GNN- ENHANCED DRUG INTERACTION DETECTION AND ADVERSE REACTION ANALYSIS WITH SENTIMENT INSIGHTS

A PROJECT REPORT

Submitted by

Abisek Kamthan - RA2111027010211
Venkatadurga Pranesh - RA2111056010009
Harith Bala - RA2111056010039
Gaurang Srivastava - RA2111027010190

Under the Guidance of

Dr. SV. Shri Bharathi

Assistant Professor (Data Science and Business Systems)

in partial fulfillment of the requirementsfor the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in (DATA SCIENCE)

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEM
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY
KATTANKULATHUR- 603 203

NOVEMBER 2024

Department of Data science and Business systems

SRM Institute of Science & Technology

Own Work* Declaration Form

Degree/ Course : **B. TECH CSE (DATA SCIENCE), B. TECH CSE (BIG DATA ANALYTICS)**

Student Name : **Abisek kamthan, Venkatadurga Pranesh, Harithbala, Gaurang Srivastava**

Registration Number : **RA2111027010211, RA2111056010009, RA2111056010039, RA2111027010190**

Title of Work : **GNN- ENHANCED DRUG INTERACTION DETECTION AND ADVERSE REACTION ANALYSIS WITH SENTIMENT INSIGHTS**

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook /University website

I understand that any false claim for this work will be penalized in accordance with theUniversity policies and regulations.

DECLARATION:	
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.	
Abisek Kamthan [RA2111027010211] DATE:	Venkatadurga Pranesh [RA2111056010009] DATE:
Harithbala k [RA2111056010039] DATE:	Gaurang Srivastava [RA2111027010190] DATE:

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP107L - Minor Project report titled "**GNN-Enhanced Drug Interaction detection and adverse reaction analysis with sentiment insights**" is the bonafide work of "**Abisek Kamthan [RA2111027010211], Venkatadurga Pranesh B[RA211056010009], Harithbala[RA2111056010039], Gaurang Srivastava[RA2111027010190]** " who carried out the project work[internship] under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. SV. Shri Bharathi

SUPERVISOR
Assistant Professor
Department of
Data science and
Business system

SIGNATURE

Dr. V. Kavitha

PROFESSOR &HEAD
Department of
Data science and Business system

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. T. V. Gopal,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C. Lakshmi,** Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. V. Kavitha**, Professor, Department of Data science and Business systems, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor **Dr. P Rajasekhar**, **Dr. M Arthy** Department of Data Science and Business Systems, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide **Dr. SV. Shri Bharathi,** Department of Data science and Business systems, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Data science and Business systems, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

Abisek Kamthan - RA2111027010211
Venkatadurga Pranesh - RA2111056010009
Harith Bala - RA2111056010039
Gaurang Srivastava - RA2111027010190

ABSTRACT

As healthcare faces increasing challenges with polypharmacy, the need for advanced tools to detect and predict Drug-Drug Interactions (DDIs) and Adverse Drug Reactions (ADRs) has become critical. Conventional approaches primarily rely on molecular data to assess DDIs, often neglecting real-world patient experiences that can provide valuable insights into the true impact of drug interactions. This study proposes a novel framework that combines Graph Neural Networks (GNNs) with sentiment analysis to enhance the prediction of DDIs and ADRs, providing a comprehensive, patient-centered approach to drug safety.

Our model integrates GNNs to capture complex relationships between drugs, treating each drug as a node and interactions as edges in a graph, thus enabling the model to learn from both direct and indirect drug interactions. Additionally, sentiment analysis is employed to analyze patient-reported feedback on ADRs, using data from the FDA's Adverse Drug Reaction database. By incorporating patient perspectives through sentiment polarity and intensity, the model offers a nuanced understanding of ADR severity, which molecular data alone cannot achieve.

The methodology involves combining data from structured molecular interaction datasets with unstructured patient feedback, creating a powerful, hybrid system. The GNN is trained to predict both the likelihood and severity of DDIs, while the sentiment analysis model assesses ADRs based on patient experience. Performance evaluations, using metrics such as the Area Under the Precision-Recall Curve (AUPR) and Area Under the Curve (AUC), show that this integrated approach significantly improves predictive accuracy compared to traditional models. Results indicate that the hybrid model achieves high precision, recall, and F1 scores, demonstrating its robustness and applicability in real-world clinical settings.

This study's findings underscore the potential of merging clinical data with patient-reported outcomes to advance predictive models for drug safety. By offering an enriched understanding of both the molecular and experiential dimensions of drug interactions, our framework not only enhances the precision of DDI detection but also facilitates improved clinical decision-making, ultimately contributing to safer and more personalized healthcare.

.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

- *GNN: Graphical Neural Network*
- *DDI: Drug-Drug Interaction*
- *ADR: Adverse Drug Reaction*
- *NCBI: National center for Biotechnology Information*
- *FDA: United states Food and Drug Adminstartion*

CHAPTER 1

INTRODUCTION

1.1 Background

In the healthcare industry, the concurrent administration of multiple medications, known as polypharmacy, is increasingly common, particularly for patients with chronic and complex health conditions. However, this practice raises the risk of Drug-Drug Interactions (DDIs), which occur when the combined effects of different drugs lead to adverse reactions or diminished efficacy. Adverse Drug Reactions (ADRs) stemming from such interactions are a significant concern, often resulting in increased healthcare costs, hospitalizations, and compromised patient safety. Thus, accurate and comprehensive methods for DDI and ADR prediction are critical for ensuring effective and safe treatment.

Traditional approaches to predicting DDIs often focus on laboratory-based molecular data, which identifies possible interactions at the chemical or structural level. While these models have proven useful, they frequently overlook valuable real-world data—particularly, feedback directly from patients experiencing ADRs. Patient experiences, recorded in regulatory databases such as the FDA's Adverse Drug Reaction database, reveal the subjective impact and severity of drug interactions that laboratory data alone cannot fully capture. This limitation has created a need for more sophisticated predictive models that can integrate both clinical and experiential data, thereby providing a holistic view of drug interactions and their impacts.

1.2 Problem Statement

Existing DDI models predominantly rely on molecular data and, therefore, lack insights into patient-reported outcomes. This limitation restricts their predictive power and practical application in real-world settings, where patient experiences are crucial for understanding the broader implications of drug interactions. To address these gaps, this study presents a hybrid approach, combining Graph Neural Networks (GNNs) for DDI prediction with sentiment analysis on patient feedback for ADR evaluation. This integration aims to create a more robust and patient-centered framework for identifying high-risk interactions and potential adverse outcomes.

1.3 Objectives

The primary objectives of this study are:

1. To develop an integrated model combining GNN for structural DDI prediction and sentiment analysis for ADR severity assessment.
2. To improve prediction accuracy by leveraging both molecular interaction data and real-world patient feedback.
3. To provide a more comprehensive tool for drug safety that can aid healthcare professionals in making informed clinical decisions.

1.4 Scope of Study

This study leverages two key data sources: the National Center for Biotechnology Information (NCBI) Drug-Drug Interaction (DDI) dataset and the FDA's Adverse Drug Reaction (ADR) database. By using the GNN to represent the complex relationships in the DDI dataset and sentiment analysis to interpret patient feedback in the ADR database, we create a comprehensive model that not only predicts the likelihood of interactions but also assess their severity based on real – world patient experiences. This approach aims to bridge the gap between laboratory research and Practical, patient-centered applications in healthcare.

1.5 Sustainability Goal

The integration of personalized, predictive models in healthcare aligns closely with global sustainability efforts, particularly the United Nations Sustainable Development Goals Responsible Consumption and Production. By leveraging artificial intelligence to improve drug safety, our research promotes a more sustainable and equitable healthcare system.

1.6 Sustainable Healthcare Practices

1. **Reducing Waste in Drug Administration:**

   Traditional approaches to DDI prediction often result in generalized treatment recommendations that may not fully account for individual patient factors. This can lead to overprescription or the use of unnecessary medications, which contributes to waste within healthcare systems. By predicting potential interactions more accurately, our model helps optimize drug administration, ensuring that treatments are more precisely targeted to the individual, thus reducing unnecessary drug use and associated waste.

2. **Promoting Patient-Centred Care:**

   Integrating patient feedback allows healthcare providers to better understand the real-world effects of drug interactions, promoting a model of care that considers individual experiences and preferences. This patient-centered approach enhances healthcare equity and accessibility, enabling more people to receive accurate and personalized treatment recommendations.

3. **Supporting Preventive and Proactive Health Measures:**

   The hybrid GNN-Sentiment model is designed to predict adverse drug interactions proactively, offering a tool that helps healthcare professionals anticipate potential risks before they manifest in severe ADRs. This proactive approach to healthcare aligns with sustainable health practices, as it reduces the

burden of chronic conditions and encourages preventative care, which can ultimately lead to longer-term positive health outcomes for communities.

1.7 Alignment with Global Health Goals

Our research aligns with the following United Nations Sustainable Development Goals:

- **Goal 3:** Good Health and Well-being – By providing a tool that enhances drug safety and improves clinical decision-making, our model supports improved health outcomes and promotes safer, more effective treatment options.
- **Goal 12:** Responsible Consumption and Production – The model encourages responsible drug use by minimizing the need for unnecessary medications and reducing the risk of drug overuse, thereby promoting a more efficient and sustainable approach to healthcare resource utilization.

In sum, this study represents a significant step toward sustainable healthcare by improving the precision and safety of drug therapies while promoting a patient-centered approach that aligns with global health and sustainability goals.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING SYSTEMS

Current systems for drug interaction detection and adverse reaction analysis are primarily based on structured molecular data and often rely on traditional machine learning or rule-based algorithms. These systems fall into three main categories:

1. **Rule-Based Systems:**

   These systems use predefined rules and guidelines derived from clinical pharmacology and molecular interaction research. Rule-based DDI systems, such as those found in drug interaction databases, provide insights based on fixed interactions between specific drug molecules. While straightforward, these systems struggle with adaptability and are limited by their reliance on pre-existing knowledge.

2. **Electronic Health Record (EHR)-Based Systems:**

   EHR-based systems collect patient data, including medication history, lab results, and observed ADRs, providing valuable insights for personalized medicine. These systems allow healthcare providers to view patient-specific information on potential interactions. However, they often function as data repositories rather than analytical tools, limiting their predictive capabilities.

3. **Basic AI-Driven Models:**

   Early AI models, such as logistic regression or decision tree classifiers, analyze drug interaction patterns using structured data from interaction databases. While these models can detect common interaction patterns, they lack the ability to interpret unstructured data, such as patient feedback, which is crucial for

assessing real-world ADR severity. Furthermore, these models are often limited by generalized training data, which may not reflect the diverse patient populations seen in real-world clinical settings.

2.2 CHALLENGES WITH THE EXISTING SYSTEM

Despite their contributions to patient safety, existing systems for DDI and ADR prediction face several limitations, impacting their accuracy and real-world applicability:

1. **Lack of Patient-Centered Data:**

   Traditional systems focus almost exclusively on molecular interactions or structured clinical data, overlooking valuable insights from real-world patient feedback. As a result, these systems may fail to capture the full impact of drug interactions on patients, particularly in cases where the ADR severity is influenced by individual patient characteristics.

2. **Limited Adaptability:**

   Rule-based systems and early AI models rely on predefined drug interaction rules or generalized training data, making them rigid and unable to adapt to emerging drug interaction patterns. This lack of flexibility limits their ability to address new, complex interactions that may arise with newly approved medications or in polypharmacy.

3. **Inadequate Analysis of Unstructured Data:**

   Many DDI and ADR systems lack the advanced natural language processing (NLP) capabilities required to interpret unstructured patient feedback or free-text ADR reports. This limitation means that patient-reported ADRs, which often contain nuanced descriptions of adverse effects, are not utilized effectively in predictive modelling.

4. **Dependency on Data Quality and Consistency:**

   Systems that rely on EHR data are often dependent on the accuracy and consistency of data entry. Variations in documentation practices, missing data, or human errors can lead to inaccuracies in drug interaction predictions and ADR assessments, affecting the reliability of these models.

5. **Reactive Rather than Proactive:**

   Existing DDI systems tend to operate reactively, flagging interactions only after ADRs have been reported. This approach limits the system's ability to prevent adverse outcomes, as it does not enable proactive identification of high-risk interactions based on patient feedback trends or patterns.

