Detecting possible frauds in relevant domains using GANs and KG.

A Project/Dissertation as a Course requirement for

Master of Science

(Data Science and Computing)

Deepam Rai

21231

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING

(Deemed to be University)

Department of Mathematics and Computer Science

Muddenahalli Campus

April 2023

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

Dept. of **Mathematics & Computer Science**
Muddenahalli Campus

CERTIFICATE

This is to certify that this Project/Dissertation titled **Detecting possible frauds in relevant domains using GANs and KG.** submitted by Deepam Rai, 21231, Department of Mathematics and Computer Science, Muddenahalli Campus is a bonafide record of the original work done under my/our supervision as a Course requirement for the Degree of M.Sc. in Data Science and Computing.

Place: Muddenahalli
Date: 25/04/2023

……..
Sri. P. Sunil Kumar
Project / Dissertation Supervisor

Countersigned by

……
Dr. (Ms.) Y Lakshmi Naidu
Head of the Department

DECLARATION

The Dissertation titled "Detecting possible frauds in relevant domains using GANs and KG" 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 M.Sc. in 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.

Deepam Rai.

……..

Place : Muddenahalli

Deepam Rai
21231
II M.Sc. Data Science and Computing
Muddenahalli Campus

Date : 25/04/2023

Acknowledgement

I would like to express my gratitude to Bhagwan Sri Sathya Sai Baba and my institute SSSIHL for giving me the opportunity to experience a one year project that would prepare us for the working environment outside the academic institute.

I thank my guide Sri. P. Sunil Kumar for providing us with PM-JAY domain knowledge and resources required for the project. His timely feedback and constructive comments has helped keep the project on track. I am also deeply grateful to brother Anil Kumar Reddy for guiding us through the landscape of tools and techniques and helping us in every step. I would also like to express my gratitude to Sri P. V. S. Prakash who provided us with an extensive workshop on Deep Learning owing to which we could explore the deep learning field for the project.

I would also like to thank my brothers Narendra Kumar Reddy and Thunga Sri Hari with whom I completed this three part project and who have been constantly motivating throughout the project.

I would again like to express my immense gratitude to Swami for giving us such an opportunity and being always with us throughout.

Content

Abstract

Advancement in technology has brought information and immense computation power to our fingertips. But access to immense information doesn't necessarily mean that the masses will be updated and informed, rather they are being misinformed and misled greatly. The deficiency of understanding and analyzing of these informations has facilitated the commitment of fraudulent activities. The project of which this particular project is the final part aims to take a step against this issue by providing the information to masses in easily comprehensible form of knowledge graphs and utilizing the same for detection of frauds in the real world data. To achieve this the project focuses on PM-JAY, the world's largest social healthcare scheme launched by the government of India.

This final part proposes an expert-system approach for fraud detection in the PM-JAY scheme. This merges rule-based with model-based fraud detection, getting best of both. Initially, the rules for rule-based are provided by the domain experts. The model is trained on detected frauds and an additional gain is that the model can adapt and learn to the fraud patterns which are indiscernible in rule based systems. And further its output can supplement the rule-based system further. The experimentations and the results of this project shows that such a system can be implemented for PM-JAY. Graph Neural Network model has been trained on the real PM-JAY data which has achieved considerable performance. Generative Adversarial Networks have been used to develop synthetic fraud data and also non-fraud data where the data was less in proportion or was not allowed to be publicly exposed. Since GANs learn the distribution of true data, this generated data too follows real data as closely as possible. Further an open source python library PyNeFrauds has been developed to help automate major parts of the process.

1. Introduction

1.1 Motivation

“23,000 fraudulent transaction were recorded at empanelled hospitals under the AB-PMJAY in 2021-22”

~Minister of State for Health Bharati Pravin Pawar

171 hospitals were de-empanelled due to fraudulent practices and transactions in 2020.

Above are the news headlines relating to the PM-JAY health scheme of India. The fraudulent activities took place despite continuous efforts of the Anti Fraud Task Force set up by NHA and SHA. This project aims to add more fraud detection literature and tools in the world's largest health insurance scheme Pradhan Mantri - Jan Arogya Yojana launched by the Indian government.

This is not a standalone project but the final part of a three part project. The first part generates knowledge graphs from the government documents, the second part merges the knowledge graphs generated from multiple documents. And my part is to leverage this result knowledge graph to detect possible frauds in the data.

1.2 Expert-System Framework

This project proposes an expert-system framework for fraud detection in PM-JAY data.

I. Rule Based System

This part leverages knowledge of domain experts in the PM-JAY scheme who provide the rules and patterns in the data based upon which frauds can be detected. The rules may be based upon past fraud data or scheme rules. The domain experts will form CYPHER queries which will be run on neo4j database.

II. Model Based System

The model based systems learn from the frauds detected in rule-based systems and also from the unexpected fraud reports. Models adapt and learn these new patterns and further might even learn new patterns altogether. This is a great advantage in a constantly changing environment.

Merging rule-based and model-based

The fraud data generated from rule-based system is used to train model-based system. Thus the rule-based system reinforces the model-based system. Further model-based system also learn

from reported fraud cases as such it learns new patterns and rules. The new frauds detected by the model-based system are investigated and reviewed by domain experts. The newly found rules and patterns are used to further strengthen the rule-based system. This framework uses the feedback and reported frauds and thus prevents the same fraud from recurring.

1.3 Significance

Definition of fraud under PM-JAY:

"intentional deception, manipulation of facts and / or documents or misrepresentation made by a person or organization with the knowledge that the deception could result in unauthorized financial or other benefit to herself/himself or some other person or organization. It includes any act that may constitute fraud under any applicable law in India."

~ Anti-Fraud PM-JAY Guidelines released by NHA

PM-JAY offers health coverage of ₹5 lakhs to more than 10,00,00,000 beneficiary families. Fraudulent activities in such a huge scale scheme would have significant negative impact at all levels. The beneficiaries would be affected the most.

The anti-fraud PM-JAY Guidelines released by National Health Agency(NHA) has set a fraud management mechanism with a zero tolerance approach. It states guidelines to be followed by EHCP, SHA and other agencies, units set up by the state government. It covers all three stages of fraud management approaches - prevention, detection and deterrence. This project aims to supplement the detection stage. The outputs can be further used to reinforce deterrence and prevention of fraudulent activities. The guideline further categorizes the fraud as Beneficiary fraud, payer fraud, provider fraud, empanelment related fraud and claims related fraud. This project has capacity to aid in detecting all these three categories given sufficient data. As per the available data the project has worked for claims related fraud only.

The IT system set up by NHA and SHA have set up triggers for certain fraudulent data entries but for obvious reasons these triggers cant cover all the cases. Further the triggers are set for fraud only and not for error, waste or abuse. Human error, waste and abuse of resources does not come under fraud - as stated in anti-fraud guidelines by NHA. This project can set up queries and models can be trained to detect these errors, waste and abuse alongside fraud. The IT systems are also expected to support basic rule-based and outlier-based analytics. But with every fraud case we see unexpected rules and analytics that could have been set up. The model-based system part of the project can mitigate such hindsight cases. This project has a retrospective approach towards fraud detection, that is, it works mostly based upon historical fraud data except for the initial times when domain experts set up the rules. Prospective approach can be taken by updation of rule continuously by the domain experts.

This project can also be used by the contracted agencies of SHA to have their own anti-fraud management systems.

The anti-fraud guideline states that overtime NHA may make use of AI and ML techniques for fraud detection. It is evident that this is inevitable. This project is a tendril towards it.

1.4 Contribution

We propose the use of an expert-system framework for fraud detection in PM-JAY. This project serves as proof of work that such a system can indeed be implemented. Further the open source PyNeFrauds library developed helps us automate many tasks in the process. This library code is publicly available in github and is also published in PyPI(Python Package Index) thus it can be installed using pip.

Also we show that we can use GANs to generate high fidelity synthetic data for PM-JAY. We show this by using CTGAN to generate synthetic data which has given considerable results in our case.

The Anti-Fraud Guidelines mentions using transaction triggers, data analysis, machine learning and deep learning models for fraud detection by Anti Fraud cells of the PM-JAY scheme. Thus though not explicitly, this guidelines does indirectly hint at the use of expert-system frameworks by government anti-fraud cells. Out there there exists systems that support rule based methods such as generation of CYPHERs, using them for fraud detection, integration with deep learning models, etc but they are either separate with no relation to each other or if they are united then they are available only as commercial software with huge costs. There is no open source library that integrates neo4j, CYPHER query and deep learning specifically models for fraud detection. This gap is covered by our library PyNeFrauds which, though in development state, offers several working modules that can be practically used.

2. Literature Review

2.1 Scope of the review

We surveyed the existing methods used for fraud detection, key concepts and tools used, the new technologies that show promising results but not yet used to full potential in the industry owing to myriad reasons. Further we also reviewed the academic papers and Guidelines on PM-JAY. The guidelines are issued by the National Health Authority of India in its official portal as well as official PM-JAY site.

2.2 Knowledge Graphs

The open course *Knowledge Graphs by Prof. Dr. Harald Sack, Dr. Mehwish Alam*[18] and book *The Rise of the Knowledge Graph*[10] provided a good introduction to knowledge graphs. The parts of course *CS224W: Machine Learning with Graphs* [16] also proved valuable in understanding the application of Machine Learning in Knowledge Graphs. The *KG Seminar* [19] by Stanford introduced many real life use-cases.

2.2.1 Ontology

The first and foremost requirement was the organization of the entities and ideas of the PM-JAY scheme. This demanded ontology for entities in the PM-JAY scheme guidelines. This ontology is developed in the previous phases of the project. In this final phase only the selected part of the ontology was taken and used.

