Prediction of Drug-Induced Side Effects Through A Graph Comprehensive Approach Using BioSNAP and Mambo

Nikhil Thota[1], Hema Venkata Sai Dumpala[2], Saba Shafiei Tehrani[3], and Somna Satoor[4]
[1]Department of Applied Data Science, San Jose State University, San Jose, California, US

In addressing the complexities of polypharmacy and the associated risks of adverse drug interactions (ADIs), this study leverages the BioSNAP datasets, with a particular focus on the DrugBank dataset. Utilizing Mambo, A real-time data extraction pipeline from DrugBank is re-developed, enabling the cleaning and processing of data for predictive modeling. A key component of this study was the application of a Random Forest algorithm to predict drug half-lives, providing foundational insights into handling and interpreting pharmaceutical data. The research further explores the chse-decagon dataset of BioSNAP, forming bipartite graphs to examine drug-side effect correlations. Due to the complexity of the side-effect graph, the focus shifted to drug nodes. Node2Vec was used for feature extraction, yielding 64-dimensional vectors per node. The processed data were then subjected to stratified splitting, scaling, and SMOTE to address class imbalance. Clustering algorithms, such as K-Nearest Neighbours and HDBScan, were evaluated, with K-means and HDBSCAN ultimately being employed for analyzing drug similarities based on shared side effects. The project culminates in converting the chchse-decagon dataset into a binary classification framework to manage the multifaceted nature of DDI prediction. Utilizing a Graph Neural Network algorithm, the model achieved an 88.9% accuracy in testing. The developed models, encapsulated as .pkl files, are integrated into a Gradio interface, facilitating real-time classification testing and demonstrating potential applications in healthcare informatics.

Index Terms—**Bi-partite graphs, Node2Vec, Graph Data Modeling, K-Nearest Neighbors, HDBSCAN, Graph Neural Networks, Polypharmacy, Drug-Drug Interactions (DDI) , Adverse drug reactions (ADR)**

I. Introduction

THE practice of polypharmacy, involving the concurrent use of multiple medications by a patient, poses significant risks in healthcare, particularly due to adverse drug reactions (ADRs) and drug-drug interactions (DDIs). These risks are quantifiably significant: ADRs account for about 106,000 deaths annually and represent 6.7% of hospital admissions, while serious ADRs number over 2,216,000 cases each year. These statistics underscore the urgent need for more effective tools to predict and manage ADRs and DDIs in polypharmacy. This project will segment the drugs based on the side-effects associated against each drug, by studying the graph structure of the components. The drugs clusters created are used to identify the similar drug families causing side-effects. A classifier is also employed to classify whether that particular drug-drug reaction causes a side-effect or not. For the purpose of this project, real time data pipeline implementation to develop Mambo is also illustrated along with Half-life prediction of the drugs. The Classification models for binary classification are evaluated using performance metrics and classification report, Gradio will be used to check the results of classification of drugs interactions.

A. Problem Statement

Accurate prediction of potential ADRs and DDIs in multi-drug regimens is a complex challenge that directly impacts patient safety. The consequences of these interactions can range from reduced efficacy of treatments to severe, life-threatening side effects. This issue is becoming increasingly critical with an aging global population and a rise in medication usage. The existing solutions for this problem statement is to use Subject matter expertise from various researchers or professionals, this project can be a first step to easily understand the drug-drug interactions and their consequences, and can be easily accessible by common people.

B. Project Background

This paper makes use of DrugBank datasets, and BioSNAP datasets to further understand the drug similarities, and side-effect associations based on their Network structure. This project is a baby step towards understanding the drug drug interactions (DDI) causing side-effects, and the side-effects caused by various drugs. The existing work is pretty complex, and requires a lot of domain knowledge to proceed with the current trends, so this project can be defined as the first step towards 'Understanding the drug-drug interactions to common people'. To meet that goal, a clustering algorithm is deployed to group similar drugs based on the number of side-effects they are associated with, and a classification model is also deployed to understand whether a particular DDI causes a side-effect or not. Inital exploration of Mambo is also done by developing a real time data pipeline with the Drugbank database to understand the Half-life prediction of the Drugs. Overall, the project aims to throw more light and highlight the importance of Adverse Drug Reactions, which are caused by various Drug drug interactions.

II. Related Work

Recent advances in data analytics and machine learning, particularly in the area of graph theory and network analysis such as the study employed by Ke Han [1] briefly explains various methods of extracting features for Drug-drug interactions. Building on these advancements, while employing comprehensive pharmaceutical datasets, such as

BioSNAP, Drugbank, and the exploration of graph-related properties within these datasets. Machine learning models, particularly those leveraging the intricate network of drug interactions and side effects represented in these datasets, have the potential to identify complex drug interactions that are not apparent through conventional pharmacological research methods. Some of these advancements are discussed in the works of Aditya Grover [2], where the authors briefly explain the latest feature extraction techniques that are current industry standards for extracting features for Network datasets. Sunjoo Bang [3] explains how Attention networks can be adapted to Graph use-cases by studying the side-effects prediction with enhanced interpretability based on graph feature attention network. Marinka Zitnik [4], a research team member of the BioSNAP research team from Standford, along with her team published a study on using GNN for side-effect prediction using the BioSNAP datasets. NICHOLAS P. TATONETTI [5] study showcases how comprehensive analysis of observational clinical data, despite its inherent limitations, can significantly enhance the prediction and understanding of adverse drug effects and interactions. This approach marks a substantial advancement in pharmacogenomics, aiming to protect future patients from serious drug-related consequences.

III. Proposed Method

In this study, Mambo is a graph network construction tool for BioMedical data to facilitate the extraction of batch and real-time data from the DrugBank and BioSNAP dataset through various techniques. Here, predicting the half-life values for various drugs. This capability is vital for ensuring that our predictive models are built upon the most current and comprehensive drug information available, which includes data on drug properties, interactions, and reported side effects. The primary focus, however is on analyzing and predicting adverse drug reactions (ADRs) and drug-drug interactions (DDIs) within the realm of polypharmacy, using the BioSNAP datasets. The approach involves categorizing drugs into distinct families or groups based on associated side effects. By understanding these relationships, the study aims to discern patterns that could indicate potential risks in drug combinations, and predict whether the DDI causes an adverse drug reaction or not.

As described in the previous paragraph, these are essentially two different components of this project. So the methodology, and process followed would be different. For Mambo, the focus is on integrating with BioSNAP dataset on batch data and DrugBank Data on real-time basis, apply initial data cleaning, and exploration, feature engineering and then predict the half-life values for various drugs, this architecture is shown in below Fig.1, where the entire architecture is detailed on an overview level.

Basically, In a graph, There are nodes and edges. For example, Node means an entity or a drug or a disease in the Drug Bank dataset. Grouping of similar nodes is called a mode in the same way For Example Disease-Disease, Drug-Drug etc.,Linking two nodes are called edges or link types or relationships For example, Disease-Drugs. Finally, Linking the different modes with a link type is the basis for Multi-Modal representation. This multi modal network graph representation is constructed by the Mambo tool which is built by SNAP group in Stanford. As it is built by the SNAP group which can be imported by the SNAP package library. But, Internally SNAP library works on Python 2.7 and the present version and most of the people use python 3.x version and also the SNAP library initially built on top C++ programming language.

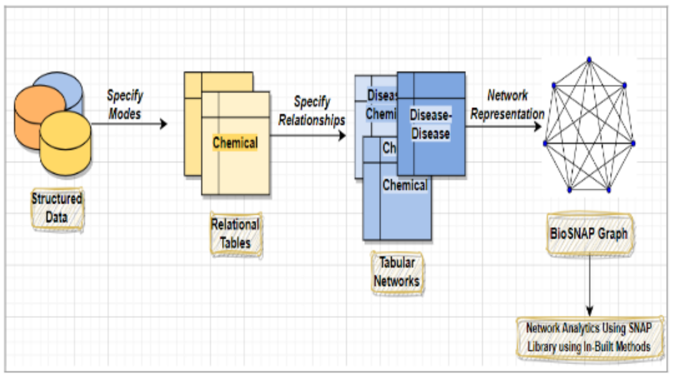


Fig. 1. Architecture for Mambo Data Exploration and Prediction

Once after the modes and relationships are specified. A graph can be represented to show the BioSNAP graph which contains nodes which can be grouped as Modes here Chemical, Disease and then these are linked as relationships or link types or edges to make them as different relationships combinations. The below Fig.2 Shows the same overview of Modes and Link Types.

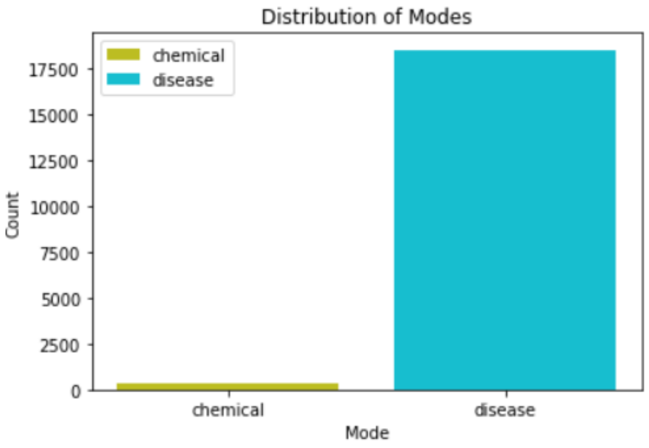
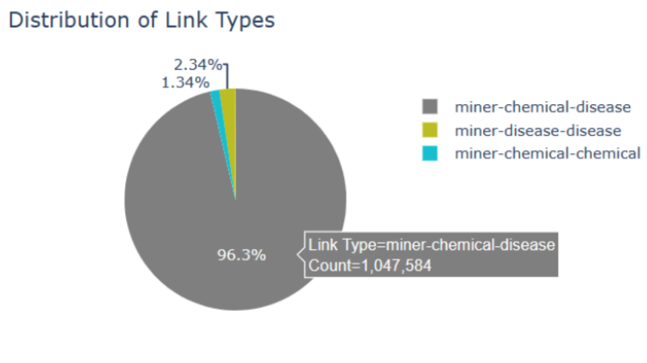


Fig. 2. Mambo Modes Creation Overview

Mambo has constructed Modes for group of chemical which are also known as drugs according to Drug Bank and other sources and the meaning is same. Other than the Half-Life Prediction, the project requires only two types of data which contains chemical, disease and the entities count individually are 359 for Chemcial and 18,554 for Disease. The below Fig.3 shows the Distribution of Link Types.



```
Link types: 3
('miner-disease-disease', 25428)
('miner-chemical-disease', 1047584)
('miner-chemical-chemical', 14592)
```

Fig. 3. Link Types Creation Overview

Once the Modes are created, Mambo focuses on Link Types or Relationships or Edges to specify the relationships between Modes and these relationships are formed using Mapping Tables to make the nodes to group similar modes without any issues caused by different terminologies used by BioMedical data providers.

The primary methodology for this project centers on a graph-based analysis of drug interactions. In this network model, drugs and their potential side effects are represented as interconnected nodes. This graphical representation facilitates an in-depth analysis of the complex relationships between drugs and their side effects. Feature extraction, a key step in the process, leverages graph-related techniques to capture essential properties of drugs and their interactions. These features are crucial for building predictive models based on Graph Machine Learning, that can assess the likelihood of a drug combination resulting in an ADR or DDI. Below Fig.4 briefly explains the architecture the team has employed while working on the ADR prediction problem for various DDIs.