2.3 KEY FEATURES OF PROPOSED SYSTEM

The proposed system addresses the limitations of traditional DDI and ADR models by combining Graph Neural Networks (GNNs) with sentiment analysis. This hybrid approach enables a comprehensive, patient-centered framework for predicting drug interactions and assessing adverse reaction severity. Key features include:

1. **Sophisticated Interaction Prediction Using GNNs:**

- Synergistic Approach: The GNN model leverages structured molecular interaction data from established DDI databases, treating drugs as nodes and interactions as edges in a complex graph. This approach enables the model to capture not only direct interactions but also indirect relationships and their combined effects.
- Contextual Interaction Detection: Unlike rule-based models, the GNN can recognize and interpret varying combinations of drug interactions, accounting

for nuanced or complex relationships that may not be captured in predefined rules.

2. **Incorporation of Patient-Reported Data via Sentiment Analysis:**

- Sentiment Analysis on ADR Feedback: By analyzing patient feedback from ADR databases, the sentiment analysis model captures the intensity and polarity of adverse reactions. This addition brings a subjective layer to the model, allowing it to understand the real-world impact of drug interactions on patients.
- Severity Assessment and Adjustment: Patient sentiment data enables the model to adjust the severity of ADR predictions based on reported experiences, thus offering a more personalized approach to interaction risk assessment.

3. **Comprehensive, Patient-Centered Recommendations:**

- Detailed Health Insights: For each predicted interaction, the system provides an easy-to-understand summary of potential effects, including drug synergy, adverse symptoms, and suggested precautions. This approach empowers healthcare providers with clear, actionable insights for patient care.
- Targeted Risk Mitigation Strategies: Based on patient-reported ADR severity, the system can suggest tailored strategies for managing high-risk interactions, such as dosage adjustments or alternative medications.

4. **Scalable and Flexible Design:**

- Adaptability to Emerging Data: The model is designed to incorporate new drug interaction data and updated patient feedback, allowing it to evolve over time and remain relevant with ongoing changes in drug safety data.
- Modular Architecture: The system's architecture enables the GNN and sentiment analysis components to operate independently or in conjunction, allowing for flexible deployment and easy integration with existing healthcare systems.

5. **Proactive Drug Safety Tool:**

- Predictive Risk Alerts: The hybrid model's ability to predict interactions and assess ADR severity enables healthcare providers to take preventive measures before adverse reactions occur, shifting the approach from reactive to proactive.
- Continuous Learning Mechanism: With the integration of machine learning, the model can improve its accuracy over time based on real-world outcomes, enabling continuous enhancement of its predictive power.

By combining GNN-based DDI prediction with patient-centered sentiment analysis, the proposed system offers a more accurate, adaptable, and proactive approach to drug safety, ultimately advancing patient care and supporting safer, more personalized medication management.

2.4 RESEARCH OBJECTIVES

1. **Develop an Integrated Model for Drug Interaction and Adverse Reaction Prediction**

   Design a hybrid predictive model that combines Graph Neural Networks (GNNs) with sentiment analysis to enhance the accuracy of Drug-Drug Interaction (DDI) and Adverse Drug Reaction (ADR) predictions. The GNN component will utilize molecular interaction data to identify both direct and complex relationships between drugs, while sentiment analysis will process patient-reported feedback to assess the severity of ADRs.

2. **Leverage Real-World Patient Feedback for ADR Severity Analysis**

   Incorporate unstructured patient feedback from the FDA's Adverse Drug Reaction (ADR) database using sentiment analysis. This objective aims to capture the polarity (positive, neutral, negative) and intensity of patient-reported experiences, providing a nuanced understanding of ADR severity that complements molecular interaction data.

3. **Optimize Prediction Accuracy and Clinical Applicability of DDI and ADR Models**

   Implement feature engineering techniques such as TF-IDF vectorization for text data and embeddings for drug properties to improve model performance. By measuring the accuracy, AUC, AUPR, F1 score, and other relevant metrics, this objective focuses on evaluating and enhancing the overall performance of the hybrid model, ensuring its clinical reliability and usability.

4. **Enhance Proactive Healthcare with Predictive Risk Alerts**

   Develop a proactive system capable of identifying potential high-risk drug interactions before adverse events occur. The model will generate predictive alerts that assist healthcare professionals in making informed decisions, minimizing the risk of ADRs for patients on multiple medications.

6. **Create a Scalable and Adaptable Model Architecture for Continuous Learning**

   Build a flexible architecture that allows for the integration of new data sources, model retraining, and updates. This objective aims to ensure that the model remains relevant and accurate with ongoing changes in drug interaction data and patient-reported ADRs, supporting long-term applicability in real-world healthcare settings.

2.5 PLAN OF ACTION

1. **Project Setup, Data Collection, and Preprocessing**

   The project begins with a detailed planning phase where the core features—DDI prediction, ADR analysis via sentiment insights, and risk alerts—are defined. This step involves identifying technical requirements, including hardware,

software, and specific libraries for GNN and NLP tasks. Data collection draws from both structured sources like the NCBI for molecular DDI data and unstructured patient-reported ADR feedback from the FDA. Preprocessing transforms DDI data into graph structures with normalized drug names, while ADR text data undergoes tokenization, stop word removal, and feature extraction to prepare for sentiment analysis.

2. **Model Development and Integration**

The model development phase is divided into creating a GNN for DDI prediction and implementing a sentiment analysis model for ADR assessment. The GNN architecture is constructed to represent drugs and their interactions, which the model learns to classify by severity and interaction type. In parallel, an NLP model, potentially a logistic regression or transformer-based model, is trained to assess ADR severity by analysing patient-reported sentiment polarity and intensity. Once these models are individually trained, they are integrated into a unified pipeline where DDI predictions from the GNN are adjusted based on the ADR feedback provided by the sentiment analysis model.

3. **System Prototyping, Testing, and Optimization**

System architecture design and prototyping involve creating a modular structure that facilitates data flow from ingestion through to output, with an interface for inputting interactions and displaying results. Testing follows, using a subset of DDI and ADR data to ensure smooth integration between the models, while debugging any data flow or prediction issues. In the final evaluation and optimization phase, hyperparameter tuning, such as adjusting learning rates and dropout rates, is applied to enhance performance. Model evaluation uses metrics like AUC, AUPR, and F1-score, followed by error analysis to refine feature engineering, model layers, and other components for precision.

CHAPTER 3

SYSTEM ARCHITECTURE

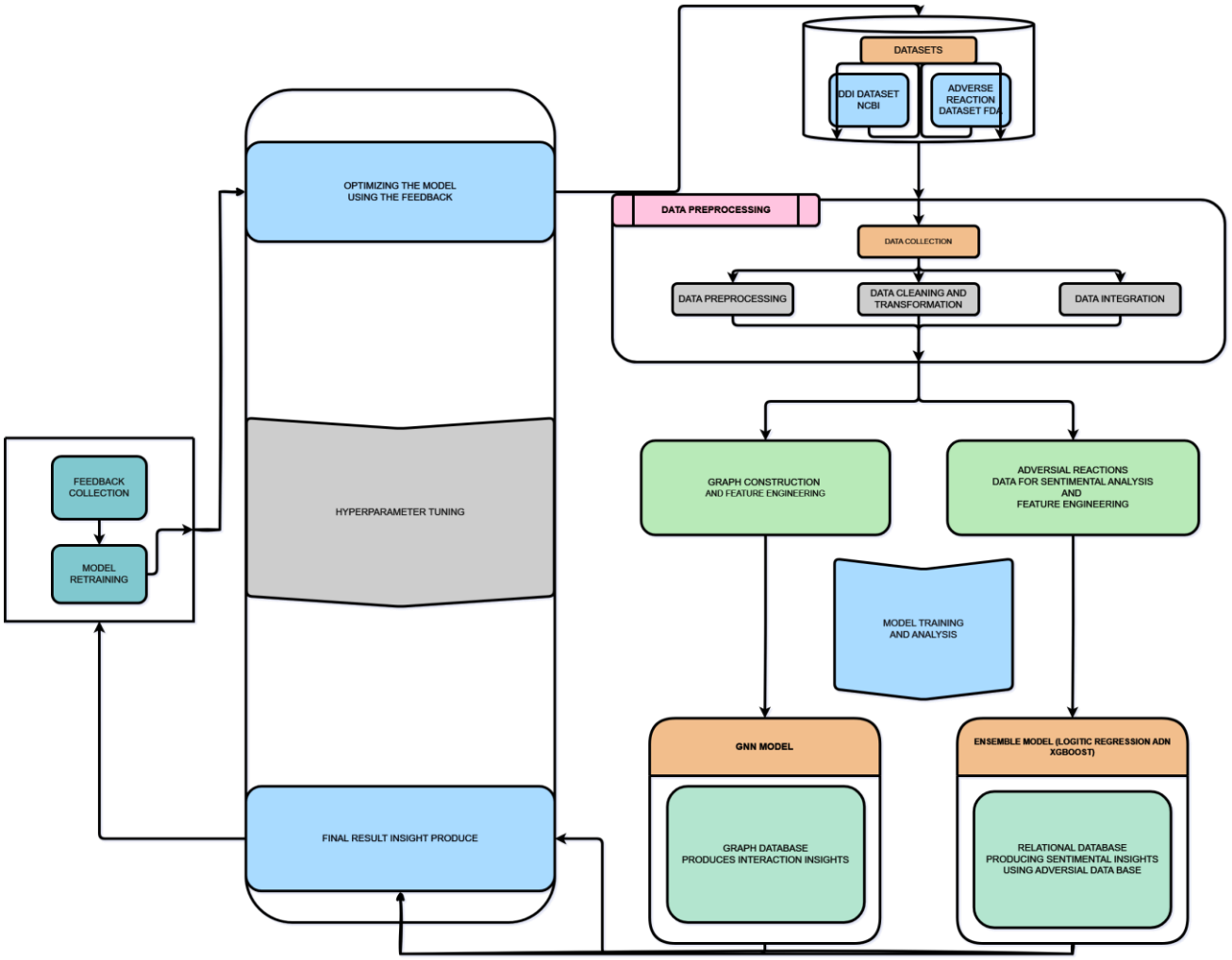

Figure 1: Architecture Diagram

1. Data Source Layer

The **Data Source Layer** integrates three critical data inputs: the Drug-Drug Interaction (DDI) Dataset, the FDA Adverse Effects Dataset, and User Feedback Data. The **DDI Dataset** is sourced from the National Center for Biotechnology Information (NCBI) and includes pharmacological details about drug interactions, providing foundational data for constructing the interaction graph. Each entry in the DDI dataset lists specific drug pairs, interaction types, severity levels, and any

clinically significant notes. This dataset enables the identification of drug nodes and the edges that connect them, representing interactions. The attributes in this dataset form the base of the graph structure, where each drug acts as a node and each interaction as an edge, facilitating detailed drug interaction modeling.

The **FDA Adverse Effects Dataset** includes a collection of TSV files detailing adverse drug reactions and patient outcomes, obtained directly from FDA sources. These data files are essential for severity analysis, as they capture demographic-specific adverse reactions and interaction patterns. A sample format of the adverse reaction dataset is as follows:

File	Adverse Reaction	Severity
ADCETRIS	Peripheral Neuropathy	Severe
ADCETRIS	Anaphylaxis	Severe
ADCETRIS	Infusion Reactions	Severe
ADCETRIS	Hematologic Toxicities	Severe
ADCETRIS	Infections	Severe

Lastly, the **User Feedback Data** comprises real-world, user-submitted feedback, capturing sentiment-laden comments about drug interactions and adverse reactions. This unstructured feedback provides insights into patient experiences, highlighting common concerns and interaction effects that may not be documented in clinical datasets. Sentiment analysis tools use this feedback to determine the polarity (positive, negative, or neutral) and intensity of user sentiments, enriching the adverse reaction data. This user-centric feedback layer enhances the GNN model's adaptability, adding a human-centric dimension to adverse reaction analysis.

2. Data Ingestion & Preprocessing Layer

The **Data Ingestion & Preprocessing Layer** is responsible for collecting, standardizing, and preparing data for downstream analysis. In the **Data Collection** stage, data from the DDI dataset, FDA adverse effects files, and user feedback are ingested and transformed into standardized formats to enable consistent processing across all sources.

The **Cleaning & Transformation** process then applies multiple data preprocessing steps to ensure data quality. This includes removing duplicate records to maintain consistency, imputing missing values where necessary, and conducting text preprocessing on user feedback data. Key text processing steps include tokenization (breaking text into individual words), removing stopwords (common words that don't add meaning, such as "the" and "is"), and lemmatization/stemming (reducing words to their root forms), which improves consistency for sentiment analysis.

The **Data Integration** stage merges all three datasets to create a comprehensive, multi-relational structure that retains key fields, such as drug IDs, interaction types, patient demographics, and outcomes. Schema mapping ensures that fields from different sources align properly, allowing each drug, interaction type, and demographic variable to be uniformly referenced across datasets. Data Aggregation is employed to combine the data meaningfully, facilitating the creation of a single schema that captures drug interaction and adverse effect information, enriched with sentiment data.

