Link Prediction in Medical Domain using Knowledge Graphs

Dissertation as a Course requirement for

MSc Data Science and Computing

GORTHIPALLI AKHIL SAI

20229

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

Department of Mathematics and Computer Science

Muddenahalli Campus

April 2022

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

Department of Mathematics and Computer Science
Muddenahalli Campus

CERTIFICATE

This is to certify that this Dissertation titled **Link Prediction in Medical Domain using Knowledge Graphs** submitted by Gorthipalli Akhil Sai, Regd No.20229, Department of Mathematics and Computer Science, Muddenahalli Campus, during the academic year 2021-22, is a bonafide record of the original work done under our supervision as a Course requirement for the Degree of Master of Science Data Science and Computing.

Countersigned by

P. Sunil Kumar

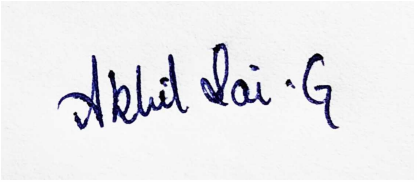
Sri. P Sunil Kumar
Supervisor

Dr. Rita Gupta
Head of the Department

Place: Muddenahalli
Date: 25th April 2022

DECLARATION

The Dissertation titled **Link Prediction in Medical Domain using Knowledge Graphs** was carried out by me under the supervision of Sri. P Sunil Kumar, Department of Mathematics and Computer Science, Muddenahalli Campus as a Course requirement for the Degree of Master of Science Data Science and Computing and has not formed the basis for the award of any degree, diploma or any other such title by this or any other University.



Place: Muddenahalli
Date: 25th April 2022

Gorthipalli Akhil Sai
20229
II MSc Data Science and Computing
Muddenahalli

ACKNOWLEDGMENTS

I am overwhelmed in all gratefulness and humbleness to acknowledge my deep and sincere gratitude to all those who helped me to shape this project.

It would have been impossible to achieve this project over a year without the backing of the elders and their guidance. I will be failing in my duty if I don't acknowledge their support throughout my continuous struggle, rise and fall to complete this project.

I bow down in unfathomable gratitude to Bhagawan Sri Sathya Sai Baba, for if not His will, I would not have been part of this project itself. I am also grateful to Him for showing me the path whenever I was stuck at the crossroads of making any decision.

I thank my professor guide Shri P Sunil Kumar sir, whose strong conviction, persistent motivation, and endless enthusiasm helped us to have that yearning within.

I also thank my external guides Dr Raghunath Sarma Sir, Sri Anil Kumar Reddy Sir, for their constant support, anticipation, and belief which helped me at every step of this project.

I would also like to thank the University Administration for providing me with the resources required in the course of this project.

I thank all my classmates for their cheerfulness and for providing perceptions and vigor for this project.

Last but not least, I would like to thank my family for their unbroken assurance of me.

ABSTRACT

Every beneficiary registered under the AB-PMJAY scheme undergoes a set of procedures at one of the impaneled hospitals equipped with one or more specializations. The beneficiary avails these procedures at different points of time as prescribed by the doctor. These procedures are generally recurring in nature and invariably follow a similar treatment sequence for similar types of patients.

In this project, we have used link prediction techniques using knowledge graphs which comprise of interesting measures involved in association rule mining (ARM) and node similarity. Data from Nine AB-PMJAY treatments and 367 procedures from SSSIHMS was used for this purpose.

Results from this study can be used by Medical Practitioners to make effective decision in identifying the next procedure in the sequence. The work also helps in identifying the most valuable procedures using centrality measures.

TABLE OF CONTENTS

PROJECT REPOSITORY

All the codes and files related to project uploaded to github repository.

Main repository link :

https://github.com/Akhil174405/Link_Prediction_in_Medical_domain_using_Knowledge_Graphs

Below are the links to github.

1. Report

   https://github.com/Akhil174405/Link_Prediction_in_Medical_domain_using_Knowledge_Graphs/tree/main/Report

2. AB-PMJAY Data

   Graph Modelling :
   https://github.com/Akhil174405/Link_Prediction_in_Medical_domain_using_Knowledge_Graphs/blob/main/Project_Files/AB-PMJAY/AB-PMJAY_Data_Graph_Modeling.ipynb

   Association Mining Rules and Node similarity :

   https://github.com/Akhil174405/Link_Prediction_in_Medical_domain_using_Knowledge_Graphs/blob/main/Project_Files/AB-PMJAY/AB-PMJAY_Data_SCL_%26_Node_Similarity.ipynb

   Centrality Measures :

   https://github.com/Akhil174405/Link_Prediction_in_Medical_domain_using_Knowledge_Graphs/blob/main/Project_Files/AB-PMJAY/AB-PMJAY_Page_Ranking.ipynb

3. SSSIHMS Data

   Graph Modelling :
   https://github.com/Akhil174405/Link_Prediction_in_Medical_domain_using_Knowledge_Graphs/main/Project_Files/SSSIHMS_DATA/SSSIHMS_Data_Graph_Modelling.ipynb

   Association Mining Rules and Node similarity :

   https://github.com/Akhil174405/Link_Prediction_in_Medical_domain_using_Knowledge_Graphs/blob/main/Project_Files/SSSIHMS_DATA/SSSIHMS_SCL_%26_Node_Similarity.ipynb

   Centrality Measures :

   https://github.com/Akhil174405/Link_Prediction_in_Medical_domain_using_Knowledge_Graphs/blob/main/Project_Files/SSSIHMS_DATA/SSSIHMS_Page_Rank.ipynb

CHAPTER-1: INTRODUCTION

1.1. THE NECESSITY OF PREDICTION PROBLEMS IN MEDICAL DOMAIN

The healthcare industry is critical since it provides care to millions of people while contributing to the local economy. Artificial intelligence helps the healthcare industry in a variety of ways. By lending a helping hand, information technology is altering the healthcare industry. As we all know, artificial intelligence (AI) is the development of computing systems capable of doing activities that would typically need human intelligence. These entail difficult tasks such as making decisions, solving complex issues, and detecting objects. The benefits of technology, such as enhanced precision and high computation, which would take humans days to translate manually, are being applied to the healthcare industry to improve services and keep data structured.

Healthcare production

- Illnesses and therapies are predicted.
- Health risks are forecasted for distinct demographics of people.
- Assists in the management of medical records and workflow.
- Distinguishes between malignancies and normal anatomy.
- Drug development is aided, and expenditures are reduced.
- Identifies potential clinical trial sites.
- Identifies healthcare graphs.
- Assists pathologists in making more accurate and timely diagnosis.

1.2. KNOWLEDGE GRAPHS AND ITS IMPLICATIONS

There has been extensive use of graphs in the recent past viz. recommender systems, search engines, and chatbots. The Graph data models are basically used to establish the relationship between the real-world entities that comprise heterogeneous resources. Thes real-world entities form the data upon which graphs are built. Graph analytical techniques can then be implemented using the data that has been represented in graph format. Using query languages to explore the multi-hop connections between items.

Decision support systems(DSS) in the medical domain and patient symptom assessments are the prominent challenges today. The sparse data sets in the medical domain and the implicit relationships among the data items can be analyzed better using knowledge graphs.. This makes it simpler for doctors and service providers to find the information they want amid a plethora of factors and data sources. Existing solutions rely on knowledge graphs created manually or automatically using simple paired statistics. As a result,

making Knowledge graphs on diseases, symptoms, diets, supplements, pregnancy, child care, and other topics might be beneficial. They become essential when efficient reasoning over such knowledge graphs.

1.3. MOTIVATION

The sacred mission of Bhagawan Sri Sathya Sai Baba to help the human kind by providing the temples of healing at Puttaparthi and Whitefield, Bangalore to help the needy requires a sincere contribution from each individual .The kind of technical help required by the doctors to treat the patients vary from department to department. However it has been observed that any kind of visual aid or support in this regard has its own advantage over statistical tables.The recent emergence of knowledge graphs and its value added characteristics motivated us to take up this project.

It is also apt to mention here that the biggest health security scheme AB-PMJAY of the world launched by Government of India in 2018 to cater health benefits to 50 lakh people of the country belonging to deprived categories is also a major motivation for taking up this project work.

1.4. SCOPE

Any domain or research area that comprises activities carried out at different points of time giving rise to time series datasets.These are termed as activity sequences.Our project work will help the stakeholders of such domains to make decisions.

However, for the medical domain the current results will be useful in analyzing the patient's journey with the available datasets.This will provide the doctors an interface to visualize the usage patterns of each procedure in activity sequences depicted using the knowledge graph.

1.5. UNIQUENESS

This Project can be used by medical practitioners as a visualization tool by observing the node interactions along with their vital properties.The building of knowledge graphs from tabular data using association rule mining is also one of the unique features of the project.

1.6. OBJECTIVES

- To find similar types of patients by using node similarity..
- Predicting the next procedure for a given procedure or procedures using link prediction that uses Association Rule Mining .
- For identifying the most valuable procedures using the centrality measures like Page Rank, Closeness centrality and HITS (Authority and Hubs) methodology.

CHAPTER-2: TOOLS & TECHNOLOGIES

2.1. SYSTEM SPECIFICATIONS

Processor	: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
Installed RAM	: 8.00 GB (7.65 GB usable)
System type	: 64-bit OS, x64-based Processor

2.2. PLATFORMS

2.2.1. Jupyter Notebook

Jupyter Notebooks are an excellent tool for writing and iterating on Python code. It's a fantastic tool for building and interactively presenting data science projects. A notebook is a document that incorporates images, narrative text, mathematical calculations, and other rich media into a single document. Notebooks are becoming an increasingly popular choice at the heart of modern data research, analysis, and science, owing to the natural workflow that supports iterative and rapid improvement.

2.2.2. Neo4j

Neo4j stores and manages data in the connected state, keeping data relationships that enable lightning-fast queries, more context for analytics, and a pain-free customizable data model, thanks to its native graph database.
It is based on Java technology, a schema-free system, and is highly scalable.

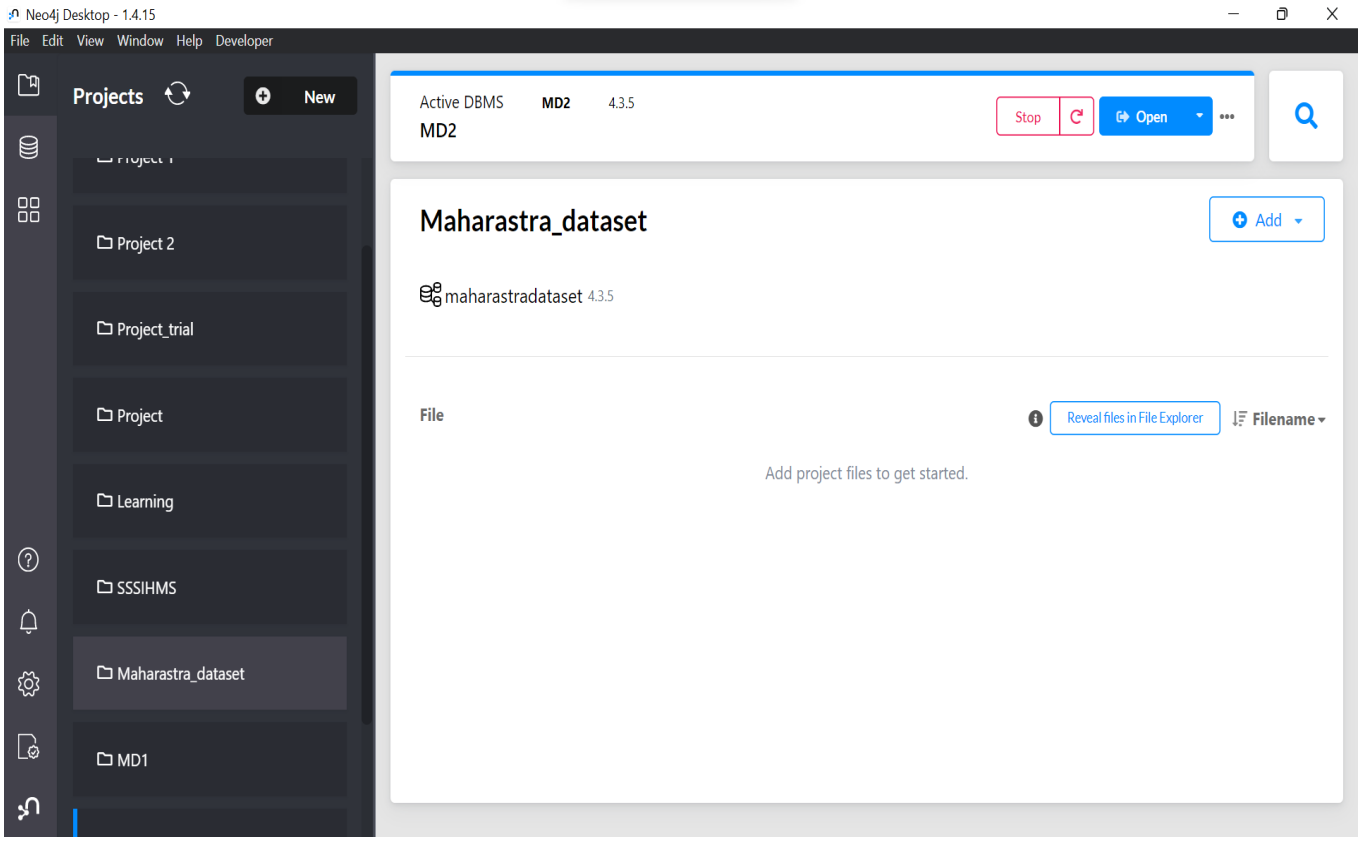
Advantages of Neo4j

Some of the advantages of Neo4j include a versatile data model, real-time analysis, handling multiple types of data, cypher query language, exclusion of joins.

2.2.3. Neo4j Desktop

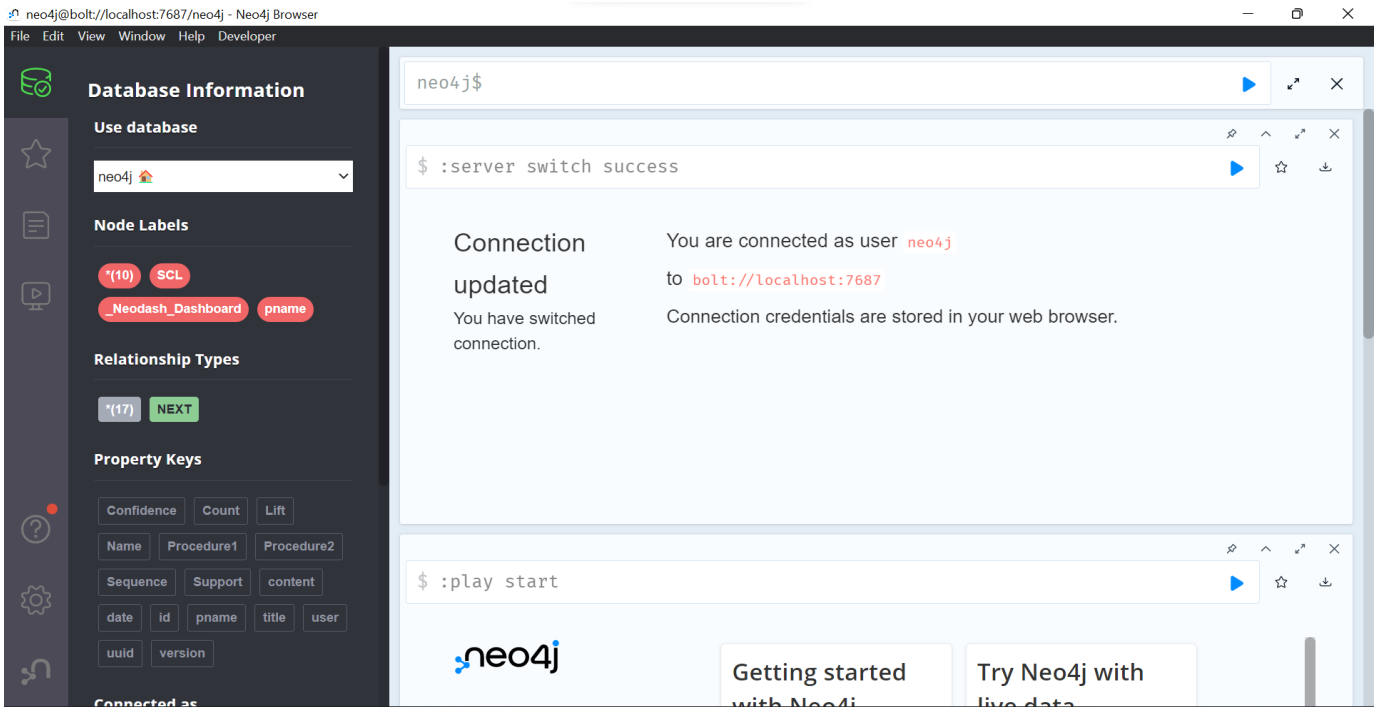
Neo4j Desktop is an installable program that will assist you in working with Neo4j, whether you are just starting or have past Neo4j knowledge. For a novice user, it is intended to help you learn and play with Neo4j locally by containing everything you need to get started. Once you've mastered Neo4j, Desktop will serve as your local development environment for Neo4j-related projects.

We create Projects, insert databases, and generate database passwords for authentication, ensuring database security.



2.2.4. Neo4j Browser

The Neo4j Browser is an interactive cypher command shell that allows you to interact with and see your graph's data. Neo4j Browser is included with Neo4j and is accessible in all Neo4j editions and versions.