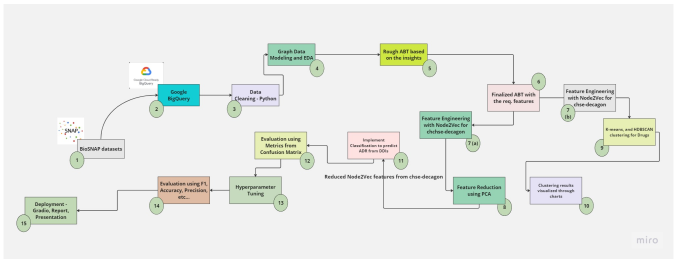

SNAP
BioSNAP datasets
1
Google Cloud Ready BigQuery
Google BigQuery
2
Data Cleaning - Python
3
Graph Data Modeling and EDA
4
Rough ABT based on the insights
5
Finalized ABT with the req. features
6
Feature Engineering with Node2Vec for chse-decagon
7 (b)
Feature Engineering with Node2Vec for chchse-decagon
7 (a)
Feature Reduction using PCA
8
K-means, and HDBSCAN clustering for Drugs
9
Clustering results visualized through charts
10
Implement Classification to predict ADR from DDIs
11
Reduced Node2Vec features from chse-decagon
Evaluation using Metrics from Confusion Matrix
12
Hyperparameter Tuning
13
Evaluation using F1, Accuracy, Precision, etc...
14
Deployment - Gradio, Report, Presentation
15
miro

Fig. 4. Architecture for Graph Data Modeling and Prediction of DDR against various DDI

From the above Fig.4. it can be followed that the study starts with acquiring the BioSNAP datasets from their homepage, and do upload them directly to BigQuery, as BigQuery is a columnar db, getting the data into database was pretty straight forward. Once the data was in the database, the team executed some SQL queries to understand the relationships of the data, as they were graph datasets, which means they are long and thin. The next step was to pull this data into Python using Google BigQuery Python connector, The team applied general data cleaning techniques, and uploaded the cleaned datasets back into BigQuery DB. As these datasets are Graph datasets, meaning there are very few features, like drug_a, drug_b, and side-effects associated with the drug-drug interaction. The next step was to do some EDA specific to the Graph datasets that were being employed. For Feature Extraction, this project banks on the Node2Vec feature extraction technique which is prominent in NLP, but can also be leveraged for Graph datasets. Node2Vec features are first normalized using standardscaler, PCA and T-SVD are experiemented for Dimension reduction techniques, as Node2Vec creates 128 dimensions for each node (Drug), based on the graph structure, generating random walks across the graph to understand the structure, connected edges and various other distance metrics. Post dimension reduction, the reduced Node2Vec feature set for chse-decagon is used for Clustering the similar drugs based on the assoicated side-effects. Using elbow method, an optimal 'k' is employed to divide the drugs into specific cluster. K-means, and HDBSCAN algorithms are experimented with, and the results are shared in further sections. Reduced Node2Vec feature set for chchse-decagon dataset is further scaled, balanced, and then split for test, train and valid datasets, and passed into various classifiers like Random forest, Logistic regression, SGD Classifier, and finally Graph Neural Networks classifer to decide whether a particular DDR results in an ADR or not. Post hyperparameter tuning employing GridsearchCV, Gradio is utilized to deply the classifier models, and show the real time classification results with the experimented models.

Ultimately, this study seeks to provide healthcare professionals with valuable tools for informed decision-making regarding drug prescriptions in polypharmacy scenarios. By accurately predicting potential ADRs and DDIs, these tools aim to significantly reduce the incidence of medication-related complications, thereby improving patient safety. These tools will contribute to the field of pharmacovigilance by enhancing the understanding of drug interactions, thus improving the monitoring, reporting, and management of medication safety. The importance of this study lies in its potential to revolutionize how medications are prescribed in scenarios of polypharmacy. By mitigating the risks associated with ADRs and DDIs, it aims to not only enhance medication safety but also to significantly improve patient care outcomes. The application of these tools in healthcare settings has the potential to transform the management of polypharmacy, making it a safer and more effective practice.

A. Process Model

As this domain was completely new to the team and its a Data Science/Machine Learning Project, CRISP-DM methodology was employed across the project, as it gives the team to be flexible with the problem statements being pursued. Initially the idea was to work around a Multi-class side-effects prediction problem, but due to the computational and technical challenges, the team had to revert back to Business Understanding. This was only possible due to the

CRISP-DM methodology that is being followed sincerely. All the six phases and various work-items of this CRISP-DM methodology are detailed below Fig. 5:

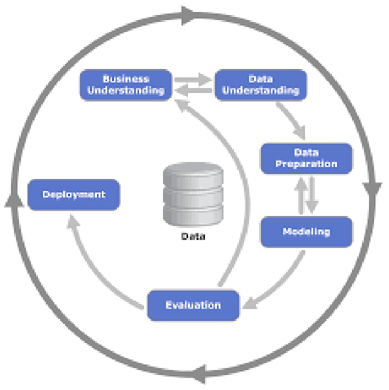


Fig. 5. CRISP-DM Methodology for the Project

1) Business Understanding: During this phase, the team discussed various project ideas of interest, with respect to the given domain, i.e... Network data and as per the suggestions, the team explored BioSNAP and SNAP datasets, and shortlisted top 3 problem statements, out of which this Polypharmacy prediction usecase has been finalized. Post finalizing the project topic, Each individual of the team has made their own literature survey to understand various components of this project like what is the existing research, scope of work, technologies that can be used, How much domain knowledge and Subject Matter Expertise does this project needs, and whether it has all the data that would be required for modeling and other ML usecases, all of these components were evaluated as a team post literature surveys.
2) Data Understanding: During the Data Understanding phase of this project, the team focused on comprehending the Drugbank dataset, accessed via a data pipeline using REST APIs from the DrugBank database. Additional primary datasets, including chchse-decagon and chse-decagon, were sourced from the BioSNAP homepage. This phase also involved comprehensive exploratory data analysis to determine the appropriate graph modeling techniques for these graph datasets. Data cleaning and transformation procedures were established as a result of this phase. A critical aspect was assessing the feasibility of multi-label classification for predicted side-effects. After evaluation, the team concluded that this approach would be computationally intensive and thus, infeasible. Consequently, they revisited their strategy to align with the project's constraints and resources.
3) Data Preparation: During this phase, various preprocessing activities were undertaken, including data cleaning, identifying and addressing outliers, and feature selection for both the sentiment analyzer classifier and the K-Means model. As the primary datasets are graph datasets, it was really important to understand the feature extraction techniques which were compatible with K-means, and Binary classification.Similary, DrugBank dataset for Half-Life Prediction is also undergone the same approach used by BioSNAP dataset Since the datasets are from Pharmacetical Knowledge database and features are extracted by using random forest regression to get the important features and generated ABT table.
4) Modeling: In the modeling phase, Major development was taken care of. Elbow method was employed to get the optimal 'K' and the drug clusters were created. Various Machine learning models were implemented for the Binary classification problem, such as Random Forest, Logistic regression, SGD, and Graph Neural Networks. The team developed models for both the side-effect classifier and the K-Means algorithm. Following the implementation of these models, they conducted a thorough evaluation of the results. To refine and optimize the models, techniques such as cross-validation and hyper-parameter tuning were employed. Specifically, for hyper-parameter tuning, the Random Search CV method was utilized to identify the most optimal parameters for the models, aiming to create an effective recommender system. In Half-Life Prediction modeling, Random forest regression is used to identify the Half-Life of a pill and tuned the hyperparameters.
5) Data Evaluation: In this stage, the team compared the accuracy of each model against the others. These models were executed using a range of performance metrics like confusion matrix, F1-score, etc... The primary objective behind utilizing diverse models was to determine which method yielded the highest accuracy. This process was crucial in enhancing the accuracy of the models, ultimately aiding the team in developing a highly precise proposed system. Out of all the other models, Graph Neural Networks performed very well with an accuracy close to 87% which is quite good for any good solution. Another Model on Half-Life prediction uses MAE,MSE,R2 and Training Time evaluation metrics to evaulate the model and predictions are not accurate as expected though the model is hyperparameterised and Future scope for further enhancement of adopting better models.
6) Deployment: Finally, after the models and results are ready, the front-end was created employing gradio python libraries. This scalable library offers a range of built-in functions to develop an interactive and advanced front-end interface. Following the front-end development, it was seamlessly integrated with the machine learning model to facilitate result prediction and list datasets are used for this project is mentioned below in Table 1.

TABLE I
DATASETS EMPLOYED IN THIS PROJECT

S.No	Dataset Name	Description
1	chchse-decagon	Side effects of DDI
2	chse-decagon	Drug, side-effect network
3	side-effect categories	side-effects classes
4	DrugBank	drugs and half-life combinations

IV. DATA PREPARATION

This phase of CRISP-DM methodology involves conducting Exploratory Data Analysis, Data pre-processing, Data transformation and data preparation for the model. In this project, this was one of the most important phases for this project, as the datasets that were employed for the course of project, and specific usecases needed a lot of pre-processing to change them into formats that were suitable for modeling. The data preparation steps that were involved in this project were Data cleaning, Dropping null columns, imputing null values with suited statistical techniques, merging datasets, filtering of the data, Data transformation to change the schema to suit for the specific modeling technique, converting graph features into numerical vectors, Dimensionality reduction, and splitting the dataset into testing, and training. The unique functionality of this project was, the team had to do the pre-processing with distinct methods for datasets chosen. Each dataset had a specific use-case, so there are significant changes in the way the data is prepared for the modeling phase. Below Fig.6 shows the data flow diagram for the Data preparation flow for the chse-decagon dataset which was extracted from BioSNAP datasets and Fig. 7 shows the different data flow diagram for chchsc-decagon and Fig. 8 Shows the Data Flow diagram for the Drug Bank dataset.

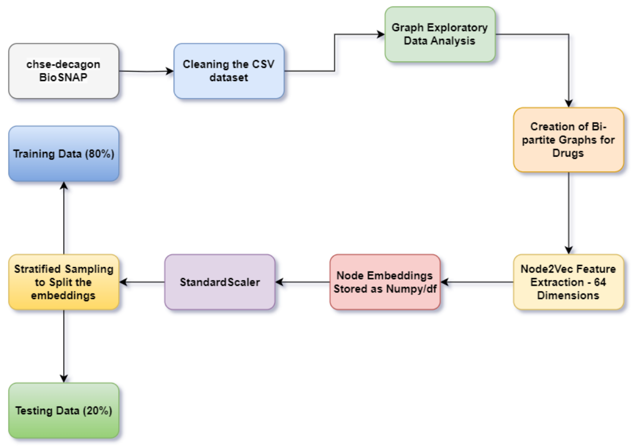

chse-decagon BioSNAP
Cleaning the CSV dataset
Graph Exploratory Data Analysis
Creation of Bi-partite Graphs for Drugs
Node2Vec Feature Extraction - 64 Dimensions
Node Embeddings Stored as Numpy/df
StandardScaler
Stratified Sampling to Split the embeddings
Training Data (80%)
Testing Data (20%)

Fig. 6. Data Flow Diagram for the chse-decagon BioSNAP dataset

The difference between the data preparation steps for chse-decagon, and chchse-decagon dataset is how these datasets are being employed for this research. The former dataset will be utilized to build Bi-partite graphs for both Drugs, and side-effects as separate nodes, and these bi-partite graphs will be used for clustering, where similar drugs are grouped together based on the intensity of the edges, which means two drugs are connected if they share a common side-effect, this computation is not done for side-effects as they are very densely connected, and the number of nodes are quite high, meaning the complexity of creating the graph, and other preparation techniques would be difficult to employ for a side-effect bi-partite graph. The later, chchse-decagon has a different use-case in this research, which is to help understand whether a DDI causes an ADR or not. This classification problem needs a data transformation, to convert the existing chchse-decagon schema to a schema which is suitable for binary classification, this is explained in further pre-processing sections. Figure 5 gives the data flow diagram for chchse-decagon BioSNAP dataset.