Finally, **Indexing & Optimization** improves retrieval and processing efficiency. Indexing frequently queried fields, such as drug names and interaction types, enhances query performance, while dimensionality reduction techniques (such as

Principal Component Analysis, PCA) minimize the computational load for downstream models, especially for high-dimensional sentiment features. This stage ensures that data is prepared in a format conducive to both relational and graph-based analysis, optimizing computational performance.

3. **Graph Construction Layer**

   The **Graph Construction Layer** constructs a drug interaction graph that models the relationships between drugs as nodes and interactions as edges. In the **Drug Interaction Graph**, nodes represent individual drugs, with attributes such as drug name, category, and associated side effects. The edges between nodes indicate specific interactions between drug pairs, classified by type (e.g., synergistic, antagonistic, or neutral), with edge weights indicating interaction severity or frequency of occurrence.

   Interactions are represented as weighted edges, where weights derived from FDA and DDI data reflect the interaction's severity or probability. The graph may include both undirected edges, representing bidirectional interactions, and directed edges for interactions that have directionality (e.g., when one drug amplifies another's effects). Complex relationships, such as multi-faceted interactions or cases where a drug interacts with itself under specific conditions, are accommodated by allowing self-loops and multi-edges.

4. **Feature Engineering** is performed to generate meaningful interaction features, such as the frequency of reported adverse effects and demographic-specific interaction tendencies. Calculated graph metrics, such as node degree (representing the number of interactions each drug has), clustering coefficient (indicating the tendency of drugs to cluster), and graph centrality (highlighting influential drugs based on their position in the network), provide further insights into interaction risks.

These metrics are essential for assessing the potential impact of drugs within the interaction network, particularly when identifying drugs that participate in frequent or high-risk interactions. Finally, **Graph Storage** employs a graph database, which efficiently stores and retrieves the graph structure, allowing fast edge traversal and neighborhood querying. This structure is essential for GNN processing, as it provides a high-performance environment for exploring relational data within the graph.

5. **Model Training & Analysis Layer**

The **Model Training & Analysis Layer** includes a Graph Neural Network (GNN) for predicting drug interactions and an ensemble sentiment analysis model for processing user feedback. The **Graph Neural Network** model is designed to learn interaction patterns within the drug network by aggregating features from each drug's neighboring nodes.

Multiple convolutional layers are employed to enable the GNN to progressively integrate local and global interaction information, facilitating accurate predictions for potential new interactions. During training, the GNN receives graph-based data where drugs are nodes, interactions are edges, and each edge is weighted by interaction severity.

The model outputs predictions for adverse interaction likelihood, leveraging both existing and learned patterns across the graph. This edge classification enables the GNN to identify and classify interactions based on interaction severity, contributing to comprehensive risk assessment.

The **Ensemble Sentiment Analysis** model processes user feedback to classify sentiment and enhance model predictions with patient-centered insights. User

feedback is vectorized through methods like TF-IDF or word embeddings, transforming text into numerical representations suitable for model processing.

The ensemble model includes **Logistic Regression** for capturing general sentiment patterns and **XGBoost** to detect non-linear relationships in complex feedback data. By aggregating these models, the ensemble output balances general sentiment predictions with advanced pattern recognition, assigning higher confidence to classifications when the models agree. The sentiment output includes both polarity (positive, negative, neutral) and intensity scores, providing valuable context to the GNN's interaction predictions.

6. **Data Storage Layer**

   The **Data Storage Layer** consists of two primary databases: a **Graph Database** and a **Relational Database**. The Graph Database, such as Neo4j, organizes drugs as nodes and interactions as edges, supporting graph operations that allow for efficient traversal and querying. This structure is designed to complement the GNN model, enabling neighborhood searches and relational lookups as required for model processing. Primary nodes, representing drugs, are indexed to expedite complex queries, particularly those used in GNN training and prediction.

   In parallel, the **Relational Database** (e.g., MySQL) stores structured data, including user feedback, demographic information, and case-specific adverse event records. This database supports SQL-based querying, enabling quick access to specific attributes such as drug names, demographic profiles, or adverse reaction types, enhancing the system's relational data management.

7. **User Interaction & Reporting Layer**

   The **User Interaction & Reporting Layer** provides a user-friendly interface and automated reporting features, ensuring that critical insights are accessible to end-

users. The **Reporting & Notifications** component generates reports on high-risk drug interactions, using insights from both the GNN model and sentiment analysis to produce comprehensive alerts.

This includes real-time notifications for interactions classified as high-risk, providing actionable alerts to clinicians or users for prompt decision-making. The **User Interface** features graphical visualizations, displaying the drug interaction graph with nodes and edges color-coded by severity. This visualization aids users in interpreting interaction risk visually. Additionally, a feedback interface allows users to submit real-world feedback or interact with the report data directly, enhancing user engagement and informed decision-making.

8. **Continuous learning & Feedback Layer**

   The **Continuous Learning & Feedback Layer** ensures that the system remains adaptive and up to date with evolving data trends. A **Feedback Loop for Continuous Learning** collects user feedback on adverse reactions and interaction risks, integrating real-world experience into model development. This feedback data is validated against existing model predictions to assess alignment with real-world outcomes, providing crucial feedback to the model. Periodic **Model Retraining** allows the GNN and sentiment analysis models to adapt over time by incorporating new data and improving prediction accuracy.

   Dynamic Adjustment further refines the ensemble model's weighting when feedback reveals prediction biases, while the GNN parameters are fine-tuned to better capture evolving drug interaction patterns. This continuous learning cycle enhances model robustness, ensuring that predictions remain reliable and accurate as new data is integrated.

CHAPTER 4

SPRINT PLANNING AND EXECUTION METHODOLOGY

4.1 SPRINT ANALYSIS

The project employs a sprint-based approach, dividing the work into distinct, goal-oriented phases. Each sprint builds upon the previous, allowing for systematic progress and iterative refinements. Here's a comprehensive breakdown of each sprint:

Sprint 1: Data Acquisition and Preprocessing

- **Objective**: Gather and preprocess data from the NCBI Drug-Drug Interaction (DDI) dataset and FDA Adverse Drug Reaction (ADR) database to form the foundational datasets.
- **Tasks**:
    - **Data Collection**: Download the NCBI DDI dataset, which includes drug interaction pairs, interaction types, and severity information. Retrieve the FDA ADR database, which contains reports of adverse reactions experienced by patients.
    - **Data Cleaning**: Address missing values, duplicates, and inconsistencies in each dataset. Remove any irrelevant data points or fields that do not contribute to interaction or adverse reaction analysis.
    - **Data Standardization**: Ensure uniform data types and formats to facilitate integration (e.g., standardizing drug names, converting text formats).
    - **Preprocessing for Sentiment Analysis**: Tokenize and clean text data from the FDA ADR dataset, including removal of punctuation, stopwords, and any personal identifiers.
- **Deliverables**: Cleaned and preprocessed DDI and ADR datasets in a compatible format for further analysis.
- **Checkpoints**:
    - **Dataset Review**: Verify the integrity and quality of cleaned data, ensuring there are no inconsistencies.
    - **Approval Milestone**: Final validation of datasets for accuracy and completeness before proceeding to the next sprint.

Sprint 2: Dataset Integration and Feature Engineering

- **Objective**: Create a unified dataset by combining DDI and ADR data, incorporating essential features that capture both molecular-level interactions and patient-reported experiences.
- **Tasks**:
    - **Dataset Merging**: Integrate the two datasets, ensuring alignment in terms of shared attributes, such as drug identifiers and interaction labels.
    - **Feature Engineering**:
        - **Interaction Features**: Encode interaction types (e.g., synergistic, antagonistic, neutral) and severity levels from the DDI dataset.
        - **Patient Feedback Features**: Incorporate sentiment scores, patient-reported adverse reactions, and reaction intensities as extracted from the FDA ADR dataset.
        - **Interaction Context**: Add contextual features, such as patient demographics or common drug combinations, to capture nuances in interaction.
- **Deliverables**: A comprehensive, feature-rich unified dataset that captures the complexity of drug interactions and patient feedback.
- **Checkpoints**:
    - **Dataset Inspection**: Validate the combined dataset to ensure accurate alignment of attributes.
    - **Feature Validation**: Conduct a correlation analysis on features to verify their contribution to predictive capabilities.

Sprint 3: Model Development with Graph Neural Networks (GNN)

- **Objective**: Design and train a Graph Neural Network (GNN) model capable of identifying and predicting complex drug-drug interactions based on the integrated dataset.
- **Tasks**:
    - **Model Architecture Design**: Define the GNN model structure, where each drug represents a node, and edges denote interactions. Customize the network to include:
        - **Graph Convolutional Layers**: Build multiple convolutional layers (e.g., 64 layers) to capture local and global interaction patterns.
        - **Edge Classification**: Integrate an edge classifier that categorizes interaction types and predicts severity.

- **Model Training**: Train the GNN model using cross-entropy loss for interaction classification and mean squared error (MSE) for severity prediction.
- **Hyperparameter Tuning**: Experiment with learning rates, dropout rates, and batch sizes to improve model performance.

- **Deliverables**: A trained GNN model capable of predicting the types and severity of drug-drug interactions.
- **Checkpoints**:
  - **Model Performance Review**: Evaluate the GNN model's performance using metrics like accuracy, AUPR, and F1-score.
  - **Overfitting Mitigation**: Ensure robustness by adjusting dropout and regularization parameters to prevent overfitting.

Sprint 4: Sentiment Analysis Model Development

- **Objective**: Develop a sentiment analysis model using logistic regression combined with XGBoost ensemble learning to classify patient-reported outcomes and capture sentiment intensity.
- **Tasks**:
  - **Text Preprocessing**: Tokenize, clean, and apply word embeddings to convert text from the FDA ADR dataset into numerical representations suitable for analysis.
  - **Model Architecture**:
    - **Logistic Regression and XGBoost Combination**: Set up a logistic regression model enhanced by XGBoost to capture nuanced sentiment signals.
    - **Sentiment Classification**: Train the model to predict sentiment polarity (positive, negative, neutral) and intensity.
  - **Model Training and Evaluation**: Train the sentiment analysis model, using labeled data to fine-tune its ability to accurately predict sentiment and correlate it with adverse reactions.
- **Deliverables**: A sentiment analysis model that provides sentiment polarity and intensity scores for patient feedback, enhancing the contextual relevance of interaction predictions.
- **Checkpoints**:
  - **Model Accuracy Check**: Validate the sentiment model's precision, recall, and F1-score.

- **Integration Readiness**: Prepare the sentiment model for combination with the GNN predictions.

Sprint 5: Model Integration and Final Evaluation

- **Objective**: Combine predictions from the GNN and sentiment analysis models to create an integrated framework that provides a holistic view of drug interactions and their clinical implications.
- **Tasks**:
  - **Model Fusion**: Combine GNN-based predictions on molecular interactions with sentiment-based insights from patient feedback to produce a composite risk score.
  - **Comprehensive Testing**: Evaluate the unified model's performance using metrics such as AUC, AUPR, accuracy, precision, and recall.
  - **Comparative Analysis**: Compare the performance of the combined model with existing state-of-the-art methods (e.g., DANN-DDI, CNN-DDI).
- **Deliverables**: A fully integrated predictive model with improved accuracy and reliability for identifying drug interactions and adverse reactions.
- **Checkpoints**:
  - **Final Model Metrics**: Verify that the integrated model meets or exceeds predefined performance benchmarks.
  - **Cross-Validation**: Ensure robustness and generalizability through cross-validation.

Sprint 6: Reporting and Documentation

- **Objective**: Document findings, challenges, and insights from each phase, creating a detailed report to communicate the project's outcomes and recommendations.
- **Tasks**:
  - **Results Summary**: Present the findings, comparing the performance of the new model with baseline approaches.
  - **Challenges and Solutions**: Document any issues encountered, and the strategies used to overcome them.
  - **Recommendations for Future Work**: Outline potential enhancements, such as disease-specific models or continuous learning mechanisms.
- **Deliverables**: A comprehensive project report detailing methodology, results, and implications for drug safety.

- **Checkpoints**:
    - **Final Report Review**: Validate completeness and clarity of the documentation before submission.

4.2 EXECUTION METHODOLOGY

This study presents a novel framework that integrates Graph Neural Networks (GNNs) with Sentiment Analysis to enhance the detection of Drug-Drug Interactions (DDIs) and adverse reactions. While previous approaches have utilized GNNs for DDI prediction, our model is unique in its incorporation of patient-reported outcomes through sentiment analysis, providing a more holistic view of drug safety and effectiveness.