2.3 PM-JAY Domain

An Analytical Model for Evaluating Social Security Schemes-A Focus on "Ayushman Bharat" Universal Health Scheme in India [6] gave an understanding on the PM-JAY domain aside from the available documents at the PM-JAY website. The Anti Fraud Guidelines[3] and enlisting of Health Benefit Packages[4] issued by the National Health Authority were really valuable in understanding the domain.

2.4 Existing methods

We researched through the papers where we found that the techniques used in the market are broadly of two types. The third one merges these two to form a hybrid approach.

I) Rule Based methods

We found Panama Papers[20], where CYPHER queries were used to find rules and patterns in the data. We further researched through already available tools that use KG to detect fraud, all of

them included rule-based methods for explainability. We realized that rule-based fraud detection is an indispensable part of fraud detection using knowledge graphs.

II) Model Based methods

We found that many Deep Learning architectures and models have been developed for fraud detection in graph databases. Graph Neural Networks was the most prominent architecture found. Hang Yin et. al., [12] uses unsupervised GNN to classify the embeddings for downstream tasks, however their major contribution is use of behavioral patterns to increase the connections in the sparse graph. Pathan and Shrivastava 2021 [13] have used Graph Convolutional Networks for fraud detection which have outperformed the conventional community detection algorithms in terms of quality and computation requirements.

III) Hybrid method

Haixia Sun et. al. 2020 [5] provides an extensive methodology in the medical domain. They have created a knowledge graph for Chinese medical knowledge and have used it for FWA(Fraud, Waste and Abuse) detection. Though they use Deep Learning models in the process of creating knowledge graphs, for fraud detection rule-based techniques have been used.

2.5 Deep Learning

It was evident that graph-structure specialized deep learning architectures were needed for the project. A github repo by Xiaoxiaoma [8] provides a collection of papers on graph anomaly detection, published algorithms and datasets. *A Comprehensive Survey on Graph Anomaly Detection with Deep Learning* [7] provided a good start by giving a high level overview of techniques used till now. Based upon this the major further learning steps in DL were taken.

Among many tried existing implementations repos in github only few ran successfully. *Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning* [11] is one of them with an AUC score of 0.89. The key ideas have been taken from all of these papers to building models for PM-JAY data.

2.6 Deep Learning Architectures

The major DL architectures required for this project:

GNN

Graph Neural Networks is the deep learning architecture that operates directly upon Graph structure. Typically it takes the node features and edge list or adjacency matrix and processes the output vector for each node and/or edge. It can further take extra parameters as required. By doing so it captures the graph structure which usual deep learning architectures fail to do. Research outputs have shown outstanding performance of GNNs on graph structure compared to

conventional architectures. GCN(Graph Convolutional Network), GAT(Graph Attention Network), etc were reviewed and tested while searching for optimal architecture for PM-JAY data.

Graph Convolutional Networks: Introduction to GNNs [15] gives a good beginner level GNN: GCN, GAT and GraphSAGE tutorial. *A Gentle Introduction to Graph Neural Networks* [14] provides a good implementation understanding of pytorch_geometric.

GANs

Generative Adversarial Networks is an architecture where two models(typically deep learning models) - generator and discriminator compete with each other. The aim of the generator being to generate synthetic data alike to real data to fool the discriminator. The discriminator's task is to classify real and fake data. Thus the better the generator gets, the better the discriminator should be and vice versa. This architecture has given quite impressive results and has been used for generation of many types of data - image, text, videos, audio, etc.

Despite there being other generative techniques why GAN architecture was chosen is described in Chapter Synthetic Data Generation.

In the project GANs is used to generate synthetic fraud data. And also to generate synthetic non-fraud data, when exposing sensitive data publicly was not feasible.

GANs in Action: Deep learning with Generative Adversarial Networks [9] provided a good understanding of GANs use cases and history.

3. Tools and Technology

Knowledge Graphs: We took a knowledge graph as the database for this project because in fraud detection we want as much information related to an entity as we can. RDBMS and such databases do not fulfill this requirement because of the tabular way the data is stored in them. The complete data of an entity is stored across multiple tables fetching all of which is a very expensive task. KG on other hand stores all the properties and related entities just a link away thus getting a complete picture of an entity is as fast as it can be. Further Knowledge graphs are best suited for unstructured data and they can run ad-hoc queries seamlessly. It also has capability to run global algorithms - those that take whole data into account e.g. pagerank, node2vec, etc. It is also highly scalable.The queries are intuitive.

Neo4j: Neo4j is a graph database management system chosen for the project. Despite there being other alternatives neo4j was chosen because:

1. It has a large and active community base.
2. It has a large ecosystem of components and extensions which are available free of cose with limited performance.
3. It provides its own easy to learn and use CYPHER - graph query language.

Neo4j Desktop 1.5.6 on Ubuntu 20 was used throughout the project. The basics for neo4j was learned from the course *Neo4j Fundamentals* provided by Neo4j Graph Academy.

CYPHER: CYPHER is a declarative query language especially developed for neo4j graph databases. It is intuitive and in-depth with added plugins like APOC, GDS, etc. *Cypher Fundamentals* provided by Neo4j Graph Academy gives a beginner course to learn CYPHER.

APOC: Awesome Procedures On Cypher(APOC) is a standard utility library for common procedures and functions of CYPHER. It is overwhelmingly extensive with nearly every possible functionalities imaginable on graph databases.

GDS: Graph Data Science(GDS) is a library that contains graph algorithms for centrality, similarity, community detection, path finding, node embeddings, etc. Each of these sections themselves have many algorithms available. The inbuilt support proves to be very valuable.

Neo4j Python Driver: Neo4j python driver was required because python was the base language platform used for fraud detection for the project. This python driver allows us to execute any possible CYPHER queries on the neo4j database once its session gets connected to it. It gives less code, easy to use interface. It works seamlessly on both python script and jupyter lab.

Neovis: Neovis is a library for graph visualization that can be used in front-end development. It is built upon vis.js and provides intricate control over the display elements, consequently the programmer also need to work more to get some quality work. It works on neo4j data, given neo4j credentials it can take CYPHER queries run on the database and visualize the result. The documentation as of yet is in the worst possible case. Nevertheless the issues raised in the repo are answered. It was used to develop CYPHER Graph User Interface as part of the rule-based system in the project. Officially neo4j gives a list of available options for front-end graph visualization. Graphlytic is one of the applications used for graph analytics and visualization that is supported by neo4j.

3.1 Deep Learning libraries

Pytorch: The model building, training, evaluation everything was done using pytorch. The specific modules were imported from pytorch_geometric.

Pytorch_geometric: Graph Neural Networks are the prevalent deep learning architecture used for fraud detection in graph databases. Pytorch_geometric a library built on top of pytorch, was chosen. This was favored because of our prior acquaintance with the pytorch environment. It is almost similar and goes hand in hand with pytorch, the regular users of pytorch would face almost no difficulty in using it. It provides all required modules for GNN like Graph Convolution Networks(GCN), Graph Attention Networks(GAT), GraphSAGE, etc. *Pytorch Geometric tutorial* [17] on the official site of pytorch geometric gives a good idea of pytorch_geometric for beginners.

Spacy: Spacy is an open-source library that is widely used for advanced Natural Language Processing. It was used for data preprocessing.

3.2 Other Python libraries

Pandas: data processing, manipulation, etc

Icypher: running cypher queries against neo4j database.

Neo4jupyter: Visualization of neo4j database in jupyter lab.

Networkx: Visualization of graph database in jupyter lab.

Py2neo: Running CYPHER queries against neo4j database.

Apache Tika(Python): Text data preprocessing.

3.3 GUI

Back-end:

Flask: Back-end server for GUI.

Front-end:

Bootstrap: Front end visual layout.

Jinja: Dynamic components in front end.

Jquery: Easy manipulation of DOM components.

4. Dataset

The goal of the project is to detect frauds in the real life PM-JAY data. This section describes the source of the data, preprocessing and the final form of data that was used.

4.1 Data Collection

The NHA released official Anti-Fraud guidelines in official NHA and PM-JAY portals formed the basis document data for building ontology, schema and understanding PM-JAY scheme and domain.

The sources of the numerical tabular data is procured from Sri Sathya Sai District NITI Aayog and also PM-JAY data from Maharashtra.

For the full fledged fraud detection many important features are significant in PM-JAY case such as Estimated Amount of procedure, Pre Authorization Approximated Amount, Patient Unique ID(Health Card No in this case), EHCP details, Admit date, discharge date, Fines imposed by NHA, etc. Most of these features were available but many were not. Example: fines imposed by NHA are not available. As per the NHA Anti Fraud Guidelines[3] the fines are imposed when the insurer neither pays the insurance nor updates the required data in the NHA portal within a predefined time interval.

4.2 Data Preprocessing

The data we primarily worked upon was *Dr.YSR AAROGYASRI - AB- PMJAY PATIENTS REGISTRATION AND TREATMENT DONE PATIENTS DATA* from Sri Sathya District, Andhra Pradesh. The data included records from 1st April 2022 to 30th September 2022.

The raw data consisted of 23369 rows of 30 features. After removing duplicate rows we were left with 23312 rows. The features were as follows:
Case Registration Date, Date Of Pre Auth, Case No, Claim No, Patient No, Patient Name, Card No, Age, Gender, Caste, Address, Contact Number, Hamlet, Village, Unnamed: 14, Patient District , Source, Category , Diagnosis, Surgery/Therapy, Provider Name, NH Code, NH Location, Date of Admit, Status, Status Date, Est Amt, Pre Auth Appr Amt, Surgery/Therapy Date, Discharged.

Of these features we took Claim No, Card No, Diagnosis, Surgery/Therapy, Provider Name, Date of Admit, Status, Status Date, Est Amt, Pre Auth Appr Amt, Surgery/Therapy Date, Discharged. They are significant and sufficient features required for fraud detection in this case. If we had more fields , then we could have made proper use of other features also. Example: If we had Fine imposed field then we could have used addresses, Anti Fraud Guidelines[3] gives different time

frames for interstate and intrastate transactions as minimum time for transaction to be carried out.