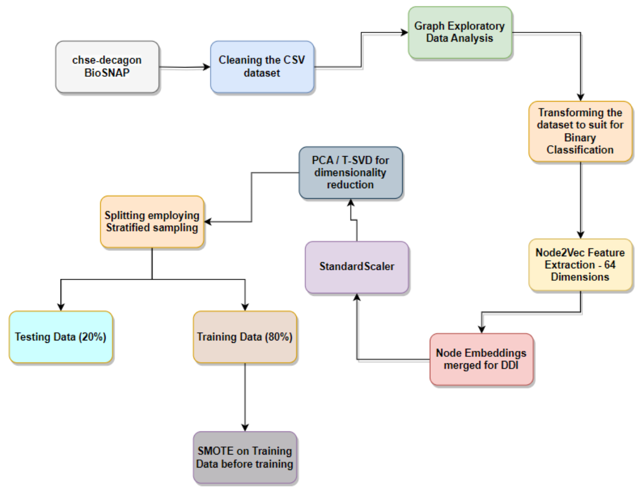

chse-decagon BioSNAP
Cleaning the CSV dataset
Graph Exploratory Data Analysis
Transforming the dataset to suit for Binary Classification
Node2Vec Feature Extraction - 64 Dimensions
Node Embeddings merged for DDI
StandardScaler
PCA / T-SVD for dimensionality reduction
Splitting employing Stratified sampling
Testing Data (20%)
Training Data (80%)
SMOTE on Training Data before training

Fig. 7. Data Flow Diagram for the chchse-decagon BioSNAP dataset

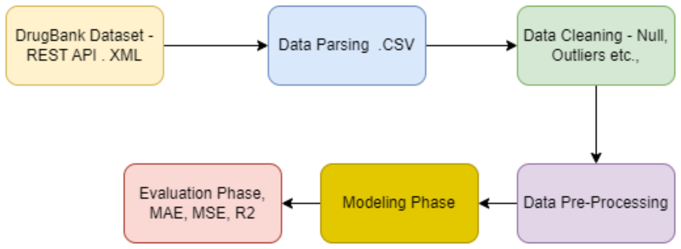

DrugBank Dataset - REST API . XML
Data Parsing .CSV
Data Cleaning - Null, Outliers etc.,
Data Pre-Processing
Modeling Phase
Evaluation Phase, MAE, MSE, R2

Fig. 8. Data Flow Diagram for the DrugBank dataset

Here, The data is collected from DrugBank using REST API by accessing the endpoint and then the data is in .XML format and that needs to be parsed to make the file type as .CSV and then data needs to be cleaned similar to BioSNAP datasets and Data Pre-Processed to make the data compatible with the model without changing the characteristics of the Drug Bank data and then model is evaluated to predict the accuracy.

A. Data Exploration

The first step of Data Exploration was to understand how to model the BioSNAP datasets. This is a very important phase in this project due to the fact that this research is majorly based on the Graph BioSNAP datasets. In Graph Modeling, each drug is depicted as a Node, and side-effects are modeled as associations between these drugs. chse-decagon dataset has a many to many mapping between 'Drug' and 'Side effect', meaning one drug can cause multiple side-effects, and one side-effect can be caused by one or more than one drug. chchse-decagon dataset has a similar representation, but each DDI has numerous side-effects associated in between them, making this a really complex graph to understand with Multi-labels for a particular instance. Below is Fig. 9 where Graph data model for the BioSNAP datasets is illustrated.

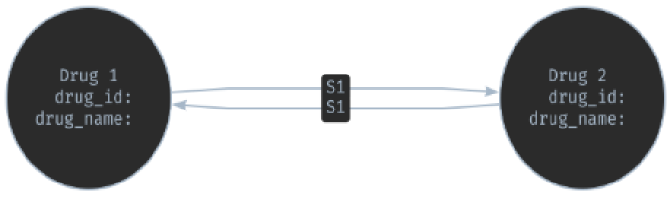


Fig. 9. Graph Data Model for chse-decagon dataset

The above Graph data model, makes sure to connect two drug nodes, if there exists a side-effect association between them, Similar Graph data modeling is also employed with chchse-decagon dataset, which is illustrated below, in Figure 10, this sub-graph is extended towards other nodes, and side-effects are still modeled as associations between the side-effects. Multiple side-effects for single DDI are modeled as properties of the edges in this graph model for chchse-decagon. Fig. 10 shows the Graph Data Model for chchse-decagon dataset.

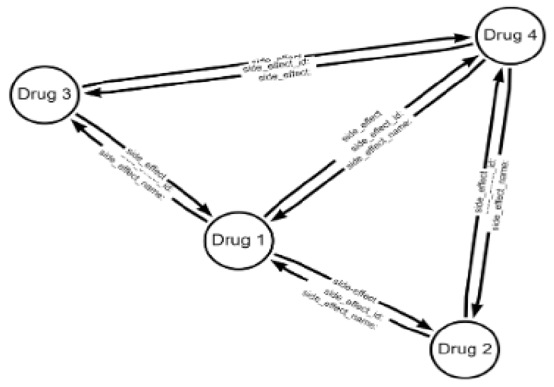
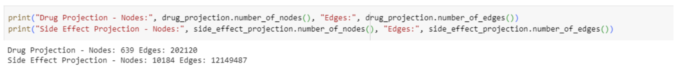


Fig. 10. Graph Data Model for chchse-decagon dataset

The objective behind this Exploratory Data Analysis is to understand the complexities of the Graph Data Models which were created. Respective graphs using Networkx were created for both the BioSNAP datasets. The next step was to compute some graph related metrics to understand the graphs. One such metric which easily helped us to identify the complexity, was to look at the Degree distribution of both the bi-partite projections that were created from chse-decagon dataset. Below is Fig. 11 which gives the number of nodes and edges.



```
print("Drug Projection - Nodes:", drug_projection.number_of_nodes(), "Edges:", drug_projection.number_of_edges())
print("Side Effect Projection - Nodes:", side_effect_projection.number_of_nodes(), "Edges:", side_effect_projection.number_of_edges())

Drug Projection - Nodes: 639 Edges: 202120
Side Effect Projection - Nodes: 10184 Edges: 12149487
```

Fig. 11. Stats from Bi-partite graph projections of Drugs, Side-effects from chse-decagon

The above numbers from Bi-Partite graph suggest that its compute easy to prepare and model the Bi-Partite projection of Drugs.

Below is Fig. 12, which visualizes the degree distribution from the side-effect Projection from chse-decagon, degree distribution plot gives us the frequency against degrees (connectivity) for Side-effect projections.

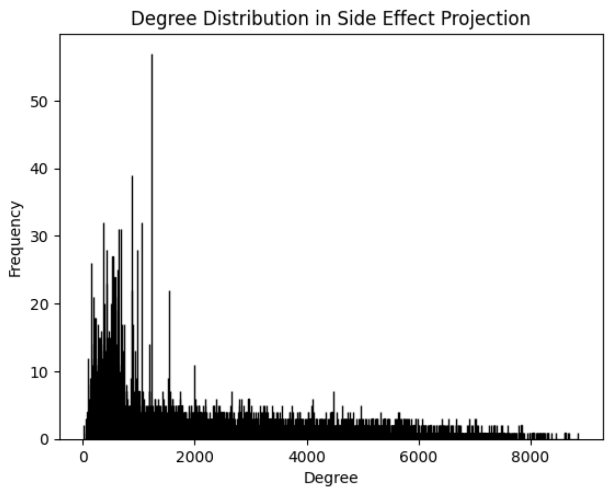
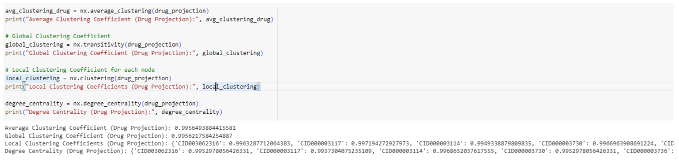


Fig. 12. Degrees of each side-effect (Node) vs Frequency in the side-effect projection of chse-decagon

The above degree distribution plot shows that frequency of the value of degree, in side-effects projection is significantly distributed, resulting in compute heavy, complex, and time taking operations while working with the dataset, making it really hard to prepare the data for Modeling phase. Below Fig. 13 shows the clustering co-efficient and other important metrics of the Drugs projection, to understand whether its worth it to go ahead with the clustering for Drug projection or not.



```
avg_clustering_drug = nx.average_clustering(drug_projection)
print("Average Clustering Coefficient (Drug Projection):", avg_clustering_drug)

# Global Clustering Coefficient
global_clustering = nx.transitivity(drug_projection)
print("Global Clustering Coefficient (Drug Projection):", global_clustering)

# Local Clustering Coefficient for each node
local_clustering = nx.clustering(drug_projection)
print("Local Clustering Coefficients (Drug Projection):", local_clustering)

degree_centrality = nx.degree_centrality(drug_projection)
print("Degree Centrality (Drug Projection):", degree_centrality)

Average Clustering Coefficient (Drug Projection): 0.9956493884415581
Global Clustering Coefficient (Drug Projection): 0.9956217584254887
Local Clustering Coefficients (Drug Projection): {'CID003062316': 0.9963287712064383, 'CID000003117': 0.997194272927973, 'CID000003114': 0.9949338879809835, 'CID000003730': 0.9966963908691224, 'CID
Degree Centrality (Drug Projection): {'CID003062316': 0.9952978056426331, 'CID000003117': 0.9937304075235109, 'CID000003114': 0.9968652037617555, 'CID000003730': 0.9952978056426331, 'CID000003736':
```

Fig. 13. Clustering metrics for Drug projection of chse-decagon dataset

Observations followed from the above clustering numbers - In simpler terms, if a node A is connected to node B and node C, there's a very high probability that nodes B and C are also connected to each other. Such a high clustering coefficient often suggests that the network has a strong tendency to form local clusters or groups. In the context of a drug interaction network, a high clustering coefficient might suggest that drugs tend to have side effects in common with many other drugs they interact with. This could be indicative of a network where many drugs share similar pharmacological properties or side effects, confirming that Drug projection can be used for Clustering.

For chchse-decagon, which comprises roughly 4.2 million rows only with 4 columns, 'drug_a', 'drug_b', 'side_effect_id' and 'side_effect_name'. Below Fig. 14 displays the number of nodes, and edges this Graph dataset is going to create when constructed as a graph. The mapping between DDI and Side-effects is one to many, as a single DDI can cause multiple side-effects and vice-versa. Visualizations from Fig. 14 below, explain how the network structure indicated by these centrality measures points towards a relatively sparse network with a limited number of highly central drugs. This suggests that a binary classification approach might be more feasible and efficient for predicting drug interactions, especially if the dataset is large and the interactions between drugs are not densely interconnected. Binary classification would simplify the model and reduce the computational complexity by focusing on the presence or absence of an interaction rather than predicting multiple potential interactions for each drug pair.

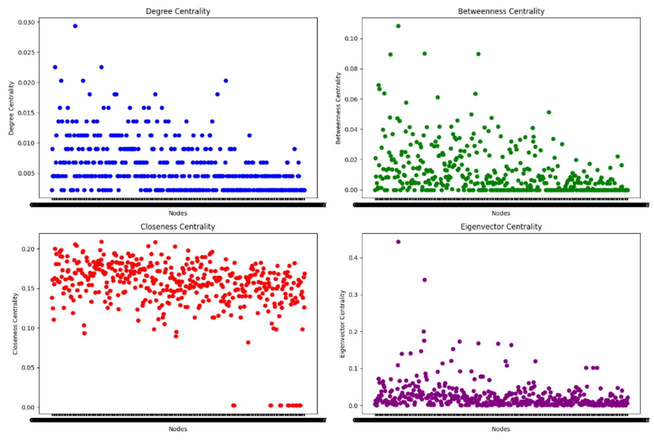


Fig. 14. Graph Metrics to Understand the characteristics of the chchse-decagon dataset

Overall, these metrics suggest a network where drugs are moderately interconnected without a strong dependency on a small set of highly influential drugs. The network exhibits a decent level of integration, with no single or small group of drugs dominating the connectivity or influence within the network, the below average metrics of the chchse-decagon dataset, confirm the theory to navigate towards a Binary classification problem. Average metrics are mentioned in the Fig. 15 below.