1. **DATASET:**

In this section, we describe the datasets used for the study, detailing their sources, structure, and preprocessing steps. The Drug-Drug Interaction (DDI) Dataset was sourced from the National Centre for Biotechnology Information (NCBI). This dataset includes a curated collection of drug interaction information, which is essential for understanding molecular-level interactions between various drugs. It is a widely recognized resource in bioinformatics and pharmacological research, aiding in the analysis of drug behavior and interaction mechanisms.

The second dataset, the FDA Adverse Drug Reaction (ADR) Database, was obtained from the United States Food and Drug Administration (FDA). This database contains reports of adverse drug reactions, submitted by healthcare professionals and consumers, offering valuable insights into the effects of drugs in real-world clinical settings. Both datasets underwent preprocessing to ensure consistency and suitability for analysis, with particular focus on standardizing drug names, handling missing data, and merging relevant interaction and ADR information for integration into the predictive models.

2. DATASET COMBINATION AND SYNERGY:

The interaction between these two datasets is crucial for the study's efficacy. The DDI dataset enables the prediction of potential interactions based on scientific models, while the FDA dataset provides real-world confirmation or contradiction of these prediction. A drug combination predicted to have an effect in the DDI dataset can be verified against the FDA dataset to determine if patients report increased benefits or adverse effects. The severity levels from the DDI dataset can be compared with actual outcomes from the FDA dataset, enabling an evaluation of the accuracy and practical relevance of molecular predictions. By integrating these datasets, it is possible to develop a comprehensive model capable of predicting not only drug interactions but also the likelihood and intensity of adverse reactions in real-world situations.

3. ARICHECTURE DIAGRAM:

The heart of our methodology is the Graph Neural Network (GNN), specifically designed to model Drug-Drug Interactions (DDIs) due to its ability to efficiently process graph-structured data. In our framework, the GNN treats drugs as nodes and their interactions as edges, allowing it to capture the intricate relationships that exist between various drugs and their combined effects. This graph-based representation is particularly advantageous for our study as it enables the GNN to discern not only the direct interactions between pairs of drugs but also the broader network of interactions in which these drugs are involved. The input to the GNN is a constructed graph, where each node represents a drug and each edge denotes the type of interaction whether synergistic, antagonistic, or neutral between them.

4. SENTIMENT ANALYSIS FOR ADVERSE DRUG REACTION:

Incorporating sentiment analysis into our framework is essential for integrating real-world patient feedback, which significantly enhances the model's ability to predict adverse reactions to medications. By analysing patient-reported outcomes from the FDA Adverse Drug Reaction (ADR) database, the sentiment analysis component captures the human experiences and clinical manifestations associated with drug interactions. This addition provides a more comprehensive understanding of how drugs affect patients, allowing for improved predictions of adverse reactions that may not be evident from molecular data alone.

The text preprocessing phase is critical for ensuring the quality of the input data for sentiment analysis. Initially, the text data is tokenized, which involves breaking the sentences into individual words or tokens. During this process, punctuation and stop words (commonly used words that may not contribute significant meaning, such as "and" "the," "is," etc.) are removed. This cleaning process enhances the focus on the meaningful content of the patient feedback, preparing it for further analysis.

Following tokenization, feature extraction is performed. The processed text is converted into numerical features using word embeddings, a technique that represents words in a continuous vector space where semantically similar words are closer together. This transformation allows the sentiment analysis model to operate effectively on the textual data, facilitating the integration of patient feedback with the predictions generated by the Graph Neural Network (GNN).

For the sentiment analysis, we employ a logistic regression classifier. This model is trained on the pre-processed patient feedback, enabling it to predict the likelihood of severe adverse reactions based on sentiment polarity and intensity. Sentiment polarity refers to the emotional tone conveyed in the text (positive, negative, or neutral), while intensity reflects the strength of that sentiment. The predictions made by the logistic regression model are then integrated with those from the GNN, enhancing the overall accuracy of our framework in predicting adverse drug reactions.

5. UNIFIED MODEL ARCHITECTURE:

By combining these two methodologies, our project creates a framework that is more comprehensive than either approach alone. The GNN enables us to predict drug interactions, and their severities based on molecular data, while sentiment analysis offers a patient-centred perspective, capturing real- world adverse reactions. This integration ensures that our model can not only predict the likelihood of drug interactions but also assess their clinical implications from both a molecular and a patient experience standpoint.

The predictions from both the GNN and sentiment analysis models are combined to form a more accurate and reliable recommendation system. The GNN provides insight into the potential interactions and severity of drug combinations, while the sentiment analysis layer adjusts these predictions based on patient-reported outcomes. This allows for more personalized and context-aware recommendations, addressing the limitations of previous models that only consider one type of data.

For example, if a drug combination is predicted to be synergistic and beneficial by the GNN, but patient feedback indicates frequent adverse reactions, our system would highlight the potential risks associated with this combination, thereby enhancing drug safety. This dual approach ensures that we not only predict drug interactions with accuracy but also account for the actual impact of these drugs on patients, improving clinical decision-making.

CHAPTER 5
RESULTS AND DISCUSSION

5.1 EXECUTION METHODOLOGY

The proposed model integrates Graph Neural Networks (GNN) with sentiment analysis to enhance the prediction of Drug-Drug Interactions (DDIs) and identify Adverse Drug Reactions (ADRs). The GNN component enables the model to analyze complex network relationships between drugs, while sentiment analysis assesses patient feedback and adverse event reports. By analyzing sentiment data, which includes real-world patient experiences, the model gains additional insights into adverse effects, thereby improving the accuracy and effectiveness of DDI predictions. This fusion of GNN with sentiment analysis offers a novel approach to leveraging patient-reported data in pharmaceutical safety and interaction prediction.

5.2 EVALUATION METRICS FOR GNN MODEL

To assess the performance of our model in predicting DDIs and their adverse outcomes, several evaluation metrics are used:

- Precision - measures the model's accuracy in identifying true positive interactions.
- Recall - indicates the model's capability to capture all relevant drug interactions.
- Accuracy - the overall correctness of the predictions.
- F1-score - balances precision and recall.
- Area Under the ROC Curve (AUC) - represents the true positive rate versus the false positive rate.
- Area Under the Precision-Recall Curve (AUPR) - prioritizes precision and recall specifically on positive samples, making it especially valuable since

known drug-drug interactions represent a small fraction of all possible interactions.

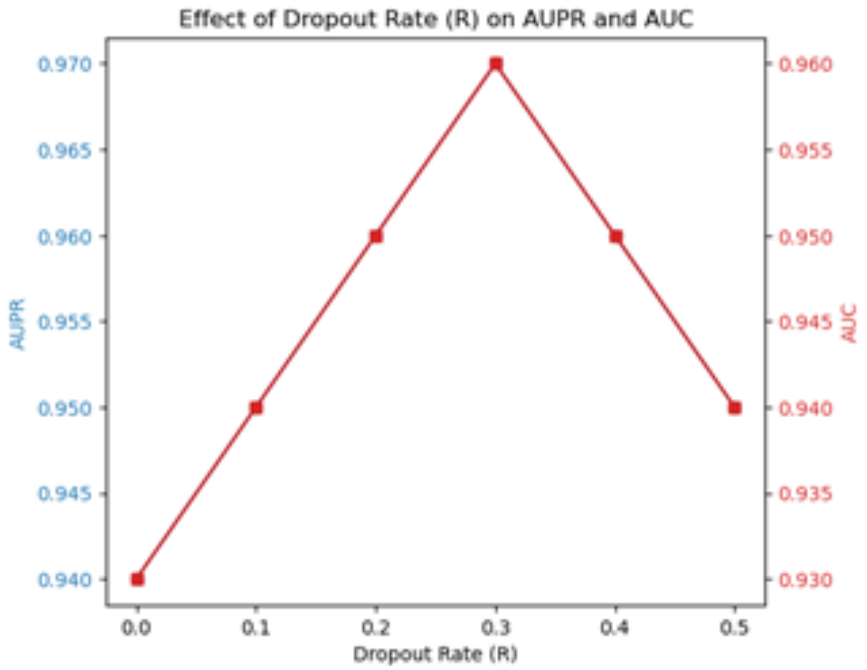

Figure 2: AUPR

1. **Explanation of AUPR Focus**: Given the low proportion of positive interactions in the dataset, AUPR is used as the primary performance metric. It ensures that the model performs well in recognizing positive drug interactions amidst many negative samples. The results in **Figure 2** suggest that these dependent and independent variables allowed us to pinpoint the best oriented architecture of the network.

2. **Cross-Validation:** A 5-fold cross-validation process is implemented to ensure the robustness of our results. This technique divides the data into five subsets, using four for training and one for testing in each iteration. By repeating this process five times, the model's generalizability is tested on different data partitions, minimizing overfitting risks and ensuring consistent performance across unseen samples.

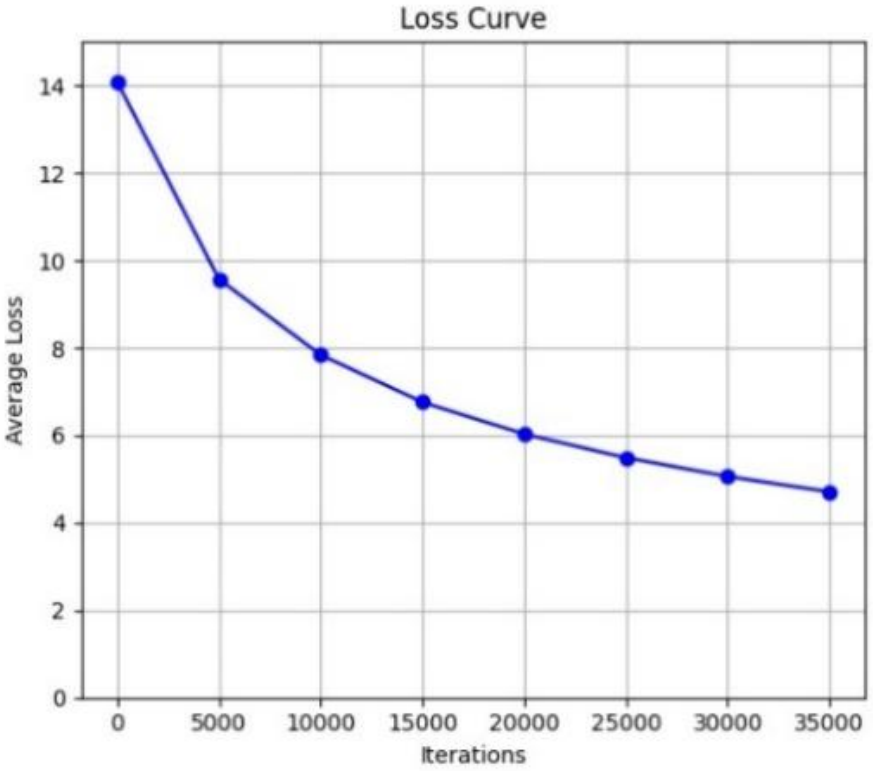

Figure 3: Loss curve

Loss Accuracy: The Loss Curve depicted below illustrates a consistent decrease in average loss as the number of iterations increased, indicating effective model training. The convergence of loss values further supports the model's ability to avoid overfitting and maintain generalizability across unseen data **in Figure 3**.

5.3 PARAMETER TUNING

To optimize the model's performance, various hyperparameters were tested:

- Dimensions (D): The range of dimensions included {32, 64, 128, 192, 256, 300}.
- Total Hidden Layers (L): The number of layers explored was between 4 and 8.
- Epochs (E): Epochs tested ranged from 50 to 250.
- Dropout Rate (R): Dropout rates considered were from 0.0 to 0.5 in increments of 0.1.

Each parameter was tuned independently, with the resulting configurations evaluated using AUC and AUPR as primary metrics. The tuning process highlighted those specific combinations, as shown in **Figure 2**, optimized the balance between model complexity and performance. This tuning enabled us to achieve a well-calibrated architecture suitable for the complexity of DDI predictions.

5.4 SENTIMENTAL ANALYSIS ENSEMBLE MODEL

To enhance the accuracy of adverse reaction prediction, sentiment analysis was incorporated using a stacking ensemble model, which combines the strengths of **Logistic Regression** and **XGBoost (XGBClassifier)**. The ensemble model's architecture is as follows:

- **Base Models**: Logistic Regression and XGBClassifier were used as primary classifiers, each independently learning patterns within the transformed sentiment data.
- **Meta-Model**: Logistic Regression served as the meta-classifier, aggregating the predictions from the base models to make a final decision on classification.