For these features the lingering spaces were trimmed, date time was converted to correct format, integer and float types were parsed.

Of these features 850 Discharge date fields and 6 contact numbers were Null explicitly, which is reasonable.

The data had 18197 unique card holders. All the patients were registered to be from Anantapur district. The beneficiaries were introduced to PM-JAY via myriad mediums - CMCO, Direct, PHC, Health Camp. PM-JAY is to beneficiaries of all castes which was reflected in the data as it consisted of all 6 caste types(BC, OC, Minorities, ST, SC, Others).

The data consisted of 235 empanelled hospitals whom now onwards we will mention as EHCP(Empanelled Health Care Providers. From our own investigation we found they were a mix of government sector and private sector hospitals and clinics. These data consisted treatment records for 29 medical departments orthopedic surgery and procedures, radiation oncology, medical oncology, ophthalmology surgery, general surgery, general medicine, neurology, nephrology, pulmonology, pediatrics, cardiology, gastroenterology, gynaecology and obstetrics surgery, critical care, cardiac and cardiothoracic surgery, poly trauma, endocrinology, dermatology, genito urinary surgeries, rheumatology, neurosurgery, cochlear implant surgery, surgical oncology, surgical gastro enterology, epidemic disease, ent surgery, plastic surgery, pediatric surgeries, psychiatry. **We took only the Cardiology department,** because further down we found the necessity to match the diagnosis and treatments given in data to the treatments enlisted in the Health Benefit Package document[4] And the overall data consisted of hundreds of such diagnoses. Further the records are using abbreviations thus making the work harder for a person not acquainted with the medical field. Thus we chose just one department Cardiology. Just for cardiology we got 863 records which consisted of 118 diagnoses for 15 unique surgeries and therapy. Thus the data of just this department proved to be sufficient.

There were 25 unique claim statuses in the data. They are as follows(the description of the status has been given after my own research):

1. Claim Paid: The claim has been processed and the payment has been made.
2. Claim sent to CEO - Bill processing: The claim has been sent to the CEO for processing of the bills.
3. CEO Claim Approved: The CEO has approved the claim.
4. Trust Doctor Rejected: The claim has been rejected by the trust doctor.
5. Cancelled By Trust (Surgery Update): The claim has been canceled by the trust due to surgery update.
6. Claim Trust Doctor Recommend Approval to EO: The claim has been recommended for approval to the EO by the trust doctor.

7. Discharge Update: There has been an update on the patient's discharge status.
8. Claim sent to Claim Executive: The claim has been sent to the claim executive for further processing.
9. Cancelled By Trust (Discharge Update): The claim has been canceled by the trust due to an update on the patient's discharge status.
10. JEO Claims Recommend Approval To EO: The Joint Executive Officer recommends approval of the claim to the Executive Officer.
11. CPD Pending - Claim Returned to NWH: The claim is pending as it has been returned to the Network Hospital.
12. Claim Forwarded to CTD: The claim has been forwarded to the Claims TPA Department.
13. EO Rejected: The claim has been rejected by the Executive Officer.
14. Treatment Schedule Update: There has been an update on the patient's treatment schedule.
15. Claim Doctor Recommend Approval to CTD: The doctor recommends approval of the claim to the Claims TPA Department.
16. CTD Pending - Claim Resubmitted by NWH: The claim is pending as it has been resubmitted by the Network Hospital to the Claims TPA Department.
17. CTD Pending Not Updated - NWH Resubmitted: The Claims TPA Department has not updated the pending claim status and it has been resubmitted by the Network Hospital.
18. Claim Trust Doctor Recommend Approval to JEO: The claim has been recommended for approval to the Joint Executive Officer by the trust doctor.
19. Claim Doctor Recommend Rejection to CTD: The doctor recommends rejection of the claim to the Claims TPA Department.
20. CTD Pending - Claim Returned to NWH: The claim is pending as it has been returned to the Network Hospital by the Claims TPA Department.
21. CPD Pending - Claim Resubmitted by NWH: The claim is pending as it has been resubmitted by the Network Hospital to the Central Processing Department.
22. Surgery Update: There has been an update on the patient's surgery status.
23. Preauthorization- Enhancement CH Rejected: The preauthorization request for enhancement of the CH has been rejected.
24. Preauthorization- Enhancement CH Pending: The preauthorization request for enhancement of the CH is pending.
25. Claim sent to Panel Doctor: The claim has been sent to the panel doctor for further evaluation.

Further some rows contained ‘--’ in the Est Amt and Pre Auth Appr Amt fields, however the duplicates of the same row with these fields filled existed. Thus the rows with these invalid values were removed.

4.3 Final Data

Thus finally we selected data pertaining to the Cardiology department with 863 records which consisted of 118 diagnoses for 15 unique surgeries and therapy. The selected features were - Claim No, Card No, Diagnosis, Surgery/Therapy, Provider Name, Date of Admit, Status, Status Date, Est Amt, Pre Auth Appr Amt, Surgery/Therapy Date, Discharged.

Out of 863 rows 25 of them could be probable fraudulent data. For these rows the treatment date of patients precedes their admit date by 1 day. Again we cannot be sure because we lack enough overall data. One probable case is: The patient could have been admitted to another branch of the hospital outside Sathya Sai District where she/he was treated a day before and on that date shifted and admitted to the current branch. Since we don't possess the data from other districts we cannot verify this. Nevertheless we can say that the ratio of non-fraud to probable fraud in our data is 97.1 : 2.9.

This data is transformed into a knowledge graph in neo4j. The same data is given as input to the CTGAN architecture for generation of synthetic data.

5. Methodology

In this section we shall be giving a high overview of the way we carried out our research and experimentation. Many topics could not be included in a section of chapter thus are assigned their own chapters in the subsequent pages.

5.1 Research Design

The objective of this research was to survey the field of fraud detection in the PM-JAY scheme and come up with an approach for fraud detection in the same. We went through the PM-JAY anti-fraud guidelines[3] and other guidelines in the portal and came to know that NHA has established an Anti Fraud Task Force cell in each PM-JAY implemented state along with other IT infrastructure[3]. Despite such efforts there have been many cases of fraudulent activities in this scheme which has led to imposing of considerable fines on empanelled hospitals and even de-empanelment of 100+ hospitals. This indicates that there is still scope for improvement in anti-fraud methods.

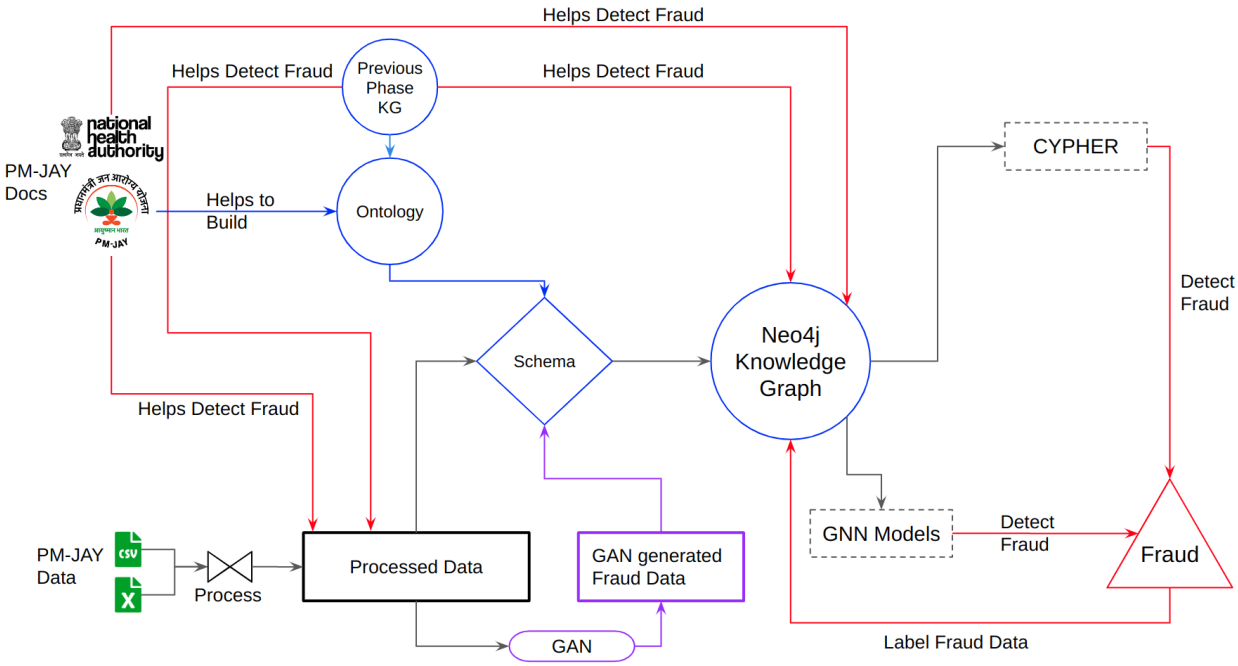

Fig 5.1: Methodology

The above diagram gives the brief overview of methodology used for the project. In the following sections we describe how the guidelines and others guided us to detect fraud in the PM-JAY data, what experimentations we performed to come up with fraud detection procedures, how rule-based and model-based methods were used side by side and how the detected fraud helped to reinforce the detection techniques in cycles.

5.2 Data

The documents and guidelines available in official NHA and PM-JAY websites proved to be valuable data resources for both the understanding of the domain and the formulation of the fraud detection framework.

Many NHA and PM-JAY released guidelines helped us formulate the fraud detection logic as they stated explicitly the process and procedures such as procedure of claim settlement, process of fine imposing when settlement not done within allotted time, the random visits and survey of beneficiaries, etc. Further the guidelines state the entities involved and thus also helped us to form ontology for fraud detection.