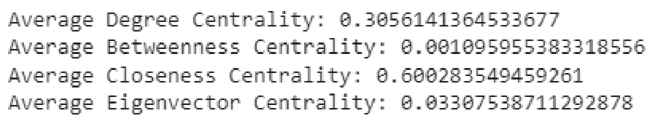

```
Average Degree Centrality: 0.3056141364533677
Average Betweenness Centrality: 0.001095955383318556
Average Closeness Centrality: 0.600283549459261
Average Eigenvector Centrality: 0.03307538711292878
```

Fig. 15. Avg Graph Metrics extracted from the chchse-decagon dataset

The provided metadata pertains to drug compounds, encapsulating various properties either extracted or calculated for each compound. Each field within the metadata serves a distinct purpose, such as the drugbankid, a unique identifier in the DrugBank database, and the name, denoting the nomenclature of the drug compound. Additionally, calculated properties like logP, water solubility, molecular weight, polar surface area, and others provide insights into the drug's lipophilicity, solubility, structure, and pharmacokinetic characteristics.Overview of the DrugBankDataset is shown below in Fig. 16. Notably, parameters like pKa values, hydrogen

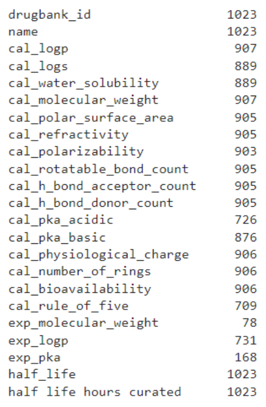

```
drugbank_id                  1023
name                         1023
cal_logp                      907
cal_logs                      889
cal_water_solubility          889
cal_molecular_weight          907
cal_polar_surface_area        905
cal_refractivity              905
cal_polarizability            903
cal_rotatable_bond_count      905
cal_h_bond_acceptor_count     905
cal_h_bond_donor_count        905
cal_pka_acidic                726
cal_pka_basic                 876
cal_physiological_charge      906
cal_number_of_rings           906
cal_bioavailability           906
cal_rule_of_five              709
exp_molecular_weight           78
exp_logp                      731
exp_pka                       168
half_life                    1023
half life hours curated      1023
```

Fig. 16. DrugBankDataset Overview and its column rows count

bond counts, and adherence to the "Rule of Five" contribute to understanding the drug's chemical behavior. Experimental measurements, including molecular weight, logP, and pKa, complement the dataset, along with details on the drug's half-life, curated and validated in hours.

In summary, these metadata fields collectively present a comprehensive overview of the chemical, physical, and pharmacokinetic attributes of the drug compounds, offering valuable insights for research and analysis.

B. Data Pre-processing

The primary objective is to conduct the pre-processing on the BioSNAP datasets, and clean the Drugbank dataset. Cleaning the drugbank dataset will be discussed in the later part of this subsection. For chse-decagon BioSNAP dataset, the pre-processing steps include creating a new Drug Bi-partite graph, which only has Drugs as nodes, and two drugs are only connected if they have common side-effects. The bi-partite graph of Drug Projection is shown in below Fig. 17.

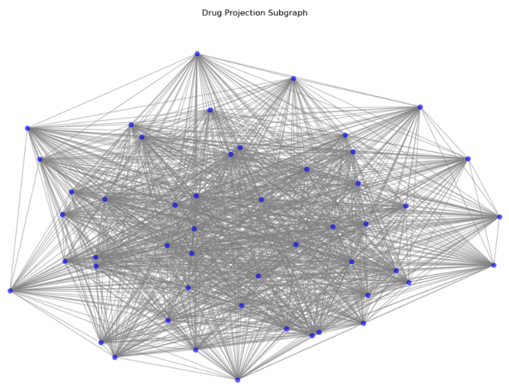
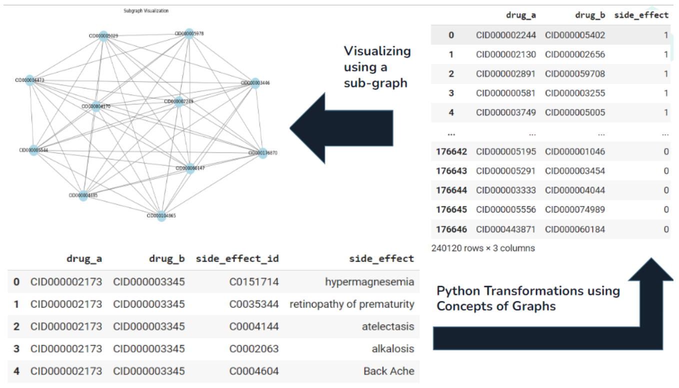


Fig. 17. Drug Bi-partite graph from chse-decagon

The above created drug bi-partite graph is stored in the form of graph object for further feature extraction techniques. It can be observed from the Visualization, that the Drugs (nodes) are tightly knit, making it a solid use-case for clustering with numerous applications. For chchse-decagon dataset, this pre-processing phase is really crucial, as the team has decided to move on with the binary classification, after multiple failed attempts of working on the Multi-class classification problem which was really interesting and very common to this kind of Graph dataset. The pre-processing that's required for chchse-decagon dataset is to convert into a tabular schema which can be used for Binary classification of the predicted sideeffects. This pre-processing is done through Python, below is an example of how the schema changed due to this pre-processing. Fig. 18 illustrates this pre-processing step for chchse-decagon dataset.


Subgraph Visualization
Visualizing using a sub-graph
Python Transformations using Concepts of Graphs

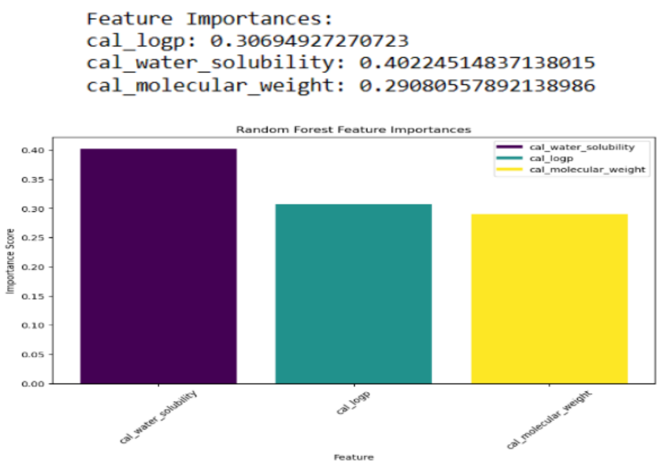
	drug_a	drug_b	side_effect
0	CID000002244	CID000005402	1
1	CID000002130	CID000002656	1
2	CID000002891	CID000059708	1
3	CID000000581	CID000003255	1
4	CID000003749	CID000005005	1
...	...	...	...
176642	CID000005195	CID000001046	0
176643	CID000005291	CID000003454	0
176644	CID000003333	CID000004044	0
176645	CID000005556	CID000074989	0
176646	CID000443871	CID000060184	0

240120 rows × 3 columns

	drug_a	drug_b	side_effect_id	side_effect
0	CID000002173	CID000003345	C0151714	hypermagnesemia
1	CID000002173	CID000003345	C0035344	retinopathy of prematurity
2	CID000002173	CID000003345	C0004144	atelectasis
3	CID000002173	CID000003345	C0002063	alkalosis
4	CID000002173	CID000003345	C0004604	Back Ache

Fig. 18. Preprocessing the chchse-decagon for Binary Classification

The transformation effectively balances the dataset with an equal representation of drug pairs with and without side effects, which is crucial for binary classification tasks to prevent model bias towards the more frequent class. Featuer importance is show in the below Fig. 19.



```
Feature Importances:
cal_logp: 0.30694927270723
cal_water_solubility: 0.40224514837138015
cal_molecular_weight: 0.29080557892138986
```

Fig. 19. Preprocessing the DrugBankDataset to get Feature Importance

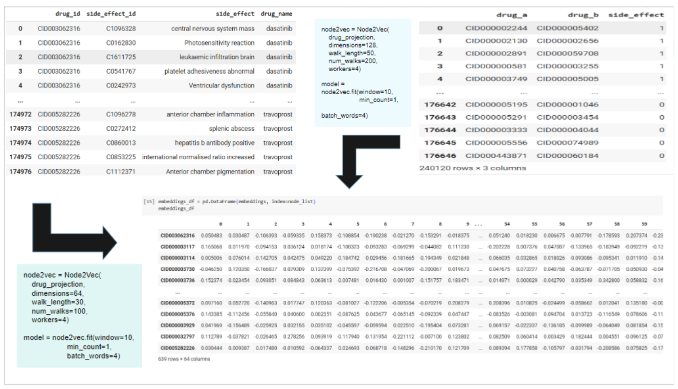
For DrugBank Dataset, Finding the Feature importance using random forest regression to provide the features to the Half-Life Prediction model for better accuracy. Finally, Built the ABT table for Half-Life Prediction.

C. Data Transformation

The most important phase for the entire machine learning adaptation of the defined problem statement is here. To apply machine learning algorithms and models, the first and foremost requirement is data, i.e... features, and ML-required problem statement. This problem statement has a well defined problem statement with lots of real-time applications, what its missing is input features. As the graph datasets are typical examples of Long and Thin datasets, due to their modeling on relationships rather than the individual data points, there are almost no features we can directly use from the datasets, and there isn't much reliable pharmaceutical information to use them as features. The way to get out of this situation is to employ Node2Vec.

Node2Vec is an algorithmic framework for learning continuous feature representations for nodes in networks. It was introduced by Aditya Grover [6] in their paper ”node2vec: Scalable Feature Learning for Networks,” presented at the KDD 2016 conference. The algorithm is designed to capture the diversity of connectivity patterns in a network, enabling a wide range of machine learning applications. Node2Vec operates on the principle of preserving network neighborhoods of nodes. It does so by efficiently exploring diverse neighborhoods by employing a biased random walk procedure, which balances the exploration between breadth-first sampling (BFS) and depth-first sampling (DFS). This approach allows Node2Vec to learn representations that encode both local and global network structures. The random walks are guided by two parameters, p and q, which control the likelihood of immediately revisiting a node and exploring the graph away from the starting node, respectively. This enables the balance between BFS-like and DFS-like neighborhood exploration. The sequences of nodes visited in the random walks are treated as sentences in a language model. The algorithm then uses techniques similar to those used in word2vec (specifically, the Skip-gram model) to learn node representations by maximizing the likelihood of preserving node neighborhoods. For bipartite graphs like drug projection graphs, which consist of two types of entities (e.g., drugs and side effects), Node2Vec can learn to represent the relationships between these entities. This can be particularly useful when trying to project one type of entity onto the other, to understand their interactions at a higher level. In the context of DDIs and ADRs, Node2Vec can help learn representations for drugs that not only reflect their interactions with other drugs but also their potential to cause adverse drug reactions. This could enable more effective prediction of new DDIs and ADRs by learning the complex patterns that lead to such events. Node2Vec's ability to learn rich feature representations makes it particularly useful for tasks such as clustering, classification, and prediction within network datasets. In the field of pharmacology, where understanding the complex relationships between different entities is crucial, Node2Vec's nuanced approach to feature learning can provide valuable insights that might not be captured by simpler graph analysis techniques.