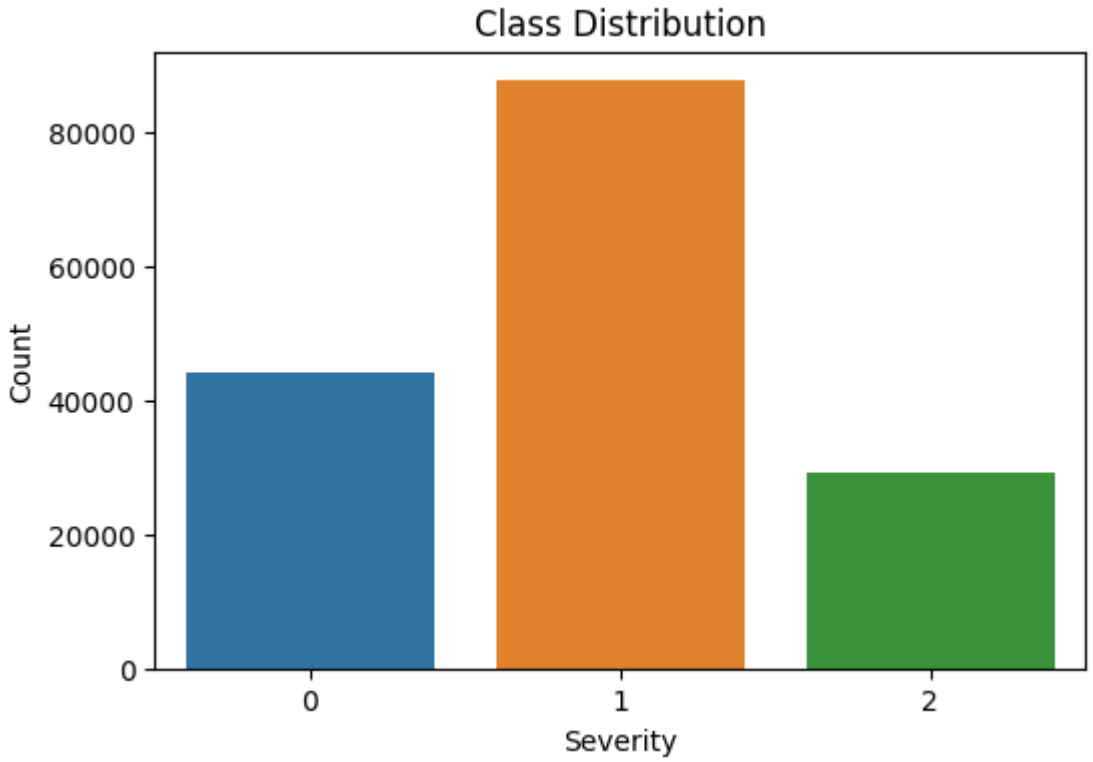

Figure 4: Class Distribution

Figure 4 shows the class distribution for the sentiment analysis dataset, providing insight into the balance of each severity level (e.g., Mild, Moderate, Severe) within the patient-reported adverse reaction data. This distribution is critical for understanding how the model may perform across various severity levels and highlights the challenges associated with class imbalance.

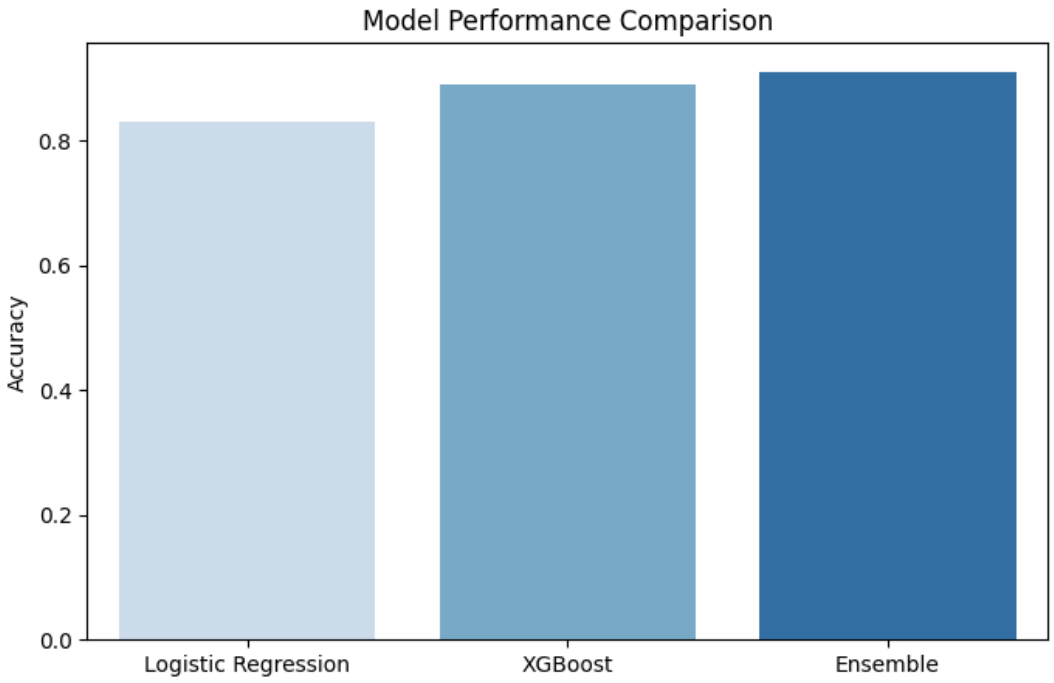

Figure 5: Model Performance Comparison

Figure 5 compares the individual performances of Logistic Regression, XGBClassifier, and their ensemble combination in terms of F1-score across each class. The stacking ensemble outperformed both base models independently, achieving an **overall accuracy of 0.9110**. This improvement underscores the ensemble model's ability to leverage the complementary strengths of Logistic Regression's interpretability and XGBClassifier's power in handling non-linear relationships, resulting in enhanced accuracy in capturing the nuances of adverse reactions.

By integrating this sophisticated sentiment analysis layer, the model's overall capacity to predict adverse outcomes was strengthened, offering a more nuanced understanding of drug-drug interaction (DDI) risks through the lens of real-world patient experiences. This approach not only supports precise predictions but also enriches the interpretability and clinical relevance of the DDI analysis.

5.5 THE RESULT OF COMBINATION OF TWO MODELS

We conducted further experiments to explore the impact of combining drug datasets with sentiment analysis. Each combination was evaluated using the same 5-fold cross validation setup. The AUPR and AUC values for this combination were 0.983 and 0.981, respectively, highlighting that including diverse drug information in combination to sentiment analysis data of adverse reactions helps improve predictive accuracy.

5.6 COMPARISION BETWEEN EXISTING MODELS

We compared our GNN-Sentiment Analysis hybrid model with several state-of-the-art methods, including DANN-DDI, CNN-DDI, and NDLM. As detailed in **Figure 6**, our model significantly outperformed existing models in predicting DDIs. For example, while the DANN-DDI model achieved an AUPR of 0.9709 and an AUC of 0.9763, our hybrid model achieved an AUPR of 0.983 and an AUC of 0.981. This improvement demonstrates that integrating both structural drug interaction data with sentiment-based insights can yield better predictive power.

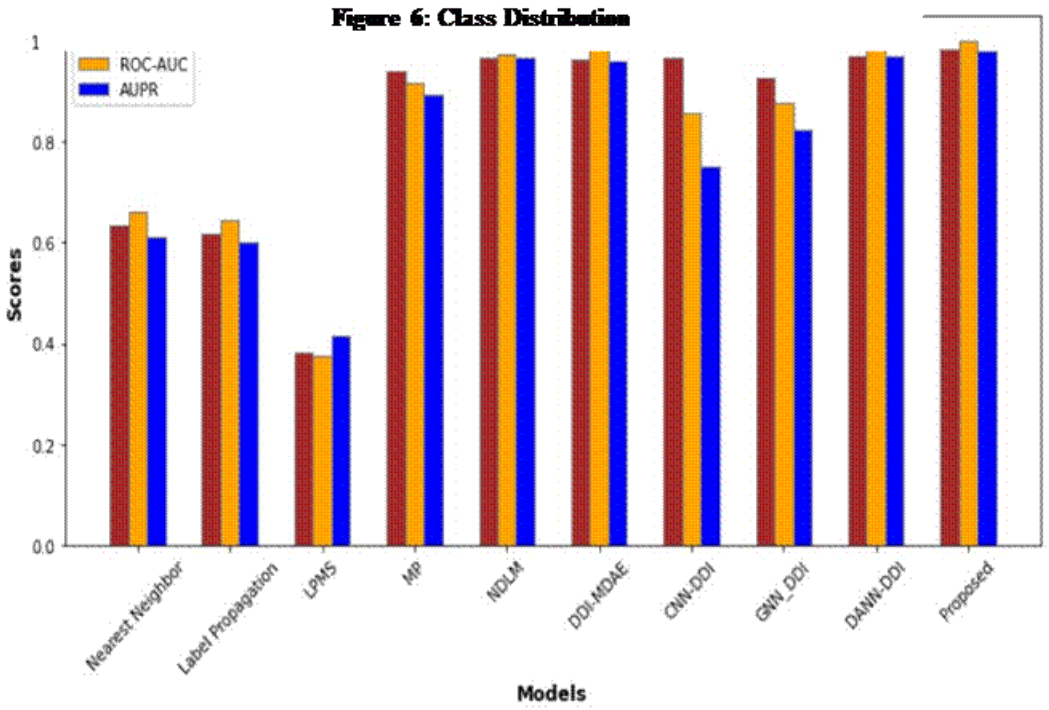

Figure 6: Class Comparison between models

5.7 PROPOSED MODEL FINAL METRICS

The final model, which combines GNN for DDI prediction with sentiment analysis for ADRs, achieved the following performance metrics: (**Figure 7)**

- **AUPR**: 0.983
- **AUC**: 0.981
- **Accuracy**: 0.997
- **Precision**: 0.998
- **Recall**: 0.962
- **F1-Score**: 0.980

These results confirm that our hybrid model not only excels in predicting drug-drug interactions but also in identifying adverse reactions, making it highly valuable for clinical applications.

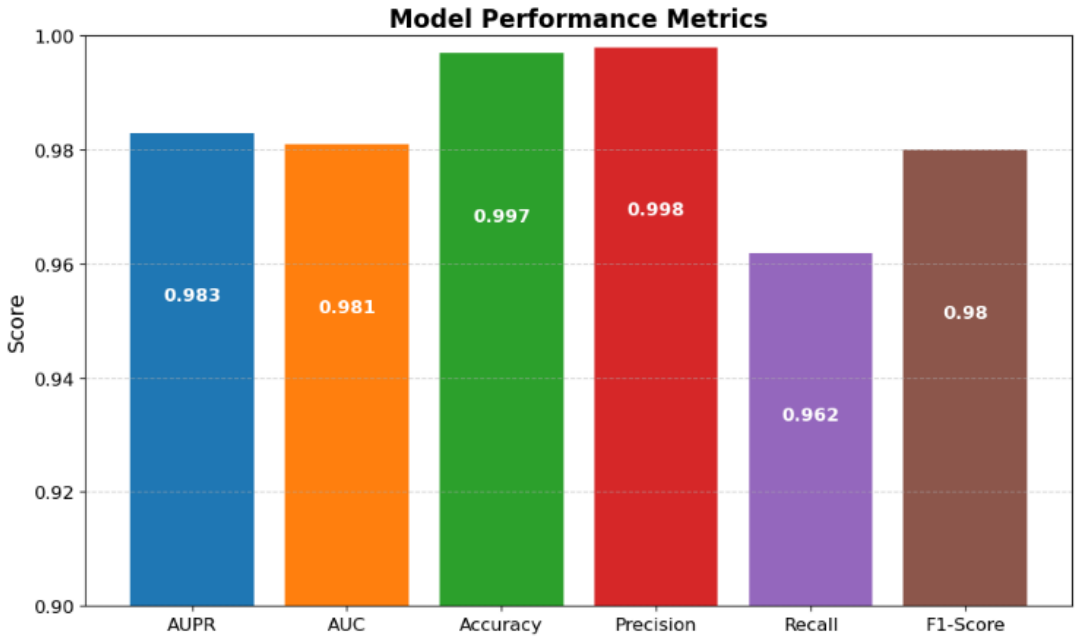

Figure 7: Final Model Metrics

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

In this study, we developed a novel methodology for predicting drug-drug interactions (DDIs) by integrating the predictive power of Graph Neural Networks (GNNs) with sentiment analysis to evaluate potential adverse drug reactions (ADRs). Traditional approaches often rely solely on molecular data to predict DDIs, which can overlook real-world patient experiences and associated risks. By incorporating patient-reported feedback through sentiment analysis, our model provides a more comprehensive understanding of DDIs, capturing both molecular interaction data and human-reported outcomes.