The treatment data related to PM-JAY was obtained from Sathya Sai District hospitals the processing of which is thoroughly explained in the Dataset chapter. The raw data were available in many formats which were preprocessed and brought to common csv format using python. We used pandas to process the data extensively to get the collective processed data. Of all the data we selected the “Cardiology” department to proceed for the project which gave us ~900 records with 100+ diagnoses of 15 unique surgeries and therapies.

We used this processed data to generate more data for various reasons, all of which is described in the Synthetic Data Generation chapter. We used CTGAN architecture which is a prominent deep learning architecture for generation of synthetic tabular data. The generated data shows high fidelity.

5.3 Ontology and Schema

The migration of our processed data to knowledge graph form required the schema. And the schema required ontology. We have made use of the output of the previous phase of this 3-part project for ontology and schema. Further we enhanced them manually by our knowledge base formed after we went through anti fraud guidelines.

Below is the schema derived from the ontology and used for knowledge graph:

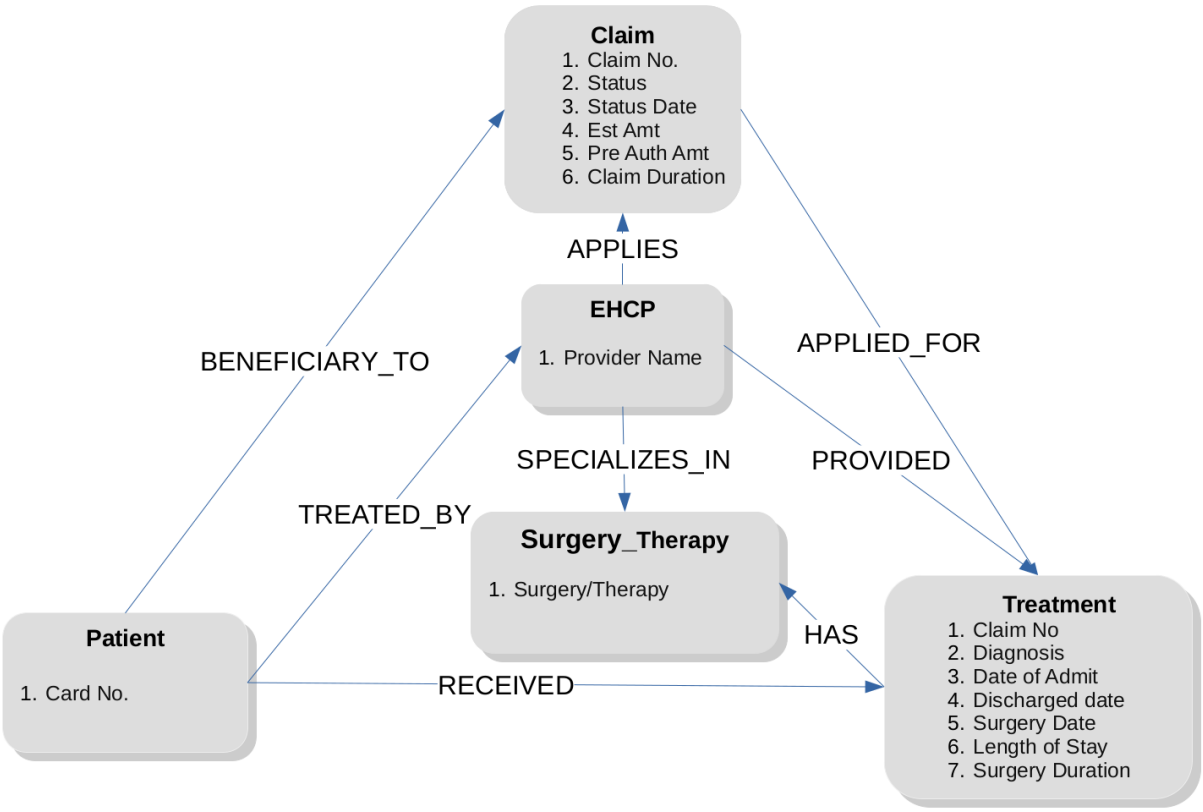

Fig 5.2: Schema

Using the above schema the data was transformed from csv files to neo4j knowledge graph database.

5.4 Fraud Detection

On the created neo4j knowledge graph we ran many algorithms for fraud detection. Graph algorithms such as centrality(Page Rank, Degree Centrality, HITS), community detection(Louvain, Label propagation, weakly connected components, triangle count, modularity optimization), node embeddings(fastRP, GraphSAGE, Node2vec) and others were used. The aim of centrality algorithms was to find the influential nodes and the edges, but in the formed knowledge graph we came to realize that it hardly made worthwhile meaning as any hospital can treat as many beneficiaries and vice versa. Thus we turned to community detection algorithms which gave us many structural communities, but the essence in our case lay in the attributes of the nodes thus the results were not as insightful. Then we turned to node embeddings, we tried other embeddings which used the structural information for node embeddings but the more significant results were yielded by GraphSAGE and similar node embedding algorithms which used the attributes of the nodes to construct node embeddings. During further proceedings we

even used the GNN modules for node embedding. Majorly we used GraphSAGE and torch embedding modules for embedding. The parameters were changed as required by the dataset. Model based methods were found to be useful in detecting frauds for which logic has not been constructed yet and further they are able to learn the pattern of frauds and accordingly adapt to new kinds of frauds by themselves. But in literature review we found that though such systems are extensively used for fraud detection, they cannot be used as a proof nevertheless based upon their findings investigation is carried out.

Further CYPHER queries were also extensively applied for fraud detection. Based upon the anti-fraud guidelines, procedural documents, health benefit packages[4] we deduced logic for possible frauds and accordingly devised CYPHER queries. These queries were run against the neo4j graph database upon which we detected some possible fraud cases in our data. In course of time we found that the CYPHER query syntax had changed a lot over time, thus rendering the tutorials useless after some time. This would mean that every CYPHER query devised would need to be updated every time such changes are made in the CYPHER. For large fraud detection cases this comes at great cost. Thus we came up with an intermediate form which would bridge between the users logic and the CYPHER query, such that users can express their logic in this intermediate form and our developed library translates this intermediate form to CYPHER queries. The advantage is that everytime the CYPHER syntax changes the user need not worry as our updated library would parse the CYPHER query accordingly. To take it one step further, a rudimentary GUI for construction of CYPHER queries is also designed.

Thus we use rule based and model based methods to detect fraud in our data. Further the detected fraud triggered probe into the data which helped us devise more fraud logic and generate more fraud data for model training. Thus the results reinforced the techniques.

5.5 Results

This section summarizes the results of our experimentations.

Firstly in the synthetic data generation where we used CTGAN architecture the generated data showed 100% authenticity with no duplicate rows from the real data. We achieved an overall CTGAN Quality Score of 73.35%, Column Shapes score of 79.2% and Column Pair trend score of 67.49%. The results are discussed in chapter Synthetic Data Generation.

Based upon the logic developed from anti-fraud guidelines we generated and ran Cypher queries. In doing so we found that around 2.90% of records from Cardiology and 1.53% from overall data had treatment date preceding admit date. Now this can be a fraud case or it could be legitimate as we lack many important features in our data. EDA shows that anomalous data are from almost all medical departments and EHCPs.

We also embedded our knowledge graph and build and ran several Deep Learning models on it. The following pic shows our results:

	precision	recall	f1-score	support
0	0.82	0.78	0.80	406
1	0.77	0.81	0.79	366
accuracy			0.80	772
macro avg	0.80	0.80	0.80	772
weighted avg	0.80	0.80	0.80	772

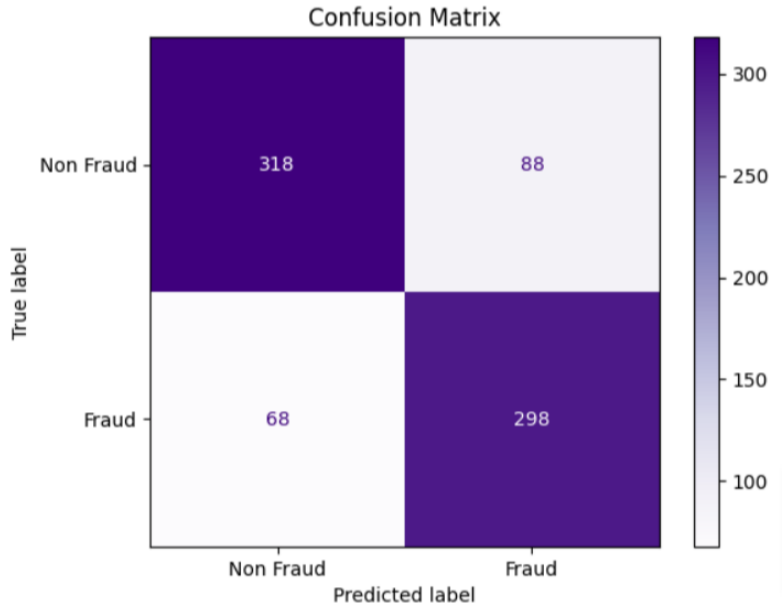

Fig 5.3: GNN result

The GNN model we built was able to achieve a f1 score of 0.80 on our test data. This shows that indeed deep learning models perform good for PM-JAY data too. Note that this model is built for just the specific data we used and thus has almost no use anywhere else. We tried varied models and architectures and at last GNN models gave us the best results. The main aim of this model and its results is that deep learning models do perform good for PM-JAY data.

5.6 Ethical considerations

The PM-JAY made available to us contained many sensitive information such as Card No, Contact No, Medical Conditions, etc of beneficiaries thus utmost caution was taken in handling the data as any minor mistake can lead to violation of data privacy rights of the beneficiaries. We took several measures in order to not expose these sensitive details of the data.

