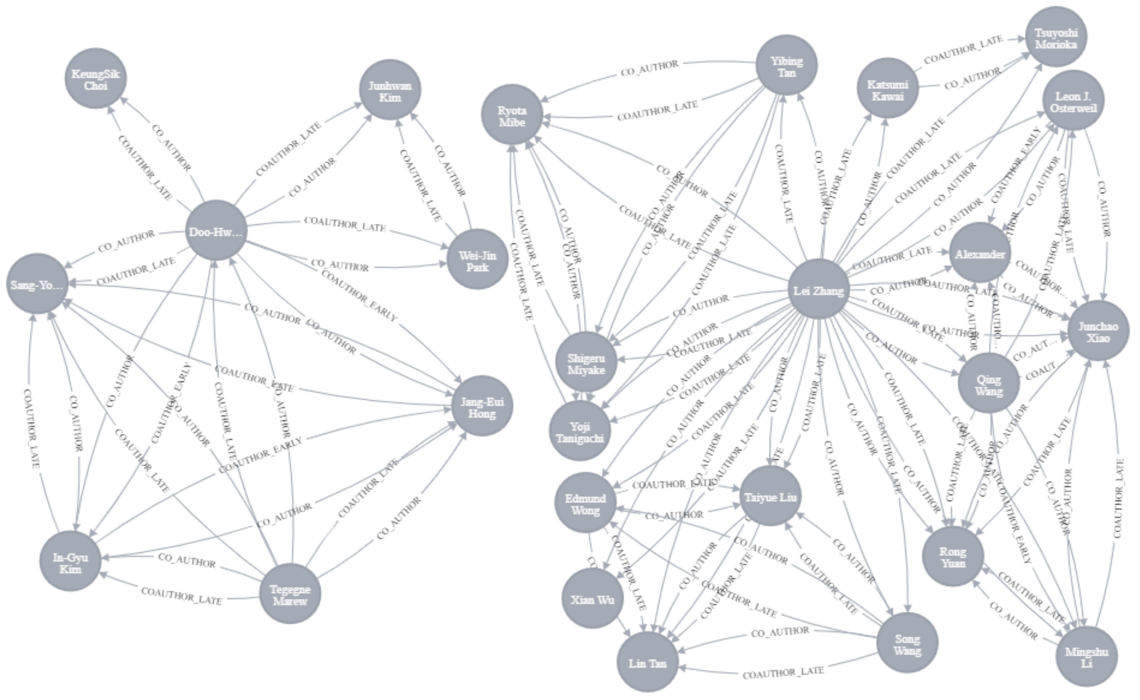


Figure 04: Final Collaboration Graph

Now that we have information about the authors who worked on same paper, this collaboration information is stored in “coauthor.collaboration”, and the dataset is sorted by year in order to do the splitting for training and testing purpose. The highest number of articles were published in the year 2006. So the dataset has been divided in 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 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 with a purpose of testing the model.

Coauthor_Early_Pairs	**Coauthor_Late_Pairs**
81096	**74128**

Table 01: Coauthor Pairs

Now that we have the training and testing dataset, we will keep the pair of authors who have collaborated. The relationships are checked to set the label. If the authors have worked together then the label will be set to 1. However, we also need to find some authors who have not collaborated so that we can train or test with the label value 0. The graph contains more authors who haven’t collaborated together. So, there will be more author pairs who have not collaborated together than the author who have collaborated. Therefore, 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 will check the nearby edges to find out the author pairs who haven’t 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 more than the coauthor pairs(Coauthor_early and coauthor_late in both cases). That’s why we need to balance our training and testing dataset.

Now for the final step of data preprocessing, the dataset is downsampled, and now the dataset is ready to be fit into the model for training and testing.

4.3.2 Preprocessing of Data Set for Traditional File Implementation

In order to work with the dataset 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) has been converted into four CSV files and named the same and then merged together into one single CSV file named 'dblp-ref-small.csv'. After merging the four files into one, the new dblp-ref-small.csv file has 51956 tuples and 8 columns.

Column Name	Data Type
authors	object
n_citations	int64
references	object
title	object
venue	object
year	int64
id	object
abstract	object

Table 02: Datatype of each Attribute

Here, we can see that only two columns are integer type and the rest of the columns are object type. The 'authors' column defines the author collaboration for a specific paper not only one author and the column is not in list format, it is in string format and this should be handled accordingly later. The 'n_citation' means the citation number of each paper. The 'id' column is unique and object typed. The 'references', 'abstract', 'title' columns are for a specific article and object typed. There are only four unique venues and this means where the articles were published. The 'year' column means when the article is published. In our datasets the year is from 1958 to 2017 which is 60 years simulation.

Before starting preprocessing data, we need to check the characteristics of the columns and rows. In preprocessing, null value adaption is a challenge. We have null values in 'references' and 'abstract' columns. All the rest rows and columns do not have any null value.

Column Name	No. of Null Values
authors	0
n_citation	0
references	13430
title	0
venue	0
year	0
id	0
abstract	7642

Table 03: NULL Check

There are 46752 unique author collaborations and 60 unique years (1958-2017). We do not have the unique author list. 50% data is till 2005. There are 30897 papers published after 2006 and 21059 papers published before 2006. There is no correlation between the columns.

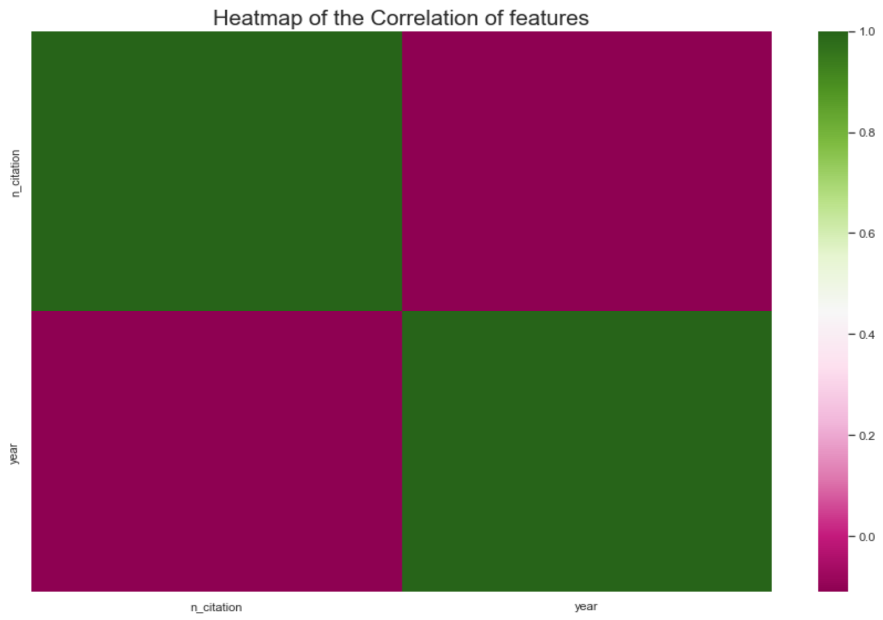


Figure 05: The correlation between n_citation and year.