Our extensive evaluation, involving multiple datasets and rigorous hyperparameter tuning, confirms that this combined approach significantly outperforms existing DDI prediction models. The final model achieved outstanding performance metrics: an AUPR of 0.983, AUC of 0.981, accuracy of 0.997, precision of 0.998, recall of 0.962, and an F1 score of 0.980. These high metrics underscore the model's strength in identifying complex drug interactions with precision and in accurately assessing the severity of ADRs reported by patients. The integration of both GNN and sentiment analysis enhances the model's predictive depth, providing a robust tool for drug safety evaluation that holds substantial value for clinical decision-making.

Our findings emphasize the importance of using a range of data sources to improve the reliability of DDI predictions. By incorporating clinical reports alongside sentiment-based insights from patient feedback, we were able to enhance the model's contextual understanding of ADRs, a crucial aspect for addressing patient safety in

complex medical settings. The addition of an attention mechanism further improves model performance by focusing on key features within drug interactions, allowing the model to better prioritize and weigh critical interactions in its predictions. Ultimately, this approach provides healthcare professionals with an evidence-based, comprehensive tool for making informed decisions, thus reducing the risk of adverse effects associated with polypharmacy.

6.2 FUTURE ENCHANCEMENTS

1. **Targeted Disease-Specific Models**

   Future work could explore the development of models tailored specifically to certain diseases, such as cardiovascular conditions, diabetes, or cancer. Disease-focused models would concentrate on drug combinations and interactions most relevant to specific health conditions, enabling a more specialized approach to DDI prediction. By refining our model to address disease-specific patterns, we can enhance clinical relevance, leading to more accurate predictions and ultimately improving treatment outcomes for patients with complex and condition-specific medication regimens.

2. **Fine-Tuning for Specific Drug Classes**

   Certain drug classes, such as anticoagulants, antibiotics, and chemotherapeutic agents, carry higher risks of interactions and adverse effects. Fine-tuning the model to focus on these classes could improve the detection of unique interaction patterns that may not be as evident in a generalized dataset. Through specialized adjustments, such as reweighting interaction severity and incorporating domain-specific pharmacodynamics data, this approach could significantly enhance the model's accuracy and reliability for high-risk drugs, providing critical support for clinical decision-making in complex cases.

3. **Enhanced Sentiment Analysis Using Large Language Models (LLMs)**

   Advanced sentiment analysis methods, including large language models (LLMs) like GPT or BERT, could be integrated to capture deeper insights into patient-reported adverse reactions. Using a wider array of patient-centred datasets, including clinical trial reports, online reviews, and healthcare forums, these LLMs would facilitate a more nuanced understanding of adverse reactions. This approach would not only capture the presence of adverse interactions but also assess their severity and context based on patient experiences, providing richer data for model training. Enhanced sentiment analysis could improve the predictive model's capability to assess the real-world impact of drug combinations.

4. **Continuous Learning Mechanisms**

   To ensure that the model remains up to date with the latest drug interaction data and adapts to emerging patterns, incorporating continuous learning mechanisms is essential. By enabling the model to retrain periodically as new clinical and patient-reported data become available, this adaptive approach would allow the model to adjust predictions and recommendations based on the latest information. Continuous learning is especially valuable in a clinical context, where new drug formulations, approvals, and patient outcomes can rapidly shift the landscape of drug safety and efficacy, ensuring that the model maintains its relevance and accuracy in real-world applications.

REFERENCES

1. Jesus de la Fuente, Uxia Veleiro. (2024). "GENNIUS: An Ultrafast Drug-Target Interaction Inference Method Based on Graph Neural Networks." *Bioinformatics*. Dataset Source: DrugBank, BioSnap, BindingDB. Focus: Improves DTI prediction tasks.
2. Yue Yu, Kexin Huang, Chao Zhang. (2021). "SumGNN: Multi-Typed Drug Interaction Prediction via Efficient Knowledge Graph Summarization." *Bioinformatics*. Dataset Source: Bitbucket. Focus: Multi-typed DDI prediction.
3. B. Lokeswara Nayak, N. Lakshmi Tulsi. (2022). "Drug Recommendation System Based on Sentiment Analysis of Drug Reviews Using Machine Learning." *Journal of Engineering Sciences*. Dataset Source: Drugs.com. Focus: Sentiment analysis for drug recommendations.
4. Satvik Garg. (2022). "Drug Recommendation System Based on Sentiment Analysis of Drug Reviews Using Machine Learning." *IEEE Access*. Dataset Source: Drugs.com. Focus: Personalized drug recommendations.
5. Priyanka V.G, Pushpalatha G. (2022). "Drug Recommendation System Using Machine Learning." *Shanlax International Journal of Arts, Science & Humanities*. Dataset Source: OSMI Mental Health in Tech Survey Dataset from Kaggle. Focus: Personalized drug recommendations.
6. Yash Ritesh Tanna, Abhinav Maindre, Vaibhav Avinash Parmar. (2022). "Drug Recommendation System." *International Journal of Creative Research Thoughts*. Dataset Source: UCI ML Repository. Focus: Enhances patient care via personalized drug recommendations.
7. GV Lavanya, Praveen KS. (2022). "Drug Recommender System Using Machine Learning for Sentiment Analysis." *International Research Journal of Modernization in Engineering, Technology and Science*. Dataset Source: UCI ML Repository. Focus: Trustworthy and personalized medicine recommendations.

8. Md. Shah Amran. (2021). "Adverse Drug Reactions and Pharmacovigilance." *IntechOpen*. Dataset Source: PIDM (Programme for International Drug Monitoring). Focus: ADRs and pharmacovigilance insights.
9. Pallavi Pradhan, Pelumi Samuel Akinola. (2023). "Causality Assessment of Adverse Drug Reaction: A Narrative Review." *Journal of Applied Biomedicine*. Dataset Source: MEDLINE Database. Focus: Causality assessment of ADRs.
10. Gurpreet S Jutley, Mark Pucci. (2023). "Adverse Drug Reactions and Interactions." *Clinical Pharmacology*. Dataset Source: Yellow Card Scheme. Focus: Early detection and reporting of ADRs.

APPENDIX A

CODING

1 GNN MODEL

```
import random
import multiprocessing
from tqdm import tqdm
import argparse
from collections import defaultdict
import numpy as np
from six import iteritems
from sklearn.metrics import (auc, f1_score, precision_recall_curve, roc_auc_score)
import math
import os
import sys
import time
import pandas as pd
import csv
import tensorflow.compat.v1 as tf
from pandas import DataFrame
import numpy as np
import pandas as pd
import csv

tf.disable_v2_behavior()
```