For chse-decagon Drug projection, 64 dimensions, and the other Parameters employed while applying Node2Vec should provide satisfactory results. When it comes to classification problem, the prediction is being done on a DDI rather than a single drug, so both the sets of 64 Dimensions, for each set of nodes (drug_a, drug_b), resulting in a total of 128 columns which is shown the below Fig. 20 and Fig. 21



	drug_id	side_effect_id	side_effect	drug_name
0	CID003062316	C1096328	central nervous system mass	dasatinib
1	CID003062316	C0162830	Photosensitivity reaction	dasatinib
2	CID003062316	C1611725	leukaemic infiltration brain	dasatinib
3	CID003062316	C0541767	platelet adhesiveness abnormal	dasatinib
4	CID003062316	C0242973	Ventricular dysfunction	dasatinib
...	...	...	...	...
174972	CID005282226	C1096278	anterior chamber inflammation	travoprost
174973	CID005282226	C0272412	splenic abscess	travoprost
174974	CID005282226	C0860013	hepatitis b antibody positive	travoprost
174975	CID005282226	C0853225	international normalised ratio increased	travoprost
174976	CID005282226	C1112371	Anterior chamber pigmentation	travoprost

```
node2vec = Node2Vec(
    drug_projection,
    dimensions=128,
    walk_length=50,
    num_walks=200,
    workers=4)

model =
node2vec.fit(window=10,
             min_count=1,

batch_words=4)
```

	drug_a	drug_b	side_effect
0	CID000002244	CID000005402	1
1	CID000002130	CID000002656	1
2	CID000002891	CID000059708	1
3	CID000000581	CID000003255	1
4	CID000003749	CID000005005	1
...	...	...	...
176642	CID000005195	CID000001046	0
176643	CID000005291	CID000003454	0
176644	CID000003333	CID000004044	0
176645	CID000005556	CID000074989	0
176646	CID000443871	CID000060184	0

240120 rows × 3 columns

```
node2vec = Node2Vec(
    drug_projection,
    dimensions=64,
    walk_length=30,
    num_walks=100,
    workers=4)

model = node2vec.fit(window=10,
                     min_count=1,
                     batch_words=4)
```

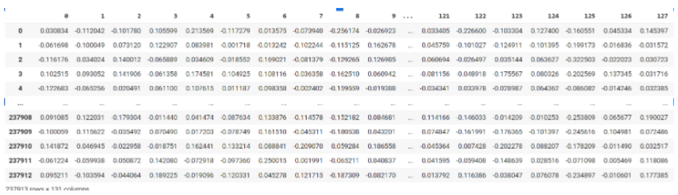
```
[15] embeddings_df = pd.DataFrame(embeddings, index=node_list)
     embeddings_df
```

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	
CID003062316	0.050483	0.030487	-0.106393	-0.059335	0.158373	-0.108854	-0.190238	-0.021270	-0.153291	-0.018375	...	0.051240	0.018230	0.006675	-0.007791	-0.178593	0.207374	-0.23
CID000003117	0.165068	0.011970	-0.094153	0.036124	0.018174	-0.108323	-0.093283	-0.069299	-0.044082	0.111230	...	-0.202228	0.007376	0.047087	-0.133965	-0.183949	-0.092219	-0.13
CID000003114	0.005006	0.076014	-0.142705	0.042475	0.049220	-0.184742	0.029456	-0.181665	-0.194349	0.021848	...	0.066035	-0.032865	0.018026	-0.093086	-0.095341	0.011910	-0.14
CID000003730	-0.046250	0.120350	-0.166037	0.029309	0.132399	-0.075392	-0.216708	-0.047069	-0.200067	0.019673	...	0.047675	0.073227	0.040758	-0.063787	-0.071705	0.050930	-0.04
CID000003736	-0.152374	-0.023454	-0.093051	-0.084843	0.063613	-0.007481	0.016430	0.001007	-0.151757	0.183471	...	0.014971	0.000029	-0.042790	0.035349	-0.342800	0.058832	-0.16
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
CID000005372	0.097160	0.052720	-0.140963	0.017747	0.120363	-0.081027	-0.122206	-0.005354	-0.070219	0.208279	...	0.208396	0.010825	-0.024499	-0.058662	0.012041	0.135180	-0.00
CID000005376	0.143385	-0.112456	-0.055840	0.040600	0.002351	-0.087625	0.043677	-0.065145	-0.092339	0.047447	...	-0.083526	-0.003081	0.094704	0.013723	-0.116549	0.078606	-0.11
CID000003929	0.041969	-0.156489	-0.025925	0.032155	0.035102	-0.045597	-0.059594	0.022510	-0.195404	0.073281	...	0.069157	-0.022337	-0.136185	-0.099989	-0.064049	0.081854	-0.15
CID000032797	0.112789	-0.037821	-0.026465	0.278256	0.093919	-0.117940	-0.131954	-0.221112	-0.007100	0.123802	...	0.082509	0.060414	0.003429	-0.182444	0.004551	-0.096125	-0.07
CID005282226	0.030444	0.009387	0.017480	-0.010592	-0.064337	0.024693	0.068718	-0.148296	-0.210170	0.121709	...	0.089394	0.177858	-0.105797	-0.031794	-0.208586	0.075825	-0.17

639 rows × 64 columns

Fig. 20. In the above transformation step, Node2Vec has been applied to both the Drug projection (chse) and the transformed binary classification (chchse), for the sake of space they are shown to be the same, but these features are extracted separately as the usecases are different

shows transformed dataset for DDI-ADR prediction problem. Post feature extraction from BioSNAP datasets, a round of



	0	1	2	3	4	5	6	7	8	9	...	121	122	123	124	125	126	127
0	0.030834	-0.112042	-0.101780	0.105599	0.213569	-0.117279	0.013575	-0.073940	-0.256174	-0.026923	...	0.033405	-0.226600	-0.103304	0.127400	-0.160551	0.045334	0.145397
1	-0.061698	-0.100049	0.073120	0.122907	0.083981	-0.001718	-0.013242	-0.102244	-0.115125	0.162678	...	0.045759	-0.101027	-0.124911	-0.101395	-0.199173	-0.016836	-0.031572
2	-0.116176	0.034024	0.140012	-0.065889	0.034609	-0.018552	0.169021	-0.081379	-0.129265	0.126985	...	0.060694	-0.026497	0.035144	0.063627	-0.322503	-0.022023	0.030723
3	0.102515	0.093052	0.141906	-0.061358	0.174581	-0.104925	0.108116	-0.036358	-0.162510	0.060942	...	-0.081156	-0.048918	-0.175567	0.080326	-0.202569	0.137345	-0.031716
4	-0.122683	-0.065256	0.020491	0.061100	0.107615	0.011187	0.098358	-0.002402	-0.159559	-0.019388	...	-0.034341	0.033978	-0.028987	0.064362	-0.086082	-0.014746	0.032385
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
237908	0.091085	0.122031	-0.179304	-0.011440	0.041474	-0.087634	0.133876	-0.114578	-0.152182	0.084681	...	0.114166	-0.146033	-0.014209	-0.010253	-0.253809	-0.065677	0.190027
237909	-0.100059	0.115622	-0.035492	0.070490	0.017203	-0.078749	0.161510	-0.045311	-0.180538	0.043201	...	0.074847	-0.161991	-0.176365	-0.101397	-0.245616	0.104981	0.072486
237910	0.141872	0.046945	-0.022958	-0.018751	0.162441	0.133214	0.088841	-0.209070	0.059284	0.186558	...	-0.045364	0.007428	-0.202278	0.088207	-0.178209	-0.011490	0.032517
237911	-0.061224	-0.059938	0.050872	0.142080	-0.072918	-0.097360	0.250015	0.001991	-0.065211	0.040837	...	0.041595	-0.059408	-0.148639	0.028516	-0.071098	0.005469	0.118086
237912	0.095211	-0.103594	-0.044064	0.189225	-0.019096	-0.120331	0.045278	0.121715	-0.187309	-0.082170	...	0.013792	0.116386	-0.038047	0.076078	-0.234897	-0.010601	0.177385

237913 rows × 131 columns

Fig. 21. Above is the transformed dataset for DDI-ADR prediction problem

scaling is usually preferred to make sure the vectors are normalized. As a next step, the team have split the feature sets into 80-20, train and test split, this is done by employing stratified sampling with sklearn, post splitting the data into training, SMOTE is applied to balance the positive classes which were initially much less than the negative (no side-effect). Then, as a next step the team tried to experiment with the dimensionality reduction techniques like PCA and T-SVD. The idea is not to vary the variance by far, so that the Node2Vec extracted feature vectors still capture the essence of the graph dataset they were extracted from. The idea behind employing a dimension reduction technique here is to decrease the training time, while trying to improve the performance of the model, without changing the variance too much. Below Fig. 22 is the variance against number of components plot for Binary classification Node2Vec (128) features. From the above distribution, to capture a variance greater than or equal to 80%, the number of reduced components are 80 and greater. Based on the level of variance that can be retained, the number of components can be chosen, and vice-versa. Its a trade off basically, trading off training time for complete n_components. Below is the Table Fig. 23 to capture the variance and reduced components for TSVD, From the above Table Fig.23, it can be seen that variance is directly proportional to variance, which is directly proportional to the run time. and Fig. 24 Captures the PCA Variance against number of components plot for Binary classification and Fig. 25 Comparing the best run times while employing PCA. From the above distribution, to capture a variance greater than or equal to 80%, the number of reduced

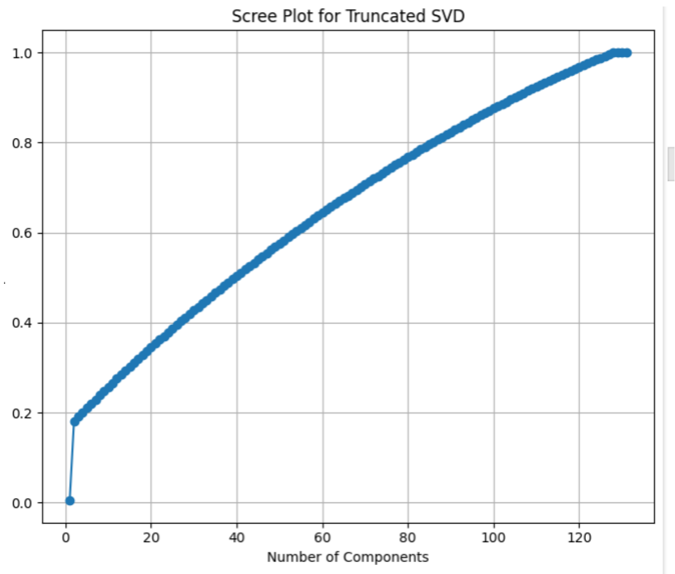
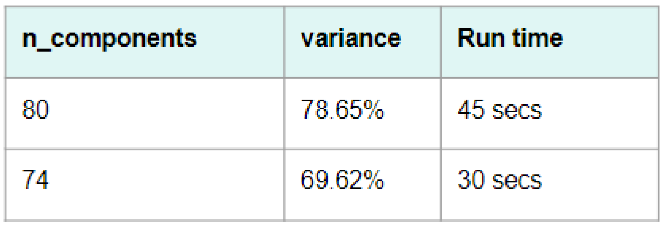