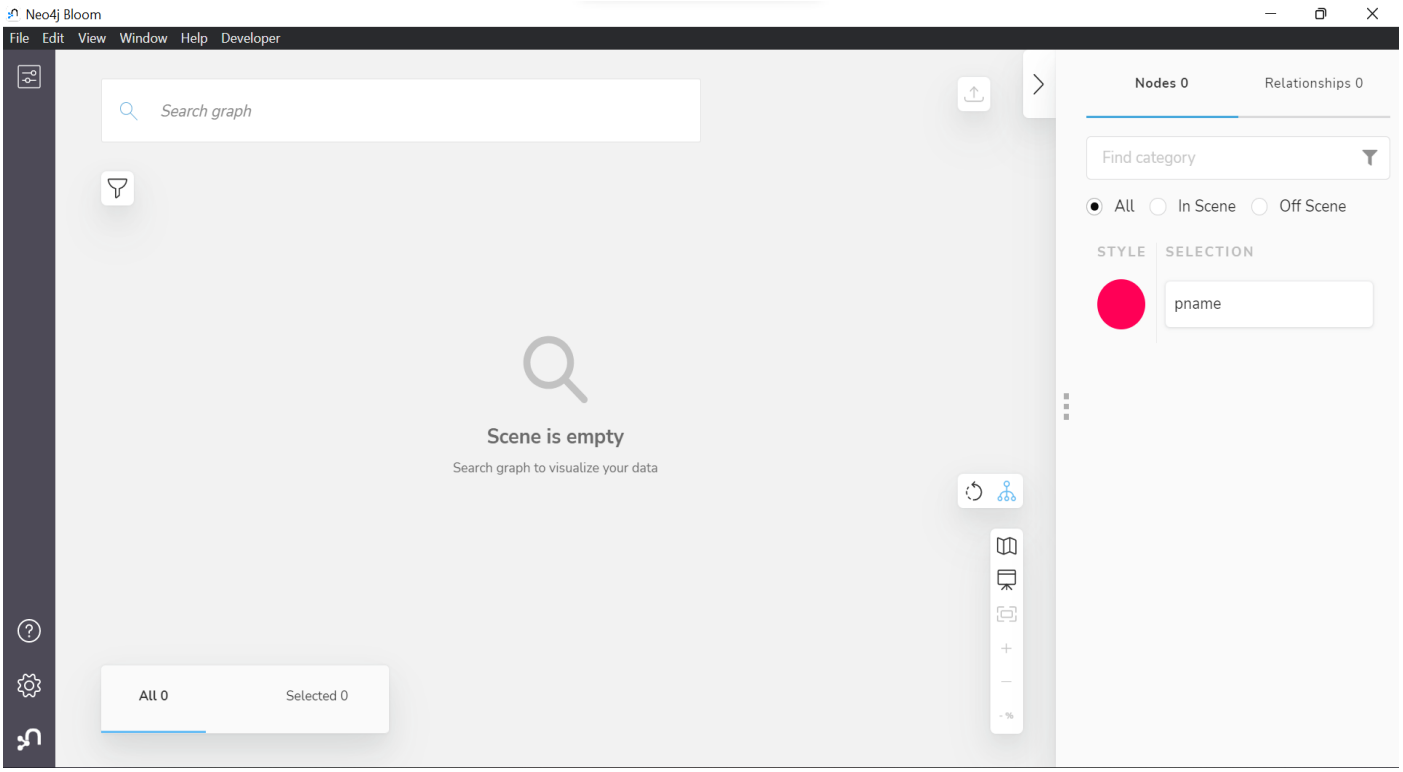


It has built-in tutorials that teach various concepts. Now that we have access to the graph, we can follow these instructions to begin working with data using Cypher Query Language (CQL).

After creating projects and configuring databases in Neo4j Browser, we generate a graph from the databases using CQL queries and work on it.

2.2.5. Neo4j Bloom

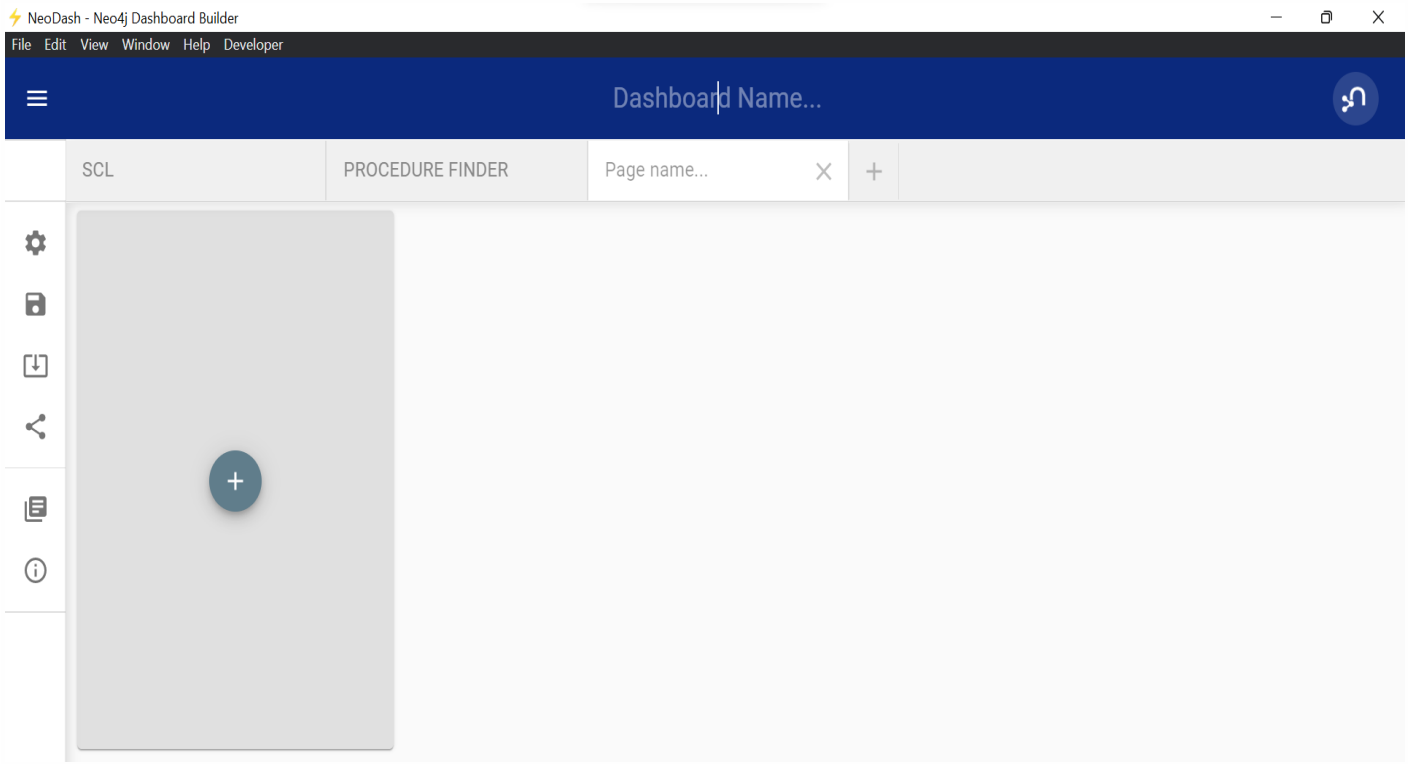
Neo4j Bloom is a graph exploration software that allows you to interact with graph data graphically. A graph interprets any information by connecting all the dots. There are people, places, and things. Accounts, products, and services Transactions, identities, and occurrences Neo4j Bloom highlights patterns in your data that you instinctively know exist and unexpected patterns you may not have predicted. This new data perspective offers new ways of thinking, working, and opportunities. And it's entertaining.



Neo4j Bloom is driven by the Neo4j graph database, a massively powerful engine for storing and accessing linked data. Bloom encapsulates such power in an interactive graph visualization environment, providing a business perspective on the graph.

2.2.2. Neo4j Neo Dashboard

NeoDash is a lightweight web app that integrates with Neo4j natively, allowing you to create a front-end without touching any code. It offers several reports that are natively compatible with Neo4j data types. Dashboards may be exported as JSON files for convenient storage and version management.



2.3. LANGUAGES USED

2.3.1. Python 3.8

1. Python is an object-oriented programming language that is high-level and interpreted.
2. Syntax is simple to learn and maintain, which saves time.
3. Python has modules and packages to let you reuse code and make programming easier.
4. Python is useful in a wide range of applications.
5. Organizes mathematical, statistical, and scientific procedures user-friendly.

2.3.2. Cypher Query Language

Why Cypher?

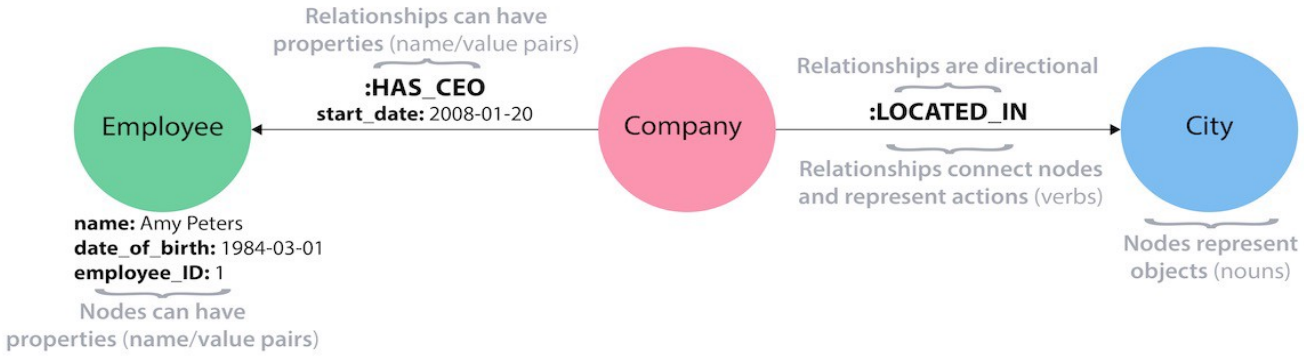
Neo4j's graph model comprises nodes and relationships, each of which can have its own set of characteristics. The pattern, which is the most valuable and powerful aspect of the property graph model, consists of nodes and relationships, which are the basic building blocks. Patterns are built up of node and relationship pieces and can create simple or sophisticated traversals and paths. The ability of the brain to recognize patterns is critical to its proper functioning. As a result, humans are adept at working with

patterns. Cypher is also heavily reliant on patterns, and it is built to recognize various permutations of these patterns in data, making it a logical and straightforward language to master.

Data may be retrieved from the graph using Neo4j's graph query language, Cypher. It's like SQL for graphs, and it's based on SQL, so you can concentrate on extracting the data you need from the graph (not how to get it). It is the most simple graph language to learn due to its resemblance to other languages and comprehensibility.

Users of Neo4j use Cypher to build expressive and efficient queries to perform any type of creating, reading, updating, or delete (CRUD) operation on their graph, and Cypher is the primary interface for Neo4j.

The Neo4j CQL query language is easy to understand and written in a human-readable style. It employs a simple yet effective data model. We can access nodes or relationships without using join operations or Indexes as we use in SQL. It is simple to obtain information about its next node or connection without the need of Joins or Indexes.



Source : Graph Data Platforms: Collections vs. Connections

2.4. LIBRARIES

2.4.1. Pandas

For manipulating and analysis of the data we use this panda library, in our case we used this to load CSV files.

2.4.2. Py2neo

We use this library for integrating with the Neo4j desktop. A high-level API, an OGM, admin tools, an interactive console, a Cypher lexer for Pygments, and other features are included in the library.

2.4.3. Neo4j - Libraries

Graph Data Science(GDS):

Neo4j Graph Data Science is a connected data analytics and machine learning platform that helps us understand the relationships or connections in large data to answer crucial questions and enhance predictions.

Advantage:

- The only connected data analysis platform that combines the ML surface and graph database into a single workspace is Neo4j Graph Data Science.
- This allows data scientists to execute algorithms and ML models without switching between tools for ETL.

Why Neo4j Graph Data Science?

Your data in a graph tells you what's significant, what's unusual, and what's coming up next.Make predictions and respond to business-critical inquiries.

Rather than focusing on row or column headings, graphs concentrate on data connections. Graphs provide a more natural, integrated way to look at and analyze data for richer context and uncover hidden patterns and insights.

Improve your models by understanding connections. A native Python client, a library of 65+ pre-tuned graph algorithms, linked data prep approaches, data connectors, and graph-native ML provide data scientists with everything they need without requiring them to jump between interfaces.

2.4.5. Networkx

NetworkX is a Python-based software package for creating, manipulating, and studying complex networks' structure, dynamics, and function. It's used to investigate big, complicated networks that are represented as graphs with nodes and edges. We can load and save complicated networks with networkx.

We may create a variety of random and traditional networks, study their structure, establish network models, create new network algorithms, and draw them.

CHAPTER-3: CONCEPTS

3.1. KEYWORDS

Nodes are the entities in the graph. They can store any number of properties (key/value pairs). Labels can be assigned to nodes to denote their various responsibilities. Node labels can also be used to associate metadata with specific nodes.

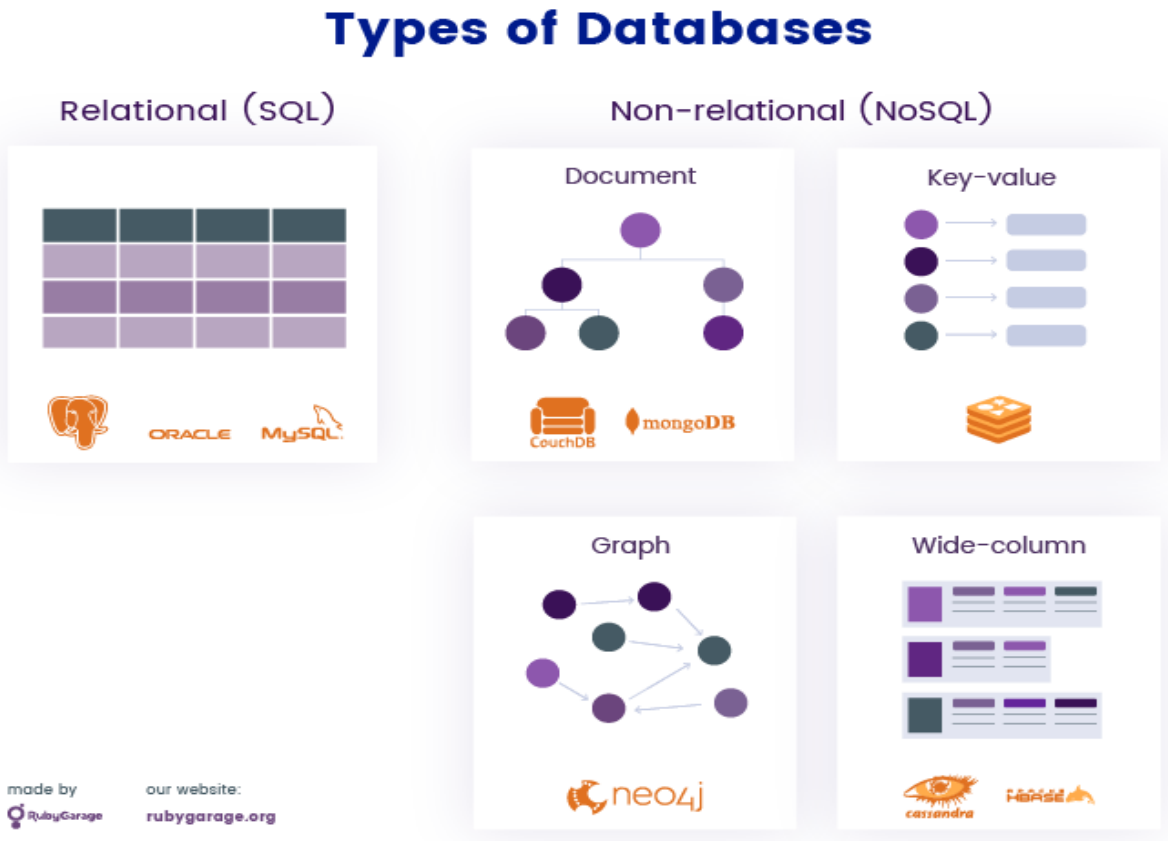
Relationships provide directed, named, semantically-relevant connections between two node entities.There is always a type, a direction, a start node, and an end node to a relationship. Properties can exist in relationships as well.

Labels are used to group nodes, and each node can be assigned multiple labels. Labels are indexed to speed up finding nodes in a graph.

Properties are attributes of both nodes and relationships. Neo4j allows for storing data as key-value pairs, which means properties can have any value (string, number, or boolean).

3.2. GRAPH DATABASE

A graph is a visual depiction of a group of things in which certain pairs of objects are linked together by connections. It's made up of nodes and relationships).



Source : https://rubygarage.org/blog/Neo4j-database-guide-with-use-cases

A graph database is a database that models data as a graph. Nodes are treated as entities and relations are treated as the connection between two nodes.

A prominent Graph Database is Neo4j. Oracle NoSQL Database, OrientDB, HypherGraphDB, GraphBase, InfiniteGraph, and AllegroGraph are some of the other graph databases available.

	Neo4j	Relational databases	NoSQL databases
Data storage	Graph storage structure	Fixed, predefined tables with rows and columns	Connected data not supported at the database level
Data modeling	Flexible data model	Database model must be developed from a logical model	Not suitable for enterprise architectures
Query performance	Great performance regardless of number and depth of connections	Data processing speed slows with growing number of joins	Relationships must be created at the application level
Query language	**Cypher**: native graph query language	**SQL**: complexity grows as the number of joins increases	Different languages are used but none is tailored to express relationships
Transaction support	Retains ACID transactions	ACID transaction support	BASE transactions prove unreliable for data relationships
Processing at scale	Inherently scalable for pattern-based queries	Scales through replication, but it's costly	Scalable, but data integrity isn't trustworthy

Source : https://rubygarage.org/blog/Neo4j-database-guide-with-use-cases

Why Graph Databases?

The majority of data nowadays occurs in relationships between various objects, and the link between the data is sometimes more helpful than the data itself.

Relational databases hold highly structured data in several records which store the same type of data, allowing them to be used to store structured data without maintaining the relationships between the records.

Graph databases in contrast to conventional databases, treat relationships and connections as first-class entities. Graph databases have a simpler data model than traditional databases, and they may be utilized with OLTP systems. Transactional integrity and operational availability are among the benefits they offer.

3.3. LINK PREDICTION

Link prediction is the problem of predicting the existence of a link between two entities in a network. The objective of link prediction is to identify pairs of nodes that will either form a link or may not form a link in the future.

Link prediction utilizes existing relations to infer new relations so as to build a more complete knowledge graph.

3.4. SIMILARITY

Similarity algorithms compute the similarity of pairs of nodes based on their neighborhoods or their properties. Several similarity metrics can be used to compute a similarity score.

3.5. NODE SIMILARITY

The Node Similarity algorithm compares a set of nodes based on the nodes they are connected to. Two nodes are considered similar if they share many of the same neighbors.

Node Similarity computes pairwise similarities based on either the Jaccard metric, also known as the Jaccard Similarity Score, or the Overlap coefficient, also known as the Szymkiewicz–Simpson coefficient.

3.6. kNN

The K-Nearest Neighbors technique calculates a distance value for each node pair in the network and establishes new interactions between each node and its k nearest neighbors. The distance is estimated depending on the attributes of the nodes.

This algorithm takes a monopartite graph as input. The network does not need to be linked; in fact, existing associations between nodes will be ignored - with the exception of random walk sampling, which is utilized as the first sample choice. Each node and its k closest neighbors form new associations.

The K-Nearest Neighbors method compares the attributes of each node. The k nodes with the most comparable attributes are the k-nearest neighbors.

3.7. CENTRALITY MEASURES

3.7.1. PAGE RANK

Based on the number of inbound relationships and the relevance of the respective source nodes, the PageRank algorithm determines the importance of each node within the network.

PageRank is an algorithm that measures the transitive, or directional, influence of nodes. All other centrality algorithms we discuss measure the direct influence of a node, whereas PageRank considers the influence of your neighbors and their neighbors.

For example, having a few influential friends could raise your PageRank more than just having a lot of low-influence friends.

PageRank is computed by either iteratively distributing one node's rank (originally based on the degree) over its neighbors or by randomly traversing the graph and counting the frequency of hitting each node during these walks.

3.7.2 CLOSENESS CENTRALITY

Closeness centrality is a way of detecting nodes that are able to spread information very efficiently through a graph.

The closeness centrality of a node measures its average farness (inverse distance) to all other nodes. Nodes with a high closeness score have the shortest distances to all other nodes.