As we want to predict co-authorship of any two authors and already in graph database (neo4j) method, we predict the coauthor link through the year. In this traditional mechanism, we do not have co-authorship pairs for a year. First, we need to sort our data by year and we can see the unique years. The most of the author collaboration and article publication were in 2006 which is 7536.

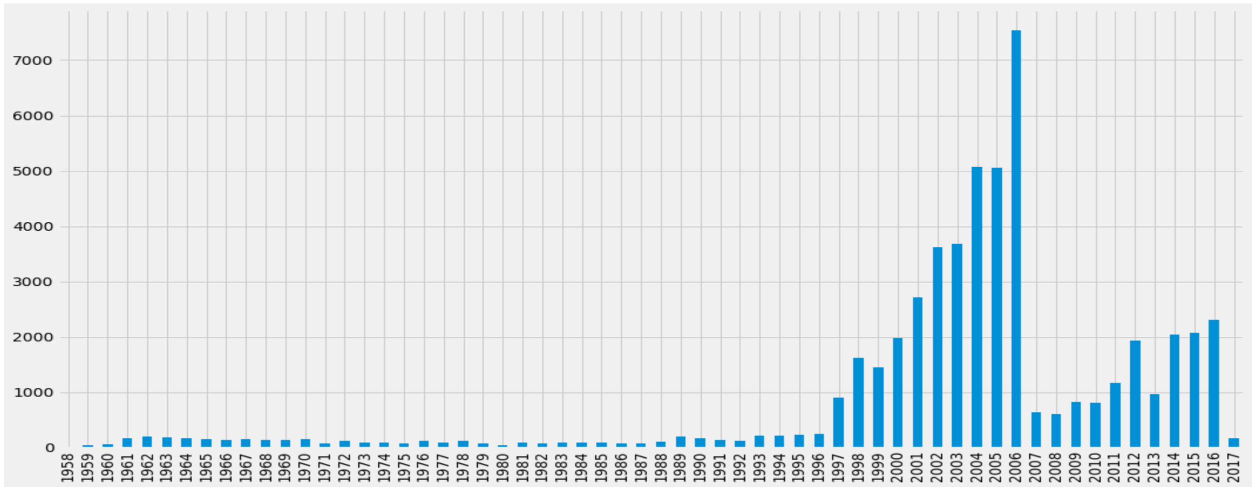


Figure 06: Total Number of Articles Published for each year from 1958 to 2017

Now, for creating the dataset 'venue', 'n_citations', 'references', 'title', 'id', 'abstract' columns are not needed for preprocessing. And with that, the problem of null value adaption is solved. We only need 'authors' and 'year' columns to predict the link between any two authors.

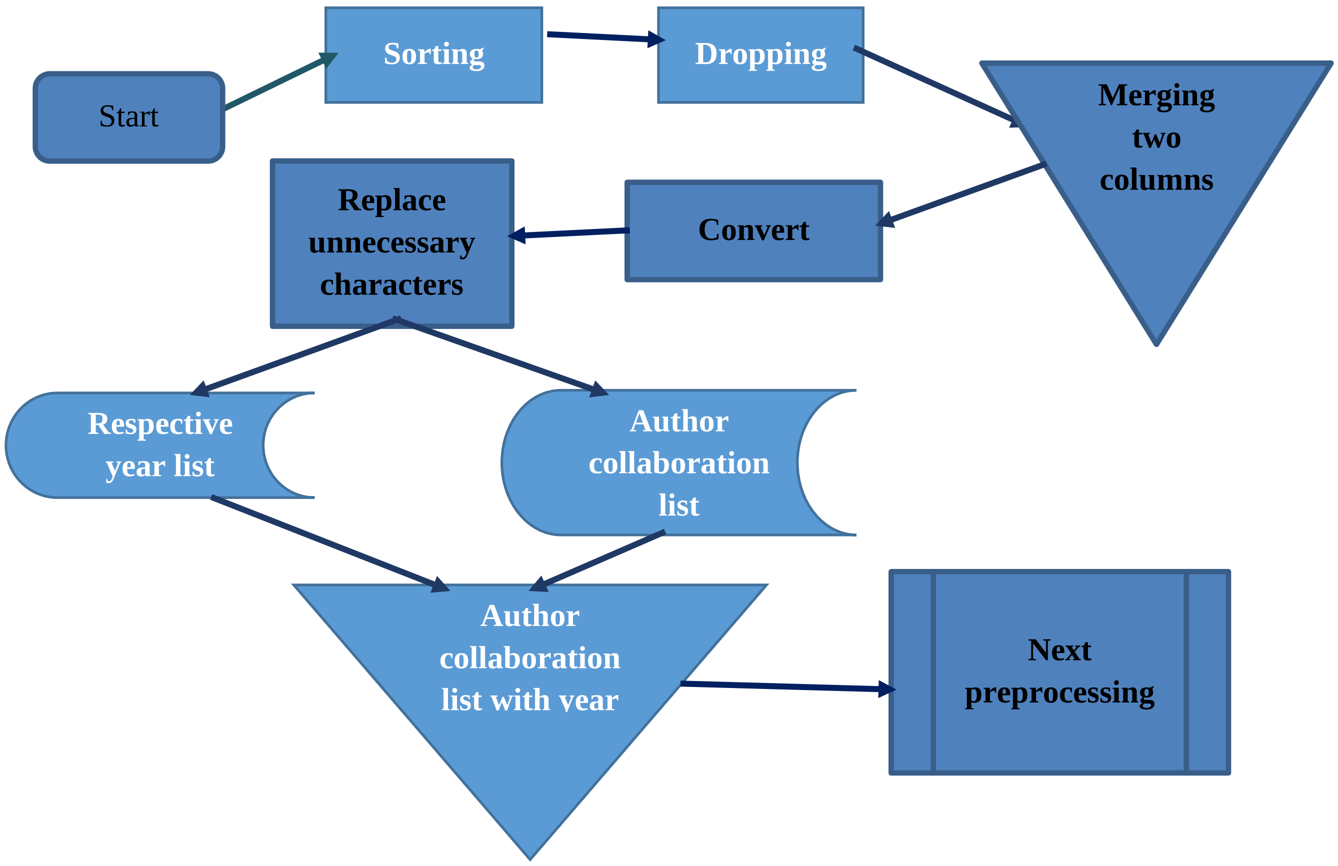

Start
Sorting
Dropping
Merging two columns
Convert
Replace unnecessary characters
Respective year list
Author collaboration list
Author collaboration list with year
Next preprocessing

Figure 07: Author Collaboration with year preprocessing

We need to create an array or list that contains author collaboration ship with respective year that can access with list indexing easily. We need ‘authors’ and ‘year’ columns for storing in a numpy array called ‘listofayer’ but from the list the author name is not easily accessible. All the names are in string form and cannot access a whole name with indexing. So, we have to make them accessible within a year. For that, we need to replace all the unnecessary characters such as space,third bracket, quotation marks with null characters. After that we will split the names with commas and put them in a list. Also we need another list for keeping respective collaboration years. The indexing is mappe and puts collaboration authors in a list called ‘bll1’ and their collaboration year is in a list called ‘bll2’. The both lists' lengths are the same as no value is missing. Then we append this list by the same mapping and we call the list as ‘bll1’ AKA author collaboration list with its respective year. And now each and every value or name as well as year is accessible. ‘Bll1’ is a list of lists and in every nested list the last position has the year and the rest position is co-authorship between one, two or many authors. Also the list is not unique because we need all collaborations to check. Now we need to find the unique authors name and make a list.