Fig. 22. Variance against number of components plot for Binary classification



n_components	variance	Run time
80	78.65%	45 secs
74	69.62%	30 secs

Fig. 23. Comparing the best training times while employing SVD

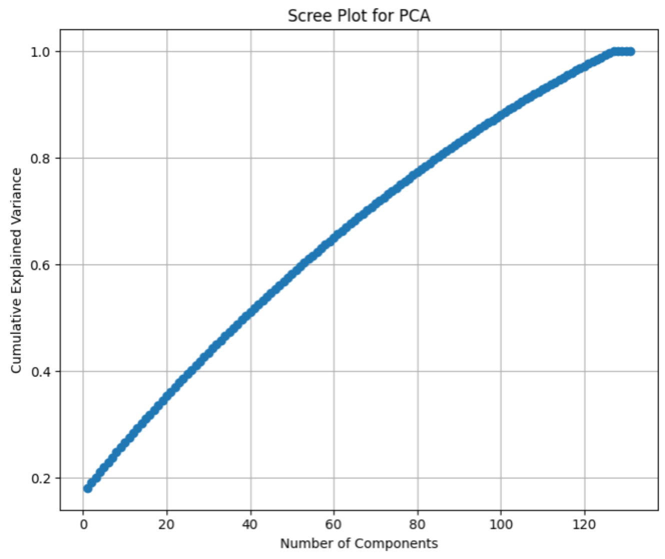
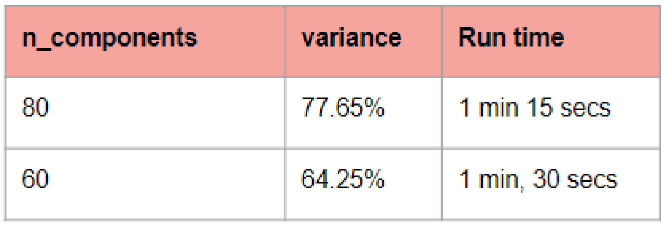


Fig. 24. PCA Variance against number of components plot for Binary classification

components are 75 and greater. The trade-offs are similar when compared to T-SVD. Below is the Table Fig. 25 to capture the variance and reduced components for PCA, From the above



n_components	variance	Run time
80	77.65%	1 min 15 secs
60	64.25%	1 min, 30 secs

Fig. 25. Comparing the best run times while employing PCA

comparison with T-SVD, the run times are better, the time difference may be very minute now, but with huge amount of data its always going to scale out.

V. Modeling

For Modeling, the Node2Vec extracted 64 dimensions feature set is used as input features. K-means is a clustering algorithm that partitions n observations into k clusters in which each observation belongs to the cluster with the nearest mean. This is particularly suitable for the vectors generated by Node2Vec because:

- **Node2Vec** first captures the local and global structures within the graph, reflecting these as feature vectors that encode the similarities and differences between nodes based on the graph's topology (in this case, shared side effects).
- **K-means** then groups these feature vectors into clusters, allowing us to discern distinct groups of drugs that share similar profiles in terms of how they interact with others and the side effects they may cause.

For a k-means clustering algorithm, as the initial value of k can be user-defined, the team started with 5, below Fig. 26 illustrates the set of clusters that k-means could come up with. There are significant number of data points which overlap or wrongly belong to other clusters. To get the optimal values of k for the feature set, the team implemented Elbow-method to understand the optimal 'k'. The Elbow Method is a technique to determine the optimal number of clusters for k-means clustering. It involves running k-means multiple times with different numbers of clusters (k) and plotting the within-cluster sum of squares (WCSS) against k. The point where the WCSS starts to decrease at a slower rate, resembling an "elbow" on the graph, is considered the best value for k. This point represents a balance between having too few and too many clusters, minimizing both the within-cluster variance and the complexity of the model. The Elbow Method is subjective and is often used with other methods to validate the choice of k. Below Fig. 27 shows the elbow is at k=3. Now using optimal 'k' value from the elbow method which is 3, and re-fitting the K-means would give us the clusters, based on the drug connectivity, graph structure captured through Node2Vec features. Below Fig. 28 is the hypertuned K-means clustering, the noise is reduced and the clusters are mostly distinct with the

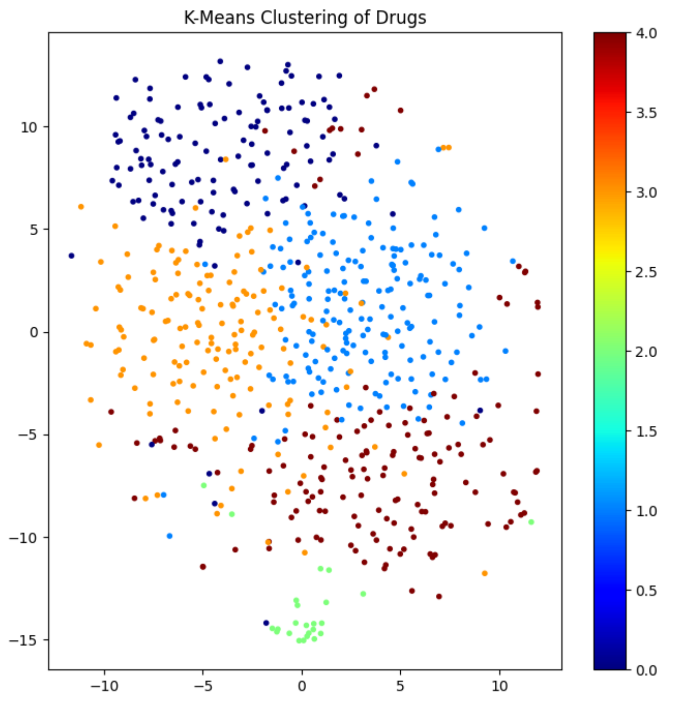


Fig. 26. K-means with experimentation (k=5)

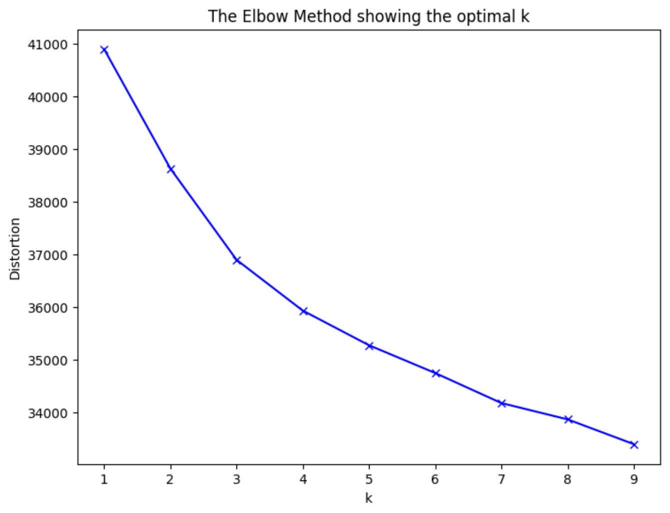


Fig. 27. Elbow method for finding Optimal 'k'

optimal value of K used. As the next step, Clustering was also extended to HDBSCAN. HDBSCAN works on the principle of density-based clustering, which means that clusters are defined as areas of higher density than the remainder of the data set. It converts the space of data points into a hierarchy of clusters and uses a novel technique to extract flat clusters out of the hierarchical tree of clusters. It can find clusters of varying shapes and sizes, unlike k-means which assumes clusters are convex and isotropic. In the context of clustering drugs based on side effects, HDBSCAN's ability to handle different cluster densities and shapes could be particularly advantageous. Drug interaction data is likely to have complex structures that k-means might oversimplify. HDBSCAN can potentially uncover

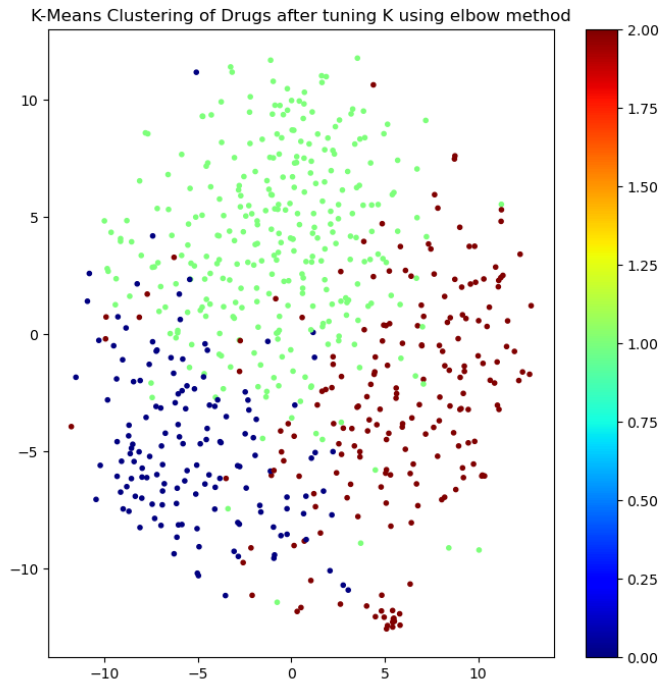


Fig. 28. K-means clustering results after Hypertuning

more nuanced relationships between drugs, such as identifying tightly-knit groups of drugs with common side effects or single drugs that have unique side effect profiles compared to the rest. Below Fig. 29 is the visualization created using tsne to depict the clusters created by HDBSCAN clustering algorithm. 'min_cluster_size' in HDBSCAN is used to define the smallest

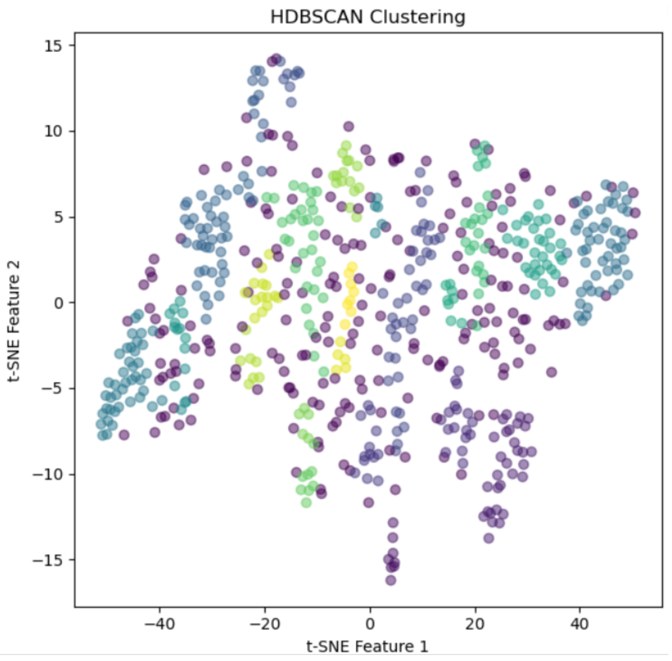


Fig. 29. HDBSCAN Drug Clustering

collection of data points in the dataset that HDBSCAN will consider a cluster. This parameter plays a significant role in the resulting cluster structure and is critical for fine-tuning the clustering outcome to suit specific data characteristics and analytical needs. Below Fig.30 shows the min_cluster_size that we achieved by calculating silhouette score. This value can be experimented with, till the results converge.

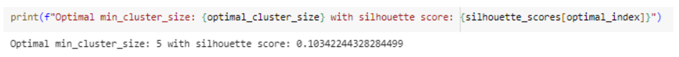

```
print(f"Optimal min_cluster_size: {optimal_cluster_size} with silhouette score: {silhouette_scores[optimal_index]}")

Optimal min_cluster_size: 5 with silhouette score: 0.10342244328284499
```

Fig. 30. HDBSCAN Drug Clustering - Parameter 'min_cluster_size'

Similarly for classifying the DDI as ADR or not, a Node2Vec feature set is created and both the the node feature vectors for both the drugs are appended to create a new feature matrix which corresponds to the entire drug-drug interaction, This dataset, when split into train and test, applying SMOTE and standardscaler, is experimented as input features for various classifiers like Logistic regression, SGD, and GNN architectures.