3.7.3 AUTHORITY AND HUBS

The Hyperlink-Induced Topic Search (HITS) is a link analysis algorithm that rates nodes based on two scores, a hub score, and an authority score. The authority score estimates the importance of the node within the network. The hub score estimates the value of its relationships to other nodes.

CHAPTER-4: LITERATURE

4.1. TABULAR DATA TO KNOWLEDGE GRAPH

This study describes an efficient method for transforming tabular data into graphs.

The suggested approach produced good results while also achieving validity through greatly better levels of accuracy.

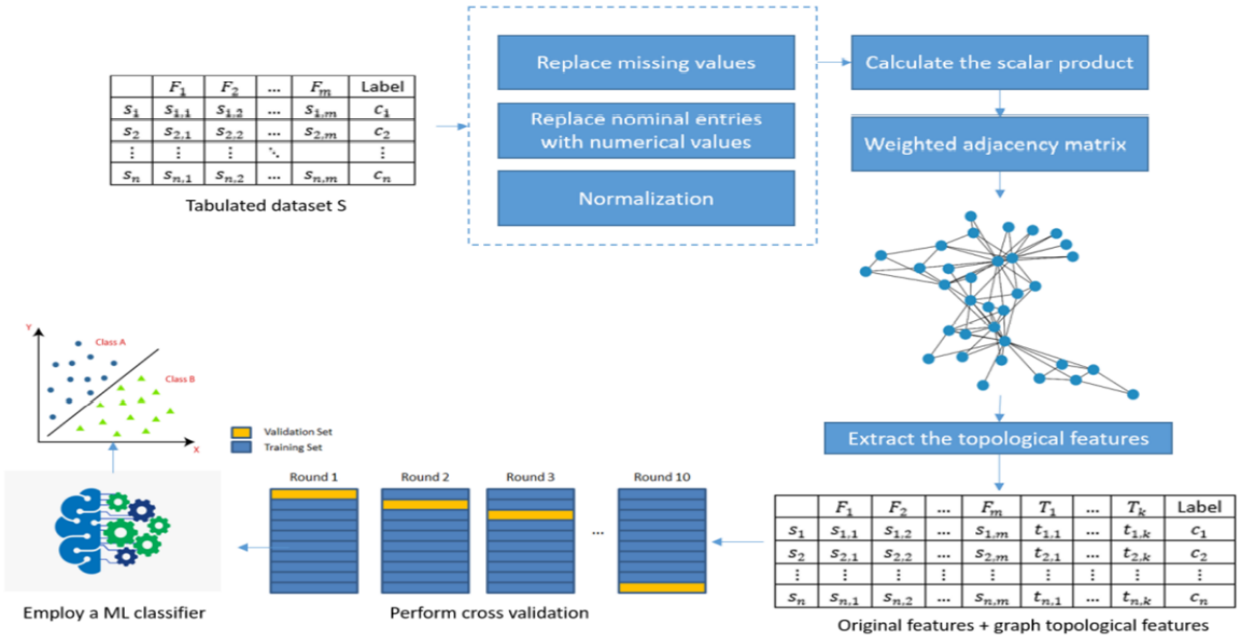


Figure 1: Overview of the suggested methodology

To put this unique approach to the test, they preprocessed the data sets, making sure that numerical values replaced nominal entries and that each missing component was imputed. The dataset's values were then normalized using either a traditional scaler or the MinMax scaler. Initially, they used a variety of classifiers, including Naive Bayes (NB), neural networks (NN), support vector machines (SVM), logistic regression (LR), k-nearest neighbor (kNN), random forest (RF), and decision tree (DT) forecast, to classify both classes in all four data sets and record classification accuracy. The only ones analyzed were the original features (OF) and default parameter settings. Only topological features were examined in the second phase. We also tested a few different combinations.

4.2. GRAPH-BASED TEMPORAL SIMILARITY OF PATIENT DATA

Computer-based decision support systems are frequently employed for certain tasks such as the diagnosis of sepsis. In this study, they offer a unique way to use temporal data from electronic health records based on similarity measurements.

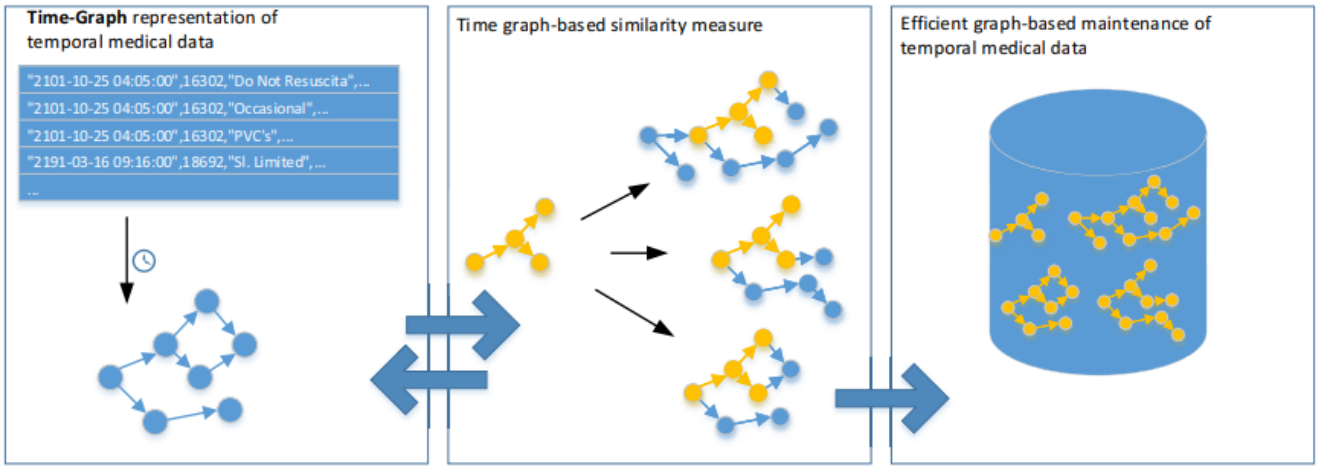


Figure 1 – Main Parts of the Time-Graph Similarity Framework.

The notion's three major parts correspond to the research topics. The initial stage is to do research on graph-based representations of temporal medical data. Second, a novel similarity measure for temporal data is developed in graph form. In the third section, existing database technologies for efficient graph-based temporal data storage and similarity retrieval are evaluated.

The most straightforward method is to create a timeline in which all data items are arranged in chronological order according to their timestamps.

Each node in this technique has one edge leading to the next node. A timeline already allows you to compare the progression of different situations across time. However, no distinction is established between different kinds of occurrences in this naive approach.

As a result, we'll look at more complex representations that take into account different types of nodes.

Each class's nodes have a linear temporal sequence, but they are intertwined with nodes from other classes. Unlike a timeline, such extra edges may aid in the representation of event-specific temporal relationships.

Further research is made regarding the most suitable topology of the resulting graph.

4.3. A SCALABLE GRAPH CONSTRUCTION FROM RELATIONAL TABLES USING MAP REDUCE

This paper introduces us to a new graph construction tool called "Table2Graph", which is used to convert relational tables into graphs by using the map-reduce framework.
Also, the performance of this tool in graph construction resulted in positive and is widely used nowadays.

We discovered that the most typical approach for these needs was to build ad hoc in-house scripts with no systematic available tools even for a single data source when exploring a variety of choices for these needs. As a result, we created Table2Graph, a scalable and flexible ETL tool.

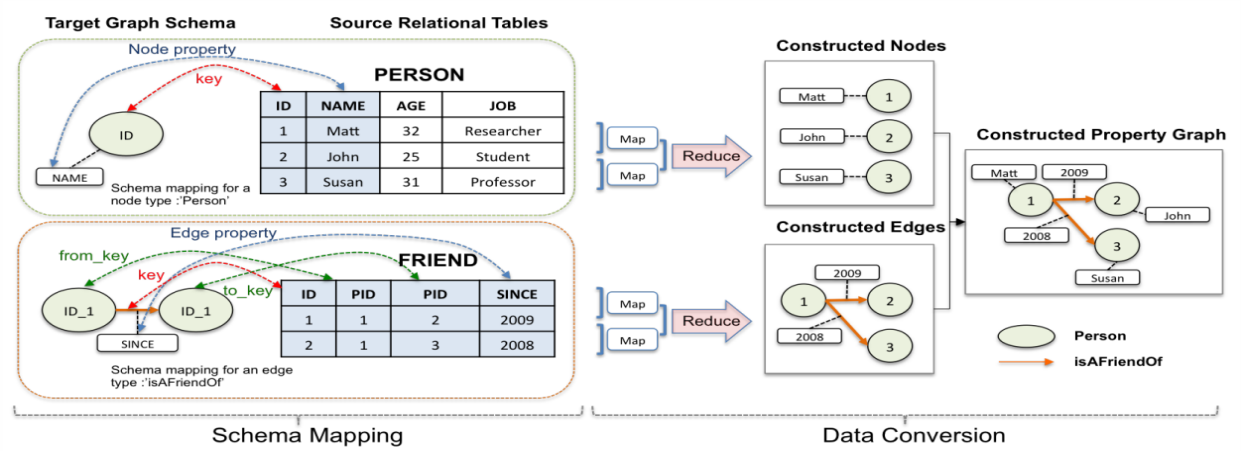


Fig. 1. An Overview of Two Main Steps, Schema Mapping and Data Conversion, in Table2Graph

Table Graph works in two stages: schema mapping and data conversion.

Schema Mapping

The Schema Mapping stage is intended to allow for a variety of setups and changes. It will help data practitioner to create a mapping for This capability allows a data practitioner to create a mapping for a graph model and then reuse it with minimal human effort for another model. An XML document is used to hold mappings from a data source to the target model. For nodes and edges, a mapping is created independently. This requires selecting a source schema property to be converted as a target node, other characteristics that are included as attributes in the target node, and a label that annotates the node for a node. Key, property, and label are the terms we use to describe those characteristics and labels. An edge schema mapping works in a similar way.

Data Conversion

Over two phases, the Data Conversion process runs a number of MapReduce tasks. During the first phase, thousands of Map-Reduce jobs create temporary nodes and edges, which we'll call node and edge

components. A universal resource identifier (URI) is used to uniquely identify a node or an edge during this phase. In the second stage, the node or edge components with the same URI are joined to form unique nodes and edges.

4.4. REAL-WORLD DATA MEDICAL KNOWLEDGE GRAPH: CONSTRUCTION AND APPLICATIONS

For its potential in smart healthcare applications, the medical knowledge graph is attracting passion for research and the delivery of healthcare.

This article presents a methodology for creating medical KG from electronic medical records (EMRs), as well as technical testing and end-to-end application samples.

Instead of the classical triplet structure in KG, this proposal offers a novel quadruplet structure to express medical knowledge. A novel related-entity ranking approach is defined that incorporates probability, specificity, and dependability into play (PSR).

$$\mathrm{PSR}_P(S_i, O_{ij}) = \mathrm{probability}_P(S_i, O_{ij}) \cdot \mathrm{specificity}_P(S_i, O_{ij}) \cdot \mathrm{reliability}_P(S_i, O_{ij}).$$

There are 22,508 entities and 579,094 quadruplets in a medical KG with 9 entity kinds, such as sickness, symptom, and so on. When normalized discounted cumulative gain (NDCG@10) compared to the term frequency-inverse document frequency (TF/IDF) approach, did rise from 0.799 to 0.906. All entities and relations had their embedding representations trained, which were shown to be effective in utilizing illness clustering.

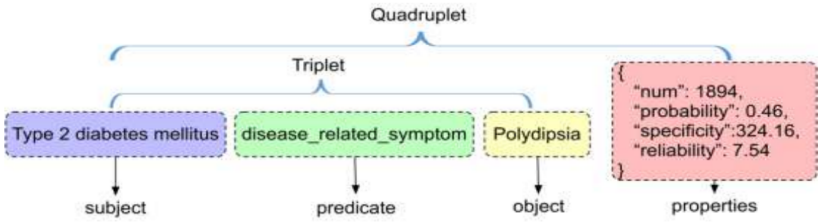


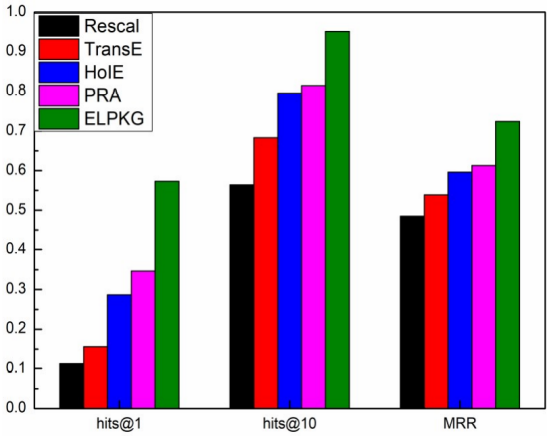
Fig: An example of a quadruplet

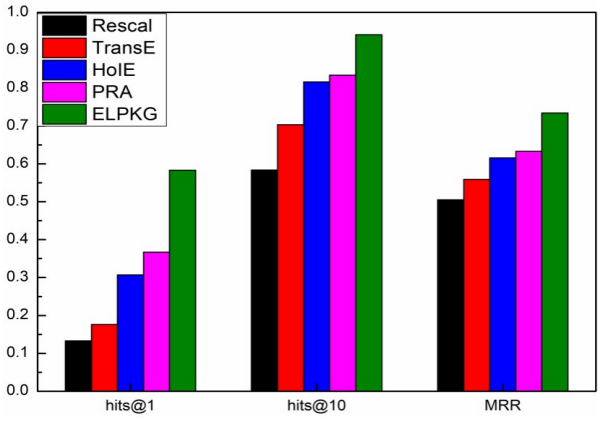
Using the outlined methodical technique, medical KG is created from EMRs on a very large scale. This embedding vector generated from the semantic representation provides the sickness categorization result verification because the proposed ranking function PSR outperforms all others. Since this follows the statistics approach using quadruplets the obtained KG has benefited from many real-time applications.

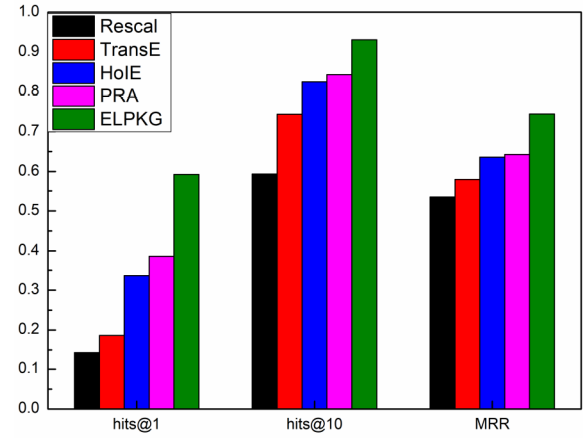
4.5. ELPKG: A HIGH ACCURACY LINK PREDICTION APPROACH FOR KNOWLEDGE GRAPH COMPLETION

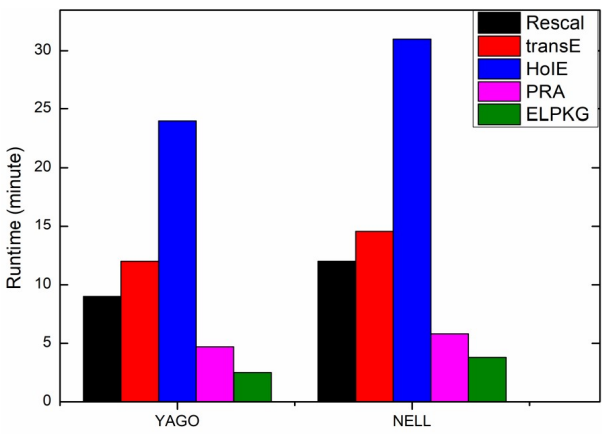
In a knowledge graph, the goal of link prediction is to leverage current relations to infer new ones in order to construct a more comprehensive knowledge graph. The inferred new relations plus the original knowledge graph form the symmetry of the entire knowledge graph. Previous link prediction research has concentrated exclusively on route or semantic-based characteristics, which may not give a full picture of features connecting entities and may result in a high proportion of wrong inference findings.

They present a unique approach called Entity Link Prediction for Knowledge Graph (ELPKG) to increase the accuracy of link prediction on large-scale knowledge graphs while maintaining This article presents a methodology for creating medical KG from electronic medical records (EMRs), and also technical testing and end-to-end application samples. o This article presents a methodology for creating medical KG from electronic medical records (EMRs), as also technical testing and end-to-end application samples. To express relationships between things, ELPKG first combines path and semantic-based attributes. After that, it employs a probabilistic soft logic-based reasoning approach to address the issue of non-deterministic knowledge reasoning. Finally, the entity link prediction approach is utilized to complete the relationship between entities. In comprehensive real-world testing, ELPKG outperforms baseline techniques on hits@1, hits@10, and MRR.









CHAPTER-5: DATA

5.1. TRANSACTIONAL DATA OF PATIENTS VISITING UNDER AB-PMJAY, MAHARASHTRA

Let us know each of the features:

Encounter_Id : Id given to each patient when he encountered the procedure.
Procedure : Procedure, which has been undergone by the patient.
Start_Date : Starting date of the procedure.
Patient_Id : For each patient, there is a unique id to identify.
Date_Order : The order in which the procedures are followed by each patient.
Age : Age of the patient
Gender : Male or Female

Shape : 7 columns and 1539 rows.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1539 entries, 0 to 1538
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Encounter_Id  1539 non-null   int64
 1   Procedure     1539 non-null   object
 2   Start_Date    1539 non-null   object
 3   Patient_Id    1539 non-null   int64
 4   Date_Order    1539 non-null   int64
 5   Age           1539 non-null   int64
 6   Gender        1539 non-null   object
dtypes: int64(4), object(3)
memory usage: 84.3+ KB
```

5.2. TRANSACTIONAL DATABASE OF PATIENTS VISITING SSSIHMS, BANGLORE.

Let us know each of the features:

Patient_Id	: For each patient, there is a unique id to identify.
Patient_Name	: Name of the patient.
Sex	: Male or Female
Age	: Age of the patient
Operation_Date	: Date of when the operation or procedure was held.
Operation_Code	: It's like procedure code (Ex: CAGM, PTCAP).
Weight	: Weight of the patient
Height	: Height of the patient
Diagnosis	: Description of procedures.

Shape : 8 columns and 8694 rows.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8694 entries, 0 to 8693
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   PATIENT_ID    8694 non-null   object
 1   PATIENT_NAME  8694 non-null   object
 2   SEX           8694 non-null   object
 3   AGE           8694 non-null   int64
 4   OPER_DT       8694 non-null   datetime64[ns]
 5   OPERCODE      8694 non-null   object
 6   WEIGHT        5512 non-null   float64
 7   HEIGHT        5475 non-null   float64
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 543.5+ KB
```

CHAPTER-6: OUR CONTRIBUTION

6.1. PROJECT WORKFLOW AND LEARNINGS:

After reviewing the fundamentals of graph knowledge that we learned during our undergraduate studies. We are looking for Python modules for creating graphs, specifically knowledge graphs. We looked at articles in the medical field that featured graphs.