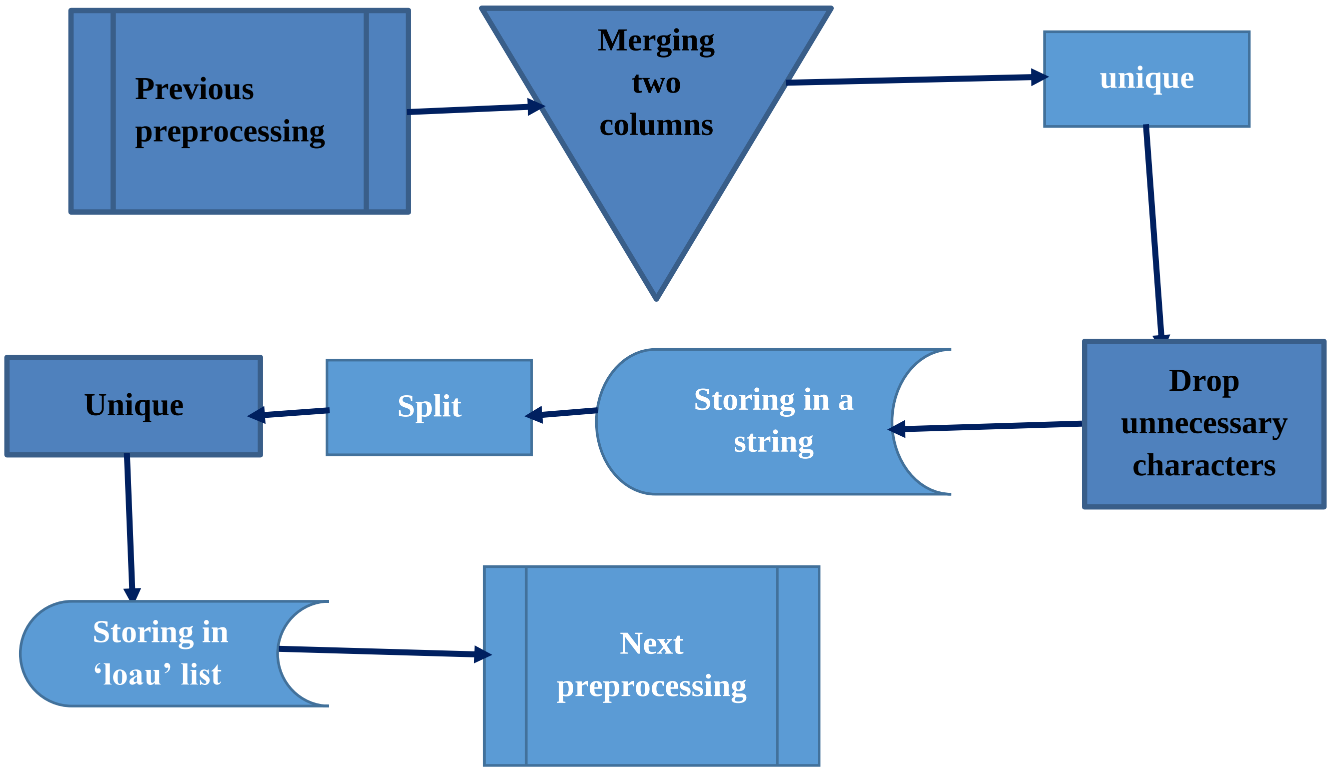


Figure 08: Unique Author List Preprocessing

For this, we have to create a unique author collaboration and put them in the 'listOfAuthors' list. As 'listofAuthors' is a string list and the authors name is not accessible, so again we have to remove all the unnecessary characters. We removed and put all the names which can be repeated into a long string called 'auth_str'. In 'auth_str' , the names of authors are in comma separation. We have to split them. After splitting, we find 132462 author names but 132462 is not the number of unique authors. We have to uniquely identify them. After uniquely identifying them, we found 80290 unique authors and put them in a list called 'loau' and the index value of each author will be known as each author's unique id. Now, we have to create a 2D matrix with 4 columns named 'author_1', 'author_2' 'collab_label' and 'year'. And store the collaboration pair with its according year.