Data privacy was one of top reasons for generation of synthetic data. Using CTGAN we generated the high fidelity synthetic data that was used during presentations and demos to other parties. The features such as Card No, Claim No, etc which were required for the process but couldn't be used as it is from the real data, were generated synthetically such that their format would be same as the real data further they would follow the same distribution as the real data.

Other sensitive features such as Names, Contacts, etc were dropped totally. Finally in the synthetic data, all the rows were authentic, none of them were copies of real data. But we emphasize that even if there were to be copies of real data then it would be just because of the random chances. We noted that when we generated 5000 or more rows the synthetic data would contain about 0.05% rows same as the real data.

Also the visualizations, reports and outputs of publicly published python notebooks, etc are cleared so as to not expose these sensitive data.

5.7 Limitations and Assumptions

The total result of our experimentations and research was an expert-system framework which leverages the use of domain experts, rules and the deep learning models. The need of domain experts to formulate the rules and to oversee the outputs of deep learning models still exists; this requirement of manual intervention by domain experts is the major limitation of this framework. As of yet no framework or technique is able to overcome this limitation and fully automate the fraud detection. The python library PyNeFrauds developed as part of the project does help to automate many tasks but still the user needs to have knowledge of using python libraries.

5.8 Conclusion

We read through the available guidelines and documents at NHA and PM-JAY portals to get acquainted with the domain and develop fraud detection logic. We preprocessed the real PM-JAY data, also generated synthetic counterparts of it and applied various CYPHER queries, graph algorithms on them. We also used several deep learning models majorly Graph Neural Networks on them for fraud detection as classification problems. There are two major results of our work. First is that it shows that CTGAN can be used to generate high quality synthetic data for PM-JAY. Second is a python library PyNeFrauds that would help domain experts in fraud detection. The limitation of our framework and library is the requirement of intervention of domain experts.

6. Synthetic Data Generation

6.1 Need of Data Generation

Some reasons for the need of synthetic data generation are as follows:

1. Lack of Fraud Data: The ratio of fraud to non-fraud data is typically less than 3%. Consequently there is not enough fraud data to train the deep learning models. Using it as it is would lead to serious class imbalance problems. We can synthetically generate fraud data based upon the available fraud data and business logic.

2. Data Privacy: The PM-JAY data contains beneficiaries personal information such as Card Number, Age, Name, Address, etc. Thus using this data directly for demo purposes and reporting would expose these sensitive information violating the data privacy of the beneficiaries. In such cases we can use synthetically generated data.

6.2 Available Data Generation Techniques

After much research the predominant techniques for *tabular* data generation in current times have been briefed below:

1. Algorithm using Business logic: Based upon the business logic we can deduce the rules non-fraud data are supposed to follow. Thus we can also generate data that violate those rules, which consequently can be considered as fraud data. Any programming language can be used and the generation speed also would be high. But the downside is that these generated data may not follow the statistical properties(distribution, variance, etc) that the real data possess. Needless to say the statistical properties are of utmost significance for any data. Thus beyond a certain limit this method may not suffice.

2. VAE(Variational AutoEncoders): This technique embeds the real data into a latent space, afterwards using vectors in the latent space we can generate the synthetic data. The generated synthetic data has high fidelity while retaining the statistical properties of the real data. This technique uses neural-networks consequently it needs large amounts of data to train properly.

3. GAN(Generative Adversarial Networks): GAN is an architecture where two deep learning models - generator and discriminator compete against each other. As the generator gets better in generating fake data, the discriminator gets better in discriminating between fake and real data and vice versa. Thus the generator learns to generate synthetic data of high fidelity. Generator learns the underlying distribution of the real data but we can't access this underlying distribution. The generated data has high fidelity and follows the distribution of real data. Further, unlike VAE, the generator neural

network doesn't need access to the real, thus it can be used in cases where data cannot be exposed to the generator entity.

6.3 GAN

Based upon the pros and cons of available data generation techniques we chose GAN for data generation for the project. Further there has been extensive research in the use of GANs for generation of different types of data - image, text, audio and also tabular. Specifically we chose CTGAN [1] because of its quality performance and existing tools provided for it.

6.4 CTGAN

CTGAN was introduced in paper *Modeling Tabular Data using Conditional GAN* [1] in 2019 and till this data it has remained the top technique in generation of tabular data. This paper introduced some new techniques:

"*Augmenting the training procedure with mode-specific normalization, architectural changes, and addressing data imbalance by employing a conditional generator and training-by-sampling*"[1]

All in all it is able to address the multiple mode problem in continuous variables and imbalance problem in discrete variables effectively. Further SDV(Synthetic Data Vault)[2] provides us with a ready-made python library to construct CTGAN as an end-to-end pipeline, which includes much of data processing too. We have used this library and acquired some considerable quality results.

6.5 Generator Architecture

The basic generator architecture is the same as used by the paper but with the addition of some more layers.

Following is the generator architecture used:

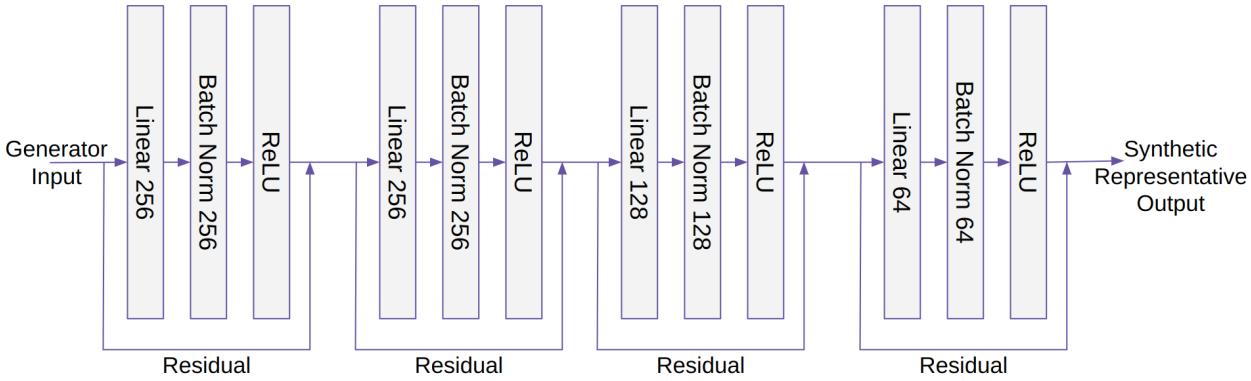

Fig 6.1: CTGAN Generator

Hyperparameters:

Training Parameters

Batch Size: 516
Pac: 12
Epochs: 5000
Embedding Dimension: 50

Adam Optimizer

Learning Rate: 0.002
Decay Rate: 1e-06

WGAN loss

Discriminator Steps: 1
Log frequency: True

6.6 Evaluation

The evaluation of generated data quality is not a trivial task. The data exists in high dimension with each feature with its own distribution which often is multimodal, further we also need to check if the joint distribution of the features are similar for synthetic data and real data. In mathematical concepts as of yet there is no efficient way to calculate or even approximate the similarity of distributions for data in higher dimensions.

6.6.1 CTGAN library score

CTGAN library analyzes the column shapes(individual data distribution) and column pair trends(joint distribution of columns) and gives its custom score to them.

```
Overall Quality Score: 73.35%

Properties:
Column Shapes: 79.2%
Column Pair Trends: 67.49%
```

Fig 6.2: Synthetic Data Score

Above is the evaluation score of our generator by CTGAN library. The score of 73.35% is considered good enough.

6.6.2 Distribution Visualization

The visual comparison of feature distribution for real and synthetic data is the best way to check and evaluate the quality of generated data.

6.6.2.1 Column Pair Trend

We can check the correlation heatmap of both real data and the synthetic data to see if synthetic data captures the feature correlation of real data.

By inspecting the heatmaps(given in next page) of synthetic data and comparing with that of real data we can see that the synthetic data closely follows the pair trend of the real data.