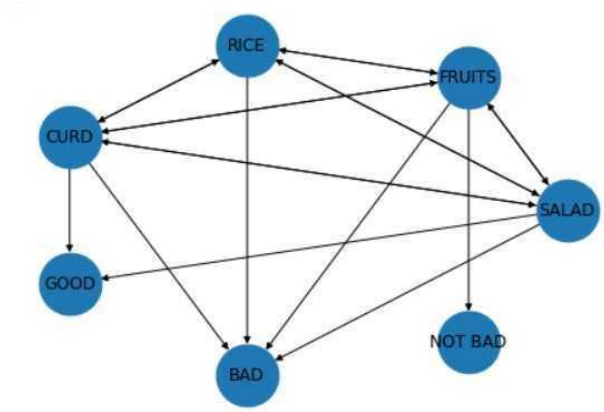
We began by studying the networkx graph library. In the networkx library, we learned how to make nodes and edges. We used the Facebook dataset to figure out how the networkx library was used to construct knowledge graphs. To have a better understanding of how graphs function, we began working with the nutritional dataset that we developed. We attempted to obtain the necessary knowledge graphs but were unable to do so.

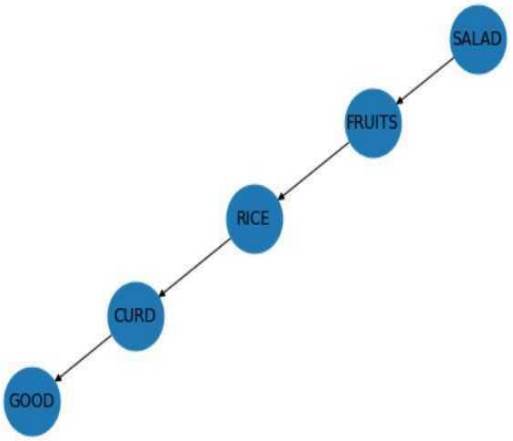
6.1.1 DIETARY DATA SET

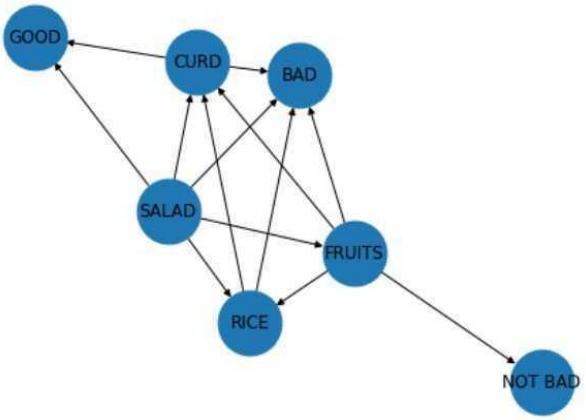
This is the dataset that we built and worked with to learn Networkx. We built this since it involves a series of procedures, with the last score being the last node to reach. We must preprocess the data before passing it to the networkx, such as providing the edges that must be connected in the dataset.

	A	B	C	D	E
1	IN1	IN2	IN3	IN4	SCORE
2	SALAD	FRUITS	RICE	CURD	GOOD
3	SALAD	FRUITS	CURD	RICE	BAD
4	SALAD	RICE	FRUITS	CURD	BAD
5	SALAD	RICE	CURD	FRUITS	NOT BAD
6	SALAD	CURD	FRUITS	RICE	BAD
7	SALAD	CURD	RICE	FRUITS	BAD
8	FRUITS	SALAD	RICE	CURD	GOOD
9	FRUITS	SALAD	CURD	RICE	BAD
10	FRUITS	RICE	SALAD	CURD	BAD
11	FRUITS	RICE	CURD	SALAD	GOOD
12	FRUITS	CURD	SALAD	RICE	BAD
13	FRUITS	CURD	RICE	SALAD	BAD
14	RICE	SALAD	FRUITS	CURD	BAD
15	RICE	SALAD	CURD	FRUITS	BAD
16	RICE	FRUITS	SALAD	CURD	BAD
17	RICE	FRUITS	CURD	SALAD	BAD
18	RICE	CURD	SALAD	FRUITS	NOT BAD
19	RICE	CURD	FRUITS	SALAD	BAD
20	CURD	SALAD	FRUITS	RICE	BAD
21	CURD	SALAD	RICE	FRUITS	BAD
22	CURD	FRUITS	SALAD	RICE	BAD
23	CURD	FRUITS	RICE	SALAD	BAD
24	CURD	RICE	SALAD	FRUITS	BAD
25	CURD	RICE	FRUITS	SALAD	BAD

These are the graphs we have got from the networkx.







Then we started learning py2neo, DGL, and Jupyter notebook itself, to produce knowledge graphs. On the dietary dataset, We attempted to make it more interactive with several sorts of graphs. Coding In the networkx, for the visualization. Later, we came up with the platform Neo4j which is interactive and specially designed for graphs. Based on the knowledge acquired on graphs we started working in the Neo4j.

To get started with Neo4j, we'll need to learn the Cypher query language (CQL), which is used to query the graph database. We learned how to use data frames to introduce properties into Neo4j for nodes and edges. Then we attempted to picture them. Then we used CQL to retrieve data from CSV and Excel files and insert nodes and edges. We attempted to insert graph attributes after constructing nodes and edges in graph databases. We used py2neo to merge Neo4j with Jupyter notebook in a Jupyter notebook environment. We may query the metrics from the graph databases using Python from Jupyter Notebook.

Then, for the tables from the graph databases, we may utilize pandas data frames. We can also employ those data frames in machine learning techniques. On the dietary dataset, we experimented with visualizations. We discovered two libraries, nxNeo4j and py2neo, to connect Jupyter notebook with Neo4j, and we corrected py2neo. It worked perfectly and also provided us with Neo4j statistics. I looked into numerous graph packages to find the visuals I needed. We choose Neo4j as our platform. We looked into it and discovered

that it is a graph database that stores information as nodes and edges. We discovered a Neo4j desktop that provides us with far more interactive knowledge graphs. We began working on the Neo4j desktop shortly after that. We must first learn Cypher Query Language in order to deal with it (CQL). For studying CQL, we watched several YouTube videos and read numerous publications.

6.2. WORK ON MAHARASHTRA DATASET

In order to work with graph databases, graph modeling is the first and main step to perform, after that just querying for retrieving information on our requirements.

6.2.1. Graph Modeling

- It's nothing but , we have to first select which column values should come under nodes and select their properties.
- Deciding which relations have to be created between the nodes.
- So accordingly we have to divide our main csv file into separate csv sfiles.

Graph modeling with respect to our problem

Nodes

I. Patients

II. Encounter

III. Procedures

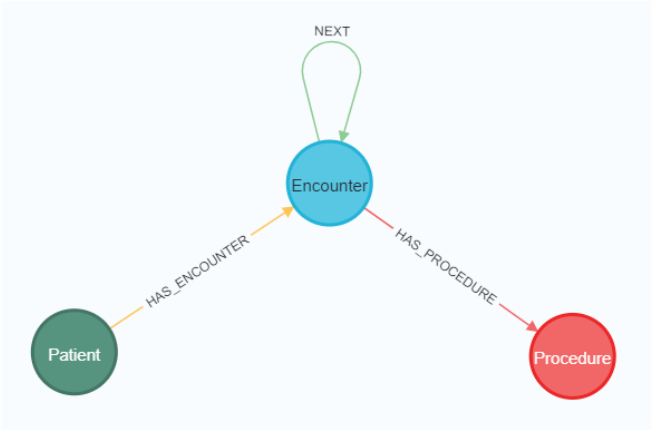
Relations

a. NEXT - relation between two encounters which says the sequence of the procedure.

b. HAS_ENCOUNTER - relation between patient and encounter.

c. HAS_PROCEDURE - relation between Encounter and Procedure.

Schema



After graph modeling we loaded the respective CSV files into the Neo4j desktop to work with them.

When a Patient encounters the procedure, his visit is considered the Encounter and a patient can visit many times, so we can have multiple encounters in it followed by a certain sequence depicted through NEXT relation every Encounter has a Procedure.

Every Patient will be having the patient details since the patient is making a visit, encounter file has the patient_id which matches with the patient_id in the Patient file.
The procedure is matched with encounter_id in the Encounter file since every patient's visit has a procedure to be followed. Files are connected through this Encounter and patient_id.

6.2.2. Workflow

1. Import required libraries

   Here we used pandas and py2neo library.

   Pandas - for working with data frames.

   py2neo - to integrate and run queries in Neo4j.

2. Before integrating the graph database, we have to create a project in Neo4j desktop and create a new database, and set a password to it.

3. Then will integrate from Jupyter notebook to Neo4j desktop, with the help of function named "Graph" from py2neo , it takes the arguments such as Neo4j browser link, auth which takes username and password as authentication. U can see the figure below.

Integrating with Neo4j using py2neo

```
graph = Graph('bolt://localhost:7687', auth=('neo4j', 'nextproject'))
```

4. After integrating with Neo4j , the next step is to load our data into Neo4j desktop in order to get the knowledge graphs and retrieve some insights from the data.

5. Maharashtra dataset is divided into three csv files such as Patients, Procedures and Encounters. This separation of the main csv file was done through graph modeling.

   I. *Patient csv file* contains the information about patients such as Patient_Id, Age, and Gender.

II. *Procedures csv file* contains Patient_Id, Procedure (Name of the procedure), Encounter_Id(Id given for each beneficiary when he/she encountered the procedure) and Date on which he/she underwent the procedure.

III. *Encounter csv file* contains Encounter_Id(Id given for each beneficiary when he/she encountered the procedure) and Date on which he/she underwent the procedure, Patient_Id.

Here *Patient_Id* and *Encounter_Id* play a vital role in mapping or creating relationships between the nodes.

6. Before loading CSV files we have to create constraints or labels for each CSV file.
   Constraints guarantee that property values are unique for all nodes having a certain label.
   So we created three constraints for the patient, procedure and encounter.

7. First we loaded the Patients csv file into Neo4j, created a node for each and every row in the csv file and set the properties Gender and Age.
   Patient_id is used as a primary key to access the information about the details
   After loading the csv file , we will get the following output which gives us the statistics like how many nodes created , relations created and properties setted using stats() function in py2neo .

```
{'labels_added': 512, 'nodes_created': 512, 'properties_set': 1536}
```

Fig: Stats after Patients file loaded into Neo4j

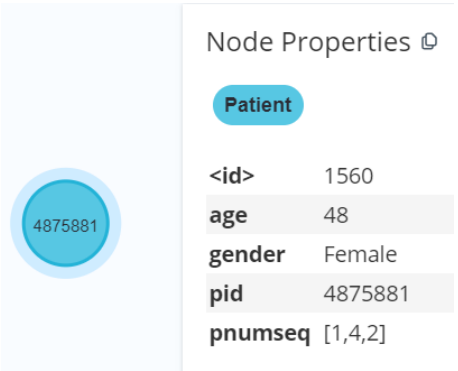


Fig: Representation of node and its properties in Neo4j Desktop.

8. Then we loaded the Encounter csv file , in addition, we created the relationship called "HAS_ENCOUNTER" between patient and encounter nodes as we discussed in graph modelling. After loading the csv file , you can see the stats below:

```
{'labels_added': 1539,
 'relationships_created': 1539,
 'nodes_created': 1539,
 'properties_set': 6156}
```

Fig: Stats after Encounter file loaded into Neo4j.

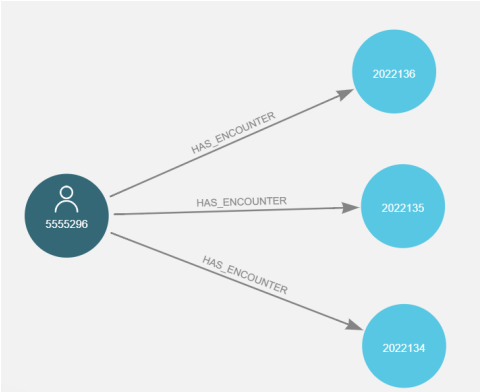


Fig: Representation of relation between patient and Encounter

9. It shows Patient with Id “555296” has encountered three procedures.

10. Finally, we loaded the Procedure CSV file, in addition, we created the relationship called “ HAS_PROCEDURE” between Encounter and Procedure nodes as we discussed in graph modeling. After loading the CSV file, you can see the stats below:

```
{'labels_added': 9,
 'relationships_created': 1539,
 'nodes_created': 9,
 'properties_set': 6165}
```

Fig: Stats after Procedure file loaded.

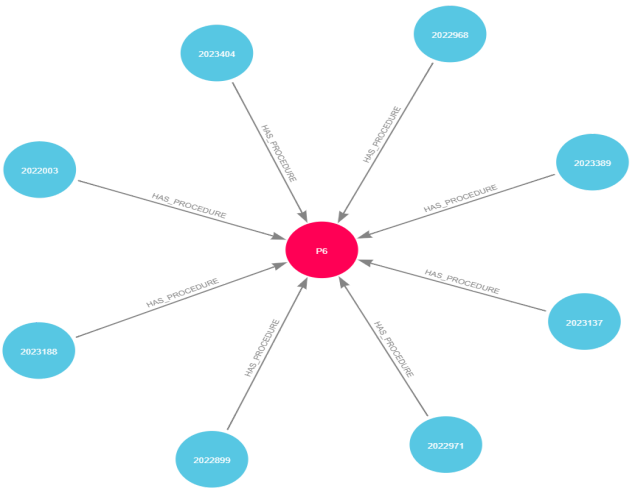


Fig: Representation of HAS_PROCEDURE relation between Encounter and Procedure.

11. Then we finally created the NEXT relationship between Encounters.

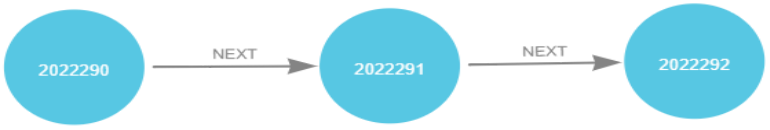


Fig: Representation of NEXT relation between Encounter and Procedure.

12. Finally, I want to combine all these and show you what the knowledge graph of a single patient looks like.

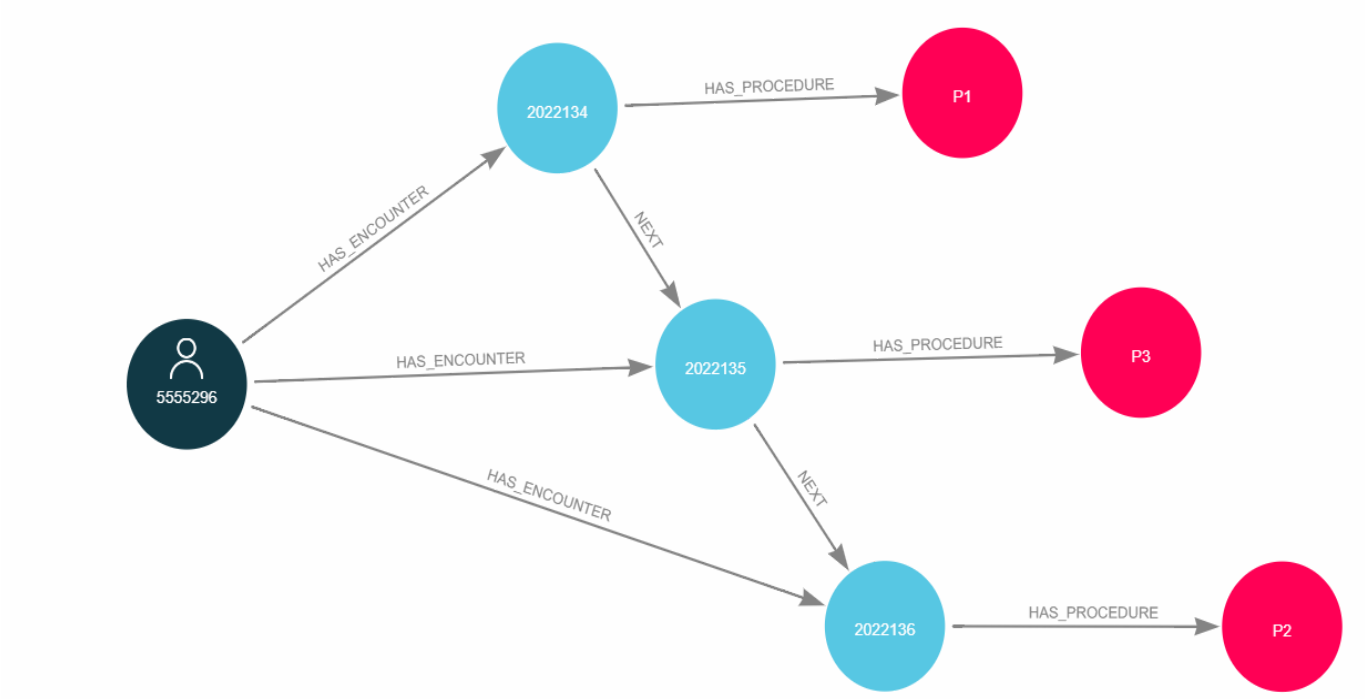


Fig: Knowledge graph of a single patient

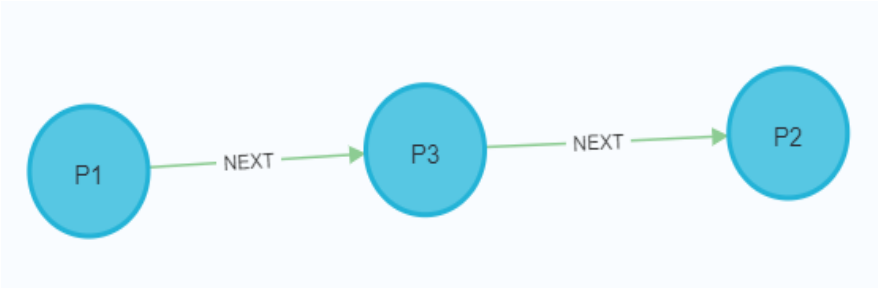
Explanation :

1. One can clearly see that it's an entire knowledge graph of a single patient's journey.
2. A Lehmann can understand by seeing the graph.
3. It says Patient (5555296) has encountered three procedures that too in sequence.
4. First the patient encountered procedure “P1”, then encountered procedure “P3” and finally “P2”.