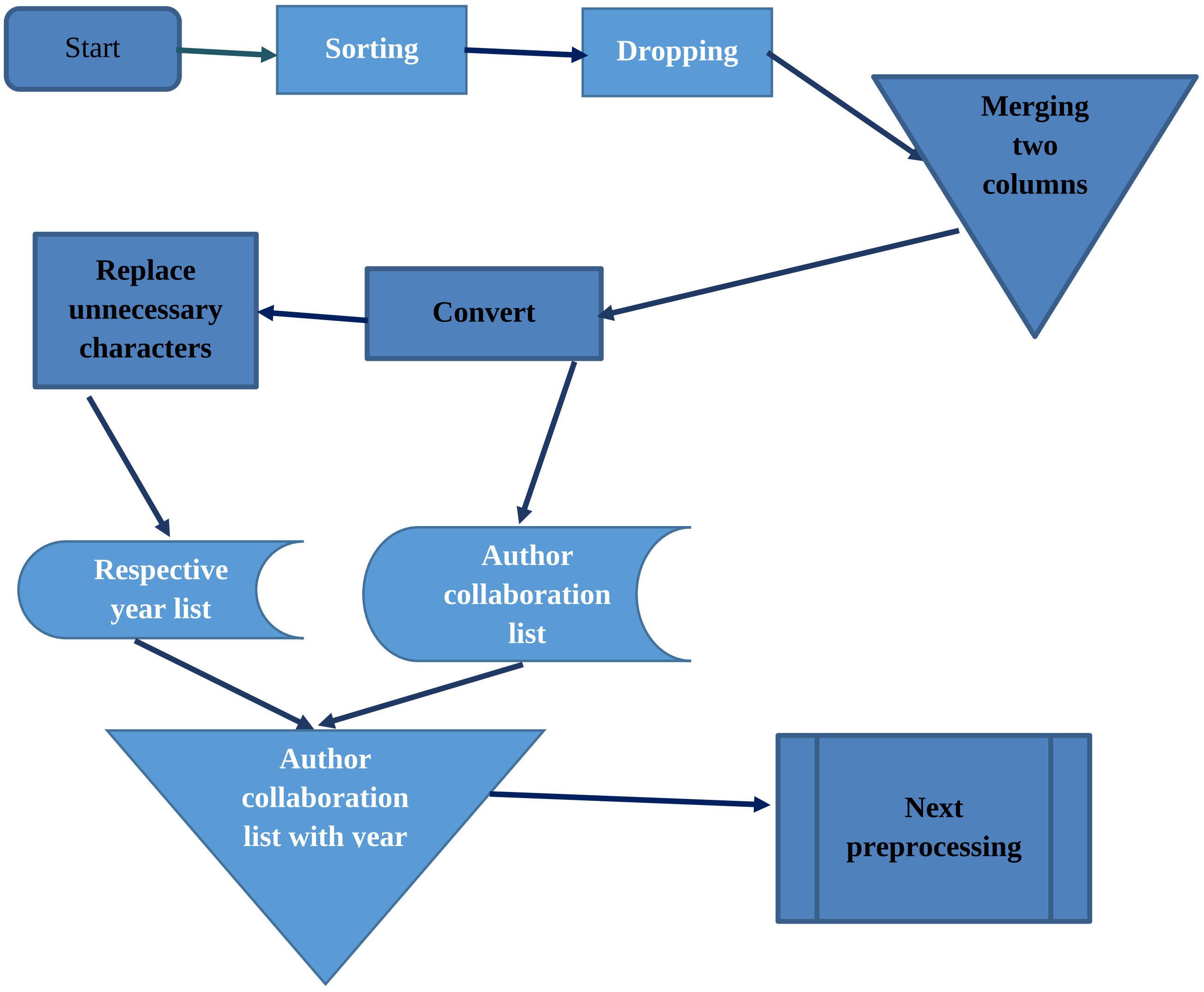

Start
Sorting
Dropping
Merging two columns
Replace unnecessary characters
Convert
Respective year list
Author collaboration list
Author collaboration list with year
Next preprocessing

Figure 09: Author Collaboration data Pre-processing

We will store the author index value which is unique for each author from the 'loau' list. We will find collaboration for one author from 'bll1' and store his/her coauthor (accepts his/her own name) with their collaboration year and put 1 in as collab_label. For each value or author of ‘loau’ we checked the collaboration list ‘bll1’ that collabed with other authors and stored their names and then found the indexing in ‘loau’ and appended them with their respective year with collab_label ‘1’.

We stored the values in a list of lists which is converted into a numpy array later. This process takes a while to fully conduct. So we store the collaboration in a separate csv file for further execution without any delay. So, 'author_collab_JN_unique.csv' is the final csv file that we can directly fit into a ML model. But in this file there is only one class label which is 1 and 1 means 'they collabed' but we got no zero values. We do not want to overfeed our ML models and because of that our ML model can predict wrong but it will show 100% accuracy. So, we have to store some zero values into another file which is called a non-collaboration array list.

To make that list, we will go 1..2 hop from the co-author index of its author and check that the author was collabed within those ranges or not. This psychology is kind of similar to our graph database method. If it finds that the author did collaborate from its initial co-authors 1..2 hop range it will not store the value because it is already stored in the collaboration file. And if it did not find any then it will store the author index with its non-coauthor index alongside the collab_label as '0' and null string for a year. The 'author_non_collab_JN_unique.csv' file is for non collaboration. And now with these two files we will learn them from different ML models. The rest of the work is in a separate ipynb file.

5. Methodology

5.1 Introduction

In this section, we are going to discuss the two methods that have been used to implement the entire project.

5.2 Graph Database: Neo4j: Neo4j is a database that is developed in Java and can be used as an embedded or server database. Neo4j implements the graph data model, which consists of fundamental elements, nodes, and their relationships. It uses the native property graph model as its data model. 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. All the ACID properties are supported by Neo4j. Moreover, it comes with Cypher, a strong declarative query language. Graphs are shown using ASCII art. Cypher is simple to learn and may be used to define and retrieve data relationships without the usage of sophisticated queries such as Joins. When opposed to other databases, Neo4j allows you to not only describe but also conveniently retrieve (traverse/navigate) related data.

5.3 Traditional File: Traditional python being the most widely used language for Machine Learning and Predictive Analysis leaves a good scope for researching about the loop-holes that exist in the system and how they can be changed, or improved for the betterment. It provides the opportunity to compare and contrast the performance of different methods that are being used, and the algorithms performing the actual work.

5.4 Applied Algorithms:

For predicting the possibility whether or not authors will collaborate in the future, the two machine learning algorithms that we have implemented in our work are Random Forest Classifier and Logistic Regression.

5.4.1 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 dataset's predicted accuracy[10]. The random forest classifier is made up of a number of tree classifiers, each of which is constructed using a random vector sampled separately from the input vector, and each tree casts a unit vote for the most popular class in order to classify an input vector. To build a tree, the random forest classifier utilized in this study uses randomly selected characteristics or a combination of features at each node. For each feature/feature combination defined, bagging, a method of generating a training dataset by randomly drawing with replacement N samples, where N is the size of the original training set, was employed. The most frequent voted class from all the tree predictors in the forest is used to classify any samples (pixels).

5.4.2 Logistic Regression

Logistic regression is a supervised Machine Learning algorithm and a statistical analysis method which is used for binary classification problems to predict a binary outcome[11]. This model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. Logistic regression provides a discrete output and the range is bounded between 0 and 1.

It is a statistical analysis for predicting binary values. It predicts the dependent features with the class label but the relationship between the features should be independent. It can take multiple inputs. This model or algorithm works better as predicting according to the datasets. It is also used for data preparation. The model is also used for transforming raw data and creates dataframe or features for other ML models. It is also used for measuring the probability. The sigmoid function is called the logistic function and it is used in the algorithm. This model has the ability to transform complex measurement into simple straightforward problems. [10]. There are many parameters in this model such as dual, tol, fit_intercept, random_state, solver, max_iter, multi_class etc.

Solver is the main algorithm for optimizing the problem. **‘**newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’ are the solving algorithms, but by default it will use ‘lbfgs’. **Max_iter** is int type and used for defining the maximum number of iterations.

5.5 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.

Common Neighbors: The term "common neighbors" refers to the fact that two strangers who share a mutual friend are more likely to be introduced than those who do not[12]. In our project,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.

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

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.
Preferential Attachment: Preferential attachment states that new writers are more likely to attach to older authors who already have a large network of contacts (collaborators)[13]. Potential attachment assigns a score to each pair of authors based on the number of coauthors they each have. Authors are more inclined to work with someone who has already co-authored a number of publications, according to the theory.

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

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

Total neighbor: Total Neighbors calculates a node's proximity based on the number of unique neighbors it has[14]. The premise behind it is that the more linked a node is, the more probable it is to get new linkages.

$$TN(x,y) = |N(x) \cup N(y)|$$

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.