1) *Logistic Regression*

Logistic Regression is a simple yet effective binary classification algorithm. It is chosen for its compatibility with Node2Vec feature vectors because it provides several advantages:

- Interpretability: Logistic Regression produces interpretable results, making it easier to understand the contributions of individual features (Node2Vec embeddings) to the classification decision. This can be crucial in the medical domain, where interpretability and transparency are often essential.
- Efficiency: Logistic Regression is computationally efficient and can handle large feature spaces. Node2Vec often generates high-dimensional feature vectors, and Logistic Regression can efficiently process such data.
- Efficiency: Logistic Regression is computationally efficient and can handle large feature spaces. Node2Vec often generates high-dimensional feature vectors, and Logistic Regression can efficiently process such data.
- Linearity: While Logistic Regression is a linear classifier, it can still capture complex patterns in data when feature vectors are informative. Node2Vec embeddings are designed to preserve structural information in graphs, allowing Logistic Regression to capture nonlinear relationships when projected into the feature space.

2) *SGD Classifier*

SGD is a popular optimization algorithm used to train various machine learning models, including logistic regression. It is chosen for its compatibility with Node2Vec feature vectors due to the following reasons:

- Scalability: SGD is well-suited for large datasets, and it can handle high-dimensional feature vectors efficiently.
- Node2Vec embeddings often result in high-dimensional feature spaces, and SGD can cope with the computational demands associated with training on such data.
- Flexibility: SGD allows for fine-grained control over the training process. This means you can adjust learning rates, batch sizes, and regularization terms to optimize the model's performance when working with Node2Vec features.

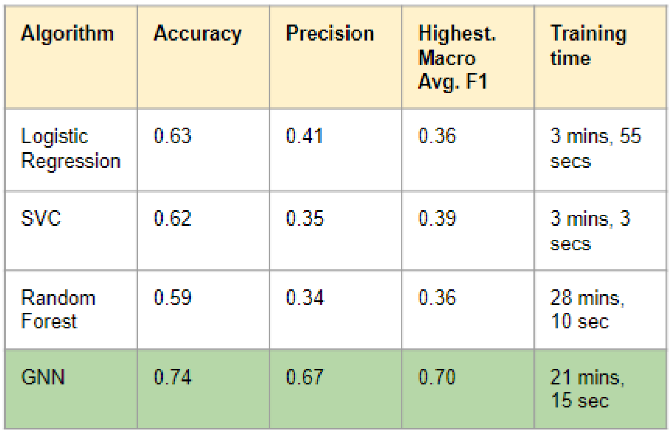
3) *Graph Neural Networks*

Graph Neural Networks are specifically designed for handling graph-structured data, making them a natural choice for compatibility with Node2Vec feature vectors extracted from graph-based datasets like the chchse-decagon dataset. Here are the reasons why GNNs are suitable:

- Graph Representation: GNNs are designed to work directly with graph data, making them well-suited for leveraging Node2Vec embeddings, which are derived from the drug interaction network. GNNs can capture complex relational information present in the graph structure.
- Message Passing: GNNs use a message-passing mechanism to aggregate information from neighboring nodes in the graph. Node2Vec embeddings inherently capture the local and global structural information in the graph, which can be effectively utilized by GNNs during message passing.
- Non-Linearity: GNNs can capture non-linear relationships within the graph data, allowing them to uncover intricate patterns in drug-drug interactions that may not be readily apparent in the raw feature vectors.

Logistic Regression provides simplicity and interpretability, SGD offers scalability and fine-tuning options, while GNNs excel at modeling complex graph structures. The training time, and accuracies for the models are below. Traditional models like Logistic Regression often assume independence between features. In the context of DDIs, this assumption may not hold, as the interactions between drugs can be interconnected. Modern approaches, like Graph Neural Networks, relax this assumption by considering the underlying graph structure.

Below is a short summary Table Fig. 31, which shows the accuracy and performance metrics which are achieved on the test data.

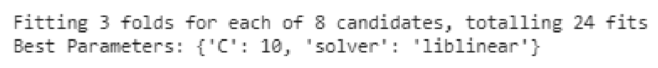


Algorithm	Accuracy	Precision	Highest. Macro Avg. F1	Training time
Logistic Regression	0.63	0.41	0.36	3 mins, 55 secs
SVC	0.62	0.35	0.39	3 mins, 3 secs
Random Forest	0.59	0.34	0.36	28 mins, 10 sec
GNN	0.74	0.67	0.70	21 mins, 15 sec

Fig. 31. Performance of the Models post applying SMOTE to training data

From the above summary, the assumptions about the models have been confirmed, the non-linearity which is compromised in other models, is compensated through GNN, though it takes a longer running time, its a good trade-off where recall, and accuracies are really important.

Hyperparameters are usually found out by employing RandomGridSearch or GridSearchCV methods, For logistic regression below Fig. 32 shows the hyperparmeters for Logistic regression post GridSearchCV method



```
Fitting 3 folds for each of 8 candidates, totalling 24 fits
Best Parameters: {'C': 10, 'solver': 'liblinear'}
```

Fig. 32. Performance of the Models post applying SMOTE to training data

Logistic Regression, a simple yet effective model, offers two primary hyperparameters that significantly influence its performance. The regularization strength parameter (C) plays a crucial role in balancing overfitting and underfitting. Smaller values of C increase regularization, preventing overfitting but potentially leading to underfitting, while larger values decrease regularization, increasing the risk of overfitting. The choice of solver is another hyperparameter that affects optimization. Different solvers have varying convergence speeds and can impact training time. Careful tuning of these hyperparameters is essential to achieve optimal model performance.

Stochastic Gradient Descent (SGD) is a versatile optimization algorithm used for training various machine learning models. Its hyperparameters include the learning rate (alpha) and the batch size. The learning rate determines the step size taken during each optimization iteration. A higher learning rate may lead to faster convergence but risks overshooting the minimum, while a smaller learning rate can result in slower convergence but increased stability. The batch size determines the number of data points processed in each iteration and impacts memory usage and training speed. Tuning these hyperparameters is critical to finding the right trade-off between convergence speed and stability.

Graph Neural Networks (GNNs) are specialized models for graph-based data, such as drug interaction networks. They offer a range of hyperparameters, including the number of layers, hidden units per layer, learning rate, and dropout. The number of layers and hidden units influence the model's capacity and complexity. Increasing these values can capture more intricate patterns but may also lead to overfitting. Learning rate, similar to SGD, affects the optimization process's convergence speed and stability. Dropout is a regularization technique used to mitigate overfitting by randomly dropping a fraction of neurons' outputs during training. Fine-tuning these hyperparameters in GNNs is crucial to strike the right balance between model complexity, generalization, and training stability.
Random Forest Regression is a robust machine learning algorithm belonging to the ensemble learning category. Utilizing an ensemble of decision trees, it improves predictive accuracy by aggregating results from multiple trees. Notably, it introduces randomness in feature selection and data sampling, employing random feature subsets and bootstrap aggregating (bagging) techniques. During prediction, each tree independently contributes an output, and the final prediction often involves averaging these outputs, providing a more accurate and generalized result. One key strength lies in its resistance to overfitting, attributed to the ensemble approach and the diversity introduced in the training process. The versatility of Random Forest extends to various regression tasks, demonstrating effectiveness in capturing complex relationships within

datasets. Furthermore, it offers insights into feature importance, aiding in the identification of significant contributors to predictive performance. Widely applied in finance, healthcare, and environmental science, Random Forest Regression stands out for its ability to handle non-linearity and high-dimensional data.

VI. Evaluation

For K-means and HDBSCAN, below is the drugs distribution across the clusters. For K-means, the number of clusters were 3, and for HDBSCAN its 5, showing their class distribution here below Fig. 33.

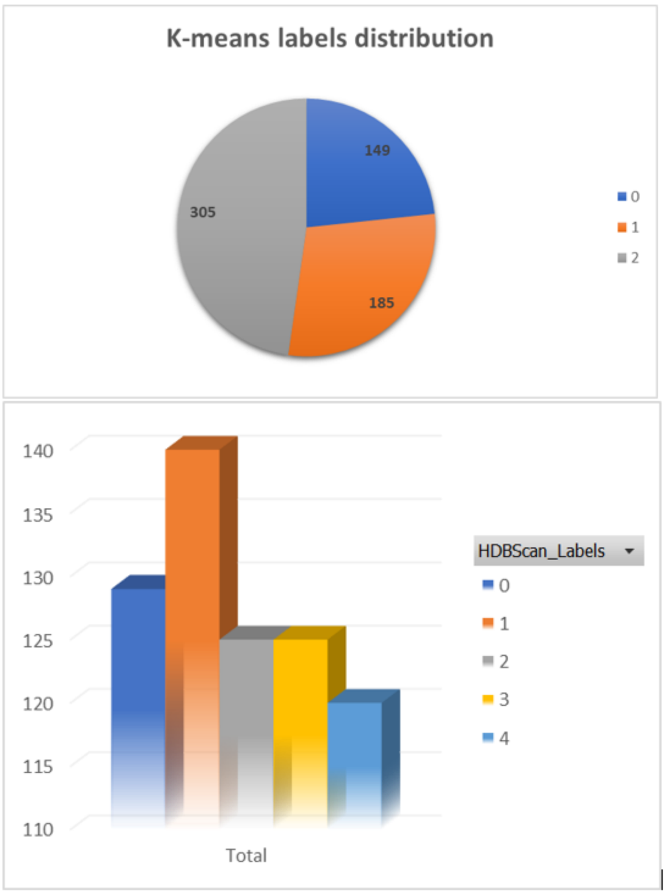
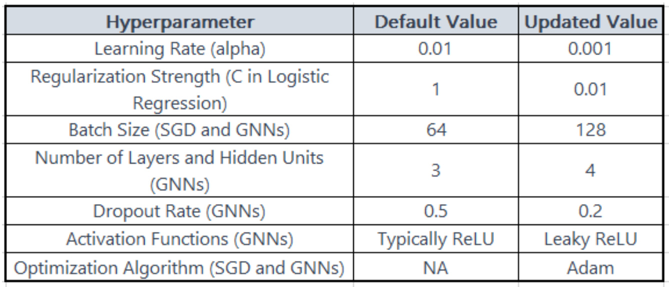


Fig. 33. Distributions of the Drugs across derived cluster labels

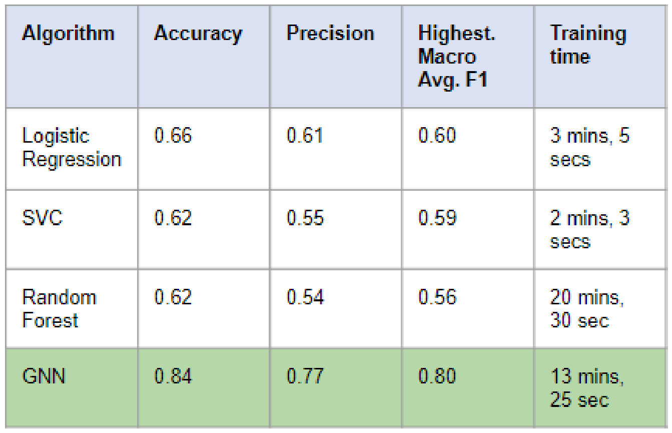
Post Hyperparameter tuning, below are the hyperparameters and their respective values shown in the Fig. 34 and Post Hyperparmeter tuning testing accuracy with split Node2Vec feature set is shown in Fig. 35.