 # In[2]:

```
def walk(args):
    walk_length, start, schema = args
    # Simulate a random walk starting from start node.
    rand = random.Random()

    if schema:
        schema_items = schema.split('-')
        assert schema_items[0] == schema_items[-1]

    walk = [start]
    while len(walk) < walk_length:
        cur = walk[-1]
        candidates = []
        for node in G[cur]:
                if schema == '' or node_type[node] == schema_items[len(walk) %
(len(schema_items) - 1)]:
                candidates.append(node)
        if candidates:
            walk.append(rand.choice(candidates))
        else:
            break
    return [str(node) for node in walk]

def initializer(init_G, init_node_type):
    global G
```

```
    G = init_G
    global node_type
    node_type = init_node_type

class RWGraph():
    def __init__(self, nx_G, node_type_arr=None, num_workers=16):
        self.G = nx_G
        self.node_type = node_type_arr
        self.num_workers = num_workers

    def node_list(self, nodes, num_walks):
        for loop in range(num_walks):
            for node in nodes:
                yield node

    def simulate_walks(self, num_walks, walk_length, schema=None):
        all_walks = []
        nodes = list(self.G.keys())
        random.shuffle(nodes)

        if schema is None:
                with multiprocessing.Pool(self.num_workers, initializer=initializer,
initargs=(self.G, self.node_type)) as pool:
                all_walks = list(pool.imap(walk, ((walk_length, node, '') for node in
tqdm(self.node_list(nodes, num_walks))), chunksize=256))
        else:
            schema_list = schema.split(',')
```

```
            for schema_iter in schema_list:
                with multiprocessing.Pool(self.num_workers, initializer=initializer, initargs=(self.G, self.node_type)) as pool:
                    walks = list(pool.imap(walk, ((walk_length, node, schema_iter) for node in tqdm(self.node_list(nodes, num_walks)) if schema_iter.split('-')[0] == self.node_type[node]), chunksize=512))
                all_walks.extend(walks)

        return all_walks

# In[3]:

class Vocab(object):

    def __init__(self, count, index):
        self.count = count
        self.index = index

def parse_args():
    parser = argparse.ArgumentParser()

    parser.add_argument('--input', type=str, default='',
                        help='Input dataset path')

    parser.add_argument('--features', type=str, default=None,
                        help='Input node features')
```

```
parser.add_argument('--walk-file', type=str, default=None,
                    help='Input random walks')

parser.add_argument('--epoch', type=int, default=1,
                    help='Number of epoch. Default is 100.')

parser.add_argument('--batch-size', type=int, default=8,
                    help='Number of batch_size. Default is 64.')

parser.add_argument('--eval-type', type=str, default='all',
                    help='The edge type(s) for evaluation.')

parser.add_argument('--schema', type=str, default=None,
                    help='The metapath schema (e.g., U-I-U,I-U-I).')

parser.add_argument('--dimensions', type=int, default=16,
                    help='Number of dimensions. Default is 200.')

parser.add_argument('--edge-dim', type=int, default=4,
                    help='Number of edge embedding dimensions. Default is 10.')

parser.add_argument('--att-dim', type=int, default=4,
                    help='Number of attention dimensions. Default is 20.')

parser.add_argument('--walk-length', type=int, default=5,
                    help='Length of walk per source. Default is 10.')
```

```
    parser.add_argument('--num-walks', type=int, default=5,
                        help='Number of walks per source. Default is 20.')

    parser.add_argument('--window-size', type=int, default=5,
                        help='Context size for optimization. Default is 5.')

    parser.add_argument('--negative-samples', type=int, default=3,
                        help='Negative samples for optimization. Default is 5.')

    parser.add_argument('--neighbor-samples', type=int, default=1,
                        help='Neighbor samples for aggregation. Default is 10.')

    parser.add_argument('--patience', type=int, default=5,
                        help='Early stopping patience. Default is 5.')

    parser.add_argument('--num-workers', type=int, default=1,
                        help='Number of workers for generating random walks. Default is
16.')
    parser.add_argument("-f", "--file", required=False)

    return parser.parse_args()

def get_G_from_edges(edges):
    edge_dict = defaultdict(set)
    for edge in edges:
        u, v = str(edge[0]), str(edge[1])
        edge_dict[u].add(v)
```

```
        edge_dict[v].add(u)
    return edge_dict

def load_training_data(f_name):
    print('We are loading data from:', f_name)
    edge_data_by_type = dict()
    all_nodes = list()
    with open(f_name, 'r') as f:
        for line in f:
            words = line[:-1].split(' ')
            if words[0] not in edge_data_by_type:
                edge_data_by_type[words[0]] = list()
            x, y = words[1], words[2]
            edge_data_by_type[words[0]].append((x, y))
            all_nodes.append(x)
            all_nodes.append(y)
    all_nodes = list(set(all_nodes))
    print('Total training nodes: ' + str(len(all_nodes)))
    return edge_data_by_type

def load_testing_data(f_name):
    print('We are loading data from:', f_name)
    true_edge_data_by_type = dict()
    false_edge_data_by_type = dict()
    all_nodes = list()
    with open(f_name, 'r') as f:
        for line in f:
```

```
            words = line[:-1].split(' ')
            x, y = words[1], words[2]
            if int(words[3]) == 1:
                if words[0] not in true_edge_data_by_type:
                    true_edge_data_by_type[words[0]] = list()
                true_edge_data_by_type[words[0]].append((x, y))
            else:
                if words[0] not in false_edge_data_by_type:
                    false_edge_data_by_type[words[0]] = list()
                false_edge_data_by_type[words[0]].append((x, y))
            all_nodes.append(x)
            all_nodes.append(y)
    all_nodes = list(set(all_nodes))
    return true_edge_data_by_type, false_edge_data_by_type


def load_node_type(f_name):
    print('We are loading node type from:', f_name)
    node_type = {}
    with open(f_name, 'r') as f:
        for line in f:
            items = line.strip().split()
            node_type[items[0]] = items[1]
    return node_type


def load_feature_data(f_name):
    feature_dic = {}
    with open(f_name, 'r') as f:
```

```
        first = True
        for line in f:
            if first:
                first = False
                continue
            items = line.strip().split()
            feature_dic[items[0]] = items[1:]
    return feature_dic


def generate_walks(network_data, num_walks, walk_length, schema, file_name, num_workers):
    if schema is not None:
                                node_type = load_node_type('/kaggle/input/ddi-dataset/DDI/data5/node_type.txt')
    else:
        node_type = None

    all_walks = []
    for layer_id, layer_name in enumerate(network_data):
        tmp_data = network_data[layer_name]
        # start to do the random walk on a layer

            layer_walker = RWGraph(get_G_from_edges(tmp_data), node_type, num_workers)
        print('Generating random walks for layer', layer_id)
            layer_walks = layer_walker.simulate_walks(num_walks, walk_length, schema=schema)
```

```
        all_walks.append(layer_walks)

    print('Finish generating the walks')

    return all_walks

def generate_pairs(all_walks, vocab, window_size, num_workers):
    pairs = []
    skip_window = window_size // 2
    for layer_id, walks in enumerate(all_walks):
        print('Generating training pairs for layer', layer_id)
        for walk in tqdm(walks):
            for i in range(len(walk)):
                for j in range(1, skip_window + 1):
                    if i - j >= 0:
                            pairs.append((vocab[walk[i]].index, vocab[walk[i - j]].index,
layer_id))
                    if i + j < len(walk):
                            pairs.append((vocab[walk[i]].index, vocab[walk[i + j]].index,
layer_id))
    return pairs

def generate_vocab(all_walks):
    index2word = []
    raw_vocab = defaultdict(int)
```

```
    for layer_id, walks in enumerate(all_walks):
        print('Counting vocab for layer', layer_id)
        for walk in tqdm(walks):
            for word in walk:
                raw_vocab[word] += 1

    vocab = {}
    for word, v in iteritems(raw_vocab):
        vocab[word] = Vocab(count=v, index=len(index2word))
        index2word.append(word)

    index2word.sort(key=lambda word: vocab[word].count, reverse=True)
    for i, word in enumerate(index2word):
        vocab[word].index = i

    return vocab, index2word

def load_walks(walk_file):
    print('Loading walks')
    all_walks = []
    with open(walk_file, 'r') as f:
        for line in f:
            content = line.strip().split()
            layer_id = int(content[0])
            if layer_id >= len(all_walks):
                all_walks.append([])
            all_walks[layer_id].append(content[1:])
```

```
    return all_walks

def save_walks(walk_file, all_walks):
    with open(walk_file, 'w') as f:
        for layer_id, walks in enumerate(all_walks):
            print('Saving walks for layer', layer_id)
            for walk in tqdm(walks):
                f.write(' '.join([str(layer_id)] + [str(x) for x in walk]) + '\n')

def generate(network_data, num_walks, walk_length, schema, file_name, window_size, num_workers, walk_file):
    if walk_file is not None:
        all_walks = load_walks(walk_file)
    else:
       all_walks = generate_walks(network_data, num_walks, walk_length, schema, file_name, num_workers)
        save_walks('/walks.txt', all_walks)
    vocab, index2word = generate_vocab(all_walks)
    train_pairs = generate_pairs(all_walks, vocab, window_size, num_workers)

    return vocab, index2word, train_pairs

def generate_neighbors(network_data, vocab, num_nodes, edge_types, neighbor_samples):
    edge_type_count = len(edge_types)
    neighbors = [[[] for __ in range(edge_type_count)] for _ in range(num_nodes)]
    for r in range(edge_type_count):
```

```
        print('Generating neighbors for layer', r)
        g = network_data[edge_types[r]]
        for (x, y) in tqdm(g):
            ix = vocab[x].index
            iy = vocab[y].index
            neighbors[ix][r].append(iy)
            neighbors[iy][r].append(ix)
        for i in range(num_nodes):
            if len(neighbors[i][r]) == 0:
                neighbors[i][r] = [i] * neighbor_samples
            elif len(neighbors[i][r]) < neighbor_samples:
                            neighbors[i][r].extend(list(np.random.choice(neighbors[i][r],
size=neighbor_samples-len(neighbors[i][r]))))
            elif len(neighbors[i][r]) > neighbor_samples:
                              neighbors[i][r]   =   list(np.random.choice(neighbors[i][r],
size=neighbor_samples))
    return neighbors


def get_score(local_model, node1, node2):
    try:
        vector1 = local_model[node1]
        vector2 = local_model[node2]
                return   np.dot(vector1,   vector2)   /   (np.linalg.norm(vector1)   *
np.linalg.norm(vector2))
    except Exception as e:
        pass
```

```
def evaluate(model, true_edges, false_edges):
    true_list = list()
    prediction_list = list()
    true_num = 0
    for edge in true_edges:
        tmp_score = get_score(model, str(edge[0]), str(edge[1]))
        if tmp_score is not None:
            true_list.append(1)
            prediction_list.append(tmp_score)
            true_num += 1

    for edge in false_edges:
        tmp_score = get_score(model, str(edge[0]), str(edge[1]))
        if tmp_score is not None:
            true_list.append(0)
            prediction_list.append(tmp_score)

    sorted_pred = prediction_list[:]
    sorted_pred.sort()
    threshold = sorted_pred[-true_num]

    y_pred = np.zeros(len(prediction_list), dtype=np.int32)
    for i in range(len(prediction_list)):
        if prediction_list[i] >= threshold:
            y_pred[i] = 1

    y_true = np.array(true_list)
```

```
    y_scores = np.array(prediction_list)
    ps, rs, _ = precision_recall_curve(y_true, y_scores)
    return roc_auc_score(y_true, y_scores), f1_score(y_true, y_pred), auc(rs, ps)

# In[4]:

def get_batches(pairs, neighbors, batch_size):
    n_batches = (len(pairs) + (batch_size - 1)) // batch_size

    for idx in range(n_batches):
        x, y, t, neigh = [], [], [], []
        for i in range(batch_size):
            index = idx * batch_size + i
            if index >= len(pairs):
                break
            x.append(pairs[index][0])
            y.append(pairs[index][1])
            t.append(pairs[index][2])
            neigh.append(neighbors[pairs[index][0]])
                    yield (np.array(x).astype(np.int32), np.array(y).reshape(-1, 1).astype(np.int32), np.array(t).astype(np.int32), np.array(neigh).astype(np.int32))

def train_model(network_data, feature_dic, log_name, f_num, file_name):
    vocab, index2word, train_pairs = generate(network_data, args.num_walks, args.walk_length, args.schema, file_name, args.window_size, args.num_workers, args.walk_file)
```

```
    edge_types = list(network_data.keys())

    num_nodes = len(index2word)
    edge_type_count = len(edge_types)
    epochs = args.epoch
    batch_size = args.batch_size
    embedding_size = args.dimensions # Dimension of the embedding vector.
    embedding_u_size = args.edge_dim
    u_num = edge_type_count
     num_sampled = args.negative_samples # Number of negative examples to sample.
    dim_a = args.att_dim
    att_head = 1
    neighbor_samples = args.neighbor_samples

    neighbors = generate_neighbors(network_data, vocab, num_nodes, edge_types, neighbor_samples)

    graph = tf.Graph()

    if feature_dic is not None:
        feature_dim = len(list(feature_dic.values())[0])
        print('feature dimension: ' + str(feature_dim))
        features = np.zeros((num_nodes, feature_dim), dtype=np.float32)
        for key, value in feature_dic.items():
            if key in vocab:
```

```
            features[vocab[key].index, :] = np.array(value)

    with graph.as_default():
        global_step = tf.Variable(0, name='global_step', trainable=False)

        if feature_dic is not None:
                    node_features = tf.Variable(features, name='node_features',
trainable=False)
                    feature_weights = tf.Variable(tf.truncated_normal([feature_dim,
embedding_size], stddev=1.0))

                    embed_trans = tf.Variable(tf.truncated_normal([feature_dim,
embedding_size], stddev=1.0 / math.sqrt(embedding_size)))
                u_embed_trans = tf.Variable(tf.truncated_normal([edge_type_count,
feature_dim, embedding_u_size], stddev=1.0 / math.sqrt(embedding_size)))
        else:
                    node_embeddings = tf.Variable(tf.random_uniform([num_nodes,
embedding_size], -1.0, 1.0))
                node_type_embeddings = tf.Variable(tf.random_uniform([num_nodes,
u_num, embedding_u_size], -1.0, 1.0))

                trans_weights = tf.Variable(tf.truncated_normal([edge_type_count,
embedding_u_size, embedding_size // att_head], stddev=1.0 /
math.sqrt(embedding_size)))
            trans_weights_s1 = tf.Variable(tf.truncated_normal([edge_type_count,
embedding_u_size, dim_a], stddev=1.0 / math.sqrt(embedding_size)))
```

```
        trans_weights_s2 = tf.Variable(tf.truncated_normal([edge_type_count, dim_a, att_head], stddev=1.0 / math.sqrt(embedding_size)))
                nce_weights = tf.Variable(tf.truncated_normal([num_nodes, embedding_size], stddev=1.0 / math.sqrt(embedding_size)))
    nce_biases = tf.Variable(tf.zeros([num_nodes]))

    # Input data
    train_inputs = tf.placeholder(tf.int32, shape=[None])
    train_labels = tf.placeholder(tf.int32, shape=[None, 1])
    train_types = tf.placeholder(tf.int32, shape=[None])
        node_neigh = tf.placeholder(tf.int32, shape=[None, edge_type_count, neighbor_samples])

    # Look up embeddings for nodes
    if feature_dic is not None:
       node_embed = tf.nn.embedding_lookup(node_features, train_inputs)
       node_embed = tf.matmul(node_embed, embed_trans)
    else:
       node_embed = tf.nn.embedding_lookup(node_embeddings, train_inputs)

    if feature_dic is not None:
            node_embed_neighbors = tf.nn.embedding_lookup(node_features, node_neigh)
                                                    node_embed_tmp = tf.concat([tf.matmul(tf.reshape(tf.slice(node_embed_neighbors, [0, i, 0, 0], [-1, 1, -1, -1]), [-1, feature_dim]), tf.reshape(tf.slice(u_embed_trans, [i, 0, 0], [1, -1, -1]), [feature_dim, embedding_u_size])) for i in range(edge_type_count)], axis=0)
```

```
                node_type_embed = tf.transpose(tf.reduce_mean(tf.reshape(node_embed_tmp, [edge_type_count, -1, neighbor_samples, embedding_u_size]), axis=2), perm=[1,0,2])
        else:
                node_embed_neighbors = tf.nn.embedding_lookup(node_type_embeddings, node_neigh)
            node_embed_tmp = tf.concat([tf.reshape(tf.slice(node_embed_neighbors, [0, i, 0, i, 0], [-1, 1, -1, 1, -1]), [1, -1, neighbor_samples, embedding_u_size]) for i in range(edge_type_count)], axis=0)
                node_type_embed = tf.transpose(tf.reduce_mean(node_embed_tmp, axis=2), perm=[1,0,2])

        trans_w = tf.nn.embedding_lookup(trans_weights, train_types)
        trans_w_s1 = tf.nn.embedding_lookup(trans_weights_s1, train_types)
        trans_w_s2 = tf.nn.embedding_lookup(trans_weights_s2, train_types)

                attention = tf.reshape(tf.nn.softmax(tf.reshape(tf.matmul(tf.tanh(tf.matmul(node_type_embed, trans_w_s1)), trans_w_s2), [-1, u_num])), [-1, att_head, u_num])
        node_type_embed = tf.matmul(attention, node_type_embed)
            node_embed = node_embed + tf.reshape(tf.matmul(node_type_embed, trans_w), [-1, embedding_size])

        if feature_dic is not None:
            node_feat = tf.nn.embedding_lookup(node_features, train_inputs)
            node_embed = node_embed + tf.matmul(node_feat, feature_weights)
```

```
        last_node_embed = tf.nn.l2_normalize(node_embed, axis=1)

        loss = tf.reduce_mean(
           tf.nn.nce_loss(
              weights=nce_weights,
              biases=nce_biases,
              labels=train_labels,
              inputs=last_node_embed,
              num_sampled=num_sampled,
              num_classes=num_nodes))
        plot_loss = tf.summary.scalar("loss", loss)

        # Optimizer.
                        optimizer = tf.train.AdamOptimizer().minimize(loss,
global_step=global_step)

        # Add ops to save and restore all the variables.
        # saver = tf.train.Saver(max_to_keep=20)

        merged = tf.summary.merge_all(key=tf.GraphKeys.SUMMARIES)

        # Initializing the variables
        init = tf.global_variables_initializer()

    # Launch the graph
    print("Optimizing")
```

```
    with tf.Session(graph=graph) as sess:
        writer = tf.summary.FileWriter("./runs/" + log_name, sess.graph) # tensorboard --logdir=./runs
        sess.run(init)

        print('Training')
        g_iter = 0
        best_score = 0
        test_score = (0.0, 0.0, 0.0)
        patience = 0
        for epoch in range(epochs):
            random.shuffle(train_pairs)
            batches = get_batches(train_pairs, neighbors, batch_size)

            data_iter = tqdm(batches,
                             desc="epoch %d" % (epoch),
                             total=(len(train_pairs) + (batch_size - 1)) // batch_size,
                             bar_format="{l_bar}{r_bar}")
            avg_loss = 0.0

            for i, data in enumerate(data_iter):
                feed_dict = {train_inputs: data[0], train_labels: data[1], train_types: data[2], node_neigh: data[3]}
                _, loss_value, summary_str = sess.run([optimizer, loss, merged], feed_dict)
                writer.add_summary(summary_str, g_iter)
```

```
            g_iter += 1

            avg_loss += loss_value

            if i % 5000 == 0:
                post_fix = {
                    "epoch": epoch,
                    "iter": i,
                    "avg_loss": avg_loss / (i + 1),
                    "loss": loss_value
                }
                data_iter.write(str(post_fix))

                        final_model = dict(zip(edge_types, [dict() for _ in range(edge_type_count)]))
        for i in range(edge_type_count):
            for j in range(num_nodes):
                                    final_model[edge_types[i]][index2word[j]] = np.array(sess.run(last_node_embed, {train_inputs: [j], train_types: [i], node_neigh: [neighbors[j]]})[0])

        valid_aucs, valid_f1s, valid_prs = [], [], []
        test_aucs, test_f1s, test_prs = [], [], []
        for i in range(edge_type_count):
            if args.eval_type == 'all' or edge_types[i] in args.eval_type.split(','):
```

```
                tmp_auc, tmp_f1, tmp_pr = evaluate(final_model[edge_types[i]],
valid_true_data_by_edge[edge_types[i]],
valid_false_data_by_edge[edge_types[i]])
                valid_aucs.append(tmp_auc)
                valid_f1s.append(tmp_f1)
                valid_prs.append(tmp_pr)

                tmp_auc, tmp_f1, tmp_pr = evaluate(final_model[edge_types[i]],
testing_true_data_by_edge[edge_types[i]],
testing_false_data_by_edge[edge_types[i]])
                test_aucs.append(tmp_auc)
                test_f1s.append(tmp_f1)
                test_prs.append(tmp_pr)
        print('valid auc:', np.mean(valid_aucs))
        print('valid pr:', np.mean(valid_prs))
        print('valid f1:', np.mean(valid_f1s))

        average_auc = np.mean(test_aucs)
        average_f1 = np.mean(test_f1s)
        average_pr = np.mean(test_prs)

        cur_score = np.mean(valid_aucs)
        if cur_score > best_score:
            best_score = cur_score
            test_score = (average_auc, average_f1, average_pr)
            patience = 0
        else:
```

```
            patience += 1
            if patience > args.patience:
                print('Early Stopping')
                break
        final_modelss=[]
        for i in range(edge_type_count):
         for j in range(num_nodes):
            final_modelss.append([edge_types[i],index2word[j],final_model[edge_types[i]][index2word[j]]])
        df = pd.DataFrame((final_modelss))
             df.to_csv('/final_modelss'+str(f_num)+'_d_'+str(embedding_size)+'.csv', header=None, index=None) #, sep=' '
        # df = pd.DataFrame((final_model))
             #  df.to_csv(file_name+'/final_model'+str(f_num)+'.csv',  header=None, index=None) #, sep=' '
        # with open('GFG.csv', 'w') as f:
        #   write = csv.writer(f)
        #   for i in range(edge_type_count):
        #     write.writerows(final_model[edge_types[i]])
        # df = pd.DataFrame((index2word))
           #  df.to_csv('/content/drive/MyDrive/DDI/index2word.csv',  header=None, index=None) #, sep=' '
    return test_score

if __name__ == "__main__":
    args = parse_args()
    file_name = args.input
```

```
    print(args)
    if args.features is not None:
        feature_dic = load_feature_data(args.features)
        f_num = str(args.features[-5])
    else:
        feature_dic = None
        f_num = str("dr")
                    log_name = file_name.split('/')[-1] +
f'_evaltype_{args.eval_type}_b_{args.batch_size}_e_{args.epoch}'

          training_data_by_type = load_training_data( '/kaggle/input/ddi-
dataset/DDI/data5/train.txt')
                valid_true_data_by_edge, valid_false_data_by_edge =
load_testing_data('/kaggle/input/ddi-dataset/DDI/data5/valid.txt')
              testing_true_data_by_edge, testing_false_data_by_edge =
load_testing_data('/kaggle/input/ddi-dataset/DDI/data5/test.txt')

     average_auc, average_f1, average_pr = train_model(training_data_by_type,
feature_dic, log_name + '_' + time.strftime('%Y-%m-%d %H-%M-
%S',time.localtime(time.time())),f_num,file_name)

    print('Overall ROC-AUC:', average_auc)
    print('Overall PR-AUC', average_pr)
    print('Overall F1:', average_f1)

# In[ ]:
```

```
import matplotlib.pyplot as plt
import numpy as np

# Loss data
iterations = [0, 5000, 10000, 15000, 20000, 25000, 30000, 35000]
avg_losses = [14.08, 9.57, 7.84, 6.76, 6.03, 5.49, 5.06, 4.71]

# Evaluation metrics
metrics = {
    'Valid AUC': 0.816,
    'Valid PR': 0.803,
    'Valid F1': 0.777,
    'Overall ROC-AUC': 0.782,
    'Overall PR-AUC': 0.773,
    'Overall F1': 0.729
}

# Plotting the Loss Curve
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(iterations, avg_losses, marker='o', color='b')
plt.title('Loss Curve')
plt.xlabel('Iterations')
plt.ylabel('Average Loss')
plt.ylim([0, 15])
plt.xticks(iterations)
```

```
plt.grid()

# Plotting Evaluation Metrics
plt.subplot(1, 2, 2)
names = list(metrics.keys())
values = list(metrics.values())

plt.barh(names, values, color='skyblue')
plt.title('Validation and Overall Metrics')
plt.xlim([0, 1])
plt.xlabel('Score')
plt.grid(axis='x')

plt.tight_layout()
plt.show()
```