5.6 Model Evaluation Properties

Precision: Precision is a statistic that measures how many correct positive forecasts have been made[15]. Precision calculates the minority class's accuracy. The ratio of accurately predicted positive instances divided by the total number of positive examples anticipated is used to compute it. The result value is between 0-1.

$$\text{Precision} = TP / (TP + FP)$$

Recall: Recall is a statistic that measures how many correct positive predictions were produced out of all possible positive predictions[15]. Unlike precision, which only considers the right positive predictions out of all positive predictions, recall considers the positive predictions that were missed. In this approach, recall offers some indication of the positive class's coverage. The result value is between 0-1.

$$\text{Recall} = TP / (TP+FN)$$

F1 Score: The F1-score takes the harmonic mean of a classifier's accuracy and recall to create a single statistic. It's mostly used to compare the results of two different classifiers[15]. When the F1 score is 1, the model is deemed perfect, but when it is 0, the model is considered a complete failure.

$$\text{F1 Score} = 2 \times \text{Precision} * \text{Recall} / \text{Precision} + \text{Recall}$$

Confusion Matrix: An N x N matrix is used to evaluate the performance of a classification model, where N is the number of target classes[16]. The matrix compares the actual goal values to the machine learning model's predictions.

	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Table 04: Confusion Matrix

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

ROC Curve: The Receiver Operating Characteristics Curve, or ROC curve, is a statistic for evaluating the performance of a classifier model[17]. The ROC curve illustrates the rate of true positives in relation to the rate of false positives, emphasizing the classifier model's sensitivity. The ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. 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.

6. Implementation

6.1 Implementation Environment:

For Graph Database Neo4j 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 Future Co-Authorship Prediction in Neo4j is because of the interface and facilities that it provides. Creating nodes, properties, relationships, 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 total 6 link prediction algorithms and we are going to apply 3 algorithms for the purpose of our prediction[18]. After pre-processing 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[19].

6.2.1 Random Forest Implementation

The result of common neighbor, preferential attachment and total neighbor for both train and test dataset is shown in the table below:

	node1	node2	label	cn	pa	tn
0	13058	120540	1	2.0	24.0	9.0
1	16173	43949	1	1.0	42.0	12.0
2	16650	16651	1	2.0	81.0	28.0
3	23590	71500	0	1.0	63.0	15.0
4	31072	31072	0	0.0	9.0	3.0

Table 05: Training Dataset for Random Forest

	node1	node2	label	cn	pa	tn
0	27545	75873	0	0.0	96.0	20.0
1	28106	77951	0	0.0	15.0	8.0
2	50936	115653	1	5.0	84.0	15.0
3	11785	11788	1	1.0	12.0	6.0
4	73129	11932	0	0.0	3.0	4.0

Table 06: Testing Dataset for Random Forest

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 dataset COAUTHOR_LATE for prediction.

6.2.2 Logistic Regression Implementation

The implementation of the Logistic Regression Algorithm follows an approach similar to that of the Random Forest Algorithm. The result of common neighbor, preferential attachment and total neighbor for both train and test dataset is given in the table below:

	node1	node2	label	cn	pa	tn
0	15570	43542	0	0.0	68.0	21.0
1	52852	100706	0	1.0	20.0	8.0
2	10959	10961	1	1.0	4.0	3.0
3	25562	25563	1	3.0	16.0	5.0
4	72820	72821	1	1.0	4.0	3.0

Table 07: Training Dataset for Logistic Regression

	node1	node2	label	cn	pa	tn
0	118412	53349	0	1.0	10.0	6.0
1	81239	71969	0	0.0	76.0	23.0
2	100859	70321	0	1.0	27.0	11.0
3	70055	71439	1	3.0	414.0	52.0
4	16783	29439	1	5.0	160.0	23.0

Table 08: Testing Dataset for Logistic Regression

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.

6.3 Traditional File Implementation

For learning our data into our models, we make the two datasets or data frames similar. There is null value in the non-collaboration dataset in the 'year' column. We replaced those with zeros. Now, the both dataframe files are the same type and ready to be merged. At first, we sorted our collaboration dataframe by year for splitting. We have split 60% train data and 40% test data from the files. In neo4j we split the data by the following: the year after 2006 is for test data and all the rest is for training data. As we need in neo4j test data is from 2006 to 2017 and train data goes from 1958 to 2005. The main dataset length is 341038. The train dataset and test dataset are 175650 and 165388 respectively. Here the splitting percentage is 52% training and 48% testing which is not done in our graph database approach. And if we only keep collaboration data the ML model will overfeed so we have to split non-collaboration data in a way that the total train test split will become 60% and 40% according to our graph database approach. If we split our non-collaboration data by 67.23% train size and then merge with collaboration train and test. As non-collaboration has no specific year value, we split them randomly. Now we merge all the train data of collaboration and non-collaboration into one and do the same for test data. Now our datasets are ready to be fit in machine learning models.

For the implementation in the conventional python approach, the previously pre-processed train and test data files are simply loaded and fitted into the machine learning algorithm: random forest and logistic regression. Here, after all the preprocessing, we marged the collaboration and non-collaboration files and split them into a train set which is like COAUTHOR_EARLY and a test set which is like COAUTHOR_LATE. And then the datasets are directly fit into our ML models. As the datasets are huge, only 5 rows are shown for better understanding.

	author_1	author_2	label
12	3	4	1
58	16	15	1
30	9	10	1
300552	59750	12178	0

Table 09: Training Dataset

	author_1	author_2	label
222550	48616	48614	1
219266	47736	7101	1
213701	46179	49387	1
169221	30950	37965	0

Table 10: Test Dataset

6.4 Conclusion

After the implementation has been done, the data and the prediction result is then further analyzed to see their performance through their accuracy, precision, recall and F1-score.

7. Result and Analysis

7.1 Introduction

In this section we are going to see the output we get by applying the algorithms in both Neo4j Approach and Traditional Python Approach. The results will give us a clear overview of which algorithm performs better in different situations.

7.2 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 the value 30), max_depth (which means the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure. We set the value of max_depth as 10. So the maximum depth is ten for each tree) and random_state (the value we have set is 0, it means when a tree is being built, there should be no randomness of the bootstrapping of the samples.)[20]. In logistic regression, we have used two parameters(solver='saga', max_iter=50). Here, solver means a solving algorithm for optimizing the problem. ‘Saga’ is a solving method and max_iter means maximum number of iterations taken for the solvers to converge. Here it will take 50 iterations to converge[21].

In the traditional python scikit learn approach, we used random forest but different parameters and values. The n_estimators is set on 100 and it 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 and it means there should be four jobs running in parallels[20]. We also used a logistic regression machine learning model, and the parameter is solver=’newton-cg’[21].

7.3 Comparative analysis of the performance of both algorithms

7.3.1 Neo4j

In neo4j, we have implemented both random forest classifier and logistic regression. In the first part the training and testing dataset were created for 2 to 3 hops of the existing co authorship early and late graphs. In the next part training and testing dataset were created for 3 to 4 hops nearly edges between the co authorship early and late graph.

7.3.2 Traditional Approach

We have also implemented both random forest classifier and logistic regression machine learning algorithm for better comparison. In the preprocessed we splitted 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 splitted and merged with collaboration testing and training data. And then fit into both ML algorithm models.