Post Hyperparameter tuning, the accuracies and performance metrics of these models are shown in the below table, The observed improvements in the performance metrics post-hyperparameter tuning indicate a more efficient and effective learning process. For logistic regression, the adjustments in regularization strength and type likely prevented overfitting, resulting in better generalization. For SVC, optimizing the error term penalty and the kernel parameters would have refined the decision boundary to better separate the classes. In the random forest, the increase in the number of estimators



Hyperparameter	Default Value	Updated Value
Learning Rate (alpha)	0.01	0.001
Regularization Strength (C in Logistic Regression)	1	0.01
Batch Size (SGD and GNNs)	64	128
Number of Layers and Hidden Units (GNNs)	3	4
Dropout Rate (GNNs)	0.5	0.2
Activation Functions (GNNs)	Typically ReLU	Leaky ReLU
Optimization Algorithm (SGD and GNNs)	NA	Adam

Fig. 34. Hyperparameters and their change

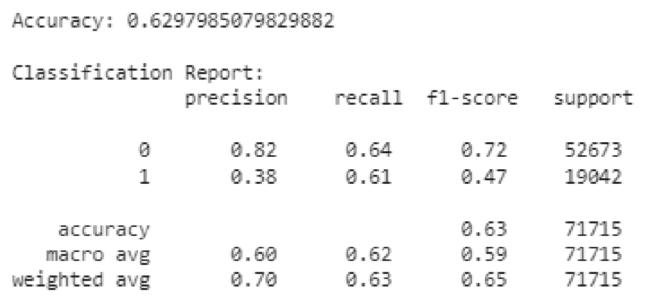


Algorithm	Accuracy	Precision	Highest. Macro Avg. F1	Training time
Logistic Regression	0.66	0.61	0.60	3 mins, 5 secs
SVC	0.62	0.55	0.59	2 mins, 3 secs
Random Forest	0.62	0.54	0.56	20 mins, 30 sec
GNN	0.84	0.77	0.80	13 mins, 25 sec

Fig. 35. Post Hyperparmeter tuning testing accuracy with split Node2Vec feature set

and the tuning of tree-specific parameters likely led to a more robust ensemble. Finally, for GNN, the enhancements are attributable to the tuning of the network architecture and learning process, enabling it to better capture and utilize the complex structural information within the graph data, as facilitated by the Node2Vec algorithm.

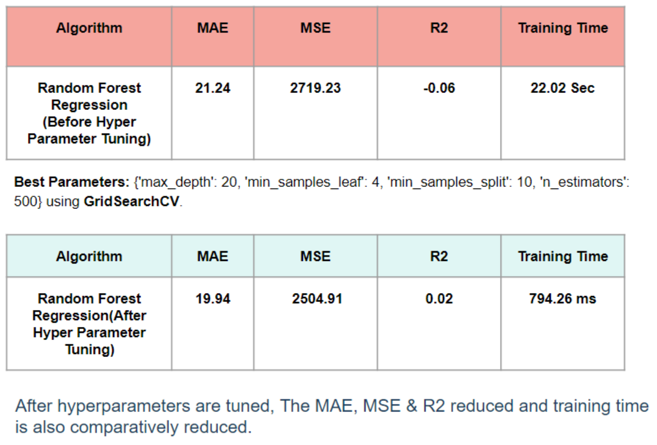
Below are the class level accuracies shown in Fig. 36 for hypertuned, and balanced Logistic Regression, and GNN Models, these metrics on class level signify that the predictions are being accurate for most of the part, Continuous hyperparameter tuning was tried out with Logistic regression, but the model did not converge at all, so the team moved on to GNN.



```
Accuracy: 0.6297985079829882

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.64      0.72     52673
           1       0.38      0.61      0.47     19042

    accuracy                           0.63     71715
   macro avg       0.60      0.62      0.59     71715
weighted avg       0.70      0.63      0.65     71715
```

Fig. 36. Class wise Prediction Scores post hyper-parameter tuning for Logistic Regression

Below Fig.37 shows the prediction accuracy of Random Forest Regression Prediction accuracy on class level and training time is compartively less and accuracy is not as ex-



Algorithm	MAE	MSE	R2	Training Time
Random Forest Regression (Before Hyper Parameter Tuning)	21.24	2719.23	-0.06	22.02 Sec

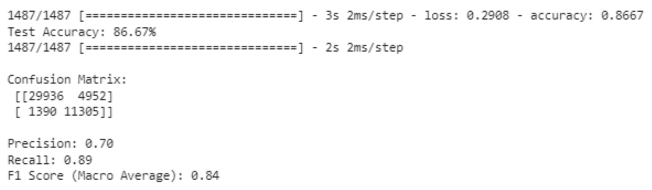
Best Parameters: {'max_depth': 20, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 500} using **GridSearchCV**.

Algorithm	MAE	MSE	R2	Training Time
Random Forest Regression(After Hyper Parameter Tuning)	19.94	2504.91	0.02	794.26 ms

After hyperparameters are tuned, The MAE, MSE & R2 reduced and training time is also comparatively reduced.

Fig. 37. Overall Random Forest Regression Evaluation

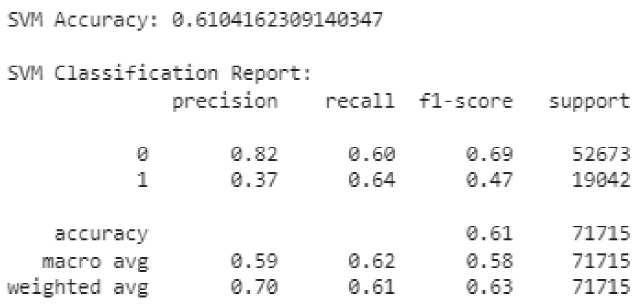
pected and classification accuracy for hypertuned, and SMOTE balanced GNN architecture, Node2Vec hyperparameters were also tuned, resulting the below performance metrics in Fig. 38 and also Class wise Prediction Scores post hyper-parameter tuning for SVM is shown in Fig. 39.



```
1487/1487 [==============================] - 3s 2ms/step - loss: 0.2908 - accuracy: 0.8667
Test Accuracy: 86.67%
1487/1487 [==============================] - 2s 2ms/step

Confusion Matrix:
 [[29936  4952]
 [ 1390 11305]]

Precision: 0.70
Recall: 0.89
F1 Score (Macro Average): 0.84
```

Fig. 38. Class wise Prediction Scores post hyper-parameter tuning for GNN

Clearly, the best performing model is GNN which is explained by its compatibility with the Node2Vec extracted features, and the least performing model is Random forest due to its irrelevance to the proposed problem statement. looks like SVM is still affected by the class imbalance which is evident in the class level performance metrics.

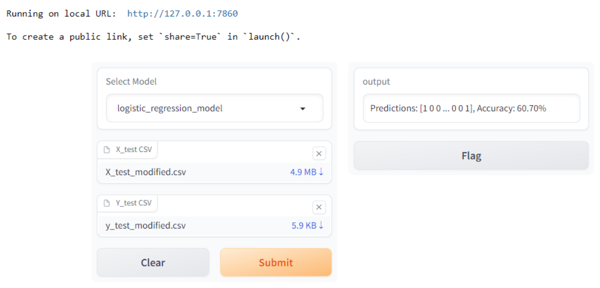


```
SVM Accuracy: 0.6104162309140347

SVM Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.60      0.69     52673
           1       0.37      0.64      0.47     19042

    accuracy                           0.61     71715
   macro avg       0.59      0.62      0.58     71715
weighted avg       0.70      0.61      0.63     71715
```

Fig. 39. Class wise Prediction Scores post hyper-parameter tuning for SVM

VII. Deployment

The deployment of the project system is designed to handle user scalability and adapt to dynamic environments effectively. The system utilizes Gradio, an open-source Python framework, to deploy machine learning models. Gradio allows for the rapid development of web interfaces that users can interact with, making it suitable for models that require instant feedback and interaction. Gradio's interfaces are hosted on scalable cloud servers that can adjust resources based on user demand. This ensures that the application remains responsive and available, even as the number of users increases. The system is capable of load balancing to distribute user requests efficiently across available resources to maintain performance. Considering the dynamic nature of user environments, Gradio interfaces are designed to be cross-platform compatible, ensuring accessibility across various devices and operating systems. The deployment architecture incorporates containerization, which encapsulates the model and its dependencies in a container. This containerization allows for quick deployment and seamless updates to the system without disruption to the user experience. Gradio facilitates the creation of custom interfaces with input fields tailored to the model's requirements. Users can input data through these fields and receive model outputs in real-time. Interfaces can be embedded into existing websites or accessed via shareable URLs, providing flexibility in how users interact with the models. The system deployment includes security protocols to protect sensitive data and user privacy. Data encryption, secure API endpoints, and user authentication are part of the deployment to ensure safe and secure access. Post-deployment, the system is monitored continuously for uptime, latency, and overall health. Automated alerts are in place for any system anomalies, and a streamlined process for updates and maintenance minimizes system downtime. The below Fig. 40 shows the deployment of developed models on gradio to test the accuracy directly with the test dataset.



```
Running on local URL:  http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

Select Model: logistic_regression_model

X_test CSV: X_test_modified.csv 4.9 MB

Y_test CSV: y_test_modified.csv 5.9 KB

output: Predictions: [1 0 0 ... 0 0 1], Accuracy: 60.70%

Fig. 40. Deployment of Developed models on Gradio to test the accuracy directly with the test dataset

VIII. Conclusion and Future Scope

In conclusion, the project has achieved its objectives by developing machine learning models that deliver enhanced predictive performance, as substantiated by comprehensive evaluation metrics. The effective application of hyperparameter tuning has resulted in models that not only exhibit high accuracy but also maintain robustness across varied datasets. The deployment via Gradio has allowed for these sophisticated models to be accessible to users through straightforward and interactive web interfaces, bridging the gap between complex machine learning algorithms and practical usability. Looking to the future, the project holds a vast potential for expansion and refinement. There is an opportunity to further refine the models through ongoing optimization techniques and the exploration of cutting-edge algorithms. Deployment strategies can be evolved to incorporate more dynamic cloud services, ensuring scalability and responsiveness to an expanding user base. Enhancing the user experience remains a priority, with iterative interface design improvements driven by user feedback. The inclusion of broader and more diverse datasets will enable the models to learn from a wider array of patterns, potentially increasing their accuracy and generalizability. Introducing real-time analytics could offer users valuable insights into the models' decision-making processes, thereby fostering transparency and trust. With the growth of the user base, maintaining rigorous data security protocols will be paramount to safeguard user privacy. Furthermore, as edge computing becomes more prevalent, it could be harnessed to decrease latency and augment the efficiency of the deployed models. Finally, and perhaps most importantly, the project will continue to address the ethical considerations of AI, ensuring that the models operate fairly and without bias. The commitment to ethical AI practices will be at the forefront of the project as it evolves, ensuring that it not only leads with innovation but also with integrity and social responsibility.

Appendix A
GitHub Link

Github URL: https://github.com/nikhileasy420/MSDA/tree/main/DATA%20245

Appendix B
Youtube Link

Youtube URL: https://youtu.be/zRqTYkKYS6k

References

[1] H. Kopka and P. W. Daly, *A Guide to* , 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] K. Han et al., "A review of Approaches for Predicting Drug–Drug Interactions Based on Machine Learning," *Frontiers in Pharmacology*, vol. 12, Jan. 2022, doi: 10.3389/fphar.2021.814858. Available: https://doi.org/10.3389/fphar.2021.814858.

[3] "Papers with Code - node2vec: Scalable Feature Learning for Networks," Jul. 03, 2016. Available: https://paperswithcode.com/paper/node2vec-scalable-feature-learning-for.

[4] S. Bang, J. H. Jhee, and H. Shin, "Polypharmacy side-effect prediction with enhanced interpretability based on graph feature attention network," *Bioinformatics*, vol. 37, no. 18, pp. 2955–2962, Mar. 2021, doi: 10.1093/bioinformatics/btab174. Available: https://doi.org/10.1093/bioinformatics/btab174.

[5] M. Žitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*. Available: https://doi.org/10.1093/bioinformatics/bty294.

[6] N. P. Tatonetti, P. P. Ye, R. Daneshjou, and R. B. Altman, "Data-Driven prediction of drug effects and interactions," *Science Translational Medicine*, vol. 4, no. 125, Mar. 2012, doi: 10.1126/scitranslmed.3003377. Available: https://doi.org/10.1126/scitranslmed.3003377.