2 SENTIMENTAL ANALYSIS ENSEMBLE MODEL

```
# Use class weighting to handle the imbalance

model = LogisticRegression(class_weight='balanced')

# Fit the model on the training data
```

```
model.fit(X_train_tfidf, y_train)


# Evaluate the model

y_pred = model.predict(X_test_tfidf)

from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))

# In[19]:

#ensemble with xgboost and random forest

# In[20]:

get_ipython().system('pip install xgboost')

# In[21]:

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report


# Train a Random Forest model with class weighting
```

```
rf_model = RandomForestClassifier(class_weight='balanced', random_state=42, n_estimators=100, max_depth=10)

rf_model.fit(X_train_tfidf, y_train)


# Evaluate the model on the test set

y_pred_rf = rf_model.predict(X_test_tfidf)

print(classification_report(y_test, y_pred_rf))

# In[22]:

import xgboost as xgb

from sklearn.metrics import classification_report


# Train an XGBoost model with class weighting

xgb_model = xgb.XGBClassifier(scale_pos_weight=3, random_state=42, n_estimators=100, max_depth=10)

xgb_model.fit(X_train_tfidf, y_train)
```

```
# Evaluate the model on the test set

y_pred_xgb = xgb_model.predict(X_test_tfidf)

print(classification_report(y_test, y_pred_xgb))

# In[24]:

#emsemble learning

# In[25]:

from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier

# Define the base models
log_reg = LogisticRegression()
xgb = XGBClassifier(eval_metric='mlogloss')

# Define the meta-model (a logistic regression here)
meta_model = LogisticRegression()

# Create StackingClassifier
stacking_clf = StackingClassifier(estimators=[
```

```
    ('log_reg', log_reg),
    ('xgb', xgb)
], final_estimator=meta_model)

# Fit the stacking model
stacking_clf.fit(X_train_tfidf, y_train)

# Evaluate the ensemble model
accuracy = stacking_clf.score(X_test_tfidf, y_test)
print(f"Stacked Model Accuracy: {accuracy:.4f}")

# In[26]:

import matplotlib.pyplot as plt
import seaborn as sns

# Assuming y_train contains the target variable (severity classes)
plt.figure(figsize=(6, 4))
sns.countplot(x=y_train)
plt.title('Class Distribution')
plt.xlabel('Severity')
plt.ylabel('Count')
plt.show()

# In[36]:

models = ['Logistic Regression', 'XGBoost', 'Ensemble']
```

```
accuracies = [0.83, 0.89, 0.911]

plt.figure(figsize=(8, 5))
sns.barplot(x=models, y=accuracies, palette='Blues')
plt.title('Model Performance Comparison')
plt.ylabel('Accuracy')
plt.show()

# In[29]:

from sklearn.metrics import precision_score, recall_score, f1_score

precision = precision_score(y_test, y_pred, average=None)
recall = recall_score(y_test, y_pred, average=None)
f1 = f1_score(y_test, y_pred, average=None)

# Create DataFrame for better visualization
metrics = pd.DataFrame({
    'Precision': precision,
    'Recall': recall,
    'F1-Score': f1
}, index=['None', 'Moderate', 'Severe'])

metrics.plot(kind='bar', figsize=(10, 6), colormap='coolwarm')
plt.title('Precision, Recall, and F1-Score Comparison')
plt.ylabel('Score')
plt.xlabel('Class')
```

```
plt.xticks(rotation=0)
plt.show()

# In[31]:

from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize

y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
y_pred_bin = label_binarize(y_pred, classes=[0, 1, 2])

fpr, tpr, _ = roc_curve(y_test_bin[:, 0], y_pred_bin[:, 0])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'Class 0 (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.title('ROC Curve for Class 0')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.show()

# In[32]:

from sklearn.model_selection import learning_curve
```

```
train_sizes, train_scores, test_scores = learning_curve(xgb_model, X_train_tfidf, y_train, cv=3)

plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_scores.mean(axis=1), label='Training score')
plt.plot(train_sizes, test_scores.mean(axis=1), label='Cross-validation score')
plt.title('Learning Curve for XGBoost')
plt.xlabel('Training Size')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

APPENDIX B

CONFERENCE SUBMISSION

ICICT 2025 <icict2025-0@easychair.org> Mon, Oct 21, 12:13 PM

to me ▾

Dear Abisek Kamthan R S

Paper ID : 215

Title : GNN – Enhanced Drug Interaction detection and adverse reaction analysis with sentiment insights

Greetings .. !!

Congratulations! On behalf of the Program Committee of ICICT 2025 - LONDON, I am happy to inform you that your above-mentioned paper has been ACCEPTED for oral presentation in ICICT 2025 and publication in Springer LNNS series subject to fulfillment of Guidelines by Springer.

An accepted paper will be published in the Springer proceedings (LNNS) only if the final version is accompanied by the payment information (i.e. transaction reference number) subject to quality check as per Springer Guidelines.

Kindly follow the below-mentioned guidelines (strictly), related to the preparation of the Final Manuscript, Copyright Transfer Form, Payment, and Final Submission. The procedure has been detailed as a five-step process (I)-(III):

The submission that has been made and yet to hear back from.

APPENDIX C

PLAGIARISM REPORT

Ds Bs

GNN

Quick Submit

Quick Submit

SRM Institute of Science & Technology

Document Details

Submission ID
trn:oid:::1:3044623678

Submission Date
Oct 17, 2024, 9:14 AM GMT+5:30

Download Date
Oct 17, 2024, 9:16 AM GMT+5:30

File Name
ection_and_adverse_reaction_analysis_with_sentiment_insights.pdf

File Size
428.4 KB

6 Pages

3,821 Words

23,054 Characters

turnitin Page 1 of 11 - Cover Page Submission ID trn:oid:::1:3044623678

turnitin Page 2 of 11 - Integrity Overview Submission ID trn:oid:::1:3044623678

11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

34 Not Cited or Quoted 10%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

3 Missing Citation 1%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

7% Internet sources

8% Publications

5% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.