7.4 Evaluating Models

7.4.1 Neo4j

In neo4j random forest classifiers performed better than logistic Regression. We can see that through the following tables and graphs.

Precision, Recall and F-1 score:

A model that produces a higher precision score generates less false positives.

A model that produces a higher recall score generates less false negatives.

A model that produces a higher F-1 score indicates more perfect precision and recall.

Random Forest Classifier:

Here for 3..4 hops we are getting better results.

	Precision	**Recall**	**F-1 Score**
2..3 hop	**0.93**	**0.90**	**0.91**
3..4 hop	**0.97**	**0.91**	**0.94**

Table 11: Precision, Recall, F-1 Score for Random Forest Classifier in neo4j

Logistic Regression:

3..4 hops is giving better results in this case too.

	Precision	**Recall**	**F-1 Score**
2..3 hop	**0.91**	**0.79**	**0.85**
3..4 hop	**0.96**	**0.81**	**0.88**

Table 12: Precision, Recall, F-1 Score for Logistic Regression in neo4j

Confusion matrix:

Random Forest Classifier:

The results we got for both 2..3 hops and 3..4 hops are given in the table also the confusion matrix visual representation is portrayed below:

	TN	**TP**	**FN**	**FP**
2..3 hop	**68730**	**66694**	**7434**	**5398**
3..4 hop	**71913**	**67675**	**6453**	**2215**

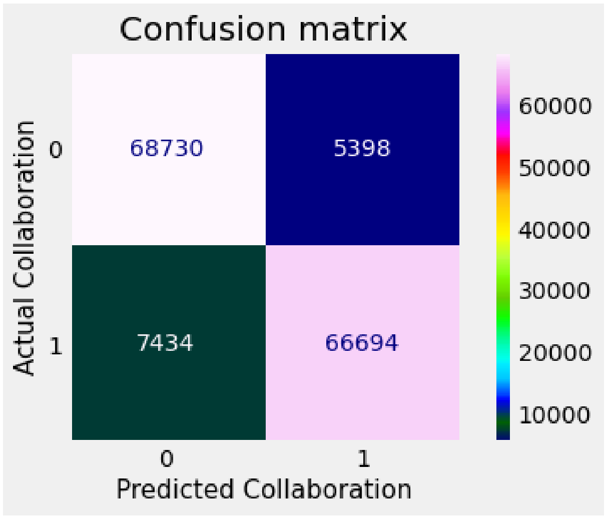


Figure 10: Confusion Matrix for 2..3 hops

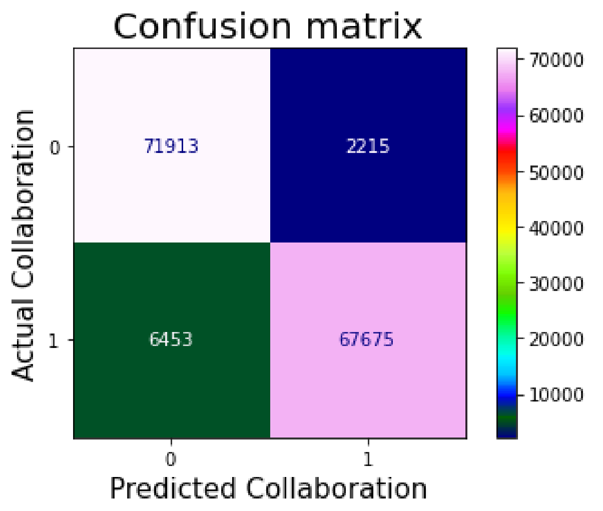


Figure 11: Confusion Matrix for 3..4 hops

Table 13: Confusion Matrix representation for Random Forest Classifier.

Logistic Regression:

In the given table we have presented the confusion matrix results and the visual representation of the plotting for both 2..3 hops and 3..4 hops.

	TN	**TP**	**FN**	**FP**
2..3 hop	**68136**	**58846**	**15282**	**5992**
3..4 hop	**71856**	**60298**	**13830**	**2272**

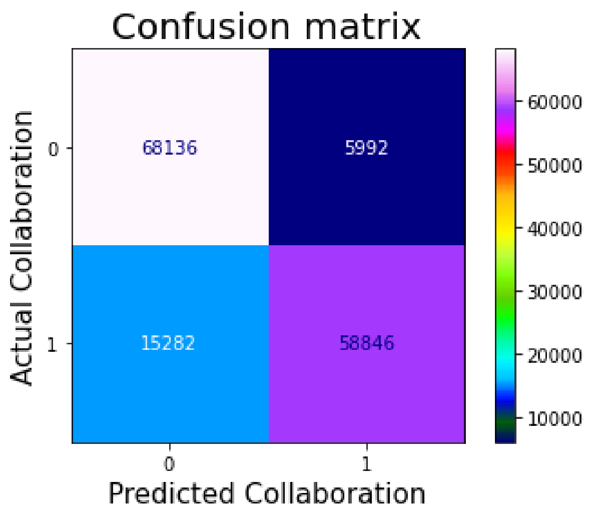
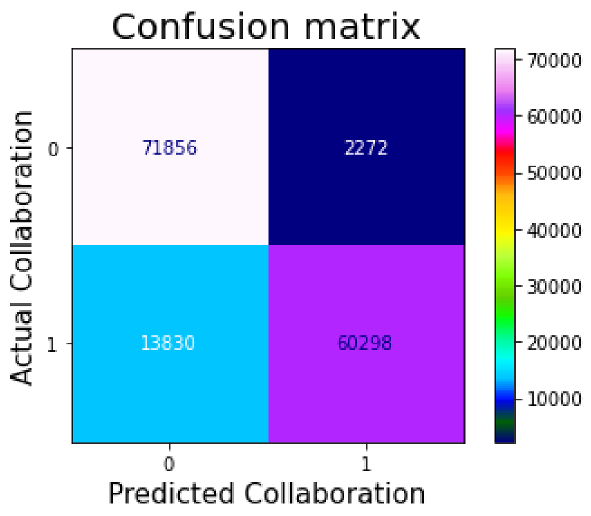
 Figure 12: Confusion Matrix for 2..3 hops(Logistic Regression)	 Figure 13: Confusion Matrix for 3..4 hops(Logistic Regression)

Table 14: Confusion Matrix representation for Logistic Regression.

ROC Curve:

From the Roc curve given in the following table we can state that, for 3..4 hops the model is performing better.