6.2.3. Insights from the data - Exploratory Data Analysis(EDA)

Using Cypher query language we tried out some queries in order to understand the data. We can say it as Exploratory Data analysis in Neo4j

1. If we provide Patient_Id in a cypher query, we can get the graph of a single patient.



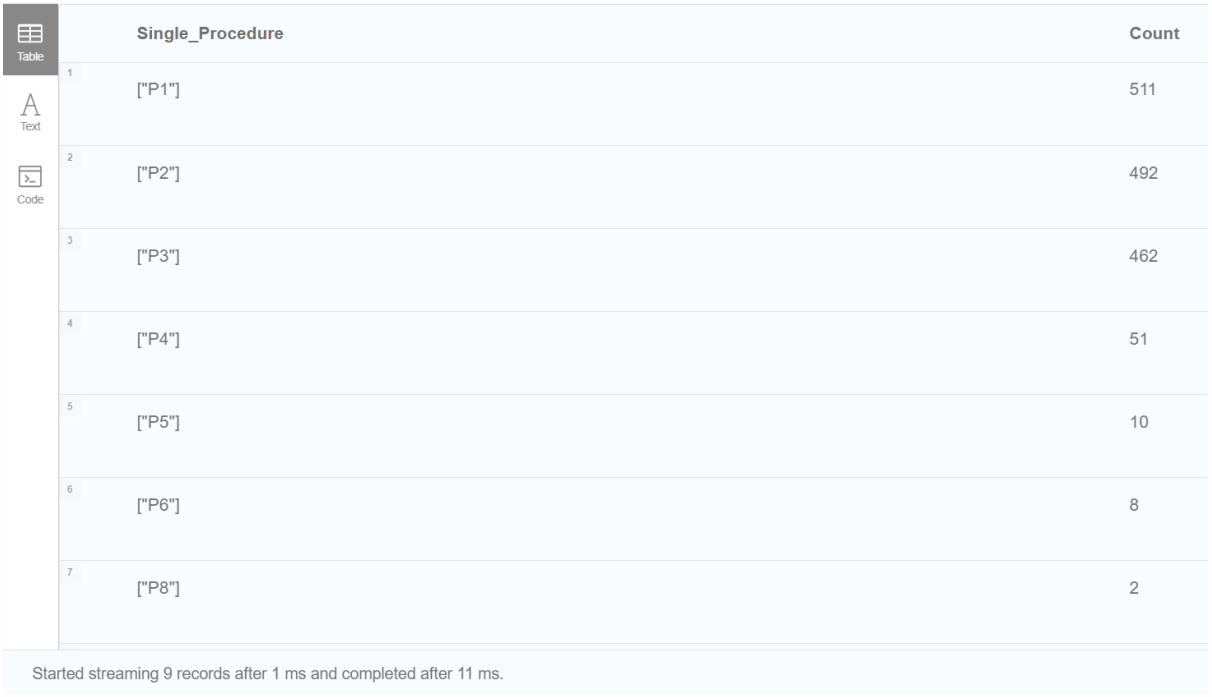
2. Patients and their treatment sequence.

Table
Text
Code

	Patient_Id	Treatment_Sequence
1	"5088343"	["P1", "P3", "P2"]
2	"4761557"	["P1", "P4", "P2"]
3	"4954536"	["P1", "P3", "P2"]
4	"4951052"	["P1", "P3", "P2"]
5	"5493957"	["P1", "P3", "P2"]
6	"5731436"	["P1", "P3", "P2"]
7	"4702183"	["P1", "P3", "P2"]

Started streaming 512 records after 38 ms and completed after 47 ms.

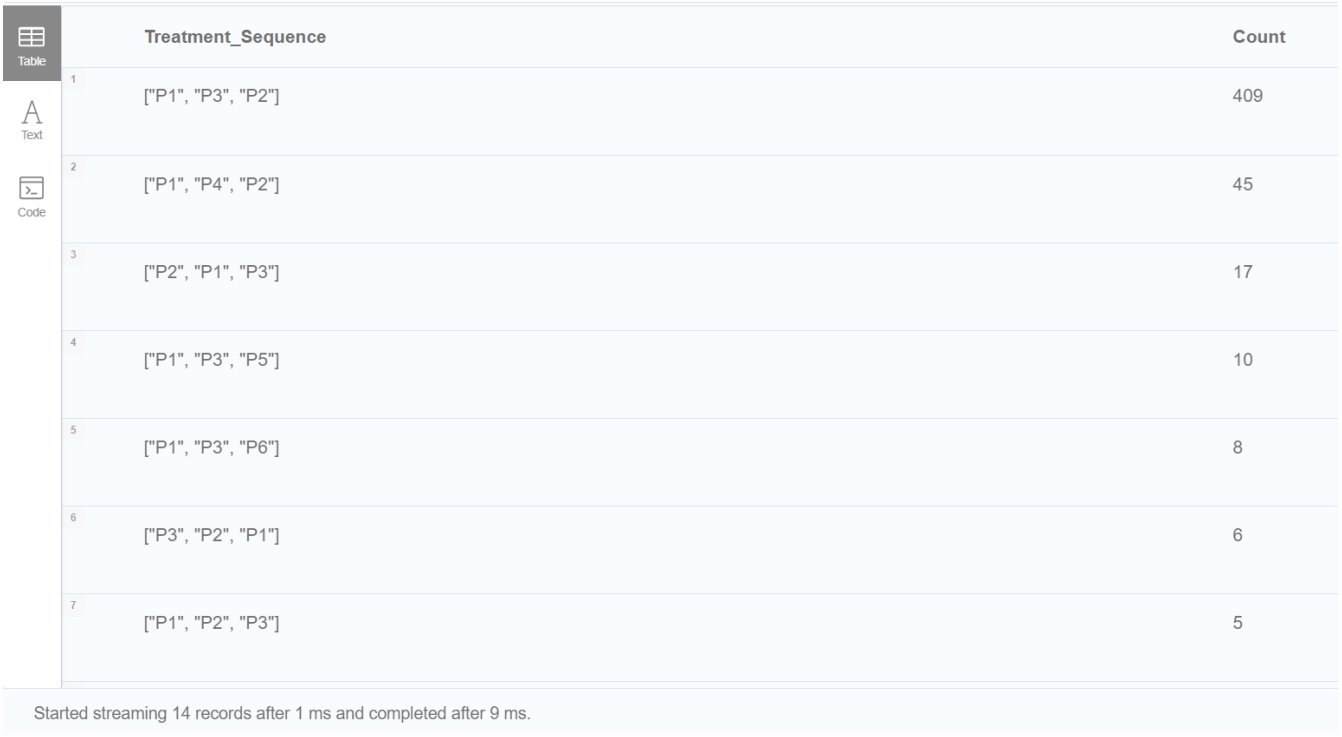
3. In the below figure you can see the no of single procedures and their count individually.



	Single_Procedure	Count
1	["P1"]	511
2	["P2"]	492
3	["P3"]	462
4	["P4"]	51
5	["P5"]	10
6	["P6"]	8
7	["P8"]	2

Started streaming 9 records after 1 ms and completed after 11 ms.

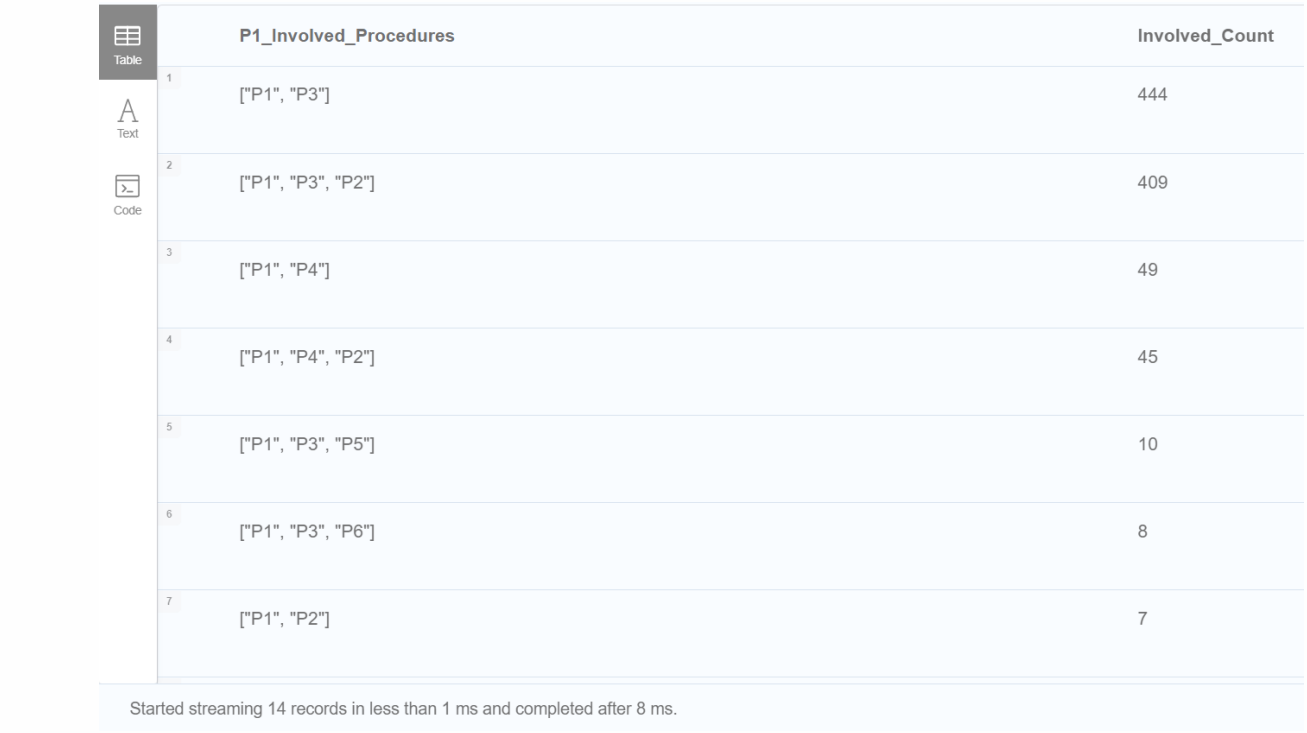
4. Here you can see the unique treatment sequence and their count out of 512 patients.



	Treatment_Sequence	Count
1	["P1", "P3", "P2"]	409
2	["P1", "P4", "P2"]	45
3	["P2", "P1", "P3"]	17
4	["P1", "P3", "P5"]	10
5	["P1", "P3", "P6"]	8
6	["P3", "P2", "P1"]	6
7	["P1", "P2", "P3"]	5

Started streaming 14 records after 1 ms and completed after 9 ms.

5. In the below, we got all the procedures related to “P1” and that too occurred first, followed by other procedures and also their count.



	P1_Involved_Procedures	Involved_Count
1	["P1", "P3"]	444
2	["P1", "P3", "P2"]	409
3	["P1", "P4"]	49
4	["P1", "P4", "P2"]	45
5	["P1", "P3", "P5"]	10
6	["P1", "P3", "P6"]	8
7	["P1", "P2"]	7

Started streaming 14 records in less than 1 ms and completed after 8 ms.

6.2.4. Link Prediction with the help of Association Rule Mining

Association rule mining (ARM) finds interesting associations and relationships among large sets of data items. This rule displays the frequency with which an item set appears in a transaction. A Market Based Analysis is a common example.

Support:

Support is an indication of how frequently the itemset appears in the dataset.

$$support = P(A \cap B) = \frac{(\text{number of transactions containing } A \text{ and } B)}{(\text{total number of transactions})}$$

Confidence:

Confidence is the percentage of all transactions satisfying X that also satisfy Y.

$$\text{conf}(X \Rightarrow Y) = P(Y|X) = \frac{\text{supp}(X \cap Y)}{\text{supp}(X)} = \frac{\text{number of transactions containing } X \text{ and } Y}{\text{number of transactions containing } X}$$

Lift:

Lift is a relative strength indicator showing the association between 2 Items.

For this dataset, we need to find out the support, confidence, and lift.

$$\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cap Y)}{\text{supp}(X) \times \text{supp}(Y)}$$

Example

For the procedures P1, P2, P3.

Let's find the support for P2 having the sequence (P1->P3->P2)

Given,

Count of P1 = 511

Count of P2 = 492

Count of P3 = 462

Count of (P1,P3) = 444

Count of the sequence = 409

Calculations:

First we need to find the support of the single procedure P1, P2,P3

Support(P1) = $\frac{Count\ of\ P1}{Total\ Count} = \frac{511}{512} = 0.998047$

Support(P2) = $\frac{Count\ of\ P2}{Total\ Count} = \frac{492}{512} = 0.960938$

Support(P3) = $\frac{Count\ of\ P3}{Total\ Count} = \frac{462}{512} = 0.902344$

Support(P1,P3) = $\frac{Count\ of\ (P1,P3)}{Total\ Count} = \frac{444}{512} = 0.867188$

Support (P1,P3,P2) = $\frac{Count\ of\ the\ sequence(P1,P3,P2)}{Total\ count} = \frac{409}{512} = 0.798828$

Confidence ((P1,P3)->P2) = $\frac{Support(P1,P3,P2)}{Support(P1,P3)} = \frac{0.798828}{0.867188} = 0.92117$

Lift ((P1,P3)->P2) = $\frac{Support(P1,P3,P2)}{Support(P1,P3) \times Support(P2)} = \frac{0.798828}{0.867188 \times 0.960938} = 0.95861$

Similarly we need to find the procedure for all the sequences in the database using CQL queries and using python we can get this dataframe.

	names	Support	Confidence	Lift
0	[P1]	0.99805	1.00000	0.00000
1	[P2]	0.96094	1.00000	0.00000
2	[P3]	0.90234	1.00000	0.00000
3	[P1, P3]	0.86719	0.86888	0.96292
4	[P3, P2]	0.81641	0.90477	0.94155
5	[P1, P3, P2]	0.79883	0.92117	0.95861
6	[P4]	0.09961	1.00000	0.00000
7	[P1, P4]	0.09570	0.09589	0.96262
8	[P4, P2]	0.08984	0.90192	0.93858
9	[P1, P4, P2]	0.08789	0.91839	0.95572
10	[P2, P1]	0.05078	0.05284	0.05295
11	[P2, P1, P3]	0.03320	0.65380	0.72456
12	[P1, P3, P5]	0.01953	0.02252	0.02496
13	[P3, P5]	0.01953	0.02164	0.02399
14	[P5]	0.01953	1.00000	0.00000
15	[P1, P3, P6]	0.01563	0.01802	0.01997
16	[P3, P6]	0.01563	0.01732	0.01920
17	[P6]	0.01563	1.00000	0.00000
18	[P1, P2]	0.01367	0.01370	0.01425
19	[P3, P2, P1]	0.01172	0.01436	0.01438
20	[P2, P3]	0.00977	0.01017	0.01127
21	[P1, P2, P3]	0.00977	0.71470	0.79206
22				
23	[P1, P4, P3, P2]	0.00391	1.00000	1.04065
24	[P4, P3]	0.00391	0.03925	0.04350
25	[P4, P3, P2]	0.00391	1.00000	1.04065
26	[P2, P1, P4]	0.00391	0.07700	0.77300
27	[P3, P1]	0.00391	0.00433	0.00434
28	[P3, P1, P2]	0.00391	1.00000	1.04065
29	[P1, P7]	0.00391	0.00392	0.00408
30	[P1, P7, P2]	0.00391	1.00000	1.04065
31	[P7]	0.00391	1.00000	0.00000
32	[P7, P2]	0.00391	1.00000	1.04065
33	[P1, P8]	0.00391	0.00392	0.00408
34	[P1, P8, P3]	0.00391	1.00000	1.10823
35	[P8]	0.00391	1.00000	0.00000
36	[P8, P3]	0.00391	1.00000	1.10823
37	[P4, P2, P1]	0.00195	0.02171	0.02175
38	[P4, P9]	0.00195	0.01958	0.01961
39	[P4, P9, P3]	0.00195	1.00000	1.10823
40	[P4, P9, P3, P2]	0.00195	1.00000	1.04065
41	[P9]	0.00195	1.00000	0.00000
42	[P9, P3]	0.00195	1.00000	1.10823
43	[P9, P3, P2]	0.00195	1.00000	1.04065

We use this knowledge of data frame to create a knowledge graph.

To create a knowledge graph we need to create nodes for all the procedures and edges.
The knowledge graph consists of procedures and their following sequences.

To get nodes which are needed to be connected, we write a CQL query along with the edge count.

Now get the edges which are need to connect with the edge count between them.

Now we create the edges between them.

	Procedure1	Procedure2	count
0	P1	P3	444
1	P3	P6	8
2	P1	P4	49
3	P4	P3	2
4	P3	P2	418
5	P4	P2	46
6	P2	P1	26
7	P3	P5	10
8	P3	P1	2
9	P1	P2	7
10	P2	P3	5
11	P1	P7	2
12	P7	P2	2
13	P4	P9	1
14	P9	P3	1
15	P1	P8	2
16	P8	P3	2

We got the knowledge graphs with the sequences followed in the databases.

Now the task goes like this: we need to get the properties for each node, consisting of Support, Confidence, and Lift values of the sequences followed until that node.

For example,
If we take the P4 procedure, we take the sequences that have ended with P4, then the support, confidence, and lift values for the P4 are accumulated and appended to the list.
Then the list is given as a property to the node P4.

Like this, each node will get the property.

Support for P4 : [[['P4'], 0.09961], [['P1', 'P4'], 0.0957], [['P2', 'P1', 'P4'], 0.00391]]

Confidence for P4 : [[['P4'], 1.0], [['P1', 'P4'], 0.09589], [['P2', 'P1', 'P4'], 0.077]]

Lift for P4 : [[['P4'], 0.0], [['P1', 'P4'], 0.96262], [['P2', 'P1', 'P4'], 0.773]]

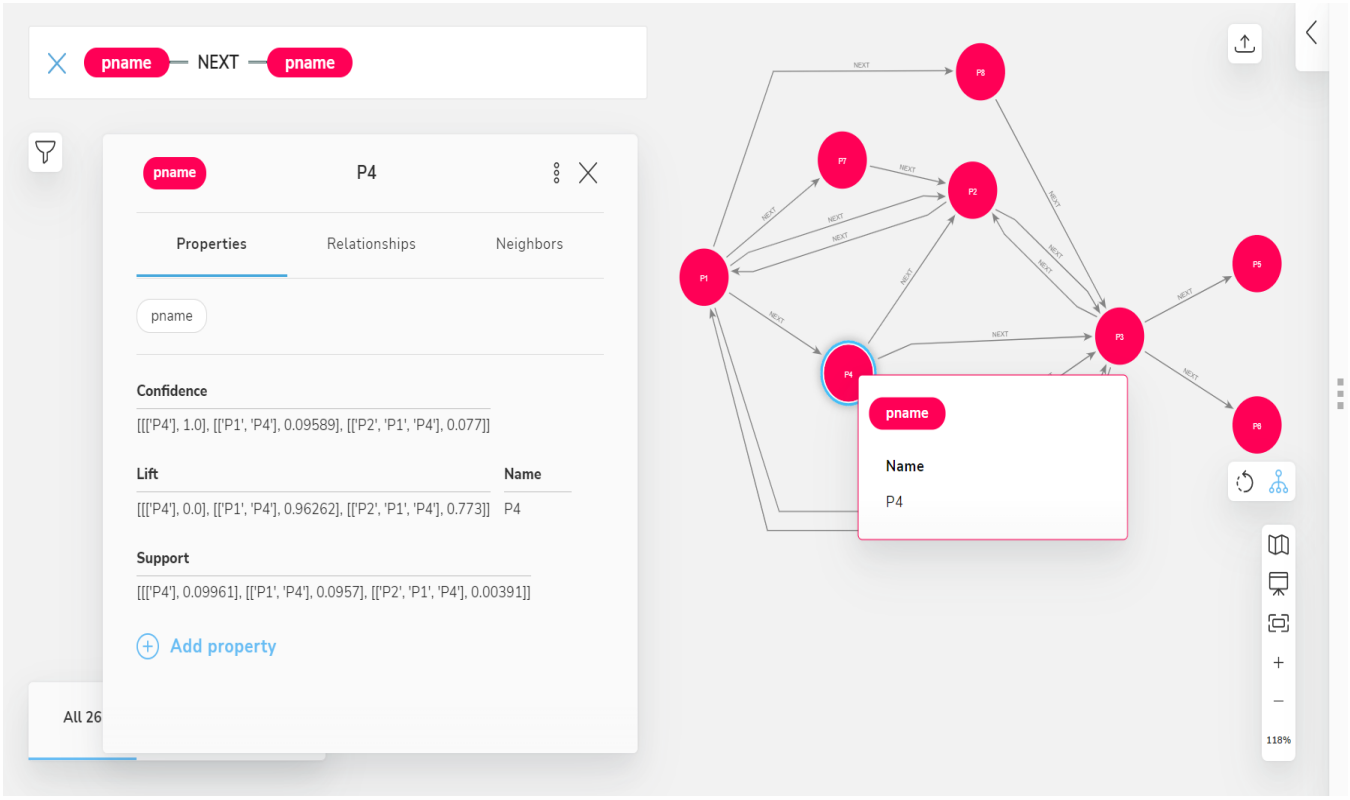


Fig : The image displayed in Neo4j Bloom

By this we can get historical information about that node and we can analyze the importance of the sequence.