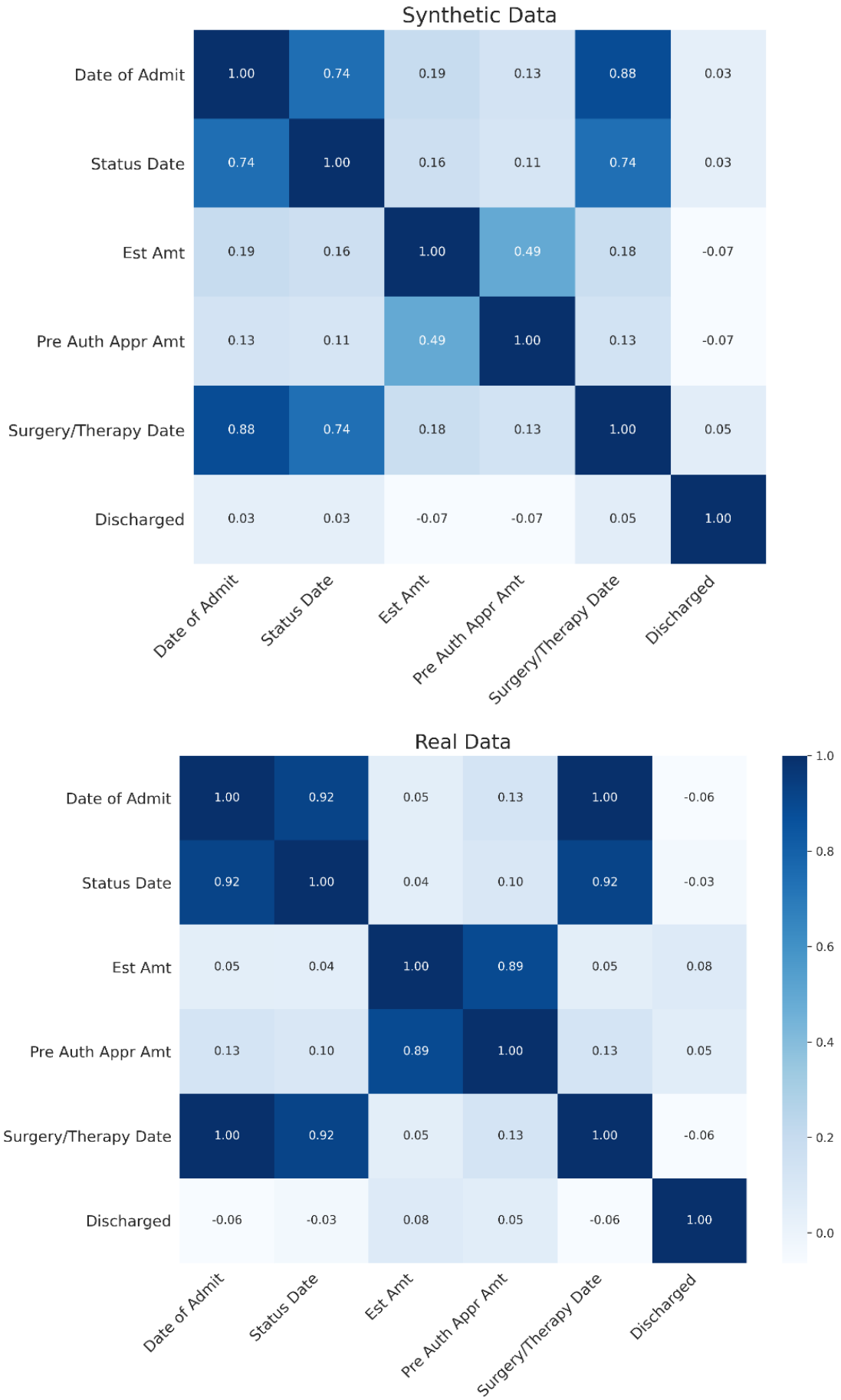

Fig 6.3: Comparison of Real and Synthetic data column pair trend.

6.6.2.2 Column Plots

Column plots help visualize the distribution of individual features. By means of column plots we can compare the feature distribution of real data and synthetic data.

Below are the column plots comparison of real vs synthetic data in our case for the features:

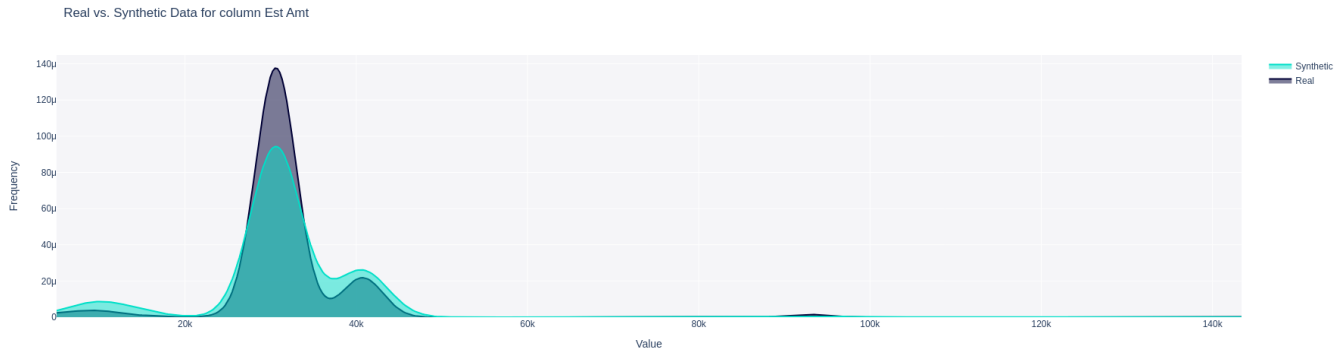

Fig 6.4 Est Amt

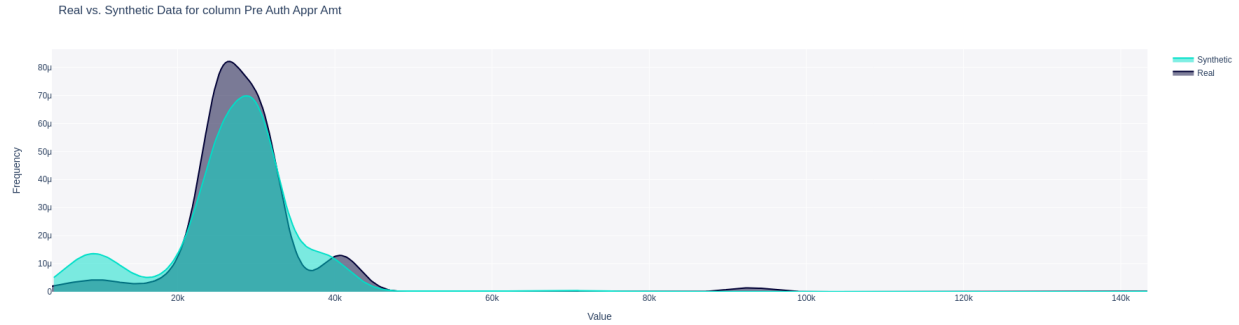

Fig 6.5: Pre Auth Appr Amt

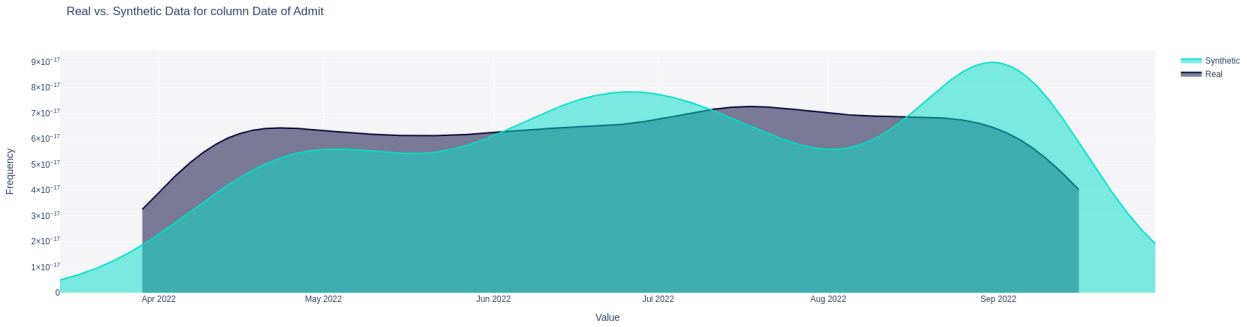

Fig 6.6: Date of Admit

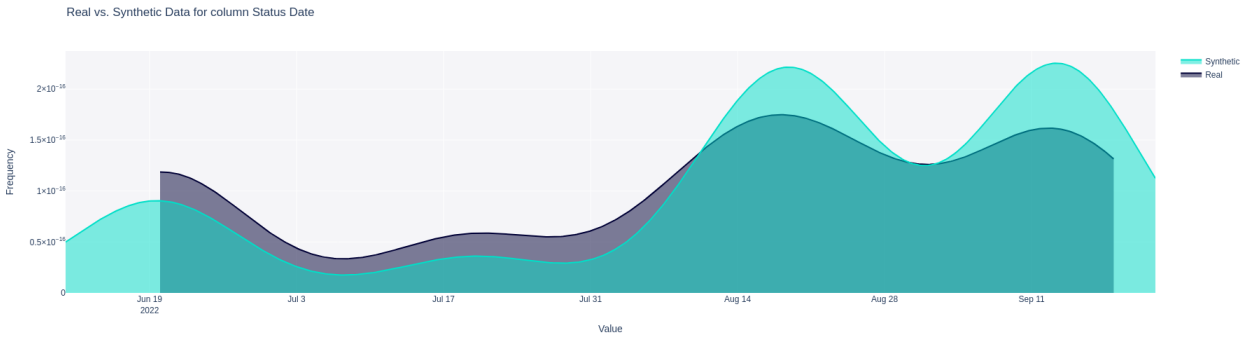

Fig 6.7: Status Set Date

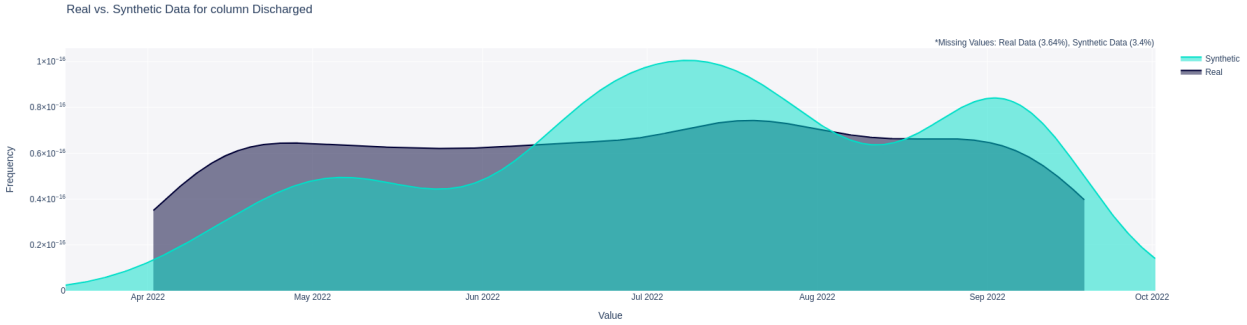

Fig 6.8: Discharge Date

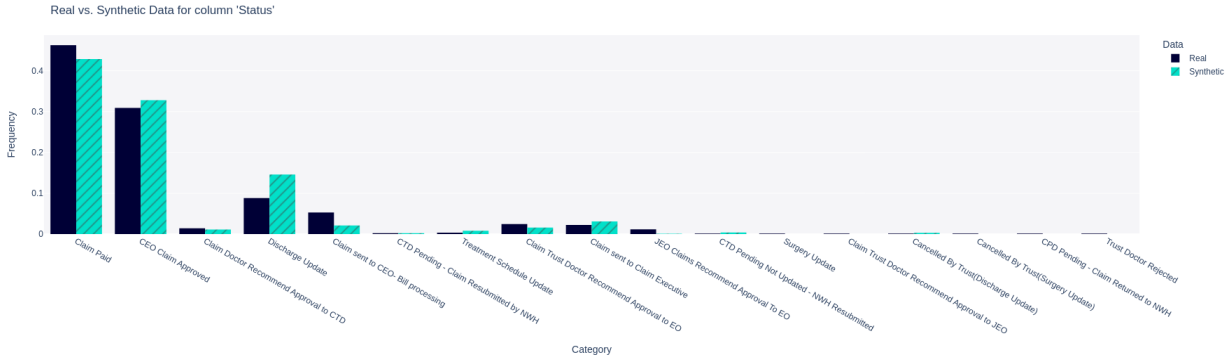

Fig 6.9: Status

Fig 6.10: EHCP

Fig 6.11: Diagnosis

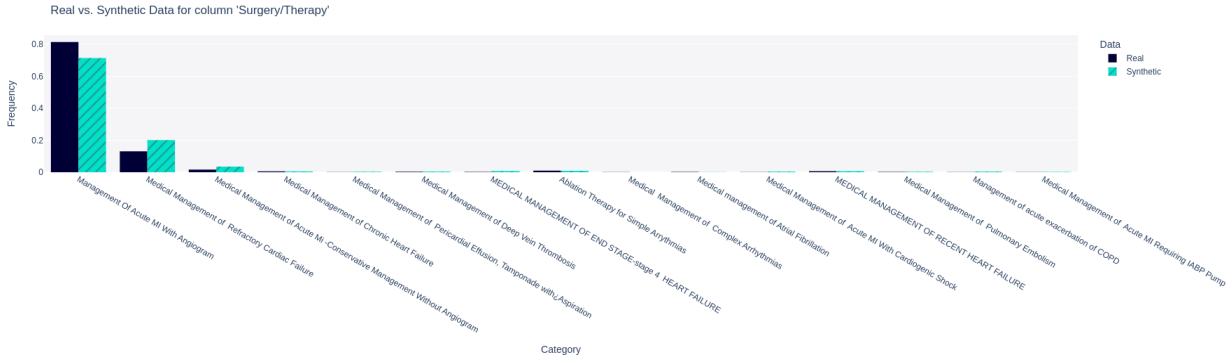

Fig 6.12: Surgery/Therapy

From the data distributions we can clearly see that the feature distribution of the synthetic data closely resembles that of the real data.

The frequencies are given in fraction ranging in [0,1] because the quantity of synthetic data generated usually would be multiple times over real data thus if the actual count be given as frequency the graph would be totally dominated by the synthetic data columns. Instead the proportion of their occurrences are given.

6.6.3 Authenticity

The quantity of synthetic data generated was 5 times more than the quantity of real data. Nevertheless we found that all the generated rows were authentic. The below report visualizes this:

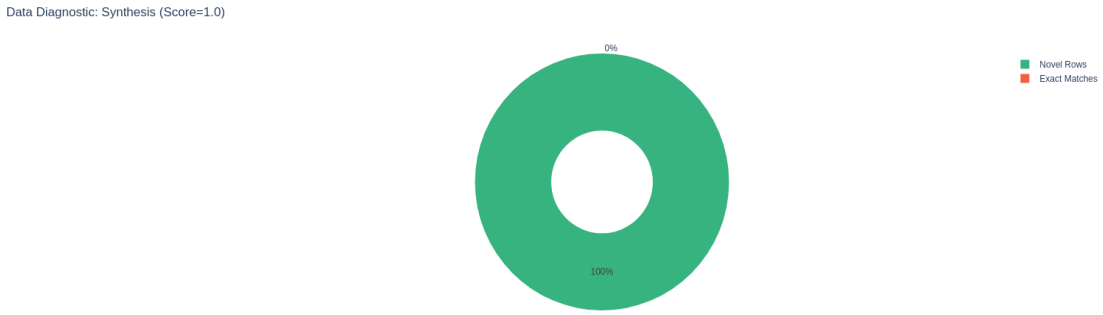

Fig 6.13: Authenticity of Synthetic Data

Even if synthetic data were to contain rows from the real data, by method it would be owing to random chance. We emphasize again that the generator never sees the real data in the entire training process.

6.6.4 Frauds suggested by Guidelines Logic

There were multiple batches of synthetic data generated. On this generated data the business logic was run and the rows that did not confine to the business logic were labeled as fraud. Such as the treatment dates preceding the admit date, discharge date preceding admit date, etc. There can be numerous such business logic. On average we found that in the synthetic data the fraud to non fraud ratio lied around ~40:60. The previous EDA visualizations correspond to 1000 generated data where the ratio of non-fraud to fraud rows was 37:63.

6.7 Data Generation Summary

We used CTGAN to generate data. Following are the inferences of generated data:

1. The generated data has high fidelity.
2. There are no duplicated rows from the real data and even if they were then they would be just because of random chances.
3. The generated data column correlation is almost identical to real data column correlation.
4. The class imbalance in discrete data has been handled by CTGAN gracefully and thus the discrete data distributions in generated data closely resembles that of real data.
5. Distribution of continuous features such as “Est Amt”, “Pre Auth Appr Amt” which are significant in claims has been properly caught by the GANs. For some other continuous model improvements can be done.
6. The fraud:non-fraud data ration in synthetic data, according to the guidelines logic, was approximately 40:60.

Overall this process shows that it is very much possible and practical to generate high fidelity data using GANs. Our trained model can be used only to generate the specific columns upon which it was trained, but for any given new tabular data we can train CTGAN similarly.

7. Expert-System framework

Our PyNeFrauds library helps implement an expert-system framework. This project uses this for the PM-JAY scheme but the library is general and can be used for others too. The framework has two parts: rule-based system and model-based system.

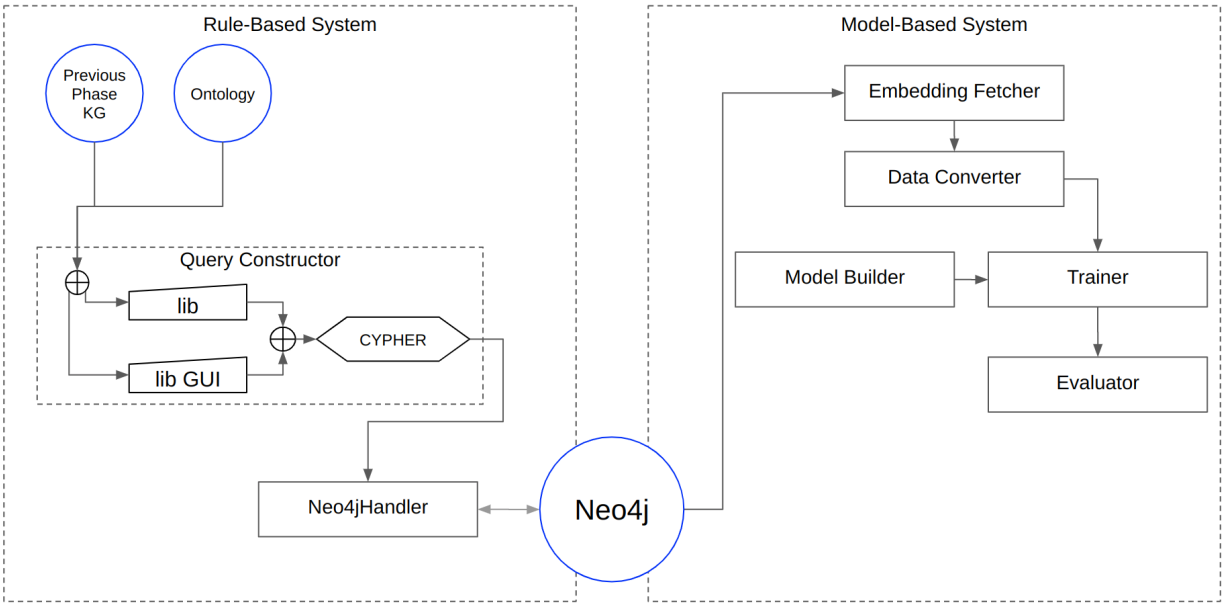

Fig 7.1: Expert-System Framework

The systems are explained as follows:

7.1 Rule Based System

In a rule based system the rules and logic for fraud detection are constructed by domain experts and executed on the database via Query Languages. In our case the domain experts shall refer to NHA issued Anti Fraud Guidelines and construct CYPHER queries that can be run on the neo4j database.

7.1.1 Challenges and Solutions

Domain experts face several challenges in constructing the CYPHER queries. One of the major challenges is the changing syntax of CYPHER. These changes often tend to be major where the frequently used command terms are either substituted or removed totally. Though these changes are made to increase the efficiency of the language it creates major challenges for systems that use those command terms. To address this challenge PyNeFrauds has formed an intermediate syntax such that users can express their logic in this syntax which shall be translated to the CYPHER queries automatically by the library. The advantage is that even if there are major

changes in the CYPHER syntax, users need not worry as the updated library would still parse the logic from the intermediate form to correct CYPHER form. As of yet only basic syntaxes are supported but the covered work gives proof that it can be extended to all other commands of CYPHER.

Another issue that arises is that from a given logic many queries can be formed. Often these queries are deducible, nevertheless users are required to form these queries. To solve this issue the intermediate syntax is formed such that given a logic the library would construct all deducible CYPHER queries. This part still needs much work to be done on library code but basic support is available.

We tried to take one step ahead by setting up basic GUI so that it would be easier for users to construct the queries. Under the hood GUI uses the same intermediate syntax to parse the GUI input to CYPHER queries.