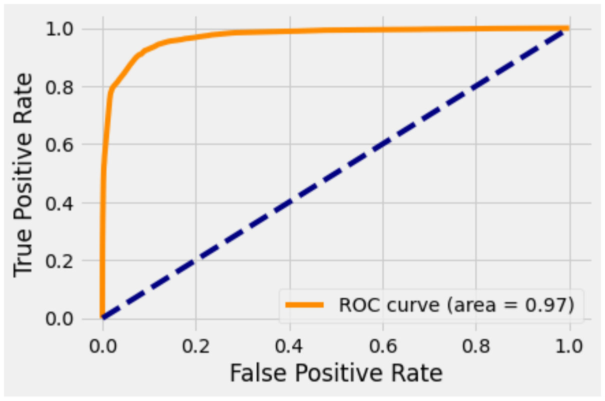
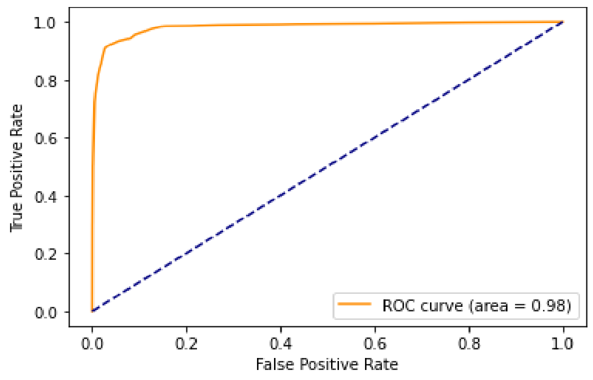
2..3 hops(Random Forest Classifier)	**3..4 hops(Random Forest Classifier)**
0.97	**0.98**
 Figure 14: ROC Curve for 2..3 hops	 Figure 15: ROC Curve for 3..4 hops

Table 15: ROC Curve for Random Forest Classifier.

For the graph given below, our model is also predicting better for 3..4 hops.

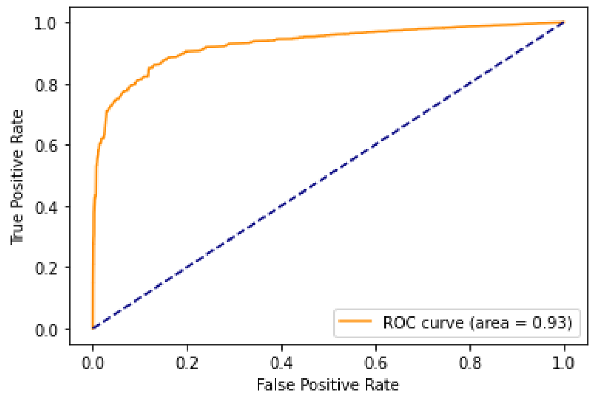
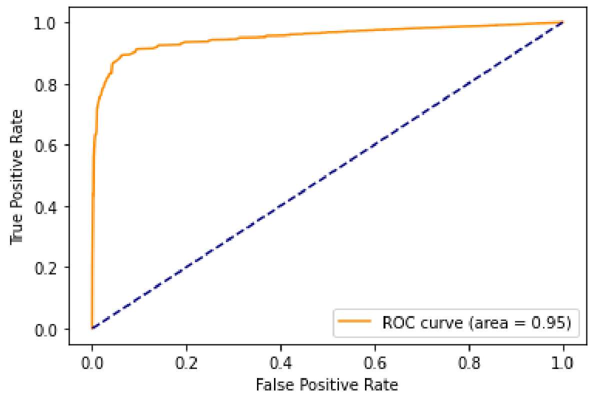
2..3 hops(Logistic Regression)	**3..4 hops(Logistic Regression)**
0.93	**0.95**
 Figure 16: ROC Curve for 2..3 hops.	 Figure 17: ROC Curve for 3..4 hops.

Table 16: ROC Curve for Logistic Regression.

7.4.2 Traditional Approach Analysis

For the Traditional python approach, Random forest classifier performed slightly better than logistic Regression. We can see that from the following 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 case of Random Forest Classifier which does not really make any sufficient changes.

	Precision	**Recall**	**F1 Score**
Random Forest Classifier	**0.05**	**0.21**	**0.08**
Logistic Regression	**0.05**	**0.20**	**0.08**

Table 17: Precision, Recall, F-1 Score for Random Forest Classifier and Logistic Regression for traditional approach.

Confusion Matrix:

The representation of Confusion Matrix for both models applied in our traditional approach.

	TN	**TP**	**FN**	**FP**
Random Forest Classifier	**98475**	**8562**	**32847**	**156826**
Logistic Regression	**98215**	**8288**	**33107**	**157100**

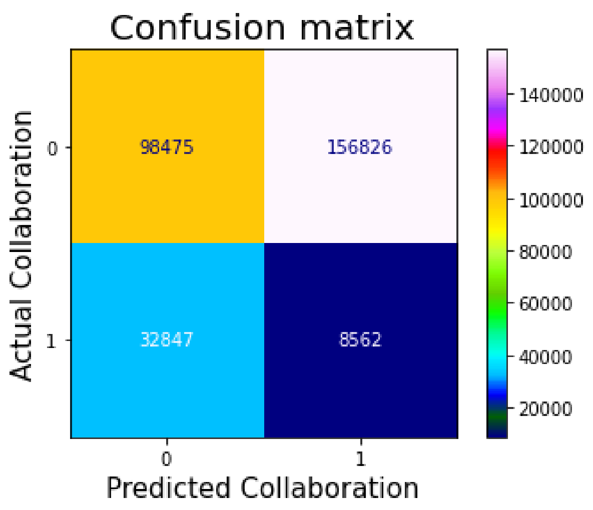


Figure 18: Confusion matrix of Random Forest Classifier.

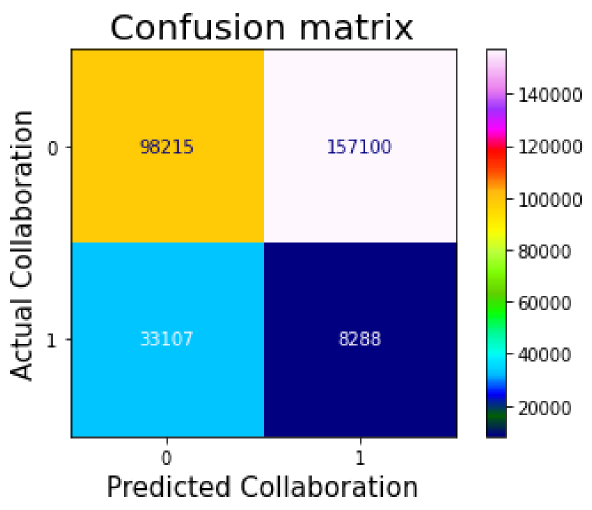


Figure 19: Confusion matrix of Logistic Regression.

Table 18: Confusion Matrix representation for Random Forest Classifier and Logistic Regression.

ROC curve:

From the Roc curve given in the following table we can see that none of the model is not giving us a satisfactory result.