The edge is named as NEXT.

The NEXT relation also includes the properties of Support, Confidence, Lift values of the connected nodes, and also the count of the sequence of the connected nodes.

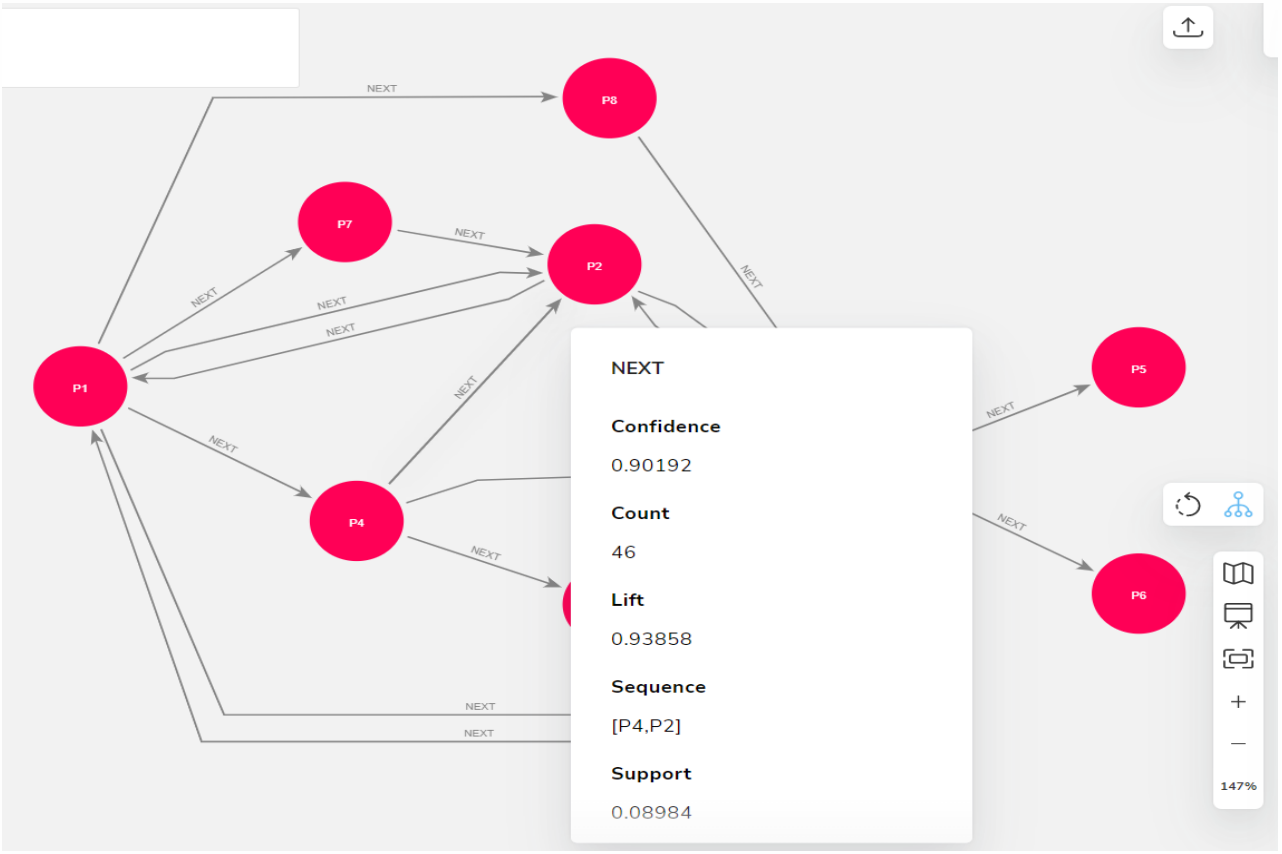


Fig: The image displayed in Neo4j Bloom

Based on Support, Confidence, and Lift, we can proceed to the next procedure by comparing the edges to that node, thus achieving the link prediction using association rule mining techniques.

6.2.5. Node Similarity

The Node Similarity algorithm compares a set of nodes based on the nodes they are connected to. Two nodes are considered similar if they share many of the same neighbors.

We want to use the K-Nearest Neighbors algorithm (kNN) to identify similar patients and base our procedure recommendations on that. In order to be able to leverage topological information about the graph in kNN, we will first create node embeddings using FastRP. These embeddings will then be the input to the kNN algorithm then we can get similar patients.

After creating the graph, we can proceed to project the graph to run algorithms for projection we use Graph Data Science (GDS).
A graph data model that is a projection of the Neo4j property graph data model is used to perform graph algorithms. A graph projection is a materialized view of the stored graph that contains only analytically significant, maybe aggregated topological and property data. Graph projections are totally kept in memory, employing compressed data structures that are optimized for topology and property search.

The graph catalog is a GDS library feature that allows you to manage various graph projections by name. A graph projection, as the name implies, can be utilized numerous times in the analytical workflow. A Native projection or a Cypher projection can be used to project named graphs. To save up main memory, named graphs can be deleted from the catalog after use.

Now we run the FastRP algorithm to generate node embeddings that capture topological information from the graph. We choose to work with embeddingDimension set to 16 which is sufficient since our example graph is very small. The iteration weights are chosen empirically to yield sensible results. Since we want to use the embeddings as input when we run kNN later we use FastRP's mutate mode.

	nodePropertiesWritten
0	2060

The node properties have been created. Now we can run kNN to identify similar nodes by using the node embeddings that we generated with FastRP as nodeProperties. Since we are working with a small graph, we can set sampleRate to 1 and deltaThreshold to 0 without having to worry about long computation times. The concurrency parameter is set to 1 in order to get a deterministic result. Please see the syntax section of the kNN documentation for more information on these parameters. Note that we use the algorithm's write mode to write the properties and relationships back to our database so that we can analyze them later using Cypher. we compute the node similarity by calculating the cosine similarity for the node embeddings creat3ed from the FastRp algorithm in Neo4j.

	Patient1	Patient2	Similarity
0	4711150	4966560	0.929694
1	4711150	4690139	0.906595
2	4711150	5051993	0.895937
3	4711150	5022550	0.894990
4	4711150	4948001	0.894649

Here Mean Similarity is high because of the fact that the sequence (P1, P3, P2) has more occurrences.

nodesCompared	relationshipsWritten	meanSimilarity
2060	16480	0.9043936227131816

	Patient1	Patient2	Similarity	P1_Sequence	P2_Sequence
0	4599476	4842921	0.950378	[1, 3, 2]	[1, 3, 2]
1	4842921	4599476	0.950378	[1, 3, 2]	[1, 3, 2]
2	5117374	4584908	0.947648	[1, 3, 2]	[1, 3, 2]
3	4584908	5117374	0.947648	[1, 3, 2]	[1, 3, 2]
4	4954905	5151996	0.942102	[1, 3, 2]	[1, 3, 2]
5	5151996	4954905	0.942102	[1, 3, 2]	[1, 3, 2]
6	5512621	4948560	0.941249	[1, 3, 2]	[1, 3, 2]
7	4948560	5512621	0.941249	[1, 3, 2]	[1, 3, 2]
8	4366483	4995818	0.941228	[1, 3, 2]	[1, 3, 2]
9	4995818	4366483	0.941228	[1, 3, 2]	[1, 3, 2]
10	5514861	5114352	0.939275	[1, 3, 2]	[1, 3, 2]

We can also see the knowledge graph after forming the relation similarity which patients are similar. This similarity can also be applied to the recommendations.

For example, we took the patient id's "5491367" and "4711150" have procedure sequences (P1, P4, P2) and (P1, P3, P2) respectively.

Here we can see similar patients to the given id's.

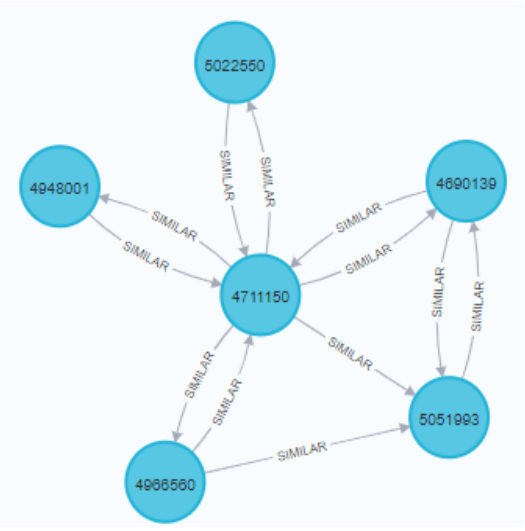


Fig: Node similarity Knowledge Graph

From the Knowledge we can get any patient's information and similarity between the patients. By this knowledge graph we can have visualization and interaction with similar patients.

6.2.6. Centrality Measures

The GDS library in Neo4j implements the centrality algorithm for the knowledge graph.

Page Rank

Based on the number of inbound relationships and the relevance of the respective source nodes, the PageRank algorithm determines the importance of each node within the network.
The GDS library in Neo4j implements the page rank algorithm for the knowledge graph.

PAGE RANKING ALGORITHM

	Procedure	score
0	P3	1.136873
1	P2	0.831220
2	P1	0.744551
3	P5	0.391465
4	P6	0.391465
5	P4	0.276501
6	P7	0.276501
7	P8	0.276501
8	P9	0.228313

Closeness Centrality

Closeness centrality is a way of detecting nodes that are able to spread information very efficiently through a graph. Nodes with a high closeness score have the shortest distances to all other nodes.

	Procedure	Centrality
0	P3	0.687500
1	P2	0.625000
2	P1	0.500000
3	P5	0.479167
4	P6	0.479167
5	P4	0.375000
6	P7	0.375000
7	P8	0.375000
8	P9	0.333333

Authorities and Hubs - HITS (Hyperlink-Induced Topic Search)

HITS is a link analysis algorithm that rates nodes based on two scores, a hub score and an authority score.

The **authority score** estimates the importance of the node within the network.

The **hub score** estimates the value of its relationships to other nodes.

We can see that the authority score is high for the top two values P3, P2, indicating network significance, and the hub score is high for P1, P2, indicating that it has greater value based on its relationships with other nodes.

Based on the problem we can have the specification of which one is more important for our problem.

Procedures ordered based on Authorities scores

	Name	Auth	Hub
0	P3	0.638641	0.362235
1	P2	0.578388	0.297379
2	P1	0.226684	0.647751
3	P4	0.222607	0.474258
4	P7	0.222607	0.198770
5	P8	0.222607	0.219477
6	P9	0.162984	0.219477
7	P5	0.124486	0.000000
8	P6	0.124486	0.000000

Procedures ordered based on Hub scores

	Name	Auth	Hub
0	P1	0.226684	0.647751
1	P4	0.222607	0.474258
2	P3	0.638641	0.362235
3	P2	0.578388	0.297379
4	P8	0.222607	0.219477
5	P9	0.162984	0.219477
6	P7	0.222607	0.198770
7	P5	0.124486	0.000000
8	P6	0.124486	0.000000

6.3. WORK ON SSSIHMS DATASET

6.3.1. Graph Modelling

In our problem, we made

Nodes

I. Patients

II. Encounter

III. Procedures

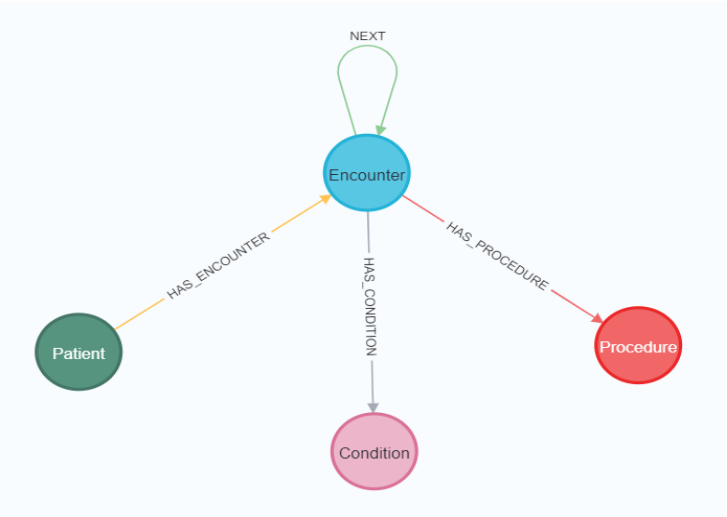
Relations

i. NEXT - relation between two encounters which says sequence of the procedure.

ii. HAS_ENCOUNTER - relation between patient and encounter.

iii. HAS_PROCEDURE - relation between Encounter and Procedure.

Schema



The patient will have similar encounters with the procedure as in the AB-PMJAY dataset. We have also included the patient's condition in this dataset. As a result, each patient's information and encounter information will be linked in the Encounter file via the patient id.

Based on the date and time of each patient's visit, the NEXT relation is arranged in the sequence of encounters.

For matching, the encounter id can be used to connect the procedures.

Conditions also match through the encounter_id.

6.3.2. Workflow

1. Import required libraries

   Here we used pandas and py2neo library.

   Pandas - for working with data frames taken from the Neo4j query.

py2neo - to integrate and run queries in Neo4j.

2. Before integrating the graph database, first we have to create a project in Neo4j desktop and create a new database, and set a password to it.
3. Then will integrate from Jupyter notebook to Neo4j desktop, with the help of a function named "Graph" from py2neo, it takes the arguments such as Neo4j browser link, auth which takes username and password as authentication. U can see the figure below.

Integrating with Neo4j using py2neo

```
graph = Graph('bolt://localhost:7687', auth=('neo4j', 'nextproject'))
```

4. After integrating with Neo4j, the next step is to load our data into the Neo4j desktop in order to get the knowledge graphs and retrieve some insights from the data.

5. The Maharashtra dataset is divided into three csv files: Patients, Procedures, and Encounters.
   This separation of the main csv file was done through graph modeling.
   I. *Patient csv file* contains the information only about patients such as Patient_Id, Age and Gender.
   II. *Procedures csv file* contains Patient_Id, Procedure (Name of the procedure), Encounter_Id(Id given for each beneficiary when he/she encountered the procedure) and Date on which he/she underwent the procedure.
   III. *Encounter csv file* contains Encounter_Id(Id given for each beneficiary when he/she encountered the procedure) and Date on which he/she underwent the procedure,Patient_Id.
   Here *Patient_Id* and *Encounter_Id* play a vital role in mapping or creating relationships between the nodes.

6. Before loading CSV files we have to create constraints or labels for each CSV file.
   Constraints ensure that property values are unique for all nodes with a specific label.
   So we created three constraints for the patient, procedure and encounter.

7. First we loaded the patient CSV file into Neo4j, created a node for each and every row in the CSV file, and set the properties accordingly.
   After loading the CSV file, we will get the following output which gives us the statistics like how many nodes were created, relations created and properties set using the stats() function in py2neo.

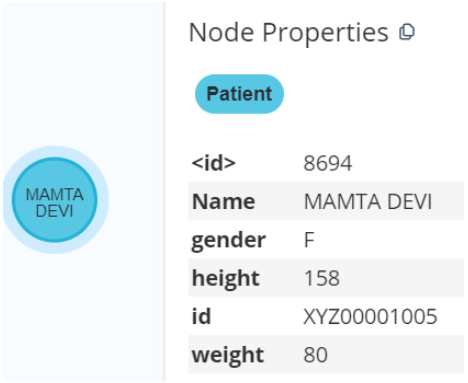


Fig: Representation of node and its properties in Neo4j Desktop.

8. Then we loaded the Encounter CSV file, in addition, we created the relationship called "HAS_ENCOUNTER" between patient and encounter nodes as we discussed in graph modeling. After loading the CSV file, you can see the stats below:

```
{'labels_added': 15501,
 'relationships_created': 8694,
 'nodes_created': 15501,
 'properties_set': 32889}
```

Fig: Stats after Encounter file loaded into Neo4j.

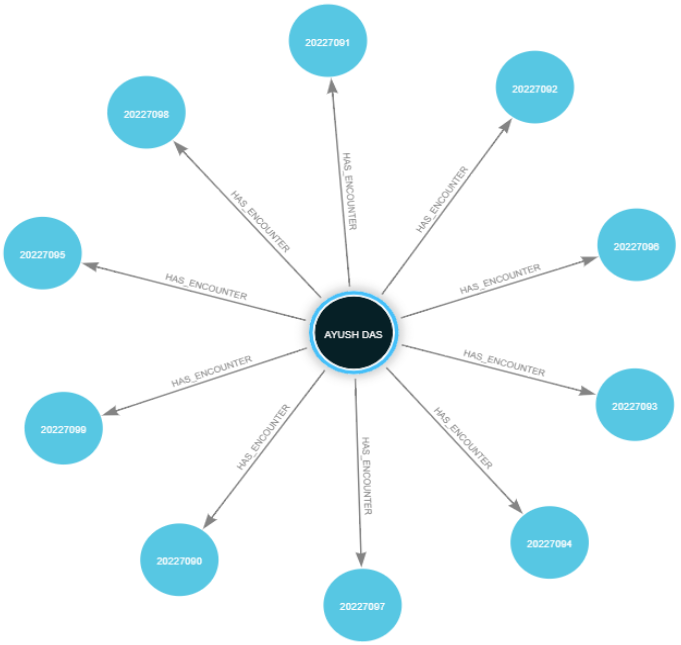


Fig: Representation of relation between patient and Encounter.

9. It shows Patient with Id "555296" has encountered three procedures.
10. Finally, we loaded the Procedure CSV file and created the relationship called " HAS_PROCEDURE" between Encounter and Procedure nodes as we discussed in graph modeling. After loading the CSV file, you can see the stats below:

```
{'labels_added': 367,
 'relationships_created': 8694,
 'nodes_created': 367,
 'properties_set': 43837}
```

Fig : Stats after Procedure file loaded.