7.2 Model Based System

Model based system refers to the systems that use deep learning models for fraud detection. The anti-fraud guidelines state that Anti Fraud cells in SHA can leverage data analysis, machine learning and deep learning models for fraud detection. But the adoption of model based systems has its own challenges.

7.2.1 Challenges and feasible Solutions

The primary challenge is the explainability of the deep learning models. Even if the model is 100% accurate and classifies a transaction as fraud, it cannot be taken as concrete evidence because deep learning models lack transparency. Being a black box we cannot tell how and why the model is classifying a transaction as fraud. Nevertheless models are extensively used in fraud detection owing to their extraordinary ability to learn new fraud patterns and capture complex fraud logics. The fraud labeling by these models triggers investigations on the transactions. Thus a primary objective of our library PyNeFrauds is to avail deep learning models for fraud detection.

Despite deep learning models being used extensively for fraud detection we came across no open source library that was specifically designed for this. The available ones come in commercial form with large costs. Thus our library PyNeFrauds primarily focuses on integrating Deep Learning techniques with the neo4j database for fraud detection. It is open source with code available in github and is published in Python Package Index too as ‘pyNeFrauds’.

8. PyNeFrauds

This is the open source python library developed that integrates Neo4j with pytorch's torch_geometric for fraud detection. The main aim of this library is to automate as many parts of the fraud detection as possible.

8.1 Availability and Installation

The code for the library is open source and made publicly available in github at https://github.com/Deepam-Rai/PyNeFrauds and is also available in Python Package Index (PyPI) and can be installed as:

pip install pyNeFrauds

And can be imported in the python files thereafter as:

import PyNeFrauds

8.2 Modules

As of now the library supports construction of CYPHER queries, running queries against neo4j database, fetching embeddings from neo4j, easy building of GNN models, training of GNN models and basic evaluation of classification GNN models.

Neo4j handler: We have a dedicated Neo4jHandler object in our library that handles the credentials of neo4j. Once the credentials are set in it, users need not worry as all the further transactions with neo4j will be done using these credentials. Still the library gives the flexibility to use new credentials for almost every executed query.

Construction of CYPHER queries: Using module PyNeFrauds.QueryConstructor we can construct queries automatically. It takes in the intermediate form specific to this library in json format and outputs corresponding CYPHER query. The aim of this module is to shield users from the changing syntax of CYPHER query such that once their logic is expressed in our intermediate form there would be no need to update the queries manually every time syntax changes as this would be taken care of by this module.

Migrating Data from Neo4j to torch_geometric: The PyNeFrauds.nn submodules handles everything related to neural networks. Within it the submodule EmbedFetcher handles fetching of embeddings from neo4j. And there is a custom class PyGDataWrapper submodule that wraps torch_geometric.data.Data in order to make manipulations of torch geometric data easier. Given required details this module automatically creates the torch_geometric.data.Data object within it.

Building Model: Submodule PyNeFrauds.nn.NNModel takes in an ordered dict of modules and builts a neural network from it. It also has its own overridden forward method that handles the different inputs for different modules of torch.nn and torch_geometric.nn.

Training: The submodule function PyNeFrauds.nn.trainer automatically has a prebuilt framework for training models using the given data.

Evaluation: The evaluation is typically case dependent, however often we come across evaluation of classification models using confusion matrix and precision, recall and f1 score. As of yet this evaluation is available in PyNeFrauds.

8.3 Improvement enhancements

The library as of now supports basic layers like GCN, Linear, activation functions, etc from both torch and torch_geometric. More layers can be added. Further the evaluation is available for basic classification models; only this can be extended to models of varying outputs.

9. Conclusion and Future Scope

9.1 Conclusion

We went through NHA's released guidelines on official PM-JAY portals and constructed fraud detection logic for available PM-JAY data of Sathya Sai District. We also constructed ontology for fraud detection in PM-JAY based upon which the knowledge graph schema was constructed. Based upon the logic deduced from Anti-Fraud guidelines we also detected some possible fraud in our data.

Since the given PM-JAY data contained many sensitive beneficiary details we collaged many techniques while handling the data and during demonstration so as to not expose these sensitive data. Synthetic data generation played a major role here.

This project also shows that it is very much practical to generate tabular data for the PM-JAY scheme. We used CTGAN architecture which yielded synthetic data of high fidelity. The generated data was 100% synthetic with none being copied from original data.

From literature review we came to know rule based systems and model based systems were used to detect frauds. However we found no open source library that integrated graph databases, specifically neo4j, with deep learning techniques. To fill this gap we developed the PyNeFrauds library that integrates Python Pytorch's torch_geometric library with Neo4j for fraud detection. It is suitable for an expert-system fraud detection framework.

9.2 Future Scope

This project more or less can be thought of as proof of concept for employment of an expert-system framework for PM-JAY and have developed the basic library for the same. Being in the infant stage there are many scope of improvements:

1. The project can be extended with larger datasets and more features. The feature that we felt significant was a fine imposed by NHA to EHCP and the insurer for failing to cooperate with the guidelines. This would also enhance the data generation part.
2. The major limitation of the project is its requirement of domain expert intervention. As of yet no system has achieved total automation in fraud detection and it's by no means a trivial task in itself. But still we can automate many processes in fraud detection.
3. The GUI of the query constructor can be enhanced significantly. In its state as it is, it just serves as proof of concept for the work.
4. Leveraging the current NLP models we can build a model that translates natural language to our intermediate syntax. We don't want direct parsing to CYPHER because it has constant updates in its syntaxes.
5. The modules of the PyNeFrauds can do with addition of more features.

6. Finding the hierarchical structure of fraud rings by leveraging centrality algorithms in knowledge graph.
7. Detecting various fraud communities by use of community detection algorithms in knowledge graph.
8. Using similarity algorithms to detect duplicate accounts.
9. Automatic feedback update in anti-fraud logic when frauds have been detected and confirmed by the system.

Reference

[1]L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, “Modeling Tabular data using Conditional GAN,” *arXiv.org*, Jul. 01, 2019. https://arxiv.org/abs/1907.00503

[2]sdv-dev, “GitHub - sdv-dev/SDV: Synthetic Data Generation for tabular, relational and time series data.,” *GitHub*. https://github.com/sdv-dev/SDV (accessed Apr. 16, 2023).

[3]National Health Security, “Anti Fraud Guidelines,” *Anti Fraud Guidelines*. https://www.pmjay.gov.in/sites/default/files/2019-04/Anti-fraud-PMJAY-Guidelines_1_2_removed.pdf (accessed Apr. 16, 2023).

[4]National Health Authority, *Health Benefit Packages 2.1*. Accessed: Apr. 17, 2023. [Online]. Available: https://nha.gov.in/img/pmjay-files/HBP-2.1.pdf

[5]H. Sun *et al.*, “Medical Knowledge Graph to Enhance Fraud, Waste, and Abuse Detection on Claim Data: Model Development and Performance Evaluation,” *JMIR Medical Informatics*, vol. 8, no. 7, p. e17653, Jul. 2020, doi: 10.2196/17653.

[6]P. S. Kumar*, D. R. Sarma, and S. S. Mudigonda, “An Analytical Model for Evaluating Social Security Schemes-A Focus on ‘Ayushman Bharat’ Universal Health Scheme in India,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 3, pp. 8929–8936, Sep. 2019, doi: 10.35940/ijrte.c6631.098319.

[7]X. Ma *et al.*, “A Comprehensive Survey on Graph Anomaly Detection with Deep Learning,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021, doi: 10.1109/tkde.2021.3118815.

[8]XiaoxiaoMa-MQ, “GitHub - XiaoxiaoMa-MQ/Awesome-Deep-Graph-Anomaly-Detection” *GitHub*. https://github.com/XiaoxiaoMa-MQ/Awesome-Deep-Graph-Anomaly-Detection (accessed Apr. 17, 2023).

[9]V. Bok and J. Langr, *GANs in Action: Deep learning with Generative Adversarial Networks*. Simon and Schuster, 2019.

[10]S. Martin, B. Szekely, and D. Allemang, *The Rise of the Knowledge Graph*. 2021.

[11]Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2378–2392, Jun. 2022, doi: 10.1109/tnnls.2021.3068344.

[12]H. Yin *et al.*, "Behavioral graph fraud detection in E-commerce," *arXiv.org*, Oct. 13, 2022. https://arxiv.org/abs/2210.06968

[13]S. Pathan and V. Shrivastava, "Identifying Linked Fraudulent Activities Using GraphConvolution Network," *arXiv.org*, Jun. 05, 2021. https://arxiv.org/abs/2106.04513

[14]B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, "A Gentle Introduction to Graph Neural Networks," *Distill*, vol. 6, no. 9, Sep. 2021, doi: 10.23915/distill.00033.

[15]M. Labonne, "Maxime Labonne," *Graph Convolutional Networks: Introduction to GNNs*. https://mlabonne.github.io/blog/posts/2022-02-20-graph_convolution_network.html (accessed Apr. 17, 2023).

[16]"CS224W," *Home*. https://web.stanford.edu/class/cs224w/ (accessed Apr. 17, 2023).

[17]"Pytorch Geometric Tutorial." https://antoniolonga.github.io/Pytorch_geometric_tutorials/ (accessed Apr. 17, 2023).

[18]"Knowledge Graphs," *openHPI*. https://open.hpi.de/courses/knowledgegraphs2020 (accessed Apr. 17, 2023).

[19]"CS 520: Knowledge Graphs." https://web.stanford.edu/~vinayc/kg/2021/index-2021.html (accessed Apr. 17, 2023).

[20]"The Panama Papers: Exposing the Rogue Offshore Finance Industry - ICIJ," *International Consortium of Investigative Journalists*, Apr. 03, 2016. https://www.icij.org/investigations/panama-papers/