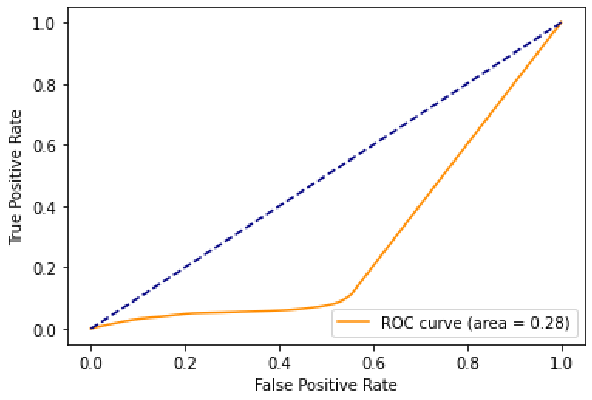
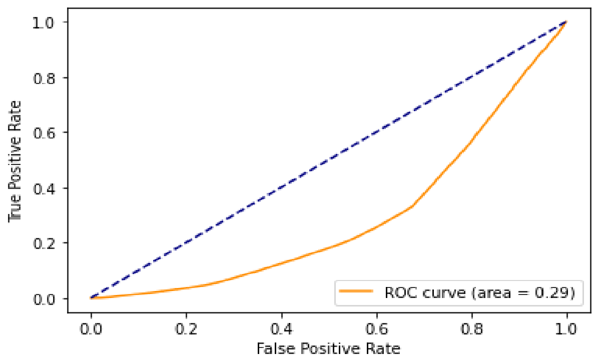
Random Forest Classifier	Logistic Regression
0.28	**0.29**
 Figure 20: ROC Curve for Random Forest Classifier	 Figure 21: ROC Curve for Logistic Regression

Table 19: ROC Curve for Random Forest Classifier and Logistic Regression.

7.5 Classification Accuracy

7.5.1. Neo4j

For 3..4 hops Random Forest Classifier is giving us a better accuracy.

2..3 hops(Random Forest Classifier)	3..4 hops(Random Forest Classifier)
0.913447	**0.941534**

Table 20: Accuracy for Random Forest Classifier.

Also for 3..4 hops, Logistic Regression is giving us a better accuracy.

2..3 hops(Logistic Regression)	**3..4 hops(Logistic Regression)**
0.85650	**0.891391**

Table 21: Accuracy for Logistic Regression.

7.5.2 Traditional File

Accuracy we got for both our models.

Random Forest Classifier	**Logistic Regression**
0.3607462	**0.3589464**

Table 22: Accuracy for Random Forest Classifier and Logistic Regression.

7.6 Conclusion

We have performed both Random Forest Classifier and Logistic Regression for both of our approaches (Graph database and traditional approach). In neo4j, Random Forest Classifier gave better accuracy than the Logistic Regression algorithm. In the traditional procedure, both algorithms did not provide satisfactory accuracy. However, in the traditional approach, Random Forest Classifier also worked slightly better than Logistic Regression. Therefore, The Random Forest Classifier worked in our favor.

8. Conclusion and Future Work

8.1 Conclusion

From the above discussion, we can clearly state that in case of coauthor link prediction, 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 access different types of 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 formed for 2 to 3 hops. For 3 to 4 hops it gave better accuracy because the model was trained with more accurate data.

In the link prediction mechanism, Graph Database works more efficiently and fast. We have preprocessed our CSV file according to the psychology of neo4j graph database method. Then author index values are basically represented as nodes but our ML model in traditional method finds them as normal nominal values.

So, in Traditional Approach, for link prediction 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 do some long term preprocess to find out the collaborations and unique authors. The most challenging part was to find coauthors pair list with year as well as to find some non collaboration pairs which was much easier in neo4j graph database and it took a little while to find out all the relationships and uniqueness.

8.2 Future Work

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

We have only used ‘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. We plan to consider ‘venue’ as an important feature to figure out the connection of co authorships.

References:

[1] "What is Graph Database." https://neo4j.com/why-graph-databases/ (accessed Feb. 06, 2022).

[2] "Graph Database." https://www.javatpoint.com/what-is-graphdb (accessed Feb. 01, 2022).

[3] P. M. Chuan, L. H. Son, M. Ali, T. D. Khang, L. T. Huong, and N. Dey, "Link prediction in co-authorship networks based on hybrid content similarity metric," *Appl. Intell.*, vol. 48, no. 8, 2018, doi: 10.1007/s10489-017-1086-x.

[4] S. Cohen and N. Cohen-Tzemach, "Implementing link-prediction for social networks in a database system," 2013, doi: 10.1145/2484702.2484710.

[5] T. Wang and G. Liao, "A review of link prediction in social networks," *Proc. - 2014 Int. Conf. Manag. e-Commerce e-Government, ICMeCG 2014*, pp. 147–150, 2014, doi: 10.1109/ICMeCG.2014.38.

[6] "Dataset." https://github.com/mneedham/link-prediction/tree/master/data (accessed Nov. 08, 2021).

[7] "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 Feb. 05, 2022).

[8] "Constraints." https://neo4j.com/docs/developer-manual/current/cypher/schema/constraints/ (accessed Feb. 04, 2022).

[9] "Data Preprocessing in Python." https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/ (accessed Feb. 07, 2022).

[10] "Understanding Random Forest." https://towardsdatascience.com/understanding-random-forest-58381e0602d2#:~:text=The random forest is a,that of any individual tree (accessed Feb. 06, 2022).

[11] "Logistic Regression." https://searchbusinessanalytics.techtarget.com/definition/logistic-regression (accessed Feb. 06, 2022).

[12] Z. Yang, R. Hu, and R. Zhang, "An improved link prediction algorithm based on common neighbors index with community membership information," *Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS,* vol. 0, pp. 90–93, 2016, doi: 10.1109/ICSESS.2016.7883022.

[13] S. Ruj and A. Pal, "Preferential attachment model with degree bound and its application to key predistribution in WSN," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, vol. 2016-May, pp. 677–683, 2016, doi: 10.1109/AINA.2016.141.

[14] B. Qiu, J. Wang, Y. Liu, and Z. Xu, "Neighbor sum distinguishing total colorings of graphs with bounded maximum degree and maximum average degree," *Proc. - 2017 IEEE Int. Conf. Comput. Sci. Eng. IEEE/IFIP Int. Conf. Embed. Ubiquitous Comput. CSE EUC 2017*, vol. 1, pp. 898–901, 2017, doi: 10.1109/CSE-EUC.2017.182.

[15] "Precision Recall F1 score." https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/ (accessed Feb. 09, 2022).

[16] "Confusion Matrix." https://www.google.com/url?sa=j&url=https%3A%2F%2Fmedium.com%2Fanalytics-vidhya%2Fconfusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd&uct=1632834385&usg=vFtIgqwBDp_RGuL-nrVLZETDpGE.&source=meet (accessed Feb. 08, 2022).

[17] "ROC Curve." https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-

learning/ (accessed Feb. 09, 2022).
[18] "Neo4j Co-authorship." https://towardsdatascience.com/link-prediction-with-neo4j-part-2-predicting-co-authors-using-scikit-learn-78b42356b44c (accessed Nov. 09, 2021).
[19] M. A. AMANU, *Graph Algorithms (Example in Spark & Neo4j)*. 2015.
[20] "Random Forest Parameter." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (accessed Feb. 09, 2022).
[21] "Logistic Parameter." https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (accessed Feb. 09, 2022).