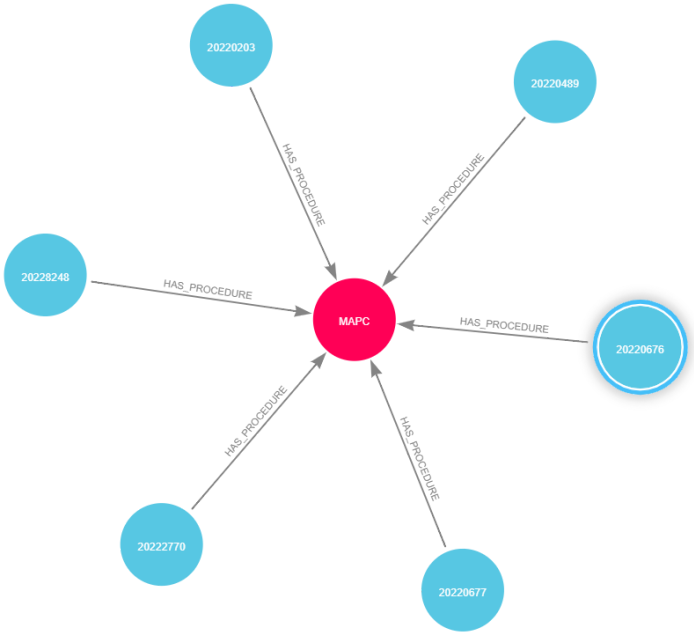


Fig : Representation of HAS_PROCEDURE relation between Encounter and Procedure.

11. Then we finally created the NEXT relationship between Encounters.

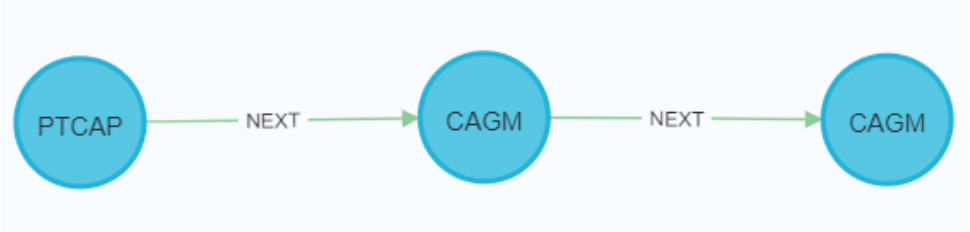


Fig : Representation of NEXT relation between Encounter and Procedure.

12. Finally, we want to combine all these and show you a knowledge graph of a single patient.

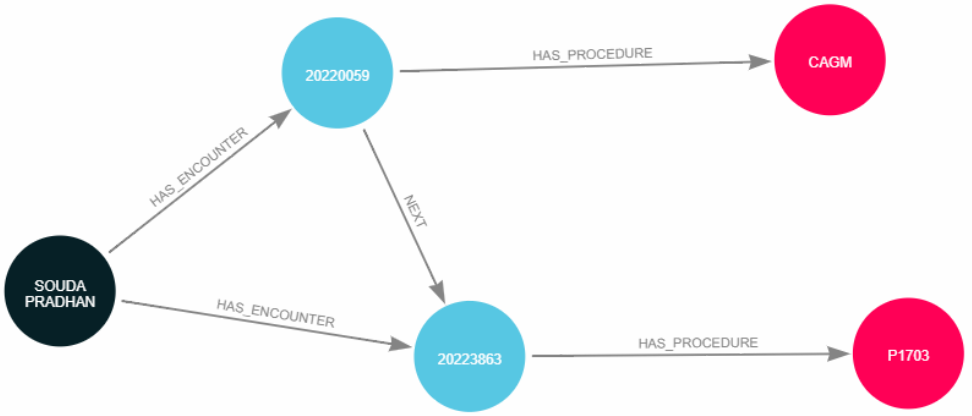
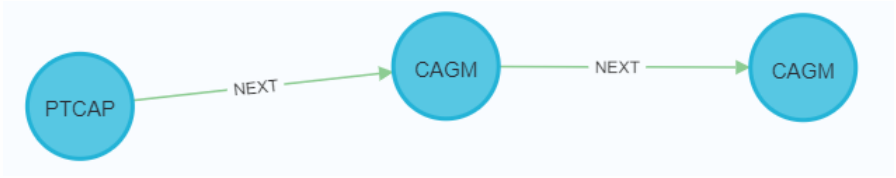


Fig : Knowledge graph of a single patient

6.3.3. Insights from the data

1. The below is the treatment sequence of a beneficiary with only procedures involved.



2. The below table shows us the entire treatment sequence followed by a single patient.

Table | Text | Code

	Patient	sequence
1	"DIBYENDU LAHA"	["EPS", "RFA", "EPS", "RFA", "CAGM", "P1703"]

Started streaming 1 records after 57 ms and completed after 67 ms.

3. In this table, we can see all the treatment sequences followed by every patient.

Table | Text | Code

	Patient_Name	Treatment_Sequence
1	"RENJITH R PAI "	["CAGM", "EPS", "RFA"]
2	"AJAY SAH"	["CAGM", "PAGM", "PTAAB"]
3	"NIRANJAN DEY "	["CAGM", "A203"]
4	"RUDRARAJU ADITYA VARMA"	["CAGM", "ICDI"]
5	"S MAIMUNNISA BEGUM"	["CAGM", "PTCAP"]
6	"AJIJUL SEIKH"	["CAGM", "PTCAP"]
7	"JAYASANKAR KENGAN"	["CAGM"]

Started streaming 6550 records after 43 ms and completed after 45 ms, displaying first 1000 rows.

4. In this table, we can see the procedure called RFA involved in every treatment sequence.

Table | Text | Code

	RFA_Involving_Sequence	Involving_Count
1	["RFA"]	265
2	["RFA", "EPS"]	126
3	["RFA", "EPS", "RFA"]	6
4	["RFA", "EPS", "RFA", "CAGM"]	2
5	["RFA", "EPS", "RFA", "CAGM", "P1703"]	1
6	["RFA", "CAGM"]	3
7	["RFA", "CAGM", "P1703"]	1

Started streaming 26 records after 14 ms and completed after 31 ms.

5. The below table shows us what all treatment sequences started with procedure RFA and its count.

Table | Text | Code

	RFS_Start_Sequence	Count
1	["RFA"]	137
2	["RFA", "EPS"]	18
3	["RFA", "EPS", "RFA"]	2
4	["RFA", "EPS", "RFA", "CAGM"]	2
5	["RFA", "CAGM"]	2
6	["RFA", "EPS", "EPS"]	4
7	["RFA", "EPS", "RFA", "CAGM", "PTCAP"]	1

Started streaming 12 records after 12 ms and completed after 20 ms.

6. In this table, we can see all the single procedures present in our dataset and their respective count. There are 367 single procedures present in our dataset.

Table | Text | Code

	Single_Procedure	Count
1	["CAGM"]	933
2	["PTCAP"]	924
3	["P103"]	335
4	["EPS"]	296
5	["PTMCM"]	288
6	["RFA"]	265
7	["ASDDV"]	253

Started streaming 367 records after 2 ms and completed after 124 ms.

6.3.4. Link Prediction with the help of Association Rule Mining

The identical technique was applied to this dataset as it was to the prior dataset, and the results are shown below.

	Procedures	Support	Confidence	Lift
0	[CAGM]	0.13706	1.00000	0.00000
1	[PTCAP]	0.13574	1.00000	0.00000
2	[P103]	0.04921	1.00000	0.00000
3	[EPS]	0.04348	1.00000	0.00000
4	[PTMCM]	0.04231	1.00000	0.00000
...	...	...	...	...
1444	[UROPER7, UROPER3]	0.00015	0.34091	288.90601
1445	[UROPER7, UROPER3, UROPER2]	0.00015	1.00000	295.85799
1446	[UROPER18]	0.00015	1.00000	0.00000
1447	[UROPER1, OBGOPER27]	0.00015	0.20548	35.86029
1448	[UROPER6, UROPER6]	0.00015	0.09259	57.15592

1449 rows × 4 columns

We use this knowledge of data frame to create a knowledge graph.

To create a knowledge graph, we need to create nodes for all the procedures and edges.
Procedures and their subsequent sequences make up the knowledge graph.

We construct a CQL query along with the edge count to retrieve the nodes that need to be linked.

Get the edges that must be linked, as well as the edge count between them.

Now we create the edges between them.

We got the knowledge graphs with the sequences followed in the databases.

Procedure1	Procedure2	count
EPS	RFA	146
CAGM	PTCAP	145
PTCAP	CAGM	141
RFA	EPS	126
CAGM	A203	63
...	...	...
UROPER29	OBGOPER27	1
UROPER5	UROPER7	1
UROPER7	UROPER3	1
UROPER1	OBGOPER27	1
UROPER6	UROPER6	1

Now the task goes like this: we need to get the properties for each node, consisting of Support, Confidence, and Lift values of the sequences followed until that node.

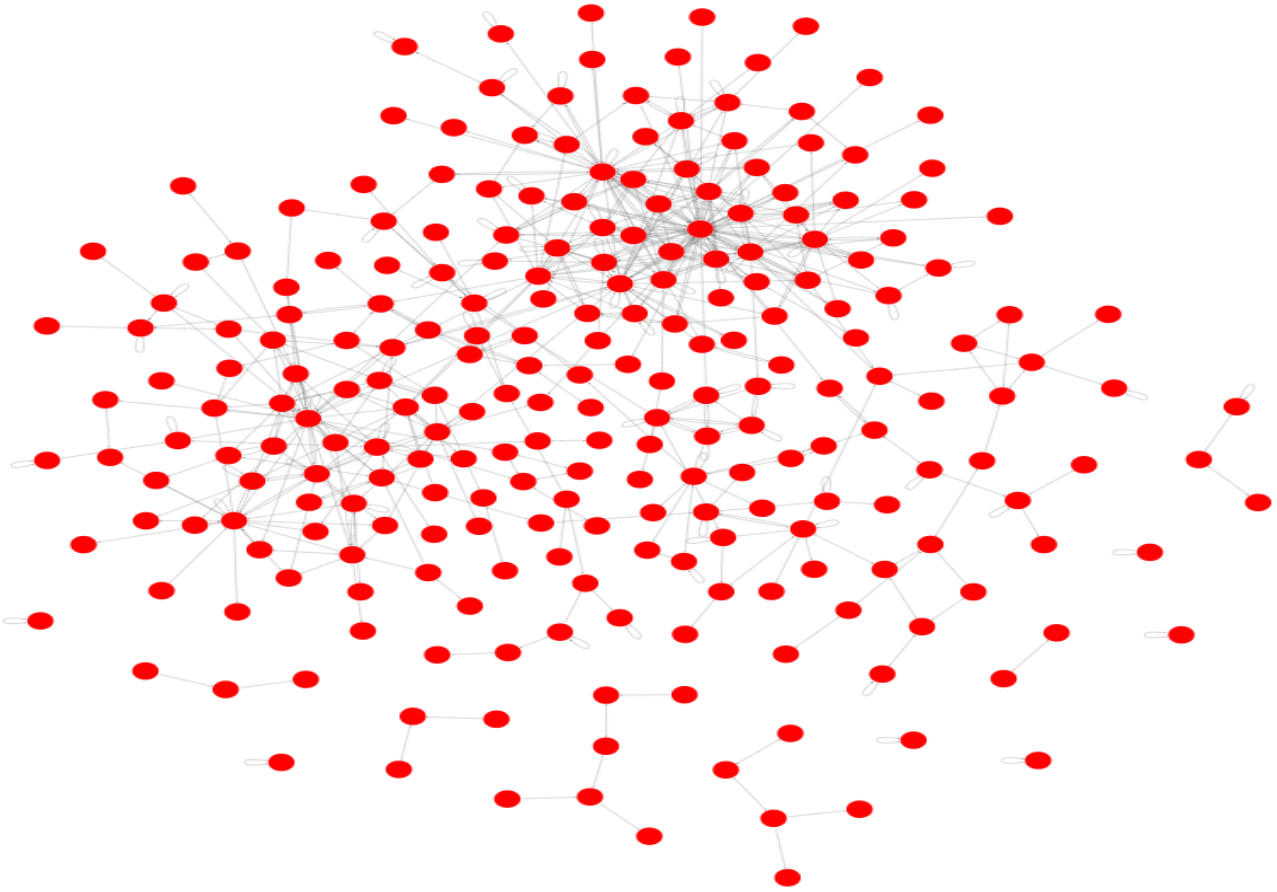
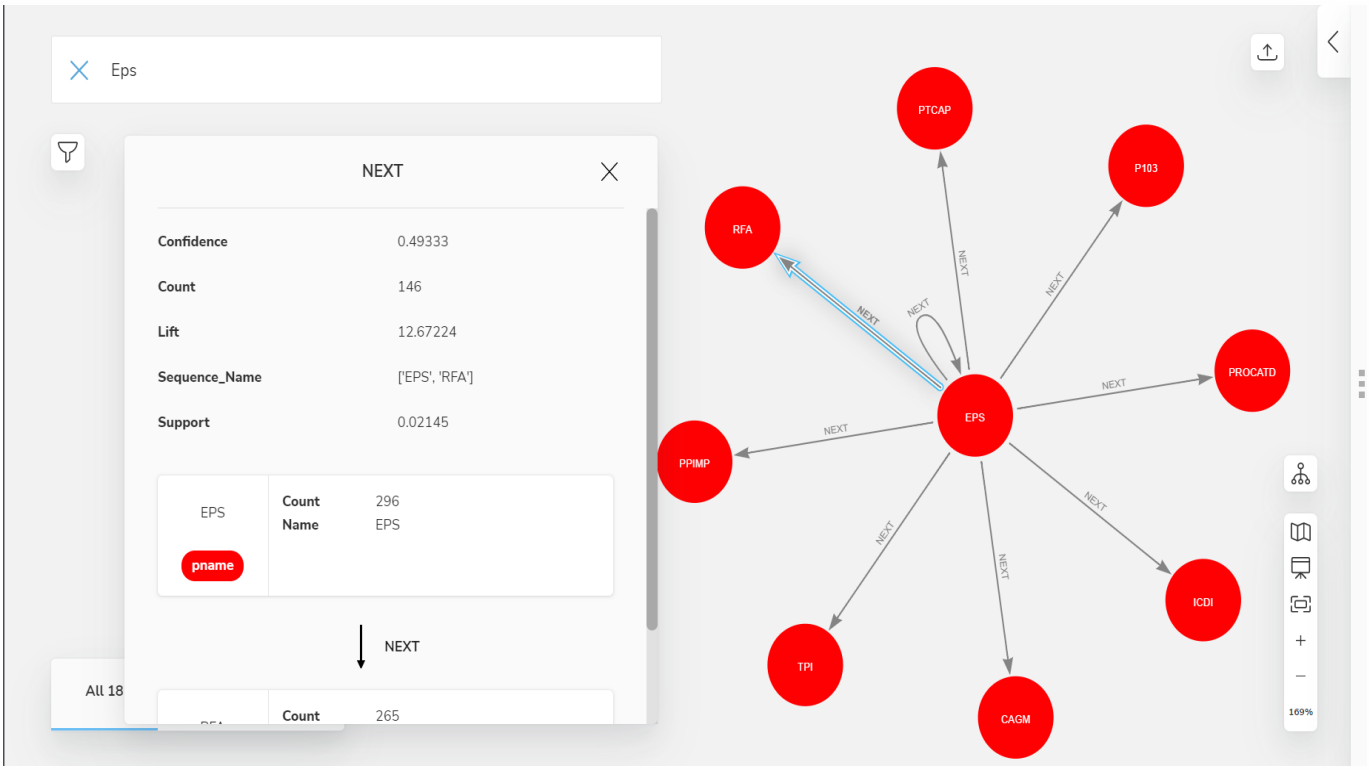


Fig: Entire knowledge graph after finding support, confidence, and Lift.

We used one procedure as an example in the diagram below to describe what's going on.
Here I took the procedure called “EPA”, we can see here eight procedures and EPA itself coming as the next procedure. Here, if we see the NEXT relation properties, it will contain properties like support, confidence,, and lift, which helps identify the next procedure.

We compare which procedure has the highest confidence, support, and Lift then we select that as the following procedure accordingly.

The below part of the knowledge graph helps us in identifying the following procedure.



6.3.5. Node Similarity

Similar to the previous dataset, we compute the node similarity by calculating the cosine similarity for the node embeddings creat3ed from the fastrp algorithm in Neo4j.

	Patient1	Patient2	Similarity
0	SUSANTA KUMAR DAS	SALMA BEGAM	1.000000
1	SALMA BEGAM	SUSANTA KUMAR DAS	1.000000
2	JOSEPH	SUMITRA CHALAK	0.973823
3	SUMITRA CHALAK	JOSEPH	0.973823
4	SHANTHALA R	SUKHEN BAGDI	0.971920
...	...	...	...
20495	PRIYANGSHU KUMAR	USMAN KHAN	0.534594
20496	PRIYANGSHU KUMAR	BARSHA KARMAKAR	0.532918
20497	KHOKU MANI ADHIKARY	MATHIVANAN S	0.530098
20498	SHASHIKALA.T.G	PRASNNA KUMAR MV	0.529070
20499	SHASHIKALA.T.G	VIDYA K R	0.526456

20500 rows × 3 columns

6.3.6. Centrality Measures

Page Rank

Based on the number of inbound relationships and the relevance of the respective source nodes, the PageRank algorithm determines the importance of each node within the network

Similar to the previous data now we got the importance of each node in this SSSIHMS data set.
We are displaying the top 10 values.

PAGE RANKING ALGORITHM

	Procedure	score
0	MINREEX	4.970530
1	P2437	4.306431
2	RESUT	4.222734
3	UROPER31	3.840624
4	VPSH	3.556806
5	CAGM	3.244598
6	REEXDE	2.859864
7	BHEVD	2.686954
8	ENTOPER16	2.457795
9	GSOPER35	2.446125

Closeness Centrality

Closeness centrality is a way of detecting nodes that are able to spread information very efficiently through a graph. Nodes with a high closeness score have the shortest distances to all other nodes.

We are displaying the top 10 values.

CLOSENESS CENTRALITY

	Procedure	Centrality
0	P2437	0.173776
1	MINREEX	0.170574
2	CAGM	0.156359
3	TPI	0.146274
4	RESUT	0.137619
5	31600	0.136236
6	PAGM	0.134414
7	BHEVD	0.132337
8	P2429	0.131227
9	REEXDE	0.130106

Authorities and Hubs - HITS(Hyperbolic-Indexed Topic Search)

HITS is a link analysis algorithm that rates nodes based on two scores, a hub score and an authority score.

The **authority score** estimates the importance of the node within the network.

The **hub score** estimates the value of its relationships to other nodes.

Procedures ordered based on Authority scores

	Name	Auth	Hub
0	CAGM	0.45317	0.57576
1	P2437	0.31159	0.11674
2	P103	0.23822	0.13868
3	LRHCATH	0.21189	0.40439
4	PAGM	0.20155	0.19416
5	TPI	0.19517	0.17669
6	PTCAP	0.19424	0.21365
7	RFA	0.18484	0.15257
8	ASDDV	0.18050	0.14135
9	EPS	0.17853	0.20594

Procedures ordered based on Hub scores

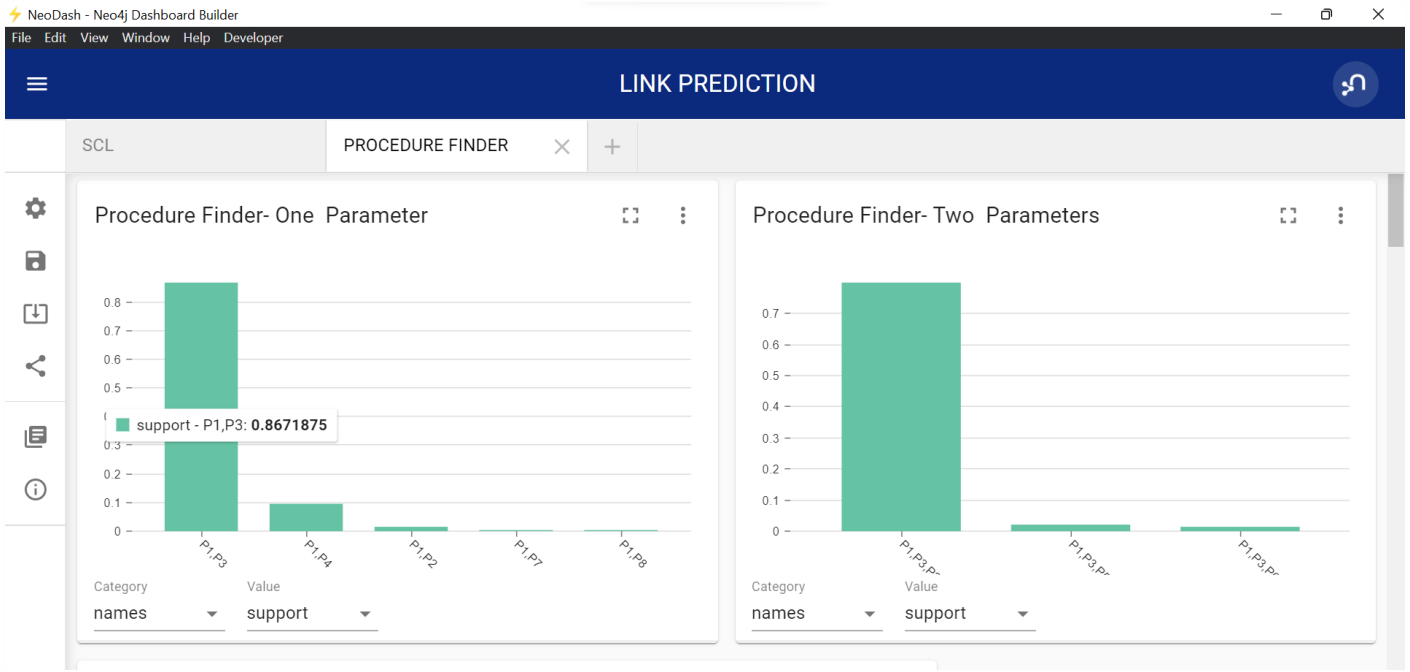
	Name	Auth	Hub
0	CAGM	0.45317	0.57576
1	LRHCATH	0.21189	0.40439
2	PTCAP	0.19424	0.21365
3	EPS	0.17853	0.20594
4	PAGM	0.20155	0.19416
5	PROCATD	0.16367	0.17891
6	TPI	0.19517	0.17669
7	A203	0.13211	0.16047
8	RFA	0.18484	0.15257
9	PVBDL	0.01556	0.14529

CHAPTER-7: DASH BOARD

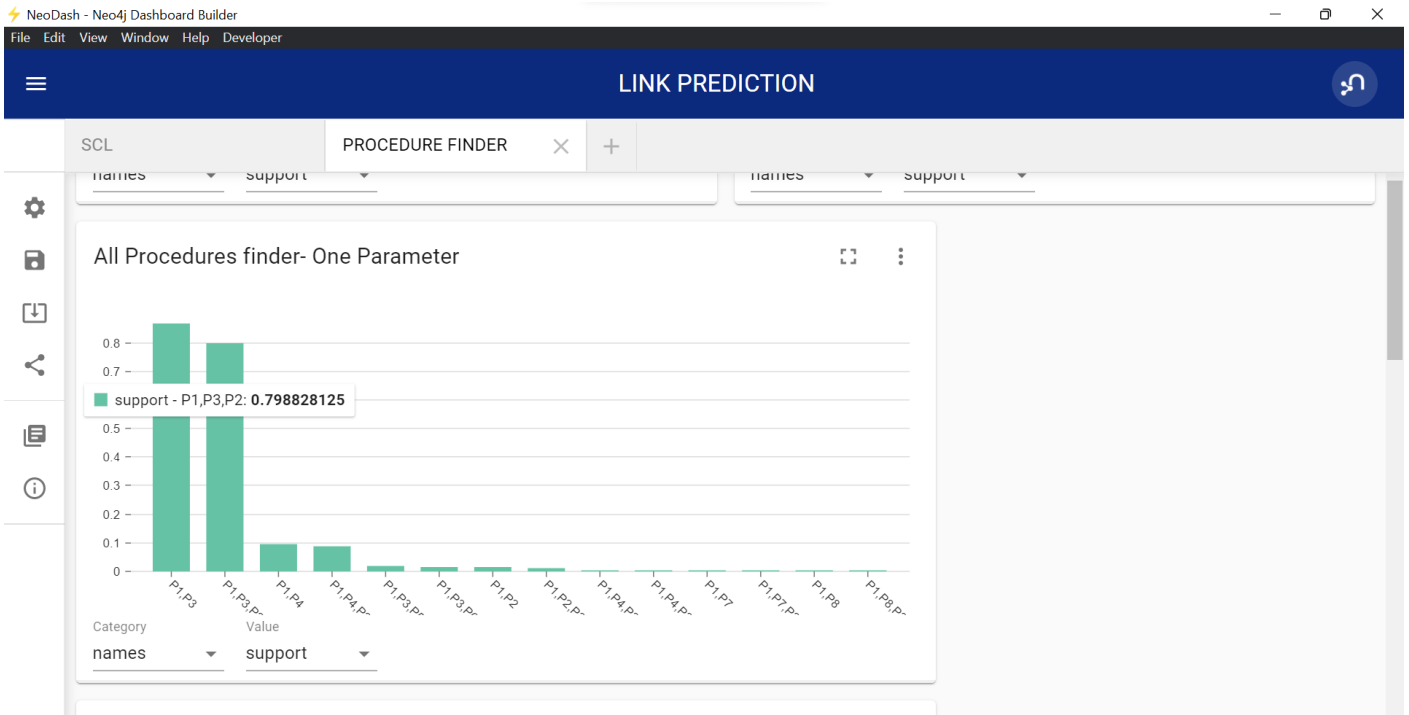
Neo4j Dashboard - NeoDash:

We have created a dashboard using NeoDash by providing a query.It allows us to select the type of visualization we require.

Based on the values of support and Count we can go with the next procedure.

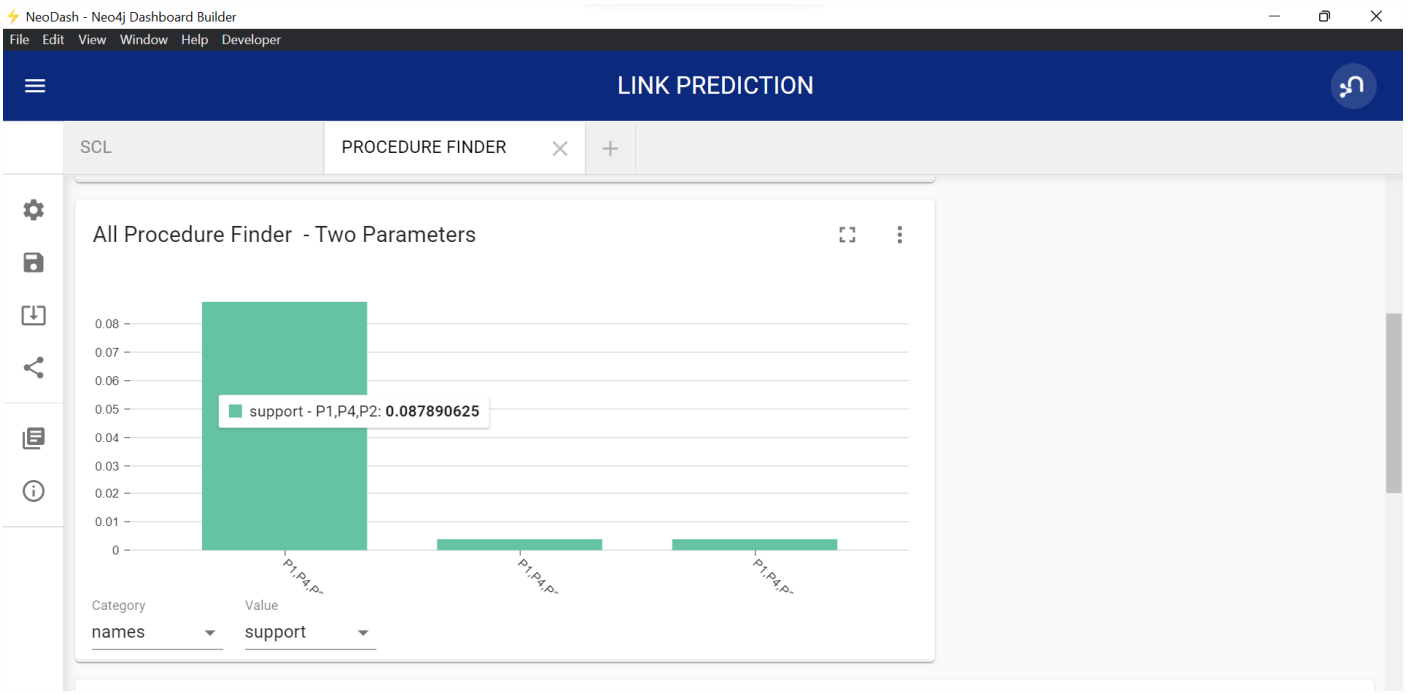


Procedure Finder- One Parameter tells that, given a single procedure analyze the next procedure.

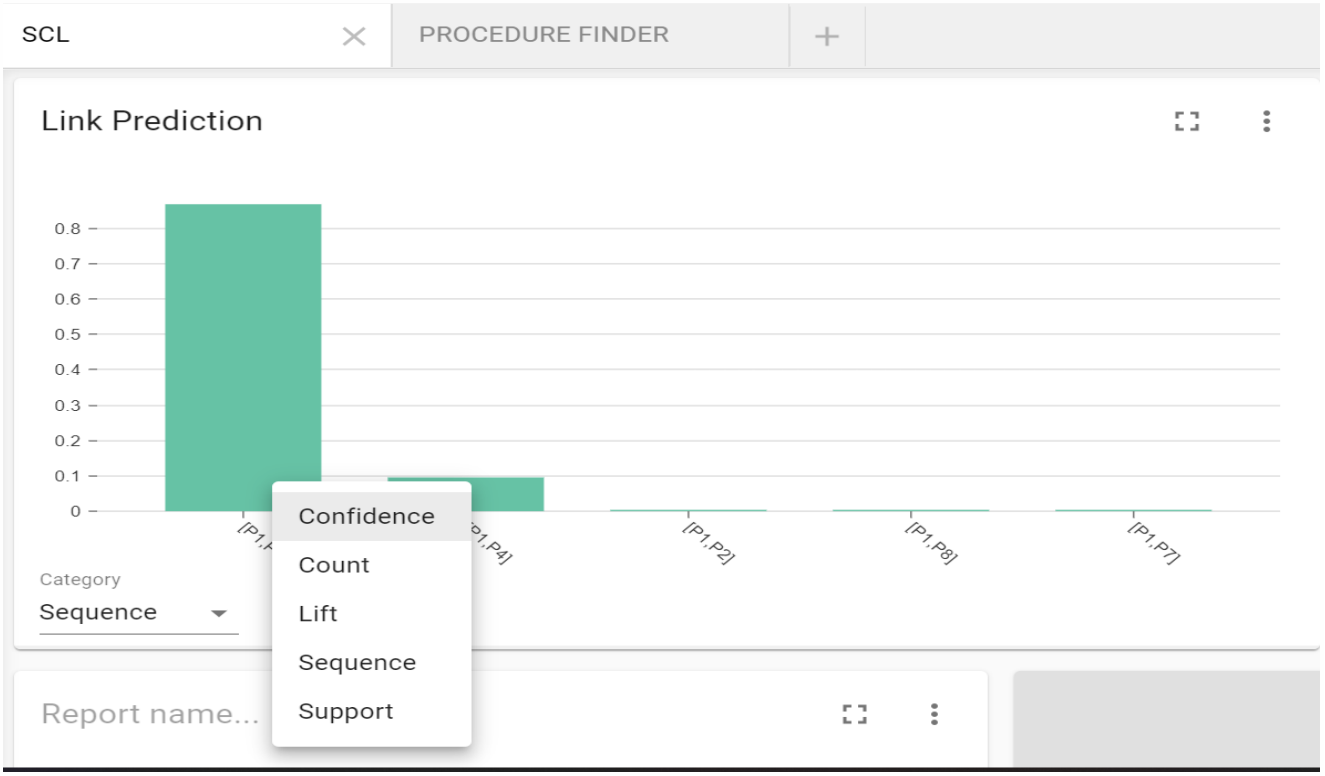


Procedure Finder - Two Parameter tells that, given two procedures analyze the next procedure.

Given one parameter we can also give all procedures that are being followed after that in a descending fashion of the bar chart.



We can find all the processes that were followed once two parameters/procedures are given.



We can also find the Values of the Support, Confidence, Lift and count values by selecting dropdown and hovering on the bars.

CHAPTER-8: FUTURE SCOPE

The need for activity sequence analysis in the medical domain has been a challenge in recent times. With the help of graph databases using Neo4j, the resulting graphs have been the focus of this work which are obtained using the machine learning algorithms like kNN.

The current work can be enhanced further by making available the cost features and drug features into the graph databases thus realizing cost-efficient sequences and treatment-efficient sequences which can then be integrated into Neo4j for carrying out cost-benefit analysis.

CHAPTER-9: CONCLUSION

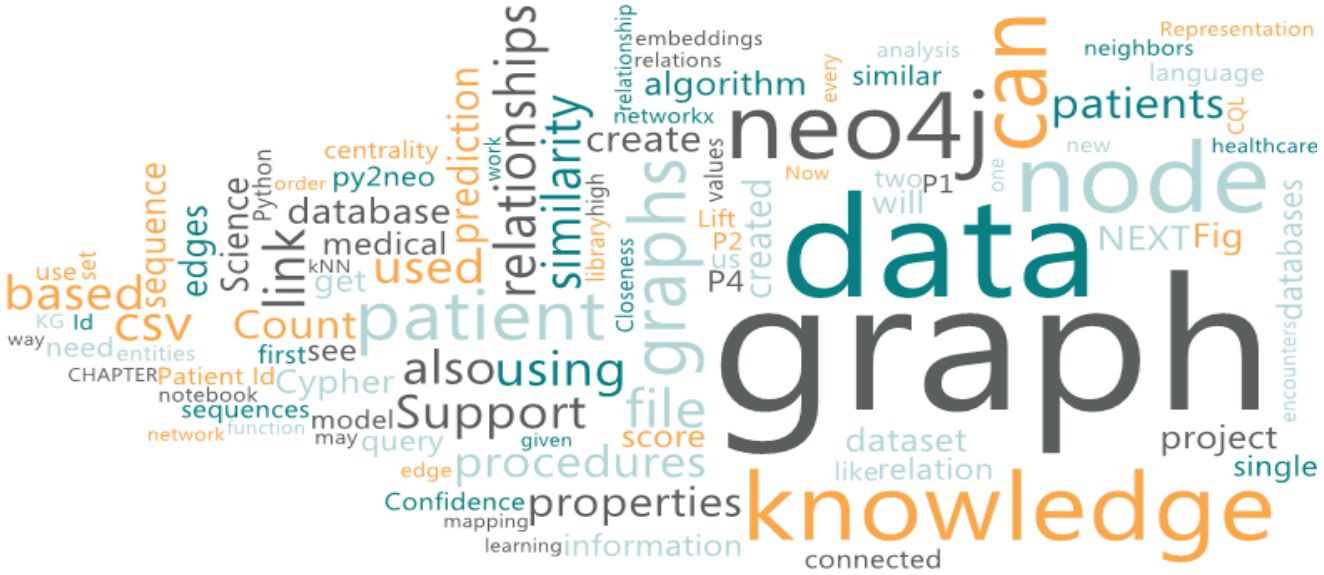
The initiative is highly important in a real-world setting since there are many places in hospitals these days where physicians are unnecessarily proposing operations to be undertaken by patients for financial gain, and people who are unaware of these things are becoming witnesses to these sorts of scams.

We expect that as a consequence of this initiative, patients will undergo just the operations that are necessary, saving time and money. Another benefit is that it will allow doctors and medical personnel to plan for the next surgery without having to wait for the results of the previous procedure. For the physicians, it will be really beneficial.

So in this way, this is also one more benefit for both the patients and also for the doctor and his medical team and this project also helped us in learning in-depth about the Neo4j graph databases, graph data science library, pandas, excel, and mainly Cypher Query Language (CQL). Association rule mining, Statistics, and mathematics were involved in our project.

We have learned the techniques of node embeddings and how it is calculated for the graphs, Embeddings play a key role in machine learning in which a lot of parameters have been considered for each node. We also applied the kNN machine learning algorithm to cluster those embeddings.

Also, about the Jupyter notebook and integrating it with the Neo4j desktop for querying, because of the above things we are able to finish our project on time and also earned a lot of knowledge about the Machine Learning concepts and also the Mathematical Concepts which are used in the Association Rule Mining, some similarity measures and centrality measures like Page rank, Closeness and about HITS.



Common Words in the Project

CHAPTER-10: REFERENCES

[1] Neo4j Documentation

[2] Bite-Sized Neo4j for Data Scientists

[3] Real-world data medical knowledge graph: construction and applications | Request PDF - Feb 2020

[4] ELPKG-A High Accuracy Link Prediction Approach for Knowledge Graph Completion - Sep 2019

[5] (PDF) From Tabulated Data to Knowledge Graph: A Novel Way of Improving the Performance of the Classification Models in the Healthcare Data (researchgate.net). - June 2021

[6] Table2Graph: A Scalable Graph Construction from Relational Tables Using Map-Reduce | IEEE Conference Publication | IEEE Xplore - Aug 2015

[7] (PDF) A Concept for Graph-Based Temporal Similarity of Patient Data - Aug 2019

[8] https://medium.com/Neo4jhttps://stackoverflow.com/

Images:

[1] Graph Data Platforms: Collections vs. Connections

[2] https://rubygarage.org/blog/Neo4j-database-guide-with-use-cases